Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Gestión de información de transporte público mediante SIRI

Autor:

Cáceres de la Calle, Eduardo

Tutor:

de la Fuente López, Eusebio

Departamento de Ingeniería de Sistemas y Automática

Valladolid, junio del 2018

Resumen

SIRI (*Service Interface for Real-time Information*) es un protocolo estandarizado de comunicación de datos relacionados con el transporte público, ratificado como tal en la norma europea UNE-EN 15531 1-5.

En este proyecto se expone el funcionamiento de SIRI y los múltiples datos que este protocolo nos permite intercambiar. Esta información comprende desde la planificación de viajes y horarios de los distintos medios de transporte público hasta la monitorización en tiempo real de los vehículos que prestan esos servicios, lo que permite tanto a los operadores y empresas de transporte como a los usuarios finales de estos sistemas (los pasajeros) estar informados en todo momento sobre esos viajes.

Por otra parte, en este documento también se desgrana el desarrollo de un servidor y un cliente capaces de comunicarse usando el estándar SIRI, implementados para una empresa del mundo de los Sistemas Inteligentes de Transporte (ITS).

Palabras Clave

SIRI, Transmodel, Transporte público, Estándar, Sistemas Inteligentes de Transporte

Abstract

SIRI (Service Interface for Real-time Information) is a standard communication protocol for exchanging public transport information, settled as European norm UNE-EN 15531 1-5.

This document describes SIRI standard, clarifying which kind of data are intended to be exchanged using this protocol. This information includes both timetable and trip planification of public transport operations and real time monitorization of those services, which allows public transport operators and final users (passengers) to be constantly informed about those trips.

Moreover, this document explains the development of a server and a client capable of communicating using SIRI standard, both of them implemented for a company focused in Intelligent Transport Systems (ITS).

Keywords

SIRI, Transmodel, Public transport, Standard, Intelligent Transport Systems (ITS).

Índice

Índice de figuras

Glosario

AFIMB: *L'Agence Française pour l'Information Multimodale et la Billettique*. Agencia encargada de promover la interoperabilidad y estandarización en las áreas de intercambio de información de transporte multimodal (es decir, la información que implica varios vehículos o medios de transporte) y venta inteligente de billetes. Está relacionada con el ministerio francés de Ecología, Desarrollo Sostenible y Energía.

API: *Application Programming Interface*. Interfaz pública a través de la cual comunicarse con un sistema software.

AVMS: *Automatic Vehicle Management System*. Ver CAD/AVL.

CAD/AVL: *Computer-aided dispatch & Automatic Vehicle Location*. Sistemas de ITS que permiten a un operador de transporte controlar sus flotas de vehículos.

CEN: Comité Europeo de Normalización. Organización privada encargada de proporcionar una estructura eficiente para el desarrollo, mantenimiento y distribución de especificaciones y sistemas estándares coherentes

GTFS: *Google/General Transit Feed Specification*. Estándar (de facto) creado por Google para transmitir información sobre horarios de transporte público y su información geográfica asociada.

gSOAP: *generic XML and SOAP*. Herramienta *open source* que facilita el serializado y des-serializado de datos entre documentos XML y tipos y estructuras de C o C++.

HTTP: *Hypertext Transfer Protocol*. Protocolo de transferencia de información que sirvió de base para la comunicación de datos de la *World Wide Web*.

HÜR: Compañía de transportes de Hovedstadsregionen (antigua región en torno a Copenhague), colaboró en el desarrollo de Transmodel.

IFOPT: *Identification of Fixed Objects in Public Transport*. Especificación técnica europea (CEN/TS 28701) basada en Transmodel y que engloba en un modelo de datos de referencia las entidades necesarias para el acceso al transporte público.

ITS: *Intelligent Transport Systems*. Conjunto de soluciones tecnológicas y telemáticas diseñadas para mejorar la operatividad y seguridad del transporte, además proveer de más y mejor información a los pasajeros sobre sus respectivos viajes.

ITxPT: *Information Technology for Public Transport*. Iniciativa internacional que busca la implantación de un estándar que logre que las diferentes máquinas instaladas en vehículos e instalaciones de transporte público sean totalmente intercambiables, pudiendo funcionar como *plug-and-play*.

FCS: *Fare Collection System*. Sistemas que se encargan de la gestión de toda la parte monética en torno a los Sistemas Inteligentes de Transporte.

FMS: *Fleet Management System*. Ver CAD/AVL.

GUID: *Global Unique Identifier*. Ver UUID.

NAPTAN: *National Public Transport Access Node*. Estándar británico que identifica todos los puntos de acceso al transporte público de Reino Unido (paradas de autobús y tranvía, estaciones de tren, aeropuertos, puntos de salida de ferris, etc.).

NeTEx: *NETwork Exchange / NEtwork and Timetable Exchange*. Especificación técnica europea (CEN/TS 16614) que define la implementación más completa de Transmodel existente en la actualidad.

NPTG: *National Public Transport Gazeteer*. Estándar británico que identifica todos los puntos relevantes de Reino Unido de cara al transporte público.

OBU: *On-Board Unit*. Dispositivo electrónico embebido en un vehículo. Centraliza las comunicaciones de éste con el exterior, tanto entrantes (recepción de instrucciones, parámetros de configuración, etc.) como salientes (envío de información).

PI(D)S: *Passenger Information (Display) System*. Información que se comunica a un usuario de transporte público relacionada con su viaje.

PTA: *Public Transport Authority*. Entidad encargada de regular el transporte público en una determinada zona. Puede encargarse ella misma de su gestión, o contratar parte o la totalidad de ella a uno o varios PTOs.

PTO: *Public Transport Operator*. Entidad que opera con vehículos de transporte público. No está necesariamente ligada a una PTA.

RATP: *La Régie Autonome des Transports Parisiens*. Empresa de transporte francesa de gran relevancia. Colaboró en el desarrollo de Transmodel.

RTIG: *The Real Time Information Group*. Organización de Reino Unido dedicada al sector ITS. Participó, entre otros, en el desarrollo de SIRI.

REST: *REpresentational State Transfer*. Interfaz de comunicación entre sistemas que utiliza HTTP directamente para el intercambio de datos.

SAE: Sistema de Ayuda a la Explotación. Ver CAD/AVL.

SITP: *Système d'Información pour le Transport Public*. Proyecto francés que lideró el desarrollo de Transmodel v4, trabajando en su reconocimiento como el pre-estándar europeo ENV12896.

SIU: Sistema de Información al Usuario. A veces usado en castellano como acrónimo equivalente a PIDS.

SGML: *Standard Generalized Markup Language*. Estándar (ISO 8879:1986) para definir lenguajes de marcado generalizado para documentos.

SOAP: *Simple Object Access Protocol.* Protocolo estándar que define la comunicación entre dos sistemas mediante el intercambio de datos XML, lo que implica una abstracción sobre el protocolo HTTP.

TCU: *Telematic Control Unit*. Terminología en ocasiones usada para referirse a una OBU muy completa.

TDD: *Test-Driven Development*. Metodología de desarrollo software basada en la creación de los tests que comprueban el comportamiento deseado de una funcionalidad del sistema antes de su propia implementación.

TITAN: *Transmodel-based Integration of Transport Applications and Normalisation*. Proyecto europeo que tenía la finalidad de ampliar Transmodel v3 y plasmarlo como un estándar europeo.

Transmodel: *European Data Model for Public Transport*. Estándar Europeo (EN 12896) que describe de manera exhaustiva un completo modelo abstracto de datos de transporte público.

UITP*: L'Union Internationale des Transports Publics*. Organización internacional sin ánimo de lucro dedicada a la investigación y promoción del transporte público.

UML: *Unified Modelling Language*. Lenguaje de modelado de sistemas de software.

VDV: *Verband Deutscher Verkehrsunternehmen*. Asociación de Compañías de Transporte Alemanas, colaboró en el desarrollo de Transmodel y SIRI.

UUID: *Universally Unique IDentifier*. Número de 128 bits utilizado para identificar unívocamente elementos dentro de un sistema computacional.

W3C: *World Wide Web Consortium*. Consorcio internacional desde el que se proponen estándares web. Entre sus organizaciones miembro se encuentran Apple, Google, Microsoft, IBM, etc.

WSDL: *Web Services Description Language*. Formato de XML usado para describir servicios web, sirviendo como interfaz a implementar por el cliente para tener acceso a ese servicio.

XML: *eXtensive Markup Language*. Metalenguaje subconjunto de SGML capaz de describir diferentes tipos de datos y cuyo propósito es facilitar la transferencia de los mismos a través de Internet

XSD: *XML Schema Definition*. Lenguaje que define la estructura de un documento XML, permitiendo su validación.

1. Introducción, objetivos

1.1. Marco teórico y justificación del proyecto

El transporte público ha sido uno de los sectores en auge durante las últimas décadas, factor que ha favorecido un creciente desarrollo tecnológico en el ámbito del tratamiento de la información generada por los vehículos y su intercambio vía telemática.

La cantidad de conceptos a los que hace referencia esta 'información de transporte público' ha ido aumentando con el tiempo, en función de la utilidad que se ha ido encontrando al tratamiento y almacenamiento histórico de los mismos. Actualmente, esta información abarca:

- Listado de las paradas y líneas disponibles.
- Horarios de los viajes.
- Tiempos de los vehículos en su paso por las paradas.
- Conteo de pasajeros.
- Registros del cobro de billetes (*fare collection*).
- Estado mecánico de los vehículos.
- Posicionamiento.
- Incidencias registradas (cortes de circulación, accidentes, etc.).
- Información sobre el entorno de los vehículos: cantidad de tráfico, contaminación del ambiente, etc.

Parte de esta información es relativa al operador de transporte, mientras que el resto tiene relación directa con un determinado viaje realizado por un determinado vehículo.

En el pasado, si existía la necesidad de transmitir parte de esta información a otros vehículos o a un controlador central, se hacía manualmente mediante sistemas de telefonía/radio.

Por el contrario, hoy en día todos los vehículos de transporte público (y cada vez mayor cantidad de vehículos privados) incluyen una OBU/TCU como la mostrada en la ilustración 1, que centraliza la información que recogen el resto de componentes y sensores de los vehículos. Estos dispositivos embebidos permiten la comunicación automática de cada vehículo con el exterior, bien para recibir instrucciones o bien para transmitir toda esta información relativa al propio autobús, tranvía, tren, *smart-car* (coche inteligente), avión, etc.

Ilustración 1: TCU destinada a ser colocada en un vehículo de transporte público (GMV)

Paralelamente al progreso de la tecnología, la necesidad de las compañías de incorporar a sus sistemas de gestión toda esta información fue haciéndose más acuciante; lo que derivó en otras necesidades: la estandarización de las comunicaciones de esta información y la unificación de los datos que se transmitían en cada una de estas comunicaciones. A estas conclusiones se llegó hace varias décadas, como se describirá con más profundidad en partes sucesivas, pero un ejemplo actual puede ser de utilidad a la hora de entenderlas:

Se puede pensar que el intercambio de toda esta información simplemente tiene lugar de manera bidireccional entre operadores de transporte (ALSA, RENFE, AUVASA), así como entre cada operador de transporte y sus vehículos. Sin embargo, el sistema económico actual diversifica esta situación, pues los operadores de transporte ni fabrican los dispositivos embebidos ni programan el software encargado de monitorizar la información proveniente de los vehículos de cada una de sus flotas; sino que contratan estos servicios a terceras empresas. Yendo aún más allá, la tendencia actual es posibilitar que un operador contrate su FMS (*Fleet Management System*) a una empresa, su FCS (*Fare Collection System*) a otra y pueda instalar validadoras de billetes de un tercer fabricante y OBUs de un cuarto, que recojan la información de dichas validadoras y la envíen a los citados servidores de FMS y FCS.

Además de lo anterior, por encima de los operadores de transporte (PTO, *Public Transport Operator*) se encuentran las autoridades (PTA, *Public Transport*

Authority), que son realmente las entidades encargadas de gestionar el transporte público de una zona geográfica (bien directamente, bien contratando a uno o varios operadores). A nivel de complejidad, esto se traduce en que un operador puede dar servicio a varias autoridades, por lo que va a tener que entenderse con todas ellas además de con el resto de los operadores que cada una de estas contrate. Esta situación se refleja en la siguiente ilustración:

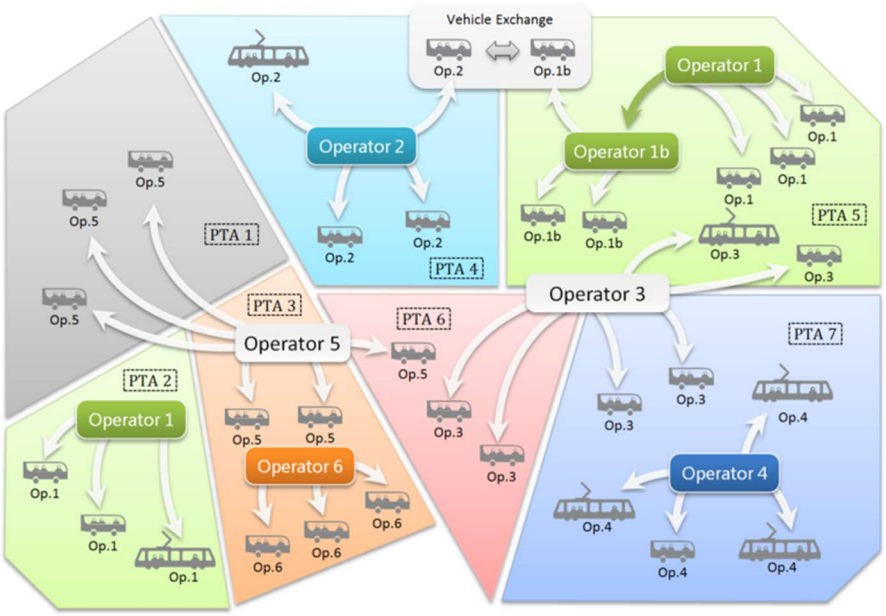


Ilustración 2: Relaciones entre autoridades y operadores de transporte público (ITxPt)

De la complejidad de las relaciones descritas y las tendencias del mercado mencionadas surgió ITxPT (*Information Technology for Public Transport*), cuyo objetivo es el descrito: la implementación de un estándar que permita la total intercambiabilidad entre el hardware usado en sistemas de transporte público [1]. Esto permitiría que un operador de transporte público no se atase al hardware de la empresa a la que comprara el servicio de FMS o FCS.

Estos requisitos son una realidad: ya están empezando a salir a concurso público proyectos cuyos pliegos de condiciones incluyen esta flexibilidad como requisito para la adjudicación de su sistema de gestión de vehículos.

La conclusión es obvia e inmediata: la necesidad de estandarización de la información de transporte público y sus comunicaciones es real y acuciante. Afortunadamente, este hecho fue refrendado por agencias e instituciones de todo el mundo hace años, y desde entonces se ha ido trabajando arduamente en la creación de estándares destinados a satisfacer la descrita necesidad.

Uno de estos estándares existentes para modelar esa información es Transmodel, y uno de los protocolos de intercambio de esa información basados en Transmodel es SIRI: *Service Interface for Real-time Information*.

1.2. Objetivos

Los objetivos principales de este proyecto son los siguientes:

- Dar una amplia visión teórica del estándar SIRI, tanto a nivel global como detallando los modos de funcionamiento que contempla y cada uno de los servicios que ofrece.
- Desarrollar un ejemplo de intercambio de información usando SIRI, mediante implementaciones basadas en el paradigma cliente-servidor.
- Validar nuestra implementación del servidor mediante herramientas externas.
- Valida nuestra implementación del cliente mediante su uso para obtener información real de un servidor externo de SIRI.

La versión de los XSD de SIRI con la que se trabajará es la versión pública más reciente hasta la fecha (v2.0o) del XSD de SIRI [2], así como las últimas versiones oficiales del estándar (CEN/TS 15531:1-5).

1.3. Estructura de la memoria

Esta memoria queda estructurada en seis bloques bien diferenciados:

- Introducción y objetivos.
- Estado del arte de los Sistemas Inteligentes de Transporte.
- Descripción teórica del estándar SIRI.
- Implementaciones de servidor y cliente y su validación.
- Uso de SIRI en la actualidad.
- Conclusiones.

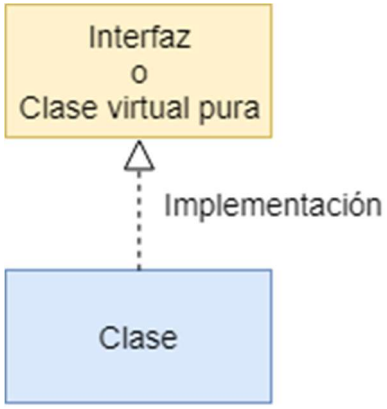
1.4. Guía de nomenclatura y estilos

La terminología en torno al estándar sobre el que trata este documento tiene en ocasiones difícil correspondencia con el castellano. En esos casos se optará por usar las **expresiones inglesas**, marcándolas con *este estilo cursivo* característico de los textos en otro idioma.

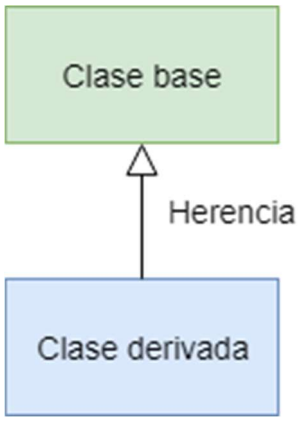
En la parte que en la que se describen las implementaciones de software, se usarán tanto diagramas UML como **fragmentos de código**. Estos últimos podrán reconocerse por usar `esta tipografía característica` para diferenciarlos del resto del texto. Dicho formato también se aplicará cuando se mencionen en el texto **nombres de campos, métodos o clases** del código implementado.

En cuanto al **formato UML**, no se seguirá de manera estricta, siendo las siguientes nociones básicas suficientes para interpretar los diagramas:

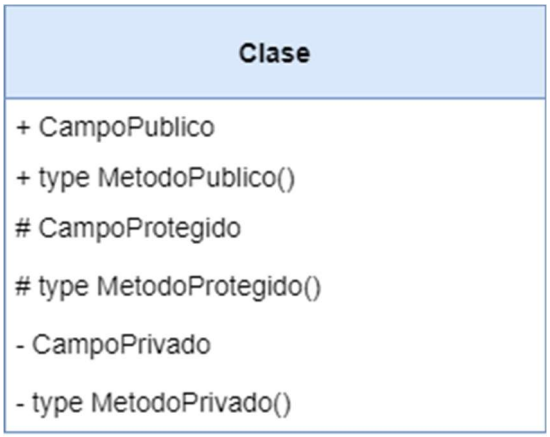
- **Implementación**: línea discontinua acabada en una flecha.



- **Herencia**: línea continua acabada en una flecha.



- Los métodos y campos **públicos** son los precedidos por '**+**', los **protegidos** son los precedidos por '**#**' y los **privados** son los precedidos por '**-**').



2. Estado del arte: Sistemas Inteligentes de Transporte

2.1. Historia de la estandarización de la gestión e intercambio de información relacionada con el transporte público

Durante la década de 1980 fue surgiendo entre agencias y compañías de varios países la necesidad de encontrar una forma de gestionar de manera lógica y uniforme la cada vez mayor cantidad de datos de que disponían, así como poder transmitirlos de manera eficiente.

En una Europa cada vez más global no tardó mucho en extenderse esta necesidad al intercambio de información entre compañías y autoridades de distintos países. Esta información comprendía una diversa cantidad de aspectos: desde los horarios de cada vehículo a los tiempos de paso por paradas, incluyendo el conteo de pasajeros, el cobro de los billetes, el estado de los vehículos, las posibles incidencias registradas, etc.

En 1989 comenzó el proyecto europeo **Cassiope**, en el que se integraron modelos de datos ya existentes usados en Francia, Reino Unido y Alemania; y que finalmente dio lugar al *European Data Model for Public Transport v0.*

El proyecto **EuroBus** (1992-1994), al que se sumaron tanto operadores de nuevos países (España, Holanda) como *partners* industriales (entre los que destacó Siemens) permitió avanzar en la estandarización hasta llegar a una nueva versión del *European Data Model for Public Transport*, lo que se empezó a denominar como **Transmodel** [3]. Esta versión del mismo, Transmodel v3, ya incluía estandarizaciones del modelo de datos para la monitorización automática del vehículo, el cobro de tarifas, etc., lo que supuso un enorme avance cualitativo en la búsqueda inicial de la estandarización.

Con los proyectos **Harpist** y **TITAN** (*Transmodel-based Integration of Transport Applications and Normalisation*) se buscó ampliar el trabajo realizado y plasmarlo como un estándar europeo. En 1992 se creó el CEN/TC 278 – ITS (Comité Técnico del Comité Europeo de Normalización relacionado con los Sistemas Inteligentes de Transporte), [4] y tres años más tarde, Transmodel v4 fue remitido a dicha entidad para su evaluación previa. A continuación, fue la versión v4.1.1 la que se remitió para su aprobación, hecho conseguido en 1997 en forma del pre-estándar europeo ENV12896 gracias al impulso del proyecto francés SITP (*Système d'Information pour le Transport Public*).

Durante los años siguientes el trabajo sobre Transmodel continuó, liderado por el Departamento de Transporte de Reino Unido y apoyado por agencias francesas (RATP), alemanas (VDV), danesas (HÜR), etc.; hasta que en 2006 Transmodel v5.1 fue aprobado como el estándar europeo EN12896.

Transmodel v5.1 fue la piedra angular de muchos otros estándares y especificaciones técnicas que se han basado y siguen basándose en su completo modelo de conceptual de datos, que abarca tanto conceptos y estructuras relacionadas puramente con la lógica de negocio empresarial como con PIDS (*Passenger Information Display System*).

Algunos de ellos son:

- **TransXChange** [5], un estándar *'de facto'* en Reino Unido cuyo desarrollo fue promovido a principios de siglo por el gobierno de ese país. Este comienzo tan temprano del desarrollo implicó que, al contrario que el resto de los estándares que se mencionarán, se basara inicialmente en una versión de Transmodel anterior a la v5.1 (la v4, más concretamente).
  TransXChange está compuesto por una serie de documentos XML que definen una determinada implementación de Transmodel (ilustración 3), conformando junto con otra serie de estándares (NPTG, *National Public Transport Gazeteer*, y NAPTAN, *National Public Transport Access Node*) un conjunto coherente de documentos basados en Transmodel que apoya el gobierno de Reino Unido [6]. Ese apoyo estatal lo ha convertido en un estándar indiscutible en su país, siendo usado principalmente para el intercambio de horarios de autobuses.

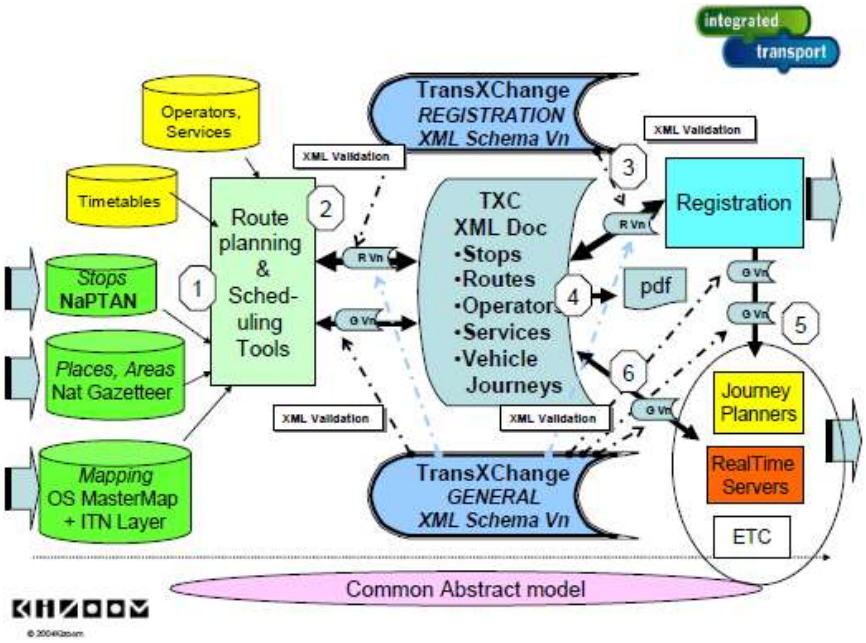


Ilustración 3: Vista general de TransXChange (TransXChange 2.5 v59)

La última versión del estándar es la v2.5, habiéndose desarrollado herramientas para hacerlo compatible con GTFS y manteniéndose en coherencia con las sucesivas actualizaciones de Transmodel. Esto es relevante en tanto lo convierte en fácilmente compatible con otras super-implementaciones de éste, como la que se describe a continuación.

- **NeTEX** (*NETwork EXChange*) [7], una especificación técnica ratificada en el estándar europeo CEN/TS 16614 [8] que define un esquema XML diseñado específicamente para el intercambio eficiente de datos de transporte público.
  Se puede afirmar NeTEX es la implementación más completa y exhaustiva del modelo abstracto que proporciona Transmodel, lo que permite su casi inmediata compatibilidad con otros estándares más reducidos y concretos que se basan en éste, generalmente siendo un super-conjunto de los mismos. En la siguiente ilustración se refleja esta relación con Transmodel y con otras especificaciones XML que lo implementan:

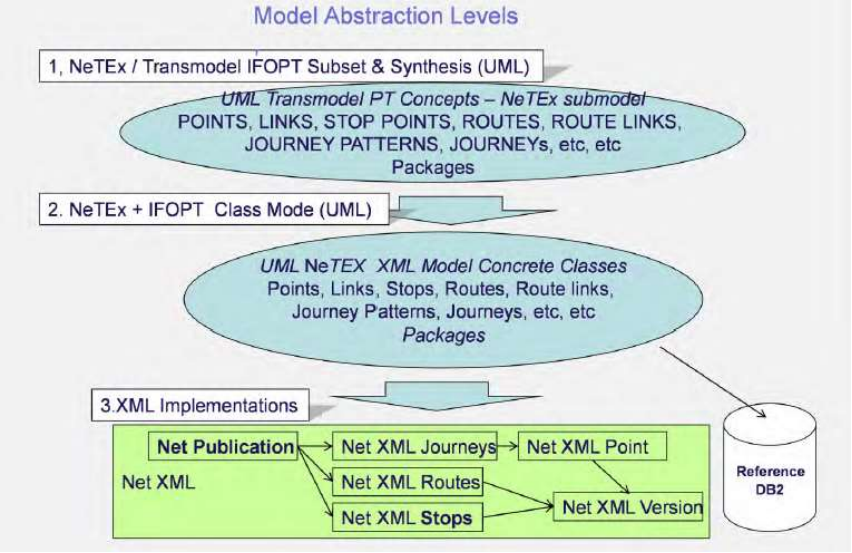


Ilustración 4: Visión general de NeTEx (EN 16614-1)

- **IFOPT** (*Identification of Fixed Objects in Public Transport*), una especificación técnica (CEN/TS 28701) que condensa en un modelo de datos de referencia todos los objetos/actores necesarios para el acceso al transporte público, posibilitando expresar de manera estandarizada el llamado 'guiado puerta a puerta' (*door-to-door guidance*).

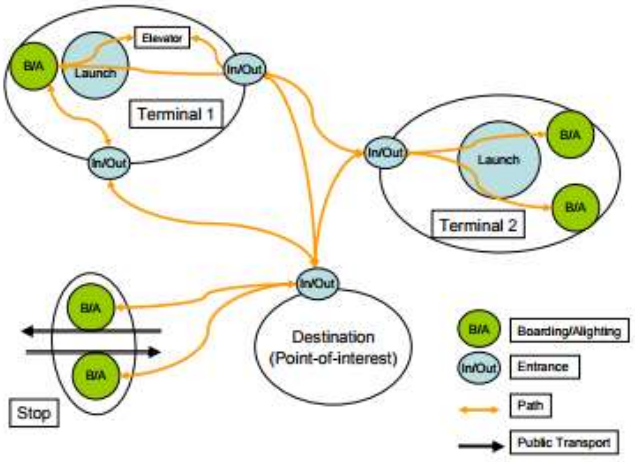


Ilustración 5: Nivel de detalle que alcanza IFOPT (EN 28701)

- **SIRI** (*Service Interface for Real-time Information*), un protocolo de comunicación basado en servicios web que buscó proveer al mundo del transporte público de una interfaz estándar basada en Transmodel para intercambiar datos (tanto planificados como de tiempo real) entre servidores de transporte público [9].
  Fue ratificado como estándar europeo (CEN/TS 15531) en 2007.

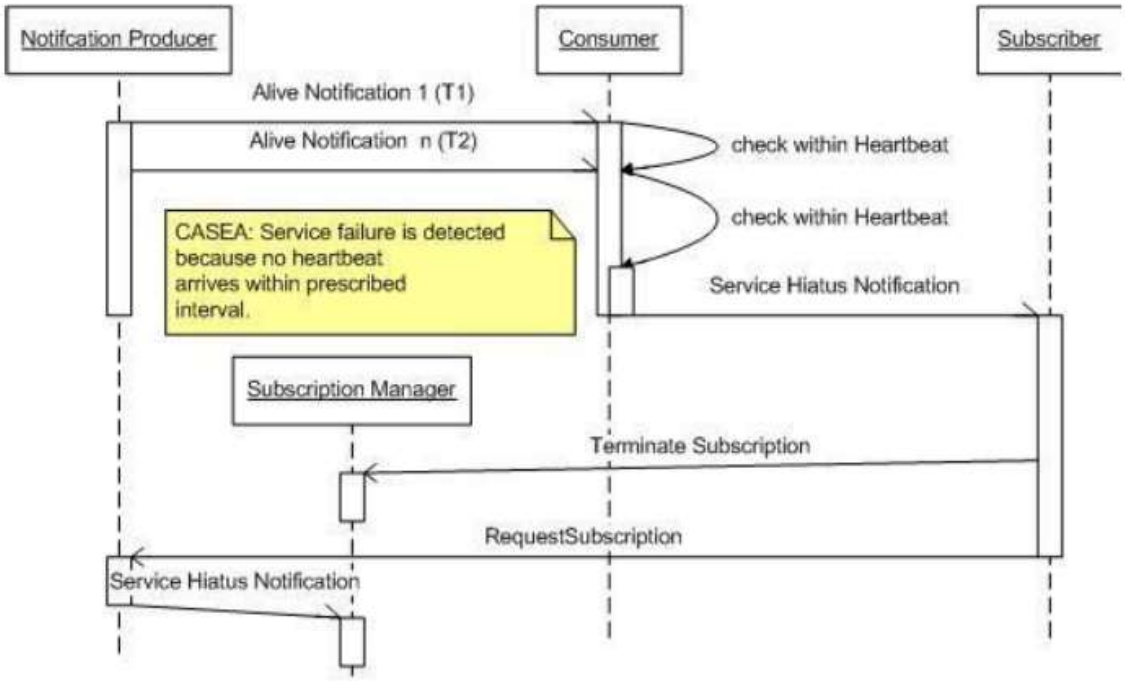


Ilustración 6: Esquema de SIRI (EN 15531-2)

Pese a que estos estándares e implementaciones puedan parecer divergencias respecto al estándar que tanto había costado alcanzar (Transmodel), la realidad es bien distinta debido a que:

- El propio NeTEx pasó a reconocer a SIRI en 2011 como un protocolo válido dentro de su especificación.
- Una de las tareas implícitamente realizadas por NeTEx fue la de unificar de alguna manera Transmodel v5.1 e IFOTP.
- Como ya se ha mencionado, NeTEx es prácticamente un super-conjunto de TransXChange, por lo que la compatibilidad entre ambos es casi inmediata.
- Transmodel v6 incorporó las numerosas aportaciones conceptuales de NeTEx, manteniendo así la coherencia entre los estándares.

Paralelamente al desarrollo europeo de Transmodel, en Estados Unidos y de la mano de Google surgió **GTFS** (*Google Transit Feed Specification*) como herramienta para ayudar a la implementación de funcionalidades de guiado en Google Maps [10].

La inexistencia de un único estándar previo y el hecho de que GTFS fuera sencillo, público y gratuito hicieron que se extendiera rápidamente hasta alcanzar una gran fama. Esa enorme expansión hizo que en 2009 pasase a llamarse *General Transit Feed Specification*, evitando así posibles reticencias a la hora de usarlo debido a la referencia explícita a la compañía de Alphabet.

Un *feed* GTFS consta de un archivo de extensión .zip dentro del cual se incluyen una serie de archivos de texto (.txt) con un formato determinado:

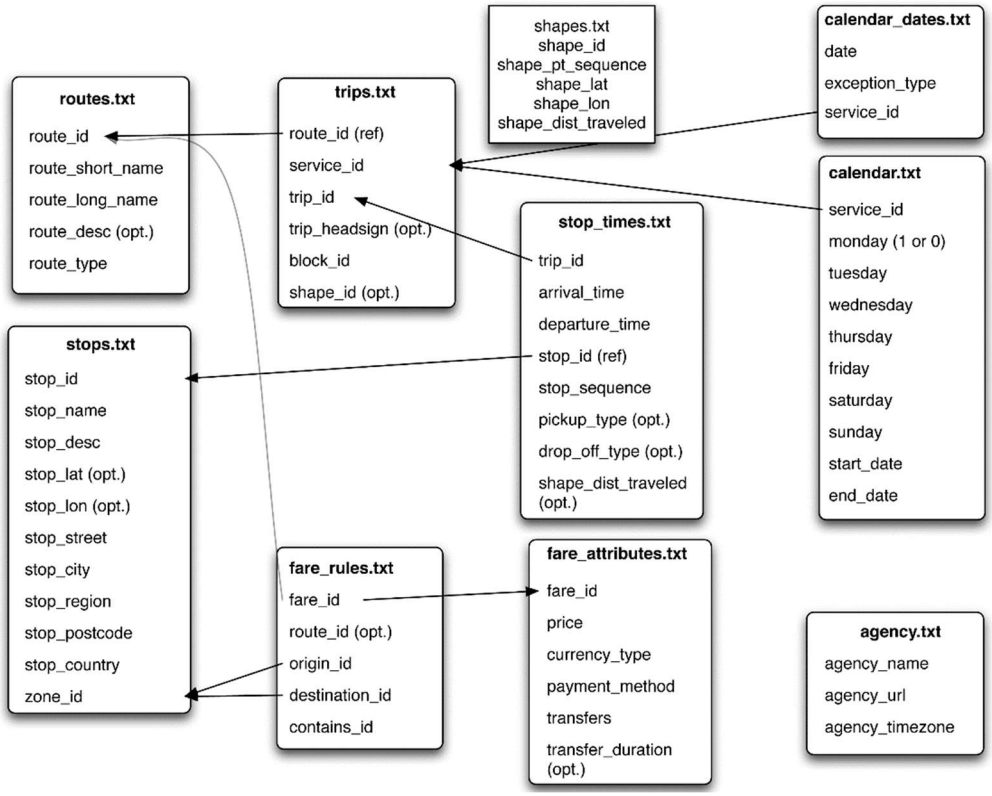


Ilustración 7: Archivos que componen un feed GTFS (GTFS examples v1.42 [11])

Los archivos .txt constan de una cabecera, para indicar los campos que incluye el archivo, y columnas de datos separadas por comas.

Un ejemplo de uno de esos archivos puede ser:

stop_times.txt

```
trip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,drop_off_type
AWE1,0:06:10,0:06:10,S1,1,0,0
AWE1,,,S2,2,1,3
AWE1,0:06:20,0:06:30,S3,3,0,0
```

Más tarde, **GTFS-R** (*General Transit Feed Specification for Real-time*) fue creado para cubrir las necesidades de intercambio de datos de tiempo real, siguiendo la misma filosofía que GTFS [12].

2.2. Transmodel: *European Data Model for Public Transport*

Como el estándar del que trata este documento (SIRI) está basado en Transmodel, conocer los conceptos básicos de este último resultará útil para la correcta y completa comprensión del primero.

El modelo abstracto de datos que nos ofrece el estándar se basa en *Entities* (entidad padre de todos los objetos) que se agrupan en *Frames* (conjuntos de entidades que comparten una *Validity Condition*) [13], como se aprecia aquí:

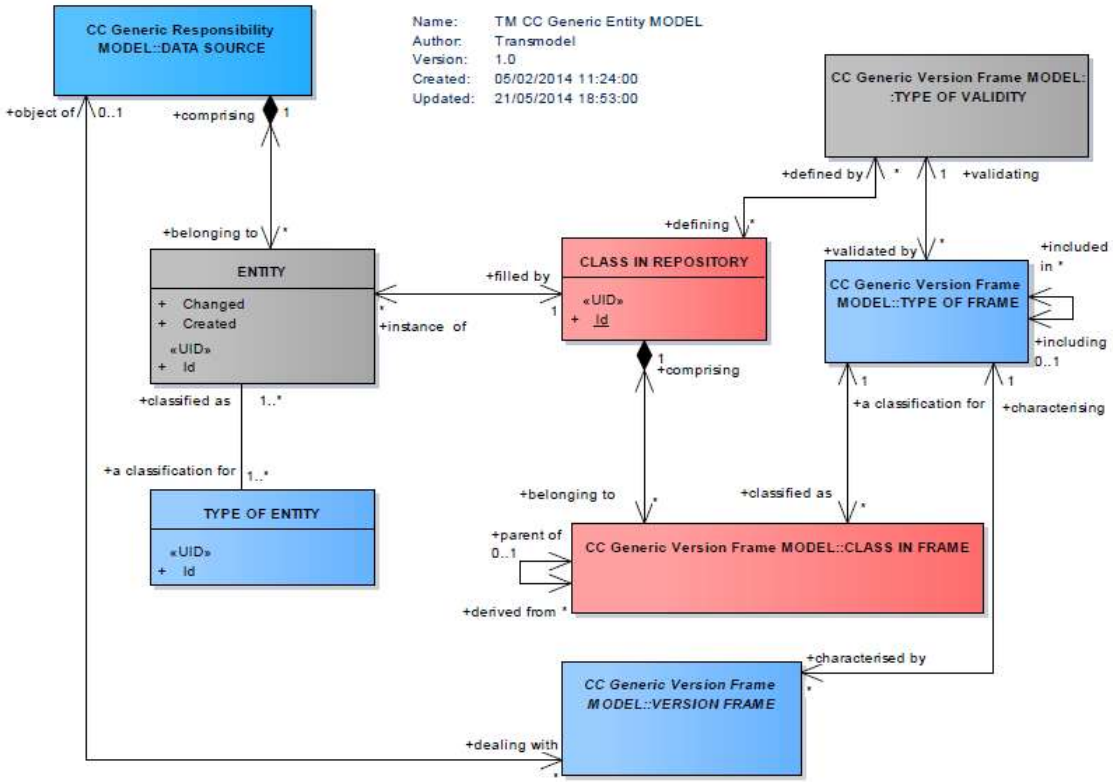


Ilustración 8: Modelo UML conceptual de Entity (Transmodel v6 - P1)

El modelo también contempla el versionado de estas entidades:

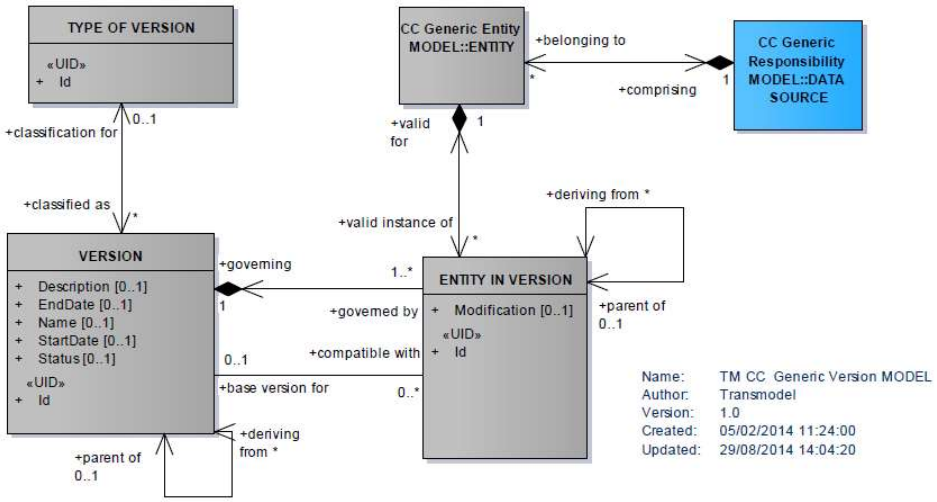


Ilustración 9: Modelo UML conceptual del versionado (Transmodel v6 - P1)

De esta manera, y teniendo en cuenta que las versiones también se agrupan en *frames*, una visión general de la estructura de cada una de las entidades que define Transmodel es la siguiente:

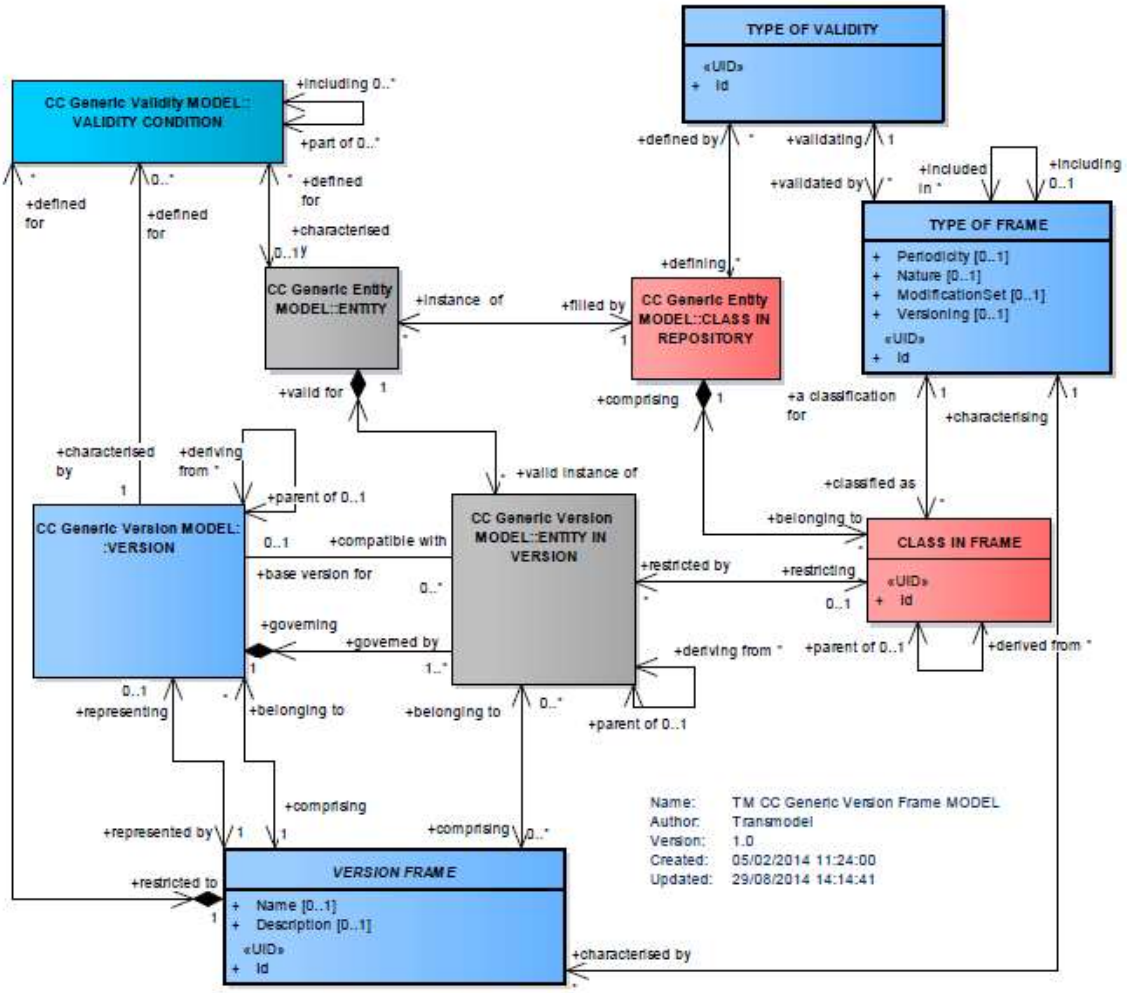


Ilustración 10: Modelo UML conceptual del 'version frame' (Transmodel v6 - P1)

De entre todas las entidades o *entities* básicas que componen Transmodel, conviene estar familiarizado con las siguientes para interpretar correctamente el estándar SIRI y los servicios que éste ofrece:

- *Point:* nodo de dimensión 0 de la red. Comprende, entre otros, a:
  - *Scheduled Stop Point*: punto en el que los pasajeros pueden subirse o bajarse de vehículos.
  - *Timing Point*: punto que puede figurar en un horario, es decir, al que se le puede asociar un tiempo medido.

  - *Connection Point:* punto en el que los pasajeros cambian de vehículo de transporte público para continuar su viaje, quedando definido por dos *Scheduled Points.*
- *Route*: lista ordenada de puntos que definen un camino. Puede contener un mismo punto varias veces.
- *Line*: grupo de *Routes* generalmente conocido por los usuarios del transporte público.
- *Direction*: clasificación de la orientación general de una *Route*.
- *Vehicle*: vehículo de transporte público usado para llevar pasajeros.
- *Journey Pattern*: lista ordenada de *Scheduled Stop Points y Timing Points* dentro de una *Route* que describe un viaje de transporte público. Tiene origen y destino.
- *Vehicle Journey*: desplazamiento de un vehículo de transporte público en un tipo de día, desde el inicio hasta el final de un *Journey Pattern* sobre una *Route* específica.
- *Service Journey*: viaje (*Vehicle Journey*) de un pasajero en un día determinado.
- *Headway Journey: Vehicle Journey* definido por la frecuencia de paso de los vehículos en vez de por un horario fijo.
- *Headway Interval*: intervalo de frecuencia de un *Headway Journey.*

Hay que tener en cuenta que Transmodel v6 se divide en ocho partes:

1. Conceptos comunes.
2. Red de Transporte Público.
3. Información de horarios y planificación de vehículos.
4. Monitorización y control de operaciones.
5. Gestión de cobros y tarifas.
6. Información al pasajero.
7. Gestión de conductores.
8. Gestión de información y estadísticas.

Las entidades anteriormente descritas representan solo una pequeña porción de las incluidas en la primera parte, llevando cada una de ellas asociado el modelo de versionado y *frames* descrito en la figura 10; lo que convierte a Transmodel en una **herramienta que nos provee de un completísimo modelo abstracto de datos sobre el que estructurar nuestras implementaciones concretas**.

La idea que subyace bajo este estándar es precisamente que esas implementaciones, por las características de los sistemas reales sobre los que se apliquen, puedan prescindir de muchas de las entidades definidas en el propio estándar; pero que no tengan la necesidad de introducir conceptos nuevos no contemplados en él, ya que dicha introducción convertiría al sistema en cuestión en incompatible con Transmodel.

Prueba de la complejidad de Transmodel es la existencia de las siguientes entidades para describir algo aparentemente tan simple como un punto:

- *Point.*
- *Timing Point.*
- *Scheduled Stop Point.*
- *Traffic Control Point.*
- *Activation Point.*
- *Beacon Point.*
- *Garage Point.*
- *Infrastructure Point.*
- *Parking Point.*
- *Point in Journey Pattern.*
- *Point in Link Sequence.*
- *Point on Link.*
- *Point of Interest (POI).*
- *Point Projection.*
- *Route Point*
- *Stop Area.*
- *Stop Place,*
- *Vehicle Stop Place.*
- *Zone.*
- *Access Zone.*
- etc.

3. Desarrollo teórico: el estándar SIRI

3.1. Introducción y alcance de SIRI

SIRI nació como un **protocolo estandarizado de comunicación de datos de transporte público**, siendo especialmente diseñado para ser usado en las **comunicaciones entre servidores de datos** e incluyendo desde sus más tempranos planteamientos el soporte a intercambios de información de **tiempo real**.

Basado en Transmodel, su desarrollo comenzó a principios de siglo, tras la ratificación de éste como pre-estándar europeo. Este desarrollo fue impulsado tanto por agencias públicas como privadas de múltiples países europeos, entre los que destacaron Francia (AFIMB), Alemania (VDV), Reino Unido (RTIG) y Suecia; contribuyendo posteriormente otros como Dinamarca, Noruega o República Checa [14]. Fue finalmente ratificado como estándar europeo en 2007.

Esta v1.0 contenía tres partes:

- CEN/TS 15531-1:2007: *Part 1: Context and Framework.*
- CEN/TS 15531-2:2007: *Part 2: Communications infrastructure.*
- CEN/TS 15531-3:2007: *Part 3. Functional Service Interfaces.*

El desarrolló continuó, y en la v1.4 se añadieron dos partes más:

- CEN/TS 15531-4:2011: *Part 4: Functional Service Interfaces: Facility Monitoring.*
- CEN/TS 15531-5:2011: *Part 5: Functional Service Interfaces: Situation Exchange.*

La versión 2.0 incorporó mejoras suficientes para renovar la mayoría de estas partes, quedando los estándares de la siguiente manera:

- CEN/TS 15531-1:2015: *Part 1: Context and Framework.*
- CEN/TS 15531-2:2015: *Part 2: Communications infrastructure.*
- CEN/TS 15531-3:2015: *Part 3. Functional Service Interfaces.*
- CEN/TS 15531-4:2011: *Part 4: Functional Service Interfaces: Facility Monitoring.*
- CEN/TS 15531-5:2016: *Part 5: Functional Service Interfaces: Situation Exchange.*

En el momento en el que se escriben estas líneas, la última versión pública de los XSD SIRI es la v2.0o, la cual incorpora algunas correcciones a los XSD publicados en la v2.0 [15].

Como ya se ha mencionado, SIRI fue especialmente diseñado para comunicar servidores de datos entre sí, y más concretamente para comunicar:

- **Centros de control de vehículos de transporte público entre sí.** Estos centros son los lugares desde los cuales cada operador monitoriza y controla sus flotas de autobuses/trenes/tranvías, etc.
- **Centros de control de vehículos con centros o sistemas de comunicaciones.** Estos últimos, diseñados específicamente para transmitir información de transporte público, suelen además cumplir la función de agrupar y dar coherencia a diferentes tipos de datos provenientes de distintas fuentes para ofrecer un conjunto coherente de información preparada para ser consumida y distribuida masivamente.
- **Centros o sistemas de comunicaciones con el propio usuario o pasajero.** A través páginas web, *apps*, etc.

De esto se deduce que SIRI no está diseñado para las comunicaciones entre los vehículos y los centros de control, sino para comunicaciones de un nivel superior. Este diseño implica una estructura de datos compleja, preparada para transmitir información completa e íntegra. Dicha complejidad no compromete la capacidad de SIRI para comunicar datos en tiempo real, pero sí nos indica que no es un protocolo preparado para ser usado en equipos embarcados, con menor capacidad de procesamiento y generalmente más restricciones en cuanto al envío y recepción de datos.

Aunque al final de este documento estudiamos en detalle algunas de las implementaciones de SIRI en el mundo, el siguiente esquema del modelo de comunicaciones de datos usado en la República Checa nos permite ver a qué niveles se usa SIRI. En este caso, desde los AVMS (*Automated Vehicle Management Systems*) hacia arriba, pero nunca para intercambiar información con vehículos o paneles.

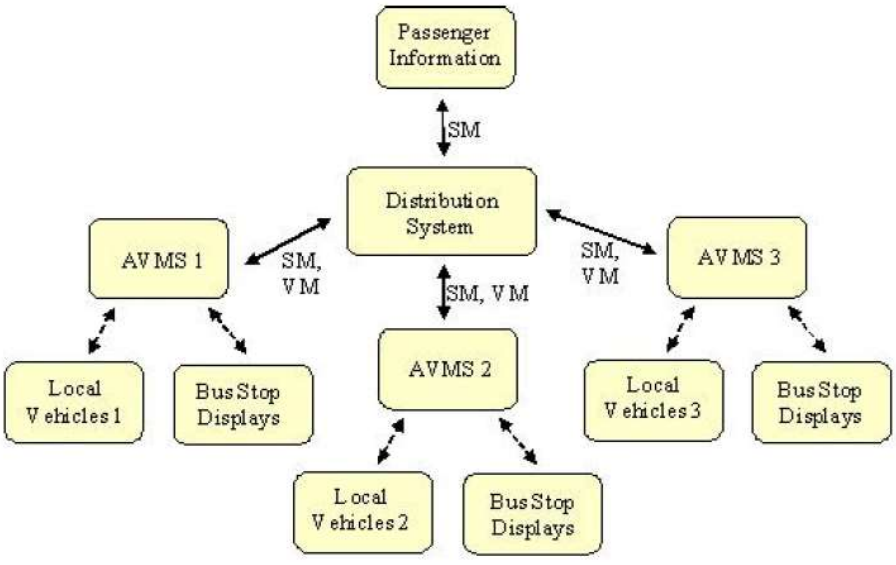


Ilustración 11: Organización de los datos de transporte en la República Checa (EN 15531-1)

3.2. Comunicaciones

3.2.1. Patrones de intercambio de datos

SIRI plantea el modelado del intercambio de datos basándose en (que no estrictamente siguiendo) el paradigma cliente-servidor de la siguiente manera:

- Un **servidor** de SIRI siempre es la fuente de datos en un intercambio de información (*Data Producer)*. Por tanto, se asume que tiene acceso a ellos de uno u otro modo.
- Un **cliente** de SIRI siempre es el receptor de la información que se quiere comunicar (*Data Consumer*), bien de manera activa o bien de manera pasiva.

Teniendo en cuenta lo anterior, los servicios de SIRI contemplan dos conocidos patrones a la hora de realizar dichos intercambios de información: *Request/Response* y *Publish/Subscribe*.

Estos patrones son complementarios, pudiendo un servicio implementar ambos y siendo en este caso el usuario el que elija el más conveniente en función de la naturaleza de su aplicación.

En caso de que ambos sean implementados para un mismo servicio, **el contenido de un mensaje debe ser el mismo independientemente del patrón de intercambio de datos que se use**.

3.2.1.1. *Request/Response*

Este tipo de interacción es el que normalmente se asocia al paradigma cliente-servidor. Funciona del siguiente modo (simplificado):

Un cliente realiza una petición (*request*) a un servidor que tiene implementado un determinado servicio.

El servidor asocia un identificador único a esa petición, la lleva a cabo mediante el servicio correspondiente y rellena la respuesta (*response* o *reply*) con la información (*payload*) pertinente: los datos solicitados, una estructura de error, etc.

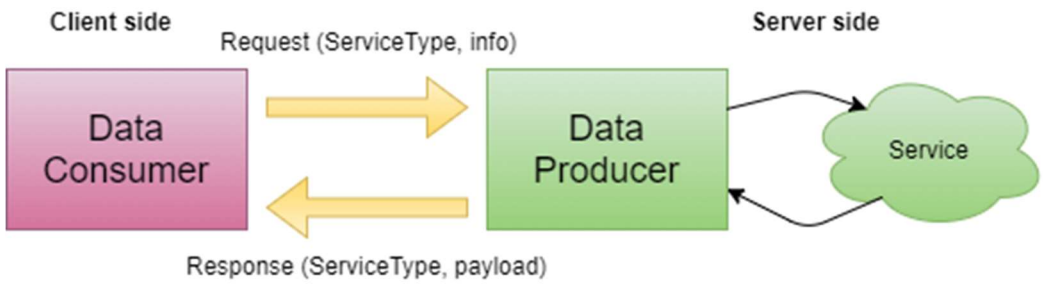


Ilustración 12: Intercambio simple tipo Request/Response (EN 15531-2)

Una *request* puede incluir a su vez peticiones a varios servicios de SIRI, o incluso bajo unos determinados supuestos, múltiples peticiones a un mismo servicio; tal y como se refleja en la siguiente imagen:

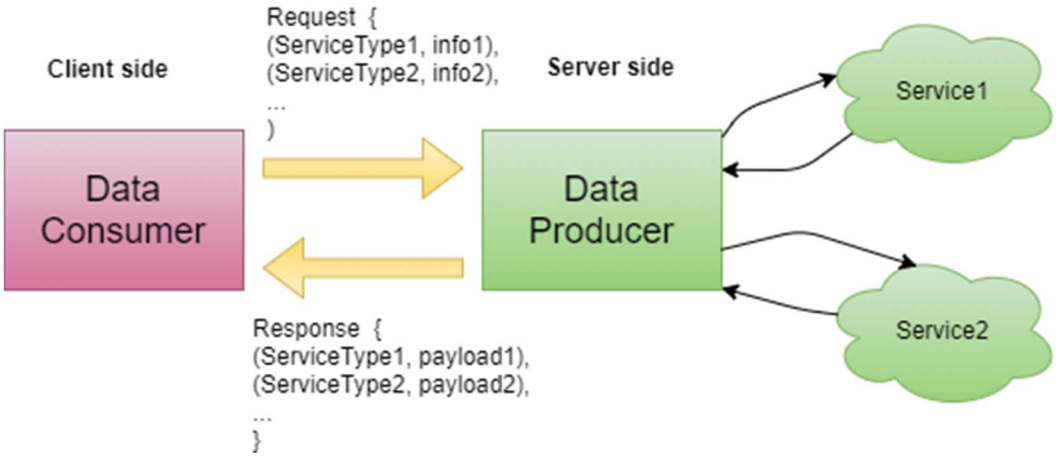


Ilustración 13: Peticiones compuestas en un intercambio tipo Request/Response

3.2.1.2. *Publish/Subscribe*

El patrón *Publish/Subscribe* que adopta SIRI sigue el paradigma descrito en el documento de la W3C titulado '*Publish-Subscribe Notificacion for Web Services (WS-PubSub)*'.

La idea básica, manteniendo de manera momentánea los conceptos de cliente y servidor, es la siguiente:

- El cliente solicita al servidor una subscripción a un servicio específico.
- En caso de que ésta sea aceptada, es el servidor el que manda periódicamente información al cliente teniendo en cuenta los criterios elegidos por éste en su petición inicial.

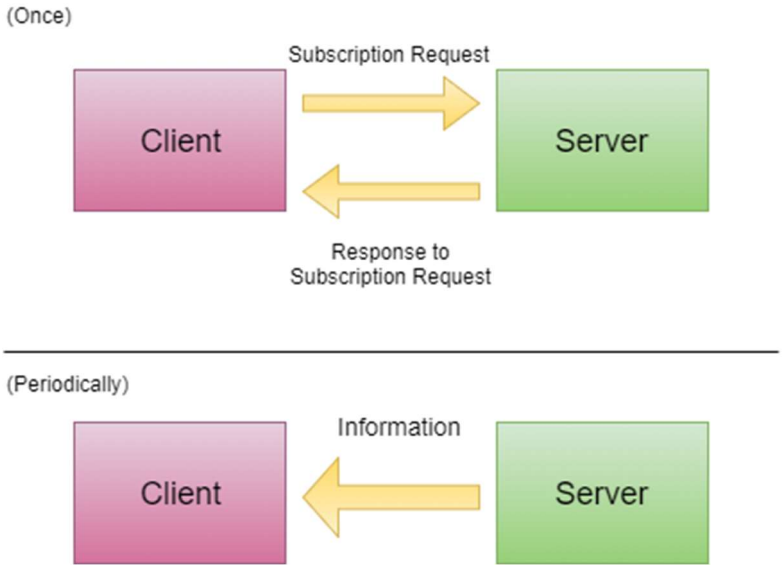


Ilustración 14: Idea básica detrás del intercambio tipo Publish/Subscribe

Como se puede deducir, la división de las entidades participantes en el intercambio de datos en cliente y servidor queda obsoleta para describir en profundidad el modelo *Publish/Subscribe*, cuyos actores reales son:

- ***Subscriber***. Solicita una subscripción a un determinado servicio.
  En los intercambios *Request/Response* su equivalente es el **Requestor**.
- ***Publisher***. Se encarga de avisar mediante eventos de cambios en el sistema o en los datos monitorizados.
- ***Notification Producer***. Recoge los eventos producidos por el *Publisher*, los compara con las subscripciones activas existentes en ese momento y se encarga de su distribución, si procede, hacia el *Notification Consumer*.
- ***Notification Consumer***. Recibe periódicamente información del *Notification Producer*.
- ***Subscription Manager***. Gestiona las subscripciones y su ciclo de vida, una vez registradas.

En el siguiente esquema, en el que a continuación profundizaremos, se describe el funcionamiento básico de las subscripciones:

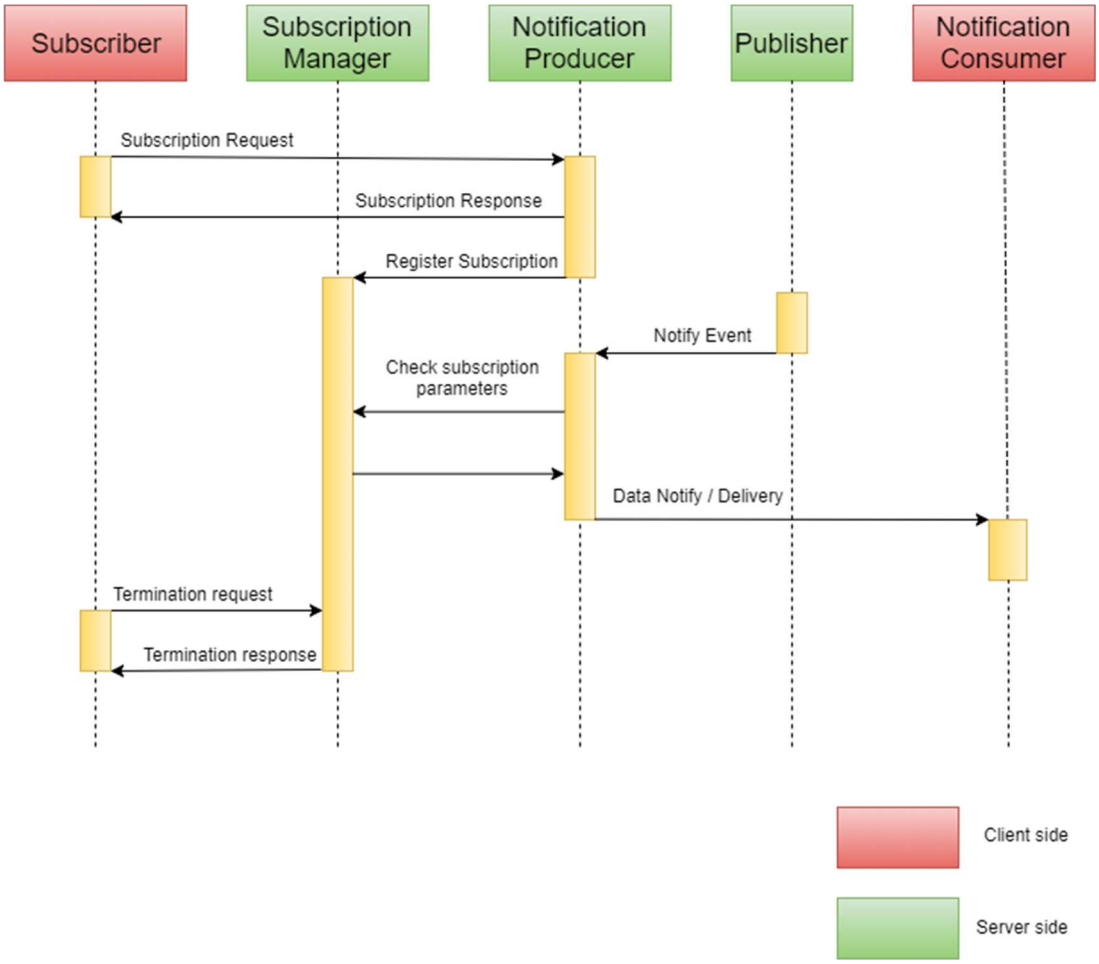


Ilustración 15: Esquema de los actores básicos de un intercambio tipo Publish/Subscribe

El flujo que se pretende describir en el esquema es el siguiente:

- Una subscripción comienza cuando un suscriptor (***Subscriber***) solicita al servidor subscribirse a un determinado servicio.
- La parte del servidor encargada de gestionar estas subscripciones es el ***Notification Producer***, quien a su vez registra la subscripción (o no) en el gestor de subscripciones (*Subscription Manager)* y responde al suscriptor confirmando o denegando su petición.
- Desde el momento en el que se registra en él una subscripción, el ***Subscription Manager*** es el encargado de gestionar la misma (lo que incluye gestionar su ciclo de vida, ya que las subscripciones pueden y suelen estar programadas para caducar después de un cierto período de inactividad).
- El ***Publisher*** es el encargado de comunicar cualquier cambio relevante en la información que se esté gestionando al *Notification Producer*, generalmente a través un evento. Lo habitual es que el *Publisher* esté asociado a su vez a eventos que detecten los cambios citados.
- El ***Notification Producer***, cuando es avisado por el *Publisher* de un cambio relevante, consulta al *Subscription Manager* sobre el interés que pueda generar ese cambio que acaba de producirse en las subscripciones activas en ese momento. En caso de que exista dicho interés, se encarga de notificar ese cambio al *Notification Consumer* asociado a cada una de las subscripciones 'interesadas' en él.
- En cualquier momento un ***Subscriber*** puede solicitar el fin de una subscripción, petición que se encargará de recoger, ejecutar y responder el *Subscription Manager*.

Aunque en esto se profundizará más adelante, una subscripción contiene, además del tipo de servicio al que desea subscribirse su emisor (y en función del mismo, campos adicionales que concretan a qué aspectos/elementos del servicio desea ser suscrito), información sobre el *Subscriber* y sobre el *Notification Consumer* (el *endpoint* al que se desea enviar la información). Esto es relevante porque supone la división el cliente inicial en la parte encargada de crear la subscripción y la parte encargada de recibir los datos de ésta. Además de la información anterior, las subscripciones también deben poder identificar el *Notification Producer* al que están asociadas, al poder darse el caso de la existencia de varias subscripciones iguales gestionadas por distintos *Notification Producer*.

Tal y como se ha explicado, una vez que el *Notification Producer* registra una subscripción, es el *Subscription Manager* el encargado de la misma. Esto implica que el *Notification Producer* tendrá que encargarse de pedir información sobre las subscripciones existentes al *Subscription Manager* y no al revés, estando justificado este diseño para cumplir con la arquitectura *WS-PubSub* anteriormente mencionada. Dicha arquitectura permite métodos muy

detallados de control de subscripciones (renovación, pausa/comienzo tras pausa, edición de la política de la subscripción sin necesidad de crearla de nuevo, etc.), pero la versión actual de SIRI no especifica ninguno de los anteriores; contemplando únicamente la solicitud de finalización de una subscripción y la solicitud de finalización de todas las subscripciones.

El flujo anterior es el flujo interno que se da en la mayoría de las implementaciones reales de SIRI, pudiendo simplificarse en unos casos y complicarse en otros. Sin embargo, desde el punto de vista de un cliente:

Partamos de la base de que la lógica del *Publisher* y del *Notification Consumer* no está necesariamente separada (de hecho, se conoce como patrón *Broker* la situación en la que sí lo está, lo que permite introducir varios *Notification Consumer* para aplicar diferentes filtros a los eventos, escalar el servicio, etc.). En cualquier caso, ambas partes (junto con el *Subscription Manager*) son transparentes para el cliente: él sólo tiene que comunicarse con un servidor, al que envía peticiones de inicio/fin subscripción y que le provee de datos cuando ésta está activa.

Si mantenemos la perspectiva del cliente y nos abstraernos completamente de la implementación, un cliente se subscribe y luego recibe datos, resultándole irrelevante si el *endpoint* del receptor es el mismo que el del suscriptor, si están implementados en la misma aplicación, etc.

Esto permite simplificar aún más el modelo conceptual, condensando un intercambio de tipo *Publish/Subscribe* en el siguiente esquema, basado de nuevo en el paradigma cliente-servidor.

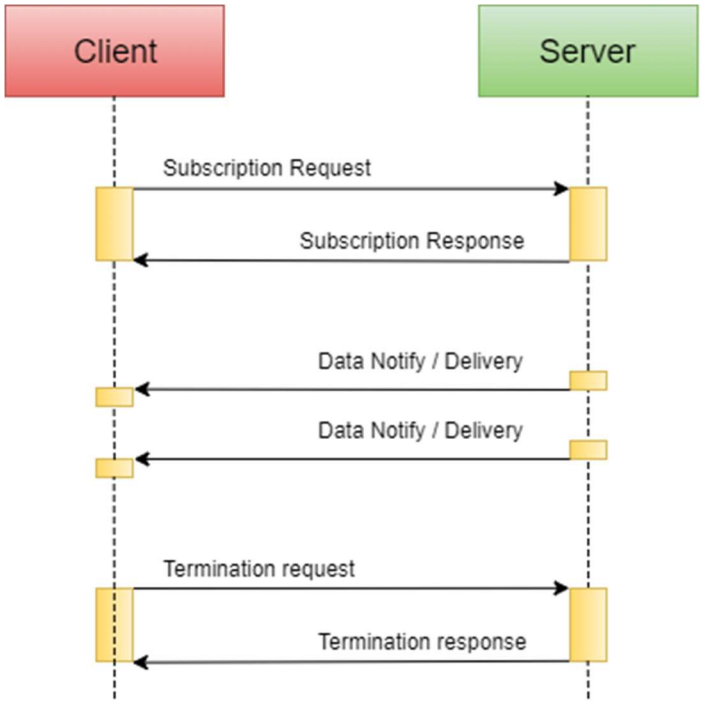


Ilustración 16: Modelo de un intercambio tipo Publish/Subscribe simplificado

Al igual que en un intercambio de tipo *Request/Response*, una *Subscription Request* en SIRI puede aunar dentro de sí varias peticiones de subscripción al mismo o a distintos servicios. El comportamiento del servidor debe ser el de registrar cada una de las subscripciones por separado y responder en un solo mensaje confirmando o descartando el correcto registro de todas ellas, de manera que todas sean rechazadas en caso de que alguna de ellas sea inválida.

Por último, es necesario mencionar que SIRI establece que los mensajes que envía el *Notification Producer* para satisfacer una subscripción a un determinado servicio deben incluir únicamente los cambios sucedidos desde el último envío a dicho *Notification Consumer* (es decir, las actualizaciones); y nunca el grueso de la información disponible.

En caso de que el comportamiento deseado sea este último, la manera de proceder del cliente debe ser la de solicitar los datos de un determinado servicio mediante una petición *Request/Response*.

3.2.2. Patrones de entrega de datos

Denominamos patrones de entrega de datos (*Delivery patterns*) a las alternativas que contempla SIRI a la hora de enviar un conjunto de datos desde el servidor hasta el cliente.

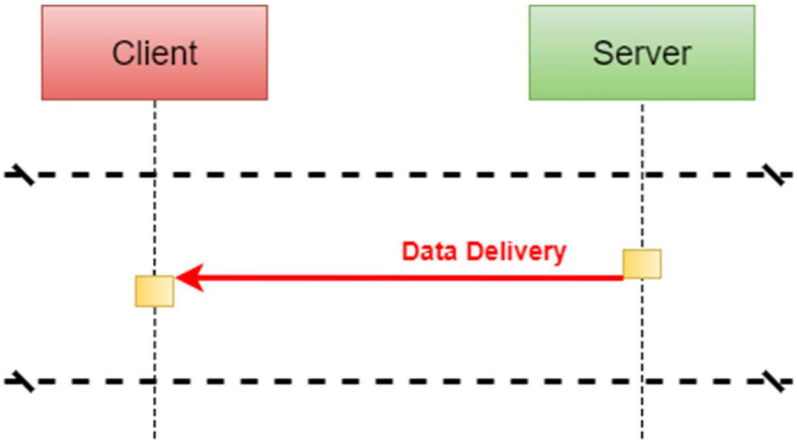


Ilustración 17: Data Delivery o entrega de datos

Es decir, en caso de un intercambio tipo *Request/Response*, el patrón de entrega es la manera que tiene el servidor de enviar esa *Response*; y en el caso de *Publish/Subscribe*, es la manera que tiene el *Notification Producer* de satisfacer una subscripción mediante la propagación hacia el *Notification Consumer* de los cambios correspondientes.

Al igual que sucedía con los dos patrones de intercambio de datos, **para un mismo servicio el contenido de un mensaje debe ser el mismo independientemente del patrón de entrega de datos que se use.**

3.2.2.1. *Direct Delivery*

Una entrega directa (o *Direct Delivery*) consta de un solo paso, que es el propio envío de los datos. Es decir, se usa un único mensaje para realizar el envío de datos, que contiene el *payload* o contenido que se quiere enviar.

En un intercambio *Request/Response*, una *Response* de tipo *Direct Delivery* contiene los datos solicitados por la *Request*.

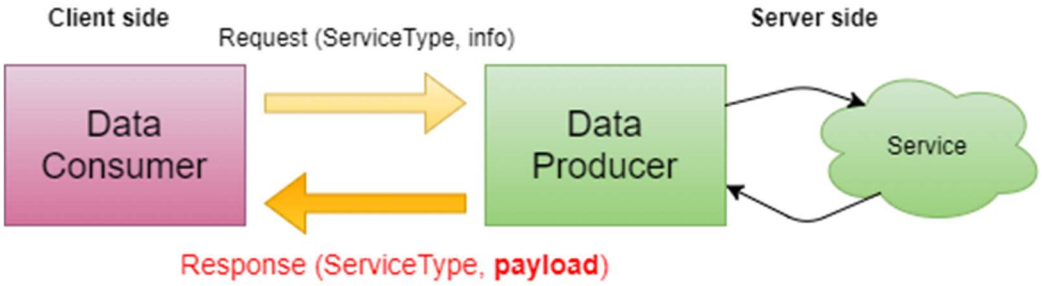


Ilustración 18: Patrón Direct Delivery en un intercambio tipo Request/Response

En un contexto de *Publish/Subscribe*, cuando el *Notification Producer* ha comprobado que hay uno o varios *Notification Consumers* a los que tiene que enviar información, simplemente envía un único mensaje a cada uno que contiene la nueva información o actualización (notificación + envío).

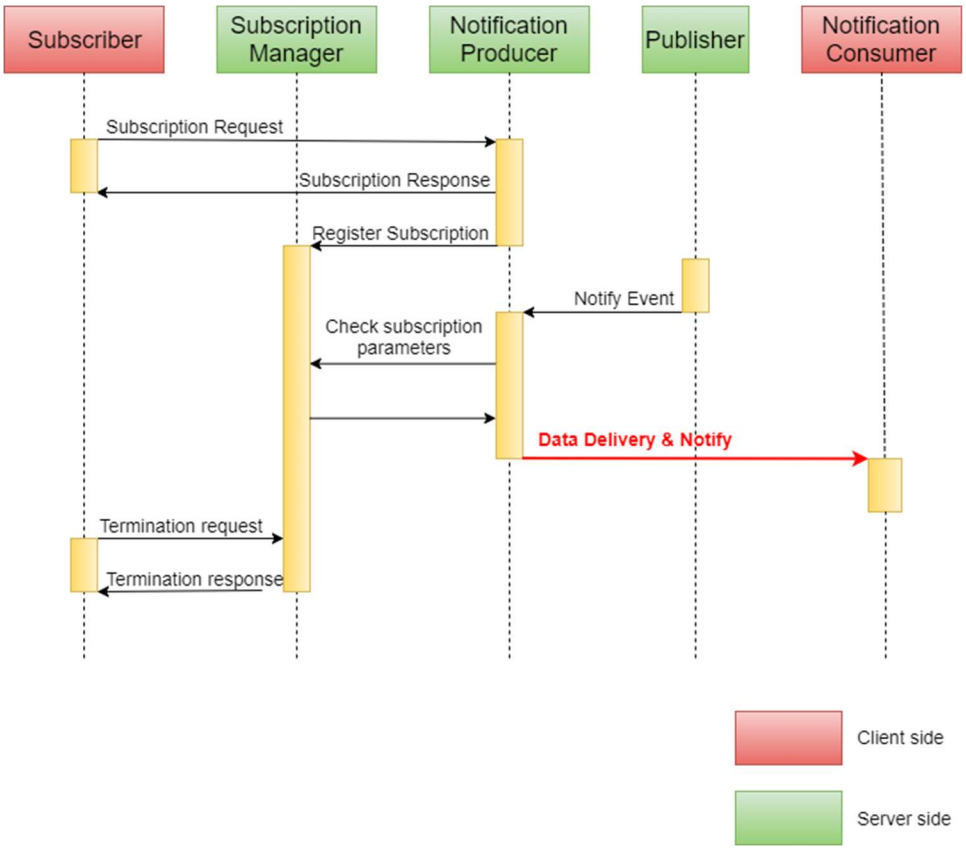


Ilustración 19: Patrón Direct Delivery en un intercambio tipo Publish/Subscribe

Analizando con más profundidad el mecanismo de *Direct Delivery*, al que nos referiremos también como entrega directa, se puede deducir que:

- La implementación de *Direct Delivery* es muy simple, tanto a la hora de desarrollarla como de mantenerla.
  Además, el uso de este mecanismo supone una ventaja a la hora del escalado, ya que el servidor no almacena ni encola datos, por lo que tampoco tiene que reservar recursos para el posterior envío de estos.
- En contraposición a lo anterior, la parte de gestión recae irremediablemente sobre el cliente (*Data/Notification Consumer*), a quien en el caso de estar en un contexto de *Publish/Subscribe*, pueden no interesarle los datos en el momento en el que estos están listos (en el caso de que, por ejemplo, su recepción retrasara otras tareas más prioritarias).

3.2.2.2. *Fetched Delivery*

La llamada *Fetched Delivery* puede traducirse por una especie de 'entrega con confirmación', lo que da sentido a la explicación anterior sobre la *Direct Delivery*.

En una *Fetched Delivery*, el servidor primero notifica al cliente que los datos están listos para ser enviados, de manera que es éste quien elige el momento en que quiere que se produzca la entrega. Por tanto, consta de cuatro pasos:

- El *Notification Producer* manda una notificación al *Notification Consumer* indicándole que existen nuevos datos disponibles.
- El *Notification Consumer* confirma al anterior que ha recibido la notificación.
- El *Notification Consumer* pide los datos al *Notification Producer* mediante una *DataSupplyRequest*.
- El *Notification Producer* manda los datos (el *payload*) solicitados en una *DataSupplyResponse* como respuesta a la anterior petición.

Otros detalles a destacar de la *Fetched Delivery* son los siguientes:

- Si mientras el *Notification Producer* espera la solicitud de datos del *Notification Consumer*, el payload que el primero está almacenando para enviárselo al segundo cambia, sólo se deben enviar los datos más actualizados; lo que como veremos tendrá sus beneficios y sus inconvenientes.
- El *Notification Producer* no mantiene los datos disponibles de manera indefinida tras notificar al cliente, denominándose *Data Horizon* a la fecha hasta la cual se garantiza que las actualizaciones seguirán disponibles.

Este esquema condensa una entrega *Fetched Delivery* en un intercambio tipo *Request/Response* (el caso más simple).

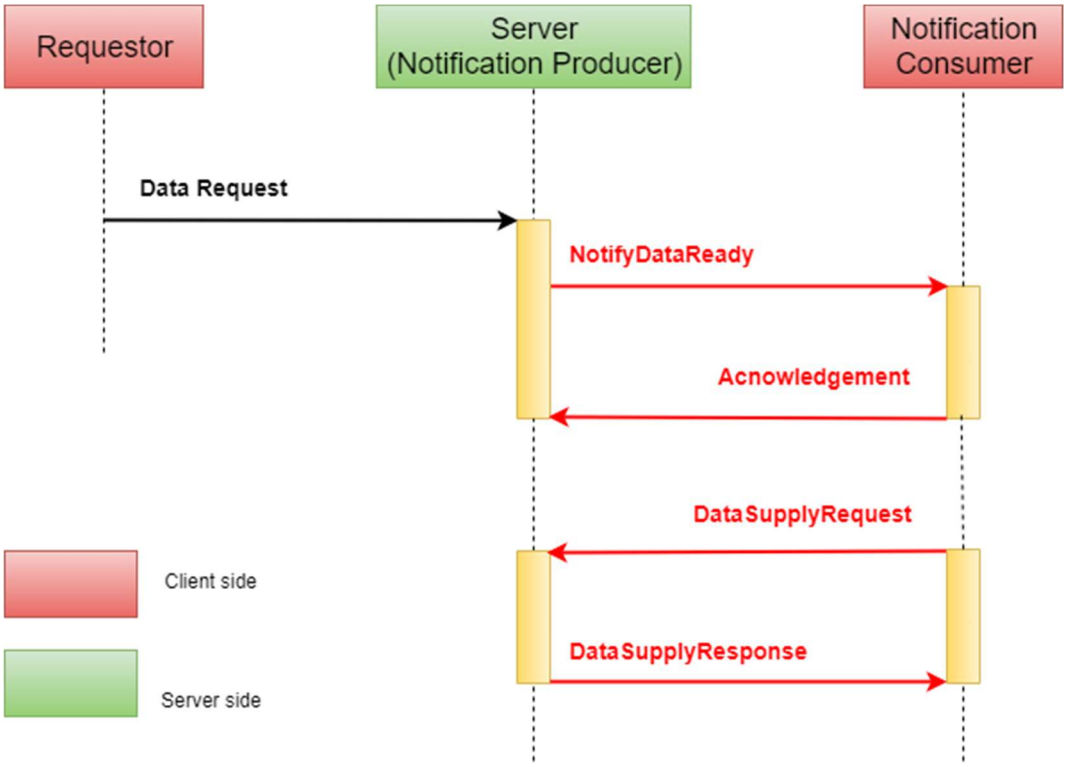


Ilustración 20: Patrón Fetched Delivery en un intercambio tipo Request/Response

Sobre la *Fetched Delivery* se podría concluir que:

- Un servidor que use *Fetched Delivery* verá sustancialmente aumentada su complejidad. No sólo tiene que enviar cuatro mensajes en vez de uno por cada *Delivery*, sino que el *Notification Producer* tiene que almacenar el *payload* hasta que el *Notification Consumer* se lo solicite.
- A cambio de esta complejidad, un servidor que implemente este tipo de envíos puede adaptarse mejor a limitaciones de ancho de banda, al no tener que satisfacer a todas las subscripciones con su *payload* de manera simultánea (por ejemplo, si 40 clientes están suscritos a cambios de un determinado vehículo, el servidor no tendrá que enviar el mensaje 'pesado' que incluye el *payload* a los 40 clientes de manera simultánea; sino que le basta con mandar 40 notificaciones, mucho más ligeras que el mensaje con el grueso de los datos).
- El cliente (*Notification Consumer*) es capaz de decidir si en un determinado momento le interesa recibir datos o no.
  Es más, si la mayoría de esas notificaciones se descartan, la carga computacional del cliente se puede reducir significativamente (aunque las condiciones para lograr tal mejora son la existencia de grandes cantidades de notificaciones, por lo que la carga reducida en el cliente se trasladará al

servidor, quien tendrá que almacenar un gran número de actualizaciones de subscripciones hasta cada uno de sus *Data Horizons*).

La *Fetched Delivery* en el caso de las subscripciones nos lleva a un esquema con una cantidad mayor de intercambios de información debido a la propia idiosincrasia del intercambio *Publish/Subscribe*:

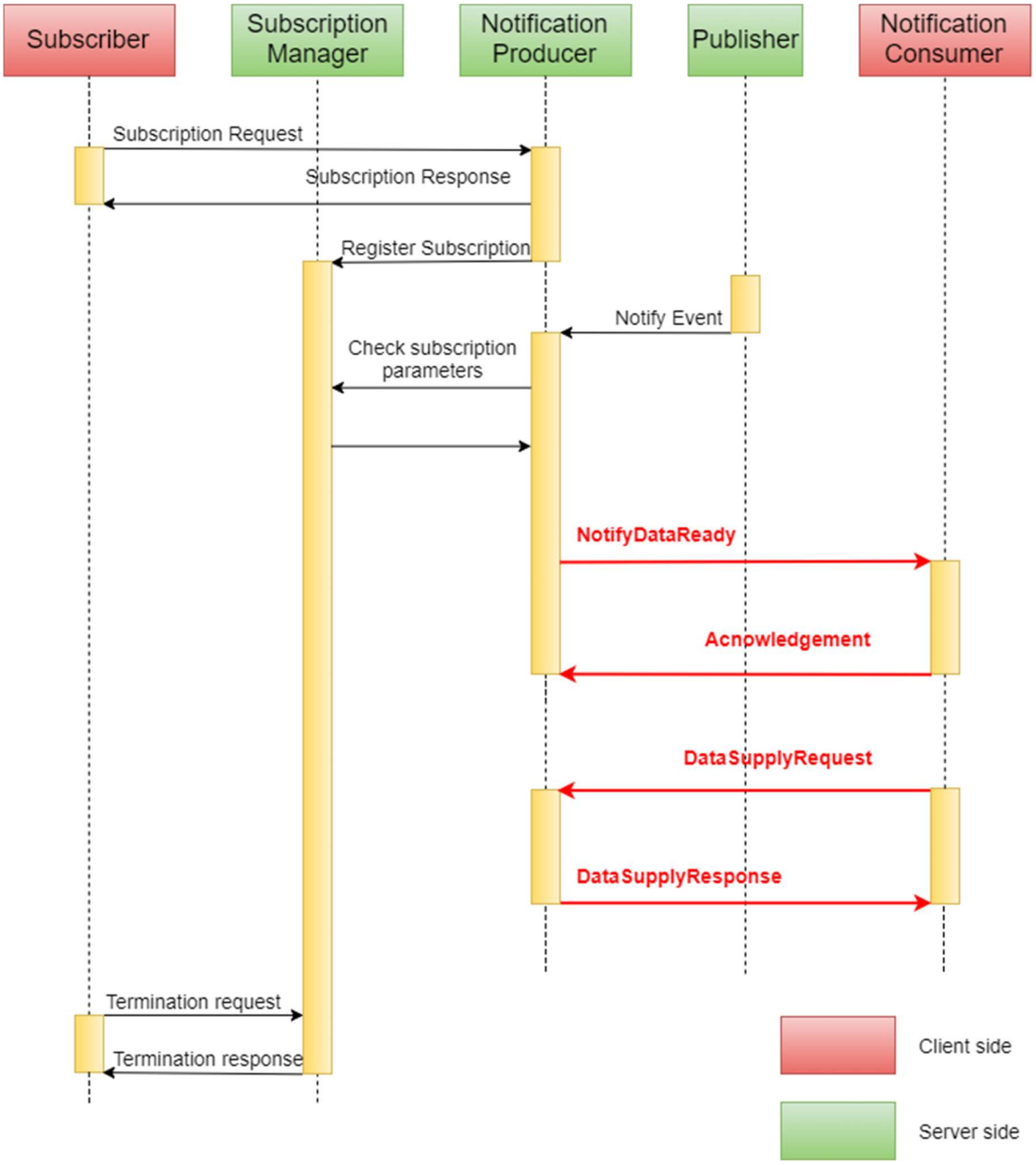


Ilustración 21: Patrón Fetched Delivery en un intercambio tipo Publish/Subscribe

La satisfacción de subscripciones en un intercambio *Publish/*Subscribe mediante un envío tipo *Fetched* o indirecto permite agrupar en el mismo envío información de distintas diferentes subscripciones de un mismo cliente. Esto a su vez abre la puerta a (excepcionalmente) separar la *DataSupplyResponse* en múltiples mensajes (como máximo, tantos como el número de subscripciones a satisfacer). Si a lo anterior sumamos la existencia de *requests* múltiples/compuestas (peticiones de subscripción al mismo, o incluso a distintos servicios) que soporta el *Publish/Subscribe*, es común que en las implementaciones reales aparezcan flujos como el siguiente:

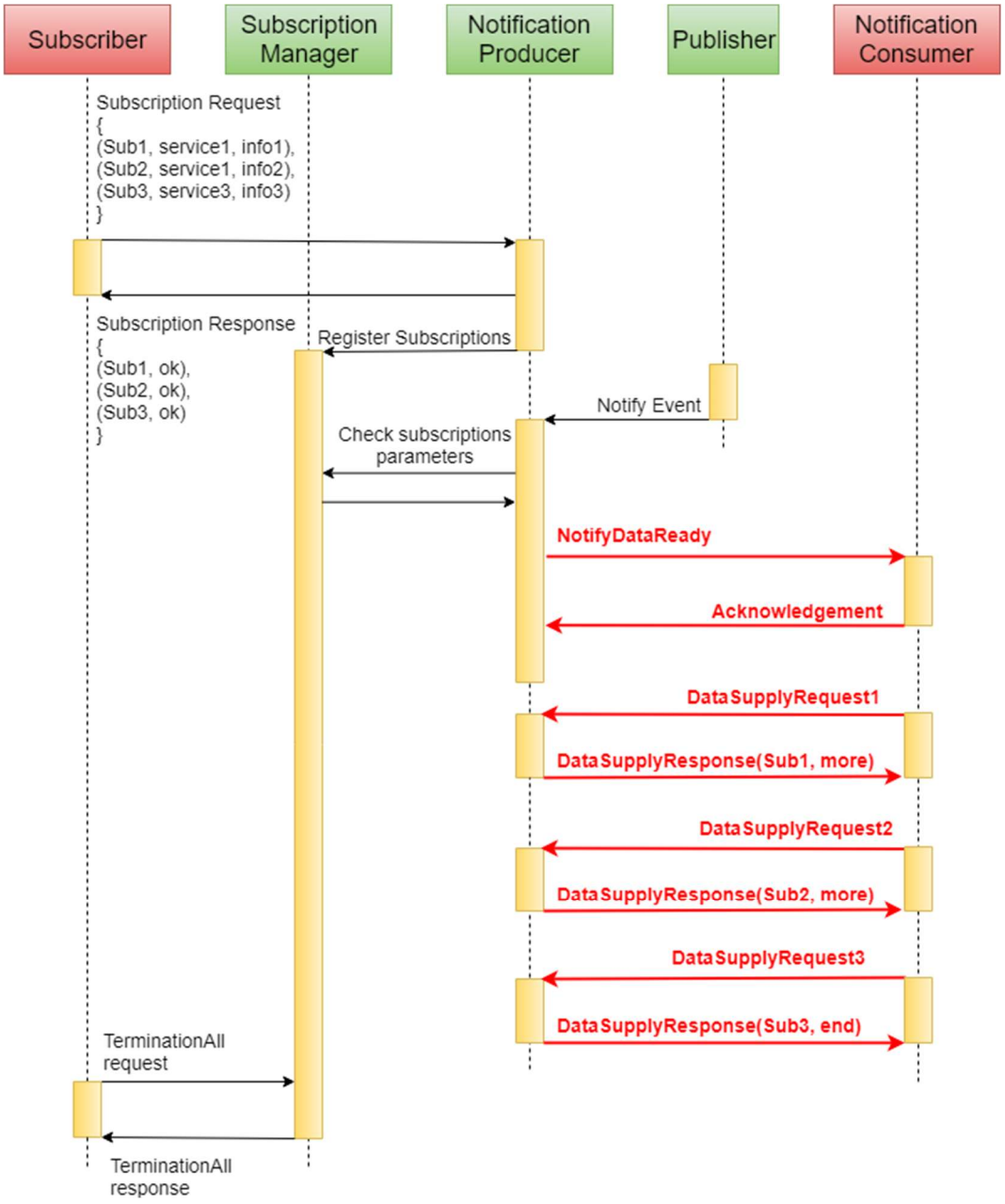


Ilustración 22: 'Multipart Dispatch' de una Fetched Delivery en un intercambio tipo Publish/Subscribe

3.2.3.SIRI *endpoints*

Se entiende por *endpoint* la dirección a la que se envía un mensaje.

De los apartados anteriores se deduce que son múltiples los *endpoints* que pueden llegar a definirse en un servicio de SIRI.

En función de dicha definición, el estándar contempla los siguientes casos:

- **Direccionamiento implícito**: en los intercambios de mensajes no se explicita *endpoint* alguno, tomándose éstos o bien de parámetros de configuración previamente definidos o bien, a la hora de responder a una *request*, de esa propia petición. Esto implica que tanto las respuestas directas (tras una *Request*) como las respuestas de satisfacción de subscripciones son enviadas a la misma dirección que las solicita.
  A pesar de que SIRI permite la deducción de los *endpoints* de este modo, está altamente desaconsejado hacer uso del mismo en aplicaciones reales, al dar pie a incontables brechas de seguridad.
- **Direccionamiento explícito**: es necesario especificar los *endpoints* hacia los que se desea enviar el *payload* de respuesta a las peticiones solicitadas (ya sea un intercambio *Request/Response* o una petición de inicio de subscripción, mandándose en este último caso la confirmación al suscriptor y el *payload*, cuando corresponda, al *Notification Consumer*).

Para cada servicio de SIRI se establecen los siguientes *endpoints* lógicos:

En el caso de un servidor:

- ***CheckStatus***: dirección que recibe las peticiones para confirmar la disponibilidad de un servicio.
- ***Subscribe***: dirección que recibe las peticiones de nuevas subscripciones.
- ***ManageSubscriptions***: dirección que recibe las peticiones para modificar o borrar subscripciones. En el caso de una petición de modificación, suele ser la misma dirección que la utilizada para suscribirse.
- ***GetData***: dirección que recibe las peticiones directas de datos (tipo *Request/Response*) y, de existir, las confirmaciones de datos recibidos.

En el caso de un cliente:

- ***ReportStatus***: dirección que recibe las confirmaciones de la disponibilidad de un servicio, así como los *Heartbeat* de los servidores.
- ***Subscriber***: dirección que recibe las respuestas a peticiones de creación/terminación de subscripciones.
- ***Notify***: dirección que recibe las notificaciones que indican la existencia de nuevos datos disponibles en el servidor.
- ***Consumer:*** dirección que recibe los datos (*payload*).

Dependiendo de la complejidad del sistema desarrollado o desplegado, una mayor o menor cantidad de estas direcciones o *endpoints* serán coincidentes.

3.3. Transporte de los mensajes

La arquitectura WS-PubSub lleva implícita la independencia del método de transporte respecto a los datos transmitidos. Esto posibilita la inclusión de varios métodos de transporte a usar para la transmisión de mensajes de SIRI, entre los que destacan:

- SOAP/WSDL.
- HTTP POST (sin SOAP).
- SIRI Lite.

3.3.1.1. SOAP/WSDL

SOAP (*Simple Object Access Protocol*) es un protocolo estándar que define la comunicación entre dos sistemas mediante el intercambio de datos XML, lo que implica una abstracción sobre protocolos como el HTTP.

Las ventajas de SOAP son: su independencia (tanto del lenguaje como de la plataforma utilizada), su neutralidad (ya que puede ser utilizado sobre protocolos como HTTP, SMTP, etc.) y su extensibilidad.

Además de lo anterior, el uso de SOAP (en tanto éste se basa intercambio de archivos XML) facilita la inclusión, junto al estándar, de documentos WSDL tanto para el cliente como para el servidor. Éstos describen cómo debe ser un servicio web que use SIRI a través de la definición de interfaces abstractas que deben ser implementadas por dicho servicio.

El estándar define tres interfaces de WSDL: *RPC Literal* (WSDL 1.1), *Document Literal* (WSDL 1.1) y una tercera interfaz para WSDL 2.0 (poco usada en la actualidad, pero incluida pensando en el largo plazo).

```
<message name="ConnectionTimetableNotify">
    <part name="ServiceDeliveryInfo" type="siri:ProducerResponseEndpointStructure"/>
    <part name="Notification" type="siri:ConnectionTimetableDeliveriesStructure"/>
    <part name="NotifyExtension" type="siri:ExtensionsStructure"/>
</message>
<message name="ConnectionMonitoringNotify">
    <part name="ServiceDeliveryInfo" type="siri:ProducerResponseEndpointStructure"/>
    <part name="Notification" type="siri:ConnectionMonitoringDeliveriesStructure"/>
    <part name="NotifyExtension" type="siri:ExtensionsStructure"/>
</message>
<message name="GeneralMessageNotify">
    <part name="ServiceDeliveryInfo" type="siri:ProducerResponseEndpointStructure"/>
    <part name="Notification" type="siri:GeneralMessageDeliveriesStructure"/>
    <part name="NotifyExtension" type="siri:ExtensionsStructure"/>
</message>
<message name="FacilityMonitoringNotify">
    <part name="ServiceDeliveryInfo" type="siri:ProducerResponseEndpointStructure"/>
    <part name="Notification" type="siri:FacilityMonitoringDeliveriesStructure"/>
    <part name="NotifyExtension" type="siri:ExtensionsStructure"/>
</message>
<message name="SituationExchangeNotify">
    <part name="ServiceDeliveryInfo" type="siri:ProducerResponseEndpointStructure"/>
    <part name="Notification" type="siri:SituationExchangeDeliveriesStructure"/>
    <part name="NotifyExtension" type="siri:ExtensionsStructure"/>
</message>
```

Ilustración 23: Ejemplo de WSDL de SIRI

Estos documentos WSDL llevan otros archivos XSD asociados que definen la estructuración de los archivos XML de los mensajes a intercambiar, de manera que cumplan con el correspondiente WSDL.

```
<!-- ==== FS: Production Timetable Subscription ======================================-->
<xsd:element name="ProductionTimetableSubscriptionRequest" substitutionGroup=
"AbstractFunctionalServiceSubscriptionRequest">
    <xsd:annotation>
        <xsd:documentation>Request for a subscription to the Production Timetable Service.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="ProductionTimetableSubscriptionStructure"/>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:element>
<xsd:complexType name="ProductionTimetableSubscriptionStructure">
    <xsd:annotation>
        <xsd:documentation>Subscription Request for Production Timetable Service.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="AbstractSubscriptionStructure">
            <xsd:sequence>
                <xsd:element ref="ProductionTimetableRequest"/>
                <xsd:group ref="ProductionTimetableSubscriptionPolicyGroup"/>
                <xsd:element ref="Extensions" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:group name="ProductionTimetableSubscriptionPolicyGroup">
```

Ilustración 24: Ejemplo de XSD de SIRI

La existencia de estos WSDL y sus XSD asociados abren la puerta a poder realizar de manera relativamente sencilla validaciones de las implementaciones de servidores y clientes de SIRI mediante la interpretación de los mensajes que éstos generan o reciben.

3.3.1.2. HTTP POST

Este método mantiene los beneficios del uso de WSDL y XSD, pero sin usar la abstracción de SOAP por encima de HTTP; realizando las peticiones directamente como HTTP POST.

3.3.1.3. SIRI Lite

A pesar de que los métodos anteriores son los más populares para el intercambio de mensajes de SIRI, el estándar también contempla el uso del llamado '*SIRI Simple Web Services*' (SIRI Lite) como protocolo alternativo para dicho intercambio.

Tal y como su nombre indica, es una versión minimalista y reducida de los anteriormente descritos que busca ganar en escalabilidad al reducir el ancho de banda utilizado, facilitar la integración de SIRI en dispositivos móviles y minimizar el desarrollo del lado cliente de las aplicaciones.

Para esto, una petición que use SIRI Lite especifica sus propios parámetros como parámetros de la petición HTTP GET correspondiente, aceptando como respuesta JSON, XML, etc.

Vemos una comparativa entre los citados métodos de transporte:

Petición de *Stop Monitoring* tipo POST:

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
    xmlns="http://www.siri.org.uk/siri"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
    xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
    <ServiceRequest>
        <ServiceRequestContext>
            <Language>en</Language>
            <DataHorizon>P1Y2M3DT10H30M</DataHorizon>
            <RequestTimeout>P1Y2M3DT10H30M</RequestTimeout>
            <DeliveryMethod>direct</DeliveryMethod>
            <MultipartDespatch>true</MultipartDespatch>
            <ConfirmDelivery>false</ConfirmDelivery>
        </ServiceRequestContext>
        <RequestTimestamp>2004-12-17T09:30:47-05:00</RequestTimestamp>
        <RequestorRef>NADER</RequestorRef>
        <StopMonitoringRequest version="2.0">
            <RequestTimestamp>2004-12-17T09:30:47-05:00</RequestTimestamp>
            <PreviewInterval>P10M</PreviewInterval>
            <MonitoringRef>EH00001</MonitoringRef>
            <OperatorRef>OPERATOR22</OperatorRef>
            <LineRef>LINE77</LineRef>
            <DirectionRef>OUTBOUND</DirectionRef>
            <DestinationRef>PLACE98765</DestinationRef>
            <StopVisitTypes>all</StopVisitTypes>
            <MaximumStopVisits>7</MaximumStopVisits>
            <MinimumStopVisitsPerLine>2</MinimumStopVisitsPerLine>
            <MaximumTextLength>20</MaximumTextLength>
            <StopMonitoringDetailLevel>calls</StopMonitoringDetailLevel>
        </StopMonitoringRequest>
    </ServiceRequest>
</Siri>
```

Ilustración 25: Ejemplo de petición de Stop Monitoring usando HTTP POST

La misma petición *Stop Monitoring* vía HTTP GET (SIRI Lite):

```
http://www.foobar.com/siri/siri?RequestorRef=NADER
  &Request=StopMonitoringRequest
  &version=1.0
  &PreviewInterval=P10M
  &MonitoringRef=EH00001
  &OperatorRef=OPERATOR22
  &LineRef=LINE77
  &DirectionRef=OUTBOUND
  &DestinationRef=PLACE98765
```

Ilustración 26: Ejemplo de petición de Stop Monitoring usando HTTP GET (SIRI Lite)

Indudablemente la petición realizada usando SIRI Lite tiene menor complejidad y tamaño que una petición POST.

Sin embargo, si tenemos en cuenta el propósito para el cual SIRI fue creado (véase: comunicación de trazas de datos completas de datos de transporte público entre servidores de aplicaciones), es fácil darse cuenta de que SIRI Lite no permite la complejidad ni vela por la integridad de los datos que se transmiten. Esto queda especialmente patente en el caso de las respuestas en formato JSON que admite SIRI Lite, difícilmente validables al no poder ser contrastadas contra los XSD que el estándar nos proporciona.

No obstante, lejos de ser marginal, el uso de SIRI Lite está ganando en popularidad en los últimos años. En el apartado correspondiente a la captura de datos reales con el cliente implementado (4.3.6) se incluyen muestras más detalladas del uso de este método de transporte.

Para completar los ejemplos expuestos, veamos cómo queda una petición similar a la anterior (servicio de *Stop Monitoring*) pero usando SOAP como método de transporte:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<S:Envelope
    xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <S:Body>
        <ns6:GetStopMonitoring
            xmlns:ns2="http://www.siri.org.uk/siri"
            xmlns:ns3="http://www.ifopt.org.uk/acsb"
            xmlns:ns4="http://www.ifopt.org.uk/ifopt"
            xmlns:ns5="http://datex2.eu/schema/2_0RC1/2_0"
            xmlns:ns6="http://wsdl.siri.org.uk">
            <ServiceRequestInfo>
                <ns2:RequestTimestamp>2017-12-28T12:19:24.819+01:00</ns2:RequestTimestamp>
                <ns2:RequestorRef>NINOXE:default</ns2:RequestorRef>
                <ns2:MessageIdentifier>StopMonitoring:Test:2</ns2:MessageIdentifier>
            </ServiceRequestInfo>
            <Request version="2.0">
                <ns2:RequestTimestamp>2017-12-28T12:19:24.819+01:00</ns2:RequestTimestamp>
                <ns2:MessageIdentifier>StopMonitoring:Test:2</ns2:MessageIdentifier>
                <ns2:MonitoringRef>206</ns2:MonitoringRef>
            </Request>
            <RequestExtension/>
        </ns6:GetStopMonitoring>
    </S:Body>
</S:Envelope>
```

Ilustración 27: Ejemplo de petición de Stop Monitoring usando SOAP

Nótese la existencia del apartado *Envelope*, característico de SOAP y por tanto ausente en el ejemplo de HTTP POST.

Asimismo, la estructura del mensaje también difiere ligeramente de ese otro método, aunque esto es algo fácilmente salvable durante la implementación si se requiere esa compatibilidad.

3.4. Servicios de SIRI

El estándar SIRI ofrece una serie de servicios, todos ellos opcionales e independientes entre sí, que permiten el **intercambio de información de transporte público de acuerdo con el modelo que proporciona Transmodel**.

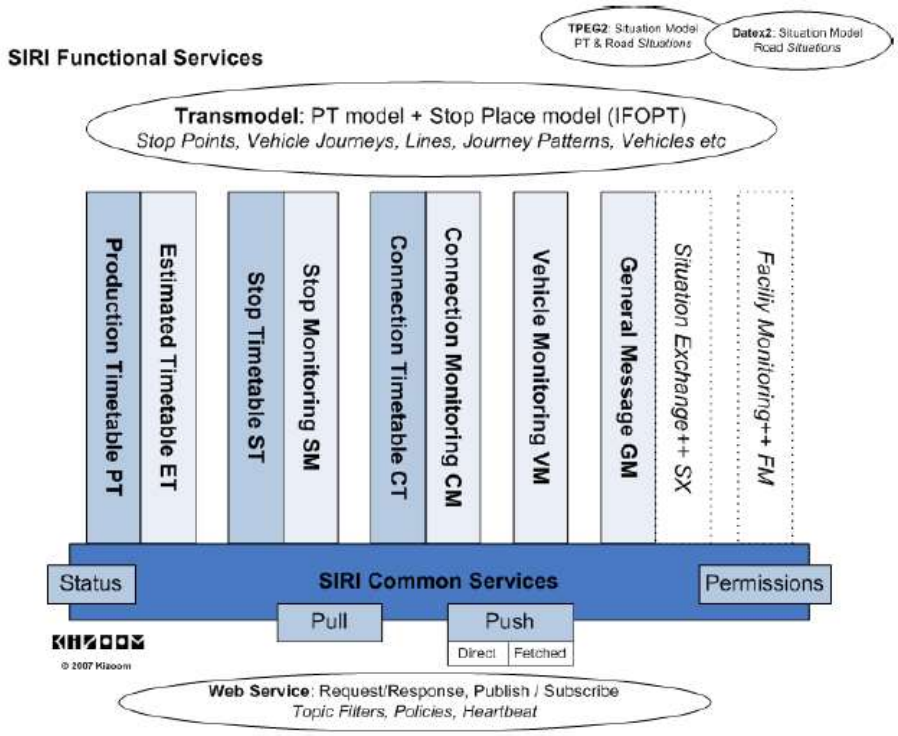


Ilustración 28: Servicios funcionales de SIRI (SIRI HandBook v0.13 [16])

Cabe destacar la agrupación por parejas de varios de ellos (PT y ET, ST y SM, CT y CM). Esto es debido a que ambos dan el mismo tipo de información, proporcionándola uno de ellos en tiempo real.

La mayor parte de estos servicios están descritos en CEN/TS 15531-3: *Part 3. Functional Service Interfaces,* siendo añadidos posteriormente los servicios *Facility Monitoring* (CEN/TS 15531-4) y *Situation Exchange* (CEN/TS 15531-5).

Para la correcta comprensión de la descripción de cada uno de los servicios que se pasan a detallar en los siguientes subapartados conviene poseer unas nociones básicas sobre Transmodel y las entidades que en ese estándar se definen. En el apartado final del segundo capítulo se hace referencia a aquellas que se ha considerado pueden ser más relevantes.

Pasamos a describirlos, dando una definición general de mismo y viendo la respuesta y petición que los caracterizan (en ese orden, pues la respuesta permitirá la mejor comprensión de los datos o filtros que pueden incluirse en la correspondiente petición que a ésta precede).

En el anexo 1 se incluyen ejemplos del XML generado por peticiones y respuestas, que complementan a estas descripciones.

3.4.1. *Production Timetable* (PT)

El servicio *Production Timetable* fue diseñado para transmitir información sobre viajes (también llamados *trips* o *journeys*), más concretamente, sobre la **planificación horaria de viajes**; debiendo ser ésta actualizada con las alteraciones en dicha planificación que se conozcan en el momento de la retransmisión del mensaje.

Esto último significa que, aunque PT no es un servicio de tiempo real (una alternación en los horarios no implica necesariamente una actualización de PT), cada vez que el servicio de *Production Timetable* es requerido, la respuesta de un servidor de SIRI debe incluir el horario actualizado con las últimas alteraciones disponibles. Generalmente, un mensaje PT incluye los viajes planificados para las veinticuatro horas inmediatamente posteriores al momento del envío del mensaje; aunque esto es mera convención que puede ser modificada fácilmente, como después veremos.

La estructura de un mensaje de PT difiere del patrón estructural seguido por la mayor parte del resto de servicios, caracterizándose por estar muy orientada a los consumidores finales de estos datos de horarios: los futuros pasajeros del transporte público.

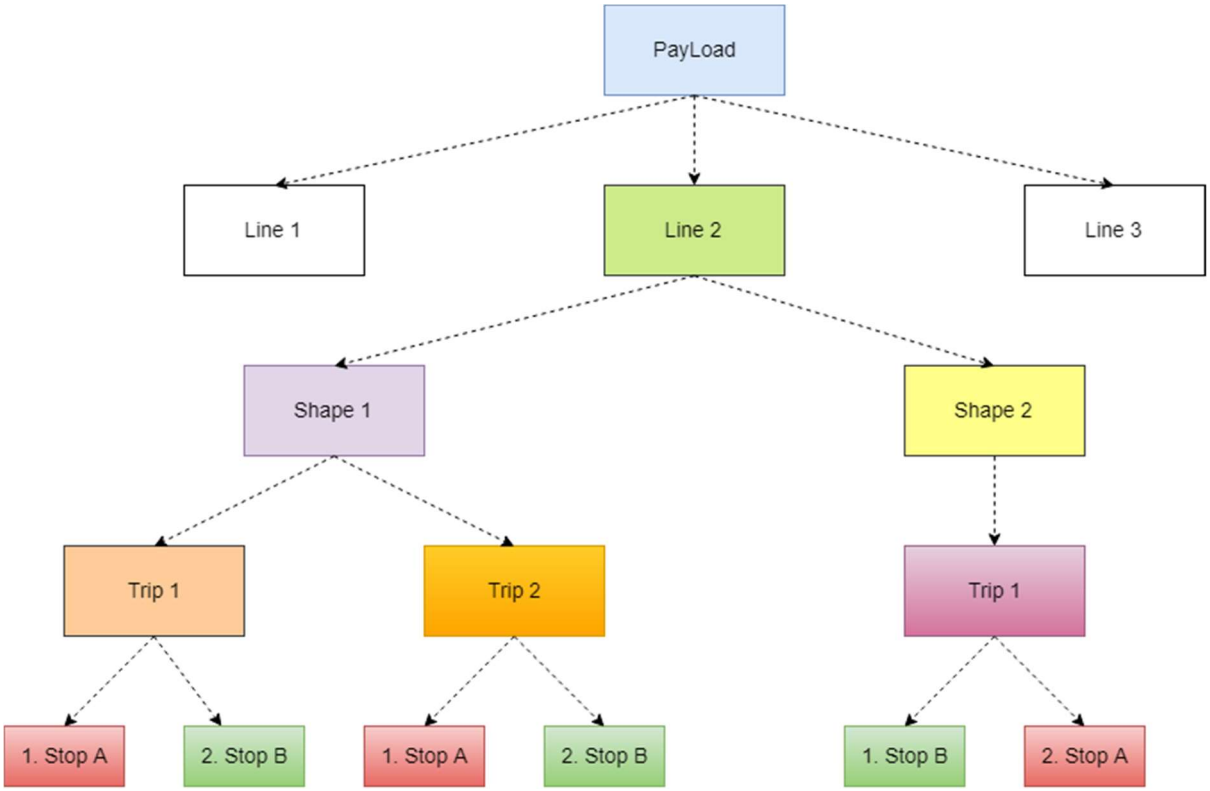


Ilustración 29: Estructura de una respuesta de Production Timetable

Una respuesta de *Production Timetable* se organiza jerárquicamente en:

- **Vector de** estructuras tipo ***ProductionTimetableDeliveryStructure***, cada una correspondiéndose con el *Payload* del diagrama,
- **Líneas** (ver Transmodel: *Route*).
  Permiten definir el nombre y código de la línea correspondiente, la dirección del viaje, su origen y destino, si está monitorizado, etc.
- **Trayectos** (ver Transmodel: *Journey Pattern*), comúnmente conocidos como *Shapes.* Comprenden una lista ordenada y única de paradas, con un inicio y un fin, por la que transcurre un viaje de transporte público.
  A este nivel se definen datos como el código del trayecto y si se trata de un viaje de frecuencia (*Headway*) o no.
- **Viajes** (ver Transmodel: *Vehicle Journey* y *Headway Journey*).
  A este nivel se definen, por ejemplo, las características del viaje (código, nombre, etc.) y el vehículo en el que se realiza el mismo.
- **Paradas** (ver Transmodel: *Stop*). Todos los viajes de un mismo trayecto contienen las mismas paradas en el mismo orden (debido a la propia definición de trayecto). A este nivel PT permite definir, para cada elemento, entre otros:
  - La referencia y nombre de la parada.
  - La hora de llegada y salida del vehículo estimadas (en caso de *Vehicle Journey*) o el intervalo de paso por parada estimado (en caso de que se trate de un *Headway Journey*).

Teniendo en cuenta lo anterior, una petición (bien sencilla o bien petición de subscripción) de *Production Timetable* puede configurar o filtrar los viajes de los que quiere recibir información de los siguientes modos:

- La hora de inicio y la hora de fin (el llamado *Validity Period*) entre las cuales deben encontrarse los viajes que se desea sean incluidos en la *response*. De no configurarse, se toma por defecto el *Data Horizon* como hora final, y la hora de la *request* como hora inicial. Como se ha mencionado anteriormente, lo habitual es que este período comprenda las veinticuatro horas posteriores al momento de petición del servicio.
- La línea o líneas a las que deben pertenecer los viajes que se devuelvan.
  Esto se puede concretar aún más, eligiéndose la dirección de los viajes, además de su línea.

Aunque SIRI ofrece la posibilidad de subscripción al servicio de *Production Timetable*, es más común usarlo mediante *Request/Response* (petición inicial) para subscribirse a su análogo de tiempo real (que veremos a continuación). Alternativamente, las suscripciones a este servicio pueden ser utilizadas para ofrecer cara al público un horario actualizado con una cierta frecuencia, algo menos habitual que la anterior alternativa, pero también común. Esta frecuencia debe ser fija, o al menos independiente de los cambios que puedan tener lugar en las expediciones que en el horario se muestran.

3.4.2. *Estimated Timetable* (ET)

La función del servicio *Estimated Timetable* es transmitir **información horaria sobre viajes actualizada en tiempo real** (frente a *Production Timetable*, que como hemos visto se encarga de transmitir el mismo tipo de datos, pero en su vertiente planificada).

Adicionalmente, ET puede ser usado en combinación con *Stop Monitoring* para dar previsiones en tiempo real de los viajes basadas en los retrasos en los pasos de los vehículos que los realizan por las distintas paradas.

Aunque este servicio esté disponible tanto por *Request/Response* como a través de subscripción, lo más común es usarlo de esta segunda manera, acompañado de los filtros correspondientes para no saturar al cliente (*Data Consumer*).

La estructura de datos de una *response* de ET es casi idéntica a la de PT:

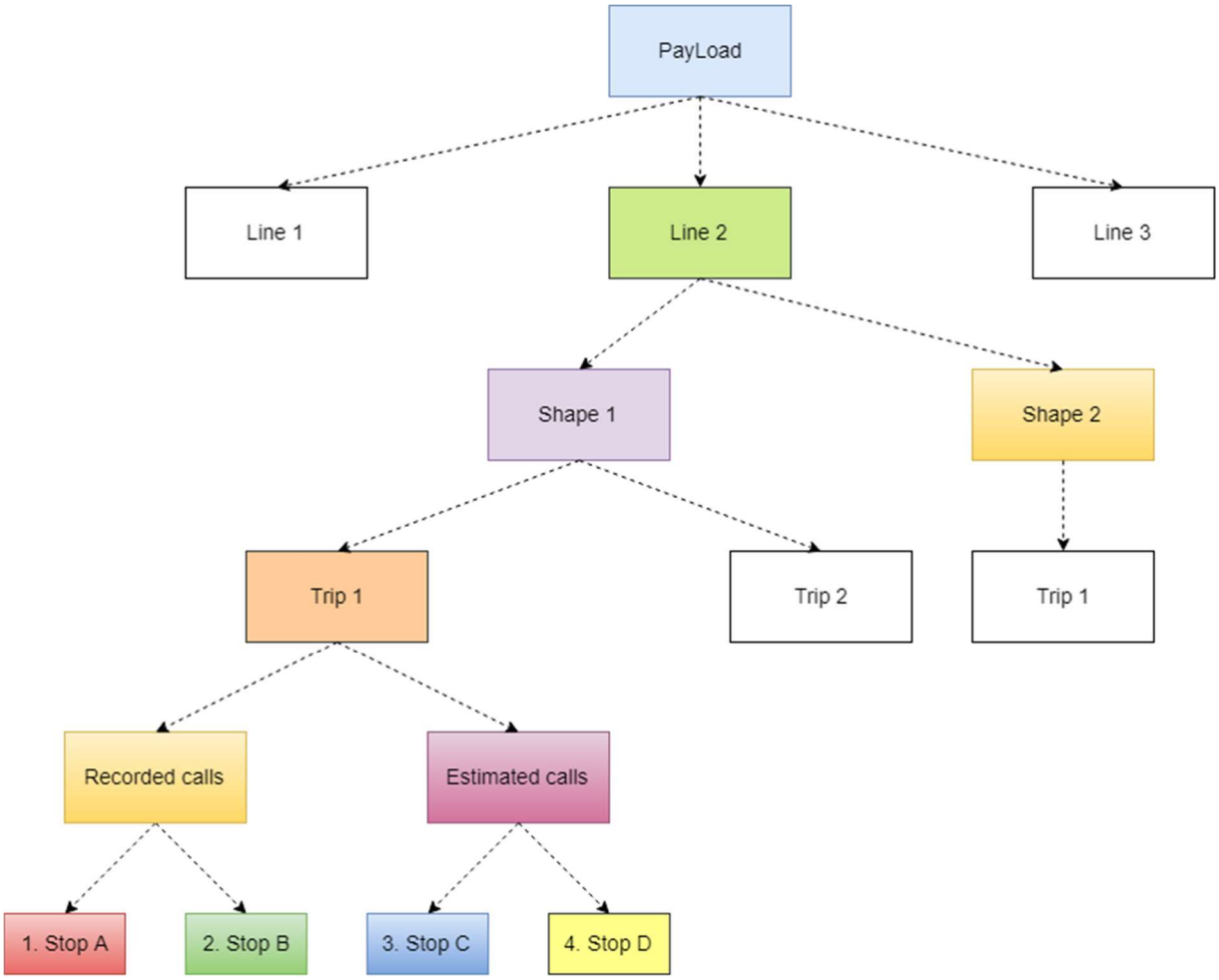


Ilustración 30: Estructura de una respuesta de Estimated Timetable

La principal diferencia, reflejada en el esquema, la podemos encontrar en la existencia del campo '*RecordedCalls'*, pensado para describir el tiempo de llegada y salida real de una parada o (en el caso de viajes de tipo frecuencia, o *Headway*) el intervalo real desde el último paso por esa parada.

Por tanto, en un mensaje de *Estimated Timetable* los viajes (*Trips*) pueden contener dos listas de paradas:

- Las llamadas *EstimatedCalls*, que contienen previsiones sobre el tiempo de paso por parada / intervalo entre paradas.
- Las *RecordedCalls*, que contienen datos reales, ya que el vehículo que está realizando el viaje correspondiente ya ha pasado por ellas.

Los datos de tiempo real de un viaje comenzado y no finalizado en el momento del envío de un mensaje de ET que lo monitoriza incluirán ambos tipos de paradas.

Análogamente al caso de Production Timetable, en una petición de ET (bien directa o bien petición de subscripción) pueden concretarse:

- La hora de inicio y la hora de fin (el llamado *Validity Period*) entre las cuales deben encontrarse los viajes que se desea sean incluidos en la *response*. De no configurarse, se toma por defecto el *Data Horizon* como hora final, y la hora de la *request* como hora inicial. Como en el caso de *Production Timetable*, lo habitual es que este período comprenda las veinticuatro horas posteriores al momento de petición del servicio.
  Si se deseara, podría configurarse este *Validity Period* de manera que los mensajes de ET sólo contuvieran un tipo de paradas: o bien *RecordedCalls*, para poder almacenar los datos históricos de pasos reales por parada; o bien *EstimatedCalls*, para proporcionar información a los usuarios finales sobre las estimaciones de pasos por parada (entiéndase usuarios finales en el amplio sentido del término: viajeros, conductor, controladores del sistema, etc.).
- La línea/líneas a las que deben pertenecer los viajes que se devuelvan. Esto se puede concretar aún más, eligiéndose la dirección de los viajes, además de su línea.

Estimated Timetable es uno de los servicios más complejos que ofrece SIRI, así como uno de los más importantes. De ser implementado correctamente en el servidor, es capaz de proporcionar al cliente una gran cantidad de información que, una vez filtrada, puede servir para propósitos como almacenar el registro del histórico de todos los viajes realizados para posteriormente analizarlos en búsqueda de posibles optimizaciones en el trayecto/frecuencia de los mismos, proporcionar información a un pasajero sobre su viaje en concreto, etc.

3.4.3.*Stop Timetable* (ST)

El servicio *Stop Timetable* es el encargado de **transmitir información sobre la llegada y salida de vehículos a las distintas paradas, desde el punto de vista de estas últimas.**

El correcto uso de este servicio requiere de un acuerdo previo entre las distintas partes que intervienen en la comunicación sobre el significado de:

- *Point* (ver Transmodel: *Scheduled Stop Point* o *Place*), el punto en el que los pasajeros se incorporan/abandonan el vehículo correspondiente.
- *Line* (ver Transmodel: *Route*), lo que en castellano conocemos por 'línea'.
- *Direction* (ver Transmodel: *Direction*), el sentido que lleva una línea (valor booleano).

La estructura de una respuesta de ST es la siguiente:

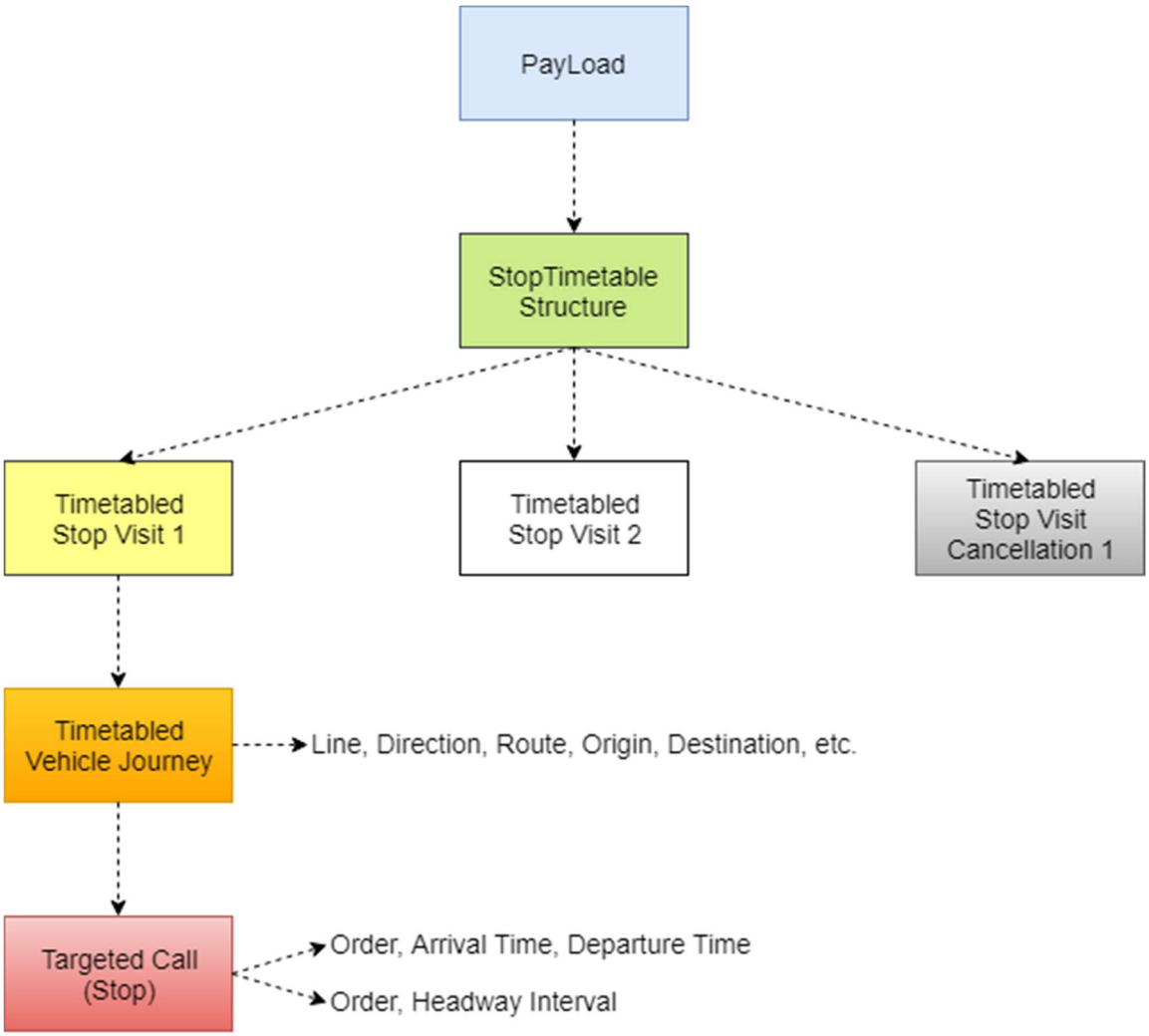


Ilustración 31: Estructura de una respuesta de Stop Timetable

La jerarquía de los mensajes de *Stop Timetable* es la siguiente:

- **Estructura** tipo ***StopTimetableDeliveryStructure***, que mantiene la concordancia con el resto de servicios de SIRI.
- **Visitas planificadas a paradas**. Por cada parada de la que proceda enviar información, existirá uno de estos elementos para agrupar dichos datos.
- **Cancelaciones de visitas planificadas a paradas.** *Stop Timetable* también permite incluir explícitamente las cancelaciones respecto a planificaciones previas, evitando tener que deducirlas de la ausencia de algún elemento en el vector de visitas planificadas.
- **Viaje planificado** (ver Transmodel: *Vehicle Journey* y *Headway Journey*). Por cada visita planificada, esta estructura describe el viaje en el que está planificada esa visita, abarcando campos como la línea, la dirección, el trayecto, el origen, el destino, si se trata de un viaje de frecuencia (*Headway*) o no, etc.
- **Parada** (ver Transmodel: *Stop*). A este nivel se definen los datos propios de la planificación relativos a la parada: el orden dentro del trayecto y, o bien el tiempo de llegada y salida del vehículo, o bien el intervalo de paso por parada estimado (en caso de viaje tipo *Headway*).

Ya que no es habitual desear información detallada de todas las paradas, y teniendo en cuenta que este servicio está construido en torno a éstas, las peticiones (tanto las simples como las peticiones de subscripción) incluyen los siguientes filtros:

- La hora de inicio y la hora de fin (período denominado, en este caso, *Departure Window*) entre las cuales deben encontrarse los viajes cuyos pasos por parada se desea sean incluidos en la *response*. De no configurarse, se toma por defecto el *Data Horizon* como hora final y la hora de la petición como hora inicial.
- La línea a la que deben pertenecer esos viajes y/o su dirección.
- El código de la parada de la que se desea información.
  A pesar de que se pueden hacer peticiones que no incluyan este código de parada, la existencia de una sola estructura tipo *StopTimetableDeliveryStructure* (en contraposición con, como veremos después, SM) ya nos indica que este servicio está preparado para devolver información de una sola parada. No es difícil conseguir este envío de múltiples paradas (nuestra implementación del servidor lo podría permitir, sin ir más lejos, si así se deseara), pero la información que podría llegar a devolver un servidor en ausencia de fuertes restricciones en el resto de los filtros es inmensa. Por tanto, lo habitual es no permitir a los clientes realizar peticiones que carezcan de este parámetro.

Al igual que sucedía con el servicio de *Production Timetable*, *Stop Timetable* permite subscripciones; a pesar de no ser éste su modo de uso más habitual.

3.4.4. *Stop Monitoring* (SM)

El servicio equivalente a *Stop Timetable* para la **monitorización en tiempo real de las paradas** es *Stop Monitoring*. Además de proporcionar información en tiempo real de los pasos previstos por parada y de los ya acontecidos, SM aporta una gran cantidad de datos extra sobre los **viajes** que recalarán en cada una de las paradas, resultando esto útil tanto para el usuario final como para las aplicaciones de *Backend* de los operadores (para, por ejemplo, recalcular las estimaciones de pasos por paradas si el centro de datos de destino contiene más información sobre los viajes que el emisor del mensaje de SM).

Éste es el servicio ideal para proporcionar información a los paneles de las paradas, a las aplicaciones móviles del usuario cuando éste requiere información sobre los buses que pasan por la parada en la que se encuentra, etc.

La estructura, simplificada, de una respuesta de SM es la siguiente:

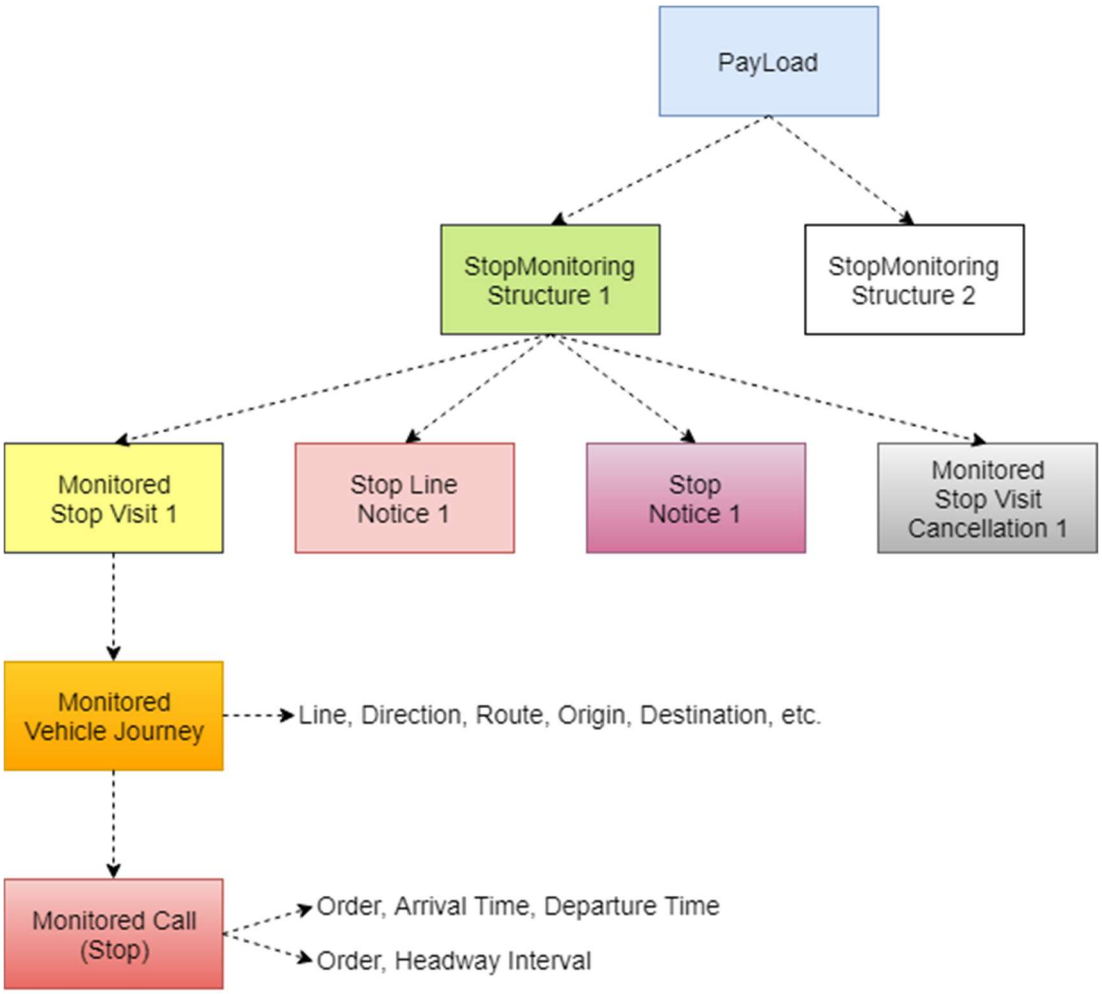


Ilustración 32: Estructura de una respuesta de Stop Monitoring

La estructura jerárquica de una respuesta de *Stop Monitoring* es, como se ha mencionado, más compleja y completa que la de *Stop Timetable*:

- **Vector de** estructuras tipo ***StopMonitoringDeliveryStructure***, que además de mantener la concordancia con el resto de servicios de SIRI, permite el llamado *Multiple Stop Monitoring Delivery*, que abarca información sobre más de una parada.
- **Novedades sobre líneas**. Por cada parada de la que proceda enviar información, un mensaje de SM podrá tener noticias o información novedosa sobre la línea de cada uno de los viajes que vayan a pasar por ella (complementando así la propia información monitorizada, por ejemplo, un gran retraso en los próximos pasos por parada).
- **Novedades sobre las propias paradas**. Asimismo, un mensaje de SM podrá contener información novedosa o noticias sobre la propia parada de la que está informando (por ejemplo, el desplazamiento del *Scheduled Stop Point* de la misma, o lo que es lo mismo, el lugar en el que se recogerá al pasajero).
- **Visitas monitorizadas a paradas**. Por cada parada de la que proceda enviar información existirá uno de estos elementos para agrupar dichos datos.
- **Cancelaciones de visitas monitorizadas a paradas.** *Stop Monitoring* también permite incluir explícitamente las cancelaciones respecto a planificaciones previas, evitando tener que deducirlas de la ausencia de algún elemento en el vector de visitas monitorizadas.
- **Viaje monitorizado** (ver Transmodel: *Vehicle Journey* y *Headway Journey*). Por cada visita monitorizada, esta estructura describe el viaje en el que está planificada esa visita:
  - Por un lado, esto abarca los campos que ya se incluían en *Stop Timetable*: la línea, la dirección, el trayecto, el origen, el destino, si se trata de un viaje de frecuencia (*Headway*) o no, etc.
  - Adicionalmente, da información en tiempo real sobre el propio viaje, como la velocidad, la orientación, si el motor está en marcha, el vehículo que lo está llevando a cabo, el retraso (*delay*) del propio viaje, etc.
- **Parada** (ver Transmodel: *Stop*). A este nivel se definen los datos propios de la planificación relativos a la parada: el orden dentro del trayecto y, o bien el tiempo de llegada y salida del vehículo, o bien el intervalo de paso por parada estimado (en caso de viaje tipo *Headway*).

Las peticiones de *Stop Monitoring* incluyen los mismos filtros que las de *Stop Timetable*, con la salvedad de permitir el ya mencionado *Multiple Stop Monitoring Delivery*; lo que se consigue permitiendo varios códigos de paradas en el correspondiente filtro.

3.4.5. *Vehicle Monitoring* (VM)

La función del servicio *Vehicle Monitoring* es **informar de la posición de un vehículo o grupo de vehículos en tiempo real**, consiguiendo con ello viajes monitorizados. La información que proporciona es complementaria a la de *Estimated Timetable* y, sobre todo, *Stop Monitoring*; resultando especialmente útil para ser intercambiada entre centros de control.

Este servicio es el indicado para proporcionar información visual sobre el movimiento del vehículo en mapas, sinópticos y otros diagramas similares.

La estructura típica de una respuesta de *Vehicle Monitoring* es la siguiente:

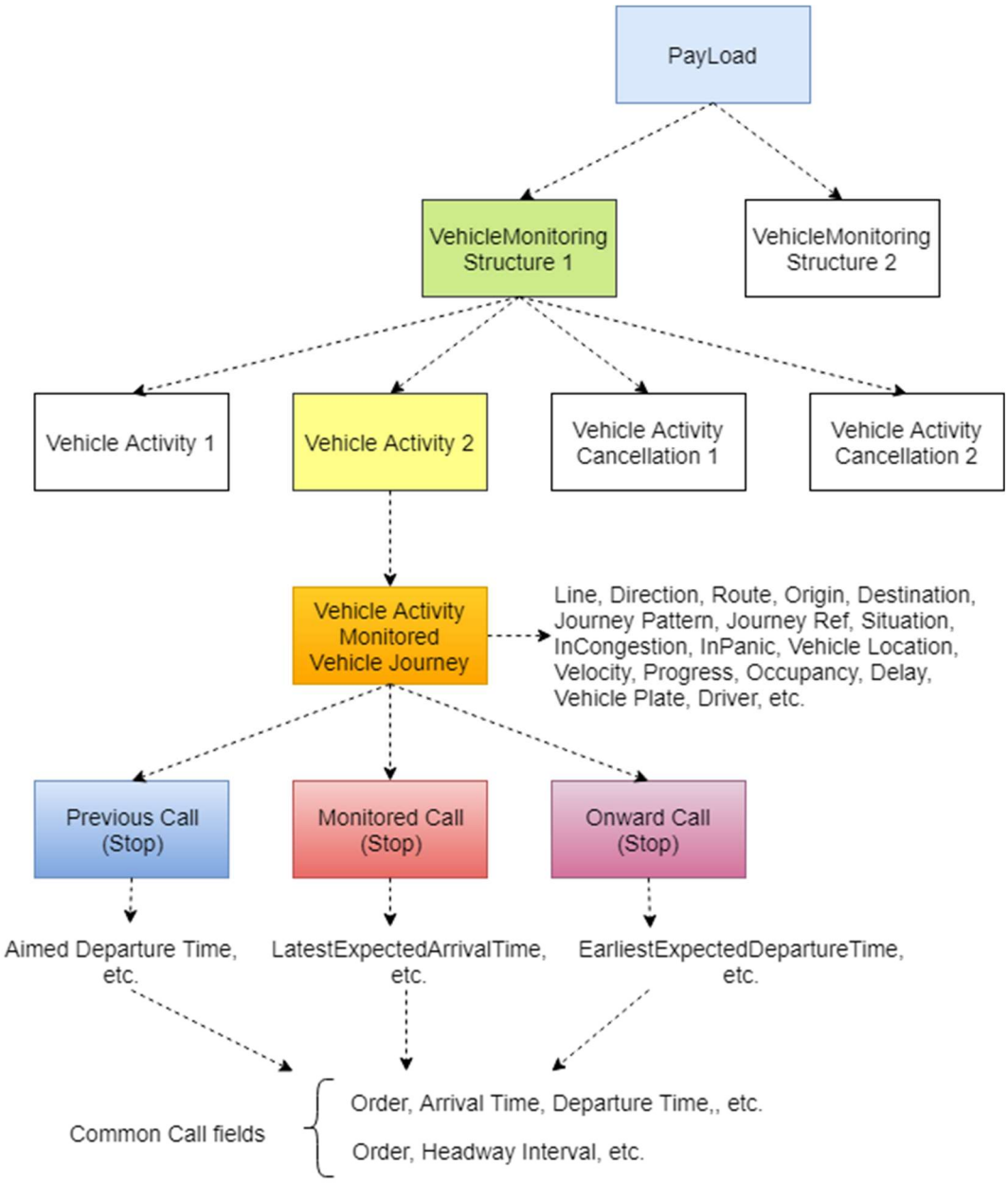


Ilustración 33: Estructura de una respuesta de Vehicle Monitoring

La similitud entre la estructura de los mensajes de *Vehicle Monitoring* y las de *Stop Timetable* y *Stop Monitoring* (especialmente de esta última, al compartir con VM el hecho de estar enfocada al tiempo real) es fácilmente observable.

- **Vector de** estructuras tipo ***VehicleMonitoringDeliveryStructure***, que soportan el envío de varios *payloads* en un mismo mensaje de VM. Un ejemplo de este tipo de casos es la posible agrupación de mensajes en las subscripciones. Es importante destacar que, en caso de enviar información sobre los viajes de varios vehículos en una sola respuesta simple de VM, se usará sólo una de estas estructuras.
- **Vector de** estructuras tipo ***VehicleActivity***, preparadas (estas sí) para almacenar los datos de múltiples viajes (ver Transmodel: *Vehicle Journey*) realizados por otros tantos vehículos, en caso de que sea pertinente el envío de información relativa a varios de ellos.
  La información de cada viaje contempla las paradas ya completadas por el viaje (*RecordedCalls*), con los tiempos reales del mismo; las paradas del viaje actual, sobre las que se informa en tiempo real; y las paradas futuras o planificadas, sobre las que se informa en base a información planificada, al no disponerse de información en tiempo real.
- **Vector de** estructuras tipo ***VehicleActivityCancellation***. Permiten incluir explícitamente las cancelaciones de viajes sobre los que se ha informado previamente, evitando el tener que deducir una cancelación de la ausencia de información sobre ese viaje en el vector de *VehicleActivity*.

Como cabe esperar una vez familiarizados con los servicios anteriores, las peticiones de *Vehicle Monitoring* incluyen los siguientes filtros:

- La línea a la que deben pertenecer los viajes que están realizando los vehículos cuya información se desea, y/o su dirección.
- El vehículo concreto sobre el cual se desea información.
  En cada petición sólo debe ser especificado un vehículo, aunque como ya hemos mencionado, en una misma respuesta puedan agruparse información sobre subscripciones a varios vehículos (que habrán sido solicitadas individualmente).

El servicio de *Vehicle Monitoring* se caracteriza por su flexibilidad y por la gran cantidad de información que aporta (más allá de la posición del vehículo y su próximo paso por parada), pudiendo ser considerado el servicio fundamental de SIRI para la transmisión de datos en tiempo real.

Al igual que los otros servicios de tiempo real ya descritos, *Stop Monitoring* y *Estimated Timetable*, este servicio suele ser usado tanto en la modalidad *Request/Response* como en la modalidad de subscripción, prevaleciendo esta última.

3.4.6. *Connection Timetable* (CT) y *Connection Monitoring* (CM)

Los servicios *Connection Timetable* y *Connection Monitoring* surgieron con el propósito de permitir **intercambiar información entre distintos operadores de transporte público sobre una zona geográfica determinada** (un *Connection Point* de Transmodel, un **punto de intercambio o transbordo**). En el caso de *Connection Timetable* (CT), esta información intercambiada es planificada; mientras que el servicio de *Connection Monitoring* (CM) se encarga de la información en tiempo real.

Estos servicios permitirían al operador del tren de un viajero comunicarle a éste que un autobús bus de otro operador le estará esperando para recogerle cuando llegue a la estación; o a un operador de autobuses interurbanos informar al pasajero de los buses urbanos que puede coger cuando llegue a su destino. CT permitiría al viajero consultarlo al reservar el viaje y CM le proporcionaría información en tiempo real sobre esas conexiones.

La estructura de una respuesta de *Connection Timetable* es la siguiente:

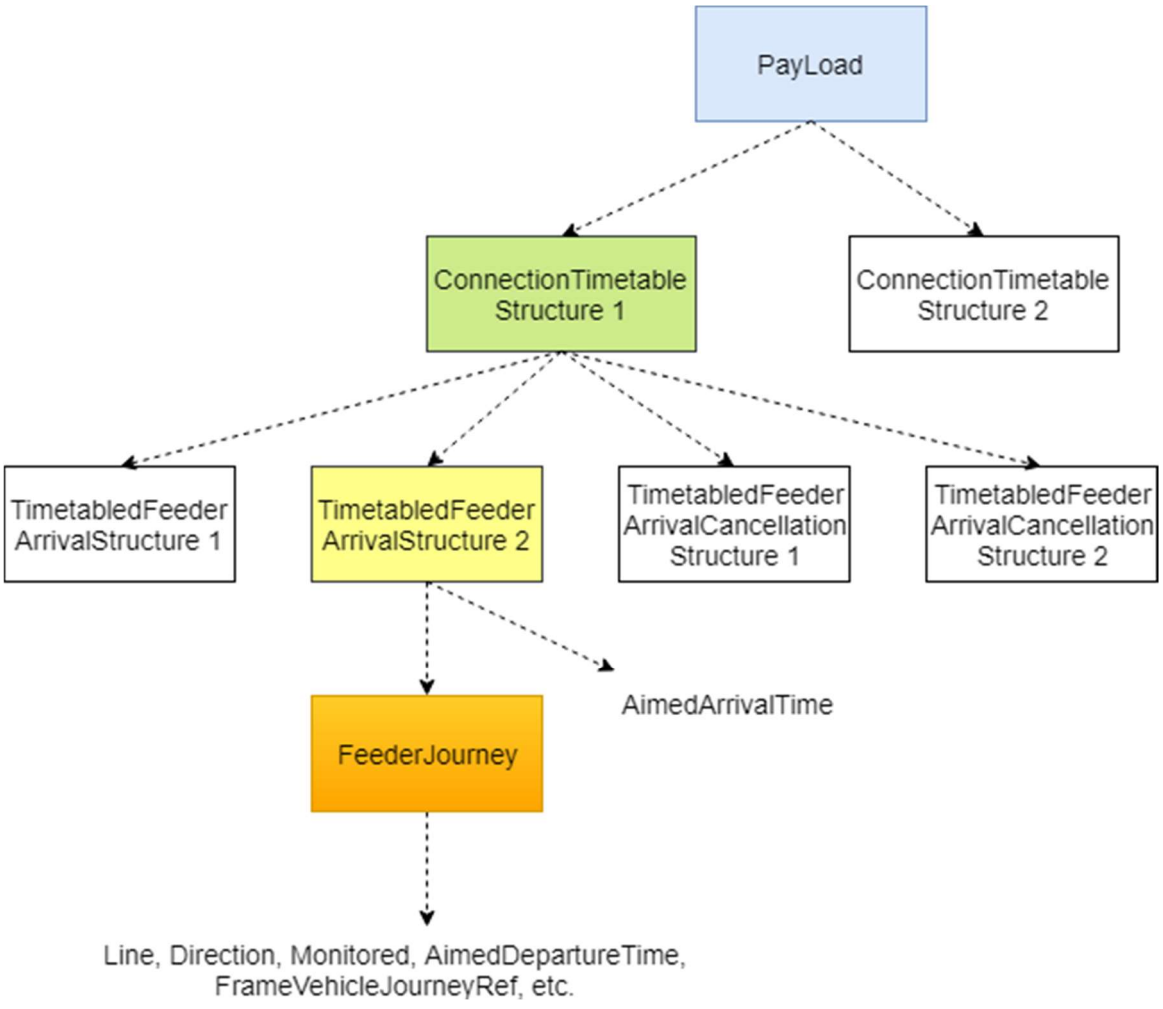


Ilustración 34: Estructura de una respuesta de Connection Timetable

Cada estructura ***ConnectionTimetableStructure*** del vector de estructuras que compone la carga del mensaje tiene a su vez dos vectores de dos tipos distintos de estructuras: ***TimetabledFeederArrivalStructure*** (viajes planificados) y ***TimetabledFeederArrivalCancellationStructure*** (viajes cancelados), incluyendo el primer tipo otra estructura tipo ***FeederJourney*** que describe el viaje.

En el caso de *Connection Monitoring*, la estructura de una respuesta es:

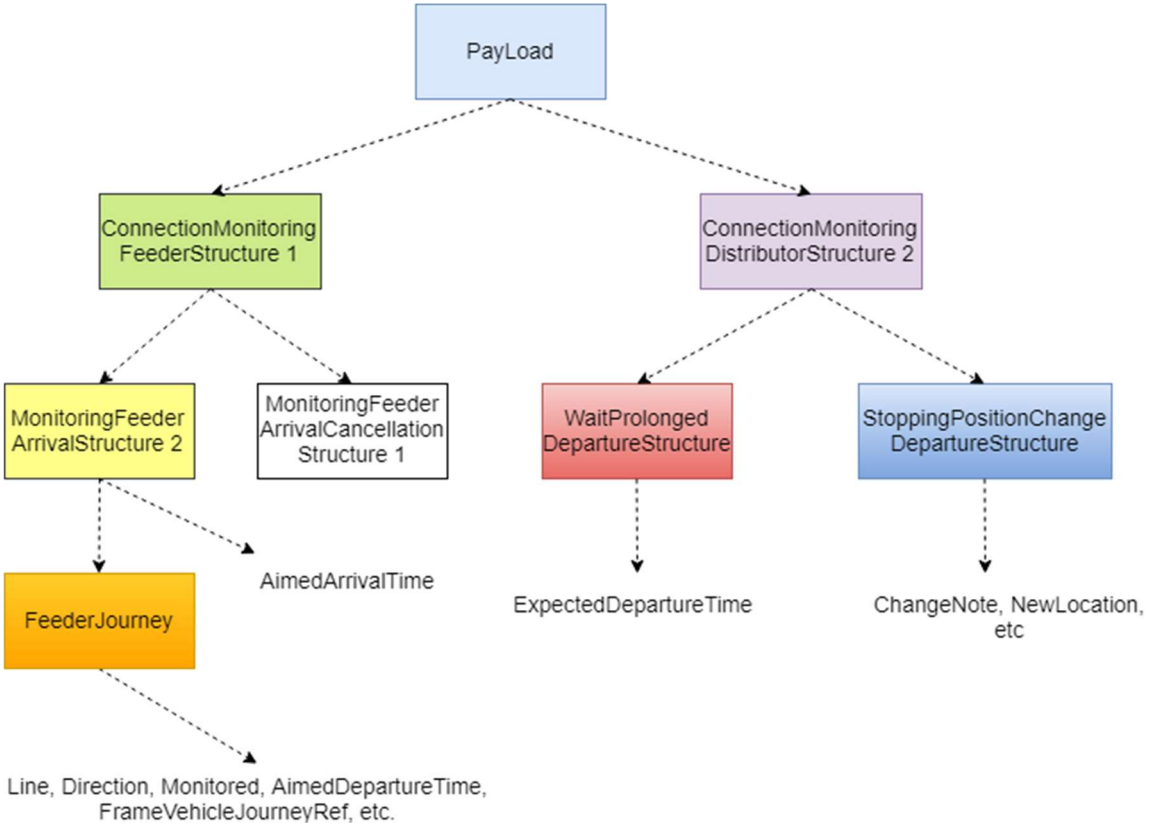


Ilustración 35: Estructura de una respuesta de Connection Monitoring

Se puede observar que la estructura ***ConnectionMonitoringFeederStructure*** es análoga a ***ConnectionTimetableStructure***.

Por otra parte, el servicio *Connection Monitoring* ofrece la estructura ***ConnectionMonitoringDistributorStructure***, que permite comunicar cambios como la ampliación del tiempo de espera del viaje con el que se va a enlazar (***WaitProlongedDepartureStructure***) o una variación del lugar de parada (***StoppingPositionChangeDepartureStructure***).

Las similitudes que se han podido observar en las respuestas de estos dos servicios se mantienen en sus peticiones, ya que ambas permiten filtros tanto por viaje como por hora de conexión entre dos viajes.

Al igual que en parejas anteriores, es más común el uso de *Request/Response* en el servicio planificado (CT) y el uso de subscripciones en el servicio de tiempo real (CM), aun siendo también habituales en el caso de este último las peticiones simples.

3.4.7. *Facility Monitoring* (FM)

El servicio *Facility Monitoring* surgió de manera posterior al resto, con el objetivo de permitir el **intercambio de información relativa a las instalaciones disponibles para los pasajeros** (**en el más amplio sentido de la palabra**) y sus alteraciones.

Proporciona información tanto estática como en tiempo real sobre dichas instalaciones, abarcando este concepto de 'instalación' todo tipo de objetos y servicios que pueden encontrarse asociados a entidades relacionadas con el transporte público, tales como las estaciones, los vehículos, las líneas, las empresas, etc.

Por nombrar algunos ejemplos, algunas de las instalaciones contempladas por SIRI son: rampa, escaleras, ascensor, entrada estrecha, barrera arquitectónica, asiento para personas con necesidades especiales, asientos reclinables, camas, espacio para carritos de bebés, punto de primeros auxilios, alquiler de bicicletas, parada de taxis, zona de fumadores, paneles de información táctil, vehículos, líneas, paradas, etc.

No en vano FM tiene un documento propio (CEN/TS 15531-4) enmarcado entre los cinco que conforman (a día de hoy) SIRI para su completa y correcta descripción.

La estructura típica de una respuesta de *Facility Monitoring* es la siguiente:

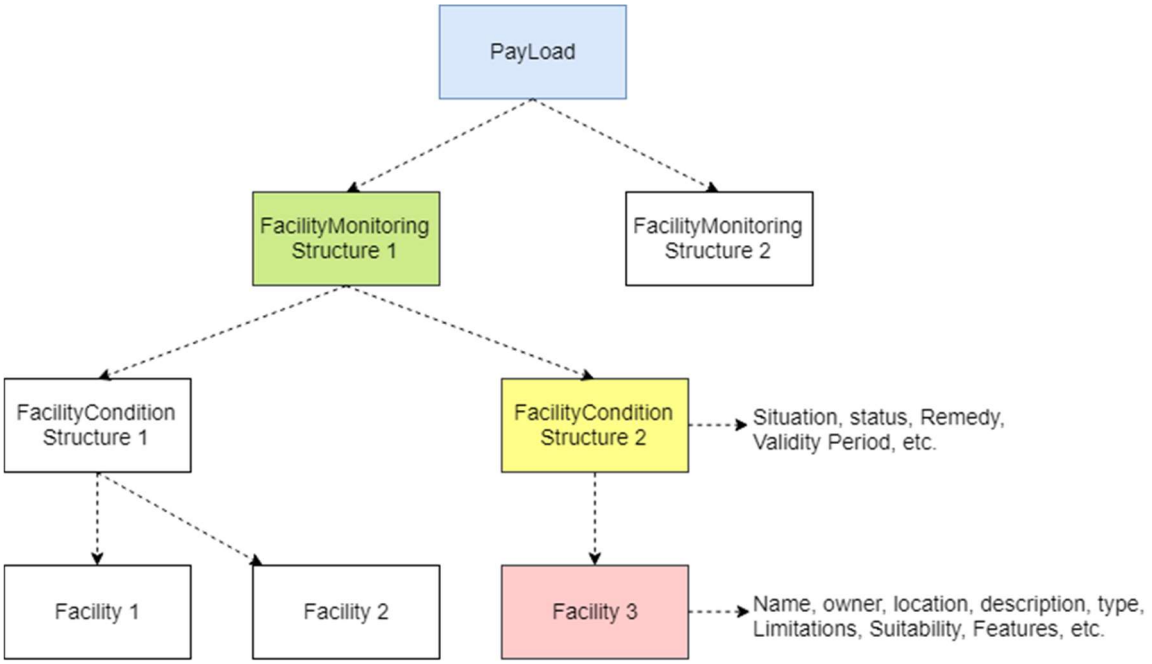


Ilustración 36: Estructura de una respuesta de Facility Monitoring

La estructura jerárquica de la imagen es una simplificación de la estructura real que propone SIRI, que puede complicarse tanto como se desee. En resumen:

- Vector de estructuras tipo ***FacilityMonitoringStructure***, que permite agrupar información de varias subscripciones en un solo mensaje (debiéndose utilizar una sola en caso de una petición simple de tipo *Request/Response*).
- Vector de estructuras tipo ***FacilityConditionStructure***, que permite agrupar las instalaciones por localización: línea, autobús, estación, parada, etc. Proporciona información añadida sobre su situación, validez, estado de observación, etc.
- Vector de estructuras tipo ***Facility***, cada una de ellas correspondiéndose con uno de los posibles campos que SIRI enmarca dentro de las *facilities*. La información que se puede adjuntar en cada una de estas estructuras es inmensa, incluyendo el nombre, el código, la descripción, las características, el dueño, la localización espacial, el tipo de *facility* (SIRI provee de varios cientos de tipos de *facilities*, dentro de los cuales se encuentran los mencionados en la página anterior), etc.

En concordancia con lo anterior, las peticiones admiten también filtros casi ilimitados, entre los que se incluyen:

- Línea en la que está enmarcada la *facility*.
- Parada en la que está enmarcada la *facility*.
- Link en el que está enmarcada la *facility*.
- Vehículo en el que está enmarcada la *facility*.

De hecho, el servicio *Facility Monitoring* puede llegar a ser usado para remplazar las utilidades de *Lines Discovery* y *Stop Points Discovery* que describiremos más tarde, al incluirse estas entidades (líneas y paradas) en la amplia definición de *facility* que nos proporciona SIRI.

Para este servicio SIRI permite tanto la modalidad de *Request/Response*, usada para obtener una gran cantidad de información en peticiones iniciales, como la modalidad *Publish/Subscribe*, que permite a los clientes mantenerse informados en tiempo real sobre los cambios en las instalaciones descritas. En función del uso que en cada implementación se haga de este servicio, es más común un tipo de envío u otro.

3.4.8. *General Message* (GM)

La funcionalidad inicial que se buscó aportar con este servicio fue la de posibilitar el **intercambio de mensajes breves y de carácter general que no queden enmarcados en el resto de los servicios**, como pueden ser advertencias, incidencias, etc.

La estructura de un mensaje tipo GM es bastante sencilla e intuitiva:

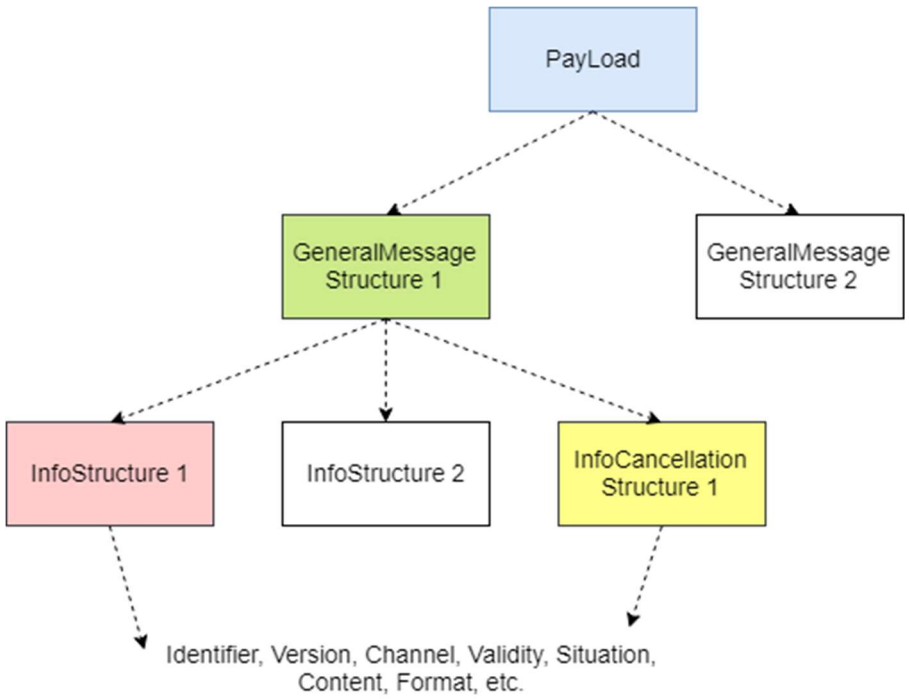


Ilustración 37: Estructura de una respuesta de General Message

Pueden observarse grandes similitudes con las estructuras del resto de servicios: un vector de estructuras tipo ***GeneralMessageStructure***, debajo del cual se encuentran las verdaderas estructuras de los mensajes (bien ***InfoStructure***, o bien ***InfoCancellationStructure*** en caso de que se quiera anular o invalidar uno anterior), ambas con innumerables campos.

Este servicio puede tener lugar mediante un intercambio de tipo *Request/Response* o bien estar compuesto por un solo mensaje que incluye el *payload* del mismo.

Hay que evitar el abuso de este servicio como comodín para transmitir mensajes que tienen cabida en alguno de los otros servicios que SIRI nos proporciona.

3.4.9.*Situation Exchange* (SX)

Este servicio, el más reciente de los que ofrece SIRI (fue publicado en 2016), surgió como respuesta precisamente al mencionado uso mayoritario de *General Message* para la transmisión de **mensajes de alerta y advertencias sobre un incorrecto funcionamiento en algún punto de una red de transporte**.

De manera similar a *Facility Monitoring*, tiene un documento propio (CEN/TS 15531-5) en el que en sus más de 150 páginas se detallan una gran cantidad de situaciones y eventos contemplados, pudiendo afectar estos a vehículos, paradas, trayectos, líneas, instalaciones, etc.

La estructura de una respuesta de *Situation Exchange* es la siguiente:

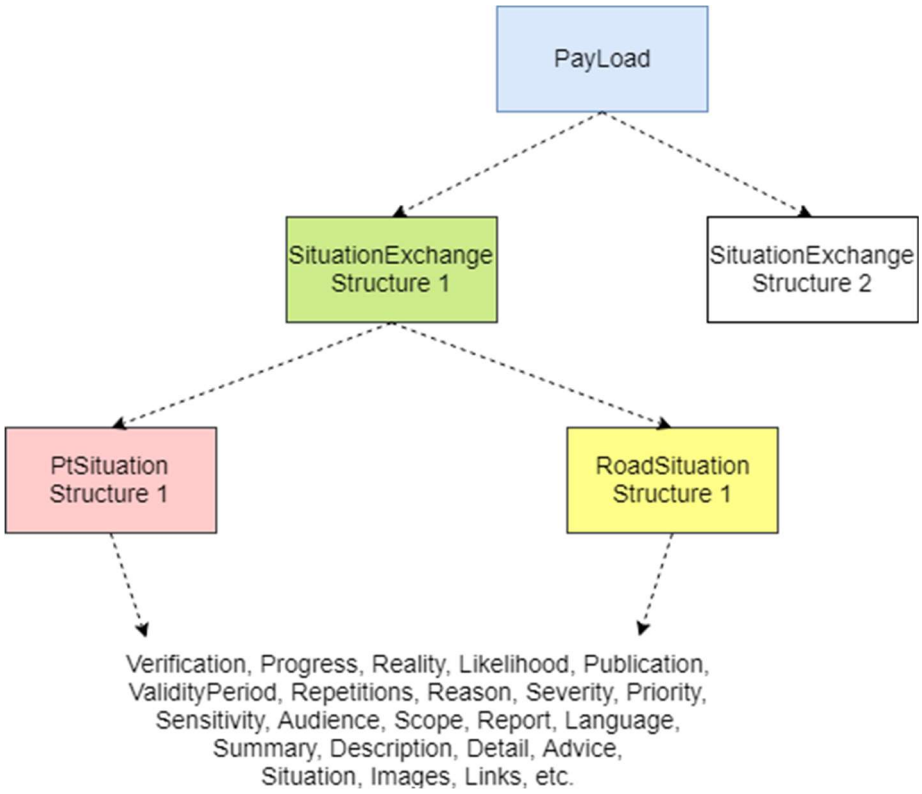


Ilustración 38: Estructura de una respuesta de Situation Exchange

Cada elemento del vector de estructuras tipo ***SituationExchangeStructure*** contiene a su vez dos vectores de dos tipos distintos de estructuras: las ***PtSituationStructure*** (situaciones referidas a puntos de una sola dimensión, véase el concepto *Point* de Transmodel y la multitud de entidades con las que se asocia ese tipo, como los cruces, paradas, estaciones, etc.) y ***RoadSituationStructure*** (referidas a conexiones de dos dimensiones).

Este servicio contempla los dos patrones de intercambio de datos descritos anteriormente, usándose la petición simple para información estática y la subscripción para la información dinámica o de tiempo real.

Las peticiones de *Situation Exchange* contemplan una gran cantidad de filtros: por línea, parada, vehículo, *facility*, etc.

3.4.10. Otros servicios y utilidades

Los siguientes subapartados no se corresponden con servicios funcionales de SIRI, sino con otros servicios y utilidades en torno a los mismos.

3.4.10.1. *Capabilities*

Asociadas a cada servicio funcional de SIRI existen una serie de 'capacidades' o ***capabilities*** de los mismos, que **describen la configuración del servidor de SIRI respecto a esos servicios**:

- Campos que se rellenan en una respuesta de ese servicio.
- Filtros que los clientes pueden aplicar en las peticiones a ese servicio: de entre la lista de filtros que SIRI provee, aquellos que el servidor está configurado para aceptar.
- Política de peticiones aceptadas: máximo número de paradas/ vehículos/ pasos por parada, etc.
- Valores por defecto, entre ellos el *DefaultPreviewInterval*, usado en muchas de las peticiones para acotar las respuestas en el tiempo.
- Tipo de envío implementado: *Request/Response* y/o *Publish/Subscribe*.
- En el caso de permitir subscripciones, si las actualizaciones son incrementales (tal y como define el estándar) o no.
- Patrón de entrega de datos: *Direct Delivery*, *Fetched Delivery*.
- Método de transporte: HTTP POST, HTTP GET JSON, SOAP, ProtoBuffers…
- Método de compresión: sin compresión, gzip, otro.
- Otros parámetros de configuración: sistema de coordenadas (como WGS84), uso de GML, idioma, etc.

Las *capabilities* permite a un cliente conocer qué posibilidades ofrece un servidor de SIRI, siendo esto especialmente importante en las situaciones en las que el servidor es ajeno a los clientes (bien por tratarse de un *endpoint* público, o bien por pertenecer a otra organización o entidad, etc.).

La estructura de los mensajes relativos a las *capabilities* es muy simple:

- En el caso de las peticiones (*GetCapabilities*), existe una sencilla estructura por cada servicio funcional de SIRI. Todas ellas deben ser rellenadas (basta con incluir la hora de la petición) para que la petición sea considerada válida
- En las respuestas también existen tantas subestructuras como servicios funcionales provee SIRI. Ya que en las peticiones no puede ser concretado ningún servicio específico, el servidor deberá de encargarse de rellenar todas aquellas estructuras de la respuesta cuyo servicio esté implementado en él. En caso de que una de esas estructuras no esté rellena, el cliente interpretará la ausencia de esa funcionalidad en el servidor que así le ha respondido.

3.4.10.2. *Discovery Services*

Además de los en ocasiones llamados servicios funcionales de SIRI, anteriormente descritos, existen otro tipo de servicios que proveen a los clientes de información sobre ciertos elementos del sistema.

- ***Lines Discovery***: proporciona el conjunto de líneas existentes en el sistema que están disponibles para su consulta.
- ***StopPoints Discovery***: proporciona el conjunto de paradas existentes en el sistema que están disponibles para su consulta.
- ***ConnectionLink Discovery***: proporciona el conjunto de puntos de conexión existente en el sistema que están disponibles para su consulta.

Ya que estos tipos de elementos están enmarcados dentro del amplio grupo de lo que SIRI considera *facilities* del sistema, se podría transmitir la información correspondiente a estos servicios utilizando el servicio funcional *Facility Monitoring*; pero esta práctica no parece adecuada al dar un uso a este servicio que no se corresponde con el descrito en el estándar.

Un ejemplo del uso de los *Discovery Services* puede encontrarse en la implementación del cliente realizada, en la que se pide en primer lugar la lista de líneas y paradas disponibles para poder usarlas en los tests, sin necesidad de saber en tiempo de compilación la información del servidor al que se atacará. Por ejemplo:

- Para probar el filtro de *Stop Monitoring* por parada, los tests eligen estocásticamente una parada de entre las disponibles en el sistema (obtenidas de *StopPoints Discovery*).
- Para probar la excepción ante una solicitud de *Vehicle Monitoring* con los datos filtrados por una línea que no existe, los tests se aseguran de que ésta no esté entre las proporcionadas por el servicio *Lines Discovery*.

3.4.10.3. *CheckStatus*

SIRI proporciona a los consumidores (clientes) un mecanismo de *pulling* a través del cual comprobar de manera sencilla el estado del servidor.

Éste puede responder con su estado (correcto o no), cualquier error que pueda estar experimentando, la fecha de inicio del servicio, etc.

Tanto la petición como la respuesta del *CheckStatus* están formadas por estructuras muy simples, procurando mantener los mensajes relacionados con este mecanismo lo más ligeros posibles al preverse su utilización regular por parte de los clientes.

3.4.10.4. Notificaciones

SIRI proporciona al *Producer* (servidor) mecanismos para comunicarse con el cliente a través de lo que el estándar denomina notificaciones.

Existen tres tipos básicos de notificaciones:

- ***DataReady***: en las comunicaciones que siguen el patrón de entrega de datos *Fetched Delivery*, es el servidor el que comunica al cliente que existen datos actualizados relacionados con una subscripción. Como se explicó en el apartado correspondiente, en el caso de intercambio tipo *Request/ Response* es el servidor quien (a través del *NotificationProducer*) comunica al cliente (al *Notification Consumer*, para ser precisos en la terminología empleada) que ya tiene listos los datos solicitados; y lo hace a través de la *DataReadyNotification* (ver ilustración 20).

  En el caso de *Publish/Subscribe*, cuando al *Notification Producer* se le informa de nuevos cambios y éste se cerciora de que estos afectan a la subscripción de un cierto cliente, éste comunica al *Notification Consumer* la existencia de novedades en la subscripción a través de la *DataReadyNotification* (ver ilustraciones 21 y 22).
- ***Heartbeat***: de igual manera que SIRI proporciona el servicio de *CheckStatus* a los clientes para que estos puedan conocer el estado de los servidores en cualquier instante, SIRI proporciona a estos últimos la notificación de *Heartbeat*, que puede ser usada para informar periódicamente a sus clientes del correcto funcionamiento del sistema.
- ***NewDataNotification***: SIRI define una notificación por servicio con la que el servidor puede informar a los clientes de que existen nuevos datos relacionados con dicho servicio.

  En uno de los casos mencionados en la explicación de la notificación tipo *DataReady* pueden usarse también este tipo de notificaciones: cuando una comunicación usa simultáneamente el patrón de intercambio de datos *Fetched Delivery* y el patrón de entrega de datos *Publish/Subscribe*, las notificaciones de nueva información que el *Notification Producer* hace llegar al *Notificacion Consumer* pueden realizarse usando *NewDataNotification*; siendo una buena práctica hacerlo en caso de que existan múltiples subscripciones pertenecientes a un mismo *Notification Consumer* y asociadas a distintos servicios de SIRI. Este es el ejemplo que se ilustra en la figura 22.

  Los tipos de notificaciones son:
  ProductionTimetableNotification, *EstimatedTimetableNotification*, *StopTimetableNotification*, *StopMonitoringNotification*, *VehicleMonitoringNotification*, *ConnectionTimetableNotification*, *ConnectionMonitoringNotification*, *GeneralMessageNotification*, *FacilityMonitoringNotification* y *SituationExchangeNotification*.

3.4.10.5. Subscripciones

Recordemos el patrón de intercambio de datos *Publish/Subscribe*:

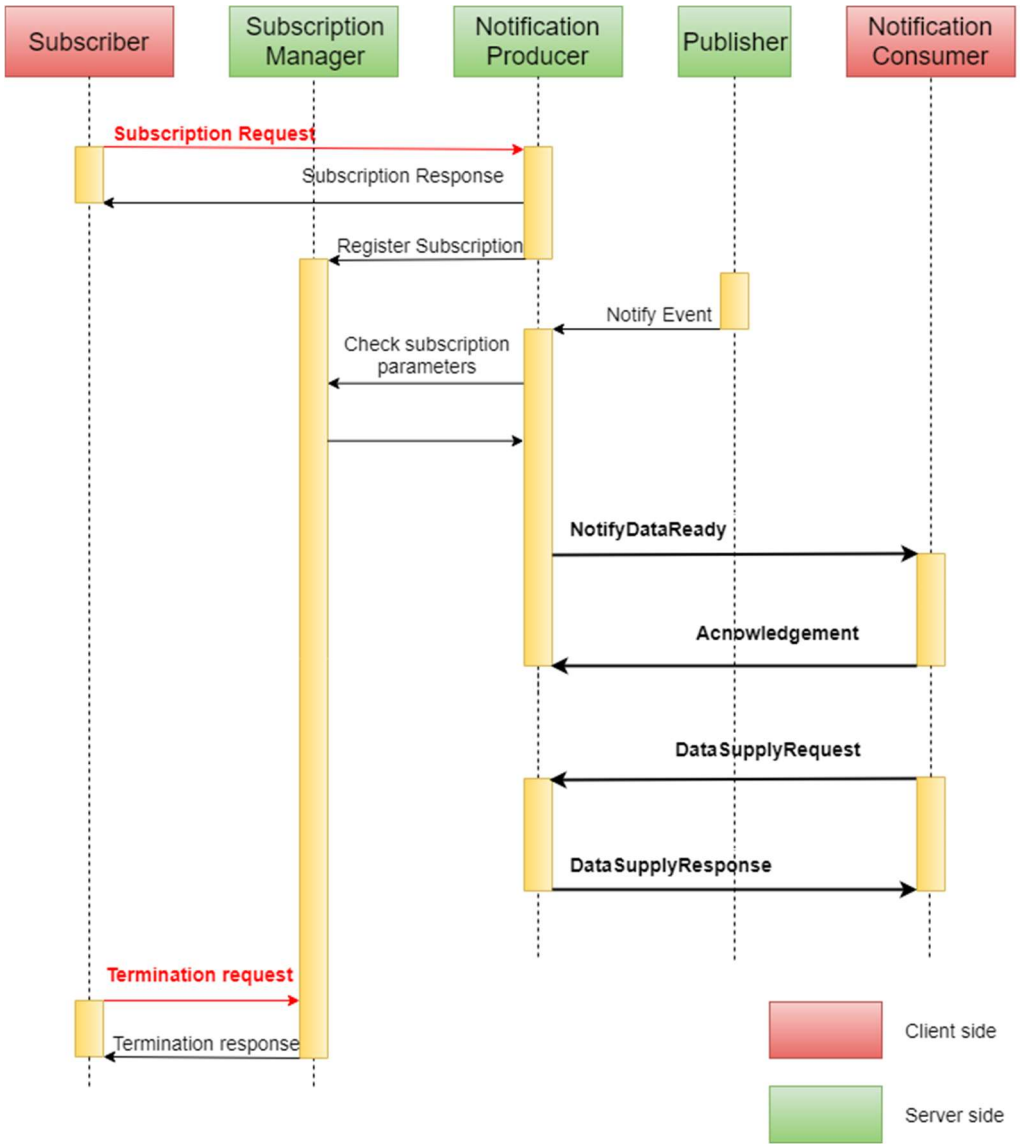


Ilustración 39: Estructura de un intercambio tipo Publish/Subscribe con entrega directa

Como se puede apreciar en la imagen, la gestión de las subscripciones se realiza principalmente a través de dos tipos de llamadas:

- **Subscribe:**
  El *Subscriber* se comunica con el *Notification Producer* para solicitar su subscripción a un determinado servicio.
  La ilustración 39 representa un esquema bastante completo de los posibles *endpoints*, pero a estos pueden añadirse otros (ver apartado de *SIRI Endpoints*), como las direcciones del servidor que reciben las peticiones de tipo *CheckStatus* o las peticiones de *DataSupplyRequest* en caso de *Fetched Delivery;* o como las direcciones del cliente a las que mandan los *NotifyDataReady* (también *Fetched Delivery*) o los *Heartbeats*.

Veamos un esquema de una petición de este tipo:

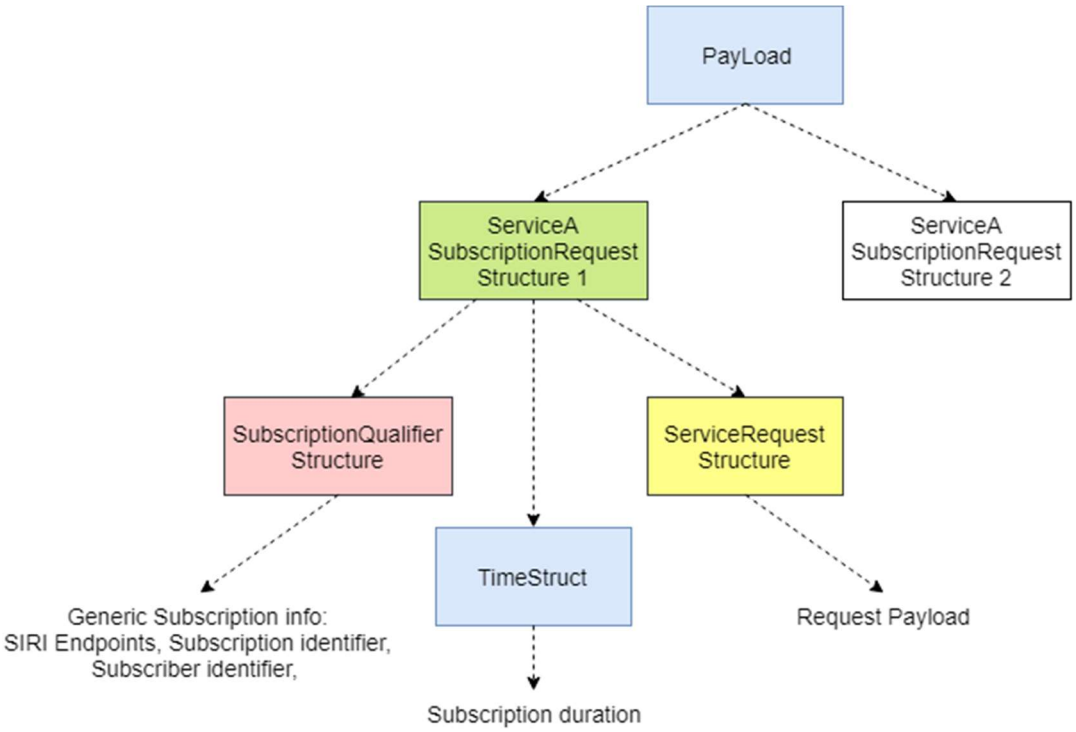


Ilustración 40: Esquema de una petición Subscribe

El vector de estructuras *SubscriptionRequest* (por ejemplo, vector de *VehicleMonitoringSubscriptionRequest*) nos permite incluir varias peticiones de subscripción, pero siempre y cuando todas ellas hagan referencia al mismo servicio.
Cada una de esas estructuras, tipo *XXXXXXSubscriptionRequestStructure*, está a su vez compuesta por tres partes:

- *SubscriptionQualifierStructure*: estructura en la que se almacena la información genérica en torno a la suscripción (*endpoints*, identificador de subscripción y suscriptor, etc.).
- Estructura para almacenar datos temporales: estructura en la que se concreta la duración o caducidad de la subscripción.
- *XXXXXXRequestStructure*: estructura que contiene la propia información a la que se quiere ser suscrito (por ejemplo, *VehicleMonitoringRequest*).

Como veremos en el apartado correspondiente al *Subscription Manager* del capítulo en el que se describe nuestra implementación del servidor, la naturaleza de estas últimas estructuras tipo *ServiceRequestStructure* cobra especial relevancia a la hora de gestionar las subscripciones.

- **DeleteSubscription:**
  El suscriptor solicita al servidor anular una subscripción, más concretamente al *Subscription Manager*, que es quien gestiona las subscripciones una vez han sido aprobadas por el *Notification Producer*.
  La estructura de estos mensajes es muy simple, bastando con introducir los identificadores de las subscripciones que se desean cancelar en un vector de cadenas de caracteres contenido en dicha estructura.

4. Aplicación práctica: Implementación de un servidor y un cliente capaces de comunicarse siguiendo el estándar SIRI

Tras contextualizar los Sistemas Inteligentes de Transporte (ITS) y su situación actual, poner de relieve la necesidad de estandarización que ha surgido paralelamente a los mismos e introducir teóricamente el estándar SIRI; en este capítulo se procede a describir la **implementación real de un servidor y un cliente capaces de comunicarse siguiendo el estándar SIRI.**

Nos referimos a las implementaciones como reales en tanto han sido desarrolladas en el **seno de una empresa dedicada a los Sistemas Inteligentes de Transporte**, GMV.

En los siguientes apartados se detalla, además de la arquitectura en profundidad de servidor y cliente, el intercambio de datos que tiene lugar entre ellos. Adicionalmente, y probando así la validez de la implementación del estándar realizada, se muestra el uso de ambos para la comunicación con fuentes externas al entorno en el que han sido desarrollados:

- El servidor es validado por un cliente de SIRI externo, de código abierto y desarrollado por la agencia francesa AFIMB.
- El cliente es usado para conectarse a *endpoints* públicos de SIRI, accediendo a información real sobre el transporte público de lugares como Nueva York, Utah o Tampere.

En relación con este capítulo, en los anexos pueden consultarse los siguientes documentos:

- Anexo 1: Ejemplos (XML) de peticiones y respuestas de los distintos servicios de SIRI.
- Anexo 2: El código fuente de la implementación del cliente.
- Anexo 3: Trazas de datos reales proporcionados por cada uno de los servicios de SIRI implementados en el servidor, que ayudan a comprender la utilidad de los distintos servicios funcionales del estándar.
- Anexo 4: Información obtenida de los *endpoints* públicos consultados (archivos con formato XML o JSON).

La parte servidor de nuestra implementación de SIRI está estrechamente ligada al modelo real de datos usado por GMV. Por lo tanto y por motivos de confidencialidad, en los anexos no se incluye el código de dicha implementación. Sí se han plasmado en este documento algunos fragmentos de código relevantes, debidamente modificados donde así se ha estimado oportuno.

4.1. Autogeneración de código con gSOAP

En el apartado correspondiente al transporte de los mensajes en SIRI se describen brevemente los métodos contemplados por estándar:

- Uso de peticiones HTTP POST directas, intercambiando archivos XML.
- Uso de SOAP como envoltorio en torno a las peticiones anteriores.
- Uso de SIRI Lite (o *SIRI Simple Web Services*), basado en peticiones HTTP GET fuertemente parametrizadas. Este método admite tanto XML como JSON como formatos de la respuesta.

Las implementaciones realizadas, tanto del cliente como del servidor, se han llevado a cabo siguiendo el segundo de los métodos mencionados: usando el **protocolo SOAP**, que define la comunicación entre dos sistemas mediante el intercambio de datos en formato **XML** a través de peticiones **HTTP POST**.

gSOAP (*generic XML and SOAP*) es una herramienta que facilita el serializado (o *binding*) y des-serializado (parseado) de datos entre nuestro lenguaje de implementación (C++) y los archivos XML. Esta utilidad nos permite **autogenerar de manera parametrizada código C++** a partir de los WSDL y XSD que definen SIRI, de manera que éste pueda ser usado por nuestros programas para el **parseo de los datos recibidos y la serialización previa al envío de los mismos**.

El procedimiento a seguir para llevar a cabo la generación de las estructuras de datos de SIRI es el siguiente:

- Uso del parseador **wsdl2h** para convertir los archivos XSD y WSDL en un archivo de código fuente (con extensión .h o .hpp en el caso de C++) [17].
- Uso del compilador **soapcpp2** para generar, a partir del archivo resultante del paso anterior, las estructuras de *Producer* y *Consumer* a usar, así como los *stubs* y el código de SOAP necesario [18].

Para comprender la magnitud de esta autogeneración de código basta con conocer dos cifras:

- El archivo parseado a partir de wsdl2h (*SIRI.h*) consta de ~60.000 líneas.
- Los archivos generados por soapcpp2 para un productor y un consumidor suman entre todos ellos unas 645.000 líneas código. Son: *soapH.h, soapC.cpp, stdsoap2.h/.cpp. soapStub.h, soapSiriProducerDocBindingProxy.h/.cpp y soapSiriProducerDocBindingService.h/.cpp.*

Aunque escape del alcance de este proyecto, cabe destacar que, posteriormente a las implementaciones, se requirió la eliminación del envoltorio SOAP para algunos clientes; por lo que también fue codificada una herramienta en C# para realizar esa tarea.

Las diferencias entre los XML con y sin envoltorio son mínimas, intercalándose XML de ambos tipos en los ejemplos de los anexos.

4.2. Implementación de un servidor para SIRI

4.2.1. Introducción y contexto

Tal y como se ha detallado anteriormente, SIRI es un protocolo estandarizado de comunicación de datos de transporte público especialmente diseñado para ser usado en las comunicaciones entre servidores de datos.

Esto nos ubica en el contexto de un **sistema de gestión de flotas y vehículos** (ver CAD/AVL o FMS), frente a la otra gran rama de los Sistemas Inteligentes de Transporte que consiste en la gestión de la parte monética en torno a los mismos (FCS).

Dado que SIRI es un protocolo adecuado para comunicaciones de alto nivel (frente a comunicaciones, por ejemplo, entre los propios vehículos y su centro de control), una implementación real de SIRI implica la preexistencia de unos datos de transporte que intercambiar.

En nuestro caso, partíamos de la existencia de un servidor de FMS que se encargaba de la obtención de esos datos de las OBUs instaladas en los propios vehículos.

Esta sería la estructura básica de un servidor de estas características:

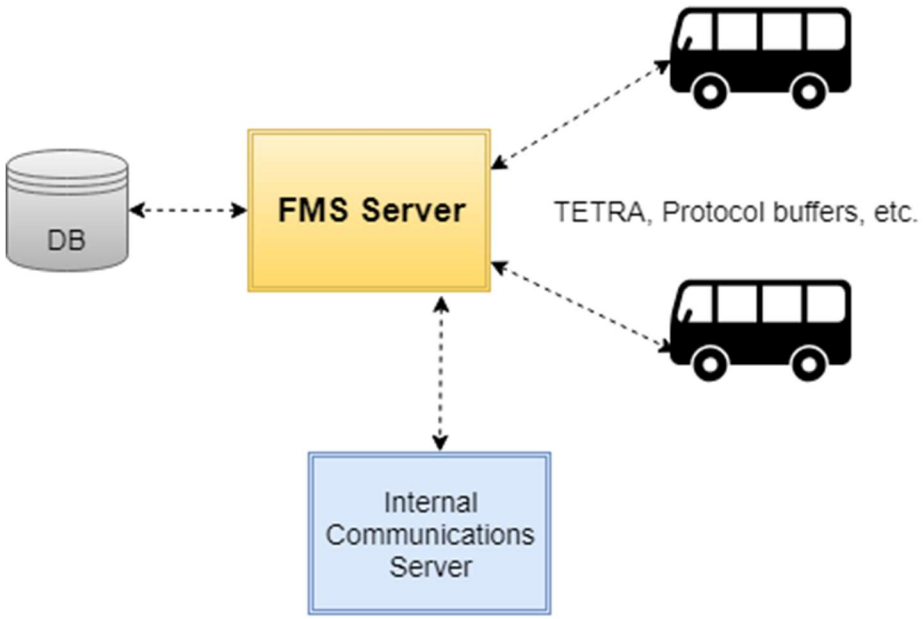


Ilustración 41: Esquema básico de un servidor de FMS funcional

El servidor esquematizado se comunica directamente con la flota de vehículos perteneciente al operador de transporte que le ha contratado, a través de protocolos de comunicación muy ligeros.

Asimismo, es capaz de comunicarse con otros servidores 'compatibles', por ejemplo, aquellos pertenecientes a propia empresa dueña de ese servidor.

Esto se traduce en que la implementación de un servidor de SIRI a la que se hace referencia en este documento **no ha implicado la implementación completa de un servidor de FMS**, algo inabarcable para un proyecto de estas características; sino a la implementación de la parte servidor del estándar SIRI, si interpretamos éste en base al paradigma cliente-servidor.

En la práctica, SIRI aporta a un servidor de CAD/AVL la posibilidad de ampliar sus protocolos estandarizados de comunicación; lo que resulta especialmente útil cuando se desea la comunicación con servidores externos e imprescindible en el caso de que estemos hablando de un servidor dedicado exclusivamente a la comunicación y redistribución de información.

Con el tiempo, en el servidor representado en la imagen anterior fue surgiendo la necesidad de integración con otros servidores externos (**necesidad de estandarización de la gestión e intercambio de información relativa al transporte público** ya descrita en el capítulo 2).

Alrededor del servidor se fueron implementando otros protocolos y estándares, como el GTFS, de tal forma que la tendencia fue pasando a ser algo parecido a lo que se muestra en esta ilustración:

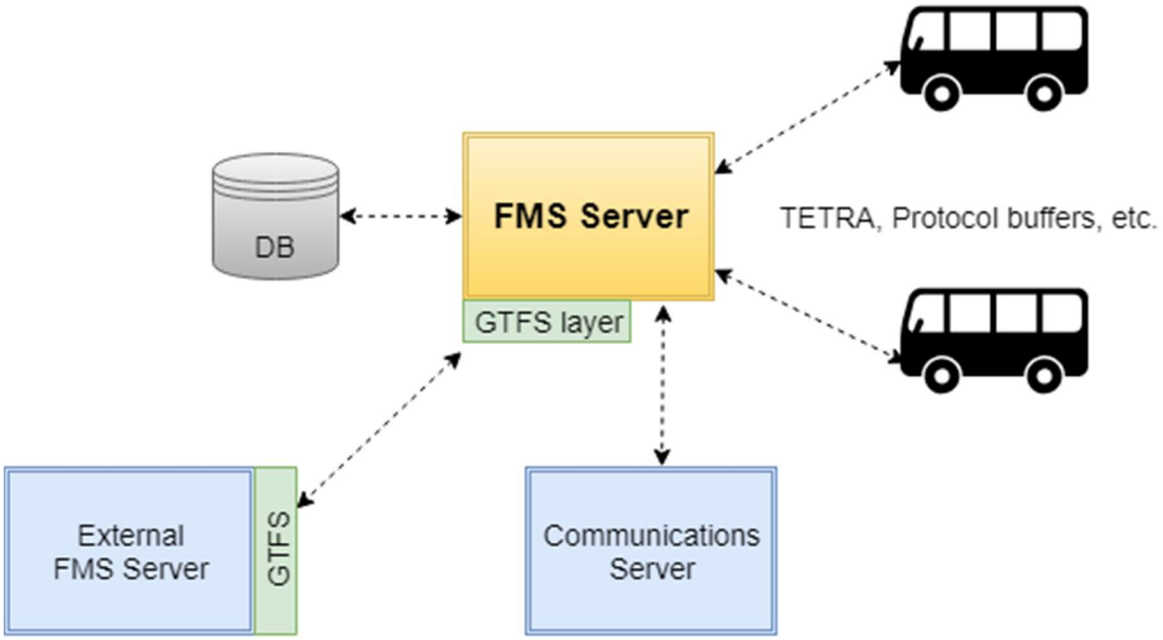


Ilustración 42: Esquema básico de un servidor de FMS funcional con GTFS

Como se puede observar, la capa de GTFS del servidor le permite comunicarse con otros servidores que tengan una API estandarizada que siga este protocolo.

De manera similar sucede con otros estándares, que fueron implementados en base a las necesidades que fueron surgiendo.

Tras esta breve contextualización de cómo los servidores encargados de obtener directamente la información pasan inevitablemente por el proceso de estandarización de sus comunicaciones, justificado y detallado en el capítulo 2, volvamos a la situación real.

El marco en el que se realizó la implementación de la parte servidor del estándar SIRI era el de un servidor de FMS ya maduro de una empresa dedicada al desarrollo de Sistemas Inteligentes de Transporte.

La creciente popularidad experimentada por SIRI en los últimos años y su aparición como requisito en los pliegos de los últimos proyectos de ITS licitados desembocó en la decisión de implementar dicho estándar como una capa más alrededor del servidor de FMS existente, tal y como se muestra en la figura:

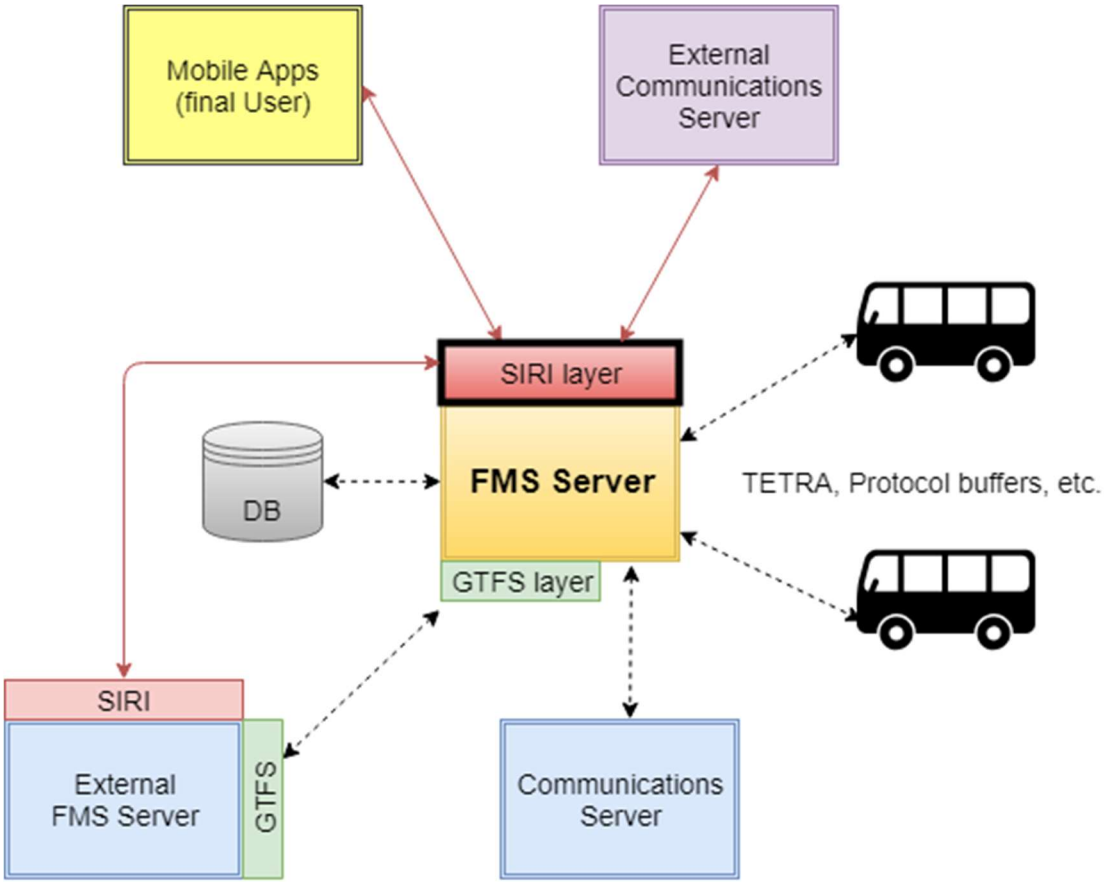


Ilustración 43: Esquema básico de un servidor de FMS funcional, con GTFS y SIRI

De esta manera, la importación y exportación de datos pasaría a poder realizarse en base a un estándar ampliamente extendido como es SIRI, quedando el servidor preparado para el intercambio de grandes cantidades de datos con otros servidores de FMS, gestores de comunicación, aplicaciones móviles, etc.

4.2.2. Características técnicas del servidor de FMS

El mencionado servidor de FMS consiste en una **aplicación monolítica multi-hilo implementada en C++** (ajustándose a los estándares C++11 y C++14), con dependencias externas como Boost, Xerces, zlib, OpenSSL, icu4c, etc.

Alrededor de la misma, y para su comunicación con el exterior, tiene una serie de ***plugins* que permiten la importación y exportación** de los datos de transporte público gestionados por el servidor (e internamente estructurados en base a un modelo de datos propio) siguiendo distintos estándares y protocolos: un plugin permite importar y exportar información siguiendo el estándar GTFS, otro siguiendo GTFS-R, otro TransXChange, etc.

La implementación del servidor de SIRI constituyó otro de esos *plugins* que facilitan la comunicación estandarizada del servidor con el exterior.

En función de las necesidades de los clientes (operadores y autoridades de transporte) de cada proyecto, se activan o desactivan los *plugins* mencionados en la configuración concreta de cada instalación.

Los *plugins* tienen **acceso a la información** a través de tres vías principales:

- Bases de datos, a través de clases internas del servidor.
- Información de la memoria de la aplicación, a través de clases internas del servidor.
- Información de tiempo real, a través de eventos (gestionados usando la librería boost.Signals2 [19]).

Las fuentes de esa información pueden ser tanto el sistema en producción del cliente final (una vez se ha hecho entrega del sistema) como un simulador temporal, generalmente asociado a la operativa real de un día previo (trabajo sobre datos reales, pero 'en diferido').

Por defecto, dicha información se puede ver reflejada en una aplicación gráfica de escritorio (la utilizada en los centros de control de los operadores de transporte para la gestión de sus flotas), a lo que se suma la exportación realizada por los *plugins* activados en cada momento.

Esta imagen recopila el sistema de ITS en el que se enmarca el servidor:

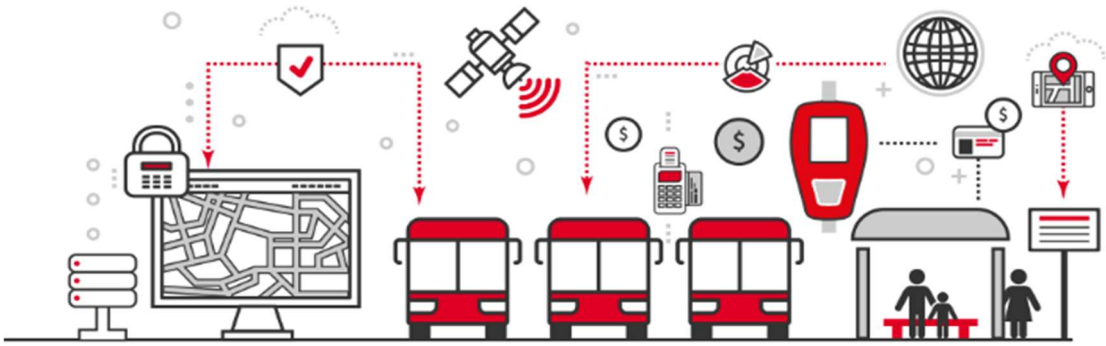


Ilustración 44: Servicios ITS ofrecidos por GMV (GMV)

4.2.3. Visión global de la arquitectura del servidor de SIRI

La implementación del servidor de SIRI que se pasa a describir se llevó a cabo desde cero, lo que permitió total libertad en el diseño de la arquitectura interna del *plugin* de SIRI, al no tener que ser compatible con implementaciones preexistentes.

Antes de adentrarnos en la descripción las distintas partes que componen nuestra implementación y la justificación de la existencia de las mismas, se muestra este diagrama arquitectónico general del conjunto implementado, que deberá servir de referencia para los siguientes apartados:

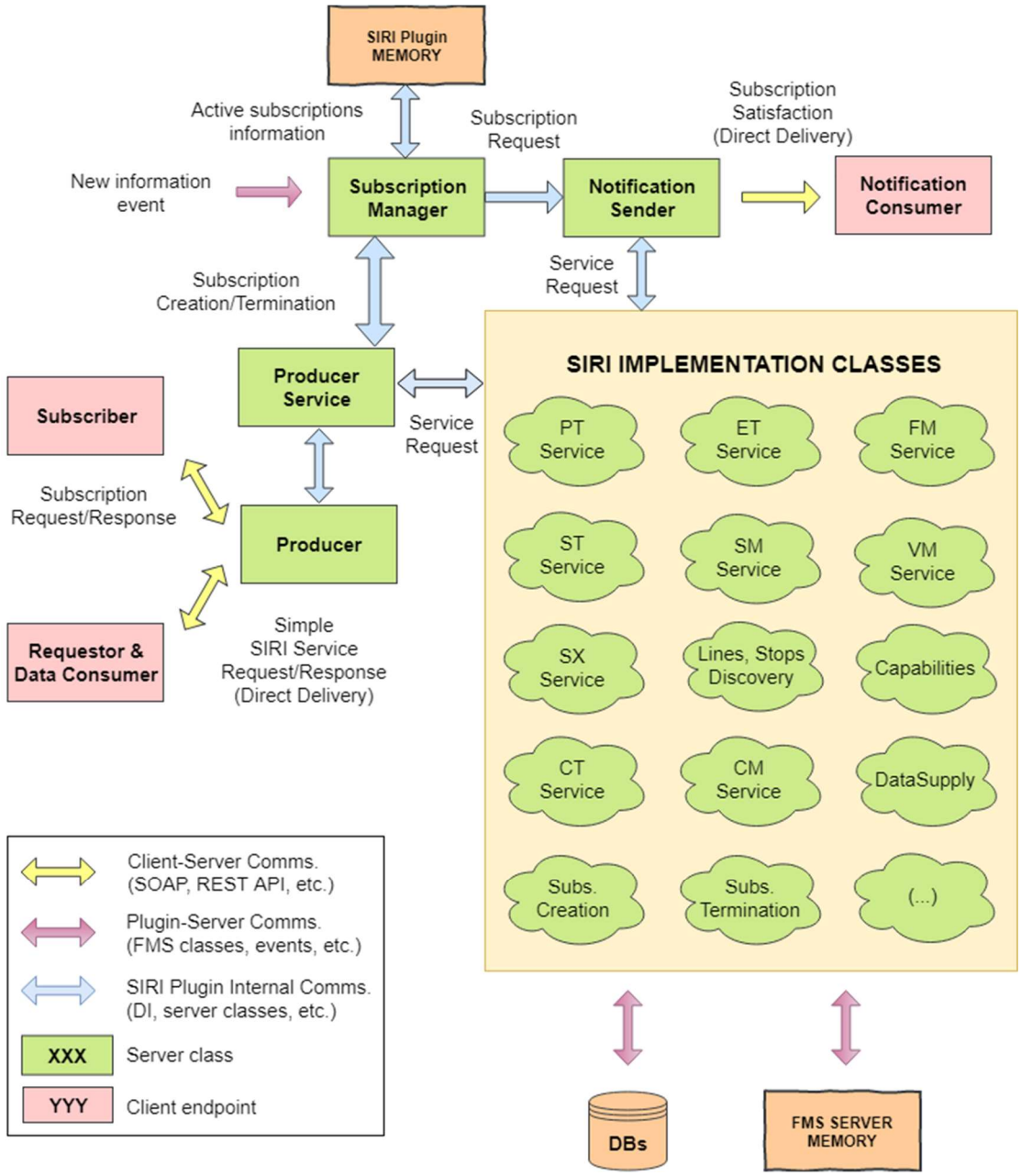


Ilustración 45: Esquema de la arquitectura del servidor de SIRI implementado

4.2.4. Clase Producer

Como ya hemos visto, la parte servidor de SIRI puede estar distribuida en numerosos *endpoints*, lo que le permitiría fácilmente estar formada por varias aplicaciones independientes. Sin embargo, de la información expuesta anteriormente se deduce que nuestra implementación de SIRI debe ser entendida como una capa para la importación y exportación de datos del servidor de FMS, por lo que estará gestionada desde una única clase a la que se ha denominado **`Producer`** y que está 'atacheada' al *threadpool* (conjunto de hebras) principal del servidor de FMS.

El **`Producer`** se valdrá de una clase sin estado llamada **`ProducerService`**, que implementa la clase autogenerada `SiriProducerDocBindingService`, para resolver las peticiones del conjunto de *endpoints* que componen la parte servidor de SIRI; así como de una clase con estado llamada **`Subscription_Manager`** que cumplirá con la función de gestión de subscripciones que tiene el *Subscription Manager* del estándar.

El párrafo previo puede resumirse en este esquema:

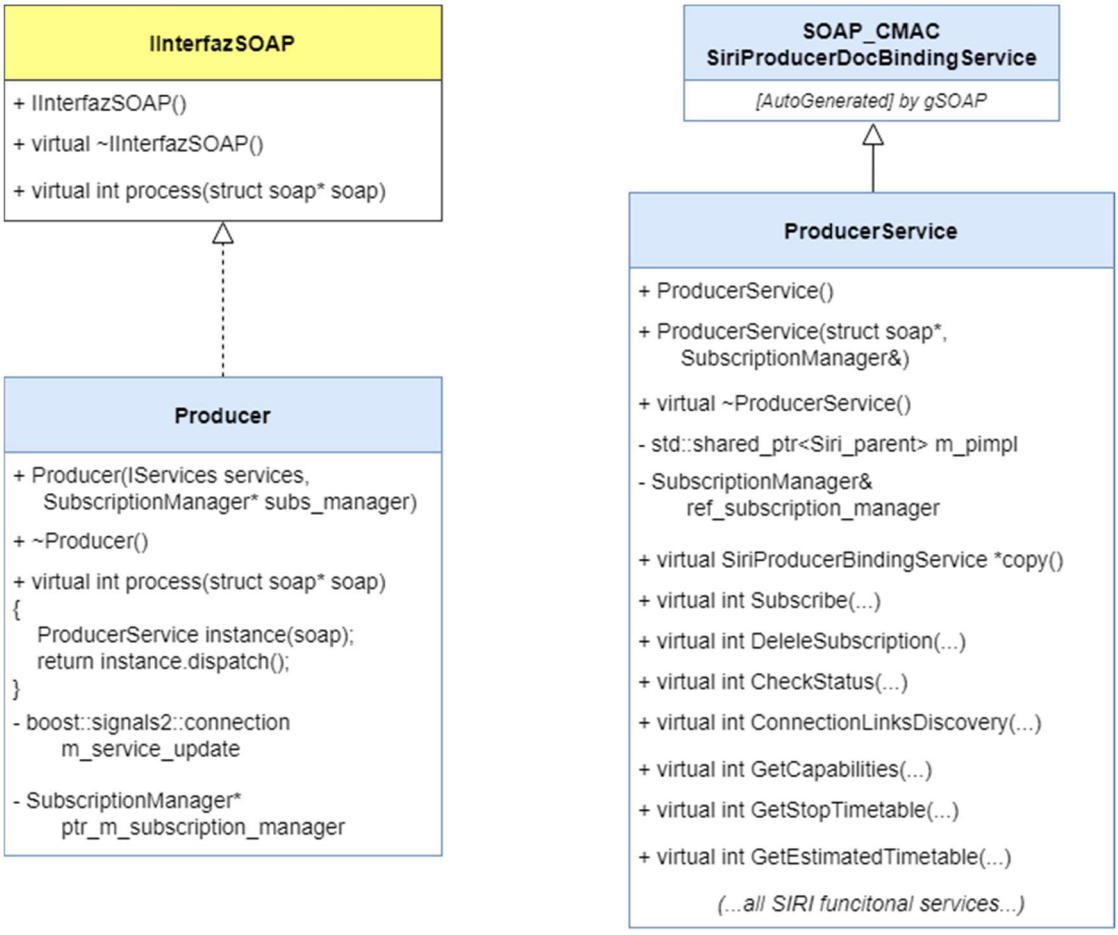


Ilustración 46: Diagramas UML de las clases Producer y ProducerService

Cuando se realiza una petición REST de acuerdo con el protocolo SOAP contra el puerto configurado de nuestro servidor, el método `process(struct soap* soap)` instancia un objeto de tipo **`ProducerService`** y resuelve la petición recibida, invocando al método adecuado de esta instancia en función de dicha petición.

Dichos métodos reciben estructuras de C++ con los datos del mensaje.

Por ejemplo:

```
int ProducerService::GetStopMonitoring(ns1__StopMonitoringRequestStructure*
ns1__GetStopMonitoring, ns1__StopMonitoringAnswerStructure&
ns1__GetStopMonitoringResponse)
```

- En la estructura a la que apunta el puntero del primer argumento, `ns1_GetStopMonitoring`, se encuentra el contenido de la petición.
- La segunda estructura, `ns1__GetStopMonitoringResponse`, se recibe por referencia para poder ser rellenada con los datos de la respuesta.

La firma de los métodos de otros servicios funcionales sigue el mismo patrón:

```
int ProducerService::GetProductionTimetable(
ns1__ProductionTimetableRequestStructure *ns1__GetProductionTimetable,
ns1__ProductionTimetableAnswerStructure& ns1__GetProductionTimetableResponse)
```

Es aquí donde verdaderamente se aprecia el trabajo de gSOAP y las ~640.000 líneas del código que éste autogenera a partir de los XSD y WSDL del estándar: se encarga de transformar una petición REST en dos estructuras de datos, una con la información parseada de dicha petición y otra preparada para ser rellenada con la respuesta deseada y convertida de nuevo a XML para ser devuelta al cliente.

La clase `Producer` contiene un puntero hacia una instancia de **`SubscriptionManager`**, clase a la que se invocará desde los métodos `Subscribe(…)` y `DeleteSubscription(…)` de `ProducerService`. Ya que esta última es instanciada con cada petición (alcance *scoped*), cada objeto de tipo `ProducerService` tiene que almacenar una referencia del `SubscriptionManager` (alcance *Singleton*), por si tiene que recurrir a él.

El otro campo privado de la clase Producer es `boost::signals2::connection m_service_update`, que almacena la conexión entre eventos de otras partes del servidor de FMS y el método que escojamos. Esto nos permitirá satisfacer las subscripciones tan pronto como sucedan cambios significativos en la información a la que estas están suscritas.

En el constructor del `Producer` se lleva a cabo dicha conexión entre los eventos y el método del `SubscriptionManager` que se encargará de la gestión de estas notificaciones, como veremos más adelante.

Teniendo en cuenta lo anterior, el código del `Producer` queda de la siguiente manera:

```
class Producer
	: public IInterfazSoap
{
public:
	Producer(IServicios* pServicios, SubscriptionManager* subs_manager)
		: ptr_m_subscription_manager(subs_manager)
	{
		if (pServicios)
		{
			m_service_update_connection =
pServicios->m_eventActServicio.connect(
[&](eTipoNotificacion tipo, const IServicio* ptr)
			{
				ptr_m_subscription_manager->OnUpdateService(tipo,
ptr);
			});
		}
	}

	~Producer()
	{
		if (m_service_update_connection.connected())
		{
			m_service_update_connection.disconnect();
		}
	}

	virtual int process(struct soap* _soap)		// Connection
	{
		ProducerService sl(_soap, *ptr_m_subscription_manager);

		return sl.dispatch();
	}

private:
	boost::signals2::connection m_service_update;

	SubscriptionManager* ptr_m_subscription_manager;
};
```

4.2.5.Integración del servidor de SIRI en el servidor de FMS

Antes de seguir profundizando en las entrañas de la implementación, merece la pena mencionar el simple **mecanismo mediante el cual se conecta la instancia del `Producer`**, que a efectos prácticos constituye el plugin de SIRI del servidor, al 'núcleo' de éste.

Esto nos ayudará a comprender ciclo de vida de cada clase y sus campos.

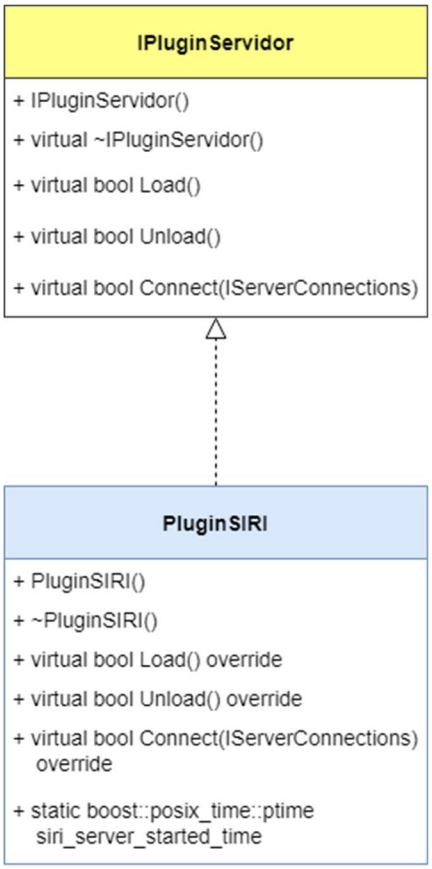


Ilustración 47: Diagrama UML de la clase PluginSIRI

Este esquema representa el *plugin* que conecta al servidor real de FMS con la implementación llevada a cabo de la parte servidor de SIRI.

La función `Load()` de `PluginSIRI` se encarga de crear dos punteros inteligentes de tipo `shared_ptr<T>` para sendas instancias de **`SubscriptionManager`** y de **`Producer`**, almacenándolos de tal manera que su ciclo de vida sea igual al del plugin de SIRI; lo que convierte a dichas instancias en *Singletons*.

Esto permite al `Producer` recibir en su método de creación el puntero crudo `SubscriptionManager* ptr_m_subscription_manager` para almacenarlo y poder pasárselo a cada uno de los objetos tipo `ProducerService` que crea sin preocuparse del ciclo de vida de la entidad a la que ese puntero hace referencia, tal y como hemos visto en el apartado anterior.

4.2.6.*Subscription Manager*

El *Subscription Manager* es un elemento fundamental en cualquier implementación de la parte servidor de SIRI en la que se den intercambios de información de tipo *Publish/Subscribe*. Debe cumplir los siguientes requisitos:

- Gestionar las subscripciones, una vez dadas de alta.
- Proporcionar al *Notification Producer* información sobre si las actualizaciones de datos afectan a alguna de esas subscripciones.

4.2.6.1. Gestión de las subscripciones: clase SubscriptionManager

Nuestra implementación del *Subscription Manager* ha sido a través de la clase `SubscriptionManager`.

El siguiente diagrama de clases nos ayudará a entender su funcionamiento:

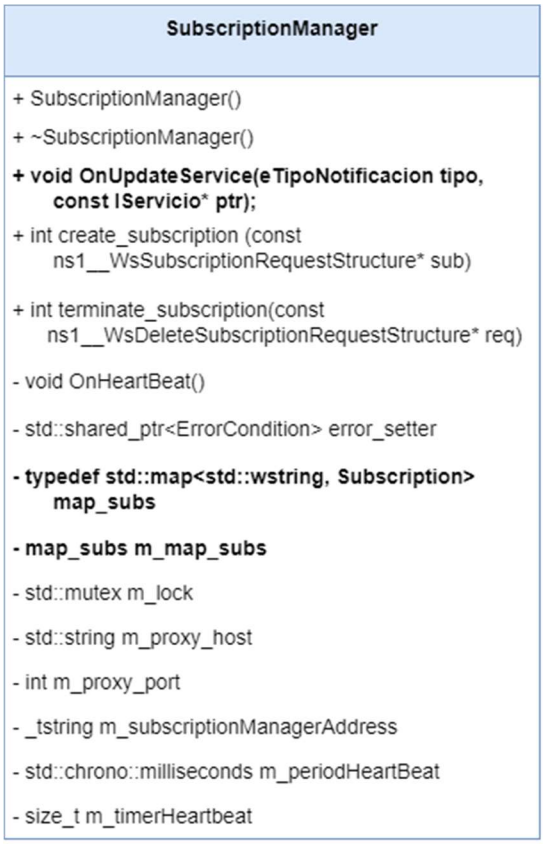


Ilustración 48: Diagrama UML de la clase SubscriptionManager

Instanciada una única vez, `SubscriptionManager` almacena en un **mapa** o diccionario el conjunto de **subscripciones activas** en cada momento. Debido al conjunto de hilos o *threads* por el que está compuesto nuestro servidor (denominado *threadpool),* es necesario proteger ese mapa de subscripciones frente al acceso concurrente, lo que se ha llevado a cabo usando un semáforo de tipo ***mutex***.

Sus métodos públicos **`create_subscription()`** y **`terminate_subscription()`** son auto-explicativos y aceptan como parámetros las estructuras de datos de las propias peticiones de SIRI, por lo que pueden ser invocados directamente desde el `ProducerService` con los datos de la petición. Su tipo de retorno (`int`) indica que estos métodos no se encargan de rellenar la respuesta que se debe enviar al cliente, quedando esa tarea fuera de la responsabilidad de nuestra clase `SubscriptionManager` (ya que tampoco está enmarcada dentro de las responsabilidades del *endpoint* lógico *Subscription Manager* de SIRI).

Como vimos en el apartado anterior, su método **`OnUpdateService (eTipoNotification, const IServicio*)`** está conectado con el núcleo del servidor de FMS, siendo el encargado de comprobar si los eventos de tiempo real del sistema afectan a alguna de las subscripciones activas. En caso de que así sea, invoca al método correspondiente de una clase llamada `NotificationSender`, encargada de rellenar la información correspondiente y enviarla al *Notification Consumer* que corresponda.

Internamente están implementados tanto los ***Heartbeats*** que se mandan a los suscriptores para indicarles el correcto funcionamiento del servidor como el control de la **caducidad de las subscripciones**.

Por último, cabe destacar que, aunque en la implementación existente en el momento en el que se están escribiendo estas líneas el servidor de SIRI esté compuesto por un solo *endpoint*, nuestro `SubscriptionManager` está preparado para ser situado en otra dirección debido a que incorpora los campos `m_proxy_host`, `m_proxy_port` y `m_subscriptionManagerAddress`.

4.2.6.2. Almacenamiento de las subscripciones: clase Subscription

La clase `SubscriptionManager` almacena en un mapa o diccionario objetos de tipo `Subscription`, entre los que itera cuando recibe una actualización de un evento. La clave de este mapa es el identificador único de cada subscripción, que en nuestro caso permitimos que genere el cliente (suscriptor) al realizar una petición de subscripción, por cuestiones de *code legacy* (código heredado); pero que podría generar el propio servidor al aceptar cada petición. Esta última opción, implementada usando UUIDs/GUIDs, es preferible sobre la primera; ya que permitiría minimizar las posibilidades de colisión en la generación de las claves, problema que podría dar lugar a que unos clientes interfirieran accidentalmente en las subscripciones de otros.

La clase `Subscription` es un **recubrimiento de la estructura autogenerada tipo `ns1__WsSubscriptionRequestStructure`**, que llega en la petición del *Subscriber* y en la que se encuentran todos los datos de la subscripción. A este recubrimiento hemos incorporado métodos que nos facilitan el acceso directo a la información más relevante de la estructura subyacente, facilitando así el

filtrado y manejo de las subscripciones que se lleva a cabo desde la clase que las gestiona, el `SubscriptionManager`.

Este es un esquema básico de la clase `Subscription`:

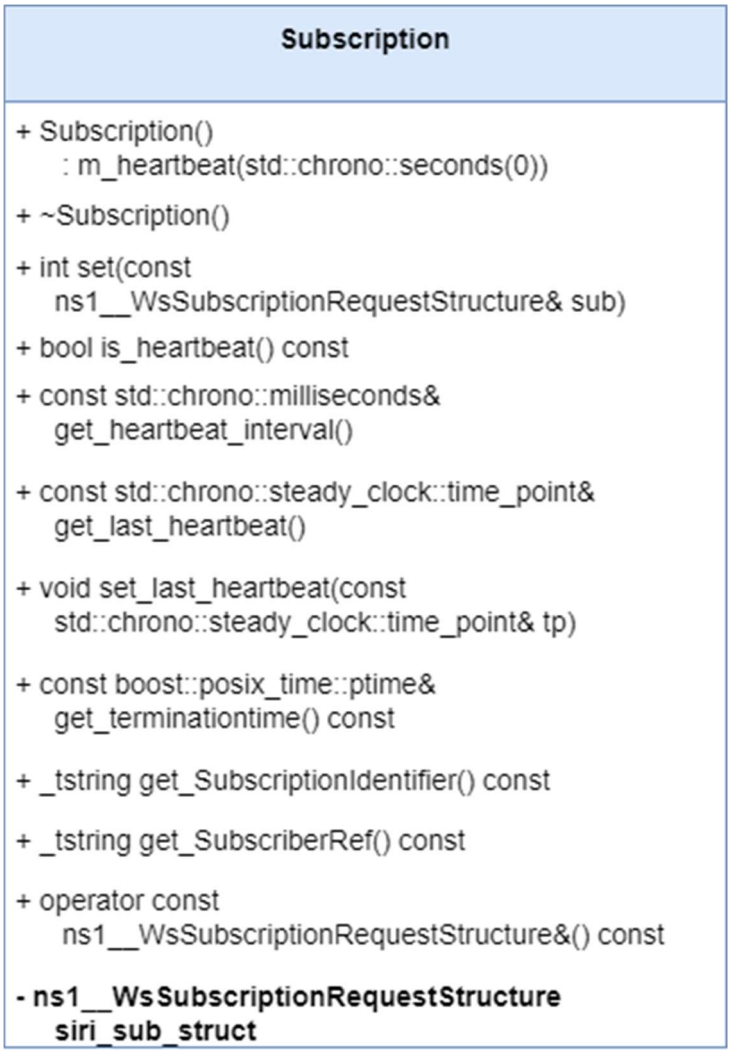


Ilustración 49: Diagrama UML de la clase Subscription

La función `set()` rellena el campo `siri_sub_struct` que hemos mencionado.

Los métodos `get_SubscriptionIdentifier() const` y `get_SubscriberRef() const` devuelven valores inalterables característicos de la subscripción y contenidos en la misma.

Los métodos relacionados con el *Heartbeat*, `get_last_heartbeat()`, `set_last_heartbeat()` e `is_heartbeat()`, bastante descriptivos en su nomenclatura, son invocados desde el método `SubscriptionManager::OnHeartbeat()` para controlar los avisos al suscriptor al que corresponde cada subscripción almacenada.

4.2.7.Clase NotificationSender

La clase `NotificationSender` está estrechamente ligada a las subscripciones, ya que como detallaremos después, nuestros servicios funcionales están implementados usando *Direct Delivery* y por tanto no necesitan del mecanismo de las notificaciones para dar respuesta a peticiones de tipo *Request/Response*.

En nuestra implementación, la **satisfacción de las subscripciones** también se lleva a cabo a través del mecanismo de ***Direct Delivery***, por lo que el cliente recibe los datos a los que está suscrito en el mismo mensaje en el que se le notifica la existencia de nuevos datos disponibles.

Esta figura, análoga a la mostrada en su momento en el apartado teórico, ilustra dicho mecanismo.

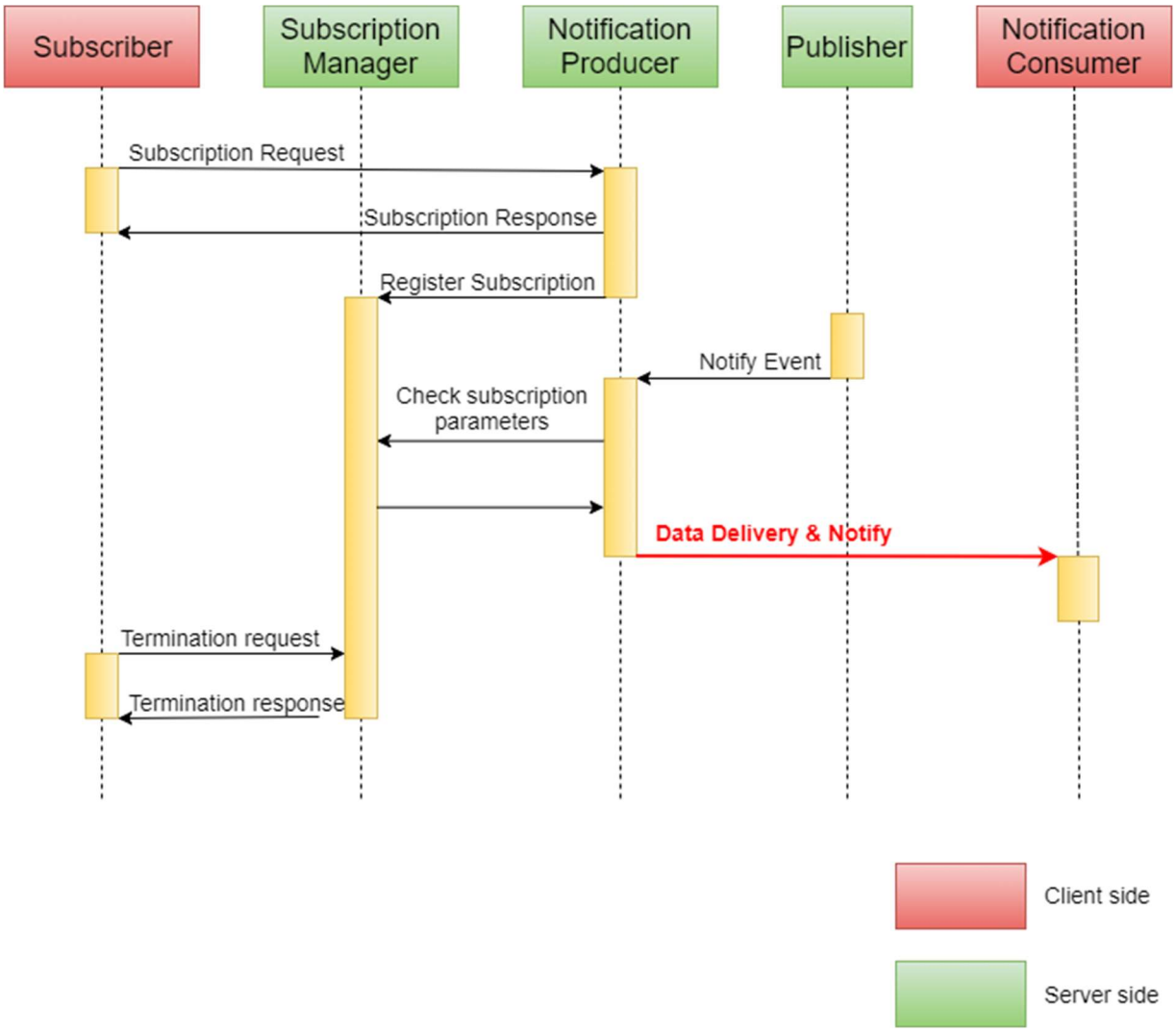


Ilustración 50: Patrón Direct Delivery en un intercambio de datos tipo Publish/Subscribe

En el `SubscriptionManager` se almacenan el conjunto de subscripciones activas mediante un diccionario, cuya clave se corresponde con el identificador de cada subscripción y cuyo valor es una instancia de tipo `Subscription`, que como hemos visto es un envoltorio de la clase `ns1__WsSubscriptionRequestStructure`.

En los eventos/novedades del servidor de FMS no se incluye toda la nueva información, por motivos de rendimiento. Por tanto, cuando la instancia del `SubscriptionManager` determina que **una subscripción debe ser satisfecha**, es necesario **recopilar esa información nueva o modificada para mandársela al *Notification Consumer* correspondiente**.

De dicha tarea se encarga la clase **`NotificationSender`**, cuya fachada se corresponde con este diagrama:

Ilustración 51: Estructura de clase NotificationSender

Para comprender la estructura de esta clase hay que tener en cuenta lo siguiente:

- Una petición simple (*Request* en un intercambio tipo *Request/Response*) se compone de dos partes fundamentales:
  - Estructura tipo `ns1__WsServiceRequestInfoStructure`, en la que se encuentra la información común a cualquier tipo de intercambio de datos en SIRI: autenticación, direcciones de los distintos *endpoints* de cliente y servidor, idioma, hora de la petición, etc.
  - Estructura tipo `ns2__XXXXXXXXRequestStructure`, característica de cada servicio funcional de SIRI y donde `XXXXXXXX` se corresponde con el nombre del servicio o está basado en el mismo.
- Análogamente, las peticiones de subscripción de SIRI están compuestas por:
  - Estructura tipo `ns1__WsSubscriptionRequestStructure`, en la que se encuentran los datos comunes a cualquier subscripción: la autenticación, los *endpoints* de *Subscriber*, *Notification Consumer* y servidor, el idioma, etc.
  - Estructura tipo `ns2__XXXXXXXXRequestStructure`, igual que la ya descrita en el párrafo anterior.

Dada la similitud entre las estructuras `ns1__WsServiceRequestInfoStructure` y `ns1__WsSubscriptionRequestStructure`, y teniendo en cuenta que la otra estructura es compartida por ambos tipos de peticiones, basta con transformar en el `NotificationSender` la primera en la segunda para poder **componer una petición simple a partir de una subscripción**.

Si además tenemos en cuenta que las estructuras de los mensajes de satisfacción de subscripciones son las mismas que las estructuras empleadas en la respuesta de una petición directa (*Request/Response*), podemos **usar el mismo código para obtener la información con la que satisfacer las subscripciones que el que usaríamos para rellenar una respuesta simple a partir de la *request* pertinente.**

Eso justifica la existencia de la siguiente función privada, `private ns1__WsServiceRequestInfoStructure* SubInfo2RequestInfo(ns1__WsSubscriptionRequestStructure&)`, que es invocada por cada método público `send_NotifyXXXXXXXX(…)`.

Las excepciones a lo anterior vienen dadas por las notificaciones no asociadas a servicios funcionales de SIRI: `send_NotifySubscriptionTerminated()`, `send_NotifyDataReady()` y `send_NotifyHeartbeat()`. Cada una de estas notificaciones posee su estructura interna característica, que ha de crearse en cada caso a partir de la información de la subscripción almacenada a la que se hace referencia.

4.2.8.Servicios funcionales de SIRI: ProducerService

La clase `ProducerService` es la que agrupa la **resolución de todas las llamadas del cliente o *Consumer***.

Ante la llegada de una petición al método `Producer::process(struct soap* soap)`, se instancia una clase de nuestro `ProducerService` con ese puntero a una estructura de tipo `soap` y se ejecuta su método **`dispatch()`**, que **se encarga de invocar internamente al método correcto de `ProducerService`** en función de la petición recibida.

Este método `dispatch()` se encuentra implementado en la clase autogenerada `SiriProducerDocBindingService`, de la que hemos hecho heredar `ProducerService`. Dicha herencia nos obliga a implementar todos los métodos virtuales puros de `SiriProducerDocBindingService`, véase:

- La función `SiriProducerDocBindingService* copy()`.
- Los métodos de servicios auxiliares `CheckStatus()`, `GetCapabilities(…)`, `GetDataSupply(…)`, `LinesDiscovery(…)`, `StopPointsDiscovery(…)` y `ConnectionLinksDiscovery(…)`.
- Las funciones `Subscribe(…)` y `DeleteSubscription(…)`.
- Los métodos encargados de resolver los servicios funcionales de SIRI:
  - `GetProductionTimetable(…)`, `GetEstimatedTimetable(…)`.
  - `GetStopTimetable(…)`, `GetStopMonitoring(…)`.
  - `GetMultipleStopMonitoring(…)`.
  - `GetConnectionTimetable(…)`, `GetConnectionMonitoring(…)`
  - `GetVehicleMonitoring(…)`, `GetFacilityMonitoring(…)`.
  - `GetServiceExchange(…)`.
  - `GetGeneralMessage()`.

Esta es la implementación (autogenerada) del método `dispatch()`:

```
int SiriProducerDocBindingService::dispatch() { return dispatch(this->soap); }

int SiriProducerDocBindingService::dispatch(struct soap* soap)
{
	SiriProducerDocBindingService_init(soap->imode, soap->omode);

	soap_peek_element(soap);
	if (!soap_match_tag(soap, soap->tag, "ns1:Subscribe"))
		return serve___ns1__Subscribe(soap, this);
	if (!soap_match_tag(soap, soap->tag, "ns1:DeleteSubscription"))
		return serve___ns1__DeleteSubscription(soap, this);
	if (!soap_match_tag(soap, soap->tag, "ns1:DataSupply"))
		return serve___ns1__DataSupply(soap, this);
	if (!soap_match_tag(soap, soap->tag, "ns1:CheckStatus"))
		return serve___ns1__CheckStatus(soap, this);
	if (!soap_match_tag(soap, soap->tag, "ns1:GetProductionTimetable"))
		return serve___ns1__GetProductionTimetable(soap, this);
	// ...
}
```

La explicación anterior se puede resumir en el siguiente diagrama UML:

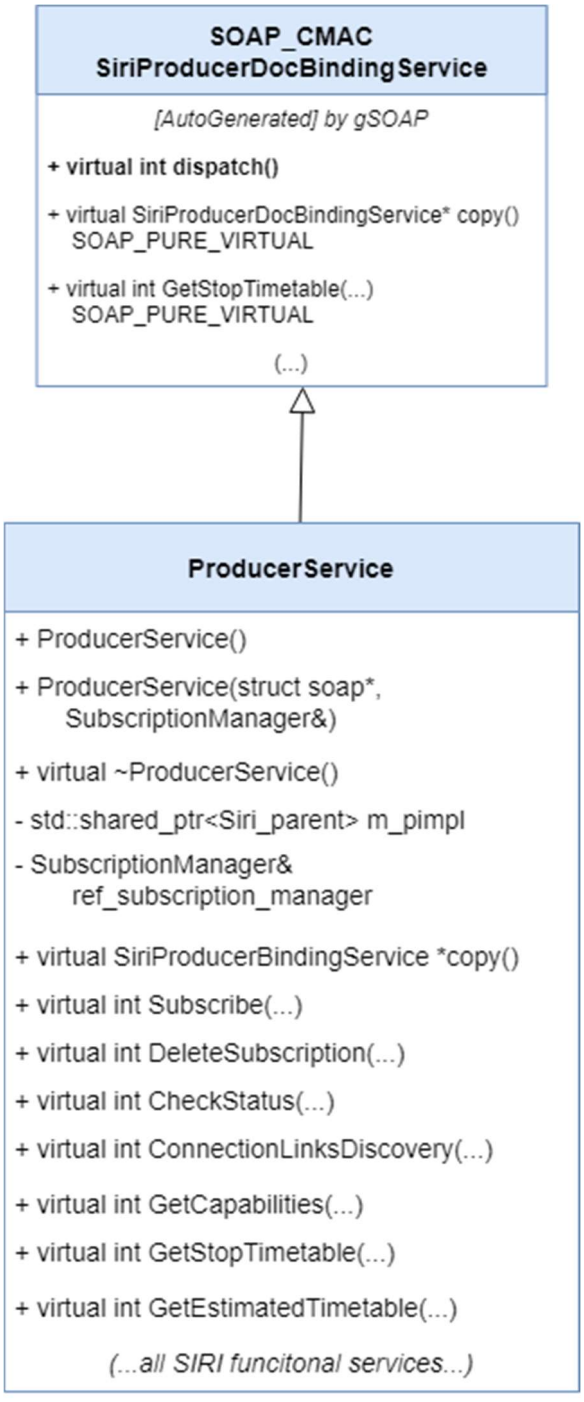


Ilustración 52: Diagrama UML de la clase ProducerService

Nótese que `stdsoap2.h` nos provee del siguiente *macro*:

```
#ifdef WITH_PURE_VIRTUAL
# define SOAP_PURE_VIRTUAL = 0
#else
# define SOAP_PURE_VIRTUAL
#endif
```

Cada uno de los métodos que se corresponden con servicios funcionales de SIRI, además de los métodos asociados a servicios auxiliares, instancian a su vez una clase encargada de procesar la petición y rellenar la respuesta correspondiente.

La excepción a lo anterior se da en los métodos `Subscribe(…)` y `DeleteSubscription(…)`, que como ya hemos visto invocan a métodos de la clase *Singleton* `SubscriptionManager` a la que tiene acceso cada instancia de `ProducerService`.

Dejando de lado estas excepciones, el aspecto del resto de métodos es muy similar entre sí. Veamos uno de ellos como ejemplo:

```
/// Web service operation 'GetProductionTimetable' (returns SOAP_OK or error
code)
int ProducerService::GetProductionTimetable(
ns1__ProductionTimetableRequestStructure* ns1__GetProductionTimetable,
ns1__ProductionTimetableAnswerStructure& ns1__GetProductionTimetableResponse)
{
	assert(ns1__GetProductionTimetable != nullptr);

	CheckerCredentialsBasic checker;

	Siri_Production_Timetable* instance = new
Siri_Production_Timetable(checker);

	instance->fill(*ns1__GetProductionTimetable,
ns1__GetProductionTimetableResponse);

	m_pimpl.reset(instance);

	return SOAP_OK;
}
```

Lo que sucede es lo siguiente:

- Se instancia un objeto tipo `CheckerCredentialsBasic`, que se usará en todos los servicios de SIRI para comprobar que las peticiones incluyen las credenciales necesarias para acceder a dichos servicios (autenticación).
- Se instancia un objeto tipo `Siri_Production_Timetable`, la clase encargada de procesar las peticiones de PT, pasándole el comprobador de credenciales previamente instanciado.
- Se invoca el método `fill(const request_structure&, answer_structure&)`, encargado de procesar la petición. En nuestro caso, `Siri_Production_Timetable::int fill( const ns1__ProductionTimetableRequestStructure& request, ns1__ProductionTimetableAnswerStructure& response);`
- Se almacena el puntero al objeto de la clase encargada del procesado de las peticiones (`Siri_Production_Timetable`) en el miembro `std::shared_ptr<Siri_parent> m_pimpl`, no mencionado hasta ahora.

Se sobreentiende que, al almacenarse cada una de estas clases que procesan peticiones de SIRI y rellenan la respuesta en el puntero inteligente `std::shared_ptr<Siri_parent> m_pimpl`, todas ellas heredan de otra a la que hemos denominado `Siri_parent`.
El almacenamiento del puntero al objeto de cada una de estas clases en `m_pimpl` permite que dicha clase no se destruya al salirse la ejecución del *scope* del método que la ha instanciado; sino que liga su ciclo de vida al de `m_pimpl`.
Otra manera de conseguir lo anterior, pero evitando usar 'punteros crudos' (*raw pointers*) es mediante `std::make_unique()`(C++14) y `std::move()` (C++11):

```
auto pInstance = std::make_unique<Siri_Production_Timetable>(checker);

pInstance->fill(*ns1__GetProductionTimetable,
ns1__GetProductionTimetableResponse);

m_pimpl = std::move(pInstance);
```

Si no pudiéramos adoptar el estándar C++14 y tuviéramos que conformarnos con el C++11, la función `make_unique()` la tendríamos disponible como parte de la librería Boost.

`m_pimpl` no será destruido hasta que no se llame al destructor de la clase a la que pertenece, `ProducerService`, lo que no sucede hasta que la ejecución no abandona el *scope* en el que fue instanciada.
Recordemos dónde tiene lugar la instanciación del `ProducerService`:

```
Producer::virtual int process(struct soap* _soap)      // Connection
{
        ProducerService sl(_soap, *ptr_m_subscription_manager);

        return sl.dispatch();
}
```

Todo esto se traduce en que **las clases que procesan las peticiones de SIRI** (y lo realmente importante, **la información de respuesta** que ha sido rellenada dentro de esas clases) **no se destruyen hasta que la petición no ha sido debidamente respondida.** Esto es especialmente relevante si tenemos en cuenta que **las estructuras SIRI autogeneradas por gSOAP para C++ están compuestas de punteros**, por lo que para rellenarlas es necesario ir reservando memoria. Dicha memoria debe ser borrada una vez se completan los envíos de respuestas al cliente, evitando caer en *memory leaks*.
El siguiente apartado está dedicado en exclusiva a explicar esa gestión de la memoria.

4.2.9. Gestión de la memoria

Veamos un ejemplo extraído de la clase `SIRI_Production_Timetable`, encargada del procesamiento de las peticiones de PT:

```
// Expeditions
ns2__DatedVehicleJourneyStructure* datedvehiclejourney_structure = new
ns2__DatedVehicleJourneyStructure;

const auto v_stops = service->getParadasBase();
for (const auto& stop : v_stops)
{
	// Stops
	ns2__DatedCallStructure* datedcall_structure = new
	ns2__DatedCallStructure;

	ns2__StopPointRefStructure* stoppoint_ref = new
	ns2__StopPointRefStructure;

	std::wstringstream ss;
		ss.imbue(std::locale::classic());
		ss << stop.idParada;
		stoppoint_ref->__item = ss.str();

	datedcall_structure->StopPointRef = stoppoint_ref;

	boost::posix_time::ptime _arrival_time(/*...*/);
	datedcall_structure->AimedArrivalTime = new  xsd__dateTime__(
		GSOAP::from_ptime(_arrival_time));

	boost::posix_time::ptime _departure_time(/*...*/);
	datedcall_structure->AimedDepartureTime = new  xsd__dateTime__(
		GSOAP::from_ptime(_departure_time));

	// ...

   datedvehiclejourney_structure->DatedCalls.DatedCall
	.push_back(datedcall_structure);
}
```

Recordemos que en una respuesta de PT pueden existir varias expediciones, cada una de ellas con varias paradas, y cada una de esas paradas con atributos como el identificador de parada, el tiempo de llegada y partida, etc.

En la estructura autogenerada de esta respuesta, esto se traduce en la existencia de:

- Un vector de punteros de estructuras de expediciones, de tipo `ns2__DatedVehicleJourneyStructure*`.
- Un vector de punteros de estructuras de paradas, de tipo `ns2__DatedCallStructure`, por cada una de las estructuras anteriores.
- Punteros a otras estructuras de datos (`ns2__StopPointRefStructure`, `xsd__dateTime__* arrival_time`, `xsd__dateTime__* departure_time`, etc.) por cada una de esas estructuras de parada.

Lo que estamos haciendo en el fragmento de código anterior es reservar memoria para una estructura de expedición, dentro de ella almacenar punteros a estructuras de paradas, y dentro de cada una de estas almacenar punteros a otras estructuras.

Este esquema intenta reflejar la situación:

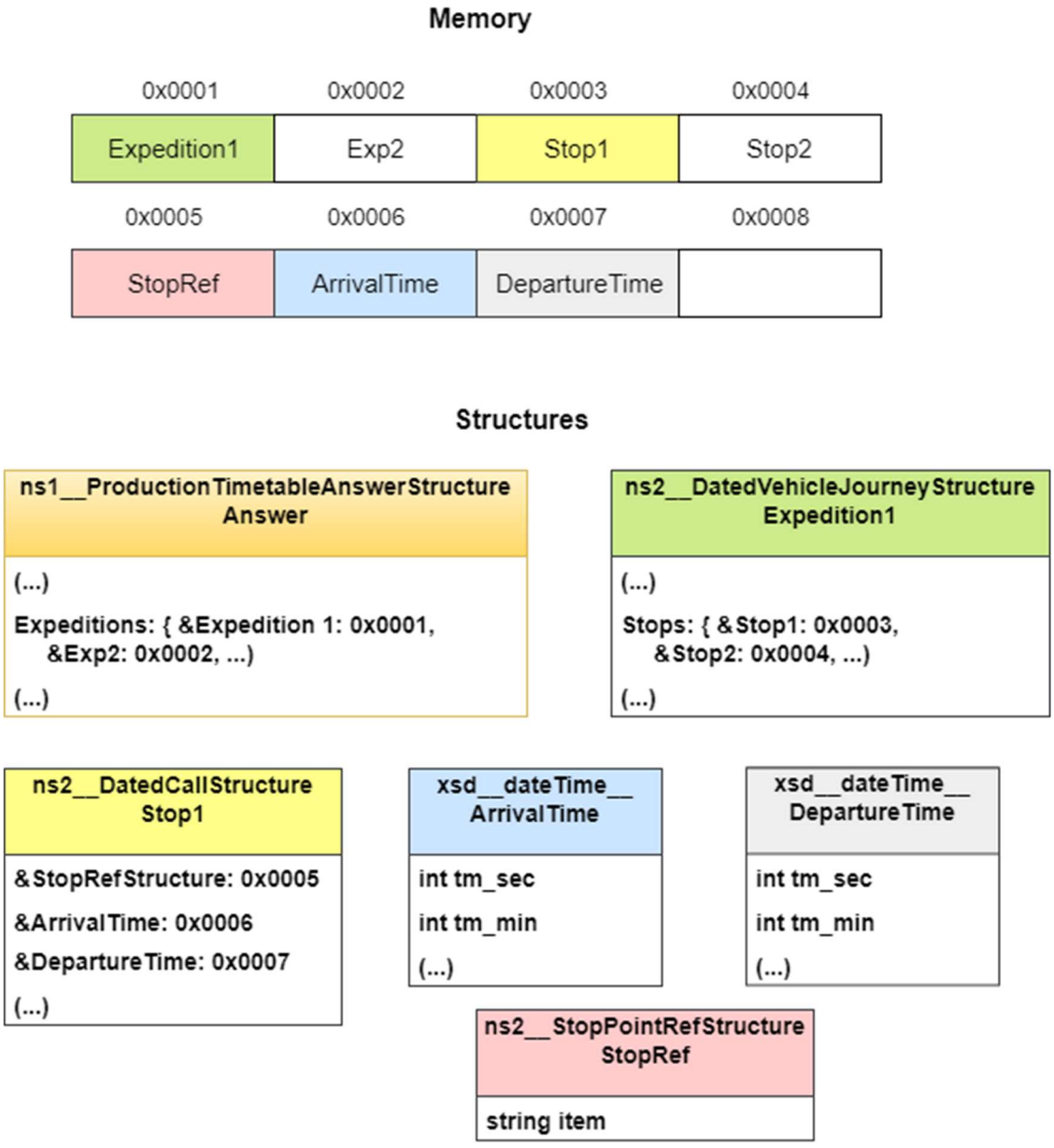


Ilustración 53: Diagrama (parcial) de la jerarquía de estructuras en Production Timetable

Si borráramos la memoria de la estructura de nivel superior (expedición) antes de haber borrado la memoria a la que apuntan los punteros que ésta contiene (estructuras de paradas), sucedería lo siguiente:

- El sistema lanzaría excepciones de referencia nula, al no poder acceder a las direcciones de memoria de las paradas que se almacenaban en la flamantemente borrada estructura de expedición.

- Lo anterior evitaría que se borrase la memoria de las estructuras de tipo parada, lo que se propagaría en cascada evitando que se borrara a su vez la memoria de las estructuras cuya dirección está almacenada en la estructura de parada. Dado que no podríamos acceder a las direcciones de memoria que hemos reservado y ocupado al haber perdido sus referencias, no tendríamos forma de liberar dicha memoria, produciéndose lo que se conoce como *memory leaks* (fugas de memoria).

La manera más sencilla de evitar lo anterior es controlar nosotros mismos el borrado de la memoria, prestando especial atención al orden en el que se efectúa este.

Los objetivos a lograr son los siguientes:

- Almacenar en memoria los datos de las respuestas el tiempo suficiente como para que se envíe la información al cliente.
- Borrar la memoria en un orden consistente para que la jerarquía de punteros que componen la respuesta no provoque *memory leaks*.

En el caso de `ProductionTimetable`, la lógica a seguir sería la siguiente:

- La función encargada de rellenar la respuesta es:
  `Siri_Production_Timetable::int fill( const ns1__ProductionTimetableRequestStructure &request, ns1__ProductionTimetableAnswerStructure &response)`
- Por tanto, **la estructura a rellenar** ante una petición de PT es `ns1__ProductionTimetableAnswerStructure`, que a su vez **está compuesta por dos punteros a sendas subestructuras**: `ns2__ProducerResponseEndpointStructure*` y `ns2__ProductionTimetableDeliveriesStructure*` (ésta última es la llamada *payload* en la descripción del servicio).
- Si en vez de simplemente instanciar ambas estructuras al comienzo del método `fill(…)`, guardándolas en el *heap* y perdiéndolas al abandonar la ejecución ese método, las instanciamos y las guardamos en memoria (*stack*) y **almacenamos un puntero a esas direcciones de memoria en la propia clase `Siri_Production_Timetable`**, estaremos evitando su destrucción temprana.

  Y si además usamos punteros inteligentes (`std::unique_ptr`) para ello, en principio nos evitaríamos tener que preocuparnos de liberar manualmente esa memoria, ligando la vida de la misma a la vida de `Siri_Production_Timetable`.
- Dado que esa memoria debe conservarse al menos hasta que el código generado por gSOAP haya parseado su contenido al XML correspondiente, debemos buscar una solución similar para que la instanciación de `Siri_Production_Timetable` creada en el método `ProducerService::GetProductionTimetable(…)` sobreviva al mismo.

- Esa solución puede ser **guardar la instancia de `Siri_Production_Timetable` en el *stack* y almacenar un puntero a su dirección de memoria en `ProducerService`**, a través de un puntero inteligente tipo `std::shared_ptr`.
- Como sucederá esto con cada instanciación de las distintas clases que usaremos para rellenar las respuestas de los distintos servicios, podemos hacer heredar todas ellas de `Siri_Parent` y usar un mismo campo privado para todas ellas: **`std::shared_ptr<Siri_parent> m_pimpl;`**

Con los pasos anteriores habríamos conseguido el primero de los objetivos: persistir las subestructuras que componen la respuesta a un servicio hasta que éste se resuelva. Dado que `ProducerService` se instancia en `int Producer::process(struct soap* _soap)` y la respuesta se envía al cliente antes de abandonar la ejecución esa función, el destructor de `Siri_Production_Timetable` y los destructores de sus miembros `unique_ptr<ns2__ProductionTimetableDeliveriesStructure> answer` y `unique_ptr<ns2__ProducerResponseEndpointStructure> service_info` se invocarían justo después del envío de la información.
Sin embargo, al estar las dos estructuras mencionadas compuestas de más estructuras de punteros, **no podemos dejar que los destructores de `std::unique_ptr` las borren sin más**, ya que como hemos visto más arriba perderíamos las referencias a los contenidos de su interior.
Vamos a plantear tres soluciones a este segundo problema:

- **La solución más simple es**:
  En el destructor de `Siri_Production_Timetable`, recorrer manualmente la jerarquía de punteros que hemos ido instanciando bajo las estructuras a las que apuntan los miembros `answer` y `service_info` e ir liberando explícitamente la memoria en el orden jerárquico correcto, teniendo en cuenta lo siguiente:
  - El valor del operador `delete` puede ser un puntero nulo, por lo que no es necesario comprobar la no-nulidad de cada una de las estructuras a las que aplicar el `delete` antes de hacerlo [20].
  - Por el contrario, es imprescindible comprobar la no-nulidad de aquellas estructuras que contengan bajo ellas punteros a los que se pretenda acceder, antes de realizar ese acceso.
    Por ejemplo, si tuviéramos una jerarquía entre objetos `X`, `Y` y `Z` tal que `X` contuviera un puntero de `Y`, e `Y` un puntero de `Z`, para acceder a `Z` podríamos usar `X->Y->Z`.
    Sin embargo, si `X` o `Y` fueran punteros nulos, la instrucción anterior lanzaría una excepción.

Un ejemplo del destructor de `Siri_Production_Timetable` en caso de que se opte por la solución anterior podría ser el siguiente:

```
Siri_Production_Timetable::~Siri_Production_Timetable()
{
  if (answer)
  {
    for (auto& line : answer->ProductionTimetableDelivery)
    {
      if(line->ValidUntil)
        delete line->ValidUntil;

      for (auto& trayecto : line->DatedTimetableVersionFrame)
      {
        delete trayecto->LineRef;
        delete trayecto->DirectionRef;
        delete trayecto->RouteRef;

        for (auto& expedition : trayecto->DatedVehicleJourney)
        {
          for (auto& stop : expedition->DatedCalls.DatedCall)
          {
            delete stop->StopPointRef;
            delete stop->AimedArrivalTime;
            delete stop->AimedDepartureTime;
            delete stop->AimedHeadwayInterval;

            //...

            delete stop;
          }

          //...

          delete expedition;
       }

        //...

        delete trayecto;
      }

      //...

      delete line;
    }

    //...

  }
}
```

La gran desventaja de esta solución reside en que la inmensidad de las estructuras de respuesta de SIRI hace imposible en la práctica cubrir la liberación de la memoria reservada por todas las posibles subestructuras, por lo que deben borrarse sólo las que se instancien (y confiar en que ningún desarrollador instancie en el futuro una sin tenerlo en cuenta en el destructor correspondiente).

El problema de 'tener que recordar' el añadir el `delete` correspondiente al destructor puede solucionarse con punteros inteligentes, pero dado que gSOAP no ofrece la posibilidad de autogenerar las macroestructuras de SIRI con punteros inteligentes**, otra posible solución** pasaría por almacenar como miembros privados de todas aquellas clases en las que se reserve memoria (como `Siri_Production_Timetable`, siguiendo con nuestro ejemplo) punteros del estilo `unique_ptr<tipo_de_estructura>`, uno por cada instanciación que realicemos.

La cantidad de campos existentes en la estructura convierte esta solución en algo prácticamente inviable (a pesar de ser más segura que la anterior), ya que implicaría almacenar y mantener entre 25 y 100 variables privadas por clase.

La tercera solución, adoptada finalmente en la implementación final del servidor tras pasar inicialmente por la primera, se basa en una idea muy simple que oculta tras de sí una implementación de un alto orden de complejidad.

A grandes rasgos, consiste en crear punteros inteligentes tipo `std::shared_ptr` de todas las estructuras que se vayan necesitando y almacenarlos todos ellos en un vector de tipo **`std::vector<boost::any>`**.

Si almacenamos ese vector como un miembro privado de cada una de las clases del estilo de `Siri_Production_Timetable`, cuando éstas sean destruidas se invocará a su vez al destructor del vector y a los destructores de todo lo que éste contiene.

Para la implementación de esta solución se han utilizado funciones como:

```
template<typename T, typename Store, class... _Types>
T* new_store(Store& store, _Types&&... _Args)
{
	std::shared_ptr<T> ptr =
std::make_shared<T>(std::forward<_Types>(_Args)...);
	T* res = ptr.get();
	store.push_back(ptr);

	return res;
}


template<typename Store, typename T>
T* copy_store(Store& store, const T* other)
{
	if (other)
	{
		std::shared_ptr<T> ptr = std::make_shared<T>(*other);
		T* res = ptr.get();
		store.push_back(ptr);
		return res;
	}

	return nullptr;
}
```

```
template<typename T, typename Store>
T* new_store(Store& store)
{
	std::shared_ptr<T> ptr = std::make_shared<T>();
	T* res = ptr.get();
	store.push_back(ptr);

	return res;
}

template<typename T, typename Store, typename O>
T* new_store_item(Store& store, const O& obj)
{
	std::shared_ptr<T> ptr = std::make_shared<T>();
	ptr->__item = obj;
	T* res = ptr.get();
	store.push_back(ptr);

	return res;
}
```

4.2.10. Servicios implementados

Pasan a describirse brevemente los servicios implementados y, de existir, sus características más destacables. La decisión de implementación de los mismos y sus citadas características responden tanto al estudio teórico sobre SIRI reflejado en este documento como a las necesidades y al uso que hacen de ellos de los clientes de la empresa para la que fue desarrollado el servidor.

Por razones de confidencialidad, el contenido de la mayoría de las clases que procesan las peticiones de SIRI no puede ser expuesto en este documento, ya que para rellenar las respuestas de los servicios funcionales se hace uso del modelo de datos de ITS interno de la empresa.

No obstante, en el anexo 3 a este documento se adjuntan logs en los que se pueden apreciar las características de la información generada por los principales los servicios funcionales.

Todos los servicios implementados siguen el patrón de entrega *Direct Delivery*.

4.2.10.1. *Production Timetable* (PT)

El servicio de *Production Timetable* se ha implementado siguiendo tanto el patrón de intercambio de datos directo (*Request/Response*) como el patrón de subscripciones (*Publish/Subscribe*), siendo uno de los más completos al ser también uno de los más usados en la operativa diaria de los clientes.

Debido a la peculiaridad de las estructuras de respuesta de PT, orientadas al usuario final y ya explicadas en el apartado teórico correspondiente, su rellenado puede resultar bastante complejo

Se permiten filtros tanto por línea como temporales, cargando por defecto los horarios correspondientes a las veinticuatro horas siguientes a la petición del cliente, en caso de que no se especifique lo contrario en dicha petición.

4.2.10.2. *Estimated Timetable* (ET)

El servicio de *Estimated Timetable*, que complementa a PT siendo su análogo para la transmisión de modificaciones de los horarios en tiempo real, se ha implementado siguiendo la línea de éste.

También se permiten los filtros temporales y por línea, teniendo el valor por defecto del llamado *Validity Period* la misma duración que en el caso de *Production Timetable*, veinticuatro horas.

4.2.10.3. *Stop Timetable* (ST)

Al igual que los dos anteriores, el servicio *Stop Timetable*, enfocado a la distribución de información estática de paradas, es uno de los más demandados y por tanto de los más completos. También está disponible tanto a través de peticiones simples como por subscripciones.

Permite filtros temporales, repitiéndose el valor por defecto del *Validity Period* de los servicios anteriores, y obliga a establecer un filtro por parada. Esto se debe a la gran cantidad de información no deseada con la que se encontraría un cliente que no estableciera dicho filtro, si se devolviera la información de todas las paradas contempladas en el sistema. No obstante, ésta lógica interna se mantiene y el servicio puede ser configurado para devolver la información estática de todas las paradas, lo que se puede aprovechar para realizar tests.

4.2.10.4. *Stop Monitoring* (SM)

Con características similares a ST en cuanto a importancia y a los modos a través de los cuales está disponible, nuestra implementación de *Stop Monitoring* se caracteriza además por dar servicio a la función `SiriProducerDocBindingService::GetMultipleStopMonitoring(…)`, marcada como descatalogada (*deprecated*) a partir de la versión 2.0 de SIRI pero necesaria en nuestro caso para mantener la compatibilidad con servidores existentes.

Los filtros son análogos a los de *Stop Timetable*, pero sin permitir en ningún caso la no concreción del filtro de paradas, ya que la mencionada petición `GetMultipleStopMonitoring(…)` puede ser usada para los citados tests.

4.2.10.5. *Vehicle Monitoring* (VM)

Vehicle Monitoring es el otro estandarte de los servicios funcionales de SIRI, junto a las dos parejas anteriores, por lo que de nuevo implementa la posibilidad de proporcionar información a través de peticiones tanto de tipo *Request/Response* como de tipo *Publish/Subscribe.*

Además de los característicos filtros temporales, permite otros por línea y dirección; y por defecto devuelve información sobre las cinco paradas futuras por las que está planificado que pase cada vehículo sobre el que se informa.

4.2.10.6. *Facilities Monitoring* (FM)

Este polifacético servicio ha sido implementado para proporcionar información únicamente a través de intercambios de tipo *Request/Response*.

Son muchos los datos que es posible intercambiar con él, estando la versión inicial preparada para devolver las listas de líneas y paradas disponibles en el sistema (como alternativa a *Lines* y *StopPoints Delivery*).

Ha quedado implementado y en funcionamiento, listo para ser completado en el futuro y personalizado según las necesidades de cada cliente.

4.2.10.7. *Situation Exchange* (SX)

Este servicio se ha implementado de manera análoga al anterior, sólo accesible a través de *Request/Response* y sin estar tan completo como los descritos previamente. Esto se debe a que no es un servicio tan demandado y al necesitarse de él un número muy reducido de alertas en comparación con todas las funcionalidades que es capaz de ofrecer.

4.2.10.8. *Capabilities*

Este servicio no funcional de SIRI que tan imprescindible resulta en intercambios de datos entre servidores de dos entidades distintas está disponible a través de peticiones simples (*Request/Response*).

Esto es así al considerase que las características que ofrece un servidor de SIRI son suficientemente estáticas como para no merecer la pena ofrecer la vía de la subscripción a los clientes, que rara vez verían modificaciones en las 'capacidades' de su servidor.

4.2.10.9. *CheckStatus*

La respuesta a las peticiones *CheckStatus* de los clientes, usadas para comprobar el estado del servidor, también ha sido desarrollada.

4.2.10.10. *Delivery Services*

Nuestra implementación del servidor incluye los servicios no funcionales *Lines Discovery* y *StopPoints Discovery*, que devuelven la lista de líneas y paradas disponibles en el servidor, respectivamente.

4.2.10.11. Notificaciones

Nuestro servidor incluye las notificaciones básicas:

- *Heartbeat*: es capaz de informar periódicamente a los distintos suscriptores de su (correcto) estado de funcionamiento.
- *NotifyTerminateSubscription*: es capaz de informar a los suscriptores de cuándo una de sus subscripciones ha caducado.
- *NotifyXXXXXXX*: es capaz de informar a los *Notification Consumers* pertinentes de la existencia de nueva información de su interés en relación con alguna subscripción.
  Las notificaciones de dicha información, que al usar entregas tipo *Direct Delivery* incluyen el *payload* de los datos actualizados, aprovechan las mismas clases que rellenan las peticiones de los servicios funcionales: `Siri_Production_Timetable`, `Siri_Vehicle_Monitoring`, etc.

4.3. Implementación de un cliente para SIRI

4.3.1. Introducción

Paralelamente al desarrollo de la parte servidor de SIRI se implementó la parte cliente de dicho estándar.

Además de cubrir la necesidad existente de un cliente capaz de extraer datos de otros *endpoints* de SIRI, el desarrollo en paralelo del mismo ayudó a validar la propia implementación del servidor.

En el anexo 2 se puede consultar el código del cliente de SIRI implementado.

4.3.2. Uso de Googletest

Para la validación de las respuestas obtenidas del servidor se ha elegido Googletest [21] como plataforma o *framework* de testeo.

El procedimiento de **TDD** (*Test Driven Development*) usado fue el siguiente:

- Para cada punto de entrada a nuestro servidor se definía en el cliente un conjunto de tests que comprobaba el comportamiento del primero en base a los requisitos de funcionalidad requeridos (extraídos tanto de proyectos existentes como de peticiones de nuevos clientes).
- Se implementaba esa funcionalidad en el servidor.
- Con el servidor arrancado, se comprobaba que el conjunto de tests previamente definido se ejecutaba correctamente y cubría de manera suficiente los requisitos funcionales previamente descritos.

Debido a que la información recibida a través de las subscripciones se genera internamente en el servidor usando las mismas clases que se usan para rellenar las peticiones directas/simples, y que todos los servicios implementados que aceptan subscripciones están preparados también recibir este otro tipo de peticiones; bastaría con incluir Googletest en las peticiones simples (*Request/Response*) para llevar a cabo las comprobaciones pertinentes que un *framework* como el elegido nos permite.

Como veremos, el cliente implementado está enfocado a la ejecución de los tests en vez de a la usabilidad del mismo por parte de una persona real: no ofrece un menú gráfico ni de consola al usuario para realizar peticiones, sino que simplemente ejecuta la batería de tests existentes. Esto es así debido a la ausencia de una necesidad real de ofrecer esta utilidad a un usuario final.

No obstante, las clases que se prueban con Googletest pueden ser fácilmente extraídas y usadas en cualquier otro contexto, como en el de la realización de peticiones a *endpoints* públicos de SIRI.

Asimismo, es trivial capturar los XML generados por el cliente y adaptarlos a otras peticiones que se deseen realizar.

4.3.3.Arquitectura general del cliente de SIRI o *Consumer*

Como ya se ha expuesto en este documento, son múltiples los *endpoints* posibles en una implementación de un cliente de SIRI. Algunos de ellos son:

- *Requestor*: se encarga de las peticiones simples de datos (de tipo *Request/Response*).
- *Data Consumer*: recibe los datos de esas peticiones *Request/Response*.
- *Subscriber*: se encarga de solicitar al servidor subscripciones a sus servicios y de cancelarlas.
- *Notificacion Consumer*: se encarga de recibir las notificaciones del servidor relacionadas con las subscripciones (lo que puede incluir el *payload* de las mismas en caso de *Direct Delivery*).
- *Endpoint* en el que recibir los *Heartbeats* del servidor indicando su correcto funcionamiento.

Nuestra implementación comprende **dos *endpoints***, lo que se traduce en dos programas independientes:

- Uno de ellos engloba al *Requestor*, al *Data Consumer* y al *Subscriber;* y fue denominado simplemente **SIRI_Client**. Se encarga de realizar peticiones al servidor y de recibir las respuestas a peticiones simples.
- El otro cumple el papel del *Notification Consumer,* procesando además otras notificaciones del servidor, como los *Heartbeats*. Fue denominado **SIRI_Notification_Consumer.**

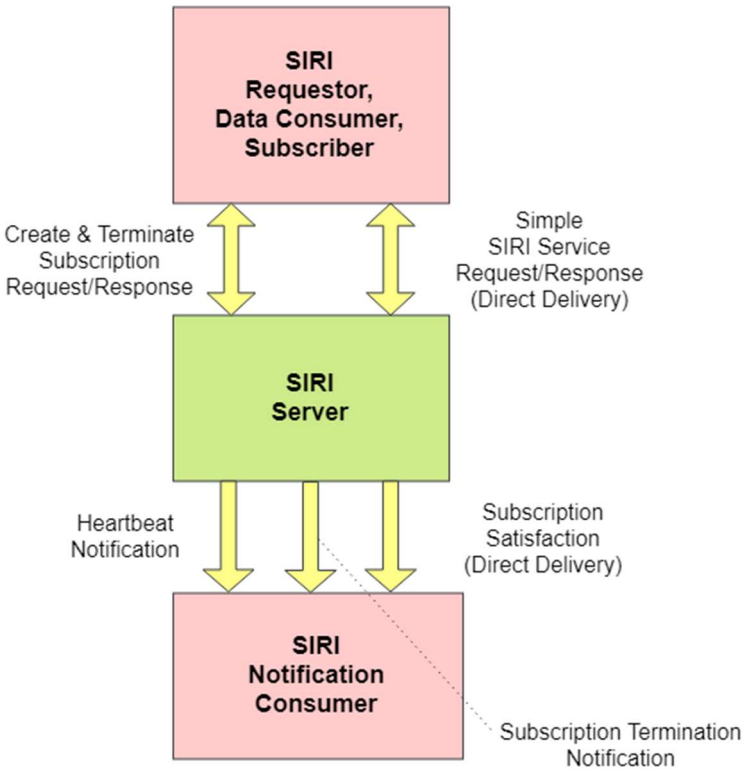


Ilustración 54: Esquema de la arquitectura del cliente de SIRI implementado

4.3.4. *Requestor, Data Consumer* y *Subscriber*

Los *endpoints* lógicos de SIRI ***Requestor***, ***Data Consumer*** y ***Subscriber*** han sido agrupados bajo una misma **aplicación de consola**, **Siri_Client**, lo que implica que desde esta parte del cliente:

- Se solicitan peticiones de *Request/Response* (*Requestor*).
- Se reciben las respuestas a dichas peticiones (*Data Consumer*).
- Se solicitan y reciben las respuestas a peticiones de creación y anulación de subscripciones (*Subscriber*).

Debido a que es en Siri_Client donde se reciben las respuestas a las peticiones simples, es esta parte la que incluye Googletest para las pruebas.

La clase en torno a la cual se han construido los tests es la llamada `BaseSIRIClient`:

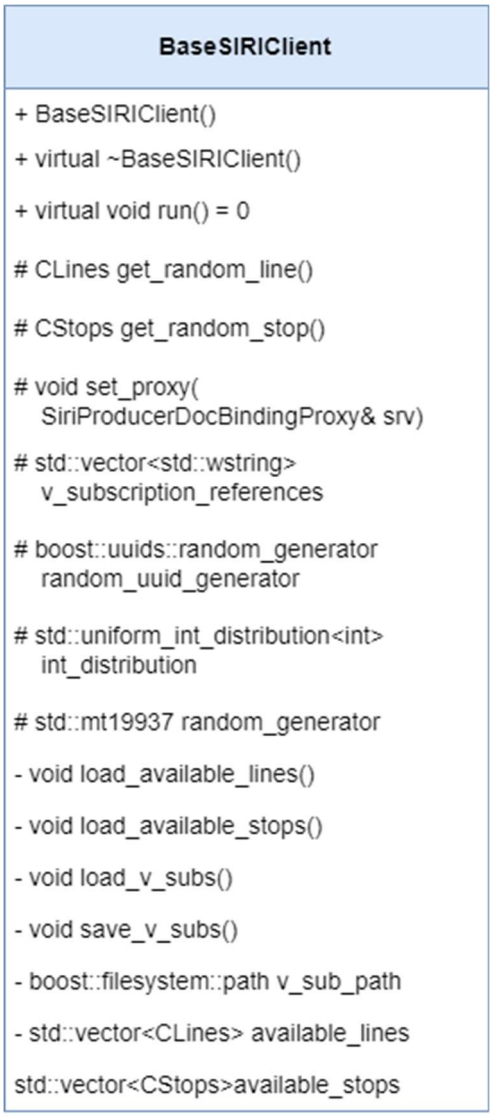


Ilustración 55: Diagrama UML de la clase BaseSIRIClient

De `BaseSIRIClient` heredan todas las clases cuyos métodos son objeto de testeo, proveyendo a estas de:

- La función `void set_proxy(SiriProducerDocBindingProxy& srv)`, que permite activar un proxy. Este proxy facilita la captura directa del XML que devuelve el servidor, lo que se puede lograr mediante el uso de aplicaciones como Fiddler 4.
  En la implementación real, la ip y puerto de este proxy se leían de un archivo de configuración (lo que evita tener que recompilar el cliente para cambiar dicha información); pero como para hacerlo se utilizaba *software* interno de GMV, se ha añadido dentro el propio código fuente del cliente dicha configuración *'hardcodeada'*.
- Los generadores aleatorios: `std::mt19937` y `boost::uuids::random_generator` y una distribución entera uniforme.
- Acceso a `std::vector<std::wstring> v_subscription_references`, vector en el que se almacenan las referencias de las subscripciones activas en cada momento, de manera que esta información sobreviva a los tests exitosos de `CreateSubscription`, y puedan ser ejecutados de manera completa los tests de `DeleteSubscription` que eliminan correctamente una subscripción. En nuestro caso, y por simplicidad, esta información se almacena en un archivo plano de texto y se carga en memoria antes de la ejecución de los tests.
- Acceso de lectura a líneas y paradas aleatorias de entre las existentes en el servidor mediante `const CLines get_random_line()` y `const CStops get_random_stop()`. De nuevo antes de la ejecución de cada test, tienen lugar sendas peticiones de tipo *Lines Delivery* y *StopPoints Delivery* al servidor. Esto permite usar posteriormente esos datos para asegurarnos de probar el servidor de la manera adecuada y siguiendo los flujos deseados en cada caso (por ejemplo, filtros por líneas inexistentes, filtros por una parada existente y otra inventada, etc.).

El constructor y destructor de `BaseSIRIClient` se han implementado de la siguiente manera:

```
BaseSIRIClient::BaseSIRIClient()
{
	random_generator.seed((unsigned int)
std::chrono::high_resolution_clock::now().time_since_epoch().count());
	int_distribution = std::uniform_int_distribution<int>(0, 10);

	load_available_lines();
	load_available_stops();

	load_v_subs();
}

BaseSIRIClient::~BaseSIRIClient() { save_v_subs(); }
```

Teniendo en cuenta lo anterior, la estructura general de los tests es la siguiente:

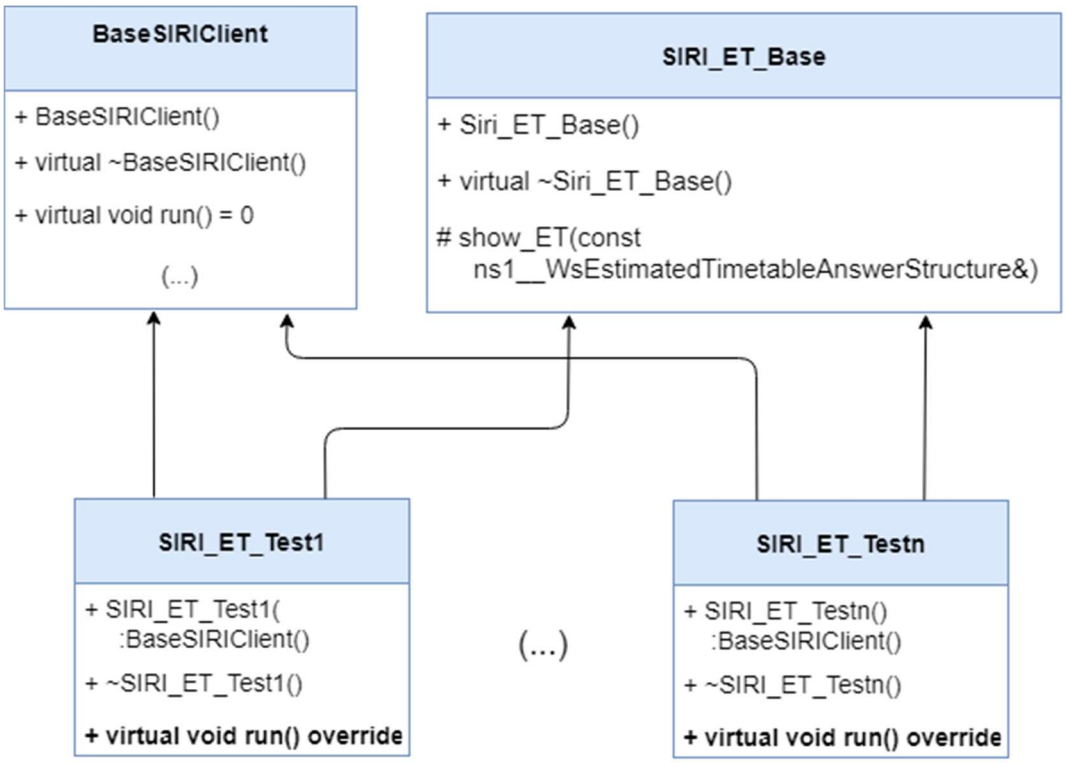


Ilustración 56: Diagrama UML de las clases de Siri_Client

Se ha encapsulado cada test a probar dentro de una clase que hereda tanto de `BaseSIRIClient` como de una clase base común a todos los tests del servicio que se está probando, `SIRI_XX_Base`.

Nótese la invocación al constructor de `BaseSIRIClient` en todos los tests.

La clase común a los tests de un mismo servicio se incluyó con el mero propósito de proporcionar un método de escritura común de los datos contenidos en la respuesta del servidor en un fichero de texto, a modo de log y para poder comprobar visualmente tras los tests la validez de los datos recibidos, haciendo hincapié en los datos deseados (ya que los datos crudos pueden verse en el XML capturado por el proxy). Esos logs conforman el anexo 3.

La estructura de clases empleada probablemente no sea la óptima, existiendo alternativas como:

- El uso de una clase estática para el almacenamiento de todos los datos de `BaseSIRIClient` (o bien que simplemente permitiera acceder a los servicios de `Delivery` cuando fuera necesario) y otra que proporcionara los métodos de escritura incluidos en `SIRI_XX_Base`.
- El aprovechamiento de los *test fixtures* de los que nos provee el *framework* de testeo, con los que se puede definir código que queremos que se ejecute antes y después de cada test y código que queremos que se ejecute antes y después del conjunto total de tests lanzados.

Veamos el funcionamiento completo de la aplicación, en la que está definido un único *test fixture*, `Siri_ClientTestFixture`.

El método principal de la aplicación Siri_Client es el siguiente:

```
int main(int argc, char* argv[])
{
	::testing::InitGoogleTest(&argc, argv);
	//testing::FLAGS_gtest_repeat = 10;
	int nRetCode = RUN_ALL_TESTS();

	//if(nRetCode)
		system("pause");

	return nRetCode;
}
```

El código sin comentar es bastante auto-explicativo, mientras que comentadas están las opciones de:

- `testing::FLAGS_gtest_repeat = n` permite la repetición iterativa de los tests.
- `if(nRetCode) system("pause")` permite evitar el cierre de la consola de comandos desde la que se ejecutan los tests en caso de error en alguno de los mismos.

Una vez que se invoca `RUN_ALL_TESTS()`, el *framework* se encarga de encontrar todos los tests definidos en el proyecto y ejecutarlos.

Repartidos por los distintos archivos, dichos tests se encuentra definidos de manera similar a estos de *Facility Monitoring*:

```
TEST_F(Siri_ClientTestFixture, FM_All_Lines)
	{ Siri_FM_All_Lines FM; FM.run(); }

TEST_F(Siri_ClientTestFixture, FM_All_Stops)
	{Siri_FM_All_Stops FM; FM.run(); }

TEST_F(Siri_ClientTestFixture, FM_Error_Unauth)
	{ Siri_FM_Error_Unauth FM; FM.run(); }

TEST_F(Siri_ClientTestFixture, FM_Error_LineRef_StopPointRef_Missing)
	{ Siri_FM_Error_LineRef_StopPointRef_Missing FM; FM.run(); }

TEST_F(Siri_ClientTestFixture, FM_Wrong_ALL_LINES_CODE)
	{ Siri_FM_Wrong_ALL_LINES_CODE FM; FM.run(); }

TEST_F(Siri_ClientTestFixture, FM_Wrong_ALL_STOPS_CODE)
	{ Siri_FM_Wrong_ALL_STOPS_CODE FM; FM.run();}
```

Se utiliza el *macro* `TEST_F` para señalar los tests, eligiendo a continuación un *fixture* y el nombre deseado para la prueba.

A continuación, se coloca el código del test, que en nuestro caso se reduce a la instanciación de cada clase y a la invocación de su método `run()`.

El código está en el anexo 2, pero veamos en detalle uno de los ejemplos para facilitar la comprensión del resto:

```
void Siri_PT_Main::run()
{
  SiriProducerDocBindingProxy ws;

  ns1__ProductionTimetableRequestStructure request;
  ns1__ProductionTimetableAnswerStructure response;

  ns2__ProductionTimetableRequestStructure req;
  request.Request = &req;
  req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

  ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      //...
  request.ServiceRequestInfo = info_structure;

  ns2__ClosedTimestampRangeStructure* timestamp_structure = new
ns2__ClosedTimestampRangeStructure;
      //...
  req.ValidityPeriod = timestamp_structure;

  //set_proxy(ws);
  ASSERT_EQ(ws.GetProductionTimetable(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

  delete timestamp_structure;
  delete info_structure;

  ASSERT_FALSE(response.Answer->ProductionTimetableDelivery.empty());

  for (const auto& line : response.Answer->ProductionTimetableDelivery)

      { /*...*/}
  ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp));
}
```

El procedimiento general en los tests es el siguiente:

- Se instancia un objeto **`SiriProducerDocBindingProxy`** (clase autogenerada) y los objetos *request* y *response* de cada servicio (en nuestro caso, `ns1__ProductionTimetableRequestStructure` y `ns1__ProductionTimetableAnswerStructure`.
- Se rellena la petición, dividida usualmente en las dos partes descritas previamente en este documento: una estructura característica de cada servicio, `ns2__XXXXXXRequestStructure`, y una clase común a todos ellos, de tipo `ns1__WsServiceRequestInfoStructure`.
- Se invoca al método deseado de `SiriProducerDocBindingProxy`, pasándole como parámetros la dirección del servidor y las estructuras de petición y respuesta mencionadas.
- Se libera la memoria que hayamos reservado para formar la petición.
- Se usan las aserciones de las que nos provee Googletest para hacer las comprobaciones pertinentes sobre dicha respuesta.

4.3.5. *Notification Consumer*

La aplicación **SIRI_Notification_Consumer** asume el rol del *endpoint* lógico de SIRI ***Notificacion Consumer***, además de encargarse de la recepción de notificaciones asociadas a otros *endpoints* del cliente (como pudiera ser la notificación *NotifyTerminateSubscription*, en principio dirigida al *Subscriber*).

4.3.5.1. Arquitectura e implementación

SIRI_Notification_Consumer en una **aplicación de consola** que levanta un servidor y queda a la espera a la llegada de datos asociados a la satisfacción de subscripciones de servicios funcionales de SIRI.

En esta ocasión es la clase **`NotificationConsumer`** la que aglutina a su alrededor el funcionamiento de la aplicación, de la misma manera que la clase `Producer` lo hacía en el servidor. Una instancia de esta clase queda 'atacheada' al *threadpool* (conjunto de hebras) principal del servidor, en nuestro caso levantado en exclusiva para ejercer de *endpoint* de *Notificacion Consumer* y por tanto dedicado enteramente a ello.

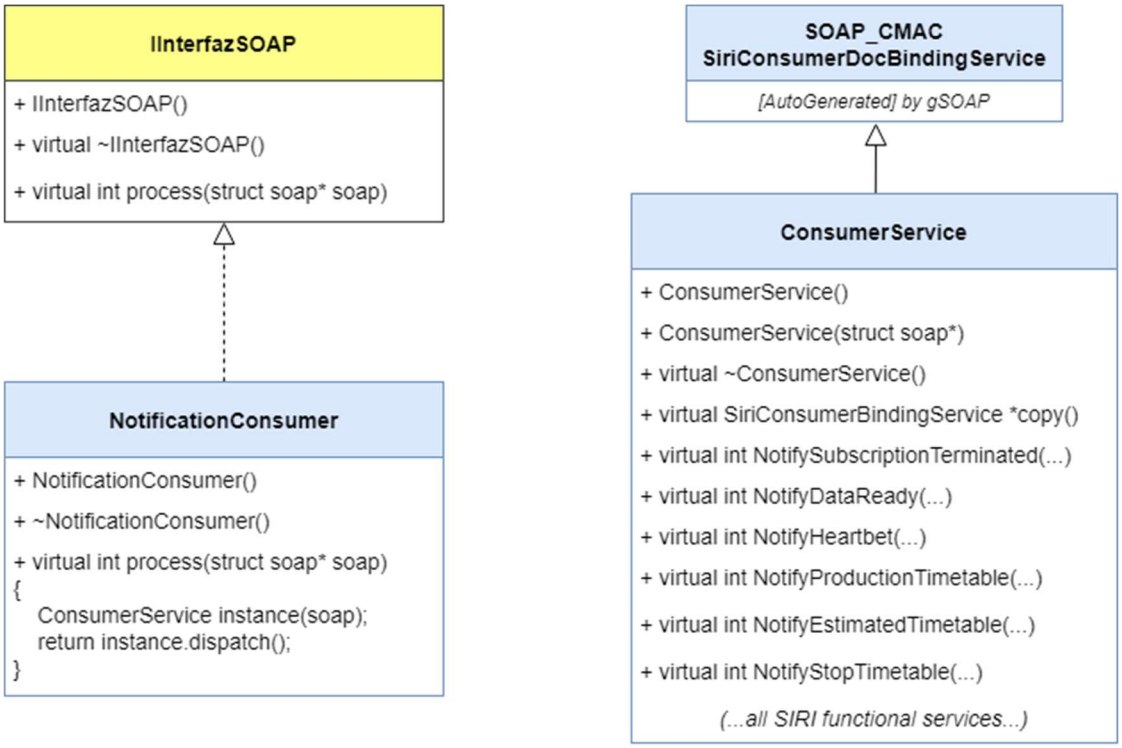


Ilustración 57: Diagramas UML de las clases NotificationConsumer y ConsumerService

El método `NotificationConsumer::process(struct soap* soap)` es invocado cada vez que se realiza una petición REST sobre el servidor en el que está levantado este servicio. Éste crea una instancia de **`ConsumerService`** por cada petición y delega en ella la resolución de la misma, invocando al método adecuado de esta instancia en función de dicha petición.

Dado que en esta aplicación sólo nos interesa visualizar los datos recibidos, los métodos de `ConsumerService` son sencillos y siguen todos ellos un mismo patrón, que es el de invocar a funciones que loguen la información recibida (en nuestro caso, en un archivo .txt de logs). Dichas funciones, llamadas `show_xxxxxxx()`, son muy parecidas a las existentes en Siri_Client con la salvedad de que en este caso no se desea sobrescribir la información, sino añadirla a la existente.

Veamos un ejemplo de las **funciones `NotifyXXXXXXXXX`**:

```
int ConsumerService::NotifyProductionTimetable(
ns1__WsProductionTimetableNotificationStructure* ns1__NotifyProductionTimetable)
{
	assert(ns1__NotifyProductionTimetable->Notification != nullptr);

	std::unique_lock<std::mutex> lock(m_lock);
	PrintUtils::show_productiontimetable(*ns1__NotifyProductionTimetable);
	std::cout << boost::posix_time::second_clock::universal_time() << " - PT
update received" << std::endl;

	return SOAP_OK;
}
```

Nótese el uso de `std::unique_lock<std::mutex> lock(m_lock)` para proteger la escritura, tanto en los archivos como en la consola de ejecución de esta aplicación, contra la concurrencia de peticiones.

Los **métodos `show_xxxxxx()`** con los que escribimos los logs también son sencillos. La única complejidad podría residir en su configuración inicial, que se resolvió de la siguiente manera:

```
#ifdef UNICODE
    std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
    std::locale loc(std::locale::classic());
#endif

    _tstring path = _T("c:/temp/siri/sub_productiontimetable.txt");
    const boost::filesystem::path p = path;
    _ofstream ofs(p.string(), std::ios::app);
    ofs.imbue(loc);


    ofs << “Now this is being appended to a .txt file”
```

Tras este fragmento de código situado al comienzo de cada método, es posible navegar por la estructura de SIRI recibida e ir imprimiendo los datos que nos interesan en cada caso usando `ofs <<`.

4.3.5.2. Implementación del servidor subyacente

En la descripción de la implementación del servidor de SIRI no fue posible explicar en detalle el nivel de abstracción inmediatamente superior el `Producer`. En esta ocasión, debido a la simplicidad y generalidad del servidor que se encuentra detrás de SIRI_Notification_Consumer, sí tenemos la oportunidad de ver detalladamente **cómo se implementa un servidor sobre el cual usar las clases autogeneradas por gSOAP**.

Para ello se usan los archivos `stdsoap2.h` y `stdsoap2.cpp` que gSOAP nos proporciona.

Este es el diagrama de la clase `CServerImp`, que levanta el mencionado servidor y se encarga de procesar la información que hasta él llega a través del *socket* de SOAP.

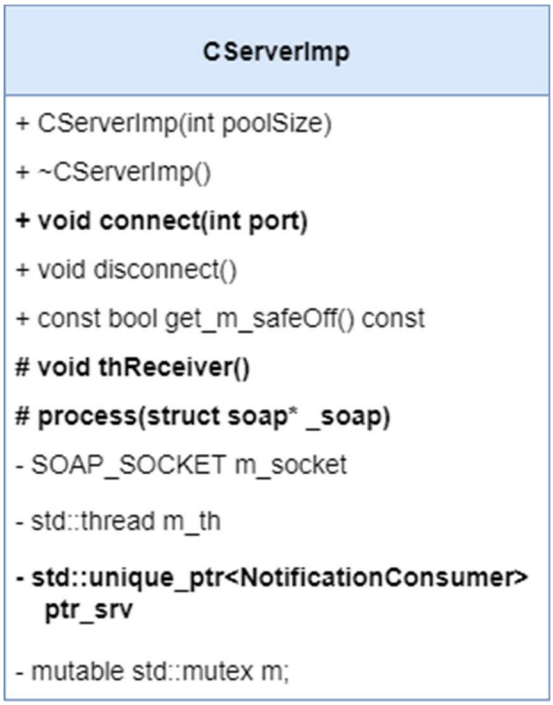


Ilustración 58: Estructura de la clase CServerImp

[El código que procede ha sido simplificado para facilitar su comprensión en este documento, siendo ligeramente diferente al incluido en el anexo correspondiente]

En su constructor se almacena un puntero a la única instancia de nuestro `NotificationConsumer` en la variable miembro `ptr_srv`, usando un `std::unique_ptr`.

```
CServerImp::CServerImp(int poolSize)
	: m_poolsize(poolSize), m_port(8080), m_safeOff(false)
{
	ptr_srv.reset(new NotificationConsumer);
}
```

Cuando se invoca el método `connect(int port)`, se toma un hilo del pool de hebras disponible y se 'atachea' al método `thReceiver`.

```
void CServerImp::connect(int port)
{
	m_bFinalizarTh = false;
	m_port = port;
	m_th = std::thread((std::bind(&CServerImp::thReceiver, this)));
}
```

En `thReceiver()` se abre el *socket*, asociado a un objeto tipo `soap`.

```
void CServerImp::thReceiver()
{
	boost::threadpool::fifo_pool tp(m_poolsize);
	struct soap *_soap = soap_new();
	m_socket = soap_bind(_soap, NULL, m_port, 100);
	//...
```

Una vez abierto, y mientras no se lance el destructor de `CServerImp`,

en un bucle se van aceptando las peticiones entrantes, asignándolas un hilo y encargando al método `CServerImp::process(…)` que las procese.

```
SOAP_SOCKET sock = soap_accept(_soap);
struct soap *_soapCopy = soap_copy(_soap);

if(soap_valid_socket(sock) && _soapCopy != nullptr)
{
	typedef std::function<void()> void_function_t;
	void_function_t f = std::bind(&CServerImp::process, this, _soapCopy);
	tp.schedule(f);
	}
}
```

`CServerImp::process(struct soap* _soap)` realiza alguna otra comprobación, invoca al método `NotificationConsumer::process()` de la instancia de esta clase que está almacenada como variable privada en `CServerImp` y libera la memoria reservada por la estructura de datos.

```
void CServerImp::process(struct soap* _soap)
{
	soap_set_namespaces(_soap, srv_namespaces); // .nsmap file

	if (soap_begin_serve(_soap) == SOAP_OK)
	{
		ptr_srv->process(_soap);
	}

	soap_destroy(_soap);
	soap_end(_soap);
	soap_free(_soap);
}
```

Ese método `process()` es el descrito al comienzo del apartado, análogo al `Producer::process()` a partir del cual se ha explicado el servidor.

4.3.6.Captura de datos públicos con el cliente implementado

En el siguiente apartado se hablará de la validación de las aplicaciones implementadas, especialmente del servidor, pero una buena manera de demostrar el correcto funcionamiento del cliente es usarlo para obtener información de un servidor ajeno.

Existen varios *endpoints* públicos de SIRI a los que es posible solicitar información, siempre y cuando se respeten ciertas limitaciones respecto a la frecuencia y volumen de las peticiones. Una pequeña parte de esos *endpoints* son totalmente públicos, mientras que para acceder a la mayoría de ellos se requiere solicitar a la empresa o autoridad competente acceso en calidad de desarrollador.

Veamos algunos ejemplos de capturas de información real, incluyendo la petición realizada y la respuesta obtenida, Ésta última está truncada en algunos casos, pudiéndose ver la original en el anexo 4.

4.3.6.1. Tampere (Finlandia)

ITS Factory se define como un entorno de innovación, desarrollo y experimentación en el que tanto compañías como desarrolladores pueden desarrollar y probar soluciones de s*mart traffic* [22].

La ciudad de Tampere (Finlandia), lugar en el que se originó esta iniciativa, colabora con esta agrupación de empresas y entidades gubernamentales haciendo públicas algunas de sus APIs, a través de las cuales es posible acceder en tiempo real a datos de transporte público de la ciudad.

Entre estos recursos se encuentran APIs públicas para acceder a los servicios funcionales de *Vehicle Monitoring* y *General Message* [23], implementados de acuerdo con la versión v1.3 de SIRI.

Petición de *General Message* (GM):

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
  xmlns="http://www.siri.org.uk/siri"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3"
xsi:schemaLocation="http://www.kizoom.com/standards/siri/schema/1.3/siri.xsd">
  <ServiceRequest>
    <RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
    <GeneralMessageRequest version="1.3">
      <RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
    </GeneralMessageRequest>
  </ServiceRequest>
</Siri>
```

Respuesta de *General Message* (GM), truncada:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Siri
  xmlns="http://www.siri.org.uk/siri"
  xmlns:ns2="http://www.ifopt.org.uk/acsb"
  xmlns:ns3="http://www.ifopt.org.uk/ifopt"
  xmlns:ns4="http://datex2.eu/schema/1_0/1_0" version="1.3">
  <ServiceDelivery>
    <ResponseTimestamp>2018-06-03T19:00:40.598+03:00</ResponseTimestamp>
    <ProducerRef>IJ2010</ProducerRef>
    <Status>true</Status>
    <MoreData>false</MoreData>
    <GeneralMessageDelivery version="1.3">
      <ResponseTimestamp>2018-06-03T19:00:40.598+03:00</ResponseTimestamp>
      <Status>true</Status>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2010-01-01T00:00:00+02:00</RecordedAtTime>
        <InfoMessageIdentifier>1014</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>errors</InfoChannelRef>
        <ValidUntilTime>2099-01-01T00:00:00+02:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Linja 3R ajaa Possilankadulta suoraan Teivaalankadulle 4.6.
alkaen. Reitiltä jää pois Ryydynkatu ja osa Teivaalantiestä. -- Bus 3R runs from
Possilankatu straight to Teivaalankatu, starting 4 June. Ryydynkatu and part of
Teivaalantie will be cut off from the route. -- nysse.fi
        </Content>
      </GeneralMessage>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2010-01-01T00:00:00+02:00</RecordedAtTime>
        <InfoMessageIdentifier>1008</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>warnings</InfoChannelRef>
        <ValidUntilTime>2099-01-01T00:00:00+02:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Reitti kulkee Kissanmaalla Karhukadun kautta 4.6.alkaen.
Pysäkit Kissanmaankatu 7/8 ja Hippoksenkatu siirretään Karhukadulle. -- The
route runs via Karhukatu at Kissanmaa, from 4 June. Stops at Kissanmaankatu 7/8
and Hippoksenkatu will be moved to Karhukatu. -- nysse.fi
        </Content>
      </GeneralMessage>
    </GeneralMessageDelivery>
  </ServiceDelivery>
</Siri>
```

Petición de *Vehicle Monitoring* (VM):

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns6:GetVehicleMonitoring
      xmlns:ns2="http://www.siri.org.uk/siri"
      xmlns:ns3="http://www.ifopt.org.uk/acsb"
      xmlns:ns4="http://www.ifopt.org.uk/ifopt"
      xmlns:ns5="http://datex2.eu/schema/2_0RC1/2_0"
      xmlns:ns6="http://wsdl.siri.org.uk">
      <ServiceRequestInfo>
        <ns2:RequestTimestamp>2017-05-
25T09:29:47.529+02:00</ns2:RequestTimestamp>
        <ns2:MessageIdentifier>VehicleMonitoring:Test:0</ns2:MessageIdentifier>
      </ServiceRequestInfo>
      <Request version="2.0">
        <ns2:RequestTimestamp>2017-05-
25T09:29:47.522+02:00</ns2:RequestTimestamp>
        <ns2:LineRef>1</ns2:LineRef>
      </Request>
      <RequestExtension/>
    </ns6:GetVehicleMonitoring>
  </S:Body>
</S:Envelope>
```

Respuesta de *Vehicle Monitoring* (VM), truncada:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Siri version="1.3"
  xmlns="http://www.siri.org.uk/siri">
  <ServiceDelivery>
    <ResponseTimestamp>2018-06-03T18:55:32.049+03:00</ResponseTimestamp>
    <ProducerRef>IJ2010</ProducerRef>
    <Status>true</Status>
    <MoreData>false</MoreData>
    <VehicleMonitoringDelivery version="1.3">
      <ResponseTimestamp>2018-06-03T18:55:32.049+03:00</ResponseTimestamp>
      <Status>true</Status>
      <VehicleActivity>
        <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
        <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
```

```
        <MonitoredVehicleJourney>
          <LineRef>1C</LineRef>
          <DirectionRef>1</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2018-06-03</DataFrameRef>
            <DatedVehicleJourneyRef>1800</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <OperatorRef>TKL</OperatorRef>
          <OriginName xml:lang="fi">Vatiala</OriginName>
          <DestinationName xml:lang="fi">Vaitti</DestinationName>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>23.6440378</Longitude>
            <Latitude>61.4657443</Latitude>
          </VehicleLocation>
          <Bearing>258.0</Bearing>
          <Delay>P0Y0M0DT0H3M47.000S</Delay>
          <VehicleRef>TKL_46</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
      <VehicleActivity>
        <RecordedAtTime>2018-06-03T18:55:32.024+03:00</RecordedAtTime>
        <ValidUntilTime>2018-06-03T18:56:02.024+03:00</ValidUntilTime>
        <MonitoredVehicleJourney>
          <LineRef>1B</LineRef>
          <DirectionRef>1</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2018-06-03</DataFrameRef>
            <DatedVehicleJourneyRef>1840</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <OperatorRef>paunu</OperatorRef>
          <OriginName xml:lang="fi">Vatiala</OriginName>
          <DestinationName xml:lang="fi">Teollisuustie</DestinationName>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>23.8767302</Longitude>
            <Latitude>61.4998658</Latitude>
          </VehicleLocation>
          <Bearing>0.0</Bearing>
          <Delay>P0Y0M0DT0H0M35.000S</Delay>
          <VehicleRef>paunu_157</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
    </VehicleMonitoringDelivery>
  </ServiceDelivery>
</Siri>
```

4.3.6.2. Utah (EE. UU.)

La Autoridad de Transportes de Utah también pone a disposición de los desarrolladores APIs públicas de los servicios funcionales *Stop Monitoring* y *Vehicle Monitoring* de SIRI, una vez más implementados según la versión 1.3 del estándar.

Como clientes, tenemos que adaptarnos y hacer la petición usando el método de transporte que acepta el *endpoint*, que en este caso no es través de una petición HTTP POST (ni con envoltorio SOAP ni sin él), tal y como estábamos acostumbrados.

Este *endpoint* acepta el método de transporte **SIRI Lite**, en el que **la *request* se realiza mediante una petición HTTP GET parametrizada** [24].

Esto quiere decir que podemos prescindir de nuestro cliente para obtener esta información.

Petición de *Stop Monitoring* filtrada por identificador de la parada:

```
[HTTP GET]
http://api.rideuta.com/SIRI/SIRI.svc/StopMonitor?stopid=133054
&minutesout=30
&onwardcalls=true
&filterroute=
&usertoken=ABCDEFGHIJ
```

Respuesta de *Stop Monitoring*:

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T10:03:50.0961263-06:00</ResponseTimestamp>
  <StopMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T10:03:50.0961263-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T10:04:00.0961263-06:00</ValidUntil>
    <MonitoredStopVisit>
      <RecordedAtTime>2018-06-03T10:03:50.0961263-06:00</RecordedAtTime>
    </MonitoredStopVisit>
    <Extensions>
      <StopName>3500 S @ 6450 W</StopName>
      <StopLongitude>-112.045255000</StopLongitude>
      <StopLatitude>40.696599000</StopLatitude>
      <StopLocation>Utah</StopLocation>
    </Extensions>
  </StopMonitoringDelivery>
</Siri>
```

Petición de *Vehicle Monitoring*, filtrada por identificador de la ruta:

```
[HTTP GET]
http://api.rideuta.com/SIRI/SIRI.svc/VehicleMonitor/ByRoute?route=2
&onwardcalls=true
&usertoken=ABCDEFGHIJ
```

Respuesta de Vehicle Monitoring:

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T11:55:13.8241503-06:00</ResponseTimestamp>
  <VehicleMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T11:55:13.8241503-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T11:55:23.8241503-06:00</ValidUntil>
    <VehicleActivity>
      <RecordedAtTime>2018-06-03T11:55:13.8241503-06:00</RecordedAtTime>
    </VehicleActivity>
  </VehicleMonitoringDelivery>
</Siri>
```

Petición de *Vehicle Monitoring*, filtrada por identificador del vehículo:

```
[HTTP GET]
http://api.rideuta.com/SIRI/SIRI.svc/VehicleMonitor/ByVehicle?vehicle=10054
&onwardcalls=true
&usertoken=ABCDEFGHIJ
```

Respuesta de Vehicle Monitoring:

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T12:05:33.8024053-06:00</ResponseTimestamp>
  <VehicleMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T12:05:33.8024053-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T12:05:43.8024053-06:00</ValidUntil>
    <VehicleActivity>
      <RecordedAtTime>2018-06-03T12:05:33.8024053-06:00</RecordedAtTime>
    </VehicleActivity>
  </VehicleMonitoringDelivery>
</Siri>
```

4.3.6.3. Nueva York (Estados Unidos)

La compañía MTA Bus Time es uno de los operadores de transporte de la ciudad de Nueva York (Estados Unidos), y también ofrece una API pública para desarrolladores [25] a través de la cual se puede acceder a varios servicios funcionales de SIRI.

MTA también nos ofrece *endpoints* que usan el método de transporte **SIRI Lite**, y además **nos permite elegir entre el JSON y XML como formatos de la respuesta**. Se adjunta la versión en formato JSON, por mostrar ejemplos de ambos formatos.

Petición de *Stop Monitoring*:

```
[HTTP GET]
http://bustime.mta.info/api/siri/stop-monitoring.json?key=xxxx-xxxx-xxxx-xxxx
&OperatorRef=MTA
&MonitoringRef=308209
&LineRef=MTA%20NYCT_B63
```

Respuesta de *Stop Monitoring* en formato JSON

```
{
  "Siri":{
    "ServiceDelivery":{
      "ResponseTimestamp":"2018-06-03T14:19:46.392-04:00",
      "StopMonitoringDelivery":[
        {
          "MonitoredStopVisit":[
            {
              "MonitoredVehicleJourney":{
                "LineRef":"MTA NYCT_B63",
                "DirectionRef":"0",
                "FramedVehicleJourneyRef":{
                  "DataFrameRef":"2018-06-03",
                  "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
083000_B63_112"
                },
                "JourneyPatternRef":"MTA_B630144",
                "PublishedLineName":"B63",
                "OperatorRef":"MTA NYCT",
                "OriginRef":"MTA_306619",
                "DestinationRef":"MTA_801007",
                "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
```

```
            "SituationRef":[
              {
                "SituationSimpleRef":"MTA NYCT_191142"
              }
            ],
            "Monitored":true,
              "VehicleLocation":{
                "Longitude":-74.011831,
                "Latitude":40.643652
              },
              "Bearing":43.79817,
              "ProgressRate":"normalProgress",
              "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_21300_B63-104",
              "VehicleRef":"MTA NYCT_383",
              "MonitoredCall":{
                "ExpectedArrivalTime":"2018-06-03T14:48:50.568-04:00",
                "ExpectedDepartureTime":"2018-06-03T14:48:50.568-04:00",
                "Extensions":{
                  "Distances":{
                    "PresentableDistance":"2.8 miles away",
                    "DistanceFromCall":4546.73,
                    "StopsFromCall":21,
                    "CallDistanceAlongRoute":8811.23
                  }
                },
                "StopPointRef":"MTA_308209",
                "VisitNumber":1,
                "StopPointName":"5 AV/UNION ST"
              },
              "OnwardCalls":{

              }
            },
            "RecordedAtTime":"2018-06-03T14:19:21.000-04:00"
          },
        ],
        "ResponseTimestamp":"2018-06-03T14:19:46.392-04:00",
        "ValidUntil":"2018-06-03T14:20:46.392-04:00"
      }
    ],
  }
}
```

4.4. Validación de las implementaciones anteriores

En un estándar como SIRI, si seguimos el paradigma cliente-servidor tal y como lo hemos interpretado a lo largo de este documento, **la parte más importante de la validación** recae sobre quien envía el grueso de los datos, el **lado servidor**.

En el contexto de un intercambio de información, es el servidor quien ostenta la mayor parte de los datos y el cliente quien desea acceder a ellos. Por tanto, es el servidor el que generalmente tiene que decidir **qué información incluir y cómo estructurarla**; limitándose el cliente a tener que interpretarla a partir de un XML, tarea a priori menos crítica.

Lo anterior es especialmente cierto en el **marco profesional concreto** en el que se desarrollaron las implementaciones descritas, donde el principal objetivo de la adopción del estándar SIRI era la exportación de datos hacia servidores externos, frente a la importación de los mismos.

Esto es debido a que el servidor al que da apoyo la capa de SIRI implementada obtiene y procesa él mismo la información de transporte público, por lo que en la amplia mayoría de los intercambios de comunicación de alto nivel en los que interviene ejerce de lado servidor.

Si por el contrario la implementación hubiera sido realizada para un servidor de comunicaciones, sería la validación del cliente a la que se le habría otorgado una mayor importancia.

4.4.1. Estructura y formato de los XML

Los archivos XSD y WSDL que acompañan al estándar SIRI tienen la función de posibilitar la validación de los ficheros XML generados por las implementaciones que se hagan del estándar.

El uso de **gSOAP** para la generación de esos archivos XML de manera automática a partir de la información rellenada en las estructuras de las que nos provee garantiza que **los XML generados cumplirán estructuralmente con el estándar**, lo que se traduce en que **la información que se intercambie estará ordenada de acuerdo con el estándar**.

Huelga decir que la **autogeneración** no es la única manera de lograr XML válidos estructuralmente, pero en el caso particular de SIRI se trata de un comportamiento altamente recomendado, al tratarse de estructuras de datos tan complejas y ser el propio estándar el que te facilita las 'reglas' de validación. Es asequible basarse en XML existentes (por ejemplo, los que ofrece el propio SIRI) y realizar sobre ellos las modificaciones pertinentes para adaptarlos a tus necesidades, pero es un método que en la práctica se desaconseja por carecer de fiabilidad.

4.4.2. Contenido de los XML: alcance y forma

La autogeneración de estructuras a partir de los XSD y WSDL nos valida cómo se organiza la información que se intercambia. Sin embargo, el estándar va más allá, definiendo **qué información debe ser intercambiada y cómo debe representarse esa información**.

Respecto a la información que debe ser intercambiada, entendiendo esto en el sentido de la información que se requiere sea enviada en un determinado tipo de intercambio, es posible que la autogeneración de estructuras te obligue a nivel programático a rellenar algunos campos de las mismas; pero el estándar SIRI describe la información que se debe transmitir, no los campos que deben rellenarse. Un XML con todos los campos que representan información obligatoria rellenos con espacios no es un XML que siga el estándar SIRI. Esta problemática se tratará en el siguiente apartado.

Respecto a cómo debe representarse esa información, SIRI define el formato que deben tener los valores de ciertos campos. Ejemplos de esto son aquellos campos que llevan asociadas unidades de medida (velocidad, distancia, tiempo, coordenadas geográficas, etc.). Igualmente, un campo de texto puede estar restringido a ciertos caracteres (alfabéticos, alfanuméricos, etc.), ciertos campos no deben estar rellenos de manera simultánea, otros deben ser coherentes con unos terceros, otros no pueden ser extensibles, etc.

Para un estándar de las características del estudiado, este tipo de inexactitudes pueden acarrear serios problemas de entendimiento entre las partes que intervienen en la comunicación, por lo que se debe procurar seguir la norma de la manera más estricta posible.

Además de tener muy en cuenta la literatura y evidenciar programática y visualmente que los XML generados por tus implementaciones siguen estrictamente el estándar en forma y contenido, existan otras maneras de comprobar este tipo de reglas: las herramientas de validación externas.

En nuestro caso, y teniendo en cuenta que la implementación se ha realizado siguiendo el estándar en su versión 2.0o (versión ligeramente posterior al 2.0), una de las herramienta encontrada que mejor se adapta a nuestras necesidades es el **"*client de test SIRI-Validator*" de IRYS** [26], un proyecto *open source* de SIRI realizado y mantenido por AFIMB (*L'Agence Française pour l'Information Multimodale et la Billettique*), una agencia asociada al ministerio francés de Ecología, Desarrollo Sostenible y Energía y encargada de buena parte de los asuntos relativos al transporte público de Francia.

Este validador es, como su nombre indica, un cliente de SIRI que sigue la versión 2.0 del estándar y cuyo cometido es probar los servicios funcionales de la parte servidor de SIRI.

Su funcionamiento se basa en la **realización de peticiones de distintos servicios funcionales** a una dirección previamente configurada para el posterior **análisis del XML devuelto en cada una de ellas** [27].

La versión de este validador utilizada, la más reciente hasta el momento de redacción de estas líneas, admite los siguientes servicios:

- *Production Timetable (Request/Response* y *Publish/Subscribe*).
- *Estimated Timetable (Request/Response* y *Publish/Subscribe*).
- *Stop Timetable (Request/Response* y *Publish/Subscribe*).
- *Stop Monitoring (Request/Response* y *Publish/Subscribe*).
- *Vehicle Monitoring (Request/Response* y *Publish/Subscribe*).
- *General Message (Request/Response* y *Publish/Subscribe*).
- *Lines Discovery (Request/Response).*
- *StopPoints Discovery (Request/Response).*
- *CheckStatus (Request/Response).*

La herramienta incorpora un cliente de consola desde el cual realizar las peticiones deseadas, admitiendo el parseado de esas peticiones desde un archivo para mayor comodidad del usuario.

Un ejemplo de cómo definir varias de esas peticiones en un solo archivo es el siguiente:

```
PTClient -LineId 1   -in trash/PT_Req -out trash/PT_Res     -d ./trash
SMClient -stopId 206 -in trash/SM_Req -out trash/SM_Res     -d ./trash
VMClient -LineId 1   -in trash/VM_Req -out trash/VM_Res     -d ./trash
DSClient -Line   -in trash/Lines_Req  -out trash/Lines_Res  -d ./trash
DSClient -Stop   -in trash/Stops_Req  -out trash/Stops_Res  -d ./trash
ETClient -Line   -in trash/ET_Req     -out trash/ET_Res     -d ./trash

Subscribe -Notify /localhost:8081/ -ValidUntil 2017/07/28-13:00 -Service
ETClient -Line
Subscribe -Notify /localhost:8081/ -ValidUntil 2017/07/28-13:00 -Service
PTClient -LineId 1
Subscribe -Notify /localhost:8081/ -ValidUntil 2017/07/28-13:00 -Service
SMClient -stopId 206
Subscribe -Notify /localhost:8081/ -ValidUntil 2017/07/28-13:00 -Service
VMClient -LineId 1
Unsubscribe -SubscriptionId all
```

Basta con indicar el nombre del servicio, los filtros deseados y las carpetas de salida de petición (*request*) y respuesta (*response*) para crear y lanzar una petición de un servicio funcional de SIRI.

Además del archivo de configuración de las peticiones, el validador tiene un archivo de configuración general (`irys-client-properties`) en el que fue necesario ajustar los siguientes parámetros:

```
# Proxy parameters if needed: comment name if useless
proxy.name=
proxy.port=8888

# General parameters
siri.version=2.0:FR-IDF-2.4
siri.server=http://localhost:8080/siri
siri.requestorRef=siriClientValidator
siri.authUser=user
siri.authPass=password

# SOAP Timeout (default = 90000)
siri.timeOut=90000

# Notify default address (Notification Consumer address)
siri.notifyAddress=http://localhost:8081siri.notifyLog=true
```

Tras obtener la respuesta a la petición elegida (o al grupo de peticiones, en el caso de haberlas agrupado en un archivo y haber seleccionado la ejecución del mismo), se genera un archivo de logs con las incongruencias detectadas con respecto al estándar. Algunos ejemplos de ellas son:

- `Sax error cvc-complex-type.2.4.a: Invalid content was found starting with element 'LinesDiscoveryAnswerInfo'. One of '{Answer}' is expected.`
- `Sax error cvc-complex-type.2.2: Element 'ns2:OriginName' must have no element [children], and the value must be valid.`
- `Sax error cvc-pattern-valid: Value 'CIRCULAR QUAY [2]' is not facet-valid with respect to pattern '[^,\[\]\{\}\?$%\^=@#;:]+' for type 'PopulatedPlaceNameType'.`
- `Sax error cvc-complex-type.2.4.b: The content of element 'ns2:EstimatedTimetableDelivery' is not complete. One of '{"http://www.siri.org.uk/siri":DefaultLanguage, "http://www.siri.org.uk/siri":EstimatedJourneyVersionFrame}' is expected.`

4.4.3. Contenido de los XML: significado

Las validaciones de la estructura, formato, alcance y forma de los XML ayudan al desarrollador en el trabajo de validación de sus implementaciones, siendo **condiciones necesarias** para el cumplimiento del estándar SIRI, **pero no condiciones suficientes** para ello.

Supongamos la siguiente situación:
En una respuesta de *Vehicle Monitoring* se mandan datos sobre un conjunto de vehículos. Cada vehículo que se envíe debe llevar asociado un identificador único (`VehicleRef`), que supongamos debe ser una cadena de caracteres que cumpla una expresión regular (*regex*) determinada.
Si al implementar el procesado de una petición de *Vehicle Monitoring* el desarrollador se equivoca y escribe en el campo `VehicleRef` de cada uno de los vehículos en cuestión el nombre de la empresa a la que pertenece el vehículo (nombre que también cumple la expresión regular que sigue `VehicleRef`), ninguna de las validaciones anteriores nos avisará; ya que la estructura y formato de los XML y el alcance y forma de su contenido son los correctos. Sin embargo, el significado del contenido no lo es, en tanto **no representa la información que debería**, según el estándar.
Realizar comprobaciones automáticas sobre el contenido de los XML supone todo un reto para el implementador. En nuestro caso, las medidas tomadas para intentar evitar este problema han sido:

- El **desarrollo, en paralelo con el servidor, del cliente de SIRI.**
  El momento en el que se puede controlar más minuciosamente el contenido de los XML es el momento del desarrollo, cuando el implementador ha leído el estándar, comprendido y definido la información que deberá transmitirse en cada campo de ese servicio funcional de SIRI e implementado el código que lo lleva a cabo.
- Unido a lo anterior**, la escritura por parte del cliente de la información relevante que ha obtenido del servidor de SIRI en un documento plano de texto**, posteriormente revisado por el propio desarrollador y por una persona externa a la implementación, pero conocedora del sistema.
  Por ejemplo, en el caso de `Vehicle Monitoring`, el archivo de log era:

```
Delivery #1,  valid until: 2017-Aug-10 08:48:04
LineRef: 1, DirectionRef: 9, VehicleJourneyName: 1063
Long. tramo: 1553.61 m     % Recorrido: 3.28935 %
Service ID: 1063 1063      Delay: 00:00:24
Speed: 2 km/h - Bearing: 2.71992 degrees
5. MOORE PARK [2] - 1063 -  from: 2016-Dec-15 22:18:48 to: 2016-Dec-15 22:18:48
6. SURRY HILLS [2] - 1063 -  from: 2016-Dec-15 22:23:50 to: 2016-Dec-15 22:23:50
7. CENTRAL [2] - 1063 -  from: 2016-Dec-15 22:27:47 to: 2016-Dec-15 22:27:47
```

Es fácil que el propio desarrollador, tras codificar la representación de los datos esperados en un archivo de texto y ver esos archivos rellenos con la información del servidor, se dé cuenta de inmediato de errores de significado del contenido, como un porcentaje no numérico, una velocidad excesiva, una fecha de partida de una parada mayor a la de llegada, etc.
Ejemplos con logs de cada uno de los servicios de SIRI implementados se adjuntan en el anexo 3 de este documento.

5. Estudio sobre el uso de SIRI en la actualidad

El estándar SIRI **ha crecido enormemente en popularidad** en los últimos años.

Esto se debe en gran parte al esfuerzo realizado por el **CEN/TC 278 – ITS** (Comité Técnico del Comité Europeo de Normalización relacionado con los Sistemas Inteligentes de Transporte) para mejorar **Transmodel** y adecuarlo a los casos de uso reales del sector, convirtiéndolo en el modelo conceptual de referencia en el sector ITS.

Este esquema muestra la composición actual del estándar Transmodel y dónde se sitúa SIRI como protocolo de intercambio de información:

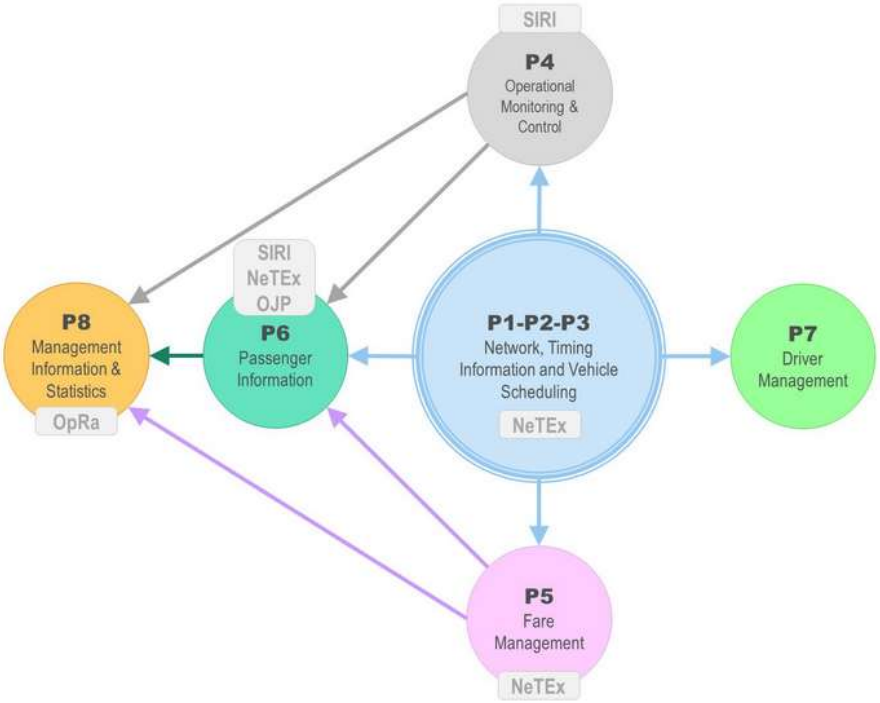


Ilustración 59: Esquema de las distintas partes de Transmodel (transmodel-cen.eu)

Otro punto clave en el crecimiento de SIRI es su propia versatilidad. El hecho de que se trate de un estándar moderno y desarrollado con la colaboración de gente estrechamente relacionada con la operativa diaria a la que aspira a dar soporte ha permitido a este protocolo posicionarse como uno de los estándares de referencia para la importación y exportación de datos.

La extensibilidad heredada de Transmodel y la propia actualización frecuente del estándar ante las necesidades reales de las empresas relacionadas con los ITS hacen de él un **protocolo suficientemente flexible como para dar cabida a los modelos de datos de miles de empresas de gestión de transporte público de todo el mundo.**

Veamos algunos ejemplos reales de implementaciones de SIRI en distintas ciudades y países:

5.1. Alemania

La VDV (Asociación de Compañías de Transporte Alemanas), histórico motor en el desarrollo de Transmodel, se ha volcado durante los últimos años en la adaptación y/o remplazo de sus propios estándares internos de ITS por estándares basados en el citado estándar europeo [28].

En el caso de las comunicaciones en tiempo real, han pasado a adoptar SIRI.

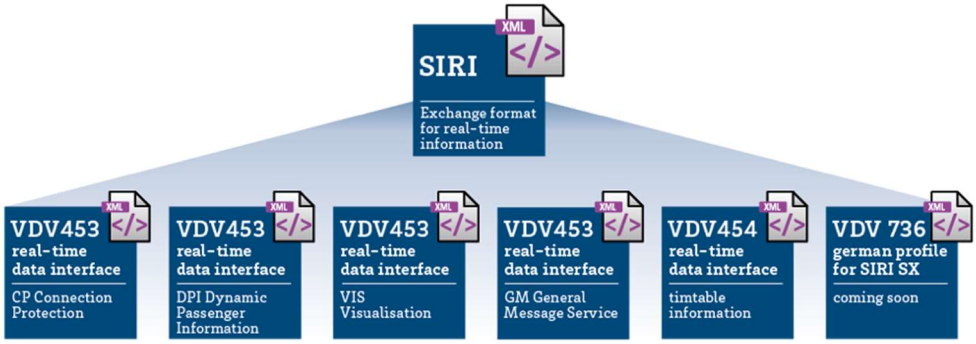


Ilustración 60: Interfaz alemana de intercambio de datos de tiempo real (transmodel-cen.eu)

5.1.1. Berlín

Los distintos operadores, a través de varios AVMS (*Automated Vehicle Management Systems*), comunican su información a un servidor central de comunicaciones usando los servicios PT, ST y SM.

Este servidor central es el encargado de hacer llegar la información al pasajero final y a otros sistemas de información externos a la ciudad.

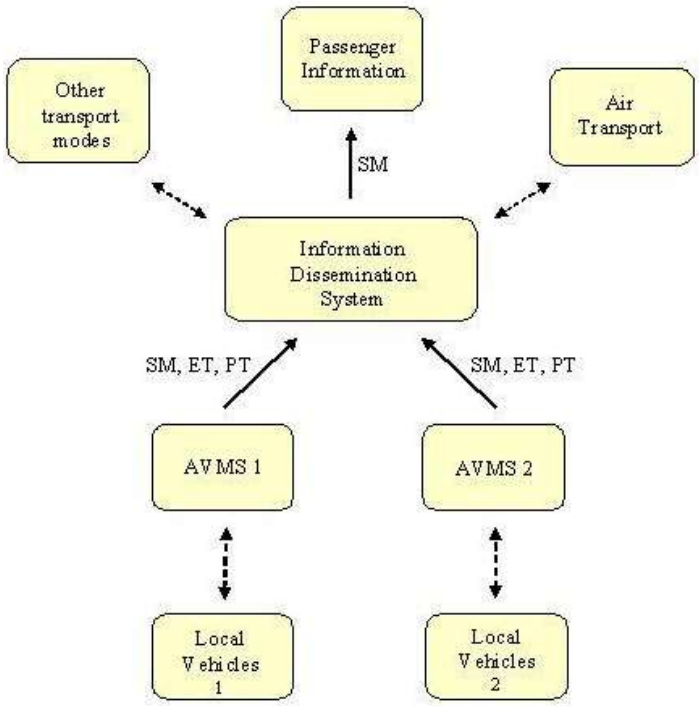


Ilustración 61: Esquema de la organización de los ITS en Berlín (EN 15531-1)

5.1.2.Hamburgo

En la ciudad de Hamburgo existen dos grandes operadores de transporte: el encargado del transporte urbano y el encargado del transporte interurbano o regional.

Los sistemas AVMS de cada uno de los operadores intercambian información directamente entre sí usando servicios funcionales de SIRI como *Stop Monitoring*, *Vehicle Monitoring*, *Connection Timetable* y *Connection Monitoring*.

Esta información es tanto planificada como de tiempo real, centrándose en la monitorización de paradas (*Stop Monitoring*) y vehículos (*Vehicle Monitoring*), así como en el intercambio de información relativa a conexiones tanto planificadas (*Connection Timetable*) como en tiempo real (*Connection Monitoring*). Esto último está pensado para facilitar los transbordos entre vehículos locales y regionales.

Adicionalmente, existe una comunicación unidireccional entre el operador de trenes y el operador encargado del transporte urbano (tranvías, autobuses) para permitir a este último tener en cuenta las conexiones que han de realizarse entre los trenes entrantes en Hamburgo y los medios de transporte locales, pudiendo así prever retrasos, cancelaciones, etc.

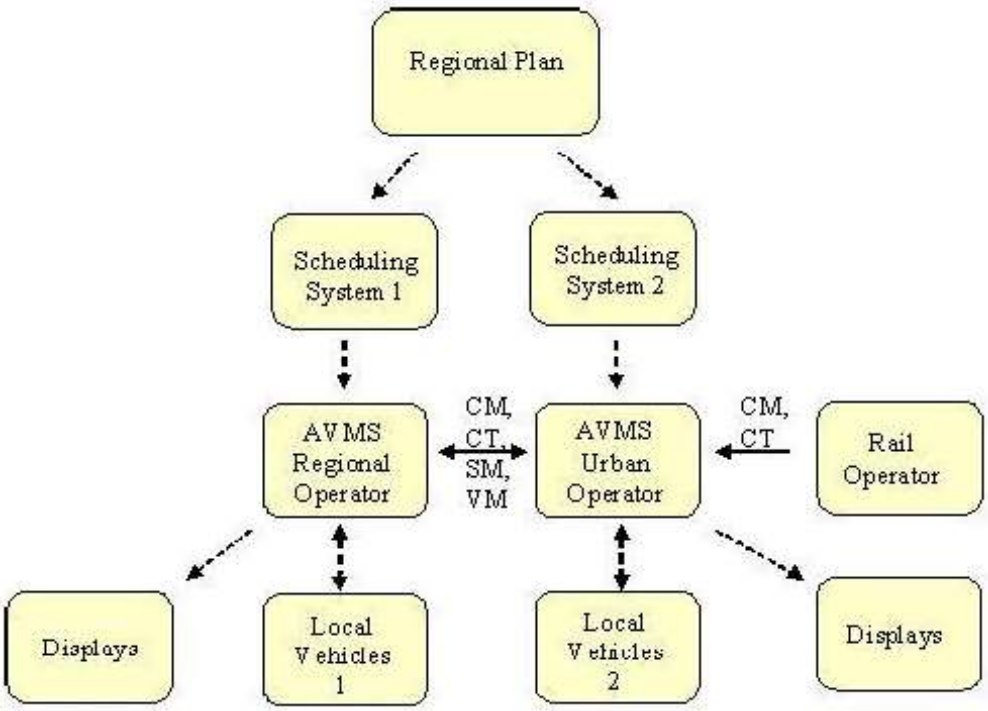


Ilustración 62: Esquema de la organización de los ITS en Hamburgo (EN 15531-1)

5.2. República Checa

El modelo checo es similar al berlinés, con la salvedad de la existencia de una mayor colaboración entre los distintos operadores de transporte, beneficiándose cada uno de ellos de la información del resto.

Existe un servidor central de comunicaciones con el que se comunican bidireccionalmente cada uno de los AVMS de los distintos operadores: mandan su información a ese servidor, pero también reciben datos del mismo. Estos intercambios de información se realizan haciendo uso de los servicios funcionales de SIRI *Stop Monitoring* y *Vehicle Monitoring*.

El servidor de comunicaciones es el encargado de la PIDS (información al usuario final), para la cual se vale una vez más de *Stop Monitoring*.

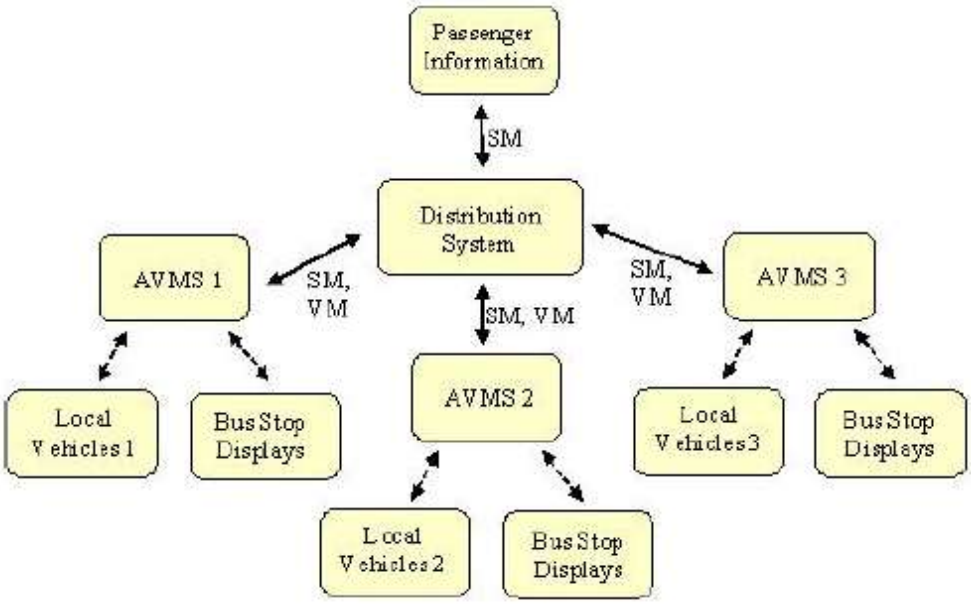


Ilustración 63: Esquema de la organización de los ITS en la República Checa (EN 15531-1)

5.3. Reino Unido

En Reino Unido los estándares predominantes son TransXChange, NPTG (*National Public Transport Gazeteer*) y NaPTAN (*National Public Transport Access Node*), todos ellos basados en Transmodel.

Juntos forman un conjunto coherente de documentos para la organización de los Sistemas Inteligentes de Transporte del país, que goza de apoyo estatal desde su creación.

La presencia de estándares como SIRI se limita por el momento al intercambio de datos con sistemas de operadores extranjeros, aunque su uso como protocolo para la comunicación con los usuarios finales del transporte público está experimentando una tendencia alcista en los últimos años.

5.4. Dinamarca: Copenhague

El modelo existente en Copenhague está basado en la centralización de toda la información en una entidad denominada '**integrador**'.

La principal diferencia entre los sistemas de distribución o sistemas centrales checo y berlinés y este integrador es que los primeros fueron diseñados como meros servicios de agrupación de información para su distribución mientras que el integrador fue diseñado para **intercomunicar todos los sistemas presentes en un entorno de ITS**.

Esto se traduce en que, si un operador nuevo entra en el sistema checo, no está obligado por modelo a comunicarse con el resto de los operadores existentes (situación similar a la que sucede en la actualidad en Berlín); mientras que el modelo danés no permitiría dicho aislamiento.

El integrador del sistema descrito agrupa los datos públicos de todas las fuentes de información existentes, permitiendo a los sistemas de gestión de vehículos (AVMS) y a los sistemas de información al pasajero (PIDS) pertenecientes al sistema la obtención flexible de los mismos.

De esta manera, se puede ofrecer de manera conjunta una planificación de horarios (*Production Timetable*), un seguimiento de vehículos (*Vehicle Monitoring*) y una gestión de conexiones/transbordos (*Connection Timetable* y *Connection Monitoring*); lo que se consigue mediante la agrupación de la información de los distintos medios de transporte, los distintos operadores que los gestionan y los distintos vehículos con los que dichos operadores ofrecen los servicios.

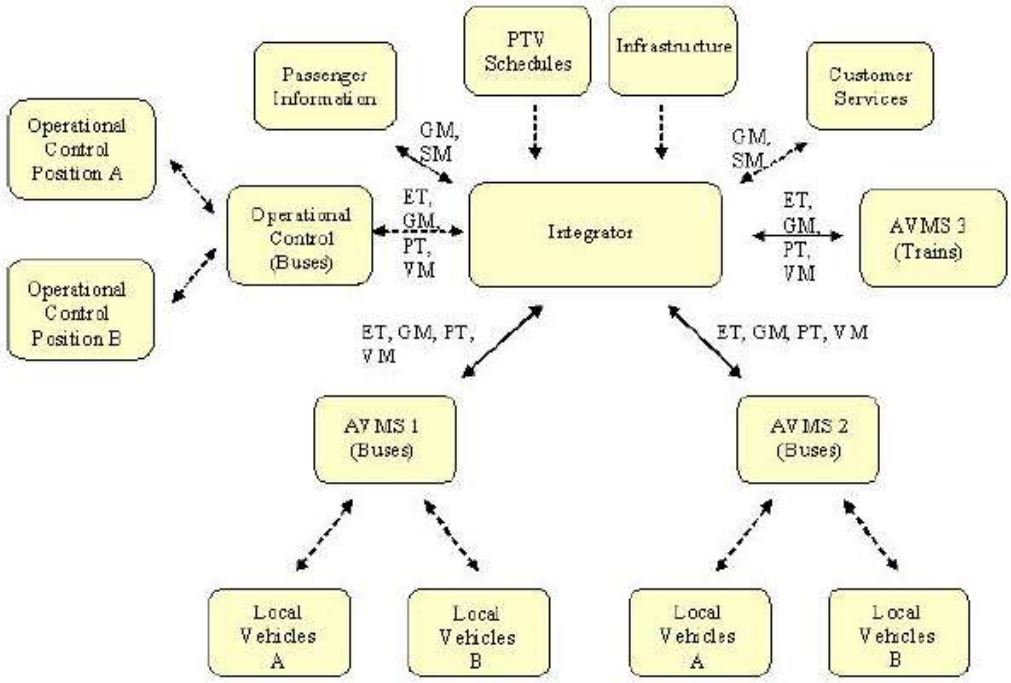


Ilustración 64: Esquema de la organización de los ITS en Copenhague (EN 15531-1)

5.5. Francia: Île de France

El objetivo que se está persiguiendo en Île de France (región francesa cuya capital es París) es lograr una distribución similar a la existente en Copenhague.

En la actualidad existe un sistema centralizado que agrupa la información de los distintos operadores locales y regionales, unificando la distribución de la información proveniente de estos, permitiendo un control operacional central y siendo usado para relacionarse con otros modelos de ITS.

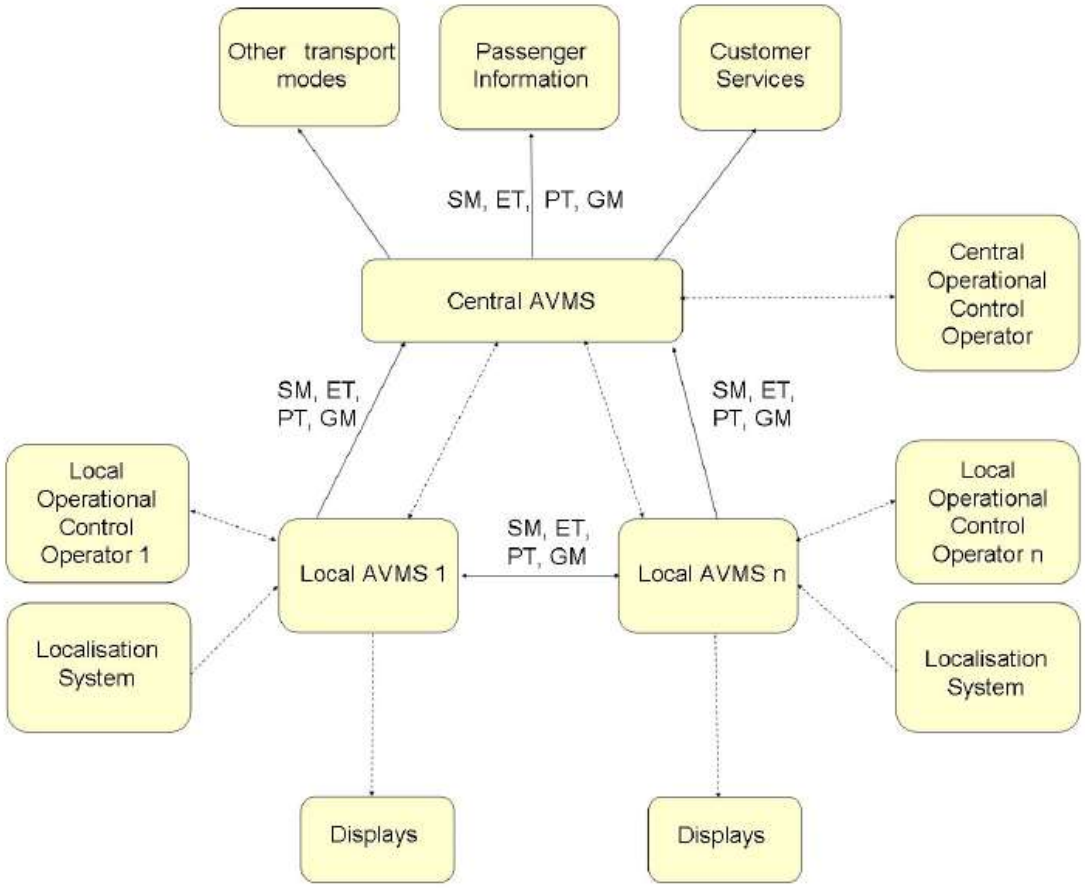


Ilustración 65: Esquema del estado actual de la organización de los ITS en la región de Paris (EN 15531-1)

La mejora que se pretende implementar es la citada relación con otros modelos de ITS.

Se desea añadir un integrador que centralice la comunicación entre el conjunto de sistemas locales y regionales (ya agrupados en el llamado AMVS 1 de la ilustración 66) y otras redes de transporte (autobuses y trenes nacionales, aviones, etc.), pero en este caso previendo la comunicación directa entre las distintas entidades para dar soporte a las conexiones/transbordos.

El esquema de la situación real que quieren alcanzar en Île de France es el siguiente:

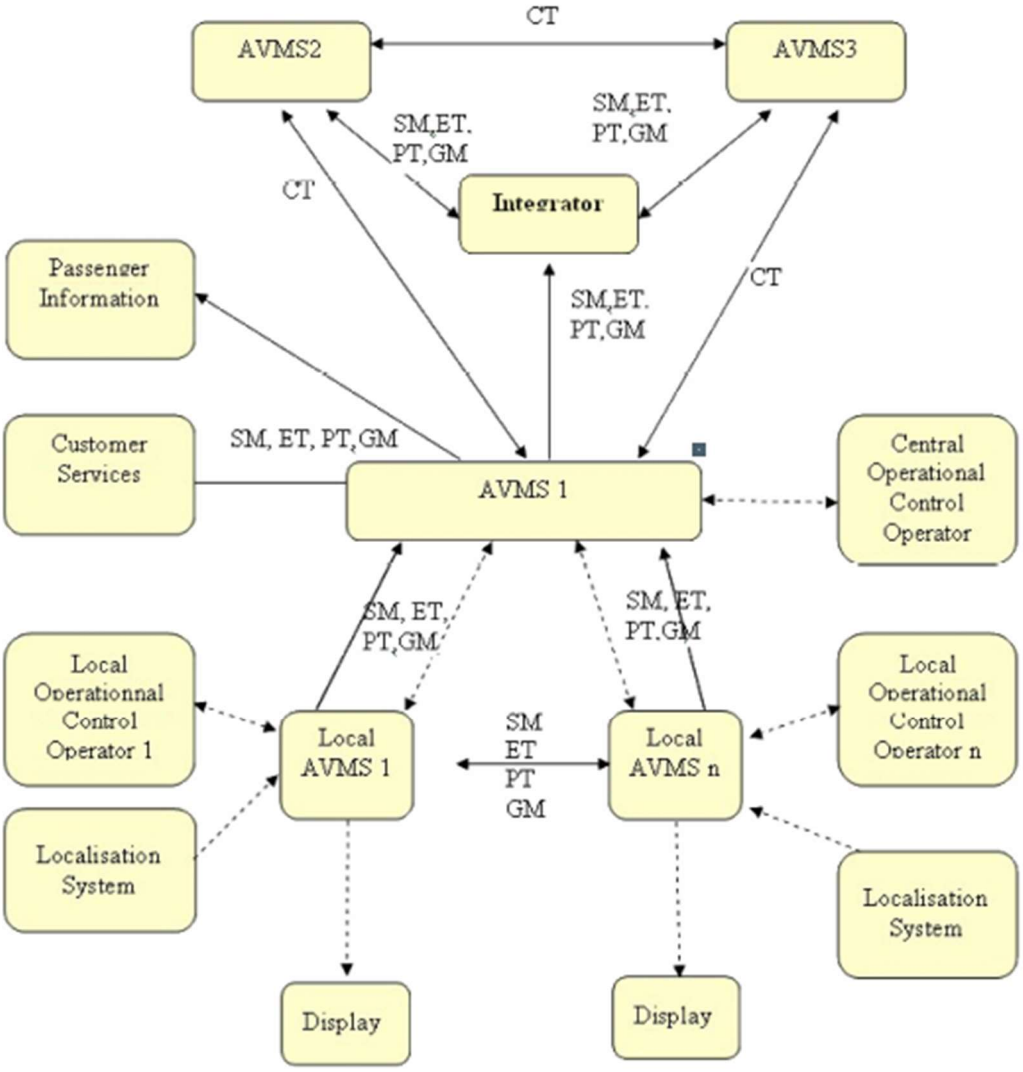


Ilustración 66: Esquema de la futura organización de los ITS en la región de Paris (EN 15531-1)

6. Conclusiones y líneas de mejora

6.1. Trabajo realizado

En este documento ha quedado reflejada la importancia que ha adquirido en las últimas décadas la información relativa al transporte público, lo que ha desembocado en una progresiva estandarización de los protocolos usados para el intercambio de dicha información.

Se ha realizado un estudio minucioso de SIRI, uno de los estándares en alza en la actualidad. En él se ha descrito el propósito general de este protocolo, así como los distintos servicios que ofrece y la utilidad de cada uno; descripciones que se han reforzado con ejemplos de datos reales de transporte público relativos a dichos servicios. Asimismo, se ha profundizado en los distintos mecanismos de comunicación y transporte de la información que contempla el estándar, explicando el funcionamiento de cada uno de ellos tanto de manera aislada como en combinación con el resto.

El análisis anterior se ha llevado a cabo con la intención de profundizar lo suficiente como para otorgar a un desarrollador el conocimiento necesario para implementar SIRI de manera correcta en un sistema de comunicaciones.

Paralelamente al citado desarrollo teórico se llevó a cabo la codificación de un servidor y un cliente capaces de comunicarse usando el estándar SIRI, implementaciones que se describen también de manera pormenorizada. De estas implementaciones se adjunta su código fuente (en el caso del cliente) y ejemplos resultado de la comunicación de datos reales que tiene lugar entre ellos en el marco empresarial en el que han sido desarrollados; a lo que hay que sumar ejemplos de su uso de manera independiente con aplicaciones públicas externas: validadores (que ejercen como clientes de SIRI) en el caso del servidor y fuentes públicas de datos (que ejercen como servidores de SIRI) en el caso del cliente.

Para concluir se ha realizado un estudio del uso de SIRI en el panorama internacional, acompañado de ejemplos públicos de su implementación y uso en distintas ciudades para la gestión de la información de transporte público que se genera y consume en ellas y en su entorno.

6.2. Líneas de mejora

Tras un análisis de las implementaciones descritas en este documento, se han localizado las siguientes líneas de mejora:

- Profundización en la implementación de los servicios funcionales ***Facility Monitoring*** y ***Service Exchange***. Debido al papel menos crítico del que gozan estos servicios en el contexto de un intercambio de información de transporte público, a su complejidad interna y a su más reciente introducción en la lista de servicios ofrecidos por el estándar; estos servicios no son un requisito exigido por los clientes de la empresa para la cual se han realizado las implementaciones. Por ello, su desarrollo ha sido llevado a cabo de manera somera en comparación con la implementación realizada para el resto de los servicios; lo que abre la puerta a una futura ampliación de los mismos.
- Implementación de la entrega indirecta o ***Fetched Delivery,*** tanto en las peticiones simples como en las subscripciones a los distintos servicios funcionales de SIRI. Las implementaciones de ambos patrones de intercambio de datos se han llevado a cabo siguiendo el método de entrega directa, o *Direct Delivery*. Si en el futuro se necesitase mejorar el rendimiento del servidor o de los clientes, el uso de la entrega indirecta sería sin duda uno de los caminos a explorar para conseguirlo.
- Implementación de un mecanismo de **autorización** y mejora del mecanismo de **autenticación** de los servicios. En el plugin implementado, dado que su puesta en producción no era inmediata, se ha incluido un mecanismo de autenticación muy básico consistente en un nombre usuario y contraseña, algo que tendrá que ser mejorado usando un cifrado potente para el envío y almacenamiento de las contraseñas. A este mecanismo se tendrá que sumar un mecanismo de autorización, que además de permitir restringir la interacción con el sistema (lo que nos ofrece la autenticación), permitirá restringir el acceso a ciertas partes del mismo (servicios funcionales, en nuestro caso) en función de los permisos que se tenga.
- Mejoras en los **tests**, que comprenden (al menos):
  - Implementación de tests unitarios en el servidor.
  - Transformación de los tests realizados en el cliente en verdaderos tests integrados del servidor, usando *mocks* debidamente.
  - Implementación de tests de carga del servidor.
  - Mejora en la estructura de los tests del cliente, completándolos con pruebas de la recepción de subscripciones en previsión un futuro cambio en la implementación de las mismas que implique que se dejen de rellenar las estructuras de información que satisfacen dichas subscripciones exactamente del mismo modo en que se rellenan las respuestas simples.

6.3. Conclusiones finales

Es mucho el conocimiento que he adquirido durante la realización de este proyecto, tanto en relación con los Sistemas Inteligentes de Transporte en general, como sobre el estándar del que trata este documento en particular.

A todo ese conocimiento teórico hay que sumar el conocimiento práctico obtenido con las implementaciones descritas, especialmente destacable en tanto han sido realizadas para una empresa puntera del sector ITS, integradas en un servidor real de gestión de datos de transporte público y supervisadas por gente con dilatada experiencia en el desarrollo de software relacionado con los Sistemas Inteligentes de Transporte.

En conclusión, considero que los objetivos marcados para el presente Trabajo Fin de Grado pueden considerarse cumplidos, habiendo además resultado la realización de éste académica y profesionalmente muy provechosa y satisfactoria.

7. Bibliografía

A continuación se presentan las referencias bibliográficas consultadas durante el desarrollo y redacción de este proyecto, por orden de aparición en el mismo:

[1] Web de ITxPT (*Information Technology for Public Transport*)
Disponible en: http://itxpt.org [Accedido: 17-10-2017]

[2] Web de SIRI
Disponible en: https://www.vdv.de/siri.aspx [Accedido: 02-08-2017]

[3] Web de Transmodel
Disponible en: http://transmodel-cen.eu [Accedido 04-09-2017]

[4] Web del Comité Técnico Europeo de Normalización CEN/TC 278 ITS
Disponible en: http://www.itsstandards.eu [Accedido 18-10-2017]

[5] Knowles, Nicholas. *TransXChange Schema Guide v2.5 – v59* [online]
Trapeze Group, 2014.
Disponible en:
http://naptan.dft.gov.uk/transxchange/schema/schemas.htm
[Accedido: 05-09-2017]

[6] Web de RTIG
Disponible en: http://rtig.org.uk [Accedido: 19-10-2017]

[7] Web de NeTEx
Disponible en: http://netex-cen.eu [Accedido: 04-09-2017]

[8] *Public transport - Network and Timetable Exchange* (NeTEx)
UNE-CEN/TS 16614-1:2014
UNE-CEN/TS 16614-2:2014
UNE-CEN/TS 16614-3:2016

[9] *Service Interface for Real-time Information relating to public transport operations* (SIRI)
CEN/TS 15531-1:2015.
CEN/TS 15531-2:2015
CEN/TS 15531-3:2015
CEN/TS 15531-4:2011
CEN/TS 15531-5:2016

[10] Web de GTFS
Disponible en: https://developers.google.com/transit/gtfs
[Accedido: 30-03-2017]

[11] Web de ejemplos de GTFS
Disponible en:
https://docs.google.com/document/d/16inL5BVcM1aU-_DcFJay_tC6Ni0wPa0nvQEstueG5k4 [Accedido: 30-03-2017]

[12] Web de GTFS-Realtime
Disponible en: https://developers.google.com/transit/gtfs-realtime
[Accedido: 10-04-2017]

[13] Transmodel
Transmodel v6 P1 – v10a
Transmodel v6 P2 – v10
Transmodel v6 P3 – v10

[14] CEN TC 278. *SIRI White Paper* [online]
Disponible en:
http://user47094.vs.easily.co.uk/siri/schema/1.0/doc/Siri White paper08.zip [Accedido: 03-05-2017]

[15] Web de SIRI (sin mantenimiento)
Disponible en: http://user47094.vs.easily.co.uk/siri
[Accedido: 03-05-2017]

[16] Knowles, Nicholas. *SIRI Handbook & Functional Service Diagrams* v0.13. KiZOOM, 2007.

[17] Wana, Thomas. wsdl2h(1) - *Linux man page*
Disponible en: https://linux.die.net/man/1/wsdl2h
[Accedido: 02-01-2018]

[18] Manual online de gSOAP 2.8.66
Disponible en: https://www.genivia.com/doc/soapdoc2.html
[Accedido: 02-01-2018]

[19] Web de Boost
Disponible en: https://www.boost.org [Accedido: 01-03-2017]

[20] "*Unary Expressions. Delete*" del Estándar de C++0x (C++11)
ISO/IEC 14882-2011

[21] Código fuente y documentación de Googletest
Disponible en: https://github.com/google/googletest
[Accedido: 13-05-2017]

[22] Web de ITS Factory
Disponible en: http://data.itsfactory.fi [Accedido: 05-03-2018]

[23] Web de la API de SIRI de *Tampere Public Transport e-services*
Disponible en:
http://developer.publictransport.tampere.fi/pages/en/siri.php
[Accedido: 05-03-2018]

[24] Web de UTA (*Utah Transit Authority*)
Disponible en: http://developer.rideuta.com [Accedido: 07-03-2018]

[25] Web de MTA Bus Time
Disponible en: http://bustime.mta.info/wiki/Developers/Index
[Accedido: 19-03-2018]

[26] Código fuente de SIRI-Validator
Disponible en: https://github.com/afimb/siri-validator
[Accedido: 02-07-2017]

[27] Etienne, Michel. *Guide d'Installation et d'Utilisation de l'application «client de test SIRI-Validator»* [online], AFIMB, 2015
Disponible en: http://docplayer.fr/11789706-Guide-d-installation-et-d-utilisation-de-l-application-client-de-test-siri-validator.html
[Accedido: 06-07-2017]

[28] Web de VDV
Disponible en: https://www.vdv.de/ist-daten-schnittstellen-en.aspx
[Accedido: 03-05-2018]

Anexos

Anexo 1: XML generado por peticiones y respuestas de los distintos servicios de SIRI

Production Timetable (PT)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ProductionTimetableRequest>
      <RequestTimestamp>2017-04-04T08:14:48.9639487Z</RequestTimestamp>
      <ValidityPeriod>
         <StartTime>2017-04-04T08:14:48.9639487Z</StartTime>
         <EndTime>2017-04-05T08:14:48.9639487Z</EndTime>
      </ValidityPeriod>
      <Lines>
         <LineDirection>
            <LineRef>200</LineRef>
         </LineDirection>
         <LineDirection>
            <LineRef>Blue_x0020_Mountains_x0020_Line</LineRef>
         </LineDirection>
         <LineDirection>
            <LineRef>F1_x0020_Manly</LineRef>
         </LineDirection>
      </Lines>
   </ProductionTimetableRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="utf-8"?>
<Siri
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.siri.org.uk/siri">
   <ServiceDelivery>
      <ResponseTimestamp>2017-04-04T12:32:49.189744+02:00</ResponseTimestamp>
      <ProducerRef>AVMLight</ProducerRef>
      <RequestMessageRef>b5bccf57-02ad-4e2d-a987-
b78ad9f91830</RequestMessageRef>
      <ProductionTimetableDelivery>
         <ResponseTimestamp>2017-04-04T12:32:49.189744+02:00</ResponseTimestamp>
         <DatedTimetableVersionFrame>
            <RecordedAtTime>2017-04-04T10:32:45.178995Z</RecordedAtTime>
            <LineRef>200</LineRef>
            <DirectionRef>inbound</DirectionRef>
            <OperatorRef>State_x0020_Transit_x0020_Sydney</OperatorRef>
            <ProductCategoryRef>Coach</ProductCategoryRef>
            <ServiceFeatureRef>ExpressService</ServiceFeatureRef>

<VehicleFeatureRef>Air_x0020_Condition_x0020_Available</VehicleFeatureRef>
            <DatedVehicleJourney>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-04-04T13:00:00Z</DataFrameRef>
                  <DatedVehicleJourneyRef>VJ_31-200-_-sj2-1-19-
TA</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <OperatorRef>State_x0020_Transit_x0020_Sydney</OperatorRef>

<VehicleFeatureRef>Air_x0020_Condition_x0020_Broken</VehicleFeatureRef>
               <DatedCalls>
                  <DatedCall>
                     <StopPointRef>202276</StopPointRef>
                     <VisitNumber>1</VisitNumber>
                     <AimedDepartureTime>2017-04-
04T13:00:00Z</AimedDepartureTime>
                  </DatedCall>
                  <DatedCall>
                     <StopPointRef>202513</StopPointRef>
                     <VisitNumber>2</VisitNumber>
                     <AimedArrivalTime>2017-04-04T13:03:00Z</AimedArrivalTime>
                     <AimedDepartureTime>2017-04-
04T13:03:00Z</AimedDepartureTime>
                  </DatedCall>
                  <DatedCall>
                     <StopPointRef>202514</StopPointRef>
                     <VisitNumber>3</VisitNumber>
                     <AimedArrivalTime>2017-04-04T13:03:00Z</AimedArrivalTime>
                     <AimedDepartureTime>2017-04-
04T13:03:00Z</AimedDepartureTime>
                  </DatedCall>
                  <DatedCall>
                     <StopPointRef>202515</StopPointRef>
                     <VisitNumber>4</VisitNumber>
                     <AimedArrivalTime>2017-04-04T13:04:00Z</AimedArrivalTime>
                     <AimedDepartureTime>2017-04-
04T13:04:00Z</AimedDepartureTime>
                  </DatedCall>
```

```
<DatedCall>
                        <StopPointRef>202516</StopPointRef>
                        <VisitNumber>5</VisitNumber>
                        <AimedArrivalTime>2017-04-04T13:05:00Z</AimedArrivalTime>
                        <AimedDepartureTime>2017-04-
04T13:05:00Z</AimedDepartureTime>
                     </DatedCall>
                     <DatedCall>
                        <StopPointRef>202517</StopPointRef>
                        <VisitNumber>6</VisitNumber>
                        <AimedArrivalTime>2017-04-04T13:05:00Z</AimedArrivalTime>
                        <AimedDepartureTime>2017-04-
04T13:05:00Z</AimedDepartureTime>
                     </DatedCall>
                  </DatedCalls>
               </DatedVehicleJourney>
               <DatedVehicleJourney>
                  <FramedVehicleJourneyRef>
                     <DataFrameRef>2017-04-05T08:30:00Z</DataFrameRef>
                     <DatedVehicleJourneyRef>VJ_9-F1-_-sj2-9-11-
F01</DatedVehicleJourneyRef>
                  </FramedVehicleJourneyRef>
                  <OperatorRef>Sydney_x0020_Ferries</OperatorRef>
                  <DatedCalls>
                     <DatedCall>
                        <StopPointRef>20004</StopPointRef>
                        <VisitNumber>1</VisitNumber>
                        <AimedDepartureTime>2017-04-
05T08:30:00Z</AimedDepartureTime>
                     </DatedCall>
                     <DatedCall>
                        <StopPointRef>20951</StopPointRef>
                        <VisitNumber>2</VisitNumber>
                        <AimedArrivalTime>2017-04-05T09:00:00Z</AimedArrivalTime>
                     </DatedCall>
                  </DatedCalls>
               </DatedVehicleJourney>
            </DatedTimetableVersionFrame>
         </ProductionTimetableDelivery>
      </ServiceDelivery>
</Siri>
```

Estimated Timetable (ET)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
    xmlns="http://www.siri.org.uk/siri"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
    <EstimatedTimetableRequest>
        <RequestTimestamp>2017-03-24T13:08:22.0049775Z</RequestTimestamp>
        <PreviewInterval>PT1H0S</PreviewInterval>
        <OperatorRef>Sydney_x0020_Trains</OperatorRef>
        <Lines>
            <LineDirection>

<LineRef>T1_x0020_North_x0020_Shore_x002C__x0020_Northern_x0020__x0026__x0020_We
stern_x0020_Line</LineRef>
            </LineDirection>
        </Lines>
    </EstimatedTimetableRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="utf-8"?>
<Siri
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://www.siri.org.uk/siri">
   <ServiceDelivery>
      <ResponseTimestamp>2017-03-24T12:36:18.014099+01:00</ResponseTimestamp>
      <ProducerRef>AVMLight</ProducerRef>
      <RequestMessageRef>0f4a97da-9260-4cf1-a332-
581b77047fba</RequestMessageRef>
      <EstimatedTimetableDelivery>
         <ResponseTimestamp>2017-03-24T12:36:18.014099+01:00</ResponseTimestamp>
         <SubscriptionRef>avm1_avm2_ET_1</SubscriptionRef>
         <EstimatedJourneyVersionFrame>
            <RecordedAtTime>2017-03-24T11:36:17.806162Z</RecordedAtTime>
            <EstimatedVehicleJourney>

<LineRef>T1_x0020_North_x0020_Shore_x002C__x0020_Northern_x0020__x0026__x0020_We
stern_x0020_Line</LineRef>
               <DirectionRef>inbound</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-03-24T11:11:01Z</DataFrameRef>
                  <DatedVehicleJourneyRef>VJ_2-T1-A-sj2-15-1414-
TA</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <OperatorRef>Sydney_x0020_Trains</OperatorRef>
               <Monitored>true</Monitored>
               <VehicleRef>VEHICLE-1</VehicleRef>
               <EstimatedCalls>
                  <EstimatedCall>
                     <StopPointRef>2067142</StopPointRef>
                     <VisitNumber>10</VisitNumber>
                     <AimedArrivalTime>2017-03-24T11:42:00Z</AimedArrivalTime>
                     <ExpectedArrivalTime>2017-03-
24T11:43:00Z</ExpectedArrivalTime>
                     <AimedDepartureTime>2017-03-
24T11:43:00Z</AimedDepartureTime>
                     <ExpectedDepartureTime>2017-03-
24T11:44:00Z</ExpectedDepartureTime>
                  </EstimatedCall>
               </EstimatedCalls>
            </EstimatedVehicleJourney>
         </EstimatedJourneyVersionFrame>
         <EstimatedJourneyVersionFrame>
            <RecordedAtTime>2017-03-24T11:36:17.806162Z</RecordedAtTime>
            <EstimatedVehicleJourney>

<LineRef>T1_x0020_North_x0020_Shore_x002C__x0020_Northern_x0020__x0026__x0020_We
stern_x0020_Line</LineRef>
               <DirectionRef>inbound</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-03-24T11:26:01Z</DataFrameRef>
                  <DatedVehicleJourneyRef>VJ_2-T1-A-sj2-15-1584-
TA</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <OperatorRef>Sydney_x0020_Trains</OperatorRef>
               <Monitored>true</Monitored>
               <VehicleRef>VEHICLE-2</VehicleRef>
               <EstimatedCalls>
```

```
                  <EstimatedCall>
                     <StopPointRef>211981</StopPointRef>
                     <VisitNumber>4</VisitNumber>
                     <AimedArrivalTime>2017-03-24T11:35:36Z</AimedArrivalTime>
                     <ExpectedArrivalTime>2017-03-
24T11:36:36Z</ExpectedArrivalTime>
                     <AimedDepartureTime>2017-03-
24T11:36:06Z</AimedDepartureTime>
                     <ExpectedDepartureTime>2017-03-
24T11:37:06Z</ExpectedDepartureTime>
                  </EstimatedCall>
               </EstimatedCalls>
            </EstimatedVehicleJourney>
         </EstimatedJourneyVersionFrame>
      </EstimatedTimetableDelivery>
   </ServiceDelivery>
</Siri>
```

Stop Timetable (ST)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <StopTimetableRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <DepartureWindow>
            <StartTime>2017-12-17T09:30:47-05:00</StartTime>
            <EndTime>2017-12-17T10:30:47-05:00</EndTime>
         </DepartureWindow>
         <MonitoringRef>EH00001</MonitoringRef>
         <LineRef>LINE77</LineRef>
      </StopTimetableRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <StopTimetableDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <SubscriberRef>NADER</SubscriberRef>
         <SubscriptionRef>2017-12-17T09:30:47-05:00</SubscriptionRef>
         <Status>true</Status>
         <ValidUntil>2017-12-17T09:30:47-05:00</ValidUntil>
         <TimetabledStopVisit>
            <RecordedAtTime>2017-12-17T09:25:46-05:00</RecordedAtTime>
            <MonitoringRef>HLTST011</MonitoringRef>
            <TargetedVehicleJourney>
               <LineRef>Line123</LineRef>
               <DirectionRef>Out</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-12-17</DataFrameRef>
                  <DatedVehicleJourneyRef>Oubound</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>123</PublishedLineName>
               <OriginRef>PLACE21</OriginRef>
               <OriginName>Highbury</OriginName>
               <DestinationRef>PLACE45</DestinationRef>
               <DestinationName>Paradise Park</DestinationName>
               <TargetedCall>
                  <VisitNumber>1</VisitNumber>
                  <AimedArrivalTime>2017-12-17T09:40:46-05:00</AimedArrivalTime>
                  <AimedDepartureTime>2017-12-17T09:42:47-
05:00</AimedDepartureTime>
               </TargetedCall>
            </TargetedVehicleJourney>
         </TimetabledStopVisit>
      </StopTimetableDelivery>
   </ServiceDelivery>
</Siri>
```

Stop Monitoring (SM)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <ServiceRequestContext>
         <Language>en</Language>
         <DataHorizon>P1Y2M3DT10H30M</DataHorizon>
         <RequestTimeout>P1Y2M3DT10H30M</RequestTimeout>
         <DeliveryMethod>direct</DeliveryMethod>
         <MultipartDespatch>true</MultipartDespatch>
         <ConfirmDelivery>false</ConfirmDelivery>
      </ServiceRequestContext>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <AccountId>MONTECHRISTO</AccountId>
      <AccountKey>2B|~2B</AccountKey>
      <RequestorRef>NADER</RequestorRef>
      <StopMonitoringRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <PreviewInterval>P10M</PreviewInterval>
         <MonitoringRef>EH00001</MonitoringRef>
         <OperatorRef>OPERATOR22</OperatorRef>
         <LineRef>LINE77</LineRef>
         <DirectionRef>OUTBOUND</DirectionRef>
         <DestinationRef>PLACE98765</DestinationRef>
         <StopVisitTypes>all</StopVisitTypes>
         <IncludeTranslations>true</IncludeTranslations>
         <MaximumStopVisits>7</MaximumStopVisits>
         <MinimumStopVisitsPerLine>2</MinimumStopVisitsPerLine>
         <MaximumTextLength>20</MaximumTextLength>
         <StopMonitoringDetailLevel>calls</StopMonitoringDetailLevel>
         <IncludeSituations>true</IncludeSituations>
      </StopMonitoringRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>KUBRICK</ProducerRef>
      <Status>true</Status>
      <MoreData>false</MoreData>
      <StopMonitoringDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <SubscriberRef>NADER</SubscriberRef>
         <SubscriptionRef>2017-12-17T09:30:47-05:00</SubscriptionRef>
         <Status>true</Status>
         <ValidUntil>2017-12-17T09:30:47-05:00</ValidUntil>
         <ShortestPossibleCycle>PT3M</ShortestPossibleCycle>
         <MonitoringRef>HLTST011</MonitoringRef>
         <MonitoringName>British Museum Stop A</MonitoringName>
         <!--====FIRST ARRIVAL ========================================== -->
         <MonitoredStopVisit>
            <RecordedAtTime>2017-12-17T09:25:46-05:00</RecordedAtTime>
            <ItemIdentifier>SED9843214675432</ItemIdentifier>
            <MonitoringRef>HLTST011</MonitoringRef>
            <ClearDownRef>CLR7654</ClearDownRef>
            <MonitoredVehicleJourney>
               <LineRef>Line123</LineRef>
               <DirectionRef>Out</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-12-17</DataFrameRef>
                  <DatedVehicleJourneyRef>Oubound</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>123</PublishedLineName>
               <OperatorRef>OP22</OperatorRef>
               <ProductCategoryRef>PDCATEXPRESS</ProductCategoryRef>
               <ServiceFeatureRef>SERVCCAT551</ServiceFeatureRef>
               <OriginRef>PLACE21</OriginRef>
               <OriginName>Highbury</OriginName>
               <Via>
                  <PlaceName>Kensall Green</PlaceName>
               </Via>
               <Via>
                  <PlaceName>Roman Road</PlaceName>
               </Via>
               <DestinationRef>PLACE45</DestinationRef>
               <DestinationName>Paradise Park</DestinationName>
               <VehicleJourneyName xml:lang="EN">Boris
Special</VehicleJourneyName>
               <JourneyNote>Kensall Green</JourneyNote>
               <Monitored>true</Monitored>
               <InCongestion>false</InCongestion>
               <InPanic>false</InPanic>
               <PredictionInaccurate>false</PredictionInaccurate>
               <ConfidenceLevel>reliable</ConfidenceLevel>
               <VehicleLocation>
                  <Longitude>180</Longitude>
                  <Latitude>90</Latitude>
               </VehicleLocation>
```

```
                <Bearing>23</Bearing>
                <Velocity>22</Velocity>
                <Occupancy>seatsAvailable</Occupancy>
                <Delay>PT2M</Delay>
                <ProgressStatus>Service running on time</ProgressStatus>
                <TrainBlockPart>
                   <NumberOfBlockParts>1</NumberOfBlockParts>
                   <TrainPartRef>3456</TrainPartRef>
                   <PositionOfTrainBlockPart>1</PositionOfTrainBlockPart>
                </TrainBlockPart>
                <BlockRef>BLOCK765</BlockRef>
                <CourseOfJourneyRef>RUN765</CourseOfJourneyRef>
                <VehicleRef>VEH987654</VehicleRef>
                <PreviousCalls>
                   <PreviousCall>
                      <StopPointRef>HLT0010</StopPointRef>
                      <VisitNumber>2</VisitNumber>
                      <StopPointName>String</StopPointName>
                      <VehicleAtStop>false</VehicleAtStop>
                      <AimedDepartureTime>2017-12-17T09:32:43-
05:00</AimedDepartureTime>
                      <ActualDepartureTime>2017-12-17T09:32:43-
05:00</ActualDepartureTime>
                   </PreviousCall>
                </PreviousCalls>
                <MonitoredCall>
                   <VisitNumber>0014</VisitNumber>
                   <VehicleAtStop>false</VehicleAtStop>
                   <VehicleLocationAtStop>
                      <Longitude>180</Longitude>
                      <Latitude>90</Latitude>
                   </VehicleLocationAtStop>
                   <AimedArrivalTime>2017-12-17T09:40:46-05:00</AimedArrivalTime>
                   <ExpectedArrivalTime>2017-12-17T09:40:46-
05:00</ExpectedArrivalTime>
                   <AimedDepartureTime>2017-12-17T09:42:47-
05:00</AimedDepartureTime>
                   <ExpectedDepartureTime>2017-12-17T09:40:47-
05:00</ExpectedDepartureTime>
                   <DeparturePlatformName>Bay 5</DeparturePlatformName>
                </MonitoredCall>
                <OnwardCalls>
                   <OnwardCall>
                      <StopPointRef>HLTST012</StopPointRef>
                      <VisitNumber>4</VisitNumber>
                      <StopPointName>Church</StopPointName>
                      <VehicleAtStop>false</VehicleAtStop>
                      <AimedArrivalTime>2017-12-17T09:30:56-
05:00</AimedArrivalTime>
                      <ExpectedArrivalTime>2017-12-17T09:30:56-
05:00</ExpectedArrivalTime>
                      <AimedDepartureTime>2017-12-17T09:30:57-
05:00</AimedDepartureTime>
                      <ExpectedDepartureTime>2017-12-17T09:30:57-
05:00</ExpectedDepartureTime>
                   </OnwardCall>
                </OnwardCalls>
                <IsCompleteStopSequence>true</IsCompleteStopSequence>
             </MonitoredVehicleJourney>
             <StopVisitNote>Hello bus for line 123 !</StopVisitNote>
          </MonitoredStopVisit>
          <!--====Second ARRIVAL ====This Vehicle is at the stop============= -->
          <MonitoredStopVisit>
             <RecordedAtTime>2017-12-17T09:25:46-06:00</RecordedAtTime>
             <!-- IDENTITY GROUP -->
             <ItemIdentifier>SED9843214675434</ItemIdentifier>
```

```
            <!-- MONITORED VEHICLE JOURNEY -->
            <MonitoredVehicleJourney>
               <!-- JOURNEY IDENTITY GROUP -->
               <LineRef>Line128</LineRef>
               <DirectionRef>Inbound</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-12-17</DataFrameRef>
                  <DatedVehicleJourneyRef>ABC576</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <!-- LINE INFO GROUP -->
               <PublishedLineName>String</PublishedLineName>
               <!-- SERVICE INFO GROUP -->
               <OperatorRef>OP22</OperatorRef>
               <ProductCategoryRef>PDCATEXPRESS</ProductCategoryRef>
               <ServiceFeatureRef>SERVCCAT551</ServiceFeatureRef>
               <!-- SERVICE POINTS GROUP -->
               <OriginRef>PLACE4675</OriginRef>
               <OriginName>Chancery Lane</OriginName>
               <Via>
                  <PlaceName>Picadilly Green</PlaceName>
               </Via>
               <DestinationRef>PLACE1245</DestinationRef>
               <DestinationName>Paradise Park</DestinationName>
               <!--JOURNEY INFO GROUP -->
               <JourneyNote>Kensall Green</JourneyNote>
               <!--JOURNEY PROGRESS GROUP -->
               <Monitored>true</Monitored>
               <InCongestion>false</InCongestion>
               <VehicleLocation>
                  <Longitude>180</Longitude>
                  <Latitude>90</Latitude>
               </VehicleLocation>
               <Delay>-PT2M</Delay>
               <ProgressStatus>Running Early</ProgressStatus>
               <!-- Operational Info GROUP -->
               <BlockRef>BLOCK6765</BlockRef>
               <CourseOfJourneyRef>RUN7865</CourseOfJourneyRef>
               <VehicleRef>VEH9876555</VehicleRef>
               <!-- MONITORED CALLING PATTERN GROUP -->
               <PreviousCalls>
                  <PreviousCall>
                     <StopPointRef>HLT0107</StopPointRef>
                     <VisitNumber>002</VisitNumber>
                     <StopPointName>Library</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <ActualDepartureTime>2017-12-17T09:30:43-
05:00</ActualDepartureTime>
                  </PreviousCall>
                  <PreviousCall>
                     <StopPointRef>HLT0109</StopPointRef>
                     <VisitNumber>007</VisitNumber>
                     <StopPointName>Library</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <ActualDepartureTime>2017-12-17T09:32:44-
05:00</ActualDepartureTime>
                  </PreviousCall>
                  <PreviousCall>
                     <StopPointRef>HLT0110</StopPointRef>
                     <VisitNumber>009</VisitNumber>
                     <StopPointName>Library</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <ActualDepartureTime>2017-12-17T09:38:47-
05:00</ActualDepartureTime>
                  </PreviousCall>
               </PreviousCalls>
               <MonitoredCall>
```

```
                  <VisitNumber>0012</VisitNumber>
                  <!-- STOP PROGRESS GROUP -->
                  <VehicleAtStop>true</VehicleAtStop>
                  <VehicleLocationAtStop>
                     <Longitude>180</Longitude>
                     <Latitude>90</Latitude>
                  </VehicleLocationAtStop>
                  <!-- STOP Visit TIMES GROUP -->
                  <ArrivalStatus>early</ArrivalStatus>
                  <AimedDepartureTime>2017-12-17T09:42:41-
05:00</AimedDepartureTime>
                  <ActualDepartureTime>2017-12-17T09:40:41-
05:00</ActualDepartureTime>
                  <DepartureStatus>onTime</DepartureStatus>
               </MonitoredCall>
               <OnwardCalls>
                  <OnwardCall>
                     <StopPointRef>HLTST112</StopPointRef>
                     <VisitNumber>0012</VisitNumber>
                     <StopPointName>Hospital</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <AimedArrivalTime>2017-12-17T09:55:47-
05:00</AimedArrivalTime>
                     <AimedDepartureTime>2017-12-17T09:56:47-
05:00</AimedDepartureTime>
                     <DepartureStatus>onTime</DepartureStatus>
                  </OnwardCall>
                  <OnwardCall>
                     <StopPointRef>HLTST113</StopPointRef>
                     <VisitNumber>0013</VisitNumber>
                     <StopPointName>Starbucks</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <AimedArrivalTime>2017-12-17T10:07:47-
05:00</AimedArrivalTime>
                     <AimedDepartureTime>2017-12-17T10:09:47-
05:00</AimedDepartureTime>
                  </OnwardCall>
                  <OnwardCall>
                     <StopPointRef>HLTST114</StopPointRef>
                     <VisitNumber>0014</VisitNumber>
                     <StopPointName>Station</StopPointName>
                     <VehicleAtStop>false</VehicleAtStop>
                     <AimedArrivalTime>2017-12-17T10:12:47-
05:00</AimedArrivalTime>
                     <AimedDepartureTime>2017-12-17T10:33:47-
05:00</AimedDepartureTime>
                  </OnwardCall>
               </OnwardCalls>
            </MonitoredVehicleJourney>
            <StopVisitNote>Hello bus for line 123 !</StopVisitNote>
         </MonitoredStopVisit>
```

```
         <!--====FIRST DELETION OF EARLIER ARRIVAL =====BY SYSTEM
KEY================== -->
         <MonitoredStopVisitCancellation>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ItemRef>SED9843214675429</ItemRef>
            <Reason>Arrived</Reason>
         </MonitoredStopVisitCancellation>
         <!--====Second DELETION OF EARLIER ARRIVAL ==IDENTIFERS ARE REF
DATA============= -->
         <MonitoredStopVisitCancellation>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <MonitoringRef>HLTST113</MonitoringRef>
            <VisitNumber>2</VisitNumber>
            <LineRef>Line123</LineRef>
            <DirectionRef>Out</DirectionRef>
            <VehicleJourneyRef>
               <DataFrameRef>2017-12-17</DataFrameRef>
               <DatedVehicleJourneyRef>0987656</DatedVehicleJourneyRef>
            </VehicleJourneyRef>
            <Reason>Arrived</Reason>
         </MonitoredStopVisitCancellation>
         <!--====STOP LINE NOTICE ============= -->
         <StopLineNotice>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ItemIdentifier>SED9843214675429</ItemIdentifier>
            <MonitoringRef>HLTST011</MonitoringRef>
            <LineRef>123</LineRef>
            <DirectionRef>Out</DirectionRef>
            <LineNote>What, will the line stretch out to the crack of
doom?</LineNote>
         </StopLineNotice>
         <!--====STOP LINE NOTICE CANCELLED============= -->
         <StopLineNoticeCancellation>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ItemRef>SED9843214675429</ItemRef>
            <MonitoringRef>HLTST011</MonitoringRef>
            <LineRef>123</LineRef>
            <DirectionRef>Out</DirectionRef>
         </StopLineNoticeCancellation>
         <Note>Hello Stop</Note>
      </StopMonitoringDelivery>
   </ServiceDelivery>
</Siri>
```

Vehicle Monitoring (VM)

Request

```
<?xml version="1.0" encoding="utf-8"?>
<Siri xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.siri.org.uk/siri">
   <ServiceRequest>
      <RequestTimestamp>2017-03-27T08:39:40.407855Z</RequestTimestamp>
      <RequestorRef>SIRITest</RequestorRef>
      <MessageIdentifier>717b4483-9bf0-42f0-b593-
80f53f919459</MessageIdentifier>
         <VehicleMonitoringRequest>
            <RequestTimestamp>2017-03-27T08:39:40.407855Z</RequestTimestamp>
            <VehicleRef>VEHICLE-22</VehicleRef>
            <DirectionRef>outbound</DirectionRef>
         </VehicleMonitoringRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<Siri
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.kizoom.com/standards/siri/schema/1.3/siri.xsd"
version="1.3"
   xmlns="http://www.siri.org.uk/siri">
   <ServiceDelivery>
      <ResponseTimestamp>2017-03-24T11:33:58+00:00</ResponseTimestamp>
      <ProducerRef>Producer</ProducerRef>
      <Status>true</Status>
      <MoreData>false</MoreData>
      <VehicleMonitoringDelivery version="1.3">
         <ResponseTimestamp>2017-03-24T11:33:58+00:00</ResponseTimestamp>
         <Status>true</Status>
         <VehicleActivity>
            <RecordedAtTime>2017-03-24T11:33:33+00:00</RecordedAtTime>
            <ValidUntilTime>2017-03-24T11:33:33+00:00</ValidUntilTime>
            <MonitoredVehicleJourney>
               <LineRef>69</LineRef>
               <DirectionRef>inbound</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-03-24</DataFrameRef>
                  <DatedVehicleJourneyRef>1110</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>69</PublishedLineName>
               <OperatorRef>SomeOperator</OperatorRef>
               <Monitored>true</Monitored>
               <VehicleLocation>
                  <Longitude>-2.167099</Longitude>
                  <Latitude>51.372858</Latitude>
               </VehicleLocation>
               <Bearing>113</Bearing>
            </MonitoredVehicleJourney>
            <Extensions>
               <VehicleJourney>
                  <Operational>
                     <TicketMachine>

<TicketMachineServiceCode>Zigzag</TicketMachineServiceCode>
                           <JourneyCode>1110</JourneyCode>
                     </TicketMachine>
                  </Operational>
                  <VehicleUniqueId>CN06BXL</VehicleUniqueId>
               </VehicleJourney>
            </Extensions>
         </VehicleActivity>
         <VehicleActivity></VehicleActivity>
         <VehicleActivity></VehicleActivity>
         <VehicleActivity></VehicleActivity>
         <VehicleActivity></VehicleActivity>
         <VehicleActivity>
            <RecordedAtTime>2017-03-23T12:54:09+00:00</RecordedAtTime>
            <ValidUntilTime>2017-03-23T12:54:09+00:00</ValidUntilTime>
            <MonitoredVehicleJourney>
               <LineRef>52</LineRef>
               <DirectionRef>outbound</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-03-23</DataFrameRef>
                  <DatedVehicleJourneyRef>1312</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>52</PublishedLineName>
               <OperatorRef>AB</OperatorRef>
```

```
                  <VehicleLocation>
                     <Longitude>-2.572699</Longitude>
                     <Latitude>51.450829</Latitude>
                  </VehicleLocation>
                  <Bearing>-1</Bearing>
               </MonitoredVehicleJourney>
               <Extensions>
                  <VehicleJourney>
                     <Operational>
                        <TicketMachine>
                           <TicketMachineServiceCode>052</TicketMachineServiceCode>
                           <JourneyCode>1312</JourneyCode>
                        </TicketMachine>
                     </Operational>
                     <VehicleUniqueId>T455PRH</VehicleUniqueId>
                  </VehicleJourney>
               </Extensions>
            </VehicleActivity>
         </VehicleMonitoringDelivery>
      </ServiceDelivery>
   </Siri>
```

Facility Monitoring (FM)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <ServiceRequestContext>
         <Language>en</Language>
         <DataHorizon>P1Y2M3DT10H30M</DataHorizon>
         <RequestTimeout>P1Y2M3DT10H30M</RequestTimeout>
         <DeliveryMethod>direct</DeliveryMethod>
         <MultipartDespatch>true</MultipartDespatch>
         <ConfirmDelivery>false</ConfirmDelivery>
      </ServiceRequestContext>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <FacilityMonitoringRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <PreviewInterval>P10M</PreviewInterval>
         <StopPointRef>PLACE98765</StopPointRef>

<MaximumNumberOfFacilityConditions>20</MaximumNumberOfFacilityConditions>
      </FacilityMonitoringRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (C) Copyright 2005-2012 CEN SIRI -->
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:acsb="http://www.ifopt.org.uk/acsb"
   xmlns:ifopt="http://www.ifopt.org.uk/ifopt"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <MoreData>false</MoreData>
      <FacilityMonitoringDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <SubscriberRef>NADER</SubscriberRef>
         <SubscriptionRef>2017-12-17T09:30:47-05:00</SubscriptionRef>
         <Status>true</Status>
         <ValidUntil>2017-12-17T09:30:47-05:00</ValidUntil>
         <ShortestPossibleCycle>PT3M</ShortestPossibleCycle>
         <FacilityCondition>
            <Facility
               xmlns:acsb="http://www.ifopt.org.uk/acsb"
               xmlns:ifopt="http://www.ifopt.org.uk/ifopt">
               <FacilityCode>134567-L4</FacilityCode>
               <Description xml:lang="FR">Les Halles sattion Lift 4
</Description>
               <FacilityClass> fixedEquipment</FacilityClass>
               <Features>
                  <Feature>
                     <AccessFacility> lift</AccessFacility>
                  </Feature>
               </Features>
               <OwnerRef>RATP</OwnerRef>
               <OwnerName>RATP</OwnerName>
               <ValidityCondition>
                  <Period>
                     <StartTime>2017-12-17T09:30:47-05:00</StartTime>
                     <EndTime>2017-12-17T11:30:47-05:00</EndTime>
                  </Period>
               </ValidityCondition>
               <FacilityLocation>
                  <LineRef>LINE4</LineRef>
                  <StopPointRef>HGALLES</StopPointRef>
                  <StopPlaceRef>HGALLES</StopPlaceRef>
                  <StopPlaceComponentId>ORT123</StopPlaceComponentId>
               </FacilityLocation>
               <Limitations>
                  <acsb:WheelchairAccess>true</acsb:WheelchairAccess>
                  <acsb:StepFreeAccess>true</acsb:StepFreeAccess>
                  <acsb:LiftFreeAccess>true</acsb:LiftFreeAccess>
               </Limitations>
               <Suitabilities>
                  <Suitability>
                     <acsb:Suitable>suitable</acsb:Suitable>
                     <acsb:UserNeed>
                        <acsb:MobilityNeed>wheelchair</acsb:MobilityNeed>
                     </acsb:UserNeed>
                  </Suitability>
               </Suitabilities>
            </Facility>
```

```
            <FacilityStatus>
               <Status>notAvailable</Status>
               <Description >Lift Broken</Description>
               <AccessibilityAssessment>

<acsb:MobilityImpairedAccess>true</acsb:MobilityImpairedAccess>
                  <acsb:Limitations>
                     <acsb:AccessibilityLimitation>
                        <acsb:ValidityCondition>
                           <ifopt:FromDateTime>2017-12-17T09:30:47-
05:00</ifopt:FromDateTime>
                           <ifopt:ToDateTime>2017-12-17T11:30:47-
05:00</ifopt:ToDateTime>
                        </acsb:ValidityCondition>
                        <acsb:WheelchairAccess>false</acsb:WheelchairAccess>
                        <acsb:StepFreeAccess>true</acsb:StepFreeAccess>
                        <acsb:LiftFreeAccess>true</acsb:LiftFreeAccess>
                        <ifopt:Extensions/>
                     </acsb:AccessibilityLimitation>
                  </acsb:Limitations>
                  <acsb:Suitabilities>
                     <acsb:Suitability>
                        <acsb:Suitable>notSuitable</acsb:Suitable>
                        <acsb:UserNeed>
                           <acsb:MobilityNeed>wheelchair</acsb:MobilityNeed>
                           <acsb:Excluded>true</acsb:Excluded>
                        </acsb:UserNeed>
                     </acsb:Suitability>
                  </acsb:Suitabilities>
               </AccessibilityAssessment>
            </FacilityStatus>
            <SituationRef>
               <SituationSimpleRef>13456</SituationSimpleRef>
            </SituationRef>
            <Remedy>
               <RemedyType> otherRoute</RemedyType>
            </Remedy>
         </FacilityCondition>
      </FacilityMonitoringDelivery>
   </ServiceDelivery>
</Siri>
```

Connection Timetable (CT)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <ConnectionTimetableRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <ArrivalWindow>
            <StartTime>2017-12-17T09:30:47-05:00</StartTime>
            <EndTime>2017-12-17T10:30:47-05:00</EndTime>
         </ArrivalWindow>
         <ConnectionLinkRef>EH00001</ConnectionLinkRef>
         <LineRef>LINE1</LineRef>
      </ConnectionTimetableRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2017/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <MoreData>false</MoreData>
      <ConnectionTimetableDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <RequestMessageRef>http://www.xmlspy.com</RequestMessageRef>
         <Status>true</Status>
         <ValidUntil>2017-12-17T09:30:47-05:00</ValidUntil>
         <TimetabledFeederArrival>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ConnectionLinkRef>98789</ConnectionLinkRef>
            <VisitNumber>2</VisitNumber>
            <StopPointName>Erehwon</StopPointName>
            <FeederJourney>
               <LineRef>123</LineRef>
               <DirectionRef>OUT</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>1967-08-13</DataFrameRef>
                  <DatedVehicleJourneyRef>09876</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>Line 123</PublishedLineName>
               <DirectionName>Outbound</DirectionName>
               <OperatorRef>123</OperatorRef>
               <ProductCategoryRef>School</ProductCategoryRef>
               <ServiceFeatureRef>CyclesPermitted</ServiceFeatureRef>
               <VehicleFeatureRef>LowFloors</VehicleFeatureRef>
               <OriginName>Purgatory</OriginName>
               <DestinationName>Paradise</DestinationName>
               <JourneyNote>from A to B</JourneyNote>
               <OriginAimedDepartureTime>2017-12-17T08:30:47-
05:00</OriginAimedDepartureTime>
               <DestinationAimedArrivalTime>2017-12-17T10:30:47-
05:00</DestinationAimedArrivalTime>
               <BlockRef>12345</BlockRef>
               <CourseOfJourneyRef>89765</CourseOfJourneyRef>
               <VehicleRef>V987</VehicleRef>
               <Monitored>true</Monitored>
               <AimedDepartureTime>2017-12-17T08:35:47-
05:00</AimedDepartureTime>
            </FeederJourney>
            <AimedArrivalTime>2017-12-17T09:30:47-05:00</AimedArrivalTime>
         </TimetabledFeederArrival>
         <TimetabledFeederArrivalCancellation>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ConnectionLinkRef>98789</ConnectionLinkRef>
            <VisitNumber>2</VisitNumber>
            <LineRef>123</LineRef>
            <DirectionRef>OUT</DirectionRef>
            <VehicleJourneyRef>
               <DataFrameRef>1967-08-13</DataFrameRef>
               <DatedVehicleJourneyRef>09876</DatedVehicleJourneyRef>
            </VehicleJourneyRef>
```

```
            <PublishedLineName>Line 123</PublishedLineName>
            <DirectionName>Outbound</DirectionName>
            <Reason>2017-12-17T09:30:47-05:00</Reason>
         </TimetabledFeederArrivalCancellation>
      </ConnectionTimetableDelivery>
   </ServiceDelivery>
</Siri>
```

Connection Monitoring (CM)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:gms="http://www.govtalk.gov.uk/CM/gms"
   xmlns:gms-xs="http://www.govtalk.gov.uk/CM/gms-xs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <ConnectionMonitoringRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <ConnectionLinkRef>EH00001</ConnectionLinkRef>
         <ConnectingJourneyFilter>
            <DatedVehicleJourneyRef>ABC56789</DatedVehicleJourneyRef>
            <TimetabledArrivalTime>2017-12-17T10:00:47-
05:00</TimetabledArrivalTime>
         </ConnectingJourneyFilter>
      </ConnectionMonitoringRequest>
      <ConnectionMonitoringRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <PreviewInterval>PT10M</PreviewInterval>
         <ConnectionLinkRef>EH00002</ConnectionLinkRef>
         <ConnectingTimeFilter>
            <LineRef>LINE77</LineRef>
            <DirectionRef>OUT</DirectionRef>
         </ConnectingTimeFilter>
      </ConnectionMonitoringRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <ConnectionMonitoringDistributorDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47.0Z</ResponseTimestamp>
         <RequestMessageRef>12345</RequestMessageRef>
         <Status>true</Status>
         <!--=======WAIT PROLONGED EVENT ============================== -->
         <WaitProlongedDeparture>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ConnectionLinkRef>HLKT00023</ConnectionLinkRef>
            <DistributorVisitNumber>001</DistributorVisitNumber>
            <DistributorJourney>
               <LineRef>ABC</LineRef>
               <DirectionRef>Out</DirectionRef>
            </DistributorJourney>
            <FeederVehicleJourneyRef>
               <DataFrameRef>2017-12-17</DataFrameRef>
               <DatedVehicleJourneyRef>8776654986</DatedVehicleJourneyRef>
            </FeederVehicleJourneyRef>
            <ExpectedDepartureTime>2017-12-17T09:30:47-
05:00</ExpectedDepartureTime>
         </WaitProlongedDeparture>
         <!--=======STOPPING POSITION CHANGED EVENT =============== -->
         <StoppingPositionChangedDeparture>
            <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
            <ConnectionLinkRef>HLKT00022</ConnectionLinkRef>
            <DistributorVisitNumber>001</DistributorVisitNumber>
            <DistributorJourney>
               <LineRef>A11C</LineRef>
               <DirectionRef>OUT</DirectionRef>
               <FramedVehicleJourneyRef>
                  <DataFrameRef>2017-12-17</DataFrameRef>
                  <DatedVehicleJourneyRef>09876</DatedVehicleJourneyRef>
               </FramedVehicleJourneyRef>
               <PublishedLineName>Line A11C</PublishedLineName>
               <DirectionName>Outbound</DirectionName>
               <OperatorRef>1s23</OperatorRef>
               <ProductCategoryRef>School</ProductCategoryRef>
               <ServiceFeatureRef>CyclesPermitted</ServiceFeatureRef>
               <VehicleFeatureRef>LowFloors</VehicleFeatureRef>
               <OriginName>Purgatory</OriginName>
               <DestinationName>Paradise</DestinationName>
               <JourneyNote>from A to B</JourneyNote>
               <OriginAimedDepartureTime>2017-12-17T08:30:47-
05:00</OriginAimedDepartureTime>
               <DestinationAimedArrivalTime>2017-12-17T10:30:47-
05:00</DestinationAimedArrivalTime>
               <BlockRef>12345</BlockRef>
               <CourseOfJourneyRef>89765</CourseOfJourneyRef>
```

```
<VehicleRef>V987</VehicleRef>
                <Monitored>true</Monitored>
                <AimedDepartureTime>2017-12-17T09:34:47-
05:00</AimedDepartureTime>
             </DistributorJourney>
             <FeederVehicleJourneyRef>
                <DataFrameRef>2017-12-17</DataFrameRef>
                <DatedVehicleJourneyRef>87766545677</DatedVehicleJourneyRef>
             </FeederVehicleJourneyRef>
             <ChangeNote>Will now leave from platform 6 </ChangeNote>
          </StoppingPositionChangedDeparture>
          <!--=======DEPARTURE CANCELLATION EVENT ================ -->
          <DistributorDepartureCancellation>
             <RecordedAtTime>2017-12-17T09:30:47-05:00</RecordedAtTime>
             <ConnectionLinkRef>987259</ConnectionLinkRef>
             <DistributorVisitNumber>2</DistributorVisitNumber>
             <DistributorJourney>
                <LineRef>123</LineRef>
                <DirectionRef>OUT</DirectionRef>
                <PublishedLineName>Line 123</PublishedLineName>
                <DirectionName>Outbound</DirectionName>
             </DistributorJourney>
             <FeederVehicleJourneyRef>
                <DataFrameRef>2017-12-17</DataFrameRef>
                <DatedVehicleJourneyRef>09867</DatedVehicleJourneyRef>
             </FeederVehicleJourneyRef>
             <Reason>Short staff</Reason>
          </DistributorDepartureCancellation>
       </ConnectionMonitoringDistributorDelivery>
    </ServiceDelivery>
</Siri>
```

Situation Exchange (SX)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <ServiceRequestContext>
         <Language>en</Language>
         <DataHorizon>P1Y2M3DT10H30M</DataHorizon>
         <RequestTimeout>P1Y2M3DT10H30M</RequestTimeout>
         <DeliveryMethod>direct</DeliveryMethod>
         <MultipartDespatch>true</MultipartDespatch>
         <ConfirmDelivery>false</ConfirmDelivery>
      </ServiceRequestContext>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Reques°tor</RequestorRef>
      <SituationExchangeRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <StartTime>2017-12-17T09:30:47-05:00 </StartTime>
         <VehicleMode>rail</VehicleMode>
         <Scope>network</Scope>
      </SituationExchangeRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:d2="http://datex2.eu/schema/2_0RC1/2_0" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceDelivery>
      <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <MoreData>false</MoreData>
      <SituationExchangeDelivery version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:46-05:00</ResponseTimestamp>
         <Situations>
            <PtSituationElement>
               <CreationTime>2017-12-17T09:30:47.0Z</CreationTime>
               <ParticipantRef>RAILCO01</ParticipantRef>
               <SituationNumber>000354</SituationNumber>
               <Version>0</Version>
               <References>
                  <RelatedToRef>
                     <CreationTime>2017-12-17T09:30:47.0Z</CreationTime>
                     <ParticipantRef>RAILCO01</ParticipantRef>
                     <SituationNumber>000354</SituationNumber>
                     <Version>0</Version>
                  </RelatedToRef>
               </References>
               <Source>
                  <SourceType>phone</SourceType>
                  <Phone>01223333337654</Phone>
                  <AgentReference>03274</AgentReference>
                  <TimeOfCommunication>2017-12-
17T09:30:47.0Z</TimeOfCommunication>
               </Source>
               <Verification>verified</Verification>
               <Progress>open</Progress>
               <QualityIndex>certain</QualityIndex>
               <ValidityPeriod>
                  <StartTime>2017-12-17T09:30:47.0Z</StartTime>
               </ValidityPeriod>
               <MiscellaneousReason>bombExplosion</MiscellaneousReason>
               <Severity>severe</Severity>
               <Audience>public</Audience>
               <ReportType>point</ReportType>
               <Summary overridden="true">Bomb at Barchester station</Summary>
               <Description overridden="true" xml:lang="EN">Building evacuated.
AVoid station untile further notice</Description>
               <Affects>
                  <Operators>
                     <AllOperators/>
                  </Operators>
                  <StopPoints>
                     <AffectedStopPoint>
                        <StopPointRef>BAAR0003</StopPointRef>
                        <StopPointName>Barchester Station</StopPointName>
                        <StopPointType>pti17_0</StopPointType>
                        <Location id="NMTOKEN" srsName="String">
                           <Longitude>-180</Longitude>
                           <Latitude>-90</Latitude>
```

```
                        <Precision>1</Precision>
                     </Location>
                  </AffectedStopPoint>
               </StopPoints>
               <StopPlaces>
                  <AffectedStopPlace>
                     <StopPlaceRef>BArF001</StopPlaceRef>
                     <AffectedComponents>
                        <AffectedComponent>
                           <ComponentRef>BAR00021</ComponentRef>
                           <ComponentName>Platfrom 3</ComponentName>
                        </AffectedComponent>
                     </AffectedComponents>
                  </AffectedStopPlace>
               </StopPlaces>
            </Affects>
            <Consequences>
               <Consequence>
                  <Period>
                     <StartTime>2017-12-17T09:30:47.0Z</StartTime>
                  </Period>
                  <Condition>pti13_0</Condition>
                  <Severity>pti26_0</Severity>
                  <Blocking>
                     <JourneyPlanner>true</JourneyPlanner>
                     <RealTime>true</RealTime>
                  </Blocking>
                  <Boarding>

<ArrivalBoardingActivity>noAlighting</ArrivalBoardingActivity>

<DepartureBoardingActivity>noBoarding</DepartureBoardingActivity>
                  </Boarding>
               </Consequence>
            </Consequences>
            <PublishingActions>
               <PublishToWebAction>
                  <Incidents>true</Incidents>
                  <HomePage>true</HomePage>
                  <Ticker>false</Ticker>
               </PublishToWebAction>
               <PublishToMobileAction>
                  <Incidents>true</Incidents>
                  <HomePage>false</HomePage>
               </PublishToMobileAction>
               <PublishToAlertsAction>
                  <ClearNotice>true</ClearNotice>
                  <ByEmail>true</ByEmail>
                  <ByMobile>true</ByMobile>
               </PublishToAlertsAction>
            </PublishingActions>
         </PtSituationElement>
         <!-- =============ROAD SITUATION ============ -->
         <RoadSituationElement>
            <CreationTime>2017-12-17T09:30:47.0Z</CreationTime>
            <ParticipantRef>RAILCO01</ParticipantRef>
            <SituationNumber>000354</SituationNumber>
            <Version>0</Version>
            <References>
               <RelatedToRef>
                  <CreationTime>2017-12-17T09:30:47.0Z</CreationTime>
                  <ParticipantRef>RAILCO01</ParticipantRef>
                  <SituationNumber>000354</SituationNumber>
                  <Version>0</Version>
               </RelatedToRef>
            </References>
```

```
                <Source>
                   <SourceType>phone</SourceType>
                   <Phone>01223333337654</Phone>
                   <AgentReference>03274</AgentReference>
                   <TimeOfCommunication>2017-12-
17T09:30:47.0Z</TimeOfCommunication>
                </Source>
                <Verification>verified</Verification>
                <Progress>open</Progress>
                <QualityIndex>certain</QualityIndex>
                <ValidityPeriod>
                   <StartTime>2017-12-17T09:30:47.0Z</StartTime>
                </ValidityPeriod>
                <MiscellaneousReason>bombExplosion</MiscellaneousReason>
                <Severity>severe</Severity>
                <Audience>public</Audience>
                <ReportType>point</ReportType>
                <Summary overridden="true">Jam on the access road to Barchester
Station</Summary>
                <Description overridden="true" xml:lang="EN">Grislock and
panic</Description>
                <Affects>
                   <Operators>
                      <AllOperators/>
                   </Operators>
                   <StopPoints>
                      <AffectedStopPoint>
                         <StopPointRef>BAAR0003</StopPointRef>
                         <StopPointName>Barchester Station</StopPointName>
                         <StopPointType>pti17_0</StopPointType>
                         <Location id="NMTOKEN" srsName="String">
                            <Longitude>-180</Longitude>
                            <Latitude>-90</Latitude>
                            <Precision>1</Precision>
                         </Location>
                      </AffectedStopPoint>
                   </StopPoints>
                   <StopPlaces>
                      <AffectedStopPlace>
                         <StopPlaceRef>BArF001</StopPlaceRef>
                         <AffectedComponents>
                            <AffectedComponent>
                               <ComponentRef>BAR00021</ComponentRef>
                               <ComponentName>Platfrom 3</ComponentName>
                            </AffectedComponent>
                         </AffectedComponents>
                      </AffectedStopPlace>
                   </StopPlaces>
                </Affects>
                <Consequences>
                   <Consequence>
                      <Period>
                         <StartTime>2017-12-17T09:30:47.0Z</StartTime>
                      </Period>
                      <Condition>delayed</Condition>
                      <Severity>severe</Severity>
                   </Consequence>
                </Consequences>
                <PublishingActions>
                   <PublishToWebAction>
                      <Incidents>true</Incidents>
                      <HomePage>true</HomePage>
                      <Ticker>false</Ticker>
                   </PublishToWebAction>
                   <PublishToMobileAction>
                      <Incidents>true</Incidents>
```

```
                     <HomePage>false</HomePage>
                  </PublishToMobileAction>
                  <PublishToAlertsAction>
                     <ClearNotice>true</ClearNotice>
                     <ByEmail>true</ByEmail>
                     <ByMobile>true</ByMobile>
                  </PublishToAlertsAction>
               </PublishingActions>
               <SituationRecord xsi:type="d2:Accident" id="GUID2A22530C-D452-
4ae8-B942-993BC2923D1">

<d2:situationRecordCreationReference>abc</d2:situationRecordCreationReference>
                  <d2:situationRecordCreationTime>2006-09-
28T16:00:00+01:00</d2:situationRecordCreationTime>
                  <d2:situationRecordVersion>1</d2:situationRecordVersion>
                  <d2:situationRecordVersionTime>2006-09-
28T16:05:00+00:00</d2:situationRecordVersionTime>
                  <d2:situationRecordFirstSupplierVersionTime>2006-09-
28T16:05:00+00:00</d2:situationRecordFirstSupplierVersionTime>

<d2:probabilityOfOccurrence>certain</d2:probabilityOfOccurrence>
                  <d2:source>
                     <d2:sourceIdentification>SRA</d2:sourceIdentification>
                     <d2:sourceName>
                        <d2:values>
                           <d2:value lang="se">Vägverket</d2:value>
                           <d2:value lang="en">Swedish Road
Administration</d2:value>
                        </d2:values>
                     </d2:sourceName>
                     <d2:sourceType>roadAuthorities</d2:sourceType>
                  </d2:source>
                  <d2:validity>

<d2:validityStatus>definedByValidityTimeSpec</d2:validityStatus>
                     <d2:validityTimeSpecification>
                        <d2:overallStartTime>2006-10-
17T14:00:00+02:00</d2:overallStartTime>
                        <d2:overallEndTime>2006-10-
17T16:00:00+02:00</d2:overallEndTime>
                     </d2:validityTimeSpecification>
                  </d2:validity>
                  <d2:impact>
                     <d2:numberOfLanesRestricted>1</d2:numberOfLanesRestricted>

<d2:numberOfOperationalLanes>3</d2:numberOfOperationalLanes>
                     <d2:originalNumberOfLanes>4</d2:originalNumberOfLanes>

<d2:trafficConstrictionType>roadBlocked</d2:trafficConstrictionType>
                  </d2:impact>
                  <d2:generalPublicComment>
                     <d2:comment>
                        <d2:value lang="se">Detta är en svensk olycka</d2:value>
                        <d2:value lang="en">This is a swedish
accident</d2:value>
                     </d2:comment>
                  </d2:generalPublicComment>
                  <d2:groupOfLocations>
                     <d2:locationContainedInGroup xsi:type="d2:Linear">
                        <d2:alertCLinear xsi:type="d2:AlertCMethod4Linear">

<d2:alertCLocationCountryCode>E</d2:alertCLocationCountryCode>

<d2:alertCLocationTableNumber>33</d2:alertCLocationTableNumber>

<d2:alertCLocationTableVersion>1</d2:alertCLocationTableVersion>
```

```
                              <d2:alertCDirection>

<d2:alertCDirectionCoded>positive</d2:alertCDirectionCoded>
                              </d2:alertCDirection>
                              <d2:alertCMethod4PrimaryPointLocation>
                                 <d2:alertCLocation>
                                    <d2:specificLocation>2030</d2:specificLocation>
                                 </d2:alertCLocation>
                                 <d2:offsetDistance>
                                    <d2:offsetDistance>10</d2:offsetDistance>
                                 </d2:offsetDistance>
                              </d2:alertCMethod4PrimaryPointLocation>
                              <d2:alertCMethod4SecondaryPointLocation>
                                 <d2:alertCLocation>
                                    <d2:specificLocation>2033</d2:specificLocation>
                                 </d2:alertCLocation>
                                 <d2:offsetDistance>
                                    <d2:offsetDistance>20</d2:offsetDistance>
                                 </d2:offsetDistance>
                              </d2:alertCMethod4SecondaryPointLocation>
                           </d2:alertCLinear>
                        </d2:locationContainedInGroup>
                     </d2:groupOfLocations>
                     <d2:accidentType>accident</d2:accidentType></SituationRecord>
                  </RoadSituationElement>
               </Situations>
            </SituationExchangeDelivery>
         </ServiceDelivery>
      </Siri>
```

General Message (GM)

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <ServiceRequest>
      <ServiceRequestContext>
         <Language>en</Language>
         <DataHorizon>P1Y2M3DT10H30M</DataHorizon>
         <RequestTimeout>P1Y2M3DT10H30M</RequestTimeout>
         <DeliveryMethod>direct</DeliveryMethod>
         <MultipartDespatch>true</MultipartDespatch>
         <ConfirmDelivery>false</ConfirmDelivery>
      </ServiceRequestContext>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <GeneralMessageRequest version="2.0">
         <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
         <InfoChannelRef>WARNING</InfoChannelRef>
      </GeneralMessageRequest>
   </ServiceRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
    xmlns="http://www.siri.org.uk/siri"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
    <ServiceDelivery>
        <ResponseTimestamp>2004-12-17T09:30:46-05:00</ResponseTimestamp>
        <ProducerRef>Sydney</ProducerRef>
        <Status>true</Status>
        <MoreData>false</MoreData>
        <GeneralMessageDelivery version="2.0">
            <ResponseTimestamp>2017-12-17T09:30:47.0Z</ResponseTimestamp>
            <RequestMessageRef>12345</RequestMessageRef>
            <Status>true</Status>
            <GeneralMessage formatRef="string">
                <RecordedAtTime>2017-12-17T09:30:47.0Z</RecordedAtTime>
                <InfoMessageIdentifier>00034567</InfoMessageIdentifier>
                <InfoMessageVersion>2</InfoMessageVersion>
                <InfoChannelRef>WARNINGS</InfoChannelRef>
                <ValidUntilTime>2017-12-17T09:30:47.0Z</ValidUntilTime>
                <Content>Beware the Ides of March</Content>
            </GeneralMessage>
            <GeneralMessageCancellation>
                <RecordedAtTime>2017-12-17T09:30:47.0Z</RecordedAtTime>
                <InfoMessageIdentifier>00034564</InfoMessageIdentifier>
                <InfoChannelRef>WARNINGS</InfoChannelRef>
            </GeneralMessageCancellation>
        </GeneralMessageDelivery>
    </ServiceDelivery>
</Siri>
```

Capabilities

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
    xmlns="http://www.siri.org.uk/siri"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
    <CapabilitiesRequest version="2.0">
        <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        <RequestorRef>12345</RequestorRef>
        <ProductionTimetableCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </ProductionTimetableCapabilitiesRequest>
        <EstimatedTimetableCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </EstimatedTimetableCapabilitiesRequest>
        <StopTimetableCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </StopTimetableCapabilitiesRequest>
        <StopMonitoringCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </StopMonitoringCapabilitiesRequest>
        <VehicleMonitoringCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </VehicleMonitoringCapabilitiesRequest>
        <ConnectionTimetableCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </ConnectionTimetableCapabilitiesRequest>
        <ConnectionMonitoringCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </ConnectionMonitoringCapabilitiesRequest>
        <GeneralMessageCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </GeneralMessageCapabilitiesRequest>
        <FacilityMonitoringCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </FacilityMonitoringCapabilitiesRequest>
        <SituationExchangeCapabilitiesRequest version="2.0">
            <RequestTimestamp>2017-12-17T09:30:47.0Z</RequestTimestamp>
        </SituationExchangeCapabilitiesRequest>
    </CapabilitiesRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <CapabilitiesResponse version="2.0">
      <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <ProductionTimetableCapabilitiesResponse version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <Status>true</Status>
         <ProductionTimetableServiceCapabilities>
            <GeneralInteraction>
               <Interaction>
                  <RequestResponse>true</RequestResponse>
                  <PublishSubscribe>false</PublishSubscribe>
               </Interaction>
               <Delivery>
                  <DirectDelivery>true</DirectDelivery>
                  <FetchedDelivery>false</FetchedDelivery>
               </Delivery>
               <MultipartDespatch>false</MultipartDespatch>
               <MultipleSubscriberFilter>true</MultipleSubscriberFilter>
               <HasConfirmDelivery>false</HasConfirmDelivery>
               <HasHeartbeat> true</HasHeartbeat>
               <VisitNumberisOrder>false</VisitNumberisOrder>
            </GeneralInteraction>
            <TransportDescription>
               <CommunicationsTransportMethod>
httpPost</CommunicationsTransportMethod>
               <CompressionMethod> none</CompressionMethod>
            </TransportDescription>
            <TopicFiltering>
               <FilterByValidityPeriod> true</FilterByValidityPeriod>
               <FilterByOperatorRef>true</FilterByOperatorRef>
               <FilterByLineRef>true</FilterByLineRef>
               <FilterByVersionRef>true</FilterByVersionRef>
            </TopicFiltering>
            <RequestPolicy>
               <NationalLanguage>en</NationalLanguage>
               <GmlCoordinateFormat>wgs84</GmlCoordinateFormat>
            </RequestPolicy>
            <SubscriptionPolicy>
               <HasIncrementalUpdates>true</HasIncrementalUpdates>
            </SubscriptionPolicy>
            <AccessControl>
               <RequestChecking>false</RequestChecking>
               <CheckOperatorRef>false</CheckOperatorRef>
               <CheckLineRef>false</CheckLineRef>
               <CheckConnectionLinkRef>false</CheckConnectionLinkRef>
            </AccessControl>
         </ProductionTimetableServiceCapabilities>
      </ProductionTimetableCapabilitiesResponse>
```

```
<EstimatedTimetableCapabilitiesResponse version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <Status>true</Status>
         <EstimatedTimetableServiceCapabilities>
            <GeneralInteraction>
               <Interaction>
                  <RequestResponse>true</RequestResponse>
                  <PublishSubscribe>false</PublishSubscribe>
               </Interaction>
               <Delivery>
                  <DirectDelivery>true</DirectDelivery>
                  <FetchedDelivery>false</FetchedDelivery>
               </Delivery>
               <MultipartDespatch>false</MultipartDespatch>
               <MultipleSubscriberFilter>true</MultipleSubscriberFilter>
               <HasConfirmDelivery>false</HasConfirmDelivery>
               <HasHeartbeat> true</HasHeartbeat>
               <VisitNumberisOrder>false</VisitNumberisOrder>
            </GeneralInteraction>
            <TransportDescription>
               <CommunicationsTransportMethod>
httpPost</CommunicationsTransportMethod>
               <CompressionMethod> none</CompressionMethod>
            </TransportDescription>
            <TopicFiltering>
               <FilterByOperatorRef>true</FilterByOperatorRef>
               <FilterByLineRef>true</FilterByLineRef>
            </TopicFiltering>
            <RequestPolicy>
               <NationalLanguage>en</NationalLanguage>
               <GmlCoordinateFormat>WGS84</GmlCoordinateFormat>
            </RequestPolicy>
            <SubscriptionPolicy>
               <HasIncrementalUpdates>true</HasIncrementalUpdates>
               <HasChangeSensitivity>true</HasChangeSensitivity>
            </SubscriptionPolicy>
            <AccessControl>
               <RequestChecking>false</RequestChecking>
               <CheckOperatorRef>false</CheckOperatorRef>
               <CheckLineRef>false</CheckLineRef>
               <CheckConnectionLinkRef>false</CheckConnectionLinkRef>
            </AccessControl>
         </EstimatedTimetableServiceCapabilities>
         <Extensions/>
      </EstimatedTimetableCapabilitiesResponse>

       (...)
```

```
<FacilityMonitoringCapabilitiesResponse version="2.0">
   <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
   <RequestMessageRef>abcde</RequestMessageRef>
   <Status> true</Status>
   <FacilityMonitoringServiceCapabilities>
      <GeneralInteraction>
         <Interaction>
            <RequestResponse>true</RequestResponse>
            <PublishSubscribe>false</PublishSubscribe>
         </Interaction>
         <Delivery>
            <DirectDelivery>true</DirectDelivery>
            <FetchedDelivery>false</FetchedDelivery>
         </Delivery>
         <MultipartDespatch>false</MultipartDespatch>
         <MultipleSubscriberFilter>true</MultipleSubscriberFilter>
         <HasConfirmDelivery>false</HasConfirmDelivery>
         <HasHeartbeat> true</HasHeartbeat>
         <VisitNumberisOrder>false</VisitNumberisOrder>
      </GeneralInteraction>
      <TransportDescription>
         <CommunicationsTransportMethod>
httpPost</CommunicationsTransportMethod>
         <CompressionMethod>none</CompressionMethod>
      </TransportDescription>
      <TopicFiltering>
         <DefaultPreviewInterval>PT60M</DefaultPreviewInterval>
         <FilterByFacilityRef>true</FilterByFacilityRef>
         <FilterByLocationRef>true</FilterByLocationRef>
         <FilterByVehicleRef>true</FilterByVehicleRef>
         <FilterByLineRef>true</FilterByLineRef>
         <FilterByStopPointRef>true</FilterByStopPointRef>
         <FilterByVehicleJourneyRef>true</FilterByVehicleJourneyRef>
         <FilterByConnectionLinkRef>false</FilterByConnectionLinkRef>
         <FilterByInterchangeRef>false</FilterByInterchangeRef>
         <FilterBySpecificNeed>true</FilterBySpecificNeed>
      </TopicFiltering>
      <RequestPolicy>
         <NationalLanguage>en</NationalLanguage>
         <GmlCoordinateFormat>WHS84</GmlCoordinateFormat>
         <HasMaximumFacilityStatus>true</HasMaximumFacilityStatus>
      </RequestPolicy>
      <SubscriptionPolicy>
         <HasIncrementalUpdates>false</HasIncrementalUpdates>
         <HasChangeSensitivity>false</HasChangeSensitivity>
      </SubscriptionPolicy>
      <AccessControl>
         <RequestChecking>false</RequestChecking>
         <CheckOperatorRef>false</CheckOperatorRef>
         <CheckLineRef>false</CheckLineRef>
      </AccessControl>
      <ResponseFeatures>
         <HasRemedy>true</HasRemedy>
         <HasFacilityLocation>true</HasFacilityLocation>
      </ResponseFeatures>
   </FacilityMonitoringServiceCapabilities>
</FacilityMonitoringCapabilitiesResponse>
```

```
(...)

<SituationExchangeCapabilitiesResponse version="2.0">
         <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
         <RequestMessageRef>abcde</RequestMessageRef>
         <Status> true</Status>
         <SituationExchangeServiceCapabilities>
            <GeneralInteraction>
               <Interaction>
                  <RequestResponse>true</RequestResponse>
                  <PublishSubscribe>false</PublishSubscribe>
               </Interaction>
               <Delivery>
                  <DirectDelivery>true</DirectDelivery>
                  <FetchedDelivery>false</FetchedDelivery>
               </Delivery>
               <MultipartDespatch>false</MultipartDespatch>
               <MultipleSubscriberFilter>true</MultipleSubscriberFilter>
               <HasConfirmDelivery>false</HasConfirmDelivery>
               <HasHeartbeat> true</HasHeartbeat>
               <VisitNumberisOrder>false</VisitNumberisOrder>
            </GeneralInteraction>
            <TransportDescription>
               <CommunicationsTransportMethod>
httpPost</CommunicationsTransportMethod>
               <CompressionMethod>none</CompressionMethod>
            </TransportDescription>
            <TopicFiltering>
               <DefaultPreviewInterval/>
               <FilterByFacilityRef>true</FilterByFacilityRef>
               <FilterByLocationRef>true</FilterByLocationRef>
               <FilterByVehicleRef>true</FilterByVehicleRef>
               <FilterByLineRef/>
               <FilterByStopPointRef/>
               <FilterByVehicleJourneyRef>true</FilterByVehicleJourneyRef>
               <FilterByConnectionLinkRef>true</FilterByConnectionLinkRef>
               <FilterByInterchangeRef>true</FilterByInterchangeRef>
               <FilterBySpecificNeed>true</FilterBySpecificNeed>
            </TopicFiltering>
            <RequestPolicy>
               <NationalLanguage>en</NationalLanguage>
               <GmlCoordinateFormat>WGS84</GmlCoordinateFormat>
               <HasMaximumFacilityStatus>true</HasMaximumFacilityStatus>
            </RequestPolicy>
            <SubscriptionPolicy>
               <HasIncrementalUpdates>true</HasIncrementalUpdates>
               <HasChangeSensitivity>true</HasChangeSensitivity>
            </SubscriptionPolicy>
            <AccessControl>
               <RequestChecking>false</RequestChecking>
               <CheckOperatorRef>false</CheckOperatorRef>
               <CheckLineRef>false</CheckLineRef>
            </AccessControl>
            <ResponseFeatures>
               <HasRemedy>true</HasRemedy>
               <HasFacilityLocation>true</HasFacilityLocation>
            </ResponseFeatures>
         </SituationExchangeServiceCapabilities>
      </SituationExchangeCapabilitiesResponse>
   </CapabilitiesResponse>
</Siri>
```

Lines Discovery

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <LinesRequest version="2.0">
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <OperatorRef>MYAGENCY</OperatorRef>
      <LinesDetailLevel>full</LinesDetailLevel>
   </LinesRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <LinesDelivery version="2.0">
      <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
      <Status>true</Status>
      <AnnotatedLineRef>
         <LineRef>Line564</LineRef>
         <LineName>564</LineName>
         <Monitored>true</Monitored>
         <Destinations>
            <Destination>
               <DestinationRef>PLace45</DestinationRef>
               <PlaceName>Paradise</PlaceName>
            </Destination>
         </Destinations>
         <Directions>
            <Direction>
               <DirectionRef>DIR01</DirectionRef>
               <DirectionName>Outbound</DirectionName>
               <JourneyPatterns>
                  <JourneyPattern>
                     <StopsInPattern>
                        <StopPointInPattern>
                           <StopPointRef>ST001</StopPointRef>
                           <StopName>Alpha Road</StopName>
                           <Lines>
                              <LineRef>Line564</LineRef>
                           </Lines>
                           <Location>
                              <Longitude>0.11</Longitude>
                              <Latitude>53.01</Latitude>
                           </Location>
                           <Order>1</Order>
                        </StopPointInPattern>
                        <StopPointInPattern>
                           <StopPointRef>ST003</StopPointRef>
                           <StopName>Charley Square</StopName>
                           <StopAreaRef>SA2001</StopAreaRef>
                           <Lines>
                              <LineDirection>
                                 <LineRef>Line564</LineRef>
                                 <DirectionRef>NORTH</DirectionRef>
                              </LineDirection>
                              <LineRef>44</LineRef>
                           </Lines>
                           <Location>
                              <Longitude>0.13</Longitude>
                              <Latitude>53.03</Latitude>
                           </Location>
                           <Order>2</Order>
                        </StopPointInPattern>
                     </StopsInPattern>
                  </JourneyPattern>
               </JourneyPatterns>
            </Direction>
            <Direction>
               <DirectionRef>DIR02</DirectionRef>
               <DirectionName>Inbound</DirectionName>
            </Direction>
         </Directions>
      </AnnotatedLineRef>
   </LinesDelivery>
</Siri>
```

StopPoints Discovery

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <StopPointsRequest version="2.0">
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
      <BoundingBox>
         <UpperLeft srsName="WGS84">
            <Longitude>0.1</Longitude>
            <Latitude>53.1</Latitude>
         </UpperLeft>
         <LowerRight srsName="WGS84">
            <Longitude>0.2</Longitude>
            <Latitude>53.2</Latitude>
         </LowerRight>
      </BoundingBox>
   </StopPointsRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <StopPointsDelivery version="2.0">
      <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
      <Status>true</Status>
      <AnnotatedStopPointRef>
         <StopPointRef>HLTS0001</StopPointRef>
         <Monitored>true</Monitored>
         <StopName>Stop 101</StopName>
         <Features>
            <ServiceFeature>
               <ServiceFeatureCode>S123</ServiceFeatureCode>
               <Name>Shelter</Name>
               <Icon>http://www.mybus.org/shelter.jpg</Icon>
            </ServiceFeature>
         </Features>
         <Lines>
            <LineRef>A23</LineRef>
         </Lines>
         <Location>
            <Longitude>0.1</Longitude>
            <Latitude>53</Latitude>
         </Location>
      </AnnotatedStopPointRef>
      <AnnotatedStopPointRef>
         <StopPointRef>HLTS0002</StopPointRef>
         <Monitored>true</Monitored>
         <StopName>Stop 102</StopName>
         <Features>
            <ServiceFeature>
               <ServiceFeatureCode>S123</ServiceFeatureCode>
               <Name>Shelter</Name>
               <Icon>http://www.mybus.org/shelter.jpg</Icon>
            </ServiceFeature>
         </Features>
         <Lines>
            <LineRef>A23</LineRef>
            <LineRef>A24</LineRef>
         </Lines>
         <Location>
            <Longitude>0.12</Longitude>
            <Latitude>53.1</Latitude>
         </Location>
      </AnnotatedStopPointRef>
   </StopPointsDelivery>
</Siri>
```

CheckStatus

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <CheckStatusRequest version="2.0">
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <RequestorRef>Requestor</RequestorRef>
   </CheckStatusRequest>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <CheckStatusResponse>
      <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
      <Status>true</Status>
      <ValidUntil>2017-12-17T19:30:47-05:00</ValidUntil>
      <ShortestPossibleCycle>PT2M</ShortestPossibleCycle>
      <ServiceStartedTime>2017-12-17T09:30:47-05:00</ServiceStartedTime>
   </CheckStatusResponse>
</Siri>
```

Heartbeat

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <HeartbeatNotification>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <ProducerRef>Sydney</ProducerRef>
      <Status>true</Status>
      <ValidUntil>2017-12-17T09:30:47-05:00</ValidUntil>
      <ShortestPossibleCycle>P1Y2M3DT10H30M</ShortestPossibleCycle>
      <ServiceStartedTime>2017-12-17T09:30:47-05:00</ServiceStartedTime>
   </HeartbeatNotification>
</Siri>
```

DataReady

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0">
   <DataReadyNotification>
      <RequestTimestamp>2017-12-17T09:30:47-05:00</RequestTimestamp>
      <ProducerRef>Sydney</ProducerRef>
   </DataReadyNotification>
</Siri>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
   xmlns="http://www.siri.org.uk/siri"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.0"
xsi:schemaLocation="http://www.siri.org.uk/siri ../../xsd/siri.xsd">
   <DataReadyAcknowledgement>
      <ResponseTimestamp>2017-12-17T09:30:47-05:00</ResponseTimestamp>
      <ConsumerRef>Consumer</ConsumerRef>
      <Status>true</Status>
   </DataReadyAcknowledgement>
</Siri>
```

Anexo 2: Código del cliente de SIRI implementado

Adjunto a este documento se proporciona (en formato digital) el **código fuente completo del cliente de SIRI implementado**.

Alternativamente, puede encontrarse el código fuente en los siguientes repositorios públicos:

- https://github.com/eduherminio/SIRI_Client
- https://github.com/eduherminio/SIRI_Notification_Consumer

Junto con el código fuente en formato digital se encuentra un ***script* de instalación** que descarga todas las dependencias necesarias para compilarlo.

Dichas dependencias, que no se adjuntan junto con el propio código por su gran tamaño, son las siguientes;

- `gSOAP 2.8.50.`
- `GoogleTest +1.8.0:` estado de la rama *master* del repositorio oficial de Google, a día 30/5/2018.
- `Boost v1.62.0.`

El *script* de instalación las descargará de sendos repositorios públicos en los que se encuentran 'congeladas' las versiones concretas que se han utilizado para la compilación del código de este proyecto.

Esos repositorios son:

- https://github.com/eduherminio/gsoap
- https://github.com/eduherminio/googletest
- https://github.com/eduherminio/boost_1_62_0

En el caso de GoogleTest y Boost, las variantes usadas para dicha compilación se encuentran además pre-compiladas en esos repositorios, de tal manera que pueden ser usadas inmediatamente después de ser descargadas.

El código fuente (C++) está preparado para ser compilado usando MSVC, garantizándose su compatibilidad con las versiones de dicho compilador incluida en las *Toolsets* v140 y v141 de Visual Studio/MSBuild.

Se adjuntan también los ejecutables (compilados para Winx64) de ambas aplicaciones, aunque sin el servidor de SIRI carecen por sí mismos de utilidad práctica.

Además del código fuente, los ejecutables resultado de la compilación del mismo y el *script* de instalación de las dependencias, se adjuntan los XSD de la versión de SIRI usada (v2.0.o) y los *scripts* con los que, a partir de la citada versión de gSOAP, se han generado los *wrappers* usados por cada una de las dos aplicaciones de nuestro cliente.

Anexo 2.1: Siri_Client

Debido a la gran extensión del código de Siri_Client, en este anexo se recogen las partes más representativas del mismo, omitido las implementaciones de aquellos métodos cuyo contenido es similar a otros que sí se adjuntan.

Sus declaraciones, no obstante, se han mantenido en todos los archivos *.h correspondientes.

El código completo puede consultarse en los archivos adjuntos a este documento, o en el siguiente repositorio público:

https://github.com/eduherminio/SIRI_Client

Estructura de archivos:

- SIRI_Client.cpp (*main*)
- Siri_ClientTestFixture.h / .cpp
- BaseSIRIClient.h /.cpp
- variables_globales.h
- Capabilities.h /.cpp
- Create_Subscriptions.h / .cpp
- Estimated_Timetable.h / .cpp
- Facility_Monitoring.h / .cpp
- Lines_Discovery.h / Lines_Discovery.cpp
- Production_Timetable.h / .cpp
- Stop_Monitoring.h / .cpp
- Stop_Monitoring_Multiple.h / .cpp
- Stop_Timetable.h / .cpp
- Stop_Discovery.h / .cpp
- Terminate_Subscription.h / .cpp
- Vehicle_Monitoring.h / .cpp

- stdsoap2.h / .cpp (gSOAP)
- duration.h / .c (gSOAP)
- struct_tm.h / .c (gSOAP)
- soapH.h (gSOAP, autogenerado)
- soapC.c (gSOAP, autogenerado)
- soapStub.h (gSOAP, autogenerado)
- soapSiriProducerDocBindingProxy.h / .cpp (gSOAP, autogenerados)
- targetver.h (Visual Studio)
- stdafx.h / stdafx.cpp (Visual Studio)
- SIRI_Client.vcxproj (Visual Studio)

SIRI_Client.cpp

```
// SiriClienteCpp.cpp

#include "stdafx.h"
#include <gtest/gtest.h>

#include "boost/date_time/gregorian/gregorian.hpp"
#include <boost/uuid/uuid.hpp>
#include <boost/uuid/uuid_generators.hpp>
#include <boost/filesystem.hpp>

#include <fstream>
#include <random>
#include <chrono>

namespace fs = boost::filesystem;

std::string    server_address        ("http://localhost:8080/");
std::wstring   consumer_address      (L"http://localhost:8081/");

bool siri_debug = true;      // true:   print debug files
bool siri_proxy = true;      // true:   proxy on 8888 must be ON (i.e. Fiddler
opened)
bool random_lines = true;    // true:   get_random_line() returns a random line from
available ones, rather than desired_line
bool random_stops = true;    // true:   get_random_line() returns a random stop from
available ones, rather than desired_stop

#ifdef UNICODE
   std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
   std::locale loc(std::locale::classic());
#endif

std::wstring user                (L"user");
std::wstring unauth_user         (L"unauthorised_user");
std::wstring pwd                 (L"password");

boost::gregorian::date start_date(2016, 12, 15);

long start_hour =            8;    // preferrably < simulation starting hour to test
all cases
int64_t desired_hours =      9;
int64_t too_many_hours =     25;   // max. 24
unsigned desired_days =      10;
unsigned too_many_days =     33;   // max. 31

std::wstring desired_language      (L"en");      // by default
std::wstring velocity_units        (L"km/h");

std::wstring desired_stop          (L"2031196");      // 10.   ALISON ROAD [2]
- 2031196
std::wstring non_existing_stop     (L"99999");

std::wstring desired_line          (L"1");
std::wstring non_existing_line     (L"99999");

std::wstring desired_vehicle       (L"1");
std::wstring non_existing_vehicle  (L"99999");

std::wstring desired_direction     (L"CIRCULAR QUAY [1]");      // Ida 1
```

```
std::wstring ALL_LINES_CODE         (L"0");
std::wstring ALL_STOPS_CODE         (L"0");
std::wstring WRONG_ALL_LINES_CODE   (L"1");
std::wstring WRONG_ALL_STOPS_CODE   (L"1");

std::wstring unauth_error_text                    (L"Unauthorised user");
std::wstring missing_parameter_error_text         (L"Missing parameter: ");
std::wstring too_many_days_error_text             (L"31");
std::wstring too_many_hours_error_text            (L"24");
std::wstring unknown_stop_error_text              (L"Unknown data: ");
std::wstring unknown_vehicle_error_text           (L"Unknown data: ");
std::wstring sub_refused_error_text               (L"Ill-formed request: sub
refused");
std::wstring all_subs_refused_error_text          (L"Ill-formed service request:
all subs refused");
std::wstring unknown_subscription_error_text      (L"Unknown subscription");
std::wstring unknown_subscriber_error_text        (L"Unknown subscriber");
std::wstring unknown_ALL_LINES_CODE_error_text    (L"Incorrect ALL_LINES_CODE");
std::wstring unknown_ALL_STOPS_CODE_error_text    (L"Incorrect ALL_STOPS_CODE");


int main(int argc, char* argv[])
{
   ::testing::InitGoogleTest(&argc, argv);
   testing::FLAGS_gtest_repeat = 10;
   int nRetCode = RUN_ALL_TESTS();

   //if(nRetCode)
      system("pause");

   return nRetCode;
}
```

Siri_ClientTestFixture.h

```
#pragma once

#include <gtest\gtest.h>

class Siri_ClientTestFixture :
   public ::testing::Test
{
protected:
   Siri_ClientTestFixture();
   virtual ~Siri_ClientTestFixture();

   virtual void SetUp();
   virtual void TearDown();
};
```

Siri_ClientTestFixture.cpp

```
#include "stdafx.h"
#include "Siri_ClientTestFixture.h"

Siri_ClientTestFixture::Siri_ClientTestFixture()
{
   // Before all tests
}

Siri_ClientTestFixture::~Siri_ClientTestFixture()
{
   // After all tests
}

void Siri_ClientTestFixture::SetUp()
{
   // Before each test
}

void Siri_ClientTestFixture::TearDown()
{
   // After each test
}
```

BaseSIRIClient.h

```
#pragma once

#include "variables_globales.h"

class CLines
{
public:
   CLines(std::wstring _lineref)
      :lineref(_lineref)   {}
   std::wstring lineref;
};

class CStops
{
public:
   CStops(std::wstring _stopref)
      :stopref(_stopref) {}
   std::wstring stopref;
};

class BaseSIRIClient
{
public:
   BaseSIRIClient();
   ~BaseSIRIClient();

   virtual void run() = 0;
protected:
   const CLines get_random_line();
   const CStops get_random_stop();
   void set_proxy(SiriProducerDocBindingProxy& srv);

   std::vector<std::wstring>         v_subscription_references;      //
v_subscription_uuids
   boost::uuids::random_generator      random_uuid_generator;         //
Subscription uuids   - re-using the generator avoids uuid conflicts
   std::uniform_int_distribution<int>   int_distribution;            // Subscriber
ids
   std::mt19937                  random_generator;
private:
   void load_available_lines();
   void load_available_stops();
   void load_v_subs();
   void save_v_subs();

   boost::filesystem::path v_sub_path;

   std::vector<CLines>      available_lines;
   std::vector<CStops>      available_stops;
};
```

BaseSIRIClient.cpp

```
#include "stdafx.h"
#include "BaseSIRIClient.h"

BaseSIRIClient::BaseSIRIClient()
{
   random_generator.seed((unsigned
int)std::chrono::high_resolution_clock::now().time_since_epoch().count());
   int_distribution = std::uniform_int_distribution<int>(0, 10);

   load_available_lines();
   load_available_stops();

   load_v_subs();
}

BaseSIRIClient::~BaseSIRIClient()
{
   save_v_subs();
}

void BaseSIRIClient::set_proxy(SiriProducerDocBindingProxy& srv)
{
   std::string      proxy_host("localhost");
   int            proxy_port(8888);

   if (siri_proxy == true)
   {
      srv.soap->proxy_host = proxy_host.c_str();
      srv.soap->proxy_port = proxy_port;
   }
}

const CLines BaseSIRIClient::get_random_line()
{
   std::uniform_int_distribution<int> rnd(0, (int)available_lines.size() - 1);

   return random_lines
      ? available_lines.empty() ? CLines(L"") :
available_lines[rnd(random_generator)]
      : CLines(desired_line);
}

const CStops BaseSIRIClient::get_random_stop()
{
   std::uniform_int_distribution<int> rnd(0, (int)available_stops.size() - 1);

   return random_stops
      ? available_stops.empty() ? CStops(L"") :
available_stops[rnd(random_generator)]
      : CStops(desired_stop);
}
```

```
void BaseSIRIClient::load_available_lines()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsLinesDiscoveryStructure request;
   ns1__WsLinesDiscoveryAnswerStructure response;

   ns2__LinesDiscoveryRequestStructure req;
   request.Request = &req;

   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.AccountId = &user;
   req.AccountKey = &pwd;

   //set_proxy(ws);
   if (ws.LinesDiscovery(server_address.c_str(), nullptr, &request, response) ==
SOAP_OK
      && response.Answer->ErrorCondition == nullptr
      && !response.Answer->AnnotatedLineRef.empty())
   {
      for (const auto& line : response.Answer->AnnotatedLineRef)
      {
         if (line != nullptr
            && line->LineRef != nullptr
            && !line->LineRef->__item.empty())
         {
            assert(line->LineRef->__item.find(L"L") != std::wstring::npos);
            std::wstring lineref(line->LineRef->__item.begin() + line->LineRef-
>__item.find(L"L") + 1,
               line->LineRef->__item.end());
            available_lines.push_back(CLines(lineref));
         }
      }
   }
}
```

```
void BaseSIRIClient::load_available_stops()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsStopPointsDiscoveryStructure request;
   ns1__WsStopPointsDiscoveryAnswerStructure response;

   ns2__StopPointsDiscoveryRequestStructure req;
   request.Request = &req;

   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.AccountId = &user;
   req.AccountKey = &pwd;

   //set_proxy(ws);
   if (ws.StopPointsDiscovery(server_address.c_str(), nullptr, &request,
response) == SOAP_OK
      && response.Answer->ErrorCondition == nullptr
      && !response.Answer->AnnotatedStopPointRef.empty())
   {
      for (const auto& stop : response.Answer->AnnotatedStopPointRef)
      {
         if (stop != nullptr
            && stop->StopPointRef != nullptr
            && !stop->StopPointRef->__item.empty())
         {
            std::wstring stoppointref(stop->StopPointRef->__item);
            available_stops.push_back(CStops(stoppointref));
         }
      }
   }
}

void BaseSIRIClient::load_v_subs()
{
   _tstring tmp_path = _T("c:/temp/siri/v_subs.txt");
   v_sub_path = tmp_path;

   std::ifstream ifs(v_sub_path.string());

   std::string str;
   while (getline(ifs, str))
   {
      if (str.empty())
         continue;
      std::wstring wstr(static_cast<const wchar_t*>(saer::ca2cw(str.c_str())));
      v_subscription_references.push_back(wstr);
   }

   ifs.close();
}

void BaseSIRIClient::save_v_subs()
{
   _ofstream ofs(v_sub_path.string()/*, std::ios::app*/);
   ofs.imbue(loc);

   for (const auto& sub : v_subscription_references)
      ofs << sub << std::endl;
   ofs.close();
}
```

variables_globales.h

```
#pragma once

#include "soapSiriProducerDocBindingProxy.h"
//#include "GMV configuration file reader"
#include "CACT/ca_ct.h"
#include <gtest/gtest.h>
#include <locale>
#include <fstream>
#include <random>
#include <chrono>
#include <boost/filesystem.hpp>
#include <boost/chrono.hpp>
#include <boost/date_time.hpp>
#include <boost/uuid/uuid.hpp>
#include <boost/uuid/uuid_io.hpp>
#include <boost/uuid/uuid_generators.hpp>


namespace fs = boost::filesystem;

#define auth_user    user
#define auth_pwd     pwd

extern bool siri_debug;
extern bool siri_proxy;
extern bool random_lines;
extern bool random_stops;

extern std::string  server_address;
extern std::wstring consumer_address;

extern std::locale  loc;

extern std::wstring ALL_LINES_CODE;
extern std::wstring ALL_STOPS_CODE;
extern std::wstring WRONG_ALL_LINES_CODE;
extern std::wstring WRONG_ALL_STOPS_CODE;

extern std::wstring user;
extern std::wstring unauth_user;
extern std::wstring pwd;
extern std::wstring desired_stop;
extern std::wstring non_existing_stop;
extern std::wstring desired_line;
extern std::wstring non_existing_line;
extern std::wstring desired_vehicle;
extern std::wstring non_existing_vehicle;
extern std::wstring desired_direction;
extern std::wstring velocity_units;
extern std::wstring desired_language;

extern boost::gregorian::date start_date;
extern long start_hour;
extern int64_t desired_hours;
extern int64_t too_many_hours;
extern unsigned desired_days;
extern unsigned too_many_days;
```

```
extern std::wstring unauth_error_text;
extern std::wstring missing_parameter_error_text;
extern std::wstring too_many_days_error_text;
extern std::wstring too_many_hours_error_text;
extern std::wstring unknown_stop_error_text;
extern std::wstring unknown_vehicle_error_text;
extern std::wstring unknown_ALL_LINES_CODE_error_text;
extern std::wstring unknown_ALL_STOPS_CODE_error_text;

extern std::wstring sub_refused_error_text;
extern std::wstring all_subs_refused_error_text;
extern std::wstring unknown_subscriber_error_text;
extern std::wstring unknown_subscription_error_text;
```

Capabilities.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_CAP_Base
{
public:
	Siri_CAP_Base() {}
	virtual ~Siri_CAP_Base() {};
};


class Siri_CAP_Main :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_Main()
		: BaseSIRIClient() {}
	~Siri_CAP_Main() {};

	virtual void run() override;
};

class Siri_CAP_MissingParameter_PT :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_PT()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_PT() {};

	virtual void run() override;
};

class Siri_CAP_MissingParameter_ST :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_ST()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_ST() {};

	virtual void run() override;
};

class Siri_CAP_MissingParameter_SM :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_SM()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_SM() {};

	virtual void run() override;
};
```

```
class Siri_CAP_MissingParameter_VM :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_VM()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_VM() {};

	virtual void run() override;
};

class Siri_CAP_MissingParameter_ET :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_ET()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_ET() {};

	virtual void run() override;
};

class Siri_CAP_MissingParameter_FM :
	public BaseSIRIClient,
	public Siri_CAP_Base
{
public:
	Siri_CAP_MissingParameter_FM()
		: BaseSIRIClient() {}
	~Siri_CAP_MissingParameter_FM() {};

	virtual void run() override;
};
```

Capabilities.cpp

```
#include "stdafx.h"
#include "Capabilities.h"

TEST_F(Siri_ClientTestFixture, CAP_Main)              {   Siri_CAP_Main
CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_PT)      {
Siri_CAP_MissingParameter_PT      CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_ST)      {
Siri_CAP_MissingParameter_ST      CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_SM)      {
Siri_CAP_MissingParameter_SM      CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_VM)      {
Siri_CAP_MissingParameter_VM      CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_ET)      {
Siri_CAP_MissingParameter_ET      CAP;      CAP.run();   }
TEST_F(Siri_ClientTestFixture, CAP_MissingParameter_FM)      {
Siri_CAP_MissingParameter_FM      CAP;      CAP.run();   }


void Siri_CAP_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsGetCapabilitiesRequestStructure request;
   ns1__WsGetCapabilitiesAnswerStructure response;

   ns2__CapabilitiesRequestStructure req;
   request.Request= &req;

   xsd__dateTime__ request_timestamp=
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   req.RequestTimestamp = request_timestamp;

   _ns2__ProductionTimetableCapabilitiesRequest* production_timetable = new
_ns2__ProductionTimetableCapabilitiesRequest;
      production_timetable->RequestTimestamp = request_timestamp;
   req.ProductionTimetableCapabilitiesRequest = production_timetable;

   ns2__ServiceCapabilitiesRequestStructure* stop_timetable = new
ns2__ServiceCapabilitiesRequestStructure;
      stop_timetable->RequestTimestamp = request_timestamp;
   req.StopTimetableCapabilitiesRequest = stop_timetable;

   ns2__ServiceCapabilitiesRequestStructure* stop_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
      stop_monitoring->RequestTimestamp = request_timestamp;
   req.StopMonitoringCapabilitiesRequest = stop_monitoring;

   ns2__ServiceCapabilitiesRequestStructure* vehicle_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
      vehicle_monitoring->RequestTimestamp = request_timestamp;
   req.VehicleMonitoringCapabilitiesRequest = vehicle_monitoring;

   ns2__ServiceCapabilitiesRequestStructure* estimated_timetable = new
ns2__ServiceCapabilitiesRequestStructure;
      estimated_timetable->RequestTimestamp = request_timestamp;
   req.EstimatedTimetableCapabilitiesRequest = estimated_timetable;
```

```
ns2__ServiceCapabilitiesRequestStructure* facility_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
       facility_monitoring->RequestTimestamp = request_timestamp;
   req.FacilityMonitoringCapabilitiesRequest = facility_monitoring;

   set_proxy(ws);
   ASSERT_EQ(ws.GetCapabilities(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete production_timetable;
   delete stop_timetable;
   delete stop_monitoring;
   delete vehicle_monitoring;
   delete estimated_timetable;
   delete facility_monitoring;

   ASSERT_NE(response.Answer, nullptr);
   ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse, nullptr);
   ASSERT_NE(response.Answer->StopTimetableCapabilitiesResponse,      nullptr);
   ASSERT_NE(response.Answer->StopMonitoringCapabilitiesResponse,      nullptr);
   ASSERT_NE(response.Answer->VehicleMonitoringCapabilitiesResponse,   nullptr);
   ASSERT_NE(response.Answer->EstimatedTimetableCapabilitiesResponse,   nullptr);
   ASSERT_NE(response.Answer->FacilityMonitoringCapabilitiesResponse,   nullptr);

   ASSERT_EQ(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition, nullptr);
   ASSERT_EQ(response.Answer->StopTimetableCapabilitiesResponse->
ErrorCondition, nullptr);
   ASSERT_EQ(response.Answer->StopMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);
   ASSERT_EQ(response.Answer->VehicleMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);
   ASSERT_EQ(response.Answer->EstimatedTimetableCapabilitiesResponse->
ErrorCondition, nullptr);
   ASSERT_EQ(response.Answer->FacilityMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);

   ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ProductionTimetableServiceCapabilities->TransportDescription, nullptr);
   ASSERT_NE(response.Answer->StopTimetableCapabilitiesResponse->
StopTimetableServiceCapabilities->      TransportDescription, nullptr);
   ASSERT_NE(response.Answer->StopMonitoringCapabilitiesResponse->
StopMonitoringServiceCapabilities->      TransportDescription, nullptr);
   ASSERT_NE(response.Answer->VehicleMonitoringCapabilitiesResponse->
VehicleMonitoringServiceCapabilities->   TransportDescription, nullptr);
   ASSERT_NE(response.Answer->EstimatedTimetableCapabilitiesResponse->
EstimatedTimetableServiceCapabilities->   TransportDescription, nullptr);
   ASSERT_NE(response.Answer->FacilityMonitoringCapabilitiesResponse->
FacilityMonitoringServiceCapabilities->   TransportDescription, nullptr);
}
```

```
void Siri_CAP_MissingParameter_PT::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsGetCapabilitiesRequestStructure request;
   ns1__WsGetCapabilitiesAnswerStructure response;

   ns2__CapabilitiesRequestStructure req;
   request.Request= &req;

   xsd__dateTime__ request_timestamp=
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   req.RequestTimestamp = request_timestamp;

   //_ns2__ProductionTimetableCapabilitiesRequest* production_timetable = new
_ns2__ProductionTimetableCapabilitiesRequest;
   //   production_timetable->RequestTimestamp = request_timestamp;
   //   req.ProductionTimetableCapabilitiesRequest = production_timetable;

   ns2__ServiceCapabilitiesRequestStructure* stop_timetable = new
ns2__ServiceCapabilitiesRequestStructure;
      stop_timetable->RequestTimestamp = request_timestamp;
   req.StopTimetableCapabilitiesRequest = stop_timetable;

   ns2__ServiceCapabilitiesRequestStructure* stop_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
      stop_monitoring->RequestTimestamp = request_timestamp;
   req.StopMonitoringCapabilitiesRequest = stop_monitoring;

   ns2__ServiceCapabilitiesRequestStructure* vehicle_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
      vehicle_monitoring->RequestTimestamp = request_timestamp;
   req.VehicleMonitoringCapabilitiesRequest = vehicle_monitoring;

   ns2__ServiceCapabilitiesRequestStructure* estimated_timetable = new
ns2__ServiceCapabilitiesRequestStructure;
      estimated_timetable->RequestTimestamp = request_timestamp;
   req.EstimatedTimetableCapabilitiesRequest = estimated_timetable;

   ns2__ServiceCapabilitiesRequestStructure* facility_monitoring = new
ns2__ServiceCapabilitiesRequestStructure;
      facility_monitoring->RequestTimestamp = request_timestamp;
   req.FacilityMonitoringCapabilitiesRequest = facility_monitoring;

   //set_proxy(ws);
   ASSERT_EQ(ws.GetCapabilities(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   //delete production_timetable;
   delete stop_timetable;
   delete stop_monitoring;
   delete vehicle_monitoring;
   delete estimated_timetable;
   delete facility_monitoring;

   ASSERT_NE(response.Answer, nullptr);
   ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse, nullptr);
   ASSERT_NE(response.Answer->StopTimetableCapabilitiesResponse,      nullptr);
   ASSERT_NE(response.Answer->StopMonitoringCapabilitiesResponse,      nullptr);
   ASSERT_NE(response.Answer->VehicleMonitoringCapabilitiesResponse,   nullptr);
   ASSERT_NE(response.Answer->EstimatedTimetableCapabilitiesResponse,
nullptr);
   ASSERT_NE(response.Answer->FacilityMonitoringCapabilitiesResponse,
nullptr);
```

```
    ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition, nullptr);
    ASSERT_EQ(response.Answer->StopTimetableCapabilitiesResponse->
ErrorCondition, nullptr);
    ASSERT_EQ(response.Answer->StopMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);
    ASSERT_EQ(response.Answer->VehicleMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);
    ASSERT_EQ(response.Answer->EstimatedTimetableCapabilitiesResponse->
ErrorCondition, nullptr);
    ASSERT_EQ(response.Answer->FacilityMonitoringCapabilitiesResponse->
ErrorCondition, nullptr);

    ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition, nullptr);
    ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition->Description->__item.find(missing_parameter_error_text +
std::wstring(L"production_timetable")), std::wstring::npos);
    ASSERT_EQ(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition->__union_ServiceDeliveryErrorConditionStructure___,
SOAP_UNION__ns2__union_ServiceDeliveryErrorConditionStructure____ParametersIgnor
edError);
    ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition-
>union_ServiceDeliveryErrorConditionStructure___.ParametersIgnoredError-
>ErrorText->find(missing_parameter_error_text +
std::wstring(L"production_timetable")), std::wstring::npos);
    ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ErrorCondition-
>union_ServiceDeliveryErrorConditionStructure___.ParametersIgnoredError-
>ParameterName.front().find(std::wstring(L"production_timetable")),
std::wstring::npos);

    //ASSERT_NE(response.Answer->ProductionTimetableCapabilitiesResponse-
>ProductionTimetableServiceCapabilities->TransportDescription, nullptr);
    ASSERT_NE(response.Answer->StopTimetableCapabilitiesResponse->
StopTimetableServiceCapabilities->      TransportDescription, nullptr);
    ASSERT_NE(response.Answer->StopMonitoringCapabilitiesResponse->
StopMonitoringServiceCapabilities->      TransportDescription, nullptr);
    ASSERT_NE(response.Answer->VehicleMonitoringCapabilitiesResponse->
VehicleMonitoringServiceCapabilities->   TransportDescription, nullptr);
    ASSERT_NE(response.Answer->EstimatedTimetableCapabilitiesResponse->
EstimatedTimetableServiceCapabilities->   TransportDescription, nullptr);
    ASSERT_NE(response.Answer->FacilityMonitoringCapabilitiesResponse->
FacilityMonitoringServiceCapabilities->   TransportDescription, nullptr);
}



void Siri_CAP_MissingParameter_ST::run() {...}

void Siri_CAP_MissingParameter_SM::run() {...}

void Siri_CAP_MissingParameter_VM::run() {...}

void Siri_CAP_MissingParameter_ET::run() {...}

void Siri_CAP_MissingParameter_FM::run() {...}
```

Create_Subscription.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_SUB_Base
{
public:
   Siri_SUB_Base() {}
   virtual ~Siri_SUB_Base() {};
};


class Siri_SUB_PT_Main :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_PT_Main()
      : BaseSIRIClient() {}
   ~Siri_SUB_PT_Main() {};

   virtual void run() override;
};

class Siri_SUB_PT_Error_Unauth :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_PT_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_SUB_PT_Error_Unauth() {};

   virtual void run() override;
};

class Siri_SUB_PT_Error_SubID_Missing :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_PT_Error_SubID_Missing()
      : BaseSIRIClient() {}
   ~Siri_SUB_PT_Error_SubID_Missing() {};

   virtual void run() override;
};
```

```
class Siri_SUB_PT_Error_Empty_Line :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_PT_Error_Empty_Line()
      : BaseSIRIClient() {}
   ~Siri_SUB_PT_Error_Empty_Line() {};

   virtual void run() override;
};


class Siri_SUB_ST_Main :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_ST_Main()
      : BaseSIRIClient() {}
   ~Siri_SUB_ST_Main() {};

   virtual void run() override;
};

class Siri_SUB_ST_Error_Unauth :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_ST_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_SUB_ST_Error_Unauth() {};

   virtual void run() override;
};

class Siri_SUB_ST_Error_SubID_Missing :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_ST_Error_SubID_Missing()
      : BaseSIRIClient() {}
   ~Siri_SUB_ST_Error_SubID_Missing() {};

   virtual void run() override;
};

class Siri_SUB_ST_Error_Long_Period :
   public BaseSIRIClient,
   public Siri_SUB_Base
{
public:
   Siri_SUB_ST_Error_Long_Period()
      : BaseSIRIClient() {}
   ~Siri_SUB_ST_Error_Long_Period() {};

   virtual void run() override;
};
```

```
class Siri_SUB_VM_Error_Unauth :
    public BaseSIRIClient,
    public Siri_SUB_Base
{
public:
    Siri_SUB_VM_Error_Unauth()
        : BaseSIRIClient() {}
    ~Siri_SUB_VM_Error_Unauth() {};

    virtual void run() override;
};

class Siri_SUB_VM_Error_SubID_Missing :
    public BaseSIRIClient,
    public Siri_SUB_Base
{
public:
    Siri_SUB_VM_Error_SubID_Missing()
        : BaseSIRIClient() {}
    ~Siri_SUB_VM_Error_SubID_Missing() {};

    virtual void run() override;
};


class Siri_SUB_ET_Main_Lines :
    public BaseSIRIClient,
    public Siri_SUB_Base
{
public:
    Siri_SUB_ET_Main_Lines()
        : BaseSIRIClient() {}
    ~Siri_SUB_ET_Main_Lines() {};

    virtual void run() override;
};

class Siri_SUB_ET_Error_Unauth :
    public BaseSIRIClient,
    public Siri_SUB_Base
{
public:
    Siri_SUB_ET_Error_Unauth()
        : BaseSIRIClient() {}
    ~Siri_SUB_ET_Error_Unauth() {};

    virtual void run() override;
};

class Siri_SUB_ET_Error_SubID_Missing :
    public BaseSIRIClient,
    public Siri_SUB_Base
{
public:
    Siri_SUB_ET_Error_SubID_Missing()
        : BaseSIRIClient() {}
    ~Siri_SUB_ET_Error_SubID_Missing() {};

    virtual void run() override;
};
```

Create_Subscription.cpp

```
#include "stdafx.h"
#include "Create_Subscription.h"

TEST_F(Siri_ClientTestFixture, SUB_PT_Main)                 {   Siri_SUB_PT_Main
SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ST_Main)                 {   Siri_SUB_ST_Main
SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_SM_Main)                 {   Siri_SUB_SM_Main
SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_VM_Main_Lines)           {
Siri_SUB_VM_Main_Lines            SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ET_Main_Lines)           {
Siri_SUB_ET_Main_Lines            SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_PT_Error_Unauth)           {
Siri_SUB_PT_Error_Unauth          SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ST_Error_Unauth)           {
Siri_SUB_ST_Error_Unauth          SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_SM_Error_Unauth)           {
Siri_SUB_SM_Error_Unauth          SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_VM_Error_Unauth)           {
Siri_SUB_VM_Error_Unauth          SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ET_Error_Unauth)           {
Siri_SUB_ET_Error_Unauth          SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_PT_Error_SubID_Missing)      {
Siri_SUB_PT_Error_SubID_Missing       SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ST_Error_SubID_Missing)      {
Siri_SUB_ST_Error_SubID_Missing       SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_SM_Error_SubID_Missing)      {
Siri_SUB_SM_Error_SubID_Missing       SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_VM_Error_SubID_Missing)      {
Siri_SUB_VM_Error_SubID_Missing       SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ET_Error_SubID_Missing)      {
Siri_SUB_ET_Error_SubID_Missing       SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_PT_Error_Empty_Line)         {
Siri_SUB_PT_Error_Empty_Line      SUB;      SUB.run();   }
TEST_F(Siri_ClientTestFixture, SUB_ST_Error_Long_Period)     {
Siri_SUB_ST_Error_Long_Period      SUB;      SUB.run();   }
```

```
void Siri_SUB_PT_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsSubscriptionRequestStructure request;
   ns1__WsSubscriptionAnswerStructure response;

   ns2__SiriSubscriptionRequestStructure req;
   request.Request = &req;

      ns1__WsSubscriptionRequestInfoStructure* info_structure = new
ns1__WsSubscriptionRequestInfoStructure;
         info_structure->AccountId = &user;
         info_structure->AccountKey = &pwd;
         info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
         info_structure->ConsumerAddress = &consumer_address;

         ns2__ParticipantRefStructure* subscriptor_ref = new
ns2__ParticipantRefStructure;
            std::wstringstream ss;
            ss.imbue(std::locale::classic());
            ss << int_distribution(random_generator);
            subscriptor_ref->__item = L"Subscriber #" + ss.str();
         info_structure->RequestorRef = subscriptor_ref;

         ns2__MessageQualifierStructure* message_ref = new
ns2__MessageQualifierStructure; // equivalent to subscription identifier
            std::wstringstream ss_message;
            ss_message.imbue(std::locale::classic());
            ss_message << int_distribution(random_generator);
            message_ref->__item = L"PT Message #" + ss_message.str();
         info_structure->MessageIdentifier = message_ref;

      request.SubscriptionRequestInfo = info_structure;

      req.__union_SiriSubscriptionRequestStructure =
SOAP_UNION__ns2__union_SiriSubscriptionRequestStructure_ProductionTimetableSubsc
riptionRequest;

      std::vector<_ns2__ProductionTimetableSubscriptionRequest *> *
v_production_timetable_requests= new
std::vector<_ns2__ProductionTimetableSubscriptionRequest *>;
         _ns2__ProductionTimetableSubscriptionRequest*
higher_production_timetable_request = new
_ns2__ProductionTimetableSubscriptionRequest;

            ns2__SubscriptionQualifierStructure* subscription_ref_struct = new
ns2__SubscriptionQualifierStructure;      // Subscription identifier
               boost::uuids::uuid uuid = random_uuid_generator();
               std::wstringstream ss_uuid;
               ss_uuid.imbue(std::locale::classic());
               ss_uuid << uuid;
               subscription_ref_struct->__item = ss_uuid.str();
            higher_production_timetable_request->SubscriptionIdentifier =
subscription_ref_struct;

            higher_production_timetable_request->SubscriberRef =
subscriptor_ref;
```

```
                boost::gregorian::date
today(boost::posix_time::second_clock::universal_time().date());
                boost::posix_time::time_duration
tohour(boost::posix_time::hours(16));
                xsd__dateTime termination =
GSOAP::from_ptime(boost::posix_time::ptime(today, tohour));
             higher_production_timetable_request->InitialTerminationTime =
termination;

             ns2__ProductionTimetableRequestStructure*
production_timetable_request_structure = new
ns2__ProductionTimetableRequestStructure;
                production_timetable_request_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

                ns2__ClosedTimestampRangeStructure* timestamp_structure = new
ns2__ClosedTimestampRangeStructure;
                   xsd__dateTime__ start_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)));
                   xsd__dateTime__ end_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)) + boost::gregorian::days(desired_days));
                   timestamp_structure->StartTime = start_time;
                   timestamp_structure->EndTime = end_time;
                production_timetable_request_structure->ValidityPeriod =
timestamp_structure;

                _ns2__ProductionTimetableRequestStructure_Lines*
request_lines_structure = new _ns2__ProductionTimetableRequestStructure_Lines;
                   std::wstring requested_line_ref1(non_existing_line);
                   std::wstring requested_line_ref2(get_random_line().lineref);

                   std::unique_ptr<ns2__LineDirectionStructure>
p_line_structure(new ns2__LineDirectionStructure);
                      std::unique_ptr<ns2__LineRefStructure>
p_lineref_structure(new ns2__LineRefStructure);
                         p_lineref_structure->__item = requested_line_ref1;
                      auto lineref_1 = std::move(p_lineref_structure);
                         p_line_structure->LineRef = lineref_1.get();
                      auto line_1 = std::move(p_line_structure);

                   request_lines_structure-
>LineDirection.push_back(line_1.get());

                   p_line_structure.reset(new ns2__LineDirectionStructure);
                      p_lineref_structure.reset(new ns2__LineRefStructure);
                         p_lineref_structure->__item = requested_line_ref2;
                      auto lineref_2 = std::move(p_lineref_structure);
                      p_line_structure->LineRef = lineref_2.get();
                      auto line_2 = std::move(p_line_structure);

                   request_lines_structure-
>LineDirection.push_back(line_2.get());

                production_timetable_request_structure->Lines =
request_lines_structure;

         higher_production_timetable_request->ProductionTimetableRequest =
production_timetable_request_structure;
      v_production_timetable_requests-
>push_back(higher_production_timetable_request);
```

```
req.union_SiriSubscriptionRequestStructure.ProductionTimetableSubscriptionReques
t = v_production_timetable_requests;

   set_proxy(ws);
   ASSERT_EQ(ws.Subscribe(server_address.c_str(), nullptr, &request, response),
SOAP_OK) << "## Server might not be running ##";

   delete message_ref;
   v_subscription_references.push_back(std::wstring(subscription_ref_struct-
>__item));
   delete subscription_ref_struct;
   delete subscriptor_ref;
   delete timestamp_structure;
   delete request_lines_structure;
   delete production_timetable_request_structure;
   delete higher_production_timetable_request;
   delete v_production_timetable_requests;
   delete info_structure;


   ASSERT_EQ(response.Answer->ResponseStatus.size(), 1);
   for (const auto& response : response.Answer->ResponseStatus)
   {
      ASSERT_NE(response, nullptr);
      ASSERT_EQ(response->ErrorCondition, nullptr);
      ASSERT_NE(response->ValidUntil, nullptr);
   }
}


void Siri_SUB_PT_Error_Unauth::run()           {...}
void Siri_SUB_PT_Error_SubID_Missing::run()    {...}
void Siri_SUB_PT_Error_Empty_Line::run()       {...}

void Siri_SUB_ST_Main::run()                   {...}
void Siri_SUB_ST_Error_Long_Period::run()      {...}
void Siri_SUB_ST_Error_Unauth::run()           {...}
void Siri_SUB_ST_Error_SubID_Missing::run()    {...}

void Siri_SUB_SM_Main::run()                   {...}
void Siri_SUB_SM_Error_Unauth::run()           {...}
void Siri_SUB_SM_Error_SubID_Missing::run()    {...}

void Siri_SUB_VM_Main_Lines::run()             {...}
void Siri_SUB_VM_Error_Unauth::run()           {...}
void Siri_SUB_VM_Error_SubID_Missing::run()    {...}

void Siri_SUB_ET_Main_Lines::run()             {...}
void Siri_SUB_ET_Error_Unauth::run()           {...}
void Siri_SUB_ET_Error_SubID_Missing::run()    {...}
```

Estimated_Timetable.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_ET_Base
{
public:
   Siri_ET_Base() {}
   virtual ~Siri_ET_Base() {};

protected:
   void show_ET(const ns1__WsEstimatedTimetableAnswerStructure& response);
};


class Siri_ET_Main :
   public BaseSIRIClient,
   public Siri_ET_Base
{
public:
   Siri_ET_Main()
      : BaseSIRIClient() {}
   ~Siri_ET_Main() {};

   virtual void run() override;
};

class Siri_ET_Main_Lines :
   public BaseSIRIClient,
   public Siri_ET_Base
{
public:
   Siri_ET_Main_Lines()
      : BaseSIRIClient() {}
   ~Siri_ET_Main_Lines() {};

   virtual void run() override;
};

class Siri_ET_Unknown_Lines :
°public BaseSIRIClient,
   public Siri_ET_Base
{
public:
   Siri_ET_Unknown_Lines()
      : BaseSIRIClient() {}
   ~Siri_ET_Unknown_Lines() {};

   virtual void run() override;
};
```

```
class Siri_ET_Preview_0:
    public BaseSIRIClient,
    public Siri_ET_Base
{
public:
    Siri_ET_Preview_0()
        : BaseSIRIClient() {}
    ~Siri_ET_Preview_0() {};

    virtual void run() override;
};

class Siri_ET_Error_Unauth :
    public BaseSIRIClient,
    public Siri_ET_Base
{
public:
    Siri_ET_Error_Unauth()
        : BaseSIRIClient() {}
    ~Siri_ET_Error_Unauth() {};

    virtual void run() override;
};
```

Estimated_Timetable.cpp

```
#include "stdafx.h"
#include "Estimated_Timetable.h"

void Siri_ET_Base::show_ET(const ns1__WsEstimatedTimetableAnswerStructure&
response)
{
   _tstring path = _T("c:/temp/siri/estimatedtimetable.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Estimated Timetable: " << std::endl << std::endl;

   int cont_line = 0;
   for (const auto& line : response.Answer->EstimatedTimetableDelivery)
   {
      ofs << "Line number " << ++cont_line;
      ofs << "\t(valid until: " << GSOAP::to_ptime(*line->ValidUntil) << ")" <<
std::endl;
      for (const auto& trajectory : line->EstimatedJourneyVersionFrame)
      {
         ofs << "\nL" << trajectory->EstimatedVehicleJourney.front()->LineRef-
>__item << ", ";
         ofs << "Direction: " << trajectory->EstimatedVehicleJourney.front()-
>DirectionRef->__item << std::endl;

         for (const auto& expedition : trajectory->EstimatedVehicleJourney)
         {
            ofs << "\nExpedition code: " << expedition->VehicleJourneyRef-
>__item << " - ";
            ofs << "from: " << expedition->OriginName.front()->__item << "(" <<
expedition->OriginRef->__item << ")";
            ofs << " to: " << expedition->DestinationName.front()->__item << "("
<< expedition->DestinationRef->__item << ")" << std::endl;

            for (const auto& stop : expedition->EstimatedCalls->EstimatedCall)
            {
               if (expedition->__EstimatedVehicleJourneyStructure_sequence___-
>HeadwayService == true)
               {
                  ofs << stop->StopPointRef->__item << " - frequency: ";
                  ofs << *stop->AimedHeadwayInterval / (1000*60) << "
minutes"<<std::endl;
               }
               else
               {
                  ofs << stop->StopPointRef->__item << " -  from: ";
                  ofs << GSOAP::to_ptime(*stop->AimedArrivalTime) << " to: ";
                  ofs << GSOAP::to_ptime(*stop->AimedDepartureTime) <<
std::endl;
               }
            }
         }
      }
   }

   ofs << "\nResponse time: " << GSOAP::to_ptime(response.ServiceDeliveryInfo-
>ResponseTimestamp) << std::endl;
}
```

```
TEST_F(Siri_ClientTestFixture, ET_Main)              {   Siri_ET_Main
ET;      ET.run();   }
TEST_F(Siri_ClientTestFixture, ET_Main_Lines)        {   Siri_ET_Main_Lines
ET;      ET.run();   }
TEST_F(Siri_ClientTestFixture, ET_Unknown_Lines)     {   Siri_ET_Unknown_Lines
ET;      ET.run();   }
TEST_F(Siri_ClientTestFixture, ET_Error_Unauth)        {   Siri_ET_Error_Unauth
ET;      ET.run();   }
TEST_F(Siri_ClientTestFixture, ET_Preview_0)         {   Siri_ET_Preview_0
ET;      ET.run();   }


void Siri_ET_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsEstimatedTimetableRequestStructure request;
   ns1__WsEstimatedTimetableAnswerStructure response;

   ns2__EstimatedTimetableRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

      boost::chrono::hours interval(desired_hours);
      boost::chrono::milliseconds ms =
boost::chrono::duration_cast<boost::chrono::milliseconds> (interval);
      LONG64* previewinterval = new LONG64(ms.count());
   req.PreviewInterval = previewinterval;

   //set_proxy(ws);
   ASSERT_EQ(ws.GetEstimatedTimetable(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete context_structure->VelocityUnits;
   delete context_structure;
   delete previewinterval;
   delete info_structure;
```

```
   ASSERT_FALSE(response.Answer->EstimatedTimetableDelivery.empty());
   for (const auto& line : response.Answer->EstimatedTimetableDelivery)
   {
      ASSERT_NE(line, nullptr);
      ASSERT_EQ(line->ErrorCondition, nullptr);
      ASSERT_FALSE(line->EstimatedJourneyVersionFrame.empty());
      for (const auto& trayecto : line->EstimatedJourneyVersionFrame)      // No
timetable services, only frequency ones (TO-CHECK: ¿filtrar los de frecuencia en
ese caso?)
      {
         ASSERT_LE(GSOAP::to_ptime(trayecto->RecordedAtTime),
boost::posix_time::second_clock::universal_time());
         for (const auto& expedition : trayecto->EstimatedVehicleJourney)
         {
            ASSERT_NE(expedition, nullptr);
            ASSERT_NE(expedition->LineRef, nullptr);
            ASSERT_FALSE(expedition->LineRef->__item.empty());
            ASSERT_NE(expedition->DirectionRef, nullptr);
            ASSERT_FALSE(expedition->DirectionRef->__item.empty());
            ASSERT_NE(expedition->VehicleJourneyRef, nullptr);
            ASSERT_FALSE(expedition->VehicleJourneyRef->__item.empty());
            ASSERT_NE(expedition->DestinationRef, nullptr);
            ASSERT_FALSE(expedition->DestinationRef->__item.empty());
            ASSERT_NE(expedition->OriginRef, nullptr);
            ASSERT_FALSE(expedition->OriginRef->__item.empty());

            ASSERT_FALSE(expedition->DestinationName.empty());
            for (const auto& name : expedition->DestinationName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }

            ASSERT_FALSE(expedition->OriginName.empty());
            for (const auto& name : expedition->OriginName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }

            ASSERT_NE(expedition->__EstimatedVehicleJourneyStructure_sequence,
nullptr);
            ASSERT_FALSE(expedition-
>__EstimatedVehicleJourneyStructure_sequence->DirectionName.empty());
            for (const auto& name : expedition-
>__EstimatedVehicleJourneyStructure_sequence->DirectionName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
            ASSERT_FALSE(expedition-
>__EstimatedVehicleJourneyStructure_sequence->PublishedLineName.empty());
            for (const auto& name : expedition-
>__EstimatedVehicleJourneyStructure_sequence->PublishedLineName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
```

```
            ASSERT_NE(expedition->__EstimatedVehicleJourneyStructure_sequence-
>JourneyPatternRef, nullptr);
            ASSERT_FALSE(expedition-
>__EstimatedVehicleJourneyStructure_sequence->JourneyPatternRef-
>__item.empty());
            ASSERT_NE(expedition->__EstimatedVehicleJourneyStructure_sequence-
>RouteRef, nullptr);
            ASSERT_FALSE(expedition-
>__EstimatedVehicleJourneyStructure_sequence->RouteRef->__item.empty());

            ASSERT_NE(expedition->__EstimatedVehicleJourneyStructure_sequence__,
nullptr);
            ASSERT_FALSE(expedition-
>__EstimatedVehicleJourneyStructure_sequence__->VehicleJourneyName.empty());
            for (const auto& name : expedition-
>__EstimatedVehicleJourneyStructure_sequence__->VehicleJourneyName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }

            ASSERT_NE(expedition-
>__EstimatedVehicleJourneyStructure_sequence___, nullptr);

            ASSERT_NE(expedition->EstimatedCalls, nullptr);
            ASSERT_FALSE(expedition->EstimatedCalls->EstimatedCall.empty());
            for (const auto& stop : expedition->EstimatedCalls->EstimatedCall)
            {
               ASSERT_NE(stop, nullptr);
               ASSERT_NE(stop->StopPointRef, nullptr);
               ASSERT_FALSE(stop->StopPointRef->__item.empty());

               if (expedition->__EstimatedVehicleJourneyStructure_sequence___-
>HeadwayService == true)
               {
                  ASSERT_EQ(stop->AimedArrivalTime, nullptr);
                  ASSERT_EQ(stop->AimedDepartureTime, nullptr);
                  ASSERT_NE(stop->AimedHeadwayInterval, nullptr);
                  ASSERT_GT(*stop->AimedHeadwayInterval, 0);
               }
               else
               {
                  ASSERT_NE(stop->AimedArrivalTime, nullptr);
                  ASSERT_NE(stop->AimedDepartureTime, nullptr);
                  ASSERT_EQ(stop->AimedHeadwayInterval, nullptr);
               }
            }   // stops

            //ASSERT_NE(expedition->RecordedCalls, nullptr);
            //ASSERT_FALSE(expedition->RecordedCalls->RecordedCall.empty());
            if (expedition->RecordedCalls == nullptr)                    // TO-
CHECK
               continue;
            for (const auto& stop : expedition->RecordedCalls->RecordedCall)
            {
               ASSERT_NE(stop, nullptr);
               ASSERT_NE(stop->StopPointRef, nullptr);
               ASSERT_FALSE(stop->StopPointRef->__item.empty());
```

```
                if (expedition->__EstimatedVehicleJourneyStructure_sequence___-
>HeadwayService == true)
                {
                   ASSERT_EQ(stop->AimedArrivalTime, nullptr);
                   ASSERT_EQ(stop->AimedDepartureTime, nullptr);
                   ASSERT_EQ(stop->ActualArrivalTime, nullptr);
                   ASSERT_EQ(stop->ActualDepartureTime, nullptr);
                   ASSERT_NE(stop->AimedHeadwayInterval, nullptr);
                   ASSERT_GT(*stop->AimedHeadwayInterval, 0);
                   ASSERT_NE(stop->ActualHeadwayInterval, nullptr);
                   ASSERT_GT(*stop->ActualHeadwayInterval, 0);
                }
                else
                {
                   ASSERT_NE(stop->AimedArrivalTime, nullptr);
                   ASSERT_NE(stop->ActualArrivalTime, nullptr);
                   ASSERT_NE(stop->AimedDepartureTime, nullptr);
                   ASSERT_NE(stop->ActualDepartureTime, nullptr);
                   ASSERT_EQ(stop->AimedHeadwayInterval, nullptr);
                   ASSERT_EQ(stop->ActualHeadwayInterval, nullptr);
                }
             }   // stops
          }   // expeditions
       }   // trayectos
    }   // lineas

    ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp));
}


void Siri_ET_Main_Lines::run()    {...}
void Siri_ET_Unknown_Lines::run() {...}
void Siri_ET_Error_Unauth::run()  {...}
void Siri_ET_Preview_0::run()     {...}
```

Facility_Monitoring.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_FM_Base
{
public:
   Siri_FM_Base() {}
   virtual ~Siri_FM_Base() {};

protected:
   void show_FM(const ns1__WsFacilityMonitoringAnswerStructure& response, const
_tstring mode);
};


class Siri_FM_All_Lines :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_All_Lines()
      : BaseSIRIClient() {}
   ~Siri_FM_All_Lines() {};

   virtual void run() override;
};

class Siri_FM_All_Stops :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_All_Stops()
      : BaseSIRIClient() {}
   ~Siri_FM_All_Stops() {};

   virtual void run() override;
};

class Siri_FM_Error_Unauth :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_FM_Error_Unauth() {};

   virtual void run() override;
};
```

```
class Siri_FM_Error_LineRef_StopPointRef_Missing :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_Error_LineRef_StopPointRef_Missing()
      : BaseSIRIClient() {}
   ~Siri_FM_Error_LineRef_StopPointRef_Missing() {};

   virtual void run() override;
};

class Siri_FM_Wrong_ALL_LINES_CODE :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_Wrong_ALL_LINES_CODE()
      : BaseSIRIClient() {}
   ~Siri_FM_Wrong_ALL_LINES_CODE() {};

   virtual void run() override;
};

class Siri_FM_Wrong_ALL_STOPS_CODE :
   public BaseSIRIClient,
   public Siri_FM_Base
{
public:
   Siri_FM_Wrong_ALL_STOPS_CODE()
      : BaseSIRIClient() {}
   ~Siri_FM_Wrong_ALL_STOPS_CODE() {};

   virtual void run() override;
};
```

Facility_Monitoring.cpp

```
#include "stdafx.h"
#include "Facility_Monitoring.h"

void Siri_FM_Base::show_FM(const ns1__WsFacilityMonitoringAnswerStructure&
response, const _tstring mode)
{
   _tstring path = _T("c:/temp/siri/facilitymonitoring_") + mode +
_tstring(_T(".txt"));
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Facility Monitoring: " << static_cast<const
wchar_t*>(saer::ct2cw(mode)) << std::endl << std::endl;

   for (const auto& delivery : response.Answer->FacilityMonitoringDelivery)
   {
      ofs << "\t(valid until: " << GSOAP::to_ptime(*delivery->ValidUntil) << ")"
<< std::endl;
      ofs << "From: " << GSOAP::to_ptime(delivery->FacilityCondition.front()-
>ValidityPeriod->StartTime);
      ofs << " to: " << GSOAP::to_ptime(*delivery->FacilityCondition.front()-
>ValidityPeriod->EndTime) << std::endl << std::endl;

      auto last_type = *delivery->FacilityCondition.front()->MonitoringInfo-
>MonitoringType;
      auto last_status = delivery->FacilityCondition.front()->FacilityStatus-
>Status;
      unsigned n_facilities = 0;
      for (const auto& facility_condition : delivery->FacilityCondition)
      {
         ofs << ++n_facilities << "\t" << facility_condition-
>union_FacilityConditionStructure.Facility->Description.front()->__item;
         if (*facility_condition->MonitoringInfo->MonitoringType != last_type ||
n_facilities == 1)
            ofs << "\t\t\t(Type: " << *facility_condition->MonitoringInfo-
>MonitoringType;
         if (facility_condition->FacilityStatus->Status != last_status ||
n_facilities == 1)
            ofs << ",\tStatus: " << facility_condition->FacilityStatus->Status
<< ")";

         ofs << std::endl;

         last_type = *facility_condition->MonitoringInfo->MonitoringType;
         last_status = delivery->FacilityCondition.front()->FacilityStatus-
>Status;
      }
   }

   ofs << "\nResponse time: " << GSOAP::to_ptime(response.ServiceDeliveryInfo-
>ResponseTimestamp) << std::endl;
}
```

```
TEST_F(Siri_ClientTestFixture, FM_All_Lines)                    {
Siri_FM_All_Lines                       FM;      FM.run();   }
TEST_F(Siri_ClientTestFixture, FM_All_Stops)                    {
Siri_FM_All_Stops                       FM;      FM.run();   }
TEST_F(Siri_ClientTestFixture, FM_Error_Unauth)                    {
Siri_FM_Error_Unauth                    FM;      FM.run();   }
TEST_F(Siri_ClientTestFixture, FM_Error_LineRef_StopPointRef_Missing)   {
Siri_FM_Error_LineRef_StopPointRef_Missing   FM;      FM.run();   }
TEST_F(Siri_ClientTestFixture, FM_Wrong_ALL_LINES_CODE)            {
Siri_FM_Wrong_ALL_LINES_CODE            FM;      FM.run();   }
TEST_F(Siri_ClientTestFixture, FM_Wrong_ALL_STOPS_CODE)            {
Siri_FM_Wrong_ALL_STOPS_CODE            FM;      FM.run();   }


void Siri_FM_All_Lines::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsFacilityMonitoringRequestStructure request;
   ns1__WsFacilityMonitoringAnswerStructure response;

   ns2__FacilityMonitoringRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &auth_user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

   xsd__dateTime__* starttime_struct = new
xsd__dateTime(GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour))));
   req.StartTime = starttime_struct;

      boost::chrono::hours interval(desired_hours);
      boost::chrono::milliseconds ms =
boost::chrono::duration_cast<boost::chrono::milliseconds> (interval);
      LONG64* previewinterval = new LONG64(ms.count());
   req.PreviewInterval = previewinterval;

   __ns2__FacilityMonitoringRequestStructure_sequence* sequence_struct = new
__ns2__FacilityMonitoringRequestStructure_sequence;
      ns2__LineRefStructure* lineref_struct = new ns2__LineRefStructure;
         lineref_struct->__item = ALL_LINES_CODE;
      sequence_struct->LineRef = lineref_struct;
   req.__FacilityMonitoringRequestStructure_sequence = sequence_struct;

   set_proxy(ws);
   ASSERT_EQ(ws.GetFacilityMonitoring(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";
```

```
    delete starttime_struct;
    delete context_structure->VelocityUnits;
    delete context_structure;
    delete info_structure;
    delete previewinterval;

    // ...
    delete lineref_struct;
    delete sequence_struct;


    ASSERT_FALSE(response.Answer->FacilityMonitoringDelivery.empty());
    for (const auto& delivery : response.Answer->FacilityMonitoringDelivery)
    {
       ASSERT_NE(delivery, nullptr);
       ASSERT_EQ(delivery->ErrorCondition, nullptr);

       ASSERT_FALSE(delivery->FacilityCondition.empty());
       for (const auto& facility_condition : delivery->FacilityCondition)
       {
          ASSERT_NE(facility_condition, nullptr);
          ASSERT_NE(facility_condition->FacilityStatus, nullptr);
          ASSERT_NE(facility_condition->ValidityPeriod, nullptr);
          ASSERT_FALSE(GSOAP::to_ptime(facility_condition->ValidityPeriod-
>StartTime).is_not_a_date_time());
          ASSERT_FALSE(GSOAP::to_ptime(*facility_condition->ValidityPeriod-
>EndTime).is_not_a_date_time());
          if (facility_condition->__union_FacilityConditionStructure ==
SOAP_UNION__ns2__union_FacilityConditionStructure_Facility)
          {
             ASSERT_NE(facility_condition-
>union_FacilityConditionStructure.Facility, nullptr);
             ASSERT_NE(facility_condition-
>union_FacilityConditionStructure.Facility->FacilityCode, nullptr);
             ASSERT_FALSE(facility_condition-
>union_FacilityConditionStructure.Facility->FacilityCode->empty());
             ASSERT_FALSE(facility_condition-
>union_FacilityConditionStructure.Facility->FacilityClass.empty());
             for (const auto& clase : facility_condition-
>union_FacilityConditionStructure.Facility->FacilityClass)
             {
                ASSERT_TRUE(clase ==
ns2__FacilityCategoryEnumeration__fixedEquipment);
             }
          }
          else if (facility_condition->__union_FacilityConditionStructure ==
SOAP_UNION__ns2__union_FacilityConditionStructure_FacilityRef)
          {
             ASSERT_NE(facility_condition-
>union_FacilityConditionStructure.FacilityRef, nullptr);
             ASSERT_FALSE(facility_condition-
>union_FacilityConditionStructure.FacilityRef->__item.empty());
          }
          else
             ASSERT_FALSE(true);
       }
    }
```

```
   ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp));

   if (siri_debug)
   {
      _tstring mode(_T("lines"));
      show_FM(response, mode);
   }
}


void Siri_FM_All_Stops::run()                                    {...}

void Siri_FM_Error_Unauth::run()                                 {...}
void Siri_FM_Error_LineRef_StopPointRef_Missing::run()           {...}
void Siri_FM_Wrong_ALL_LINES_CODE::run()                         {...}
void Siri_FM_Wrong_ALL_STOPS_CODE::run()                         {...}
```

Lines_Discovery.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_LD_Base
{
public:
   Siri_LD_Base() {}
   virtual ~Siri_LD_Base() {};

protected:
   void show_LD(const ns1__WsLinesDiscoveryAnswerStructure& response);
};


class Siri_LD_Main :
   public BaseSIRIClient,
   public Siri_LD_Base
{
public:
   Siri_LD_Main()
      : BaseSIRIClient() {}
   ~Siri_LD_Main() {};

   virtual void run() override;
};

class Siri_LD_Error_Unauth :
   public BaseSIRIClient,
   public Siri_LD_Base
{
public:
   Siri_LD_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_LD_Error_Unauth() {};

   virtual void run() override;
};

class Siri_LD_Error_Unauth_Missing_User :
   public BaseSIRIClient,
   public Siri_LD_Base
{
public:
   Siri_LD_Error_Unauth_Missing_User()
      : BaseSIRIClient() {}
   ~Siri_LD_Error_Unauth_Missing_User() {};

   virtual void run() override;
};
```

Lines_Discovery.cpp

```
#include "stdafx.h"
#include "Lines_Discovery.h"

void Siri_LD_Base::show_LD(const ns1__WsLinesDiscoveryAnswerStructure& response)
{
   _tstring path = _T("c:/temp/siri/lines_discovery.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Lines Discovery: " << std::endl << std::endl;

   ofs << "Valid until: " << GSOAP::to_ptime(*response.Answer->ValidUntil) <<
std::endl;
   int iter = 0;
   for (const auto& line : response.Answer->AnnotatedLineRef)
   {
      ofs << ++iter << '\t';
      ofs << line->LineName.front()->__item << " ";
      ofs << line->LineRef->__item << " ";
      ofs << std::boolalpha << line->Monitored << std::endl;
      for (const auto& destination : line->Destinations->Destination)
      {
         ofs << "\t\t";
         ofs << destination->DestinationRef->__item << " " << destination-
>PlaceName.front()->__item << std::endl;
      }
   }
   ofs << '\n' << GSOAP::to_ptime(response.Answer->ResponseTimestamp) <<
std::endl;
}

TEST_F(Siri_ClientTestFixture, LD_Main   )                    {   Siri_LD_Main
LD;      LD.run();   }
TEST_F(Siri_ClientTestFixture, LD_Error_Unauth)                  {
Siri_LD_Error_Unauth            LD;      LD.run();   }
TEST_F(Siri_ClientTestFixture, LD_Error_Unauth_Missing_User)      {
Siri_LD_Error_Unauth_Missing_User   LD;      LD.run();   }

void Siri_LD_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsLinesDiscoveryStructure request;
   ns1__WsLinesDiscoveryAnswerStructure response;

   ns2__LinesDiscoveryRequestStructure req;
   request.Request = &req;
   req.Language = desired_language;   // en by default
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.LinesDetailLevel = ns2__LinesDetailEnumeration__normal;

   req.AccountId = &user;
   req.AccountKey = &pwd;
```

```
   set_proxy(ws);
   ASSERT_EQ(ws.LinesDiscovery(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   ASSERT_EQ   (response.Answer->ErrorCondition, nullptr);
   ASSERT_FALSE(response.Answer->AnnotatedLineRef.empty());
   for (const auto& line : response.Answer->AnnotatedLineRef)
   {
      ASSERT_NE   (line, nullptr);
      ASSERT_NE   (line->LineRef, nullptr);
      ASSERT_FALSE(line->LineRef->__item.empty());
      ASSERT_NE   (line->Destinations, nullptr);
      ASSERT_FALSE(line->Destinations->Destination.empty());
      for (const auto& destination : line->Destinations->Destination)
      {
         ASSERT_NE   (destination->DestinationRef, nullptr);
         ASSERT_FALSE(destination->DestinationRef->__item.empty());
         ASSERT_FALSE(destination->PlaceName.empty());
         ASSERT_FALSE(destination->PlaceName.front()->__item.empty());
      }

      ASSERT_FALSE(line->LineName.empty());
      for (const auto& name : line->LineName)
      {
         ASSERT_NE   (name, nullptr);
         ASSERT_FALSE(name->__item.empty());
      }
      ASSERT_NE   (response.Answer->ValidUntil, nullptr);
      ASSERT_GE   (boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.Answer->ResponseTimestamp));
   }

   if(siri_debug)
      show_LD(response);
}

void Siri_LD_Error_Unauth::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsLinesDiscoveryStructure request;
   ns1__WsLinesDiscoveryAnswerStructure response;

   ns2__LinesDiscoveryRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.LinesDetailLevel = ns2__LinesDetailEnumeration__normal;

   //req.AccountId = &unauth_user;
   req.AccountKey = &pwd;

   //set_proxy(ws);
   ASSERT_EQ(ws.LinesDiscovery(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   ASSERT_NE(response.Answer->ErrorCondition, nullptr);
   ASSERT_NE(response.Answer->ErrorCondition->Description, nullptr);
   ASSERT_EQ(response.Answer->ErrorCondition->Description->__item,
unauth_error_text);
```

```
    ASSERT_EQ(response.Answer->ErrorCondition-
>__union_ServiceDeliveryErrorConditionStructure__,
SOAP_UNION__ns2__union_ServiceDeliveryErrorConditionStructure___EndpointDeniedAc
cessError);
    ASSERT_EQ(*response.Answer->ErrorCondition-
>union_ServiceDeliveryErrorConditionStructure__.EndpointDeniedAccessError-
>ErrorText, unauth_error_text);

    ASSERT_TRUE(response.Answer->AnnotatedLineRef.empty());
}

void Siri_LD_Error_Unauth_Missing_User::run()
{
    SiriProducerDocBindingProxy ws;

    ns1__WsLinesDiscoveryStructure request;
    ns1__WsLinesDiscoveryAnswerStructure response;

    ns2__LinesDiscoveryRequestStructure req;
    request.Request = &req;
    req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
    req.LinesDetailLevel = ns2__LinesDetailEnumeration__normal;

    req.AccountId = &unauth_user;
    req.AccountKey = &pwd;

    //set_proxy(ws);
    ASSERT_EQ(ws.LinesDiscovery(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

    ASSERT_NE(response.Answer->ErrorCondition, nullptr);
    ASSERT_NE(response.Answer->ErrorCondition->Description, nullptr);
    ASSERT_EQ(response.Answer->ErrorCondition->Description->__item,
unauth_error_text);
    ASSERT_EQ(response.Answer->ErrorCondition-
>__union_ServiceDeliveryErrorConditionStructure__,
SOAP_UNION__ns2__union_ServiceDeliveryErrorConditionStructure___EndpointDeniedAc
cessError);
    ASSERT_EQ(*response.Answer->ErrorCondition-
>union_ServiceDeliveryErrorConditionStructure__.EndpointDeniedAccessError-
>ErrorText, unauth_error_text);

    ASSERT_TRUE(response.Answer->AnnotatedLineRef.empty());
}
```

Production_Timetable.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_PT_Base
{
public:
   Siri_PT_Base() {}
   virtual ~Siri_PT_Base() {};

protected:
   void show_PT(const ns1__ProductionTimetableAnswerStructure& response);
};


class Siri_PT_Main :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Main()
      : BaseSIRIClient() {}
   ~Siri_PT_Main() {};

   virtual void run() override;
};

class Siri_PT_Main_Lines :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Main_Lines()
      : BaseSIRIClient() {}
   ~Siri_PT_Main_Lines() {};

   virtual void run() override;
};

class Siri_PT_Unknown_Line :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Unknown_Line()
      : BaseSIRIClient() {}
   ~Siri_PT_Unknown_Line() {};

   virtual void run() override;
};
```

```
class Siri_PT_Error_Unauth :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_PT_Error_Unauth() {};

   virtual void run() override;
};

class Siri_PT_Error_Period :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Error_Period()
      : BaseSIRIClient() {}
   ~Siri_PT_Error_Period() {};

   virtual void run() override;
};

class Siri_PT_Error_ValidityPeriod_Missing :
   public BaseSIRIClient,
   public Siri_PT_Base
{
public:
   Siri_PT_Error_ValidityPeriod_Missing()
      : BaseSIRIClient() {}
   ~Siri_PT_Error_ValidityPeriod_Missing() {};

   virtual void run() override;
};
```

Production_Timetable.cpp

```
#include "stdafx.h"
#include "Production_Timetable.h"

void Siri_PT_Base::show_PT(const ns1__ProductionTimetableAnswerStructure&
response)
{
   _tstring path = _T("c:/temp/siri/productiontimetable.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Production Timetable: " << std::endl << std::endl;

   int cont_line = 0;
   for (const auto& line : response.Answer->ProductionTimetableDelivery)
   {
      ofs << "Line number " << ++cont_line;
      ofs << "\t(valid until: " << GSOAP::to_ptime(*line->ValidUntil) << ")" <<
std::endl;
      for (const auto& trajectory : line->DatedTimetableVersionFrame)
      {
         ofs << "\nL" << trajectory->LineRef->__item << ", ";
         ofs << "Trajectory: " << trajectory->RouteRef->__item << ", ";
         ofs << "Direction: " << trajectory->DirectionRef->__item << std::endl;

         for (const auto& expedition : trajectory->DatedVehicleJourney)
         {
            ofs << "\nExpedition code: " << *expedition->DatedVehicleJourneyCode
<< " - ";
            ofs << "from: " << expedition->OriginDisplay.front()->__item;
            ofs << " to: " << expedition->DestinationDisplay.front()->__item <<
std::endl;

            for (const auto& stop : expedition->DatedCalls.DatedCall)
            {
               if (expedition->HeadwayService && *expedition->HeadwayService ==
true)
               {
                  ofs << stop->StopPointRef->__item << " - frequency: ";
                  ofs << *stop->AimedHeadwayInterval / (1000*60) << "
minutes"<<std::endl;
               }
               else
               {
                  ofs << stop->StopPointRef->__item << " -  from: ";
                  ofs << GSOAP::to_ptime(*stop->AimedArrivalTime) << " to: ";
                  ofs << GSOAP::to_ptime(*stop->AimedDepartureTime) <<
std::endl;
               }
            }
         }
      }
   }

   ofs << "\nResponse time: " << GSOAP::to_ptime(response.ServiceDeliveryInfo-
>ResponseTimestamp) << std::endl;
}
```

```
TEST_F(Siri_ClientTestFixture, PT_Main)                    {   Siri_PT_Main
PT;      PT.run();   }
TEST_F(Siri_ClientTestFixture, PT_Main_Lines)              {
Siri_PT_Main_Lines                 PT;      PT.run();   }
TEST_F(Siri_ClientTestFixture, PT_Unknown_Line)              {
Siri_PT_Unknown_Line               PT;      PT.run();   }
TEST_F(Siri_ClientTestFixture, PT_Error_Unauth)              {
Siri_PT_Error_Unauth               PT;      PT.run();   }
TEST_F(Siri_ClientTestFixture, PT_Error_Period)              {
Siri_PT_Error_Period               PT;      PT.run();   }
TEST_F(Siri_ClientTestFixture, PT_Error_ValidityPeriod_Missing)   {
Siri_PT_Error_ValidityPeriod_Missing   PT;      PT.run();   }



void Siri_PT_Main_Lines::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__ProductionTimetableRequestStructure request;
   ns1__ProductionTimetableAnswerStructure response;

   ns2__ProductionTimetableRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId= &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

   ns2__ClosedTimestampRangeStructure* timestamp_structure = new
ns2__ClosedTimestampRangeStructure;
      xsd__dateTime__ start_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)));
      xsd__dateTime__ end_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)) + boost::gregorian::days(desired_days));
      timestamp_structure->StartTime = start_time;
      timestamp_structure->EndTime = end_time;
   req.ValidityPeriod = timestamp_structure;

   _ns2__ProductionTimetableRequestStructure_Lines* request_lines_structure =
new _ns2__ProductionTimetableRequestStructure_Lines;
      std::wstring requested_line_ref1(non_existing_line);
      std::wstring requested_line_ref2(get_random_line().lineref);
```

```
      std::unique_ptr<ns2__LineDirectionStructure> p_line_structure(new
ns2__LineDirectionStructure);
         std::unique_ptr<ns2__LineRefStructure> p_lineref_structure(new
ns2__LineRefStructure);
            p_lineref_structure->__item = requested_line_ref1;
         auto lineref_1 = std::move(p_lineref_structure);
            p_line_structure->LineRef = lineref_1.get();
         auto line_1 = std::move(p_line_structure);

         request_lines_structure->LineDirection.push_back(line_1.get());

      p_line_structure.reset(new ns2__LineDirectionStructure);
         p_lineref_structure.reset(new ns2__LineRefStructure);
            p_lineref_structure->__item = requested_line_ref2;
         auto lineref_2 = std::move(p_lineref_structure);
         p_line_structure->LineRef = lineref_2.get();
         auto line_2 = std::move(p_line_structure);

         request_lines_structure->LineDirection.push_back(line_2.get());
      req.Lines = request_lines_structure;

   const boost::posix_time::ptime ptime_start_time =
GSOAP::to_ptime(req.ValidityPeriod->StartTime);
   const boost::posix_time::ptime ptime_end_time =
GSOAP::to_ptime(req.ValidityPeriod->EndTime);

   set_proxy(ws);
   ASSERT_EQ(ws.GetProductionTimetable(server_address.c_str(), nullptr,
&request, response), SOAP_OK) << "## Server might not be running ##";

   delete timestamp_structure;
   delete context_structure->VelocityUnits;
   delete context_structure;
   delete info_structure;
   delete request_lines_structure;
   // ...

   ASSERT_FALSE(response.Answer->ProductionTimetableDelivery.empty());
   for (const auto& line : response.Answer->ProductionTimetableDelivery)
   {
      ASSERT_NE   (line, nullptr);
      ASSERT_EQ   (ptime_end_time, GSOAP::to_ptime(*line->ValidUntil));
      ASSERT_EQ   (line->ErrorCondition, nullptr);
      ASSERT_FALSE(line->DatedTimetableVersionFrame.empty());
      for (const auto& trajectory : line->DatedTimetableVersionFrame)
      {
         ASSERT_NE   (trajectory, nullptr);
         ASSERT_NE   (trajectory->LineRef, nullptr);
         ASSERT_FALSE(trajectory->LineRef->__item.empty());
         ASSERT_EQ   (trajectory->LineRef->__item, requested_line_ref2);
         ASSERT_NE   (trajectory->DirectionRef, nullptr);
         ASSERT_FALSE(trajectory->DirectionRef->__item.empty());
         ASSERT_FALSE(trajectory->DatedVehicleJourney.empty());
         for (const auto& expedition : trajectory->DatedVehicleJourney)
         {
            ASSERT_NE   (expedition, nullptr);
            ASSERT_FALSE(expedition->DatedVehicleJourneyCode->empty());
            ASSERT_FALSE(expedition->DestinationDisplay.empty());
            for (const auto& name : expedition->DestinationDisplay)
            {
               ASSERT_NE   (name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
```

```
            ASSERT_FALSE(expedition->OriginDisplay.empty());
            for (const auto& name : expedition->OriginDisplay)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
            ASSERT_FALSE(expedition->DatedCalls.DatedCall.empty());
            for (const auto& stop : expedition->DatedCalls.DatedCall)
            {
               ASSERT_NE   (stop, nullptr);
               ASSERT_NE   (stop->StopPointRef, nullptr);
               ASSERT_FALSE(stop->StopPointRef->__item.empty());

               if (expedition->HeadwayService && *expedition->HeadwayService ==
true)
               {
                  ASSERT_EQ(stop->AimedArrivalTime, nullptr);
                  ASSERT_EQ(stop->AimedDepartureTime, nullptr);
                  ASSERT_NE(stop->AimedHeadwayInterval, nullptr);
                  ASSERT_GT(*stop->AimedHeadwayInterval, 0);
               }
               else
               {
                  ASSERT_NE(stop->AimedArrivalTime, nullptr);
                  ASSERT_NE(stop->AimedDepartureTime, nullptr);
                  ASSERT_EQ(stop->AimedHeadwayInterval, nullptr);
                  //ASSERT_LE(GSOAP::to_ptime(*stop->AimedDepartureTime),
ptime_end_time);
                  //ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*stop-
>AimedArrivalTime));
               }
            }
         }
      }
   }
   ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp));

   if (siri_debug)
      show_PT(response);
}



void Siri_PT_Main::run()                                {...}
void Siri_PT_Unknown_Line::run()                        {...}
void Siri_PT_Error_Unauth::run()                        {...}
void Siri_PT_Error_Period::run()                        {...}
void Siri_PT_Error_ValidityPeriod_Missing::run()        {...}
```

Stop_Monitoring.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_SM_Base
{
public:
   Siri_SM_Base() {}
   virtual ~Siri_SM_Base() {};

protected:
   void show_SM(const ns1__StopMonitoringAnswerStructure& response);
};


class Siri_SM_Main :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Main()
      : BaseSIRIClient() {}
   ~Siri_SM_Main() {};

   virtual void run() override;
};

class Siri_SM_Start_Time_Missing :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Start_Time_Missing()
      : BaseSIRIClient() {}
   ~Siri_SM_Start_Time_Missing() {};

   virtual void run() override;
};

class Siri_SM_Unknown_Stop :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Unknown_Stop()
      : BaseSIRIClient() {}
   ~Siri_SM_Unknown_Stop() {};

   virtual void run() override;
};
```

```
class Siri_SM_Error_Unauth :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_SM_Error_Unauth() {};

   virtual void run() override;
};

class Siri_SM_Error_Context_Missing :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Error_Context_Missing()
      : BaseSIRIClient() {}
   ~Siri_SM_Error_Context_Missing() {};

   virtual void run() override;
};

class Siri_SM_Error_Period :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Error_Period()
      : BaseSIRIClient() {}
   ~Siri_SM_Error_Period() {};

   virtual void run() override;
};

class Siri_SM_Error_Parameter_Missing :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Error_Parameter_Missing()
      : BaseSIRIClient() {}
   ~Siri_SM_Error_Parameter_Missing() {};

   virtual void run() override;
};

class Siri_SM_Error_Parameter_null :
   public BaseSIRIClient,
   public Siri_SM_Base
{
public:
   Siri_SM_Error_Parameter_null()
      : BaseSIRIClient() {}
   ~Siri_SM_Error_Parameter_null() {};

   virtual void run() override;
};
```

Stop_Monitoring.cpp

```
#include "stdafx.h"
#include "Stop_Monitoring.h"

void Siri_SM_Base::show_SM(const ns1__StopMonitoringAnswerStructure& response)
{
   _tstring path = _T("c:/temp/siri/stopmonitoring.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Stop Monitoring: " << std::endl << std::endl;

   unsigned cont = 0;
   for (const auto& delivery : response.Answer->StopMonitoringDelivery)
   {
      ofs << "Delivery #" << ++cont << std::endl;
      ofs << "Valid until: " << GSOAP::to_ptime(*delivery->ValidUntil) <<
std::endl;
      for (const auto& stopvisit : delivery->MonitoredStopVisit)
      {
         ofs << std::endl;
         ofs << "MonitoringRef: "      << stopvisit-
>__MonitoredStopVisitStructure_sequence->MonitoringRef->__item   << ", ";
         ofs << "LineRef: "            << stopvisit->MonitoredVehicleJourney-
>LineRef->__item                 << ", ";
         ofs << "DirectionRef: "         << stopvisit->MonitoredVehicleJourney-
>DirectionRef->__item              << ", ";
         ofs << "VehicleJourneyName: "   << stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName.front()->__item   << std::endl;

         ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall->Order
<< ". ";
         ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointName.front()->__item         << " - ";
         if (stopvisit->MonitoredVehicleJourney-
>__MonitoredVehicleJourneyStructure_sequence__ && stopvisit-
>MonitoredVehicleJourney->__MonitoredVehicleJourneyStructure_sequence__-
>HeadwayService==true)
         {
            ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " - frequency: ";
            ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval / (1000 * 60) << " minutes" << std::endl;
         }
         else
         {
            ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " -  from: ";
            ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedArrivalTime) << " to: ";
            ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedDepartureTime) << std::endl;
         }

         boost::posix_time::time_duration delay=
boost::posix_time::milliseconds(*stopvisit->MonitoredVehicleJourney->Delay);
         ofs << "delay: " << delay << std::endl;
      }
```

```
      for (const auto& stopcancellation : delivery-
>MonitoredStopVisitCancellation)
      {

      }
      ofs << "\nResponse time: " << GSOAP::to_ptime(delivery-
>ResponseTimestamp);
   }
}

TEST_F(Siri_ClientTestFixture, SM_Main)                    {   Siri_SM_Main
SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Start_Time_Missing)        {
Siri_SM_Start_Time_Missing      SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Unknown_Stop)              {
Siri_SM_Unknown_Stop         SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Error_Unauth)              {
Siri_SM_Error_Unauth         SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Error_Context_Missing)     {
Siri_SM_Error_Context_Missing   SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Error_Period)              {
Siri_SM_Error_Period         SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Error_Parameter_Missing)     {
Siri_SM_Error_Parameter_Missing SM;      SM.run();   }
TEST_F(Siri_ClientTestFixture, SM_Error_Parameter_null)        {
Siri_SM_Error_Parameter_null   SM;      SM.run();   }


void Siri_SM_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__StopMonitoringRequestStructure request;
   ns1__StopMonitoringAnswerStructure response;

   ns2__StopMonitoringRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

   xsd__dateTime* starttime_struct = new
xsd__dateTime(GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour))));
   req.StartTime = starttime_struct;
```

```
   boost::chrono::hours interval(desired_hours);
   boost::chrono::milliseconds ms =
boost::chrono::duration_cast<boost::chrono::milliseconds> (interval);

   req.PreviewInterval = new LONG64(ms.count());

   ns2__MonitoringRefStructure* monitoringref_structure = new
ns2__MonitoringRefStructure;
      monitoringref_structure->__item = get_random_stop().stopref;
   req.MonitoringRef = monitoringref_structure;

   const boost::posix_time::ptime ptime_start_time =
GSOAP::to_ptime(*req.StartTime);      // >= ptime_start_time of the period of
the data we get, but no guarateed to be =
   const boost::posix_time::millisec
preview_interval((int64_t)*req.PreviewInterval);
   //const boost::posix_time::ptime ptime_end_time = ptime_start_time +
preview_interval;   // Not guarenteed to be equal to the ValidUntil date of the
data we get

   set_proxy(ws);
   ASSERT_EQ(ws.GetStopMonitoring(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete context_structure->VelocityUnits;
   delete context_structure;
   delete info_structure;
   delete starttime_struct;
   delete monitoringref_structure;


   ASSERT_FALSE(response.Answer->StopMonitoringDelivery.empty());
   for (const auto& delivery : response.Answer->StopMonitoringDelivery)
   {
      ASSERT_NE(delivery, nullptr);
      ASSERT_EQ(delivery->ErrorCondition, nullptr);
      //ASSERT_EQ(ptime_end_time, GSOAP::to_ptime(*delivery->ValidUntil));   //
It's possible that delivery->ValidUntil > requested_valid_until due to a request
with a StartTime in the past,
      ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*delivery->ValidUntil) -
preview_interval);         // since PreviewInterval keeps its value

      ASSERT_FALSE(delivery->MonitoredStopVisit.empty());
      for (const auto& stopvisit : delivery->MonitoredStopVisit)
      {
         ASSERT_NE(stopvisit, nullptr);
         ASSERT_NE(stopvisit->__MonitoredStopVisitStructure_sequence, nullptr);
         ASSERT_NE(stopvisit->__MonitoredStopVisitStructure_sequence-
>MonitoringRef, nullptr);
         ASSERT_FALSE(stopvisit->__MonitoredStopVisitStructure_sequence-
>MonitoringRef->__item.empty());
         ASSERT_NE(stopvisit->MonitoredVehicleJourney, nullptr);
         ASSERT_NE(stopvisit->MonitoredVehicleJourney->DirectionRef, nullptr);
         ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->DirectionRef-
>__item.empty());
         ASSERT_NE(stopvisit->MonitoredVehicleJourney->LineRef, nullptr);
         ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->LineRef-
>__item.empty());
         ASSERT_FALSE(stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName.empty());
```

```
            for (const auto& name : stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->Delay, nullptr);

            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall, nullptr);
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall->Order,
nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall->Order-
>empty());
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointName.empty());
            for (const auto& name : stopvisit->MonitoredVehicleJourney-
>MonitoredCall->StopPointName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }

            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef, nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item.empty());

            if (stopvisit->MonitoredVehicleJourney-
>__MonitoredVehicleJourneyStructure_sequence__ && stopvisit-
>MonitoredVehicleJourney->__MonitoredVehicleJourneyStructure_sequence__-
>HeadwayService == true)
            {
               ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedArrivalTime, nullptr);
               ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedDepartureTime, nullptr);
               ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, nullptr);
               ASSERT_GT(*stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, 0);
            }
            else
            {
               ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedArrivalTime, nullptr);
               ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedDepartureTime, nullptr);
               ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, nullptr);
               ASSERT_LE(GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedDepartureTime), GSOAP::to_ptime(*delivery->ValidUntil));
               //ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*stopvisit-
>MonitoredVehicleJourney->MonitoredCall->AimedArrivalTime));
            }
         }
         ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(delivery->ResponseTimestamp));
      }

      if (siri_debug)
         show_SM(response);
   }
```

```
void Siri_SM_Start_Time_Missing::run()                {...}
void Siri_SM_Unknown_Stop::run()                      {...}
void Siri_SM_Error_Unauth::run()                      {...}
void Siri_SM_Error_Context_Missing::run()             {...}
void Siri_SM_Error_Period::run()                      {...}
void Siri_SM_Error_Parameter_Missing::run()           {...}
void Siri_SM_Error_Parameter_null::run()              {...}
```

Stop_Monitoring_Multiple.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_SMM_Base
{
public:
   Siri_SMM_Base() {}
   virtual ~Siri_SMM_Base() {};

protected:
   void show_SMM(const ns1__StopMonitoringAnswerStructure& response);
};


class Siri_SMM_Main :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Main()
      : BaseSIRIClient() {}
   ~Siri_SMM_Main() {};

   virtual void run() override;
};

class Siri_SMM_Start_Time_Missing :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Start_Time_Missing()
      : BaseSIRIClient() {}
   ~Siri_SMM_Start_Time_Missing() {};

   virtual void run() override;
};

class Siri_SMM_Unknown_Stop :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Unknown_Stop()
      : BaseSIRIClient() {}
   ~Siri_SMM_Unknown_Stop() {};

   virtual void run() override;
};
```

```
class Siri_SMM_Error_Unauth :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_SMM_Error_Unauth() {};

   virtual void run() override;
};

class Siri_SMM_Error_Context_Missing :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Error_Context_Missing()
      : BaseSIRIClient() {}
   ~Siri_SMM_Error_Context_Missing() {};

   virtual void run() override;
};

class Siri_SMM_Error_Period :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Error_Period()
      : BaseSIRIClient() {}
   ~Siri_SMM_Error_Period() {};

   virtual void run() override;
};

class Siri_SMM_Error_Parameter_Missing :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Error_Parameter_Missing()
      : BaseSIRIClient() {}
   ~Siri_SMM_Error_Parameter_Missing() {};

   virtual void run() override;
};

class Siri_SMM_Error_Parameter_null :
   public BaseSIRIClient,
   public Siri_SMM_Base
{
public:
   Siri_SMM_Error_Parameter_null()
      : BaseSIRIClient() {}
   ~Siri_SMM_Error_Parameter_null() {};

   virtual void run() override;
};
```

Stop_Monitoring_Multiple.cpp

```
#include "stdafx.h"
#include "Stop_Monitoring_Multiple.h"

void Siri_SMM_Base::show_SMM(const ns1__StopMonitoringAnswerStructure& response)
{
   _tstring path = _T("c:/temp/siri/stopmonitoring.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Stop Monitoring: " << std::endl << std::endl;

   unsigned cont = 0;
   for (const auto& delivery : response.Answer->StopMonitoringDelivery)
   {
      ofs << "Delivery #" << ++cont << std::endl;
      ofs << "Valid until: " << GSOAP::to_ptime(*delivery->ValidUntil) <<
std::endl;
      for (const auto& stopvisit : delivery->MonitoredStopVisit)
      {
         ofs << std::endl;
         ofs << "MonitoringRef: "      << stopvisit-
>__MonitoredStopVisitStructure_sequence->MonitoringRef->__item   << ", ";
         ofs << "LineRef: "            << stopvisit->MonitoredVehicleJourney-
>LineRef->__item                  << ", ";
         ofs << "DirectionRef: "         << stopvisit->MonitoredVehicleJourney-
>DirectionRef->__item               << ", ";
         ofs << "VehicleJourneyName: "   << stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName.front()->__item   << std::endl;

         ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall->Order
<< ". ";
         ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointName.front()->__item         << " - ";
         if (stopvisit->MonitoredVehicleJourney-
>__MonitoredVehicleJourneyStructure_sequence__ && stopvisit-
>MonitoredVehicleJourney->__MonitoredVehicleJourneyStructure_sequence__-
>HeadwayService==true)
         {
            ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " - frequency: ";
            ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval / (1000 * 60) << " minutes" << std::endl;
         }
         else
         {
            ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " -  from: ";
            ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedArrivalTime) << " to: ";
            ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedDepartureTime) << std::endl;
         }

         boost::posix_time::time_duration delay=
boost::posix_time::milliseconds(*stopvisit->MonitoredVehicleJourney->Delay);
         ofs << "delay: " << delay << std::endl;
      }
```

```
      for (const auto& stopcancellation : delivery-
>MonitoredStopVisitCancellation)
      {

      }
      ofs << "\nResponse time: " << GSOAP::to_ptime(delivery-
>ResponseTimestamp);
   }
}

TEST_F(Siri_ClientTestFixture, SMM_Main)                    {   Siri_SMM_Main
SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Start_Time_Missing)            {
Siri_SMM_Start_Time_Missing         SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Siri_SMM_Unknown_Stop)         {
Siri_SMM_Unknown_Stop             SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Error_Unauth)                {
Siri_SMM_Error_Unauth             SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Error_Context_Missing)         {
Siri_SMM_Error_Context_Missing      SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Error_Period)                {
Siri_SMM_Error_Period             SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Error_Parameter_Missing)         {
Siri_SMM_Error_Parameter_Missing    SMM;      SMM.run();   }
TEST_F(Siri_ClientTestFixture, SMM_Error_Parameter_null)          {
Siri_SMM_Error_Parameter_null       SMM;      SMM.run();   }



void Siri_SMM_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__StopMonitoringMultipleRequestStructure request;
   ns1__StopMonitoringAnswerStructure response;

   ns2__StopMonitoringMultipleRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

        xsd__dateTime* starttime_struct = new
xsd__dateTime(GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour))));
```

```
        boost::chrono::hours interval(desired_hours);
        boost::chrono::milliseconds ms =
boost::chrono::duration_cast<boost::chrono::milliseconds> (interval);

      LONG64* previewinterval = new LONG64(ms.count());


   std::unique_ptr<ns2__StopMonitoringFilterStructure> p_filter(new
ns2__StopMonitoringFilterStructure);
      std::unique_ptr<ns2__MonitoringRefStructure> p_monitoringref_structure(new
ns2__MonitoringRefStructure);
         p_monitoringref_structure->__item = get_random_stop().stopref;
      p_filter->StartTime= starttime_struct;
      p_filter->PreviewInterval = previewinterval;
      auto first_monitoring_structure = std::move(p_monitoringref_structure);
         p_filter->MonitoringRef = first_monitoring_structure.get();
      auto first_filter = std::move(p_filter);

      req.StopMonitoringFIlter.push_back(first_filter.get());

   p_filter.reset(new ns2__StopMonitoringFilterStructure);
      p_monitoringref_structure.reset(new ns2__MonitoringRefStructure);
      p_monitoringref_structure->__item = get_random_stop().stopref;
      p_filter->StartTime = starttime_struct;
      p_filter->PreviewInterval = previewinterval;
      auto second_monitoring_structure = std::move(p_monitoringref_structure);
         p_filter->MonitoringRef = second_monitoring_structure.get();
      auto second_filter = std::move(p_filter);

      req.StopMonitoringFIlter.push_back(second_filter.get());

   const boost::posix_time::ptime ptime_start_time =
GSOAP::to_ptime(*first_filter->StartTime);      // >= ptime_start_time of the
period of the data we get, but no guarateed to be =.    Assuming shared
StartTime for all filters
   const boost::posix_time::millisec preview_interval((int64_t)*first_filter-
>PreviewInterval);
   //const boost::posix_time::ptime ptime_end_time = ptime_start_time +
preview_interval;   // Not guarenteed to be equal to the ValidUntil date of the
data we get

   set_proxy(ws);
   ASSERT_EQ(ws.GetMultipleStopMonitoring(server_address.c_str(), nullptr,
&request, response), SOAP_OK) << "## Server might not be running ##";

   delete context_structure->VelocityUnits;
   delete context_structure;
   delete info_structure;
   delete starttime_struct;
   delete previewinterval;


   ASSERT_FALSE(response.Answer->StopMonitoringDelivery.empty());
   for (const auto& delivery : response.Answer->StopMonitoringDelivery)
   {
      ASSERT_NE(delivery, nullptr);
      ASSERT_EQ(delivery->ErrorCondition, nullptr);
      //ASSERT_EQ(ptime_end_time, GSOAP::to_ptime(*delivery->ValidUntil));
// It's possible that delivery->ValidUntil > requested_valid_until due to a
request with a StartTime in the past,
      ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*delivery->ValidUntil) -
preview_interval);         // since PreviewInterval keeps its value
```

```
        ASSERT_FALSE(delivery->MonitoredStopVisit.empty());
        for (const auto& stopvisit : delivery->MonitoredStopVisit)
        {
            ASSERT_NE(stopvisit, nullptr);
            ASSERT_NE(stopvisit->__MonitoredStopVisitStructure_sequence, nullptr);
            ASSERT_NE(stopvisit->__MonitoredStopVisitStructure_sequence-
>MonitoringRef, nullptr);
            ASSERT_FALSE(stopvisit->__MonitoredStopVisitStructure_sequence-
>MonitoringRef->__item.empty());
            ASSERT_NE(stopvisit->MonitoredVehicleJourney, nullptr);
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->DirectionRef, nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->DirectionRef-
>__item.empty());
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->LineRef, nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->LineRef-
>__item.empty());
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName.empty());
            for (const auto& name : stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->Delay, nullptr);

            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall, nullptr);
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall->Order,
nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall->Order-
>empty());
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointName.empty());
            for (const auto& name : stopvisit->MonitoredVehicleJourney-
>MonitoredCall->StopPointName)
            {
               ASSERT_NE(name, nullptr);
               ASSERT_FALSE(name->__item.empty());
            }
            ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef, nullptr);
            ASSERT_FALSE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item.empty());

            if (stopvisit->MonitoredVehicleJourney-
>__MonitoredVehicleJourneyStructure_sequence__ && stopvisit-
>MonitoredVehicleJourney->__MonitoredVehicleJourneyStructure_sequence__-
>HeadwayService == true)
            {
               ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedArrivalTime, nullptr);
               ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedDepartureTime, nullptr);
               ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, nullptr);
               ASSERT_GT(*stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, 0);
            }
```

```
            else
            {
                ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedArrivalTime, nullptr);
                ASSERT_NE(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedDepartureTime, nullptr);
                ASSERT_EQ(stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval, nullptr);
                ASSERT_LE(GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedDepartureTime), GSOAP::to_ptime(*delivery->ValidUntil));
                //ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*stopvisit-
>MonitoredVehicleJourney->MonitoredCall->AimedArrivalTime));
            }
        }
        ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(delivery->ResponseTimestamp));
    }

    if (siri_debug)
        show_SMM(response);
}



void Siri_SMM_Start_Time_Missing::run()         {...}
void Siri_SMM_Unknown_Stop::run()               {...}
void Siri_SMM_Error_Unauth::run()               {...}
void Siri_SMM_Error_Context_Missing::run()      {...}
void Siri_SMM_Error_Period::run()               {...}
void Siri_SMM_Error_Parameter_Missing::run()    {...}
void Siri_SMM_Error_Parameter_null::run()       {...}
```

Stop_Timetable.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_ST_Base
{
public:
   Siri_ST_Base() {}
   virtual ~Siri_ST_Base() {};

protected:
   void show_ST(const ns1__StopTimetableAnswerStructure& response);
};


class Siri_ST_Main :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_Main()
      : BaseSIRIClient() {}
   ~Siri_ST_Main() {};

   virtual void run() override;
};

class Siri_ST_No_Info :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_No_Info()
      : BaseSIRIClient() {}
   ~Siri_ST_No_Info() {};

   virtual void run() override;
};

class Siri_ST_Error_Unauth :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_ST_Error_Unauth() {};

   virtual void run() override;
};
```

```
class Siri_ST_Error_Period :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_Error_Period()
      : BaseSIRIClient() {}
   ~Siri_ST_Error_Period() {};

   virtual void run() override;
};

class Siri_ST_Error_Parameter_Missing :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_Error_Parameter_Missing()
      : BaseSIRIClient() {}
   ~Siri_ST_Error_Parameter_Missing() {};

   virtual void run() override;
};

class Siri_ST_Error_Parameter_null :
   public BaseSIRIClient,
   public Siri_ST_Base
{
public:
   Siri_ST_Error_Parameter_null()
      : BaseSIRIClient() {}
   ~Siri_ST_Error_Parameter_null() {};

   virtual void run() override;
};
```

Stop_Timetable.cpp

```
#include "stdafx.h"
#include "Stop_Timetable.h"

void Siri_ST_Base::show_ST(const ns1__StopTimetableAnswerStructure& response)
{
   _tstring path = _T("c:/temp/siri/stoptimetable.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Stop Timetable: " << std::endl << std::endl;

   ofs << "Valid until: " << GSOAP::to_ptime(*response.Answer-
>StopTimetableDelivery->ValidUntil) << std::endl;
   for (const auto& stopvisit : response.Answer->StopTimetableDelivery-
>TimetabledStopVisit)
   {
      ofs << std::endl;
      ofs << "MonitoringRef: " << stopvisit->MonitoringRef->__item << ", ";
      ofs << "LineRef: " << stopvisit->TargetedVehicleJourney->LineRef->__item
<< ", ";
      ofs << "DirectionRef: " << stopvisit->TargetedVehicleJourney-
>DirectionRef->__item << std::endl;

      ofs << *stopvisit->TargetedVehicleJourney->TargetedCall->Order << ". ";

      if (stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__ &&
         true == stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__->HeadwayService)
      {
         ofs << stopvisit->TargetedVehicleJourney->TargetedCall->StopPointRef-
>__item << " - frequency: ";
         ofs << *stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedHeadwayInterval / (1000 * 60) << " minutes" << std::endl;
      }
      else
      {
         ofs << stopvisit->TargetedVehicleJourney->TargetedCall->StopPointRef-
>__item << " -  from: ";
         ofs << GSOAP::to_ptime(*stopvisit->TargetedVehicleJourney-
>TargetedCall->AimedArrivalTime) << " to: ";
         ofs << GSOAP::to_ptime(*stopvisit->TargetedVehicleJourney-
>TargetedCall->AimedDepartureTime) << std::endl;
      }
   }

   for (const auto& stopcancellation : response.Answer->StopTimetableDelivery-
>TimetabledStopVisitCancellation)
   {

   }

   ofs << "\nResponse time: " << GSOAP::to_ptime(response.Answer-
>StopTimetableDelivery->ResponseTimestamp);
}
```

```
TEST_F(Siri_ClientTestFixture, ST_Main)                  {   Siri_ST_Main
ST;      ST.run();   }
TEST_F(Siri_ClientTestFixture, ST_No_Info)                  {   Siri_ST_No_Info
ST;      ST.run();   }
TEST_F(Siri_ClientTestFixture, ST_Error_Unauth)            {
Siri_ST_Error_Unauth         ST;      ST.run();   }
TEST_F(Siri_ClientTestFixture, ST_Error_Period)            {
Siri_ST_Error_Period         ST;      ST.run();   }
TEST_F(Siri_ClientTestFixture, ST_Error_Parameter_Missing)      {
Siri_ST_Error_Parameter_Missing ST;      ST.run();   }
TEST_F(Siri_ClientTestFixture, ST_Error_Parameter_null)      {
Siri_ST_Error_Parameter_null   ST;      ST.run();   }


void Siri_ST_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__StopTimetableRequestStructure request;
   ns1__StopTimetableAnswerStructure response;

   ns2__StopTimetableRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;
   request.ServiceRequestInfo = info_structure;

   ns2__ClosedTimestampRangeStructure* timestamp_structure = new
ns2__ClosedTimestampRangeStructure;
      xsd__dateTime__ start_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)));
      xsd__dateTime__ end_time =
GSOAP::from_ptime(boost::posix_time::ptime(start_date,
boost::posix_time::hours(start_hour)) + boost::gregorian::days(desired_days));
      timestamp_structure->StartTime = start_time;
      timestamp_structure->EndTime = end_time;
   req.DepartureWindow = timestamp_structure;

   ns2__MonitoringRefStructure* monitoringref_structure = new
ns2__MonitoringRefStructure;
      monitoringref_structure->__item = get_random_stop().stopref;
   req.MonitoringRef = monitoringref_structure;

   ns2__LineRefStructure* lineref_structure = new ns2__LineRefStructure;
      lineref_structure->__item = get_random_line().lineref;
   req.LineRef = lineref_structure;
```

```
   const boost::posix_time::ptime ptime_start_time =
GSOAP::to_ptime(req.DepartureWindow->StartTime);
   const boost::posix_time::ptime ptime_end_time =
GSOAP::to_ptime(req.DepartureWindow->EndTime);

   set_proxy(ws);
   ASSERT_EQ(ws.GetStopTimetable(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete context_structure->VelocityUnits;
   delete context_structure;
   delete info_structure;
   delete timestamp_structure;
   delete monitoringref_structure;
   delete lineref_structure;


   ASSERT_NE(response.Answer->StopTimetableDelivery, nullptr);
   ASSERT_EQ(response.Answer->StopTimetableDelivery->ErrorCondition, nullptr);
   //ASSERT_LE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(*response.Answer->StopTimetableDelivery->ValidUntil));

   ASSERT_FALSE(response.Answer->StopTimetableDelivery-
>TimetabledStopVisit.empty());
   for (const auto& stopvisit : response.Answer->StopTimetableDelivery-
>TimetabledStopVisit)
   {
      ASSERT_NE   (stopvisit, nullptr);
      ASSERT_NE   (stopvisit->MonitoringRef, nullptr);
      ASSERT_FALSE(stopvisit->MonitoringRef->__item.empty());
      ASSERT_NE   (stopvisit->TargetedVehicleJourney, nullptr);
      ASSERT_NE   (stopvisit->TargetedVehicleJourney->LineRef, nullptr);
      ASSERT_FALSE(stopvisit->TargetedVehicleJourney->LineRef->__item.empty());
      ASSERT_NE   (stopvisit->TargetedVehicleJourney->DirectionRef, nullptr);
      ASSERT_FALSE(stopvisit->TargetedVehicleJourney->DirectionRef-
>__item.empty());
      ASSERT_NE   (stopvisit->TargetedVehicleJourney->TargetedCall, nullptr);
      ASSERT_NE   (stopvisit->TargetedVehicleJourney->TargetedCall->Order,
nullptr);
      ASSERT_FALSE(stopvisit->TargetedVehicleJourney->TargetedCall->Order-
>empty());

      if (stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__ &&
         true== stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__->HeadwayService)
      {
         ASSERT_EQ(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedArrivalTime, nullptr);
         ASSERT_EQ(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedDepartureTime, nullptr);
         ASSERT_NE(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedHeadwayInterval, nullptr);
         ASSERT_GT(*stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedHeadwayInterval, 0);
      }
```

```
      else
      {
         ASSERT_NE(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedArrivalTime, nullptr);
         ASSERT_NE(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedDepartureTime, nullptr);
         ASSERT_EQ(stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedHeadwayInterval, nullptr);
         ASSERT_LE(GSOAP::to_ptime(*stopvisit->TargetedVehicleJourney-
>TargetedCall->AimedDepartureTime), ptime_end_time);
         ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*stopvisit-
>TargetedVehicleJourney->TargetedCall->AimedArrivalTime));
      }
   }
   ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.Answer->StopTimetableDelivery->ResponseTimestamp));

   if (siri_debug)
      show_ST(response);
}


void Siri_ST_Error_Unauth::run()                    {...}
void Siri_ST_Error_Period::run()                    {...}
void Siri_ST_Error_Parameter_Missing::run()         {...}
void Siri_ST_Error_Parameter_null::run()            {...}
void Siri_ST_No_Info::run()                         {...}
```

Stops_Discovery.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_SD_Base
{
public:
   Siri_SD_Base() {}
   virtual ~Siri_SD_Base() {};

protected:
   void show_SD(const ns1__WsStopPointsDiscoveryAnswerStructure& response);
};


class Siri_SD_Main :
   public BaseSIRIClient,
   public Siri_SD_Base
{
public:
   Siri_SD_Main()
      : BaseSIRIClient() {}
   ~Siri_SD_Main() {};

   virtual void run() override;
};

class Siri_SD_Error_Unauth :
   public BaseSIRIClient,
   public Siri_SD_Base
{
public:
   Siri_SD_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_SD_Error_Unauth() {};

   virtual void run() override;
};
```

Stops_Discovery.cpp

```
#include "stdafx.h"
#include "Stops_Discovery.h"

void Siri_SD_Base::show_SD(const ns1__WsStopPointsDiscoveryAnswerStructure&
response)
{
   _tstring path = _T("c:/temp/siri/stops_discovery.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "StopPoints discovery: " << std::endl << std::endl;

   ofs << "Valid until: " << GSOAP::to_ptime(*response.Answer->ValidUntil) <<
std::endl;
   int iter = 0;
   for (const auto& stop : response.Answer->AnnotatedStopPointRef)
   {
      ofs << ++iter << '\t';
      ofs << stop->StopName.front()->__item << " ";
      ofs << stop->StopPointRef->__item << " ";
      ofs << std::boolalpha << stop->Monitored << " ";
      ofs << *(stop->Location->Latitude) << " ";
      ofs << *(stop->Location->Longitude) << std::endl;
   }
   ofs << '\n' << GSOAP::to_ptime(response.Answer->ResponseTimestamp) << std::endl;
}

TEST_F(Siri_ClientTestFixture, SD_Main)            {   Siri_SD_Main         SD;
SD.run();   }
TEST_F(Siri_ClientTestFixture, SD_Error_Unauth)      {   Siri_SD_Error_Unauth   SD;
SD.run();   }

void Siri_SD_Main::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsStopPointsDiscoveryStructure request;
   ns1__WsStopPointsDiscoveryAnswerStructure response;

   ns2__StopPointsDiscoveryRequestStructure req;

   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.StopPointsDetailLevel = ns2__StopPointsDetailEnumeration__normal;

   req.AccountId = &user;
   req.AccountKey = &pwd;

   set_proxy(ws);
   ASSERT_EQ(ws.StopPointsDiscovery(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";
```

```
   ASSERT_EQ(response.Answer->ErrorCondition, nullptr);
   ASSERT_FALSE(response.Answer->AnnotatedStopPointRef.empty());
   for (const auto& stop : response.Answer->AnnotatedStopPointRef)
   {
      ASSERT_NE   (stop, nullptr);
      ASSERT_FALSE(stop->StopName.empty());
      for (const auto& name : (stop->StopName))
      {
         ASSERT_NE   (name, nullptr);
         ASSERT_FALSE(name->__item.empty());
      }
      ASSERT_NE   (stop->StopPointRef, nullptr);
      ASSERT_FALSE(stop->StopPointRef->__item.empty());
      ASSERT_NE   (stop->Location, nullptr);
      ASSERT_NE   (stop->Location->Latitude, nullptr);
      ASSERT_NE   (stop->Location->Longitude, nullptr);
      ASSERT_FALSE(stop->Location->Latitude->empty());
      ASSERT_FALSE(stop->Location->Longitude->empty());
   }
   ASSERT_NE   (response.Answer->ValidUntil, nullptr);
   ASSERT_GE   (boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(response.Answer->ResponseTimestamp));

   if (siri_debug)
      show_SD(response);
}

void Siri_SD_Error_Unauth::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsStopPointsDiscoveryStructure request;
   ns1__WsStopPointsDiscoveryAnswerStructure response;

   ns2__StopPointsDiscoveryRequestStructure req;

   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
   req.StopPointsDetailLevel = ns2__StopPointsDetailEnumeration__normal;

   req.AccountId = &unauth_user;
   req.AccountKey = &pwd;

   //set_proxy(ws);
   ASSERT_EQ(ws.StopPointsDiscovery(server_address.c_str(), nullptr, &request,
response),0);

   ASSERT_NE(response.Answer->ErrorCondition, nullptr);
   ASSERT_NE(response.Answer->ErrorCondition->Description, nullptr);
   ASSERT_EQ(response.Answer->ErrorCondition->Description->__item,
unauth_error_text);
   ASSERT_EQ(response.Answer->ErrorCondition-
>__union_ServiceDeliveryErrorConditionStructure__,
SOAP_UNION__ns2__union_ServiceDeliveryErrorConditionStructure___EndpointDeniedAc
cessError);
   ASSERT_EQ(*response.Answer->ErrorCondition-
>union_ServiceDeliveryErrorConditionStructure__.EndpointDeniedAccessError-
>ErrorText, unauth_error_text);

   ASSERT_TRUE(response.Answer->AnnotatedStopPointRef.empty());
}
```

Terminate_Subscription.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_TSUB_Base
{
public:
   Siri_TSUB_Base() {}
   virtual ~Siri_TSUB_Base() {};
};


class Siri_TSUB_One :
   public BaseSIRIClient,
   public Siri_TSUB_Base
{
public:
   Siri_TSUB_One()
      : BaseSIRIClient() {}
   ~Siri_TSUB_One() {};

   virtual void run() override;
};

class Siri_TSUB_All :
   public BaseSIRIClient,
   public Siri_TSUB_Base
{
public:
   Siri_TSUB_All()
      : BaseSIRIClient() {}
   ~Siri_TSUB_All() {};

   virtual void run() override;
};

class Siri_TSUB_Error_Unauth :
   public BaseSIRIClient,
   public Siri_TSUB_Base
{
public:
   Siri_TSUB_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_TSUB_Error_Unauth() {};

   virtual void run() override;
};

class Siri_TSUB_Error_SubID_Missing :
   public BaseSIRIClient,
   public Siri_TSUB_Base
{
public:
   Siri_TSUB_Error_SubID_Missing()
      : BaseSIRIClient() {}
   ~Siri_TSUB_Error_SubID_Missing() {};

   virtual void run() override;
};
```

Terminate_Subscription.cpp

```
#include "stdafx.h"
#include "Terminate_Subscription.h"

TEST_F(Siri_ClientTestFixture, TSUB_Error_Unauth)            {
Siri_TSUB_Error_Unauth         TSUB;      TSUB.run();   }
TEST_F(Siri_ClientTestFixture, TSUB_Error_SubID_Missing)      {
Siri_TSUB_Error_SubID_Missing      TSUB;      TSUB.run();   }
TEST_F(Siri_ClientTestFixture, TSUB_One)                   { Siri_TSUB_One
TSUB;      TSUB.run();   }
TEST_F(Siri_ClientTestFixture, TSUB_All)                   { Siri_TSUB_All
TSUB;      TSUB.run();   }

void Siri_TSUB_One::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsDeleteSubscriptionRequestStructure request;
   ns1__WsDeleteSubscriptionAnswerStructure response;

   ns2__TerminateSubscriptionRequestBodyStructure req;
   request.Request = &req;

      ns2__RequestStructure* info_structure = new ns2__RequestStructure;
         info_structure->AccountId = &user;
         info_structure->AccountKey = &pwd;
         info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
         info_structure->Address = &consumer_address;

         ns2__ParticipantRefStructure* requestor_ref = new
ns2__ParticipantRefStructure;
            std::wstringstream ss;
            ss.imbue(std::locale::classic());
            ss << int_distribution(random_generator);
            requestor_ref->__item = L"Subscriber #" + ss.str();
         info_structure->RequestorRef = requestor_ref;
         // ...

      request.DeleteSubscriptionInfo = info_structure;

      req.SubscriberRef = requestor_ref;
      req.__union_TerminateSubscriptionRequestBodyStructure =
SOAP_UNION__ns2__union_TerminateSubscriptionRequestBodyStructure_SubscriptionRef
;

      std::vector<ns2__SubscriptionQualifierStructure *>*
v_termination_requests_struct = new
std::vector<ns2__SubscriptionQualifierStructure *>;
         ns2__SubscriptionQualifierStructure* subscription_ref_struct = new
ns2__SubscriptionQualifierStructure;      // Subscription identifier
            std::vector<std::wstring>::const_iterator it;
            if (!v_subscription_references.empty())
            {
               it = v_subscription_references.begin();
               std::uniform_int_distribution<std::size_t> random_ref(0,
v_subscription_references.size() - 1);
               std::advance(it, random_ref(random_generator));
               subscription_ref_struct->__item = *it;
            }
```

```
            else
               subscription_ref_struct->__item = L"5908c305-6cbb-4817-8c95-
22de48c6772b";   // Avoids crash test if v_termination_requests_struct.empty()
      v_termination_requests_struct->push_back(subscription_ref_struct);
   req.union_TerminateSubscriptionRequestBodyStructure.SubscriptionRef =
v_termination_requests_struct;

   set_proxy(ws);
   ASSERT_EQ(ws.DeleteSubscription(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete requestor_ref;
   delete info_structure;
   delete subscription_ref_struct;
   delete v_termination_requests_struct;

   ASSERT_EQ(response.Answer->TerminationResponseStatus.size(), 1);
   for (const auto& response : response.Answer->TerminationResponseStatus)
   {
      ASSERT_NE(response, nullptr);
      ASSERT_EQ(response->ErrorCondition, nullptr);
      ASSERT_EQ(response->Status, true);
   }

   if(!v_subscription_references.empty())
      v_subscription_references.erase(it);
}

void Siri_TSUB_All::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__WsDeleteSubscriptionRequestStructure request;
   ns1__WsDeleteSubscriptionAnswerStructure response;

   ns2__TerminateSubscriptionRequestBodyStructure req;
   request.Request = &req;

      ns2__RequestStructure* info_structure = new ns2__RequestStructure;
         info_structure->AccountId = &user;
         info_structure->AccountKey = &pwd;
         info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
         info_structure->Address = &consumer_address;

         ns2__ParticipantRefStructure* requestor_ref = new
ns2__ParticipantRefStructure;
            std::wstringstream ss;
            ss.imbue(std::locale::classic());
            ss << int_distribution(random_generator);
            requestor_ref->__item = L"Subscriber #" + ss.str();
         info_structure->RequestorRef = requestor_ref;
         // ...
```

```
      request.DeleteSubscriptionInfo = info_structure;

   req.SubscriberRef = requestor_ref;
   req.__union_TerminateSubscriptionRequestBodyStructure =
SOAP_UNION__ns2__union_TerminateSubscriptionRequestBodyStructure_All;
   req.union_TerminateSubscriptionRequestBodyStructure.All =
ns2__EmptyType___x0000;

   set_proxy(ws);
   ASSERT_EQ(ws.DeleteSubscription(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete requestor_ref;
   delete info_structure;


   ASSERT_EQ(response.Answer->TerminationResponseStatus.size(), 1);
   for (const auto& response : response.Answer->TerminationResponseStatus)
   {
      ASSERT_NE(response, nullptr);
      ASSERT_EQ(response->ErrorCondition, nullptr);
      ASSERT_EQ(response->Status, true);
   }

   v_subscription_references.clear();
}


void Siri_TSUB_Error_Unauth::run()                    {...}
void Siri_TSUB_Error_SubID_Missing::run()             {...}
```

Vehicle_Monitoring.h

```
#pragma once

#include "BaseSIRIClient.h"
#include "Siri_ClientTestFixture.h"

class Siri_VM_Base
{
public:
   Siri_VM_Base() {}
   virtual ~Siri_VM_Base() {};

protected:
   void show_VM(const ns1__VehicleMonitoringAnswerStructure& response);
};


class Siri_VM_Main_Line :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Main_Line()
      : BaseSIRIClient() {}
   ~Siri_VM_Main_Line() {};

   virtual void run() override;
};

class Siri_VM_Main_Line_Direction :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Main_Line_Direction()
      : BaseSIRIClient() {}
   ~Siri_VM_Main_Line_Direction() {};

   virtual void run() override;
};

class Siri_VM_Main_Vehicle :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Main_Vehicle()
      : BaseSIRIClient() {}
   ~Siri_VM_Main_Vehicle() {};

   virtual void run() override;
};
```

```
class Siri_VM_Main_Vehicle_Direction :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Main_Vehicle_Direction()
      : BaseSIRIClient() {}
   ~Siri_VM_Main_Vehicle_Direction() {};

   virtual void run() override;
};

class Siri_VM_Error_Unknown_Vehicle :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Unknown_Vehicle()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Unknown_Vehicle() {};

   virtual void run() override;
};

class Siri_VM_Error_Unknown_Line :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Unknown_Line()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Unknown_Line() {};

   virtual void run() override;
};

class Siri_VM_Error_Unauth :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Unauth()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Unauth() {};

   virtual void run() override;
};
```

```
class Siri_VM_Error_Context_Missing :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Context_Missing()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Context_Missing() {};

   virtual void run() override;
};

class Siri_VM_Error_Parameter_Missing :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Parameter_Missing()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Parameter_Missing() {};

   virtual void run() override;
};

class Siri_VM_Error_Parameter_null :
   public BaseSIRIClient,
   public Siri_VM_Base
{
public:
   Siri_VM_Error_Parameter_null()
      : BaseSIRIClient() {}
   ~Siri_VM_Error_Parameter_null() {};

   virtual void run() override;
};
```

Vehicle_Monitoring.cpp

```
#include "stdafx.h"
#include "Vehicle_Monitoring.h"

void Siri_VM_Base::show_VM(const ns1__VehicleMonitoringAnswerStructure&
response)
{
   _tstring path = _T("c:/temp/siri/vehiclemonitoring.txt");
   const fs::path p = path;

   _ofstream ofs(p.string());
   ofs.imbue(loc);

   ofs << "Vehicle Monitoring: " << std::endl << std::endl;

   unsigned cont = 0;
   for (const auto& delivery : response.Answer->VehicleMonitoringDelivery)
   {
      ofs << "Delivery #" << ++cont << ",  ";
      ofs << "valid until: " << GSOAP::to_ptime(*delivery->ValidUntil) <<
std::endl;
      for (const auto& activity : delivery->VehicleActivity)
      {
         ofs << std::endl;
         ofs << "LineRef: "            << activity-
>MonitoredVehicleJourney.LineRef->__item                  << ", ";
         ofs << "DirectionRef: "         << activity-
>MonitoredVehicleJourney.DirectionRef->__item               << ", ";
         ofs << "VehicleJourneyName: "    << activity-
>MonitoredVehicleJourney.VehicleJourneyName.front()->__item      << std::endl;
         ofs << "Long. tramo: "          << *activity->ProgressBetweenStops-
>LinkDistance                      << " m" << std::endl;
         ofs << "% Recorrido: "          << *activity->ProgressBetweenStops-
>Percentage                         << " %" << std::endl;
         ofs << "Service ID: "         << *activity->ItemIdentifier << " " <<
activity->VehicleMonitoringRef->__item << std::endl;
         boost::posix_time::time_duration delay=
boost::posix_time::milliseconds(*activity->MonitoredVehicleJourney.Delay);
         ofs << "Delay: " << delay << std::endl;
         ofs << "Speed: " << *activity->MonitoredVehicleJourney.Velocity << "
km/h - ";
         ofs << "Bearing: " << *activity->MonitoredVehicleJourney.Bearing << "
degrees" << std::endl;
         for (const auto& future_stop : activity-
>MonitoredVehicleJourney.OnwardCalls->OnwardCall)
         {
            ofs << *future_stop->Order                           << ". ";
            ofs << future_stop->StopPointName.front()->__item         << " - ";
            if (activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__&& activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__->HeadwayService == true)
            {
               ofs << future_stop->StopPointRef->__item << " - frequency: ";
               ofs << *future_stop->AimedHeadwayInterval / (1000 * 60) << "
minutes" << std::endl;
            }
```

```
            else
            {
               ofs << future_stop->StopPointRef->__item << " -  from: ";
               ofs << GSOAP::to_ptime(*future_stop->AimedArrivalTime) << " to:
";
               ofs << GSOAP::to_ptime(*future_stop->AimedDepartureTime) <<
std::endl;
            }
         }
      }

      for (const auto& stopcandellation : delivery->VehicleActivityCancellation)
      {

      }
      ofs << "\nResponse time: " << GSOAP::to_ptime(delivery-
>ResponseTimestamp);
   }
}

TEST_F(Siri_ClientTestFixture, VM_Main_Line)              {   Siri_VM_Main_Line
VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Main_Line_Direction)         {
Siri_VM_Main_Line_Direction      VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Main_Vehicle)             {
Siri_VM_Main_Vehicle         VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Main_Vehicle_Direction)      {
Siri_VM_Main_Vehicle_Direction   VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Unknown_Line)         {
Siri_VM_Error_Unknown_Line      VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Unknown_Vehicle)      {
Siri_VM_Error_Unknown_Vehicle   VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Unauth)             {
Siri_VM_Error_Unauth         VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Context_Missing)      {
Siri_VM_Error_Context_Missing   VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Parameter_Missing)      {
Siri_VM_Error_Parameter_Missing VM;      VM.run();   }
TEST_F(Siri_ClientTestFixture, VM_Error_Parameter_null)      {
Siri_VM_Error_Parameter_null   VM;      VM.run();   }

void Siri_VM_Main_Line_Direction::run()
{
   SiriProducerDocBindingProxy ws;

   ns1__VehicleMonitoringRequestStructure request;
   ns1__VehicleMonitoringAnswerStructure response;

   ns2__VehicleMonitoringRequestStructure req;
   request.Request = &req;
   req.RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());

   ns1__WsServiceRequestInfoStructure* info_structure = new
ns1__WsServiceRequestInfoStructure;
      info_structure->AccountId = &user;
      info_structure->AccountKey = &pwd;
      info_structure->RequestTimestamp =
GSOAP::from_ptime(boost::posix_time::second_clock::universal_time());
```

```
      ns2__ServiceRequestContextStructure* context_structure = new
ns2__ServiceRequestContextStructure;
         context_structure->Language = desired_language;
         context_structure->VelocityUnits = new std::wstring(velocity_units);
      info_structure->ServiceRequestContext = context_structure;

   request.ServiceRequestInfo = info_structure;

   ns2__VehicleMonitoringRefStructure* vehicle_monitoringref_structure = new
ns2__VehicleMonitoringRefStructure;
      vehicle_monitoringref_structure->__item = get_random_line().lineref;
   req.VehicleMonitoringRef= vehicle_monitoringref_structure;

   req.__union_VehicleMonitoringRequestStructure = 2;

   ns2__LineRefStructure* line_ref_structure = new ns2__LineRefStructure;
      line_ref_structure->__item = std::wstring(req.VehicleMonitoringRef-
>__item);
   req.union_VehicleMonitoringRequestStructure.LineRef = line_ref_structure;

   ns2__DirectionRefStructure* direction_ref_structure = new
ns2__DirectionRefStructure;
      direction_ref_structure->__item = desired_direction;
   req.DirectionRef = direction_ref_structure;

   //set_proxy(ws);
   ASSERT_EQ(ws.GetVehicleMonitoring(server_address.c_str(), nullptr, &request,
response), SOAP_OK) << "## Server might not be running ##";

   delete context_structure->VelocityUnits;
   delete context_structure;
   delete info_structure;
   delete line_ref_structure;
   delete direction_ref_structure;


   ASSERT_FALSE(response.Answer->VehicleMonitoringDelivery.empty());
   for (const auto& delivery : response.Answer->VehicleMonitoringDelivery)
   {
      ASSERT_NE(delivery, nullptr);
      ASSERT_EQ(delivery->ErrorCondition, nullptr);
      ASSERT_LE(GSOAP::to_ptime(req.RequestTimestamp),
GSOAP::to_ptime(*delivery->ValidUntil));
      ASSERT_LE(GSOAP::to_ptime(req.RequestTimestamp), GSOAP::to_ptime(delivery-
>ResponseTimestamp));

      //EXPECT_GT(delivery->VehicleActivity.size(), 0) << "DirectionRef correct,
but there might not be vehicles now";
      if (delivery->VehicleActivity.size())
      {
         const std::wstring dir(delivery->VehicleActivity.front()-
>MonitoredVehicleJourney.DirectionRef->__item);
```

```
            for (const auto& activity : delivery->VehicleActivity)
            {
                ASSERT_NE   (activity, nullptr);
                ASSERT_NE   (activity->ItemIdentifier, nullptr);
                ASSERT_FALSE(activity->ItemIdentifier->empty());
                ASSERT_NE   (activity->ProgressBetweenStops, nullptr);
                ASSERT_NE   (activity->ProgressBetweenStops->LinkDistance, nullptr);
                ASSERT_FALSE(activity->ProgressBetweenStops->LinkDistance->empty());
                ASSERT_NE   (activity->MonitoredVehicleJourney.DirectionRef,
nullptr);
                ASSERT_FALSE(activity->MonitoredVehicleJourney.DirectionRef-
>__item.empty());
                //EXPECT_EQ   (activity->MonitoredVehicleJourney.DirectionRef-
>__item, dir) << "Might fail if a non existing direction is requested";      //
TO-CHECK
                ASSERT_NE   (activity->MonitoredVehicleJourney.LineRef, nullptr);
                ASSERT_FALSE(activity->MonitoredVehicleJourney.LineRef-
>__item.empty());
                ASSERT_FALSE(activity-
>MonitoredVehicleJourney.VehicleJourneyName.empty());
                ASSERT_NE   (activity->MonitoredVehicleJourney.Bearing, nullptr);
                ASSERT_NE   (activity->MonitoredVehicleJourney.Velocity, nullptr);
                ASSERT_FALSE(activity->MonitoredVehicleJourney.Velocity->empty());
                ASSERT_NE   (activity->MonitoredVehicleJourney.ProgressRate,
nullptr);

                for (const auto& name : activity-
>MonitoredVehicleJourney.VehicleJourneyName)
                {
                    ASSERT_NE(name, nullptr);
                    ASSERT_FALSE(name->__item.empty());
                }
                ASSERT_NE(activity->MonitoredVehicleJourney.Delay, nullptr);

                ASSERT_NE(activity->MonitoredVehicleJourney.OnwardCalls, nullptr);
                ASSERT_FALSE(activity->MonitoredVehicleJourney.OnwardCalls-
>OnwardCall.empty());
                for (const auto& future_stop : activity-
>MonitoredVehicleJourney.OnwardCalls->OnwardCall)
                {
                    ASSERT_NE(future_stop->Order, nullptr);
                    ASSERT_FALSE(future_stop->Order->empty());

                    ASSERT_FALSE(future_stop->StopPointName.empty());
                    for (const auto& name : future_stop->StopPointName)
                    {
                        ASSERT_NE(name, nullptr);
                        ASSERT_FALSE(name->__item.empty());
                    }
                    ASSERT_NE(future_stop->StopPointRef, nullptr);
                    ASSERT_FALSE(future_stop->StopPointRef->__item.empty());
                    if (activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__ && activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__->HeadwayService == true)
                    {
                        ASSERT_EQ(future_stop->AimedArrivalTime, nullptr);
                        ASSERT_EQ(future_stop->AimedDepartureTime, nullptr);
                        ASSERT_NE(future_stop->AimedHeadwayInterval, nullptr);
                        ASSERT_GT(*future_stop->AimedHeadwayInterval, 0);
                    }
```

```
                else
                {
                    ASSERT_NE(future_stop->AimedArrivalTime, nullptr);
                    ASSERT_NE(future_stop->AimedDepartureTime, nullptr);
                    ASSERT_EQ(future_stop->AimedHeadwayInterval, nullptr);
                    ASSERT_LE(GSOAP::to_ptime(*future_stop->AimedDepartureTime),
GSOAP::to_ptime(*delivery->ValidUntil));
                    //ASSERT_LE(ptime_start_time, GSOAP::to_ptime(*future_stop-
>AimedArrivalTime));
                }
            }
        }
    }
        ASSERT_GE(boost::posix_time::second_clock::universal_time(),
GSOAP::to_ptime(delivery->ResponseTimestamp));
    }

    //if (siri_debug)
    //    show_VM(response);
}



void Siri_VM_Main_Line::run()                          {...}
void Siri_VM_Main_Vehicle::run()                       {...}
void Siri_VM_Main_Vehicle_Direction::run()             {...}
void Siri_VM_Error_Unknown_Line::run()                 {...}
void Siri_VM_Error_Unknown_Vehicle::run()              {...}
void Siri_VM_Error_Unauth::run()                       {...}
void Siri_VM_Error_Context_Missing::run()              {...}
void Siri_VM_Error_Parameter_Missing::run()            {...}
void Siri_VM_Error_Parameter_null::run()               {...}
```

Anexo 2.2: Notification_Consumer

El código puede consultarse también en los archivos adjuntos a este documento o en el siguiente repositorio público:
https://github.com/eduherminio/SIRI_Notification_Consumer

Estructura de archivos:

- SIRI_Notification_Consumer.cpp (*main*)
- IInterfazSOAP.h
- ServerImp.h / .cpp
- NotificationConsumer.h
- ConsumerService.h / .cpp
- PrintUtils.hpp
- soap_srv.nsmap

- stdsoap2.h / .cpp (gSOAP)
- duration.h / .c (gSOAP)
- struct_tm.h / .c (gSOAP)
- long_double.h / .c (gSOAP)
- soapH.h (gSOAP, autogenerado)
- soapC.c (gSOAP, autogenerado)
- soapStub.h (gSOAP, autogenerado)
- soapSiriConsumerDocBindingService.h / .cpp (gSOAP, autogenerados)
- targetver.h (Visual Studio)
- stdafx.h / stdafx.cpp (Visual Studio)
- SIRI_Notification_Consumer.vcxproj (Visual Studio)

SIRI_Notification_Consumer.cpp

```
// SIRI_Notification_Consumer.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include "ServerImp.h"

const int defaultPort = 8081;

int main(int argc, char* argv[])
{
   CServerImp srv(20);

   int connectionPort = argc == 2 ? atoi(argv[1]) : defaultPort;

   srv.connect(connectionPort);

   std::cout << "Awaiting info on http://localhost:" << connectionPort << std::endl;

   bool input;
   do
   {
      std::cin >> input;
   } while (input == true);

   srv.disconnect();

   while (true)
      if (srv.get_m_safeOff() == true)
         return 0;
}
```

ServerImp.h

```
#pragma once

#include "stdsoap2.h"

#include "NotificationConsumer.h"
#include <mutex>
#include <atomic>
#include <memory>

class CServerImp
{
public:
   CServerImp(int poolSize);
   ~CServerImp(void);

   void connect(int port);
   void disconnect();

   const bool get_m_safeOff() const { return m_safeOff; }
protected:
   void thReceiver();
   void process(struct soap* _soap);

private:
   mutable std::mutex m;

   int m_port;
   int m_poolsize;
   std::atomic_bool m_bFinalizarTh;
   std::atomic_bool m_safeOff;

   std::unique_ptr<NotificationConsumer> ptr_srv;

   std::thread m_th;

   SOAP_SOCKET m_socket;
};
```

ServerImp.cpp

```
#include "stdafx.h"
#include "ServerImp.h"

#include "threadpool.hpp"

//Necesario este .nsmap para que se reconozcan los namespaces de SOAP
//Evitar WITH_NONAMESPACE, o las peticiones SOAP no se parsearan correctamente
//Evitar también soapcpp2 -n
#include "soap_srv.nsmap"

CServerImp::CServerImp(int poolSize)
   : m_poolsize(poolSize), m_port(8080), m_safeOff(false)
{
   ptr_srv.reset(new NotificationConsumer);
}

CServerImp::~CServerImp(void)
{
   if (m_th.joinable())
   {
      std::cout<<"Closing soap interface";

      m_bFinalizarTh = true;
      //Cerrar el socket
#ifdef _WIN32
      closesocket(m_socket);
#else
      close(m_socket);
#endif
      m_th.join();
   }
}

void CServerImp::connect(int port)
{
   m_bFinalizarTh = false;
   m_port = port;
   m_th = std::thread((std::bind(&CServerImp::thReceiver, this)));
}

void CServerImp::disconnect()
{
   m_bFinalizarTh = true;
}

void CServerImp::thReceiver()
{
   boost::threadpool::fifo_pool tp(m_poolsize);

   struct soap *_soap = soap_new();

   {
#ifndef _WIN32
      //Establece un timeout de 1 segundo porque en linux close no cierra el
socket master
      _soap->accept_timeout = 1;
#endif
```

```
      m_socket = soap_bind(_soap, NULL, m_port, 100);

      if (!soap_valid_socket(m_socket))
      {
         std::cout << "Error al hacer el bind al puerto " << m_port <<
std::endl;
         soap_destroy(_soap);
         soap_end(_soap);
         soap_free(_soap); // only safe when abc, uvw, xyz are also deleted

         return;
      }

      while (m_bFinalizarTh == false)   // TO-CHECK
      {
         SOAP_SOCKET sock = soap_accept(_soap);

         if (m_bFinalizarTh == false && !soap_valid_socket(sock))
         {
            if (_soap->errnum != 0) //En Linux como accept_timeout = 1 cada
segundo entra aquí, a pesar de no ser un error
               std::cout << "Error en soap_accept %d" << _soap->errnum <<
std::endl;
         }
         else
         {
            if (m_bFinalizarTh == false)
            {
               struct soap *_soapCopy = soap_copy(_soap);
               if (_soapCopy != nullptr)
               {
                  typedef std::function<void()> void_function_t;
                  void_function_t f = std::bind(&CServerImp::process, this,
_soapCopy);
                  tp.schedule(f);
               }
            }
         }
      }
   }

   tp.wait();
   tp.clear();

   soap_destroy(_soap);
   soap_end(_soap);
   soap_free(_soap); // only safe when abc, uvw, xyz are also deleted

   std::cout<<"Closed soap interface";

   m_safeOff = true;
}
```

```
void CServerImp::process(struct soap* _soap)
{
   soap_set_namespaces(_soap, srv_namespaces);   // srv_namespaces defined in
soap_srv.nsmap, won't work if not included

   int err = soap_begin_serve(_soap);
   if (err != SOAP_OK)
   {
      std::cout << "SOAP incoming call error " << _soap->errnum << std::endl;
   }
   else
   {
       //if (ptr_srv->process(_soap) != SOAP_NO_METHOD)   TO-CHECK
         //break;

      ptr_srv->process(_soap);
   }

   soap_destroy(_soap);
   soap_end(_soap);
   soap_free(_soap);
}
```

Notification_Consumer.h

```
#include "IInterfazSOAP.h"
#include "ConsumerService.h"

class NotificationConsumer :
   public IInterfazSOAP
{
public:
   NotificationConsumer(void) {}
   ~NotificationConsumer(void) {}

   virtual int process(struct soap* _soap)      // Detect connections from
clients
   {
      ConsumerService sl(_soap);
      return sl.dispatch();
   }
};
```

IInterfazSOAP.h

```
#pragma once

class IInterfazSOAP
{
public:
   IInterfazSOAP() {}
   virtual ~IInterfazSOAP() {}

   virtual int process(struct soap* _soap) = 0;
};
```

ConsumerService.h

```
#pragma once

#include "stdsoap2.h"
#include "soapSiriConsumerDocBindingService.h"
#include <mutex>
#include <memory>

class ConsumerService :
   public SiriConsumerDocBindingService
{
public:
   ConsumerService();
   ConsumerService(struct soap*);
   virtual ~ConsumerService();

   virtual   SiriConsumerDocBindingService *copy() override { return new
ConsumerService(*this); }

   // Only Consumer methods
   /// Web service one-way operation 'NotifySubscriptionTerminated' (return
SOAP_OK (no response) or error code, or use
send_NotifySubscriptionTerminated_empty_response())
   virtual int
NotifySubscriptionTerminated(ns1__WsSubscriptionTerminatedNotificationStructure
*ns1__NotifySubscriptionTerminated) override;
   /// Web service one-way operation 'NotifyDataReady' (return SOAP_OK (no
response) or error code, or use send_NotifyDataReady_empty_response())
   virtual int NotifyDataReady(ns1__WsDataReadyNotificationStructure
*ns1__NotifyDataReady) override;
   /// Web service one-way operation 'NotifyHeartbeat' (return SOAP_OK (no
response) or error code, or use send_NotifyHeartbeat_empty_response())
   virtual int NotifyHeartbeat(ns1__WsHeartbeatNotificationStructure
*ns1__NotifyHeartbeat) override;
   /// Web service one-way operation 'NotifyProductionTimetable' (return SOAP_OK
(no response) or error code, or use
send_NotifyProductionTimetable_empty_response())
   virtual int
NotifyProductionTimetable(ns1__WsProductionTimetableNotificationStructure
*ns1__NotifyProductionTimetable) override;
   /// Web service one-way operation 'NotifyEstimatedTimetable' (return SOAP_OK
(no response) or error code, or use
send_NotifyEstimatedTimetable_empty_response())
   virtual int
NotifyEstimatedTimetable(ns1__WsEstimatedTimetableNotificationStructure
*ns1__NotifyEstimatedTimetable) override;
   /// Web service one-way operation 'NotifyStopTimetable' (return SOAP_OK (no
response) or error code, or use send_NotifyStopTimetable_empty_response())
   virtual int NotifyStopTimetable(ns1__WsStopTimetableNotificationStructure
*ns1__NotifyStopTimetable) override;
   /// Web service one-way operation 'NotifyStopMonitoring' (return SOAP_OK (no
response) or error code, or use send_NotifyStopMonitoring_empty_response())
   virtual int NotifyStopMonitoring(ns1__WsStopMonitoringNotificationStructure
*ns1__NotifyStopMonitoring) override;
```

```
   /// Web service one-way operation 'NotifyVehicleMonitoring' (return SOAP_OK
(no response) or error code, or use
send_NotifyVehicleMonitoring_empty_response())
   virtual int
NotifyVehicleMonitoring(ns1__WsVehicleMonitoringNotificationStructure
*ns1__NotifyVehicleMonitoring) override;
   /// Web service one-way operation 'NotifyConnectionTimetable' (return SOAP_OK
(no response) or error code, or use
send_NotifyConnectionTimetable_empty_response())
   virtual int
NotifyConnectionTimetable(ns1__WsConnectionTimetableNotificationStructure
*ns1__NotifyConnectionTimetable) override;
   /// Web service one-way operation 'NotifyConnectionMonitoring' (return
SOAP_OK (no response) or error code, or use
send_NotifyConnectionMonitoring_empty_response())
   virtual int
NotifyConnectionMonitoring(ns1__WsConnectionMonitoringNotificationStructure
*ns1__NotifyConnectionMonitoring) override;
   /// Web service one-way operation 'NotifyGeneralMessage' (return SOAP_OK (no
response) or error code, or use send_NotifyGeneralMessage_empty_response())
   virtual int NotifyGeneralMessage(ns1__WsGeneralMessageNotificationStructure
*ns1__NotifyGeneralMessage) override;
   /// Web service one-way operation 'NotifyFacilityMonitoring' (return SOAP_OK
(no response) or error code, or use
send_NotifyFacilityMonitoring_empty_response())
   virtual int
NotifyFacilityMonitoring(ns1__WsFacilityMonitoringNotificationStructure
*ns1__NotifyFacilityMonitoring) override;
   /// Web service one-way operation 'NotifySituationExchange' (return SOAP_OK
(no response) or error code, or use
send_NotifySituationExchange_empty_response())
   virtual int
NotifySituationExchange(ns1__WsSituationExchangeNotificationStructure
*ns1__NotifySituationExchange) override;
};
```

ConsumerService.cpp

```
#include "stdafx.h"
#include "ConsumerService.h"
#include "PrintUtils.hpp"

bool siri_proxy = false;
std::string proxy_host("localhost");
int proxy_port(8888);

ConsumerService::ConsumerService()
{
}

ConsumerService::ConsumerService(struct soap* _soap)
   : SiriConsumerDocBindingService(_soap)
{
   if (siri_proxy == true)
   {
      this->soap->proxy_host = proxy_host.c_str();
      this->soap->proxy_port = proxy_port;
   }
}

ConsumerService::~ConsumerService()
{
}

std::mutex m_lock;

// Implemented
/// Web service one-way operation 'NotifyProductionTimetable' (return SOAP_OK
(no response) or error code, or use
send_NotifyProductionTimetable_empty_response())
int
ConsumerService::NotifyProductionTimetable(ns1__WsProductionTimetableNotificatio
nStructure *ns1__NotifyProductionTimetable)
{
   assert(ns1__NotifyProductionTimetable->Notification != nullptr);

   std::unique_lock<std::mutex> lock(m_lock);
   PrintUtils::show_productiontimetable(*ns1__NotifyProductionTimetable);
   std::cout << boost::posix_time::second_clock::universal_time() << " - PT
update received" << std::endl;

   return SOAP_OK;
}

/// Web service one-way operation 'NotifyEstimatedTimetable' (return SOAP_OK (no
response) or error code, or use send_NotifyEstimatedTimetable_empty_response())
int
ConsumerService::NotifyEstimatedTimetable(ns1__WsEstimatedTimetableNotificationS
tructure *ns1__NotifyEstimatedTimetable)
{
   assert(ns1__NotifyEstimatedTimetable->Notification != nullptr);

   std::unique_lock<std::mutex> lock(m_lock);
   PrintUtils::show_estimatedtimetable(*ns1__NotifyEstimatedTimetable);
   std::cout << boost::posix_time::second_clock::universal_time() << " - ET
update received" << std::endl;

   return SOAP_OK;
}
```

```
/// Web service one-way operation 'NotifyStopTimetable' (return SOAP_OK (no
response) or error code, or use send_NotifyStopTimetable_empty_response())
int
ConsumerService::NotifyStopTimetable(ns1__WsStopTimetableNotificationStructure
*ns1__NotifyStopTimetable)
{
   assert(ns1__NotifyStopTimetable->Notification != nullptr);

   std::unique_lock<std::mutex> lock(m_lock);
   PrintUtils::show_stoptimetable(*ns1__NotifyStopTimetable);
   std::cout << boost::posix_time::second_clock::universal_time() << " - ST
update received" << std::endl;

   return SOAP_OK;
}

/// Web service one-way operation 'NotifyStopMonitoring' (return SOAP_OK (no
response) or error code, or use send_NotifyStopMonitoring_empty_response())
int
ConsumerService::NotifyStopMonitoring(ns1__WsStopMonitoringNotificationStructure
*ns1__NotifyStopMonitoring)
{
   assert(ns1__NotifyStopMonitoring->Notification != nullptr);

   std::unique_lock<std::mutex> lock(m_lock);
   PrintUtils::show_stopmonitoring(*ns1__NotifyStopMonitoring);
   std::cout << boost::posix_time::second_clock::universal_time() << " - SM
update received" << std::endl;

   return SOAP_OK;
}

/// Web service one-way operation 'NotifyVehicleMonitoring' (return SOAP_OK (no
response) or error code, or use send_NotifyVehicleMonitoring_empty_response())
int
ConsumerService::NotifyVehicleMonitoring(ns1__WsVehicleMonitoringNotificationStr
ucture *ns1__NotifyVehicleMonitoring)
{
   assert(ns1__NotifyVehicleMonitoring->Notification != nullptr);

   std::unique_lock<std::mutex> lock(m_lock);
   PrintUtils::show_vehiclemonitoring(*ns1__NotifyVehicleMonitoring);
   std::cout << boost::posix_time::second_clock::universal_time() << " - VM
update received" << std::endl;

   return SOAP_OK;
}
```

```
/// Web service one-way operation 'NotifySubscriptionTerminated' (return SOAP_OK
(no response) or error code, or use
send_NotifySubscriptionTerminated_empty_response())°
int
ConsumerService::NotifySubscriptionTerminated(ns1__WsSubscriptionTerminatedNotif
icationStructure *ns1__NotifySubscriptionTerminated)
{
   assert(ns1__NotifySubscriptionTerminated->Notification != nullptr);

   for (const auto& terminatedSubscription : ns1__NotifySubscriptionTerminated-
>Notification->__SubscriptionTerminatedNotificationStructure_sequence)
   {
      std::wcout << "Subscription " << terminatedSubscription.SubscriptionRef-
>__item << " from " <<
         terminatedSubscription.SubscriberRef->__item << " ended" << std::endl;
   }

   return SOAP_OK;
}

// Not implemented
/// Web service one-way operation 'NotifyDataReady' (return SOAP_OK (no
response) or error code, or use send_NotifyDataReady_empty_response())
int ConsumerService::NotifyDataReady(ns1__WsDataReadyNotificationStructure
*ns1__NotifyDataReady)
{
   return SOAP_OK;
}
/// Web service one-way operation 'NotifyHeartbeat' (return SOAP_OK (no
response) or error code, or use send_NotifyHeartbeat_empty_response())
int ConsumerService::NotifyHeartbeat(ns1__WsHeartbeatNotificationStructure
*ns1__NotifyHeartbeat)
{
   return SOAP_OK;
}
/// Web service one-way operation 'NotifyConnectionTimetable' (return SOAP_OK
(no response) or error code, or use
send_NotifyConnectionTimetable_empty_response())
int
ConsumerService::NotifyConnectionTimetable(ns1__WsConnectionTimetableNotificatio
nStructure *ns1__NotifyConnectionTimetable)
{
   return SOAP_OK;
}
/// Web service one-way operation 'NotifyConnectionMonitoring' (return SOAP_OK
(no response) or error code, or use
send_NotifyConnectionMonitoring_empty_response())
int
ConsumerService::NotifyConnectionMonitoring(ns1__WsConnectionMonitoringNotificat
ionStructure *ns1__NotifyConnectionMonitoring)
{
   return SOAP_OK;
}
```

```
/// Web service one-way operation 'NotifyGeneralMessage' (return SOAP_OK (no
response) or error code, or use send_NotifyGeneralMessage_empty_response())
int
ConsumerService::NotifyGeneralMessage(ns1__WsGeneralMessageNotificationStructure
*ns1__NotifyGeneralMessage)
{
   return SOAP_OK;
}
/// Web service one-way operation 'NotifyFacilityMonitoring' (return SOAP_OK (no
response) or error code, or use send_NotifyFacilityMonitoring_empty_response())
int
ConsumerService::NotifyFacilityMonitoring(ns1__WsFacilityMonitoringNotificationS
tructure *ns1__NotifyFacilityMonitoring)
{
   return SOAP_OK;
}
/// Web service one-way operation 'NotifySituationExchange' (return SOAP_OK (no
response) or error code, or use send_NotifyFacilityMonitoring_empty_response())
int
ConsumerService::NotifySituationExchange(ns1__WsSituationExchangeNotificationStr
ucture *ns1__NotifySituationExchange)
{
   return SOAP_OK;
}
```

PrintUtils.hpp

```
#pragma once

#include "soapSiriConsumerDocBindingService.h"
#include <boost/filesystem.hpp>

class PrintUtils
{
public:
   static void show_vehiclemonitoring(const
ns1__WsVehicleMonitoringNotificationStructure& response)
   {
#ifdef UNICODE
      std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
      std::locale loc(std::locale::classic());
#endif

      _tstring path = _T("c:/temp/siri/sub_vehiclemonitoring.txt");
      const boost::filesystem::path p = path;

      _ofstream ofs(p.string(), std::ios::app);
      ofs.imbue(loc);



      ofs << "\n\n\n**************************\n" << "Vehicle Monitoring
subscription: " << std::endl << std::endl;

      if (response.ServiceDeliveryInfo->ProducerRef)
         ofs << "ProducerRef: " << response.ServiceDeliveryInfo->ProducerRef-
>__item << std::endl;
      if (response.ServiceDeliveryInfo->RequestMessageRef)
         ofs << "RequestMessageRef: " << response.ServiceDeliveryInfo-
>RequestMessageRef->__item << std::endl;
      if (response.ServiceDeliveryInfo->ResponseMessageIdentifier)
         ofs << "ResponseMessageIdentifier: " << response.ServiceDeliveryInfo-
>ResponseMessageIdentifier->__item << std::endl;
      if (response.ServiceDeliveryInfo->Address)
         ofs << "Address: " << response.ServiceDeliveryInfo->Address <<
std::endl;

      unsigned cont = 0;
      for (const auto& sub : response.Notification->VehicleMonitoringDelivery)
      {
         ofs << "sub #" << ++cont << ",  ";
         ofs << "valid until: " << GSOAP::to_ptime(*sub->ValidUntil) <<
std::endl;
         ofs << "SubscriptionRef:\t" << sub->SubscriptionRef->__item <<
std::endl;
         ofs << "SubscriberRef:\t" << sub->SubscriberRef->__item << std::endl;

         for (const auto& activity : sub->VehicleActivity)
         {
            ofs << std::endl;
            ofs << "LineRef: " << activity->MonitoredVehicleJourney.LineRef-
>__item << ", ";
            ofs << "DirectionRef: " << activity-
>MonitoredVehicleJourney.DirectionRef->__item << ", ";
            ofs << "VehicleJourneyName: " << activity-
>MonitoredVehicleJourney.VehicleJourneyName.front()->__item << std::endl;
```

```
            ofs << "Long. tramo: " << *activity->ProgressBetweenStops-
>LinkDistance << " m" << std::endl;
            ofs << "% Recorrido: " << *activity->ProgressBetweenStops-
>Percentage << " %" << std::endl;
            ofs << "Service ID: " << *activity->ItemIdentifier << " " <<
activity->VehicleMonitoringRef->__item << std::endl;
            boost::posix_time::time_duration delay =
boost::posix_time::milliseconds(*activity->MonitoredVehicleJourney.Delay);
            ofs << "Delay: " << delay << std::endl;
            ofs << "Speed: " << *activity->MonitoredVehicleJourney.Velocity << "
km/h - ";
            ofs << "Bearing: " << *activity->MonitoredVehicleJourney.Bearing <<
" degrees" << std::endl;
            for (const auto& future_stop : activity-
>MonitoredVehicleJourney.OnwardCalls->OnwardCall)
            {
               ofs << *future_stop->Order << ". ";
               ofs << future_stop->StopPointName.front()->__item << " - ";
               if (activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__&& activity-
>MonitoredVehicleJourney.__VehicleActivityStructure_MonitoredVehicleJourney_sequ
ence__->HeadwayService == true)
               {
                  ofs << future_stop->StopPointRef->__item << " - frequency: ";
                  ofs << *future_stop->AimedHeadwayInterval / (1000 * 60) << "
minutes" << std::endl;
               }
               else
               {
                  ofs << future_stop->StopPointRef->__item << " -  from: ";
                  ofs << GSOAP::to_ptime(*future_stop->AimedArrivalTime) << "
to: ";
                  ofs << GSOAP::to_ptime(*future_stop->AimedDepartureTime) <<
std::endl;
               }
            }
         }

         for (const auto& stopcandellation : sub->VehicleActivityCancellation)
         {

         }
         ofs << "\nResponse time: " << GSOAP::to_ptime(sub->ResponseTimestamp);
      }
   }
```

```
   static void show_stopmonitoring(const
ns1__WsStopMonitoringNotificationStructure& response)
   {
#ifdef UNICODE
      std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
      std::locale loc(std::locale::classic());
#endif

      _tstring path = _T("c:/temp/siri/sub_stopmonitoring.txt");
      const boost::filesystem::path p = path;

      _ofstream ofs(p.string(), std::ios::app);
      ofs.imbue(loc);


      ofs << "\n\n\n*****************************\n" << "Stop Monitoring
subscription: " << std::endl << std::endl;

      if (response.ServiceDeliveryInfo->ProducerRef)
         ofs << "ProducerRef: " << response.ServiceDeliveryInfo->ProducerRef-
>__item << std::endl;
      if (response.ServiceDeliveryInfo->RequestMessageRef)
         ofs << "RequestMessageRef: " << response.ServiceDeliveryInfo-
>RequestMessageRef->__item << std::endl;
      if (response.ServiceDeliveryInfo->ResponseMessageIdentifier)
         ofs << "ResponseMessageIdentifier: " << response.ServiceDeliveryInfo-
>ResponseMessageIdentifier->__item << std::endl;
      if (response.ServiceDeliveryInfo->Address)
         ofs << "Address: " << response.ServiceDeliveryInfo->Address <<
std::endl;

      unsigned cont = 0;
      for (const auto& sub : response.Notification->StopMonitoringDelivery)
      {
         ofs << "sub #" << ++cont << ",  ";
         ofs << "valid until: " << GSOAP::to_ptime(*sub->ValidUntil) <<
std::endl;
         ofs << "SubscriptionRef:\t" << sub->SubscriptionRef->__item <<
std::endl;
         ofs << "SubscriberRef:\t" << sub->SubscriberRef->__item << std::endl;
         for (const auto& stopvisit : sub->MonitoredStopVisit)
         {
            ofs << std::endl;
            ofs << "MonitoringRef: " << stopvisit-
>__MonitoredStopVisitStructure_sequence->MonitoringRef->__item << ", ";
            ofs << "LineRef: " << stopvisit->MonitoredVehicleJourney->LineRef-
>__item << ", ";
            ofs << "DirectionRef: " << stopvisit->MonitoredVehicleJourney-
>DirectionRef->__item << ", ";
            ofs << "VehicleJourneyName: " << stopvisit->MonitoredVehicleJourney-
>VehicleJourneyName.front()->__item << std::endl;

            ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall->Order <<
". ";
            ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointName.front()->__item << " - ";
```

```
            if (stopvisit->MonitoredVehicleJourney-
>__MonitoredVehicleJourneyStructure_sequence__ && stopvisit-
>MonitoredVehicleJourney->__MonitoredVehicleJourneyStructure_sequence__-
>HeadwayService == true)
            {
               ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " - frequency: ";
               ofs << *stopvisit->MonitoredVehicleJourney->MonitoredCall-
>AimedHeadwayInterval / (1000 * 60) << " minutes" << std::endl;
            }
            else
            {
               ofs << stopvisit->MonitoredVehicleJourney->MonitoredCall-
>StopPointRef->__item << " -  from: ";
               ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedArrivalTime) << " to: ";
               ofs << GSOAP::to_ptime(*stopvisit->MonitoredVehicleJourney-
>MonitoredCall->AimedDepartureTime) << std::endl;
            }

            boost::posix_time::time_duration delay =
boost::posix_time::milliseconds(*stopvisit->MonitoredVehicleJourney->Delay);
            ofs << "delay: " << delay << std::endl;
         }

         for (const auto& stopcancellation : sub-
>MonitoredStopVisitCancellation)
         {

         }
         ofs << "\nResponse time: " << GSOAP::to_ptime(sub->ResponseTimestamp);
      }
   }
```

```
   static void show_stoptimetable(const
ns1__WsStopTimetableNotificationStructure& response)
   {
#ifdef UNICODE
      std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
      std::locale loc(std::locale::classic());
#endif

      _tstring path = _T("c:/temp/siri/sub_stoptimetable.txt");
      const boost::filesystem::path p = path;

      _ofstream ofs(p.string(), std::ios::app);
      ofs.imbue(loc);


      ofs << "\n\n\n*****************************\n" << "Stop Timetable
subscription: " << std::endl << std::endl;

      ofs << "Valid until: " << GSOAP::to_ptime(*response.Notification-
>StopTimetableDelivery->ValidUntil) << std::endl;
      for (const auto& stopvisit : response.Notification->StopTimetableDelivery-
>TimetabledStopVisit)
      {
         ofs << std::endl;
         ofs << "MonitoringRef: " << stopvisit->MonitoringRef->__item << ", ";
         ofs << "LineRef: " << stopvisit->TargetedVehicleJourney->LineRef-
>__item << ", ";
         ofs << "DirectionRef: " << stopvisit->TargetedVehicleJourney-
>DirectionRef->__item << std::endl;

         ofs << *stopvisit->TargetedVehicleJourney->TargetedCall->Order << ". ";

         if (stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__ &&
            true == stopvisit->TargetedVehicleJourney-
>__TargetedVehicleJourneyStructure_sequence__->HeadwayService)
         {
            ofs << stopvisit->TargetedVehicleJourney->TargetedCall-
>StopPointRef->__item << " - frequency: ";
            ofs << *stopvisit->TargetedVehicleJourney->TargetedCall-
>AimedHeadwayInterval / (1000 * 60) << " minutes" << std::endl;
         }
         else
         {
            ofs << stopvisit->TargetedVehicleJourney->TargetedCall-
>StopPointRef->__item << " -  from: ";
            ofs << GSOAP::to_ptime(*stopvisit->TargetedVehicleJourney-
>TargetedCall->AimedArrivalTime) << " to: ";
            ofs << GSOAP::to_ptime(*stopvisit->TargetedVehicleJourney-
>TargetedCall->AimedDepartureTime) << std::endl;
         }
      }

      for (const auto& stopcancellation : response.Notification-
>StopTimetableDelivery->TimetabledStopVisitCancellation)
      {

      }

      ofs << "\nResponse time: " << GSOAP::to_ptime(response.Notification-
>StopTimetableDelivery->ResponseTimestamp);
   }
```

```
   static void show_productiontimetable(const
ns1__WsProductionTimetableNotificationStructure& response)
   {
#ifdef UNICODE
      std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
      std::locale loc(std::locale::classic());
#endif

      _tstring path = _T("c:/temp/siri/sub_productiontimetable.txt");
      const boost::filesystem::path p = path;

      _ofstream ofs(p.string(), std::ios::app);
      ofs.imbue(loc);


      ofs << "\n\n\n*****************************\n" << "Production Timetable
subscription: " << std::endl << std::endl;

      int cont_line = 0;
      for (const auto& line : response.Notification-
>ProductionTimetableDelivery)
      {
         ofs << "Line number " << ++cont_line;
         ofs << "\t(valid until: " << GSOAP::to_ptime(*line->ValidUntil) << ")"
<< std::endl;
         for (const auto& trajectory : line->DatedTimetableVersionFrame)
         {
            ofs << "\nL" << trajectory->LineRef->__item << ", ";
            ofs << "Trajectory: " << trajectory->RouteRef->__item << ", ";
            ofs << "Direction: " << trajectory->DirectionRef->__item <<
std::endl;

            for (const auto& expedition : trajectory->DatedVehicleJourney)
            {
               ofs << "\nExpedition code: " << expedition-
>DatedVehicleJourneyCode << " - ";
               ofs << "(road to: " << expedition->DestinationDisplay[0]->__item
<< ")" << std::endl;

               for (const auto& stop : expedition->DatedCalls.DatedCall)
               {
                  if (expedition->HeadwayService && *expedition->HeadwayService
== true)
                  {
                     ofs << stop->StopPointRef->__item << " - frequency: ";
                     ofs << *stop->AimedHeadwayInterval / (1000 * 60) << "
minutes" << std::endl;
                  }
                  else
                  {
                     ofs << stop->StopPointRef->__item << " -  from: ";
                     ofs << GSOAP::to_ptime(*stop->AimedArrivalTime) << " to: ";
                     ofs << GSOAP::to_ptime(*stop->AimedDepartureTime) <<
std::endl;
                  }
               }
            }
         }
      }

      ofs << "\nResponse time: " <<
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp) << std::endl;
   }
```

```
   static void show_estimatedtimetable(const
ns1__WsEstimatedTimetableNotificationStructure& response)
   {
#ifdef UNICODE
      std::locale loc(std::locale::classic(), new std::codecvt_utf8<wchar_t>);
#else
      std::locale loc(std::locale::classic());
#endif

      _tstring path = _T("c:/temp/siri/sub_estimatedtimetable.txt");
      const boost::filesystem::path p = path;

      _ofstream ofs(p.string(), std::ios::app);
      ofs.imbue(loc);


      ofs << "\n\n\n*****************************\n" << "Estimated Timetable
subscription: " << std::endl << std::endl;

      int cont_line = 0;
      for (const auto& line : response.Notification->EstimatedTimetableDelivery)
      {
         ofs << "Line number " << ++cont_line;
         ofs << "\t(valid until: " << GSOAP::to_ptime(*line->ValidUntil) << ")"
<< std::endl;
         for (const auto& trajectory : line->EstimatedJourneyVersionFrame)
         {
            for (const auto& expedition : trajectory->EstimatedVehicleJourney)
            {
               ofs << "\nExpedition code: " << expedition->VehicleJourneyRef-
>__item << " - " << std::endl;
               ofs << "From: " << expedition->OriginName.front()->__item << "("
<< expedition->OriginRef->__item << ")" << std::endl;
               ofs << "To: " << expedition->DestinationName.front()->__item <<
"(" << expedition->DestinationRef->__item << ")" << std::endl;

               for (const auto& stop : expedition->EstimatedCalls-
>EstimatedCall)
               {
                  if (expedition-
>__EstimatedVehicleJourneyStructure_sequence___->HeadwayService && expedition-
>__EstimatedVehicleJourneyStructure_sequence___->HeadwayService == true)
                  {
                     ofs << stop->StopPointRef->__item << " - frequency: ";
                     ofs << *stop->AimedHeadwayInterval / (1000 * 60) << "
minutes" << std::endl;
                  }
                  else
                  {
                     ofs << stop->StopPointRef->__item << " -  from: ";
                     ofs << GSOAP::to_ptime(*stop->AimedArrivalTime) << " to: ";
                     ofs << GSOAP::to_ptime(*stop->AimedDepartureTime) <<
std::endl;
                  }
               }
            }
         }
      }

      ofs << "\nResponse time: " <<
GSOAP::to_ptime(response.ServiceDeliveryInfo->ResponseTimestamp) << std::endl;
   }
};
```

soap_srv.nsmap

```
#include "soapH.h"
SOAP_NMAC struct Namespace srv_namespaces[] = {
   {"SOAP-ENV", "http://schemas.xmlsoap.org/soap/envelope/",
"http://www.w3.org/*/soap-envelope", NULL},
   {"SOAP-ENC", "http://schemas.xmlsoap.org/soap/encoding/",
"http://www.w3.org/*/soap-encoding", NULL},
   {"xsi", "http://www.w3.org/2001/XMLSchema-instance",
"http://www.w3.org/*/XMLSchema-instance", NULL},
   {"xsd", "http://www.w3.org/2001/XMLSchema", "http://www.w3.org/*/XMLSchema",
NULL},
   {"ns4", "http://www.ifopt.org.uk/acsb", NULL, NULL},
   {"ns5", "http://datex2.eu/schema/2_0RC1/2_0", NULL, NULL},
   {"ns2", "http://www.siri.org.uk/siri", NULL, NULL},
   {"ns3", "http://www.ifopt.org.uk/ifopt", NULL, NULL},
   {"ns1", "http://wsdl.siri.org.uk", NULL, NULL},
   {NULL, NULL, NULL, NULL}
    };
```

Anexo 3: Trazas de datos resultado de la ejecución de los tests del cliente

Fuente de los datos: Tranvías de Sydney (Australia)
Simulación (en diferido) de la operativa de diciembre de 2016

Production Timetable (PT)

```
Production Timetable:

Line number 1 (valid until: 2016-Dec-25 08:00:00)

L1, Trajectory: 3, Direction: CIRCULAR QUAY [2]

Expedition code: 549 - from: RANDWICK [2] to: CIRCULAR QUAY [2]
10201 -  from: 2016-Dec-16 00:00:00 to: 2016-Dec-16 00:00:00
10203 -  from: 2016-Dec-16 00:02:28 to: 2016-Dec-16 00:03:00
10204 -  from: 2016-Dec-16 00:05:28 to: 2016-Dec-16 00:06:00
10206 -  from: 2016-Dec-16 00:08:28 to: 2016-Dec-16 00:09:00
10219 -  from: 2016-Dec-16 00:19:28 to: 2016-Dec-16 00:20:00
10220 -  from: 2016-Dec-16 00:24:28 to: 2016-Dec-16 00:25:00
10222 -  from: 2016-Dec-16 00:28:28 to: 2016-Dec-16 00:29:00
10226 -  from: 2016-Dec-16 00:31:28 to: 2016-Dec-16 00:32:00
10229 -  from: 2016-Dec-16 00:35:28 to: 2016-Dec-16 00:36:00
10230 -  from: 2016-Dec-16 00:40:28 to: 2016-Dec-16 00:41:00
10233 -  from: 2016-Dec-16 00:44:28 to: 2016-Dec-16 00:45:00
10235 -  from: 2016-Dec-16 00:47:28 to: 2016-Dec-16 00:48:00
10237 -  from: 2016-Dec-16 00:50:28 to: 2016-Dec-16 00:51:00
10238 -  from: 2016-Dec-16 00:52:00 to: 2016-Dec-16 00:52:00

Expedition code: 573 - from: RANDWICK [2] to: CIRCULAR QUAY [2]
10201 -  from: 2016-Dec-16 00:04:00 to: 2016-Dec-16 00:04:00
10203 -  from: 2016-Dec-16 00:06:28 to: 2016-Dec-16 00:07:00
10204 -  from: 2016-Dec-16 00:09:28 to: 2016-Dec-16 00:10:00
10206 -  from: 2016-Dec-16 00:12:28 to: 2016-Dec-16 00:13:00
10219 -  from: 2016-Dec-16 00:23:28 to: 2016-Dec-16 00:24:00
10220 -  from: 2016-Dec-16 00:28:28 to: 2016-Dec-16 00:29:00
10222 -  from: 2016-Dec-16 00:32:28 to: 2016-Dec-16 00:33:00
10226 -  from: 2016-Dec-16 00:35:28 to: 2016-Dec-16 00:36:00
10229 -  from: 2016-Dec-16 00:39:28 to: 2016-Dec-16 00:40:00
10230 -  from: 2016-Dec-16 00:44:28 to: 2016-Dec-16 00:45:00
10233 -  from: 2016-Dec-16 00:48:28 to: 2016-Dec-16 00:49:00
10235 -  from: 2016-Dec-16 00:51:28 to: 2016-Dec-16 00:52:00
10237 -  from: 2016-Dec-16 00:54:28 to: 2016-Dec-16 00:55:00
10238 -  from: 2016-Dec-16 00:56:00 to: 2016-Dec-16 00:56:00

(…)
```

```
L1, Trajectory: 4, Direction: CIRCULAR QUAY [3]

Expedition code: 597 - from: KINGSFORD [2] to: CIRCULAR QUAY [3]
10209 -  from: 2016-Dec-16 00:00:00 to: 2016-Dec-16 00:00:00
10211 -  from: 2016-Dec-16 00:02:28 to: 2016-Dec-16 00:03:00
10213 -  from: 2016-Dec-16 00:05:28 to: 2016-Dec-16 00:06:00
10215 -  from: 2016-Dec-16 00:08:28 to: 2016-Dec-16 00:09:00
10217 -  from: 2016-Dec-16 00:11:28 to: 2016-Dec-16 00:12:00
10219 -  from: 2016-Dec-16 00:14:28 to: 2016-Dec-16 00:15:00
10220 -  from: 2016-Dec-16 00:17:28 to: 2016-Dec-16 00:18:00
10222 -  from: 2016-Dec-16 00:20:28 to: 2016-Dec-16 00:21:00
10226 -  from: 2016-Dec-16 00:23:28 to: 2016-Dec-16 00:24:00
10229 -  from: 2016-Dec-16 00:26:28 to: 2016-Dec-16 00:27:00
10230 -  from: 2016-Dec-16 00:29:28 to: 2016-Dec-16 00:30:00
10233 -  from: 2016-Dec-16 00:32:28 to: 2016-Dec-16 00:33:00
10235 -  from: 2016-Dec-16 00:35:28 to: 2016-Dec-16 00:36:00
10237 -  from: 2016-Dec-16 00:38:28 to: 2016-Dec-16 00:39:00
10239 -  from: 2016-Dec-16 00:42:00 to: 2016-Dec-16 00:42:00


Expedition code: 599 - from: KINGSFORD [2] to: CIRCULAR QUAY [3]
10209 -  from: 2016-Dec-16 02:00:00 to: 2016-Dec-16 02:00:00
10211 -  from: 2016-Dec-16 02:02:28 to: 2016-Dec-16 02:03:00
10213 -  from: 2016-Dec-16 02:05:28 to: 2016-Dec-16 02:06:00
10215 -  from: 2016-Dec-16 02:08:28 to: 2016-Dec-16 02:09:00
10217 -  from: 2016-Dec-16 02:11:28 to: 2016-Dec-16 02:12:00
10219 -  from: 2016-Dec-16 02:14:28 to: 2016-Dec-16 02:15:00
10220 -  from: 2016-Dec-16 02:17:28 to: 2016-Dec-16 02:18:00
10222 -  from: 2016-Dec-16 02:20:28 to: 2016-Dec-16 02:21:00
10226 -  from: 2016-Dec-16 02:23:28 to: 2016-Dec-16 02:24:00
10229 -  from: 2016-Dec-16 02:26:28 to: 2016-Dec-16 02:27:00
10230 -  from: 2016-Dec-16 02:29:28 to: 2016-Dec-16 02:30:00
10233 -  from: 2016-Dec-16 02:32:28 to: 2016-Dec-16 02:33:00
10235 -  from: 2016-Dec-16 02:35:28 to: 2016-Dec-16 02:36:00
10237 -  from: 2016-Dec-16 02:38:28 to: 2016-Dec-16 02:39:00
10239 -  from: 2016-Dec-16 02:42:00 to: 2016-Dec-16 02:42:00

(…)


L1, Trajectory: 1, Direction: RANDWICK [1]

Expedition code: 548 - from: CIRCULAR QUAY [2] to: RANDWICK [1]
10238 -  from: 2016-Dec-15 23:00:00 to: 2016-Dec-15 23:00:00
10236 -  from: 2016-Dec-15 23:02:28 to: 2016-Dec-15 23:03:00
10234 -  from: 2016-Dec-15 23:05:28 to: 2016-Dec-15 23:06:00
10232 -  from: 2016-Dec-15 23:09:28 to: 2016-Dec-15 23:10:00
10231 -  from: 2016-Dec-15 23:12:28 to: 2016-Dec-15 23:13:00
10228 -  from: 2016-Dec-15 23:17:28 to: 2016-Dec-15 23:18:00
10225 -  from: 2016-Dec-15 23:21:28 to: 2016-Dec-15 23:22:00
10223 -  from: 2016-Dec-15 23:24:28 to: 2016-Dec-15 23:25:00
10221 -  from: 2016-Dec-15 23:28:28 to: 2016-Dec-15 23:29:00
10218 -  from: 2016-Dec-15 23:33:28 to: 2016-Dec-15 23:34:00
10207 -  from: 2016-Dec-15 23:37:28 to: 2016-Dec-15 23:38:00
10205 -  from: 2016-Dec-15 23:41:28 to: 2016-Dec-15 23:42:00
10241 -  from: 2016-Dec-15 23:44:28 to: 2016-Dec-15 23:45:00
10202 -  from: 2016-Dec-15 23:48:00 to: 2016-Dec-15 23:48:00
```

```
Expedition code: 572 - from: CIRCULAR QUAY [2] to: RANDWICK [1]
10238 -  from: 2016-Dec-15 23:04:00 to: 2016-Dec-15 23:04:00
10236 -  from: 2016-Dec-15 23:06:28 to: 2016-Dec-15 23:07:00
10234 -  from: 2016-Dec-15 23:09:28 to: 2016-Dec-15 23:10:00
10232 -  from: 2016-Dec-15 23:13:28 to: 2016-Dec-15 23:14:00
10231 -  from: 2016-Dec-15 23:16:28 to: 2016-Dec-15 23:17:00
10228 -  from: 2016-Dec-15 23:21:28 to: 2016-Dec-15 23:22:00
10225 -  from: 2016-Dec-15 23:25:28 to: 2016-Dec-15 23:26:00
10223 -  from: 2016-Dec-15 23:28:28 to: 2016-Dec-15 23:29:00
10221 -  from: 2016-Dec-15 23:32:28 to: 2016-Dec-15 23:33:00
10218 -  from: 2016-Dec-15 23:37:28 to: 2016-Dec-15 23:38:00
10207 -  from: 2016-Dec-15 23:41:28 to: 2016-Dec-15 23:42:00
10205 -  from: 2016-Dec-15 23:45:28 to: 2016-Dec-15 23:46:00
10241 -  from: 2016-Dec-15 23:48:28 to: 2016-Dec-15 23:49:00
10202 -  from: 2016-Dec-15 23:52:00 to: 2016-Dec-15 23:52:00

(…)


L1, Trajectory: 2, Direction: KINGSFORD [1]

Expedition code: 596 - from: CIRCULAR QUAY [3] to: KINGSFORD [1]
10239 -  from: 2016-Dec-15 23:00:00 to: 2016-Dec-15 23:00:00
10236 -  from: 2016-Dec-15 23:02:28 to: 2016-Dec-15 23:03:00
10234 -  from: 2016-Dec-15 23:05:28 to: 2016-Dec-15 23:06:00
10232 -  from: 2016-Dec-15 23:08:28 to: 2016-Dec-15 23:09:00
10231 -  from: 2016-Dec-15 23:11:28 to: 2016-Dec-15 23:12:00
10228 -  from: 2016-Dec-15 23:14:28 to: 2016-Dec-15 23:15:00
10225 -  from: 2016-Dec-15 23:17:28 to: 2016-Dec-15 23:18:00
10223 -  from: 2016-Dec-15 23:20:28 to: 2016-Dec-15 23:21:00
10221 -  from: 2016-Dec-15 23:23:28 to: 2016-Dec-15 23:24:00
10218 -  from: 2016-Dec-15 23:26:28 to: 2016-Dec-15 23:27:00
10216 -  from: 2016-Dec-15 23:29:28 to: 2016-Dec-15 23:30:00
10214 -  from: 2016-Dec-15 23:32:28 to: 2016-Dec-15 23:33:00
10212 -  from: 2016-Dec-15 23:35:28 to: 2016-Dec-15 23:36:00
10210 -  from: 2016-Dec-15 23:38:28 to: 2016-Dec-15 23:39:00
10208 -  from: 2016-Dec-15 23:42:00 to: 2016-Dec-15 23:42:00

Expedition code: 598 - from: CIRCULAR QUAY [3] to: KINGSFORD [1]
10239 -  from: 2016-Dec-16 01:00:00 to: 2016-Dec-16 01:00:00
10236 -  from: 2016-Dec-16 01:02:28 to: 2016-Dec-16 01:03:00
10234 -  from: 2016-Dec-16 01:05:28 to: 2016-Dec-16 01:06:00
10232 -  from: 2016-Dec-16 01:08:28 to: 2016-Dec-16 01:09:00
10231 -  from: 2016-Dec-16 01:11:28 to: 2016-Dec-16 01:12:00
10228 -  from: 2016-Dec-16 01:14:28 to: 2016-Dec-16 01:15:00
10225 -  from: 2016-Dec-16 01:17:28 to: 2016-Dec-16 01:18:00
10223 -  from: 2016-Dec-16 01:20:28 to: 2016-Dec-16 01:21:00
10221 -  from: 2016-Dec-16 01:23:28 to: 2016-Dec-16 01:24:00
10218 -  from: 2016-Dec-16 01:26:28 to: 2016-Dec-16 01:27:00
10216 -  from: 2016-Dec-16 01:29:28 to: 2016-Dec-16 01:30:00
10214 -  from: 2016-Dec-16 01:32:28 to: 2016-Dec-16 01:33:00
10212 -  from: 2016-Dec-16 01:35:28 to: 2016-Dec-16 01:36:00
10210 -  from: 2016-Dec-16 01:38:28 to: 2016-Dec-16 01:39:00
10208 -  from: 2016-Dec-16 01:42:00 to: 2016-Dec-16 01:42:00

(…)
```

```
L1, Trajectory: 1001, Direction: RUqaCv

Expedition code: sKzhfO - from: RANDWICK [2] to: CIRCULAR QUAY [1]
10201 - frequency: 1 minutes
10203 - frequency: 1 minutes
10204 - frequency: 1 minutes
10206 - frequency: 1 minutes
10219 - frequency: 1 minutes
10220 - frequency: 1 minutes
10222 - frequency: 1 minutes
10226 - frequency: 1 minutes
10229 - frequency: 1 minutes
10230 - frequency: 1 minutes
10233 - frequency: 1 minutes
10235 - frequency: 1 minutes
10237 - frequency: 1 minutes
10240 - frequency: 1 minutes

Expedition code: Goxq8H - from: RANDWICK [2] to: CIRCULAR QUAY [1]
10201 - frequency: 1 minutes
10203 - frequency: 1 minutes
10204 - frequency: 1 minutes
10206 - frequency: 1 minutes
10219 - frequency: 1 minutes
10220 - frequency: 1 minutes
10222 - frequency: 1 minutes
10226 - frequency: 1 minutes
10229 - frequency: 1 minutes
10230 - frequency: 1 minutes
10233 - frequency: 1 minutes
10235 - frequency: 1 minutes
10237 - frequency: 1 minutes
10240 - frequency: 1 minutes

(…)


Response time: 2017-Aug-10 09:16:47
```

Estimated Timetable (ET)

```
Estimated Timetable:

Line number 1 (valid until: 2016-Dec-16 07:09:45)

L1, Direction: CIRCULAR QUAY [2]

Expedition code: 627 - from: RANDWICK [2](10201) to: CIRCULAR QUAY [2](10238)
10201 -  from: 2016-Dec-15 22:12:00 to: 2016-Dec-15 22:12:00
10203 -  from: 2016-Dec-15 22:14:28 to: 2016-Dec-15 22:15:00
10204 -  from: 2016-Dec-15 22:17:28 to: 2016-Dec-15 22:18:00
10206 -  from: 2016-Dec-15 22:20:28 to: 2016-Dec-15 22:21:00
10219 -  from: 2016-Dec-15 22:31:28 to: 2016-Dec-15 22:32:00
10220 -  from: 2016-Dec-15 22:36:28 to: 2016-Dec-15 22:37:00
10222 -  from: 2016-Dec-15 22:40:28 to: 2016-Dec-15 22:41:00
10226 -  from: 2016-Dec-15 22:43:28 to: 2016-Dec-15 22:44:00
10229 -  from: 2016-Dec-15 22:47:28 to: 2016-Dec-15 22:48:00
10230 -  from: 2016-Dec-15 22:52:28 to: 2016-Dec-15 22:53:00
10233 -  from: 2016-Dec-15 22:56:28 to: 2016-Dec-15 22:57:00
10235 -  from: 2016-Dec-15 22:59:28 to: 2016-Dec-15 23:00:00
10237 -  from: 2016-Dec-15 23:02:28 to: 2016-Dec-15 23:03:00
10238 -  from: 2016-Dec-15 23:04:00 to: 2016-Dec-15 23:04:00

Expedition code: 651 - from: RANDWICK [2](10201) to: CIRCULAR QUAY [2](10238)
10201 -  from: 2016-Dec-15 22:16:00 to: 2016-Dec-15 22:16:00
10203 -  from: 2016-Dec-15 22:18:28 to: 2016-Dec-15 22:19:00
10204 -  from: 2016-Dec-15 22:21:28 to: 2016-Dec-15 22:22:00
10206 -  from: 2016-Dec-15 22:24:28 to: 2016-Dec-15 22:25:00
10219 -  from: 2016-Dec-15 22:35:28 to: 2016-Dec-15 22:36:00
10220 -  from: 2016-Dec-15 22:40:28 to: 2016-Dec-15 22:41:00
10222 -  from: 2016-Dec-15 22:44:28 to: 2016-Dec-15 22:45:00
10226 -  from: 2016-Dec-15 22:47:28 to: 2016-Dec-15 22:48:00
10229 -  from: 2016-Dec-15 22:51:28 to: 2016-Dec-15 22:52:00
10230 -  from: 2016-Dec-15 22:56:28 to: 2016-Dec-15 22:57:00
10233 -  from: 2016-Dec-15 23:00:28 to: 2016-Dec-15 23:01:00
10235 -  from: 2016-Dec-15 23:03:28 to: 2016-Dec-15 23:04:00
10237 -  from: 2016-Dec-15 23:06:28 to: 2016-Dec-15 23:07:00
10238 -  from: 2016-Dec-15 23:08:00 to: 2016-Dec-15 23:08:00

(…)


L1, Direction: CIRCULAR QUAY [3]

Expedition code: 597 - from: KINGSFORD [2](10209) to: CIRCULAR QUAY [3](10239)
10209 -  from: 2016-Dec-16 00:00:00 to: 2016-Dec-16 00:00:00
10211 -  from: 2016-Dec-16 00:02:28 to: 2016-Dec-16 00:03:00
10213 -  from: 2016-Dec-16 00:05:28 to: 2016-Dec-16 00:06:00
10215 -  from: 2016-Dec-16 00:08:28 to: 2016-Dec-16 00:09:00
10217 -  from: 2016-Dec-16 00:11:28 to: 2016-Dec-16 00:12:00
10219 -  from: 2016-Dec-16 00:14:28 to: 2016-Dec-16 00:15:00
10220 -  from: 2016-Dec-16 00:17:28 to: 2016-Dec-16 00:18:00
10222 -  from: 2016-Dec-16 00:20:28 to: 2016-Dec-16 00:21:00
10226 -  from: 2016-Dec-16 00:23:28 to: 2016-Dec-16 00:24:00
10229 -  from: 2016-Dec-16 00:26:28 to: 2016-Dec-16 00:27:00
10230 -  from: 2016-Dec-16 00:29:28 to: 2016-Dec-16 00:30:00
10233 -  from: 2016-Dec-16 00:32:28 to: 2016-Dec-16 00:33:00
10235 -  from: 2016-Dec-16 00:35:28 to: 2016-Dec-16 00:36:00
10237 -  from: 2016-Dec-16 00:38:28 to: 2016-Dec-16 00:39:00
10239 -  from: 2016-Dec-16 00:42:00 to: 2016-Dec-16 00:42:00
```

Expedition code: 599 - from: KINGSFORD [2](10209) to: CIRCULAR QUAY [3](10239)
10209 - from: 2016-Dec-16 02:00:00 to: 2016-Dec-16 02:00:00
10211 - from: 2016-Dec-16 02:02:28 to: 2016-Dec-16 02:03:00
10213 - from: 2016-Dec-16 02:05:28 to: 2016-Dec-16 02:06:00
10215 - from: 2016-Dec-16 02:08:28 to: 2016-Dec-16 02:09:00
10217 - from: 2016-Dec-16 02:11:28 to: 2016-Dec-16 02:12:00
10219 - from: 2016-Dec-16 02:14:28 to: 2016-Dec-16 02:15:00
10220 - from: 2016-Dec-16 02:17:28 to: 2016-Dec-16 02:18:00
10222 - from: 2016-Dec-16 02:20:28 to: 2016-Dec-16 02:21:00
10226 - from: 2016-Dec-16 02:23:28 to: 2016-Dec-16 02:24:00
10229 - from: 2016-Dec-16 02:26:28 to: 2016-Dec-16 02:27:00
10230 - from: 2016-Dec-16 02:29:28 to: 2016-Dec-16 02:30:00
10233 - from: 2016-Dec-16 02:32:28 to: 2016-Dec-16 02:33:00
10235 - from: 2016-Dec-16 02:35:28 to: 2016-Dec-16 02:36:00
10237 - from: 2016-Dec-16 02:38:28 to: 2016-Dec-16 02:39:00
10239 - from: 2016-Dec-16 02:42:00 to: 2016-Dec-16 02:42:00

(…)

L1, Direction: RANDWICK [1]

Expedition code: 548 - from: CIRCULAR QUAY [2](10238) to: RANDWICK [1](10202)
10238 - from: 2016-Dec-15 23:00:00 to: 2016-Dec-15 23:00:00
10236 - from: 2016-Dec-15 23:02:28 to: 2016-Dec-15 23:03:00
10234 - from: 2016-Dec-15 23:05:28 to: 2016-Dec-15 23:06:00
10232 - from: 2016-Dec-15 23:09:28 to: 2016-Dec-15 23:10:00
10231 - from: 2016-Dec-15 23:12:28 to: 2016-Dec-15 23:13:00
10228 - from: 2016-Dec-15 23:17:28 to: 2016-Dec-15 23:18:00
10225 - from: 2016-Dec-15 23:21:28 to: 2016-Dec-15 23:22:00
10223 - from: 2016-Dec-15 23:24:28 to: 2016-Dec-15 23:25:00
10221 - from: 2016-Dec-15 23:28:28 to: 2016-Dec-15 23:29:00
10218 - from: 2016-Dec-15 23:33:28 to: 2016-Dec-15 23:34:00
10207 - from: 2016-Dec-15 23:37:28 to: 2016-Dec-15 23:38:00
10205 - from: 2016-Dec-15 23:41:28 to: 2016-Dec-15 23:42:00
10241 - from: 2016-Dec-15 23:44:28 to: 2016-Dec-15 23:45:00
10202 - from: 2016-Dec-15 23:48:00 to: 2016-Dec-15 23:48:00

Expedition code: 572 - from: CIRCULAR QUAY [2](10238) to: RANDWICK [1](10202)
10238 - from: 2016-Dec-15 23:04:00 to: 2016-Dec-15 23:04:00
10236 - from: 2016-Dec-15 23:06:28 to: 2016-Dec-15 23:07:00
10234 - from: 2016-Dec-15 23:09:28 to: 2016-Dec-15 23:10:00
10232 - from: 2016-Dec-15 23:13:28 to: 2016-Dec-15 23:14:00
10231 - from: 2016-Dec-15 23:16:28 to: 2016-Dec-15 23:17:00
10228 - from: 2016-Dec-15 23:21:28 to: 2016-Dec-15 23:22:00
10225 - from: 2016-Dec-15 23:25:28 to: 2016-Dec-15 23:26:00
10223 - from: 2016-Dec-15 23:28:28 to: 2016-Dec-15 23:29:00
10221 - from: 2016-Dec-15 23:32:28 to: 2016-Dec-15 23:33:00
10218 - from: 2016-Dec-15 23:37:28 to: 2016-Dec-15 23:38:00
10207 - from: 2016-Dec-15 23:41:28 to: 2016-Dec-15 23:42:00
10205 - from: 2016-Dec-15 23:45:28 to: 2016-Dec-15 23:46:00
10241 - from: 2016-Dec-15 23:48:28 to: 2016-Dec-15 23:49:00
10202 - from: 2016-Dec-15 23:52:00 to: 2016-Dec-15 23:52:00

(…)

```
L1, Direction: KINGSFORD [1]

Expedition code: 596 - from: CIRCULAR QUAY [3](10239) to: KINGSFORD [1](10208)
10239 -  from: 2016-Dec-15 23:00:00 to: 2016-Dec-15 23:00:00
10236 -  from: 2016-Dec-15 23:02:28 to: 2016-Dec-15 23:03:00
10234 -  from: 2016-Dec-15 23:05:28 to: 2016-Dec-15 23:06:00
10232 -  from: 2016-Dec-15 23:08:28 to: 2016-Dec-15 23:09:00
10231 -  from: 2016-Dec-15 23:11:28 to: 2016-Dec-15 23:12:00
10228 -  from: 2016-Dec-15 23:14:28 to: 2016-Dec-15 23:15:00
10225 -  from: 2016-Dec-15 23:17:28 to: 2016-Dec-15 23:18:00
10223 -  from: 2016-Dec-15 23:20:28 to: 2016-Dec-15 23:21:00
10221 -  from: 2016-Dec-15 23:23:28 to: 2016-Dec-15 23:24:00
10218 -  from: 2016-Dec-15 23:26:28 to: 2016-Dec-15 23:27:00
10216 -  from: 2016-Dec-15 23:29:28 to: 2016-Dec-15 23:30:00
10214 -  from: 2016-Dec-15 23:32:28 to: 2016-Dec-15 23:33:00
10212 -  from: 2016-Dec-15 23:35:28 to: 2016-Dec-15 23:36:00
10210 -  from: 2016-Dec-15 23:38:28 to: 2016-Dec-15 23:39:00
10208 -  from: 2016-Dec-15 23:42:00 to: 2016-Dec-15 23:42:00

Expedition code: 598 - from: CIRCULAR QUAY [3](10239) to: KINGSFORD [1](10208)
10239 -  from: 2016-Dec-16 01:00:00 to: 2016-Dec-16 01:00:00
10236 -  from: 2016-Dec-16 01:02:28 to: 2016-Dec-16 01:03:00
10234 -  from: 2016-Dec-16 01:05:28 to: 2016-Dec-16 01:06:00
10232 -  from: 2016-Dec-16 01:08:28 to: 2016-Dec-16 01:09:00
10231 -  from: 2016-Dec-16 01:11:28 to: 2016-Dec-16 01:12:00
10228 -  from: 2016-Dec-16 01:14:28 to: 2016-Dec-16 01:15:00
10225 -  from: 2016-Dec-16 01:17:28 to: 2016-Dec-16 01:18:00
10223 -  from: 2016-Dec-16 01:20:28 to: 2016-Dec-16 01:21:00
10221 -  from: 2016-Dec-16 01:23:28 to: 2016-Dec-16 01:24:00
10218 -  from: 2016-Dec-16 01:26:28 to: 2016-Dec-16 01:27:00
10216 -  from: 2016-Dec-16 01:29:28 to: 2016-Dec-16 01:30:00
10214 -  from: 2016-Dec-16 01:32:28 to: 2016-Dec-16 01:33:00
10212 -  from: 2016-Dec-16 01:35:28 to: 2016-Dec-16 01:36:00
10210 -  from: 2016-Dec-16 01:38:28 to: 2016-Dec-16 01:39:00
10208 -  from: 2016-Dec-16 01:42:00 to: 2016-Dec-16 01:42:00

(…)


L1, Direction: O4MJyi

Expedition code: NIlCtq - from: RANDWICK [2](10201) to: CIRCULAR QUAY [1](10240)
10203 - frequency: 1 minutes
10204 - frequency: 1 minutes
10206 - frequency: 1 minutes
10219 - frequency: 1 minutes
10220 - frequency: 1 minutes
10222 - frequency: 1 minutes
10226 - frequency: 1 minutes
10229 - frequency: 1 minutes
10230 - frequency: 1 minutes
10233 - frequency: 1 minutes
10235 - frequency: 1 minutes
10237 - frequency: 1 minutes
10240 - frequency: 1 minutes
```

```
Response time: 2017-Aug-10 08:48:0
Expedition code: XE57lv - from: RANDWICK [2](10201) to: CIRCULAR QUAY
[1](10240)
10204 - frequency: 1 minutes
10206 - frequency: 1 minutes
10219 - frequency: 1 minutes
10220 - frequency: 1 minutes
10222 - frequency: 1 minutes
10226 - frequency: 1 minutes
10229 - frequency: 1 minutes
10230 - frequency: 1 minutes
10233 - frequency: 1 minutes
10235 - frequency: 1 minutes
10237 - frequency: 1 minutes
10240 - frequency: 1 minutes

Response time: 2017-Aug-10 08:48:0
```

Stop Timetable (ST)

```
Stop Timetable:

Valid until: 2016-Dec-25 08:00:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 135 -  from: 2016-Dec-16 09:08:28 to: 2016-Dec-16 09:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 137 -  from: 2016-Dec-16 11:08:28 to: 2016-Dec-16 11:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 139 -  from: 2016-Dec-16 13:08:28 to: 2016-Dec-16 13:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 141 -  from: 2016-Dec-16 15:08:28 to: 2016-Dec-16 15:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 143 -  from: 2016-Dec-16 17:08:28 to: 2016-Dec-16 17:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 145 -  from: 2016-Dec-16 19:08:28 to: 2016-Dec-16 19:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 147 -  from: 2016-Dec-16 21:08:28 to: 2016-Dec-16 21:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 149 -  from: 2016-Dec-16 23:08:28 to: 2016-Dec-16 23:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 151 -  from: 2016-Dec-17 01:08:28 to: 2016-Dec-17 01:09:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 153 -  from: 2016-Dec-16 09:12:28 to: 2016-Dec-16 09:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 155 -  from: 2016-Dec-16 11:12:28 to: 2016-Dec-16 11:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 157 -  from: 2016-Dec-16 13:12:28 to: 2016-Dec-16 13:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 159 -  from: 2016-Dec-16 15:12:28 to: 2016-Dec-16 15:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 161 -  from: 2016-Dec-16 17:12:28 to: 2016-Dec-16 17:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 163 -  from: 2016-Dec-16 19:12:28 to: 2016-Dec-16 19:13:00

MonitoringRef: ALR, LineRef: 1, DirectionRef: 0
5. 165 -  from: 2016-Dec-16 21:12:28 to: 2016-Dec-16 21:13:00

(…)

MonitoringRef: ALR, LineRef: 1, DirectionRef: RUqaCv
11. RUqaCv - frequency: 1 minutes

Response time: 2017-Aug-10 09:16:48
```

Stop Monitoring (SM)

```
Stop Monitoring:

Delivery #1
Valid until: 2016-Dec-16 07:38:30

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 555
4. ALISON ROAD [2] - 555 -  from: 2016-Dec-16 06:08:28 to: 2016-Dec-16 06:09:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 555
4. ALISON ROAD [2] - 555 -  from: 2016-Dec-16 06:08:28 to: 2016-Dec-16 06:09:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 579
4. ALISON ROAD [2] - 579 -  from: 2016-Dec-16 06:12:28 to: 2016-Dec-16 06:13:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 579
4. ALISON ROAD [2] - 579 -  from: 2016-Dec-16 06:12:28 to: 2016-Dec-16 06:13:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 611
4. ALISON ROAD [2] - 611 -  from: 2016-Dec-16 06:16:28 to: 2016-Dec-16 06:17:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 611
4. ALISON ROAD [2] - 611 -  from: 2016-Dec-16 06:16:28 to: 2016-Dec-16 06:17:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 635
4. ALISON ROAD [2] - 635 -  from: 2016-Dec-16 06:20:28 to: 2016-Dec-16 06:21:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 635
4. ALISON ROAD [2] - 635 -  from: 2016-Dec-16 06:20:28 to: 2016-Dec-16 06:21:00
delay: 00:00:00

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: non_specified,
VehicleJourneyName: 659
4. ALISON ROAD [2] - 659 -  from: 2016-Dec-16 06:24:28 to: 2016-Dec-16 06:25:00
delay: 00:00:00

(…)

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: ENZQN4,
VehicleJourneyName: ENZQN4
4. ALISON ROAD [2] - ENZQN4 - frequency: 1 minutes
delay: -00:02:54
```

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: rHHUjp,
VehicleJourneyName: rHHUjp
4. ALISON ROAD [2] - rHHUjp - frequency: 1 minutes
delay: -00:02:12

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: ZqMSP3,
VehicleJourneyName: ZqMSP3
4. ALISON ROAD [2] - ZqMSP3 - frequency: 1 minutes
delay: -00:03:46

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: O4MJyi,
VehicleJourneyName: O4MJyi
4. ALISON ROAD [2] - O4MJyi - frequency: 1 minutes
delay: -00:02:03

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: D8jKEF,
VehicleJourneyName: D8jKEF
4. ALISON ROAD [2] - D8jKEF - frequency: 1 minutes
delay: -00:02:28

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: 85q0QC,
VehicleJourneyName: 85q0QC
4. ALISON ROAD [2] - 85q0QC - frequency: 1 minutes
delay: -00:02:28

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: 4Gt2FH,
VehicleJourneyName: 4Gt2FH
4. ALISON ROAD [2] - 4Gt2FH - frequency: 1 minutes
delay: -01:20:13

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: ENZQN4,
VehicleJourneyName: ENZQN4
4. ALISON ROAD [2] - ENZQN4 - frequency: 1 minutes
delay: -00:02:54

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: 4Gt2FH,
VehicleJourneyName: 4Gt2FH
4. ALISON ROAD [2] - 4Gt2FH - frequency: 1 minutes
delay: -01:20:13

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: 85q0QC,
VehicleJourneyName: 85q0QC
4. ALISON ROAD [2] - 85q0QC - frequency: 1 minutes
delay: -00:02:28

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: D8jKEF,
VehicleJourneyName: D8jKEF
4. ALISON ROAD [2] - D8jKEF - frequency: 1 minutes
delay: -00:02:28

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: O4MJyi,
VehicleJourneyName: O4MJyi
4. ALISON ROAD [2] - O4MJyi - frequency: 1 minutes
delay: -00:02:03

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: ZqMSP3,
VehicleJourneyName: ZqMSP3
4. ALISON ROAD [2] - ZqMSP3 - frequency: 1 minutes
delay: -00:03:46

MonitoringRef: ALISON ROAD [2], LineRef: 1, DirectionRef: rHHUjp,
VehicleJourneyName: rHHUjp
4. ALISON ROAD [2] - rHHUjp - frequency: 1 minutes
delay: -00:02:12

Response time: 2017-Aug-10 09:16:48

Vehicle Monitoring (VM)

```
Vehicle Monitoring:

Delivery #1,  valid until: 2017-Aug-10 08:48:04

LineRef: 1, DirectionRef: 9, VehicleJourneyName: 1063
Long. tramo: 1553.61 m
% Recorrido: 3.28935 %
Service ID: 1063 1063
Delay: 00:00:24
Speed: 2 km/h - Bearing: 2.71992 degrees
5. MOORE PARK [2] - 1063 -  from: 2016-Dec-15 22:18:48 to: 2016-Dec-15 22:18:48
6. SURRY HILLS [2] - 1063 -  from: 2016-Dec-15 22:23:50 to: 2016-Dec-15 22:23:50
7. CENTRAL [2] - 1063 -  from: 2016-Dec-15 22:27:47 to: 2016-Dec-15 22:27:47
8. RAWSON PLACE [2] - 1063 -  from: 2016-Dec-15 22:30:51 to: 2016-Dec-15
22:30:51
9. CHINATOWN [2] - 1063 -  from: 2016-Dec-15 22:34:56 to: 2016-Dec-15 22:34:56

LineRef: 1, DirectionRef: RANDWICK [1], VehicleJourneyName: 548
Long. tramo: 473.499 m
% Recorrido: 0 %
Service ID: 548 548
Delay: 00:00:00
Speed: 14 km/h - Bearing: -1 degrees
1. CIRCULAR QUAY [2] - 548 -  from: 2016-Dec-15 23:00:00 to: 2016-Dec-15
23:00:00
2. GROSVENOR STREET [1] - 548 -  from: 2016-Dec-15 23:02:28 to: 2016-Dec-15
23:03:00
3. WYNYARD [1] - 548 -  from: 2016-Dec-15 23:05:28 to: 2016-Dec-15 23:06:00
4. QUEEN VICTORIA BUILDING [1] - 548 -  from: 2016-Dec-15 23:09:28 to: 2016-Dec-
15 23:10:00
5. TOWN HALL [1] - 548 -  from: 2016-Dec-15 23:12:28 to: 2016-Dec-15 23:13:00

LineRef: 1, DirectionRef: rHHUjp, VehicleJourneyName: rHHUjp
Long. tramo: 856.998 m
% Recorrido: 4.83931 %
Service ID: rHHUjp rHHUjp
Delay: -00:02:41
Speed: 3 km/h - Bearing: -1 degrees
4. ALISON ROAD [2] - rHHUjp - frequency: 1 minutes
5. MOORE PARK [2] - rHHUjp - frequency: 1 minutes
6. SURRY HILLS [2] - rHHUjp - frequency: 1 minutes
7. CENTRAL [2] - rHHUjp - frequency: 1 minutes
8. RAWSON PLACE [2] - rHHUjp - frequency: 1 minutes

LineRef: 1, DirectionRef: O4MJyi, VehicleJourneyName: O4MJyi
Long. tramo: 725.343 m
% Recorrido: 6.08836 %
Service ID: O4MJyi O4MJyi
Delay: -01:40:17
Speed: 5 km/h - Bearing: 2.9869 degrees
3. WANSEY ROAD [2] - O4MJyi - frequency: 1 minutes
4. ALISON ROAD [2] - O4MJyi - frequency: 1 minutes
5. MOORE PARK [2] - O4MJyi - frequency: 1 minutes
6. SURRY HILLS [2] - O4MJyi - frequency: 1 minutes
7. CENTRAL [2] - O4MJyi - frequency: 1 minutes

Response time: 2017-Aug-10 08:48:04
```

Facility Monitoring (FM)

Lines

```
Facility Monitoring: lines

       (valid until: 2017-Aug-10 09:16:47)
From: 2016-Dec-15 22:38:30 to: 2016-Dec-16 07:38:30

1      (L2)CSELR                  (Type: 2,     Status: 1)
2      (L1)IWLR

Response time: 2017-Aug-10 09:16:47
```

Stops

```
Facility Monitoring: stops

       (valid until: 2017-Aug-10 09:16:47)
From: 2016-Dec-15 22:38:30 to: 2016-Dec-16 07:38:30

1      (LB001)LB001               (Type: 2,     Status: 1)
2      (LB002)LB002
3      (LB003)LB003
4      (LB004)LB004
5      (LB005)LB005
6      (LB006)LB006
7      (LB007)LB007
8      (LB008)LB008
9      (LB009)LB009
10     (LB010)LB010
11     (LB011)LB011
12     (LB012)LB012
13     (LB013)LB013
14     (LB014)LB014
15     (LB015)LB015

(…)

166    (LB170)LB170
167    (LB171)LB171
168    (LB172)LB172
169    (LB173)LB173
170    (LB174)LB174
171    (LB175)LB175
172    (LB176)LB176
173    (LB177)LB177
174    (LB178)LB178
175    (LB179)LB179
176    (LB180)LB180
177    (LB181)LB181
178    (LB182)LB182
179    (LB183)LB183
180    (LB184)LB184
181    (2031202)RANDWICK [2]
182    (2031201)RANDWICK [1]
183    (2031200)UNSW HIGH STREET [2]
184    (2031198)WANSEY ROAD [2]
185    (2031197)WANSEY ROAD [1]
186    (2031196)ALISON ROAD [2]
187    (2031103)ALISON ROAD [1]
```

```
188    (203289)KINGSFORD [1]
189    (203288)KINGSFORD [2]
190    (203287)STRACHAN STREET [1]
191    (203286)STRACHAN STREET [2]
192    (203357)UNSW ANZAC PARADE [1]
193    (203358)UNSW ANZAC PARADE [2]
194    (203355)TODMAN AVENUE [1]
195    (203356)TODMAN AVENUE [2]
196    (203353)CARLTON STREET [1]
197    (203354)CARLTON STREET [2]
198    (2021101)MOORE PARK [1]
199    (2021102)MOORE PARK [2]
200    (2010108)SURRY HILLS [2]
201    (2010109)SURRY HILLS [1]
202    (2000448)CENTRAL [2]
203    (2000447)CENTRAL [3]
204    (2010107)CENTRAL [1]
205    (2000445)RAWSON PLACE [1]
206    (2000446)RAWSON PLACE [2]
207    (2000443)CHINATOWN [1]
208    (2000444)CHINATOWN [2]
209    (2000458)TOWN HALL [2]
210    (2000459)TOWN HALL [1]
211    (2000457)QUEEN VICTORIA BUILDING [1]
212    (2000456)QUEEN VICTORIA BUILDING [2]
213    (2000455)WYNYARD [1]
214    (2000454)WYNYARD [2]
215    (2000453)GROSVENOR STREET [1]
216    (2000452)GROSVENOR STREET [2]
217    (2000449)CIRCULAR QUAY [2]
218    (2000450)CIRCULAR QUAY [3]
219    (2000451)CIRCULAR QUAY [1]
220    (2031199)UNSW HIGH STREET [1]
221    (10300)KINGSFORD [3]
222    (10301)KINGSFORD [4]
223    (10302)MOORE PARK [3]
224    (10303)ALISON ROAD [3]
225    (10304)RANDWICK DEPOT [1]
226    (10305)RANDWICK DEPOT [2]
227    (10306)IWLR [UP]
228    (10307)IWLR [DN]
229    (EL001)EL001
230    (EL002)EL002
231    (EL003)EL003
232    (EL004)EL004
233    (EL005)EL005
234    (EL006)EL006
235    (EL007)EL007
236    (EL009)EL009
237    (EL010)EL010
238    (EL011)EL011
239    (EL012)EL012
240    (EL013)EL013
241    (EL014)EL014
242    (EL015)EL015
243    (EL016)EL016
244    (EL017)EL017
245    (EL018)EL018
246    (EL019)EL019
247    (EL020)EL020
248    (EL021)EL021
249    (EL022)EL022
```

```
250    (EL023)EL023
251    (EL024)EL024
252    (EL025)EL025
253    (EL026)EL026
254    (LA001)DULWICH HILL
255    (LA003)DULWICH GROVE [1]
256    (LA004)DULWICH GROVE [2]
257    (LA005)ARLINGTON [1]
258    (LA006)ARLINGTON[2]
259    (LA007)WARATAH MILLS [1]
260    (LA008)WARATAH MILLS [2]
261    (LA009)LEWISHAM WEST [1]
262    (LA010)LEWISHAM WEST [2]
263    (LA011)TAVERNERS HILL [1]
264    (LA012)TAVERNERS HILL [2]
265    (LA013)MARION [1]
266    (LA014)MARION [2]
267    (LA015)HAWTHORNE [1]
268    (LA016)HAWTHORNE [2]
269    (LA017)LEICHHARDT NORTH [1]
270    (LA018)LEICHHARDT NORTH [2]
271    (LA019)LILYFIELD [1]
272    (LA020)LILYFIELD [2]
273    (LA021)ROZELLE BAY [1]
274    (LA022)ROZELLE BAY [2]
275    (LA023)JUBILEE PARK [1]
276    (LA024)JUBILEE PARK [2]
277    (LA025)GLEBE [1]
278    (LA026)GLEBE [2]
279    (LA027)WENTWORTH PARK [1]
280    (LA028)WENTWORTH PARK [2]
281    (LA029)FISH MARKET [1]
282    (LA030)FISH MARKET [2]
283    (LA031)JOHN STREET [1]
284    (LA032)JOHN STREET [2]
285    (LA033)THE STAR [1]
286    (LA034)THE STAR [2]
287    (LA035)PYRMONT BAY [1]
288    (LA036)PYRMONT BAY [2]
289    (LA037)CONVENTION [1]
290    (LA038)CONVENTION [2]
291    (LA039)EXHIBITION CENTRE [1]
292    (LA040)EXHIBITION CENTRE [2]
293    (LA041)PADDY'S MARKETS [1]
294    (LA042)PADDY'S MARKETS [2]
295    (LA043)CENTRAL LIGHT RAIL
296    (LA044)CAPITOL SQUARE STOP[2]
297    (LA045)CAPITOL SQUARE STOP [1]
298    (EN001)EN001
299    (EN002)EN002
300    (EN003)EN003
301    (EN004)EN004
302    (EN005)EN005
303    (EN006)EN006
304    (EN007)EN007
305    (AG001)AG001
306    (AG002)AG002
307    (AG003)AG003
308    (AG004)AG004
309    (AG005)AG005
```

```
310    (AG006)AG006
311    (AG007)AG007
312    (AG008)AG008
313    (AG009)AG009
314    (AG010)AG010
315    (AG011)AG011
316    (AG012)AG012
317    (AG013)AG013
318    (TU001)TU001
319    (TU002)TU002
320    (OS001)OS001
321    (OS002)OS002
322    (NS001)NS001
323    (DE001)DE001
324    (DE002)DE002

Response time: 2017-Aug-10 09:16:47
```

Lines Discovery

```
Lines Discovery:

Valid until: 2017-Aug-10 09:16:47
1       CSELR L2 true
                2031202 2031202
                2031201 2031201
                2031200 2031200
                2031197 2031197
                2031196 2031196
                2031103 2031103
                203289 203289
                203288 203288
                203353 203353
                203354 203354
                2021102 2021102
                2010108 2010108
                2010109 2010109
                2000447 2000447
                2010107 2010107
                2000446 2000446
                2000443 2000443
                2000444 2000444
                2000458 2000458
                2000459 2000459
                2000457 2000457
                2000453 2000453
                2000449 2000449
                2000450 2000450
                2000451 2000451
                10302 10302
                10304 10304
                10305 10305
                10307 10307
2       IWLR L1 true
                LA001 LA001
                LA043 LA043

2017-Aug-10 09:16:47
```

StopPoints Discovery

```
StopPoints discovery:

Valid until: 2017-Aug-10 09:16:48
1       LB001 LB001 true -33.9073 151.227
2       LB002 LB002 true -33.9073 151.227
3       LB003 LB003 true -33.9071 151.227
4       LB004 LB004 true -33.907 151.227
5       LB005 LB005 true -33.907 151.227
6       LB006 LB006 true -33.9071 151.227
7       LB007 LB007 true -33.9071 151.227
8       LB008 LB008 true -33.9071 151.227
9       LB009 LB009 true -33.907 151.227
10      LB010 LB010 true -33.907 151.227
11      LB011 LB011 true -33.9069 151.227
12      LB012 LB012 true -33.9069 151.227
13      LB013 LB013 true -33.9065 151.227
14      LB014 LB014 true -33.9064 151.227
15      LB015 LB015 true -33.9068 151.227
16      LB016 LB016 true -33.9068 151.227
17      LB017 LB017 true -33.9065 151.227
18      LB018 LB018 true -33.9065 151.227
19      LB019 LB019 true -33.9067 151.227
20      LB020 LB020 true -33.9067 151.227
21      LB021 LB021 true -33.9065 151.227
22      LB022 LB022 true -33.9065 151.227
23      LB023 LB023 true -33.9066 151.227
24      LB024 LB024 true -33.9066 151.227
25      LB025 LB025 true -33.9065 151.227
26      LB026 LB026 true -33.9065 151.227
27      LB027 LB027 true -33.9076 151.227
28      LB028 LB028 true -33.9076 151.227
29      LB029 LB029 true -33.9073 151.227
30      LB030 LB030 true -33.9074 151.227
31      LB033 LB033 true -33.9071 151.227
32      LB034 LB034 true -33.9071 151.227
33      LB035 LB035 true -33.906 151.227
34      LB036 LB036 true -33.906 151.227
35      LB037 LB037 true -33.9057 151.228
36      LB038 LB038 true -33.9057 151.228
37      LB039 LB039 true -33.9052 151.228
38      LB040 LB040 true -33.9052 151.228
39      LB041 LB041 true -33.8994 151.223
40      LB042 LB042 true -33.8994 151.223
41      LB043 LB043 true -33.8994 151.223
42      LB044 LB044 true -33.8994 151.223
43      LB045 LB045 true -33.8943 151.222
44      LB046 LB046 true -33.8944 151.222
45      LB047 LB047 true -33.8944 151.222
46      LB048 LB048 true -33.8944 151.222
47      LB049 LB049 true -33.8941 151.222
48      LB050 LB050 true -33.8941 151.222
49      LB051 LB051 true -33.8933 151.222
50      LB052 LB052 true -33.8933 151.222
51      LB053 LB053 true -33.9054 151.224
52      LB054 LB054 true -33.9054 151.224
53      LB055 LB055 true -33.9062 151.224
54      LB056 LB056 true -33.9062 151.224
```

```
55     LB057 LB057 true -33.909 151.223
56     LB058 LB058 true -33.909 151.223
57     LB059 LB059 true -33.9098 151.223
58     LB060 LB060 true -33.9097 151.223
59     LB061 LB061 true -33.9163 151.226
60     LB062 LB062 true -33.9162 151.226
61     LB063 LB063 true -33.917 151.226
62     LB064 LB064 true -33.917 151.226
63     LB065 LB065 true -33.9215 151.227
64     LB066 LB066 true -33.9214 151.227
65     LB067 LB067 true -33.9223 151.227
66     LB068 LB068 true -33.9223 151.227
67     LB069 LB069 true -33.9247 151.229
68     LB070 LB070 true -33.9249 151.229
69     LB071 LB071 true -33.9254 151.23
70     LB072 LB072 true -33.9255 151.23
71     LB073 LB073 true -33.9257 151.23
72     LB074 LB074 true -33.9257 151.23
73     LB075 LB075 true -33.9044 151.227
74     LB076 LB076 true -33.9044 151.227
75     LB077 LB077 true -33.9044 151.227
76     LB078 LB078 true -33.9044 151.227
77     LB079 LB079 true -33.9047 151.228
78     LB080 LB080 true -33.9047 151.228
79     LB081 LB081 true -33.905 151.228
80     LB082 LB082 true -33.905 151.228
81     LB083 LB083 true -33.905 151.228
82     LB084 LB084 true -33.9051 151.228
83     LB085 LB085 true -33.9053 151.229
84     LB086 LB086 true -33.9054 151.229
85     LB087 LB087 true -33.9054 151.23
86     LB088 LB088 true -33.9054 151.229
87     LB089 LB089 true -33.9054 151.23
88     LB090 LB090 true -33.9054 151.23
89     LB091 LB091 true -33.9058 151.231
90     LB092 LB092 true -33.9058 151.231
91     LB093 LB093 true -33.9103 151.235
92     LB094 LB094 true -33.9102 151.235
93     LB095 LB095 true -33.911 151.236
94     LB096 LB096 true -33.9108 151.236
95     LB097 LB097 true -33.9163 151.235
96     LB098 LB098 true -33.9163 151.235
97     LB099 LB099 true -33.9164 151.236
98     LB100 LB100 true -33.9165 151.236
99     LB101 LB101 true -33.9169 151.239
100    LB102 LB102 true -33.917 151.239
101    LB103 LB103 true -33.917 151.24
102    LB104 LB104 true -33.917 151.24
103    LB107 LB107 true -33.913 151.225
104    LB108 LB108 true -33.913 151.225
105    LB109 LB109 true -33.9023 151.224
106    LB110 LB110 true -33.9023 151.224
107    LB111 LB111 true -33.9073 151.233
108    LB112 LB112 true -33.9073 151.233
109    LB113 LB113 true -33.9136 151.235
110    LB114 LB114 true -33.9136 151.235
111    LB115 LB115 true -33.8973 151.222
112    LB116 LB116 true -33.8973 151.222
113    LB117 LB117 true -33.8973 151.222
114    LB118 LB118 true -33.8973 151.222
115    LB119 LB119 true -33.8615 151.21
116    LB120 LB120 true -33.8615 151.209
117    LB121 LB121 true -33.8614 151.209
118    LB122 LB122 true -33.8614 151.209
119    LB123 LB123 true -33.8613 151.209
120    LB124 LB124 true -33.8614 151.209
```

```
121    LB125 LB125 true -33.8638 151.207
122    LB126 LB126 true -33.8638 151.208
123    LB127 LB127 true -33.8645 151.207
124    LB128 LB128 true -33.8645 151.207
125    LB129 LB129 true -33.8663 151.207
126    LB130 LB130 true -33.8663 151.207
127    LB131 LB131 true -33.867 151.207
128    LB132 LB132 true -33.867 151.207
129    LB133 LB133 true -33.8673 151.207
130    LB134 LB134 true -33.8673 151.207
131    LB135 LB135 true -33.8673 151.207
132    LB136 LB136 true -33.8673 151.207
133    LB137 LB137 true -33.8711 151.207
134    LB138 LB138 true -33.8711 151.207
135    LB139 LB139 true -33.8719 151.207
136    LB140 LB140 true -33.8719 151.207
137    LB141 LB141 true -33.8735 151.207
138    LB142 LB142 true -33.8735 151.207
139    LB143 LB143 true -33.8743 151.207
140    LB144 LB144 true -33.8743 151.207
141    LB145 LB145 true -33.8746 151.207
142    LB146 LB146 true -33.8747 151.207
143    LB147 LB147 true -33.8749 151.207
144    LB148 LB148 true -33.8749 151.207
145    LB149 LB149 true -33.8782 151.206
146    LB150 LB150 true -33.8782 151.206
147    LB151 LB151 true -33.879 151.205
148    LB152 LB152 true -33.879 151.206
149    LB153 LB153 true -33.8796 151.205
150    LB154 LB154 true -33.8795 151.205
151    LB155 LB155 true -33.8795 151.205
152    LB156 LB156 true -33.8795 151.205
153    LB157 LB157 true -33.8812 151.205
154    LB158 LB158 true -33.8812 151.205
155    LB159 LB159 true -33.8817 151.206
156    LB160 LB160 true -33.8816 151.206
157    LB161 LB161 true -33.8838 151.208
158    LB162 LB162 true -33.8838 151.208
159    LB163 LB163 true -33.8838 151.208
160    LB164 LB164 true -33.8838 151.208
161    LB165 LB165 true -33.8838 151.208
162    LB166 LB166 true -33.8845 151.208
163    LB167 LB167 true -33.8845 151.207
164    LB168 LB168 true -33.8845 151.207
165    LB169 LB169 true -33.8847 151.207
166    LB170 LB170 true -33.8848 151.207
167    LB171 LB171 true -33.8847 151.207
168    LB172 LB172 true -33.8848 151.207
169    LB173 LB173 true -33.8847 151.207
170    LB174 LB174 true -33.8848 151.207
171    LB175 LB175 true -33.8879 151.211
172    LB176 LB176 true -33.8879 151.211
173    LB177 LB177 true -33.8883 151.212
174    LB178 LB178 true -33.8883 151.212
175    LB179 LB179 true -33.89 151.218
176    LB180 LB180 true -33.8899 151.218
177    LB181 LB181 true -33.8925 151.222
178    LB182 LB182 true -33.8925 151.222
179    LB183 LB183 true -33.8928 151.222
180    LB184 LB184 true -33.8928 151.222
181    RANDWICK [2] 2031202 true -33.9172 151.24
182    RANDWICK [1] 2031201 true -33.9171 151.24
183    UNSW HIGH STREET [2] 2031200 true -33.9164 151.236
184    WANSEY ROAD [2] 2031198 true -33.9106 151.235
185    WANSEY ROAD [1] 2031197 true -33.9105 151.235
186    ALISON ROAD [2] 2031196 true -33.9053 151.229
```

```
187    ALISON ROAD [1] 2031103 true -33.9051 151.229
188    KINGSFORD [1] 203289 true -33.925 151.229
189    KINGSFORD [2] 203288 true -33.9252 151.229
190    STRACHAN STREET [1] 203287 true -33.9217 151.227
191    STRACHAN STREET [2] 203286 true -33.9219 151.227
192    UNSW ANZAC PARADE [1] 203357 true -33.9165 151.226
193    UNSW ANZAC PARADE [2] 203358 true -33.9166 151.226
194    TODMAN AVENUE [1] 203355 true -33.9092 151.223
195    TODMAN AVENUE [2] 203356 true -33.9093 151.223
196    CARLTON STREET [1] 203353 true -33.9057 151.224
197    CARLTON STREET [2] 203354 true -33.9057 151.224
198    MOORE PARK [1] 2021101 true -33.8938 151.222
199    MOORE PARK [2] 2021102 true -33.8938 151.222
200    SURRY HILLS [2] 2010108 true -33.8881 151.212
201    SURRY HILLS [1] 2010109 true -33.8881 151.212
202    CENTRAL [2] 2000448 true -33.8842 151.208
203    CENTRAL [3] 2000447 true -33.8843 151.208
204    CENTRAL [1] 2010107 true -33.8842 151.208
205    RAWSON PLACE [1] 2000445 true -33.8814 151.205
206    RAWSON PLACE [2] 2000446 true -33.8815 151.205
207    CHINATOWN [1] 2000443 true -33.8785 151.206
208    CHINATOWN [2] 2000444 true -33.8785 151.206
209    TOWN HALL [2] 2000458 true -33.8739 151.207
210    TOWN HALL [1] 2000459 true -33.8739 151.207
211    QUEEN VICTORIA BUILDING [1] 2000457 true -33.8715 151.207
212    QUEEN VICTORIA BUILDING [2] 2000456 true -33.8715 151.207
213    WYNYARD [1] 2000455 true -33.8667 151.207
214    WYNYARD [2] 2000454 true -33.8667 151.207
215    GROSVENOR STREET [1] 2000453 true -33.8641 151.208
216    GROSVENOR STREET [2] 2000452 true -33.8641 151.207
217    CIRCULAR QUAY [2] 2000449 true -33.8615 151.21
218    CIRCULAR QUAY [3] 2000450 true -33.8615 151.21
219    CIRCULAR QUAY [1] 2000451 true -33.8616 151.21
220    UNSW HIGH STREET [1] 2031199 true -33.9163 151.236
221    KINGSFORD [3] 10300 true -33.9258 151.23
222    KINGSFORD [4] 10301 true -33.9259 151.23
223    MOORE PARK [3] 10302 true -33.8948 151.222
224    ALISON ROAD [3] 10303 true -33.9056 151.23
225    RANDWICK DEPOT [1] 10304 true -33.9051 151.228
226    RANDWICK DEPOT [2] 10305 true -33.9051 151.228
227    IWLR [UP] 10306 true -33.8796 151.205
228    IWLR [DN] 10307 true -33.8796 151.205
229    EL001 EL001 true -33.8615 151.21
230    EL002 EL002 true -33.8616 151.21
231    EL003 EL003 true -33.8616 151.21
232    EL004 EL004 true -33.9172 151.241
233    EL005 EL005 true -33.9172 151.241
234    EL006 EL006 true -33.9261 151.231
235    EL007 EL007 true -33.9261 151.231
236    EL009 EL009 true -33.8796 151.205
237    EL010 EL010 true -33.9051 151.227
238    EL011 EL011 true -33.9051 151.227
239    EL012 EL012 true -33.9051 151.227
240    EL013 EL013 true -33.9051 151.227
241    EL014 EL014 true -33.9051 151.227
242    EL015 EL015 true -33.9051 151.227
243    EL016 EL016 true -33.9051 151.227
244    EL017 EL017 true -33.9051 151.227
245    EL018 EL018 true -33.9051 151.227
246    EL019 EL019 true -33.9051 151.227
247    EL020 EL020 true -33.9051 151.227
248    EL021 EL021 true -33.9051 151.227
249    EL022 EL022 true -33.9051 151.227
250    EL023 EL023 true -33.9082 151.226
251    EL024 EL024 true -33.906 151.228
252    EL025 EL025 true -33.906 151.228
```

```
253    EL026 EL026 true -33.8951 151.222
254    DULWICH HILL LA001 true -33.9103 151.14
255    DULWICH GROVE [1] LA003 true -33.9051 151.139
256    DULWICH GROVE [2] LA004 true -33.9051 151.139
257    ARLINGTON [1] LA005 true -33.9018 151.138
258    ARLINGTON[2] LA006 true -33.9018 151.138
259    WARATAH MILLS [1] LA007 true -33.899 151.14
260    WARATAH MILLS [2] LA008 true -33.8989 151.14
261    LEWISHAM WEST [1] LA009 true -33.8937 151.144
262    LEWISHAM WEST [2] LA010 true -33.8939 151.143
263    TAVERNERS HILL [1] LA011 true -33.8891 151.145
264    TAVERNERS HILL [2] LA012 true -33.8891 151.145
265    MARION [1] LA013 true -33.884 151.145
266    MARION [2] LA014 true -33.884 151.145
267    HAWTHORNE [1] LA015 true -33.8796 151.147
268    HAWTHORNE [2] LA016 true -33.8796 151.147
269    LEICHHARDT NORTH [1] LA017 true -33.875 151.154
270    LEICHHARDT NORTH [2] LA018 true -33.875 151.154
271    LILYFIELD [1] LA019 true -33.8742 151.165
272    LILYFIELD [2] LA020 true -33.8741 151.165
273    ROZELLE BAY [1] LA021 true -33.8719 151.173
274    ROZELLE BAY [2] LA022 true -33.8718 151.173
275    JUBILEE PARK [1] LA023 true -33.8755 151.179
276    JUBILEE PARK [2] LA024 true -33.8754 151.179
277    GLEBE [1] LA025 true -33.8772 151.187
278    GLEBE [2] LA026 true -33.8772 151.187
279    WENTWORTH PARK [1] LA027 true -33.8745 151.194
280    WENTWORTH PARK [2] LA028 true -33.8744 151.194
281    FISH MARKET [1] LA029 true -33.8708 151.192
282    FISH MARKET [2] LA030 true -33.8709 151.192
283    JOHN STREET [1] LA031 true -33.8675 151.192
284    JOHN STREET [2] LA032 true -33.8674 151.192
285    THE STAR [1] LA033 true -33.8677 151.195
286    THE STAR [2] LA034 true -33.8677 151.195
287    PYRMONT BAY [1] LA035 true -33.8697 151.198
288    PYRMONT BAY [2] LA036 true -33.8697 151.198
289    CONVENTION [1] LA037 true -33.8728 151.198
290    CONVENTION [2] LA038 true -33.8728 151.198
291    EXHIBITION CENTRE [1] LA039 true -33.8772 151.2
292    EXHIBITION CENTRE [2] LA040 true -33.8772 151.2
293    PADDY'S MARKETS [1] LA041 true -33.8794 151.203
294    PADDY'S MARKETS [2] LA042 true -33.8793 151.203
295    CENTRAL LIGHT RAIL LA043 true -33.8824 151.207
296    CAPITOL SQUARE STOP[2] LA044 true -33.8798 151.206
297    CAPITOL SQUARE STOP [1] LA045 true -33.8798 151.206
298    EN001 EN001 true -33.9161 151.235
299    EN002 EN002 true -33.9161 151.235
300    EN003 EN003 true -33.8916 151.221
301    EN004 EN004 true -33.891 151.221
302    EN005 EN005 true -33.9121 151.224
303    EN006 EN006 true -33.911 151.223
304    EN007 EN007 true -33.8615 151.208
305    AG001 AG001 true -33.8845 151.207
306    AG002 AG002 true -33.917 151.24
307    AG003 AG003 true -33.8613 151.209
308    AG004 AG004 true -33.917 151.24
309    AG005 AG005 true -33.8615 151.209
310    AG006 AG006 true -33.8614 151.209
311    AG007 AG007 true -33.8992 151.223
312    AG008 AG008 true -33.8794 151.205
313    AG009 AG009 true -33.8849 151.207
314    AG010 AG010 true -33.8837 151.208
315    AG011 AG011 true -33.8846 151.207
316    AG012 AG012 true -33.8847 151.207
317    AG013 AG013 true -33.8839 151.208
318    TU001 TU001 true -33.8926 151.222
```

```
319    TU002 TU002 true -33.8896 151.215
320    OS001 OS001 true -33.8959 151.222
321    OS002 OS002 true -33.8984 151.223
322    NS001 NS001 true -33.9021 151.224
323    DE001 DE001 true -33.9053 151.229
324    DE002 DE002 true -33.9109 151.235

2017-Aug-10 09:16:48
```

Anexo 4: XML intercambiado con *endpoints* reales de SIRI

Tampere (ITS Factory)

Ciudad/Región: Tampere (Finlandia)

Compañía/Organización: ITS Factory

Petición de General Message

```
<?xml version="1.0" encoding="UTF-8"?>
<Siri
  xmlns="http://www.siri.org.uk/siri"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.3"
xsi:schemaLocation="http://www.kizoom.com/standards/siri/schema/1.3/siri.xs
d">
  <ServiceRequest>
    <RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
    <GeneralMessageRequest version="1.3">
      <RequestTimestamp>2012-06-11T09:30:50-03:00</RequestTimestamp>
    </GeneralMessageRequest>
  </ServiceRequest>
</Siri>
```

Respuesta de General Message

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Siri
  xmlns="http://www.siri.org.uk/siri"
  xmlns:ns2="http://www.ifopt.org.uk/acsb"
  xmlns:ns3="http://www.ifopt.org.uk/ifopt"
  xmlns:ns4="http://datex2.eu/schema/1_0/1_0" version="1.3">
  <ServiceDelivery>
    <ResponseTimestamp>2018-06-03T19:00:40.598+03:00</ResponseTimestamp>
    <ProducerRef>IJ2010</ProducerRef>
    <Status>true</Status>
    <MoreData>false</MoreData>
    <GeneralMessageDelivery version="1.3">
      <ResponseTimestamp>2018-06-03T19:00:40.598+03:00</ResponseTimestamp>
      <Status>true</Status>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2010-01-01T00:00:00+02:00</RecordedAtTime>
        <InfoMessageIdentifier>1014</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>errors</InfoChannelRef>
        <ValidUntilTime>2099-01-01T00:00:00+02:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Linja 3R ajaa Possilankadulta suoraan Teivaalankadulle
4.6. alkaen. Reitiltä jää pois Ryydynkatu ja osa Teivaalantiestä. -- Bus 3R
runs from Possilankatu straight to Teivaalankatu, starting 4 June.
Ryydynkatu and part of Teivaalantie will be cut off from the route. --
nysse.fi
        </Content>
      </GeneralMessage>
```

```
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2010-01-01T00:00:00+02:00</RecordedAtTime>
        <InfoMessageIdentifier>1008</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>warnings</InfoChannelRef>
        <ValidUntilTime>2099-01-01T00:00:00+02:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Reitti kulkee Kissanmaalla Karhukadun kautta
4.6.alkaen. Pysäkit Kissanmaankatu 7/8 ja Hippoksenkatu siirretään
Karhukadulle. -- The route runs via Karhukatu at Kissanmaa, from 4 June.
Stops at Kissanmaankatu 7/8 and Hippoksenkatu will be moved to Karhukatu. -
- nysse.fi
        </Content>
      </GeneralMessage>

      <GeneralMessage formatRef="string">
        <RecordedAtTime>2018-05-31T00:00:00+03:00</RecordedAtTime>
        <InfoMessageIdentifier>1015</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>warnings</InfoChannelRef>
        <ValidUntilTime>2018-06-04T23:59:00+03:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Pysäkki Ratapihankatu (3546) otetaan käyttöön uudella
paikalla Kalevantiellä 4.6. -- The stop Ratapihankatu (3546) will be re-
placed on Kalevantie, on 4 June. --nysse.fi
        </Content>
      </GeneralMessage>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2018-06-01T00:00:00+03:00</RecordedAtTime>
        <InfoMessageIdentifier>1016</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>warnings</InfoChannelRef>
        <ValidUntilTime>2018-08-08T23:59:00+03:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Arkiaamujen vuoro TTY - Atala klo 06.25 on muutettu
lähteväksi klo 06.55. -- The weekday schedule from TTY to Atala, at 06.25
am has been changed to 06.55 am. -- nysse.fi
        </Content>
      </GeneralMessage>
      <GeneralMessage formatRef="string">
        <RecordedAtTime>2018-06-01T00:00:00+03:00</RecordedAtTime>
        <InfoMessageIdentifier>1017</InfoMessageIdentifier>
        <InfoMessageVersion>1</InfoMessageVersion>
        <InfoChannelRef>warnings</InfoChannelRef>
        <ValidUntilTime>2018-06-04T23:59:00+03:00</ValidUntilTime>
        <Content
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Pysäkki Yliopisto B (0565) siirretään työmaan vuoksi
170 m taaksepäin 4.6. klo 18. -- Stop Yliopisto (0565) will be moved 170 m
backwards due construction, 6 pm on 4 June. -- nysse.fi
        </Content>
      </GeneralMessage>
```

```
        <GeneralMessage formatRef="string">
          <RecordedAtTime>2018-05-30T00:00:00+03:00</RecordedAtTime>
          <InfoMessageIdentifier>1018</InfoMessageIdentifier>
          <InfoMessageVersion>1</InfoMessageVersion>
          <InfoChannelRef>warnings</InfoChannelRef>
          <ValidUntilTime>2018-06-04T23:59:00+03:00</ValidUntilTime>
          <Content
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Nyssen kesäkausi alkaa maanantaina 4.6. Tarkista
muutokset netissä! -- The summer season of Nysse starts on Monday, 4 June.
Check the changes online! -- nysse.fi
          </Content>
        </GeneralMessage>
      </GeneralMessageDelivery>
    </ServiceDelivery>
</Siri>
```

Petición de Vehicle Monitoring

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns6:GetVehicleMonitoring
      xmlns:ns2="http://www.siri.org.uk/siri"
      xmlns:ns3="http://www.ifopt.org.uk/acsb"
      xmlns:ns4="http://www.ifopt.org.uk/ifopt"
      xmlns:ns5="http://datex2.eu/schema/2_0RC1/2_0"
      xmlns:ns6="http://wsdl.siri.org.uk">
      <ServiceRequestInfo>
        <ns2:RequestTimestamp>2017-05-
25T09:29:47.529+02:00</ns2:RequestTimestamp>

<ns2:MessageIdentifier>VehicleMonitoring:Test:0</ns2:MessageIdentifier>
      </ServiceRequestInfo>
      <Request version="2.0">
        <ns2:RequestTimestamp>2017-05-
25T09:29:47.522+02:00</ns2:RequestTimestamp>
        <ns2:LineRef>1</ns2:LineRef>
      </Request>
      <RequestExtension/>
    </ns6:GetVehicleMonitoring>
  </S:Body>
</S:Envelope>
```

Respuesta de Vehicle Monitoring

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Siri version="1.3"
  xmlns="http://www.siri.org.uk/siri">
  <ServiceDelivery>
    <ResponseTimestamp>2018-06-03T18:55:32.049+03:00</ResponseTimestamp>
    <ProducerRef>IJ2010</ProducerRef>
    <Status>true</Status>
    <MoreData>false</MoreData>
    <VehicleMonitoringDelivery version="1.3">
      <ResponseTimestamp>2018-06-03T18:55:32.049+03:00</ResponseTimestamp>
      <Status>true</Status>
      <VehicleActivity>
        <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
        <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
        <MonitoredVehicleJourney>
          <LineRef>1C</LineRef>
          <DirectionRef>1</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2018-06-03</DataFrameRef>
            <DatedVehicleJourneyRef>1800</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <OperatorRef>TKL</OperatorRef>
          <OriginName xml:lang="fi">Vatiala</OriginName>
          <DestinationName xml:lang="fi">Vaitti</DestinationName>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>23.6440378</Longitude>
            <Latitude>61.4657443</Latitude>
          </VehicleLocation>
          <Bearing>258.0</Bearing>
          <Delay>P0Y0M0DT0H3M47.000S</Delay>
          <VehicleRef>TKL_46</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
      <VehicleActivity>
        <RecordedAtTime>2018-06-03T18:55:32.024+03:00</RecordedAtTime>
        <ValidUntilTime>2018-06-03T18:56:02.024+03:00</ValidUntilTime>
        <MonitoredVehicleJourney>
          <LineRef>1B</LineRef>
          <DirectionRef>1</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2018-06-03</DataFrameRef>
            <DatedVehicleJourneyRef>1840</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <OperatorRef>paunu</OperatorRef>
          <OriginName xml:lang="fi">Vatiala</OriginName>
          <DestinationName xml:lang="fi">Teollisuustie</DestinationName>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>23.8767302</Longitude>
            <Latitude>61.4998658</Latitude>
          </VehicleLocation>
          <Bearing>0.0</Bearing>
          <Delay>P0Y0M0DT0H0M35.000S</Delay>
          <VehicleRef>paunu_157</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
```

```
<VehicleActivity>
  <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
  <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
  <MonitoredVehicleJourney>
    <LineRef>1C</LineRef>
    <DirectionRef>2</DirectionRef>
    <FramedVehicleJourneyRef>
      <DataFrameRef>2018-06-03</DataFrameRef>
      <DatedVehicleJourneyRef>1830</DatedVehicleJourneyRef>
    </FramedVehicleJourneyRef>
    <OperatorRef>Paunu</OperatorRef>
    <OriginName xml:lang="fi">Teollisuustie</OriginName>
    <DestinationName xml:lang="fi">Vatiala</DestinationName>
    <Monitored>true</Monitored>
    <VehicleLocation>
      <Longitude>23.7548063</Longitude>
      <Latitude>61.4711505</Latitude>
    </VehicleLocation>
    <Bearing>53.0</Bearing>
    <Delay>P0Y0M0DT0H2M28.000S</Delay>
    <VehicleRef>Paunu_149</VehicleRef>
  </MonitoredVehicleJourney>
</VehicleActivity>
<VehicleActivity>
  <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
  <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
  <MonitoredVehicleJourney>
    <LineRef>1C</LineRef>
    <DirectionRef>2</DirectionRef>
    <FramedVehicleJourneyRef>
      <DataFrameRef>2018-06-03</DataFrameRef>
      <DatedVehicleJourneyRef>1750</DatedVehicleJourneyRef>
    </FramedVehicleJourneyRef>
    <OperatorRef>TKL</OperatorRef>
    <OriginName xml:lang="fi">Teollisuustie</OriginName>
    <DestinationName xml:lang="fi">Vatiala</DestinationName>
    <Monitored>true</Monitored>
    <VehicleLocation>
      <Longitude>23.9702253</Longitude>
      <Latitude>61.4805132</Latitude>
    </VehicleLocation>
    <Bearing>134.0</Bearing>
    <Delay>P0Y0M0DT0H1M34.000S</Delay>
    <VehicleRef>TKL_85</VehicleRef>
  </MonitoredVehicleJourney>
</VehicleActivity>
<VehicleActivity>
  <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
  <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
  <MonitoredVehicleJourney>
    <LineRef>1B</LineRef>
    <DirectionRef>1</DirectionRef>
    <FramedVehicleJourneyRef>
      <DataFrameRef>2018-06-03</DataFrameRef>
      <DatedVehicleJourneyRef>1820</DatedVehicleJourneyRef>
    </FramedVehicleJourneyRef>
    <OperatorRef>Paunu</OperatorRef>
    <OriginName xml:lang="fi">Vatiala</OriginName>
    <DestinationName xml:lang="fi">Teollisuustie</DestinationName>
    <Monitored>true</Monitored>
    <VehicleLocation>
      <Longitude>23.7693227</Longitude>
```

```
            <Latitude>61.4950417</Latitude>
          </VehicleLocation>
          <Bearing>156.0</Bearing>
          <Delay>P0Y0M0DT0H2M11.000S</Delay>
          <VehicleRef>Paunu_147</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
      <VehicleActivity>
        <RecordedAtTime>2018-06-03T18:55:32.019+03:00</RecordedAtTime>
        <ValidUntilTime>2018-06-03T18:56:02.019+03:00</ValidUntilTime>
        <MonitoredVehicleJourney>
          <LineRef>1C</LineRef>
          <DirectionRef>2</DirectionRef>
          <FramedVehicleJourneyRef>
            <DataFrameRef>2018-06-03</DataFrameRef>
            <DatedVehicleJourneyRef>1850</DatedVehicleJourneyRef>
          </FramedVehicleJourneyRef>
          <OperatorRef>Paunu</OperatorRef>
          <OriginName xml:lang="fi">Teollisuustie</OriginName>
          <DestinationName xml:lang="fi">Vatiala</DestinationName>
          <Monitored>true</Monitored>
          <VehicleLocation>
            <Longitude>23.6291215</Longitude>
            <Latitude>61.4667943</Latitude>
          </VehicleLocation>
          <Bearing>109.0</Bearing>
          <Delay>-P0Y0M0DT0H1M26.000S</Delay>
          <VehicleRef>Paunu_166</VehicleRef>
        </MonitoredVehicleJourney>
      </VehicleActivity>
    </VehicleMonitoringDelivery>
  </ServiceDelivery>
</Siri>
```

Utah (UTA)

Ciudad/Región: Utah (Estados Unidos)

Compañía/Organización: Utah Transport Authority (UTA)

Petición de Stop Monitoring

```
[HTTP GET]
http://api.rideuta.com/SIRI/SIRI.svc/StopMonitor?stopid=133054
&minutesout=30
&onwardcalls=true
&filterroute=
&usertoken=ABCDEFGHIJ
```

Respuesta de Stop Monitoring

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T10:03:50.0961263-06:00</ResponseTimestamp>
  <StopMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T10:03:50.0961263-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T10:04:00.0961263-06:00</ValidUntil>
    <MonitoredStopVisit>
      <RecordedAtTime>2018-06-03T10:03:50.0961263-06:00</RecordedAtTime>
    </MonitoredStopVisit>
    <Extensions>
      <StopName>3500 S @ 6450 W</StopName>
      <StopLongitude>-112.045255000</StopLongitude>
      <StopLatitude>40.696599000</StopLatitude>
      <StopLocation>Utah</StopLocation>
    </Extensions>
  </StopMonitoringDelivery>
</Siri>
```

Petición de *Vehicle Monitoring* 1 (filtro por ruta)

```
[HTTP GET]
http://api.rideuta.com/SIRI/SIRI.svc/VehicleMonitor/ByRoute?route=2
&onwardcalls=true
&usertoken=ABCDEFGHIJ
```

Respuesta de Vehicle Monitoring 1

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T11:55:13.8241503-06:00</ResponseTimestamp>
  <VehicleMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T11:55:13.8241503-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T11:55:23.8241503-06:00</ValidUntil>
    <VehicleActivity>
      <RecordedAtTime>2018-06-03T11:55:13.8241503-06:00</RecordedAtTime>
    </VehicleActivity>
  </VehicleMonitoringDelivery>
</Siri>
```

Petición de *Vehicle Monitoring* 2 (filtro por vehículo)

```
[HTTP GET]

http://api.rideuta.com/SIRI/SIRI.svc/VehicleMonitor/ByVehicle?vehicle=10054
&onwardcalls=true
&usertoken=ABCDEFGHIJ
```

Respuesta de Vehicle Monitoring 2

```
<?xml version="1.0" encoding="utf-8"?>
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ResponseTimestamp>2018-06-03T12:05:33.8024053-06:00</ResponseTimestamp>
  <VehicleMonitoringDelivery version="1.3">
    <ResponseTimestamp>2018-06-03T12:05:33.8024053-06:00</ResponseTimestamp>
    <ValidUntil>2018-06-03T12:05:43.8024053-06:00</ValidUntil>
    <VehicleActivity>
      <RecordedAtTime>2018-06-03T12:05:33.8024053-06:00</RecordedAtTime>
    </VehicleActivity>
  </VehicleMonitoringDelivery>
</Siri>
```

Nueva York (MTA)

Ciudad/Región: Nueva York City (Estados Unidos)

Compañía/Organización: MTA Bus Time

Petición de Stop Monitoring

```
[HTTP GET]
http://bustime.mta.info/api/siri/stop-monitoring.json?key=xxxx-xxxx-xxxx-xxxx
&OperatorRef=MTA
&MonitoringRef=308209
&LineRef=MTA%20NYCT_B63
```

Respuesta de Stop Monitoring


```
{
  "Siri":{
    "ServiceDelivery":{
      "ResponseTimestamp":"2018-06-03T14:19:46.392-04:00",
      "StopMonitoringDelivery":[
        {
          "MonitoredStopVisit":[
            {
              "MonitoredVehicleJourney":{
                "LineRef":"MTA NYCT_B63",
                "DirectionRef":"0",
                "FramedVehicleJourneyRef":{
                  "DataFrameRef":"2018-06-03",
                  "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
083000_B63_112"
                },
                "JourneyPatternRef":"MTA_B630144",
                "PublishedLineName":"B63",
                "OperatorRef":"MTA NYCT",
                "OriginRef":"MTA_306619",
                "DestinationRef":"MTA_801007",
                "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                "SituationRef":[
                  {
                    "SituationSimpleRef":"MTA NYCT_191142"
                  }
                ],
                "Monitored":true,
                "VehicleLocation":{
                  "Longitude":-74.011831,
                  "Latitude":40.643652
                },
                "Bearing":43.79817,
                "ProgressRate":"normalProgress",
                "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_21300_B63-104",
                "VehicleRef":"MTA NYCT_383",
                "MonitoredCall":{
                  "ExpectedArrivalTime":"2018-06-03T14:48:50.568-04:00",
                  "ExpectedDepartureTime":"2018-06-03T14:48:50.568-04:00",
```

```
                    "Extensions":{
                      "Distances":{
                        "PresentableDistance":"2.8 miles away",
                        "DistanceFromCall":4546.73,
                        "StopsFromCall":21,
                        "CallDistanceAlongRoute":8811.23
                      }
                    },
                    "StopPointRef":"MTA_308209",
                    "VisitNumber":1,
                    "StopPointName":"5 AV/UNION ST"
                  },
                  "OnwardCalls":{

                  }
                },
                "RecordedAtTime":"2018-06-03T14:19:21.000-04:00"
              },
              {
                "MonitoredVehicleJourney":{
                  "LineRef":"MTA NYCT_B63",
                  "DirectionRef":"0",
                  "FramedVehicleJourneyRef":{
                    "DataFrameRef":"2018-06-03",
                    "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
082000_B63_110"
                  },
                  "JourneyPatternRef":"MTA_B630144",
                  "PublishedLineName":"B63",
                  "OperatorRef":"MTA NYCT",
                  "OriginRef":"MTA_306619",
                  "DestinationRef":"MTA_801007",
                  "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                  "SituationRef":[
                    {
                      "SituationSimpleRef":"MTA NYCT_191142"
                    }
                  ],
                  "Monitored":true,
                  "VehicleLocation":{
                    "Longitude":-74.014368,
                    "Latitude":40.641211
                  },
                  "Bearing":43.943977,
                  "ProgressRate":"normalProgress",
                  "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_42420_B63-110",
                  "VehicleRef":"MTA NYCT_422",
                  "MonitoredCall":{
                    "ExpectedArrivalTime":"2018-06-03T14:51:41.539-04:00",
                    "ExpectedDepartureTime":"2018-06-03T14:51:41.539-04:00",
                    "Extensions":{
                      "Distances":{
                        "PresentableDistance":"3.0 miles away",
                        "DistanceFromCall":4892.35,
                        "StopsFromCall":23,
                        "CallDistanceAlongRoute":8811.23
                      }
                    },
                    "StopPointRef":"MTA_308209",
                    "VisitNumber":1,
                    "StopPointName":"5 AV/UNION ST"
                  },
                  "OnwardCalls":{

                  }
                },
```

```
            "RecordedAtTime":"2018-06-03T14:19:16.000-04:00"
          },
          {
            "MonitoredVehicleJourney":{
              "LineRef":"MTA NYCT_B63",
              "DirectionRef":"0",
              "FramedVehicleJourneyRef":{
                "DataFrameRef":"2018-06-03",
                "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
085000_B63_114"
              },
              "JourneyPatternRef":"MTA_B630144",
              "PublishedLineName":"B63",
              "OperatorRef":"MTA NYCT",
              "OriginRef":"MTA_306619",
              "DestinationRef":"MTA_801007",
              "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
              "SituationRef":[
                {
                  "SituationSimpleRef":"MTA NYCT_191142"
                }
              ],
              "Monitored":true,
              "VehicleLocation":{
                "Longitude":-74.026597,
                "Latitude":40.621046
              },
              "Bearing":49.267895,
              "ProgressRate":"normalProgress",
              "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_44220_B63-114",
              "VehicleRef":"MTA NYCT_238",
              "MonitoredCall":{
                "ExpectedArrivalTime":"2018-06-03T15:05:58.717-04:00",
                "ExpectedDepartureTime":"2018-06-03T15:05:58.717-04:00",
                "Extensions":{
                  "Distances":{
                    "PresentableDistance":"4.6 miles away",
                    "DistanceFromCall":7397.24,
                    "StopsFromCall":34,
                    "CallDistanceAlongRoute":8811.23
                  }
                },
                "StopPointRef":"MTA_308209",
                "VisitNumber":1,
                "StopPointName":"5 AV/UNION ST"
              },
              "OnwardCalls":{

              }
            },
            "RecordedAtTime":"2018-06-03T14:19:31.000-04:00"
          },
          {
            "MonitoredVehicleJourney":{
              "LineRef":"MTA NYCT_B63",
              "DirectionRef":"0",
              "FramedVehicleJourneyRef":{
                "DataFrameRef":"2018-06-03",
                "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
086000_B63_107"
              },
```

```
                "JourneyPatternRef":"MTA_B630144",
                "PublishedLineName":"B63",
                "OperatorRef":"MTA NYCT",
                "OriginRef":"MTA_306619",
                "DestinationRef":"MTA_801007",
                "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                "SituationRef":[
                  {
                    "SituationSimpleRef":"MTA NYCT_191142"
                  }
                ],
                "Monitored":true,
                "VehicleLocation":{
                  "Longitude":-74.034403,
                  "Latitude":40.611954
                },
                "Bearing":49.710014,
                "ProgressRate":"normalProgress",
                "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_39780_B63-107",
                "VehicleRef":"MTA NYCT_230",
                "MonitoredCall":{
                  "ExpectedArrivalTime":"2018-06-03T15:12:24.502-04:00",
                  "ExpectedDepartureTime":"2018-06-03T15:12:24.502-04:00",
                  "Extensions":{
                    "Distances":{
                      "PresentableDistance":"5.3 miles away",
                      "DistanceFromCall":8605.15,
                      "StopsFromCall":39,
                      "CallDistanceAlongRoute":8811.23
                    }
                  },
                  "StopPointRef":"MTA_308209",
                  "VisitNumber":1,
                  "StopPointName":"5 AV/UNION ST"
                },
                "OnwardCalls":{

                }
              },
              "RecordedAtTime":"2018-06-03T14:19:43.000-04:00"
            },
            {
              "MonitoredVehicleJourney":{
                "LineRef":"MTA NYCT_B63",
                "DirectionRef":"0",
                "FramedVehicleJourneyRef":{
                  "DataFrameRef":"2018-06-03",
                  "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
087000_B63_108"
                },
                "JourneyPatternRef":"MTA_B630144",
                "PublishedLineName":"B63",
                "OperatorRef":"MTA NYCT",
                "OriginRef":"MTA_306619",
                "DestinationRef":"MTA_801007",
                "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                "OriginAimedDepartureTime":"2018-06-03T14:30:00.000-04:00",
                "SituationRef":[
                  {
                    "SituationSimpleRef":"MTA NYCT_191142"
                  }
                ],
```

```
      "Monitored":true,
      "VehicleLocation":{
        "Longitude":-74.029593,
        "Latitude":40.617555
      },
      "Bearing":229.39441,
      "ProgressRate":"normalProgress",
      "ProgressStatus":"prevTrip",
      "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_45420_B63-108",
      "VehicleRef":"MTA NYCT_245",
      "MonitoredCall":{
        "Extensions":{
          "Distances":{
            "PresentableDistance":"6.1 miles away",
            "DistanceFromCall":9848.5,
            "StopsFromCall":45,
            "CallDistanceAlongRoute":8811.23
          }
        },
        "StopPointRef":"MTA_308209",
        "VisitNumber":1,
        "StopPointName":"5 AV/UNION ST"
      },
      "OnwardCalls":{

      }
    },
    "RecordedAtTime":"2018-06-03T14:19:20.000-04:00"
  },
  {
    "MonitoredVehicleJourney":{
      "LineRef":"MTA NYCT_B63",
      "DirectionRef":"0",
      "FramedVehicleJourneyRef":{
        "DataFrameRef":"2018-06-03",
        "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-089000_B35_10"
      },
      "JourneyPatternRef":"MTA_B630144",
      "PublishedLineName":"B63",
      "OperatorRef":"MTA NYCT",
      "OriginRef":"MTA_306619",
      "DestinationRef":"MTA_801007",
      "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
      "OriginAimedDepartureTime":"2018-06-03T14:50:00.000-04:00",
      "SituationRef":[
        {
          "SituationSimpleRef":"MTA NYCT_191142"
        }
      ],
      "Monitored":true,
      "VehicleLocation":{
        "Longitude":-74.01623,
        "Latitude":40.639419
      },
      "Bearing":223.98701,
      "ProgressRate":"normalProgress",
      "ProgressStatus":"prevTrip",
      "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_41400_B35-10",
      "VehicleRef":"MTA NYCT_448",
```

```
      "MonitoredCall":{
        "Extensions":{
          "Distances":{
            "PresentableDistance":"7.8 miles away",
            "DistanceFromCall":12562.84,
            "StopsFromCall":58,
            "CallDistanceAlongRoute":8811.23
          }
        },
        "StopPointRef":"MTA_308209",
        "VisitNumber":1,
        "StopPointName":"5 AV/UNION ST"
      },
      "OnwardCalls":{

      }
    },
    "RecordedAtTime":"2018-06-03T14:19:13.000-04:00"
  },
  {
    "MonitoredVehicleJourney":{
      "LineRef":"MTA NYCT_B63",
      "DirectionRef":"0",
      "FramedVehicleJourneyRef":{
        "DataFrameRef":"2018-06-03",
        "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-090000_B35_35"
      },
      "JourneyPatternRef":"MTA_B630144",
      "PublishedLineName":"B63",
      "OperatorRef":"MTA NYCT",
      "OriginRef":"MTA_306619",
      "DestinationRef":"MTA_801007",
      "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
      "OriginAimedDepartureTime":"2018-06-03T15:00:00.000-04:00",
      "SituationRef":[
        {
          "SituationSimpleRef":"MTA NYCT_191142"
        }
      ],
      "Monitored":true,
      "VehicleLocation":{
        "Longitude":-74.012308,
        "Latitude":40.643193
      },
      "Bearing":224.03308,
      "ProgressRate":"normalProgress",
      "ProgressStatus":"prevTrip",
      "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_31980_B35-21",
      "VehicleRef":"MTA NYCT_481",
      "MonitoredCall":{
        "Extensions":{
          "Distances":{
            "PresentableDistance":"8.1 miles away",
            "DistanceFromCall":13097.34,
            "StopsFromCall":61,
            "CallDistanceAlongRoute":8811.23
          }
        },
```

```
                    "StopPointRef":"MTA_308209",
                    "VisitNumber":1,
                    "StopPointName":"5 AV/UNION ST"
                  },
                  "OnwardCalls":{

                  }
                },
                "RecordedAtTime":"2018-06-03T14:19:35.000-04:00"
              },
              {
                "MonitoredVehicleJourney":{
                  "LineRef":"MTA NYCT_B63",
                  "DirectionRef":"0",
                  "FramedVehicleJourneyRef":{
                    "DataFrameRef":"2018-06-03",
                    "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
091000_B63_106"
                  },
                  "JourneyPatternRef":"MTA_B630144",
                  "PublishedLineName":"B63",
                  "OperatorRef":"MTA NYCT",
                  "OriginRef":"MTA_306619",
                  "DestinationRef":"MTA_801007",
                  "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                  "OriginAimedDepartureTime":"2018-06-03T15:10:00.000-04:00",
                  "SituationRef":[
                    {
                      "SituationSimpleRef":"MTA NYCT_191142"
                    }
                  ],
                  "Monitored":true,
                  "VehicleLocation":{
                    "Longitude":-73.999844,
                    "Latitude":40.655178
                  },
                  "Bearing":224.03926,
                  "ProgressRate":"normalProgress",
                  "ProgressStatus":"prevTrip",
                  "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_42600_B63-106",
                  "VehicleRef":"MTA NYCT_354",
                  "MonitoredCall":{
                    "Extensions":{
                      "Distances":{
                        "PresentableDistance":"9.2 miles away",
                        "DistanceFromCall":14794.92,
                        "StopsFromCall":68,
                        "CallDistanceAlongRoute":8811.23
                      }
                    },
                    "StopPointRef":"MTA_308209",
                    "VisitNumber":1,
                    "StopPointName":"5 AV/UNION ST"
                  },
                  "OnwardCalls":{

                  }
                },
                "RecordedAtTime":"2018-06-03T14:19:29.000-04:00"
              },
```

```
            {
              "MonitoredVehicleJourney":{
                "LineRef":"MTA NYCT_B63",
                "DirectionRef":"0",
                "FramedVehicleJourneyRef":{
                  "DataFrameRef":"2018-06-03",
                  "DatedVehicleJourneyRef":"MTA NYCT_JG_B8-Sunday-
092000_B63_113"
                },
                "JourneyPatternRef":"MTA_B630144",
                "PublishedLineName":"B63",
                "OperatorRef":"MTA NYCT",
                "OriginRef":"MTA_306619",
                "DestinationRef":"MTA_801007",
                "DestinationName":"PIER 6 BKLYN BRIDGE PK via 5 AV",
                "OriginAimedDepartureTime":"2018-06-03T15:20:00.000-04:00",
                "SituationRef":[
                  {
                    "SituationSimpleRef":"MTA NYCT_191142"
                  }
                ],
                "Monitored":true,
                "VehicleLocation":{
                  "Longitude":-73.986137,
                  "Latitude":40.669371
                },
                "Bearing":229.94548,
                "ProgressRate":"normalProgress",
                "ProgressStatus":"prevTrip",
                "BlockRef":"MTA NYCT_JG_B8-Sunday_D_JG_43200_B63-113",
                "VehicleRef":"MTA NYCT_367",
                "MonitoredCall":{
                  "Extensions":{
                    "Distances":{
                      "PresentableDistance":"10.4 miles away",
                      "DistanceFromCall":16753.72,
                      "StopsFromCall":77,
                      "CallDistanceAlongRoute":8811.23
                    }
                  },
                  "StopPointRef":"MTA_308209",
                  "VisitNumber":1,
                  "StopPointName":"5 AV/UNION ST"
                },
                "OnwardCalls":{

                }
              },
              "RecordedAtTime":"2018-06-03T14:19:22.000-04:00"
            }
          ],
          "ResponseTimestamp":"2018-06-03T14:19:46.392-04:00",
          "ValidUntil":"2018-06-03T14:20:46.392-04:00"
        }
      ],
```

```
      "SituationExchangeDelivery":[
        {
          "Situations":{
            "PtSituationElement":[
              {
                "PublicationWindow":{
                  "StartTime":"2018-06-03T00:00:00.000-04:00",
                  "EndTime":"2018-06-03T23:59:00.000-04:00"
                },
                "Severity":"undefined",
                "Summary":"B63 buses rerouted from 5 Av between Bay Ridge Av and
86 St",
                "Description":"B63 buses rerouted from 5 Av between Bay Ridge Av
and 86 St 9 AM to 10 PM, Sunday, Jun 3Due to a local event and clean up, buses
run via 4 Av making corresponding stops in both directions.Show Reroute
DetailsNorthboundVia 5 AvLeft on 86 StRight on 4 AvRight on Bay Ridge AvLeft on
5 Av then regular routeSouthboundVia 5 AvRight on Bay Ridge AvLeft on 4 AvLeft
on 86 StRight on 5 Av then regular route",
                "Affects":{
                  "VehicleJourneys":{
                    "AffectedVehicleJourney":[
                      {
                        "LineRef":"MTA NYCT_B63",
                        "DirectionRef":"0"
                      },
                      {
                        "LineRef":"MTA NYCT_B63",
                        "DirectionRef":"1"
                      }
                    ]
                  }
                },
                "Consequences":{
                  "Consequence":[
                    {
                      "Condition":"diverted"
                    }
                  ]
                },
                "CreationTime":"2018-06-03T00:00:00.000-04:00",
                "SituationNumber":"MTA NYCT_191142"
              }
            ]
          }
        }
      ]
    }
  }
}
```