PREDIM

Etude Subventionnée - Lot 1

VERSION 1.1 du 8/02/2005

Référence : 226/75/01/2004/63-43 Article 05

- Glossaire -

AO	Autorité Organisatrice des transports
AXIS	API Java pour les Web Services et le protocole SOAP
CHOUETTE	Création d'Horaires avec un OUtil d'Échange de données TC selon le format Trident Européen
CORBA	Common Object Request Brocker Architecture
ERT	Exploitant d'un Réseau de Transport
MT	MT System - mobiliTime®
PREDIM	Plate-forme de Recherche et d'Expérimentationpour le Développement de l'Information Multimodale
NAPTAN	National Public Transport Access Node
NPTG	National Public Transport Gazetteer
RI	Moteur ou Calculateur de Recherche d'Itinéraires
RIDER	Recherche d'Itinéraires multimoDale Etendue et temps Réel
SIM	Système d'Information Multimodal
SOAP	Simple Object Access protocol
STI	Système de Transport Intelligent
TC	Transport en Commun
TRANSMODEL	Modèle de données de référence pour le Transport Public, devenu une norme expérimentale ENV12896
TRIDENT	TRansport Intermodality Data sharing and Exchange NeTwork Modèle d'échange et de partage d'information multimodale et intermodale
XML	eXtended Markup Language
WSDL	Web Services Description Language

- Historique -

Édition	Date	Opération / Libellé	Auteurs
1.0	16/12/2004	Création / Etude de Journey Web messages et protocole, Spécification des données et messages sur la base de Trident	P. ISORCE
1.1	08/02/2005	Modification / Protocole de communication et interfaçage avec les Services Web, Interfaçage à Rider, exemple de Service Web avec WSDL	P. ISORCE T. ANH NGUEN

- Sommaire -

1 Objectif 5
2 Périmètre 5
3 Etude de JourneyWeb 7
3.1 Introduction 7
3.2 Principe de fonctionnement 7
3.3 Description d'un Scénario 9
3.4 Spécification XML 10
3.5 Les Requêtes aux Services de Calcul d'Itinéraire 10
3.6 Les Réponses Des Services de Calcul d'Itinéraire 17
3.7 Diagnostic 23
4 Utilisation de Trident pour la Spécification 24
4.1 Le Calcul d'Itinéraire 24
4.2 L'Outil CHOUETTE 24
4.3 Réutiliser Trident comme Base de Spécifications 24
4.4 Comparaison de JourneyWeb et de Trident 25
4.5 Types de messages 26
4.6 Description des messages 26
4.7 mise en œuvre d'un Calcul d'Itinéraire 33
4.8 mise en œuvre d'une Recherche d'Horaires 33
5 Choix des Services Web pour le Protocole de Communication 34
5.1 Modèle de Communication 34
5.2 Interopérabilité 35
5.3 Décrire une spécification d'interface avec WSDL 36
5.4 Proposition de Mise en Œuvre 38
6 Interfaçage de Rider comme Exemple de Logiciel de Calcul d'Itinéraire 39
6.1 Introduction 40
6.2 Interface d'Accès via SOAP 40
6.3 Gestion des droits d'accès 40
6.4 Recherche d'adresse 40
6.5 Recherche des Arrêts 41
6.6 Calcul d'itinéraire 42
6.7 Diagnostic 44

- Figures -

Figure 1 - Positionnement de l'étude sur un réseau global 6
Figure 2 – diagramme général de communication 8
Figure 3 - Diagramme de séquence du protocole 8
Figure 4 - Schéma du scénario présenté 9
Figure 5 - Modèle du protocole JourneyWeb 10
Figure 6 - Diagramme d'une requête 10
Figure 7 Diagramme de PointsRequest 11
Figure 8 - Diagramme de JourneyRequest 11
Figure 9 - Diagrammes de localisation 12
Figure 10 - Diagramme de limitation de recherche 13
Figure 11 - Diagramme de StopTimetableRequest 14
Figure 12 - Diagramme ServiceTimetableRequest 15
Figure 13 - Diagramme de StopEventsRequest 16
Figure 14 - Diagramme de ServiceRequest 17
Figure 15 - Diagramme de OperatorsRequest 17
Figure 16 - Diagramme de Response 18
Figure 17 - Diagramme de PointsResponse 19
Figure 18 - Diagramme de JourneyResponse 20
Figure 19 - Diagramme de TimedLeg 21
Figure 20 - Diagrammes de OriginTimeGroup et BoardTimeGroup 21
Figure 21 - Diagramme de TimetableResponse 22
Figure 22 - Diagramme de ServiceResponse 23
Figure 23 - Diagramme de OperatorsResponse 23
Figure 24 - Diagramme ObjectRequest 26
Figure 25 - Le type ObjectResponse 27
Figure 26 - Le type PtStopPointsRequest 27
Figure 27 - Le type PtStopPointsResponse 27
Figure 28 - Le type ItineraryCalculationRequest 28
Figure 29 - Le type ItineraryCalculationResponse 28
Figure 30 – Le type Itinerary 29
Figure 31 - Le type TripSegment 30
Figure 32 – Le type TimeTableRequest 31
Figure 33 - Le type TimeTableResponse 31
Figure 34 - Le type Timetable 32
Figure 35 - Diagramme de classe de ITridentPeer 34
Figure 36 - Annuaire des Services Web 35
Figure 37 - Transparence de SOAP et des Services Web 36
Figure 38 – Structure générale d'un message 36
Figure 39 - Description d'un Service Web avec WSDL 37
Figure 40 - Exemple de description d'un port d'un Service Web de calcul d'itinéraire 38
Figure 41 - Service Web comme interface logicielle 38
Figure 42 - Assemblage des Services Web pour un calcul d'itinéraire global 39
Figure 43 - Diagramme de la méthode StreetRequest 40
Figure 44 - Diagramme de la méthode StreetResponse 41
Figure 45 - Diagramme de la méthode VpPointRequest 41
Figure 46 - Diagramme de la méthode VpPointResponse 41
Figure 47 - Diagramme de la méthode TcVpRequest 42
Figure 48 - Diagramme de la méthode TcVpResponse 43
Figure 49 - Diagramme de la méthode TcVpResponse 44

1 OBJECTIF

L'étude proposée a pour objectif la standardisation des données destinées à être échangées avec les multiples logiciels de calcul d'itinéraire réparti, liés aux métiers des transports dans le cadre d'une demande d'un itinéraire global.

Une approche générale des systèmes de transports intelligents suppose la prise en compte de la fonction d'optimisation des itinéraires d'une façon globale, afin d'éviter que la fonction ne reste centralisée chez l'exploitant. Cette décentralisation de l'information permet ainsi de concevoir un nouveau mode d'optimisation des itinéraires, multimodal, capable par exemple de renseigner les usagers sur leur trajet de porte à porte, quels que soient la destination et le moyen de transport choisi.

Cette étude s'inscrit dans une réflexion globale d'élaboration du futur Système d'Information Multimodal (SIM) de MT System – mobiliTime® et s'appuie sur les recommandations du projet Actif d'architecture de Système de Transport Intelligent (STI), sur les spécifications de la pré-norme Européenne d'échange de données Trident et les résultats de l'étude antérieure et labellisées PREDIM, l'outil CHOUETTE, qui est proposé par MT System pour son nœud d'échange de données.

L'étude conduira d'une part à la fourniture d'un document spécifiant les échanges des flux d'information entrant et sortant entre un calculateur d'itinéraire et un réseau d'échange de données Trident et d'autre part un prototype « démonstrateur ».

D'autre part cette étude vise aussi à développer de nouveaux services pour l'outil CHOUETTE. Le fait de rendre public (sous la responsabilité de la PREDIM) cette étude critique, ces interfaces et le guide d'implémentation, favoriseront l'interfaçage d'autres outils logiciels de calcul d'itinéraire et procureront alors à l'outil CHOUETTE des perspectives plus larges d'utilisation.

2 PERIMETRE

Le thème particulier que nous proposons dans cette étude concerne le calcul d'itinéraire global limité au domaine des Transports en Commun (TC) et à la standardisation des échanges de données entre le réseau d'échange de données mobiliTime®, réseau ouvert de manière native qui s'appui notamment sur la prénorme Trident, et le calculateur d'itinéraire de la société Dryade utilisé sur le site www.citefutee.com.

Le périmètre métiers couvert par l'étude est celui de l'information voyageurs pour les usagers des TC.

Les domaines d'information voyageurs couverts par l'étude sont : les réseaux, les horaires et les itinéraires avec un minimum de deux modes de transport. Les données utilisées seront soit issues de jeux d'essaies extraits du réseau de la RATP (données utilisées dans le cadre d'un projet INSA) soit fournies par une des AO (Conseil Général) avec lesquelles nous collaborons.

L'étude se décompose en trois lots distincts :
Lot 1 :

- D'extraire des documents de normalisation Trident et TransXchange, les éléments pertinents et de les synthétiser dans un ensemble de schémas et modèles compréhensibles et **vulgarisés** au sein d'un document de spécifications.
- De faire un **diagnostic**, au niveau des interfaces, de l'outil de calcul d'itinéraire de la société Dryade et de spécifier une standardisation des échanges de données aussi bien en entrée qu'en sortie de ces calculateurs.

Lot 2 :

- D'implémenter par extension de l'outil CHOUETTE, un service d'échange de données interfaçant un logiciel de calcul d'itinéraire et jouant le rôle d'un **traducteur** de la base de données. Cette implémentation sera étendue au domaine du calcul d'itinéraire global. A ce

titre les sources et sa documentation seront mis à disposition de la PREDIM afin que d'autres partenaires puissent à leur tour implémenter leur propre traducteur interfacé à leur logiciel de calcul d'itinéraire.

Lot 3 :

- D'évaluer et de consigner des **préconisations** de déploiement, des éléments de coût et de délai. Ces recommandations pourront servir de guide méthodologique pour l'élaboration de cahier des charges préalable à la mise en œuvre d'un système d'information multimodal.

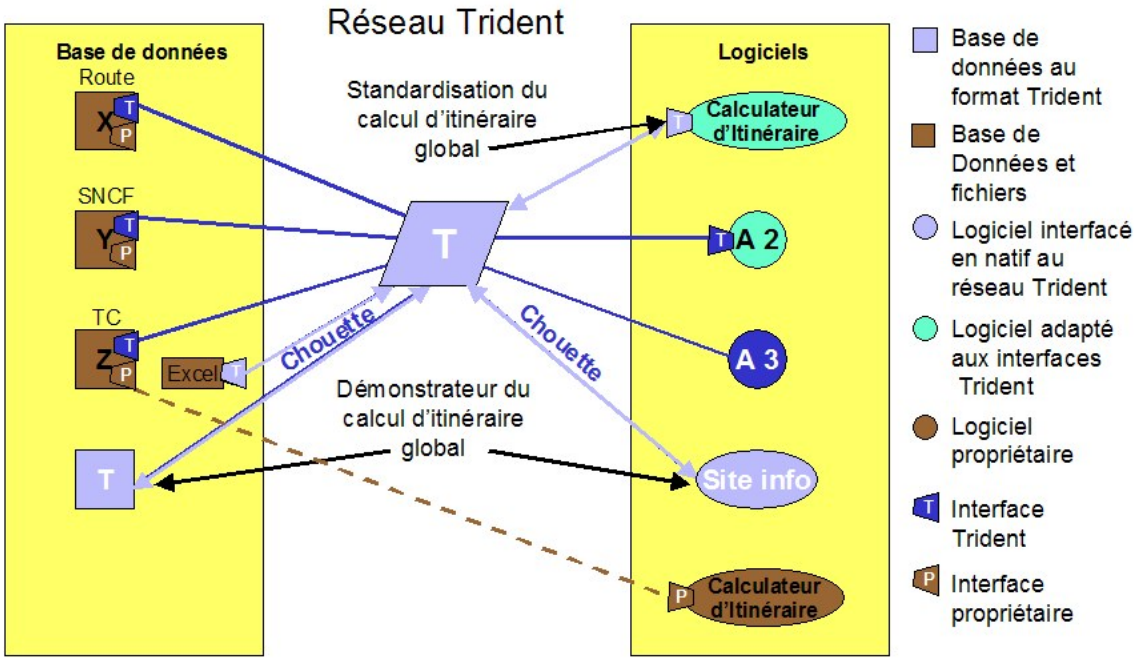


Figure 1 - Positionnement de l'étude sur un réseau global

Nous allons donc commencer par une étude du projet Anglais, **JourneyWeb** (sous-projet d'un vaste projet, Transport Direct).

On ne reviendra pas sur le projet Transport Direct et, TransXchange initié par le gouvernement anglais en 1998. On pourra se référer par exemple au rapport d'étude sur ce projet réalisé par le CETE Méditerranée et publié par le CERTU. Dans ce document on trouve notamment une description claire du contexte anglais relatif à leur politique de mise en place d'un système d'information multimodal étendu à l'ensemble du Royaume Uni, ainsi qu'une analyse technique des deux référentiels, NPTG et NaPTAN .

Deux autres études sur les expériences allemande et suisse et les projets respectifs, Delphi et Netzwerk Direct ont aussi été publiées par le CERTU. On notera dans ces études que les projets allemand et suisse ont des architectures similaires à celle du projet JourneyWeb.

3 ETUDE DE JOURNEYWEB

3.1 INTRODUCTION

JourneyWeb se présente comme un protocole ouvert et indépendant qui doit permettre à de multiples logiciels de calcul d'itinéraires d'interagir entre eux ou d'être interrogés par un système fédérateur, dans le but de constituer un itinéraire à un niveau géographique national et mettant en jeu plusieurs systèmes d'information transport, implantés localement au niveau d'une région ou d'un département.

3.2 PRINCIPE DE FONCTIONNEMENT

Pour fonctionner, JourneyWeb utilise en outre deux référentiels (bases de données) :

- un répertoire national des noms de lieux (villes, villages, hameaux, point d'intérêts, etc.), pour savoir quel système distant contacter dans le cas d'un déplacement longue distance mettant en jeu plusieurs calculateurs **NPTG**, (National Public Transport Gazetteer)
- un répertoire local qui identifie l'ensemble des points d'accès au réseau (arrêts, entrée de gares, etc.) que couvre le calculateur local d'itinéraire **NaPTAN**, (National Public Transport Access Node)

Avec JourneyWeb, chaque calculateur d'itinéraire local est capable d'effectuer une recherche d'itinéraire d'un point de départ de sa zone de couverture, jusqu'à un point d'échange national dans une zone couverte par un calculateur distant. Ainsi chaque calculateur local contient aussi les informations nationales sur les trains et les autocars interurbains.

Un point d'échange national (National Exchange Point) est un arrêt physique sur le réseau interurbain national (par exemple un pôle d'échange, une gare de train ou routière), représenté par un identifiant dans le NPTG. Le NPTG contient la liste des localités ainsi que la liste des points d'échange pour chacune de ces localités.

Le protocole fonctionne en mode requête/réponse et comporte six requêtes élémentaires :

- POINTS (points d'arrêts) : demande la liste des arrêts au sein d'une localité du référentiel NPTG, ou à partir des coordonnées d'un point particulier,
- JOURNEYS (itinéraire) : demande un calcul un itinéraire entre plusieurs points, à l'origine ou à la destination,
- TIMETABLES (tables d'horaires): demande une matrice composée d'horaires, horaires de départ, d'arrivée et de passage aux points d'arrêts,
- STOPEVENTS (événements) : demande une liste des événements qui sont déclarés sur un point d'arrêt ou une liste de points d'arrêts,
- SERVICES (services, missions et courses) : demande une liste des services assurant les horaires de l'itinéraire recherché
- OPERATORS (exploitants) : demande une liste des exploitants ou opérateurs qui assurent un service.

On verra dans la suite du document que le service TIMETABLES se décompose en deux sous-services et donc en deux requêtes différentes, *StopTimetable* et *ServiceTimetable*.

Le protocole fonctionne dans un mode client-serveur. Chaque message échangé dans le protocole est soit une question (requête), soit une réponse comme le montre le diagramme suivant.

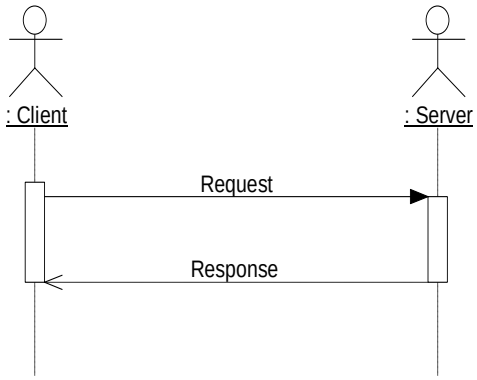


Figure 2 – diagramme général de communication

Les requêtes sont échangées par exemple à l'aide d'un formulaire Web de type POST et via le protocole de communication HTTP du Web. Mais on peut implémenter JourneyWeb sur d'autres modes de communication comme par exemple, le protocole de messagerie SMTP ou le protocole des services Web, SOAP.

La spécification de JourneyWeb ne précise rien sur ce type d'implémentation et ne fournit aucun outils spécifiques. La figure suivante illustre un diagramme de séquence de mise en œuvre de ce protocole.

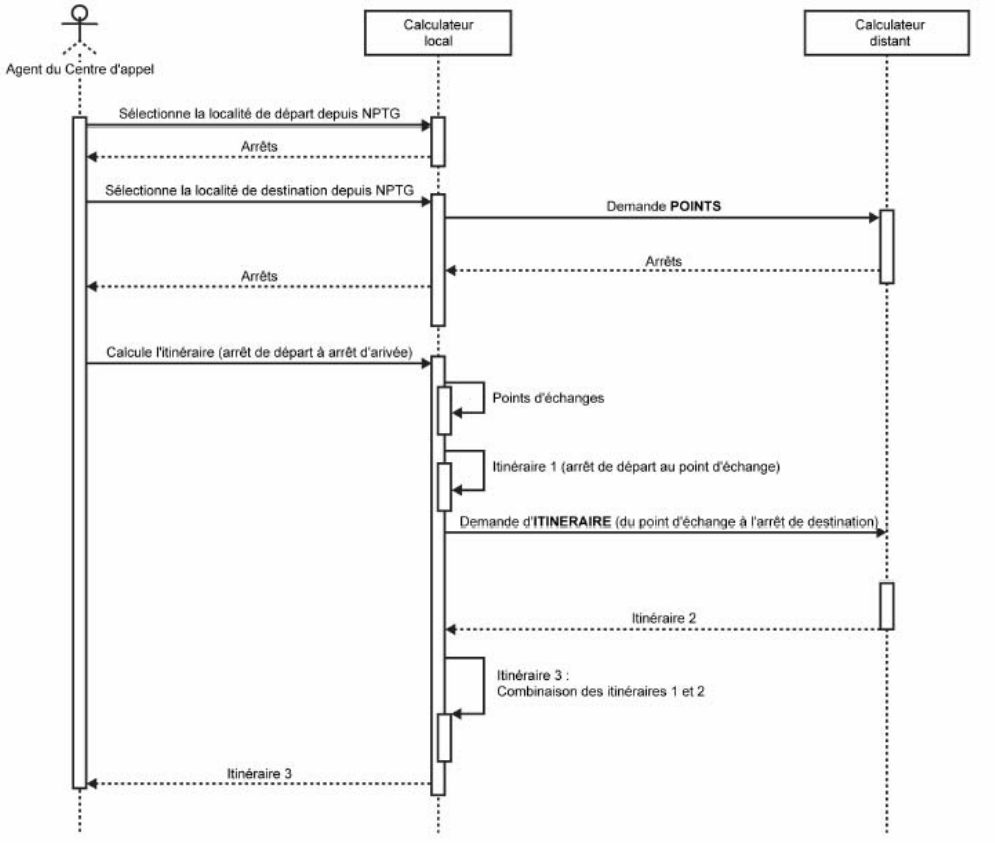


Figure 3 - Diagramme de séquence du protocole

3.3 DESCRIPTION D'UN SCENARIO

Prenons un exemple de fonctionnement de ce scénario illustré dans la figure précédente.
Un usager du Royaume Uni contacte un centre d'appel ou se connecte au portail Web de Transport Direct, et constitue sur la page Web, une demande d'itinéraire d'une localité de départ (Kings Lynn par exemple) jusqu'à un arrêt situé à proximité d'une localité distante (Elgin en Ecosse).

Usager	Calcul d'Itinéraire JourneyWeb
L'usager saisit Kings Lynn comme ville de départ	Recherche dans le référentiel local de NPTG et vérifie que Kings Lynn est une localité locale. Renvoie à l'usager la liste des points d'arrêts de Kings Lynn.
L'usager choisit un arrêt dans la liste	
L'usager saisit Elgin comme ville de destination	Recherche dans le référentiel local de NPTG et vérifie qu'Elgin est une localité distante. Extrait à partir de NPTG l'adresse URL du calculateur distant. Exécute une requête de type POINTS au calculateur d'itinéraire distant pour obtenir la liste des arrêts à Elgin. Renvoie à l'usager la liste des arrêts
L'usager choisit un arrêt dans la liste retournée	
L'usager demande le calcul du trajet	Exécute une requête de type POINTS pour obtenir les points d'échange pour l'arrêt distant sélectionné. Vérifie si la zone de destination est adjacente à la zone de départ. En l'occurrence, la réponse est « non ». Exécute ensuite une recherche d'itinéraire sur le calculateur local entre l'arrêt de départ à Kings Lynn et le point d'échange de la localité distante (Elgin). Exécute une requête de type JOURNEYS au calculateur distant entre les points d'échange distants et l'arrêt de destination à Elgin. Fusionne les résultats pour donner un itinéraire complet.
L'usager lit le résultat	

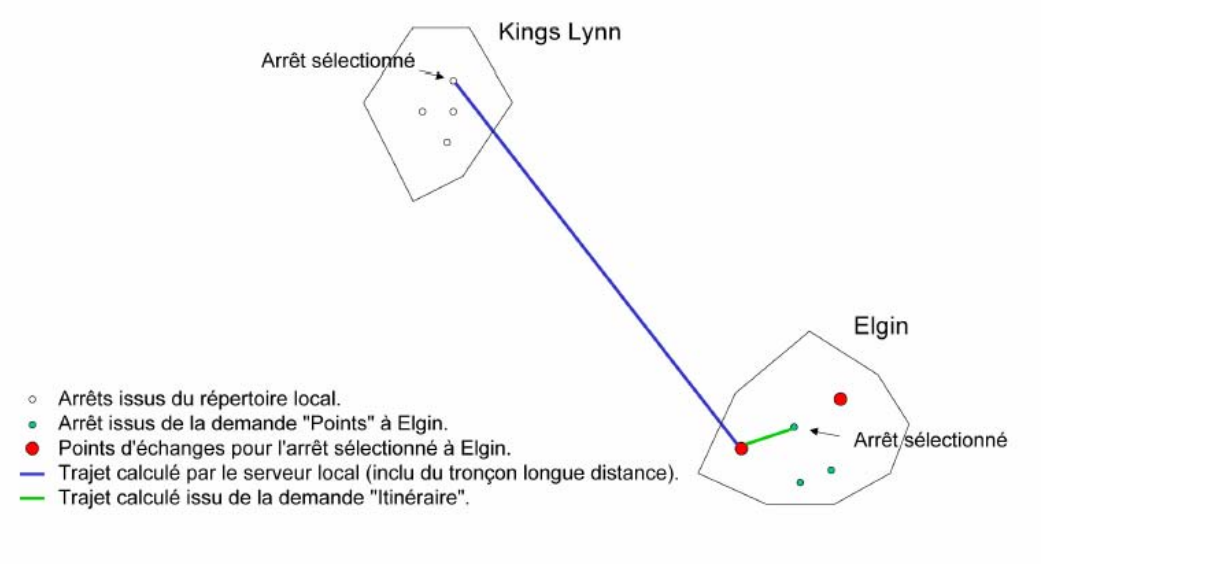


Figure 4 - Schéma du scénario présenté

3.4 SPECIFICATION XML

On n'examinera pas en détail toutes les spécifications en **XML Schema** du protocole JourneyWeb mais on regardera principalement quelques uns de ces modèles ou diagrammes **UML**.

On peut représenter le modèle général du protocole JourneyWeb par le diagramme suivant.

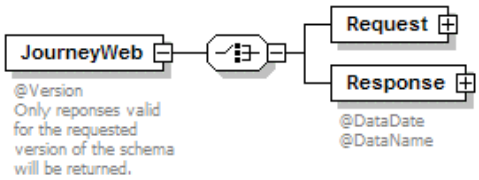


Figure 5 - Modèle du protocole JourneyWeb

L'élément XML principal de JourneyWeb doit contenir une référence de version. Dans toute demande formulée dans une requête, ceci indique au serveur quelle version du schéma est utilisée par le client.

Le serveur doit renvoyer une réponse qui est construite en utilisant la même version de schéma. Si la version de schéma n'est pas supportée, alors un message d'erreur de type « schéma non supporté" doit être retourné dans la réponse. La réponse doit contenir la date et l'identifiant de la base de données qui a servi à produire les résultats. Ceci peut prévenir le client du service que certaines données seront provisoires ou peut être périmées.

3.5 LES REQUETES AUX SERVICES DE CALCUL D'ITINERAIRE

La langue peut être spécifiée dans la requête à l'aide du paramètre **Language**. Si c'est le cas tous les noms des points d'arrêts par exemple, doivent être retournés par le service dans la langue spécifiée. Cette particularité a été mise en place afin de s'adapter au contexte multi-lingue de l'angleterre.

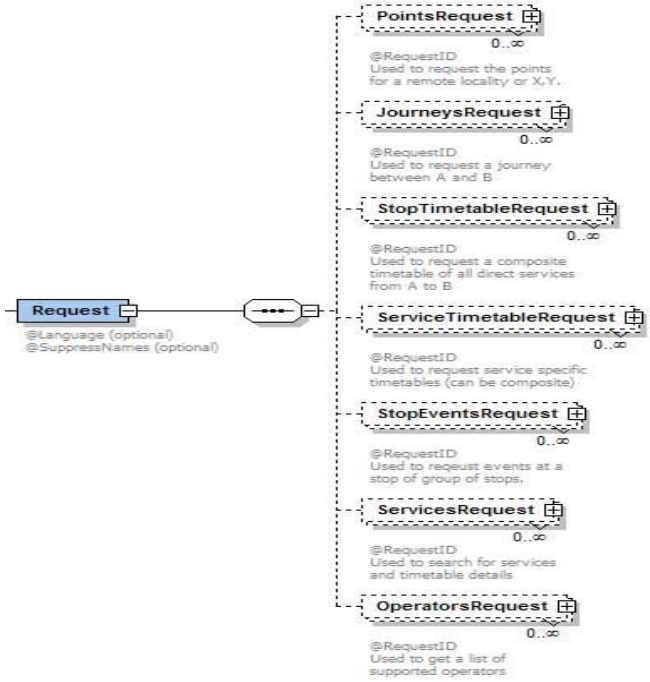


Figure 6 - Diagramme d'une requête

La figure précédente montre toutes les requêtes que peut supporter le protocole JourneyWeb. Plusieurs requêtes peuvent être effectuées dans la même transaction. En l'absence d'un attribut permettant d'identifier de manière unique chaque requête, un **RequestID** est alors utilisé.

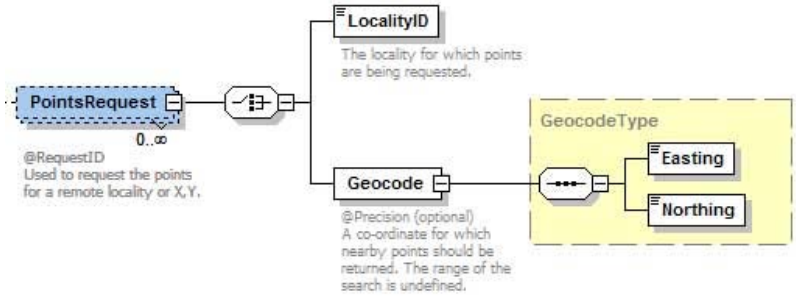


Figure 7 Diagramme de PointsRequest

Cette requête est utilisée pour demander tous les points dans une localité ou à proximité d'un point géodésique. Un point géodésique est défini par ses coordonnées latitude et longitude.

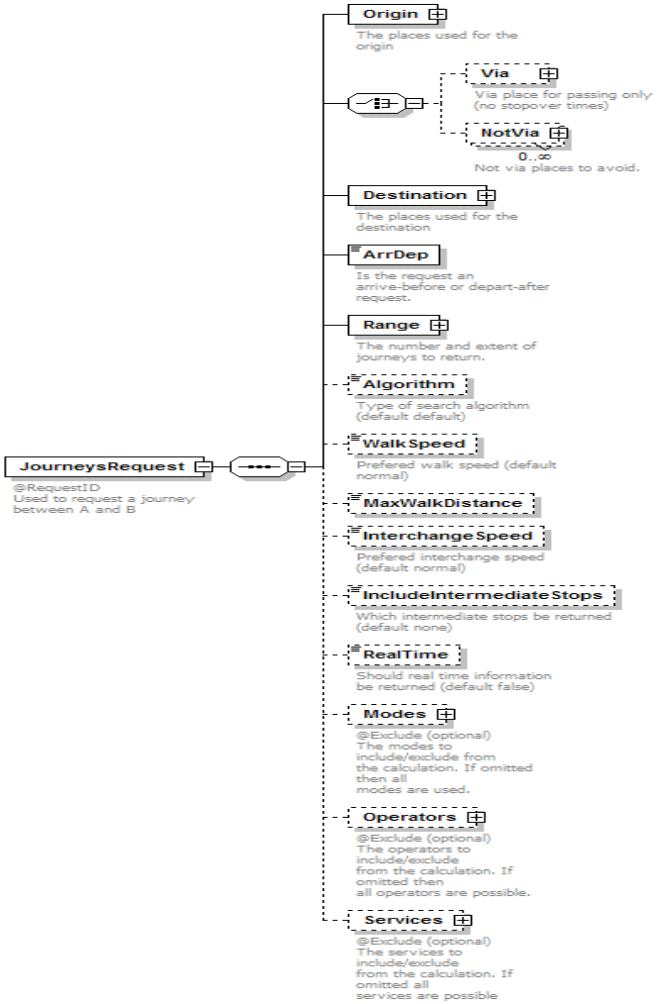


Figure 8 - Diagramme de JourneyRequest

La figure précédente illustre le diagramme de la requête précédente, JourneyRequest. Cette requête assure le service d'une demande d'un calcul d'itinéraire.

Le paramètre *RequestID* doit être défini pour bien distinguer deux requêtes dans une même transaction.

D'autres paramètres sont obligatoires comme le montre le tableau suivant :

Paramètre	Définition
ArrDep	Indique si la requête précise un départ au plus tôt, ou une arrivée au plus tard
Origin	Une origine doit être définie si on cherche un itinéraire de type « départ après »
Destination	Une destination doit être définie si on veut un itinéraire de type « arrivée avant »
Range	Une séquence, un segment ouvert, semi-fermé,

Une origine ou une destination peut être définie selon trois manières :

- Un ou plusieurs identifiant de points d'arrêts (IDs du référentiel de NaPTAN),
- Un ou plusieurs codes géodésiques
- Un ou plusieurs identifiants de localité (Ids du référentiel NPTG)

Si la localisation est indiquée que ce soit pour l'origine ou pour la destination

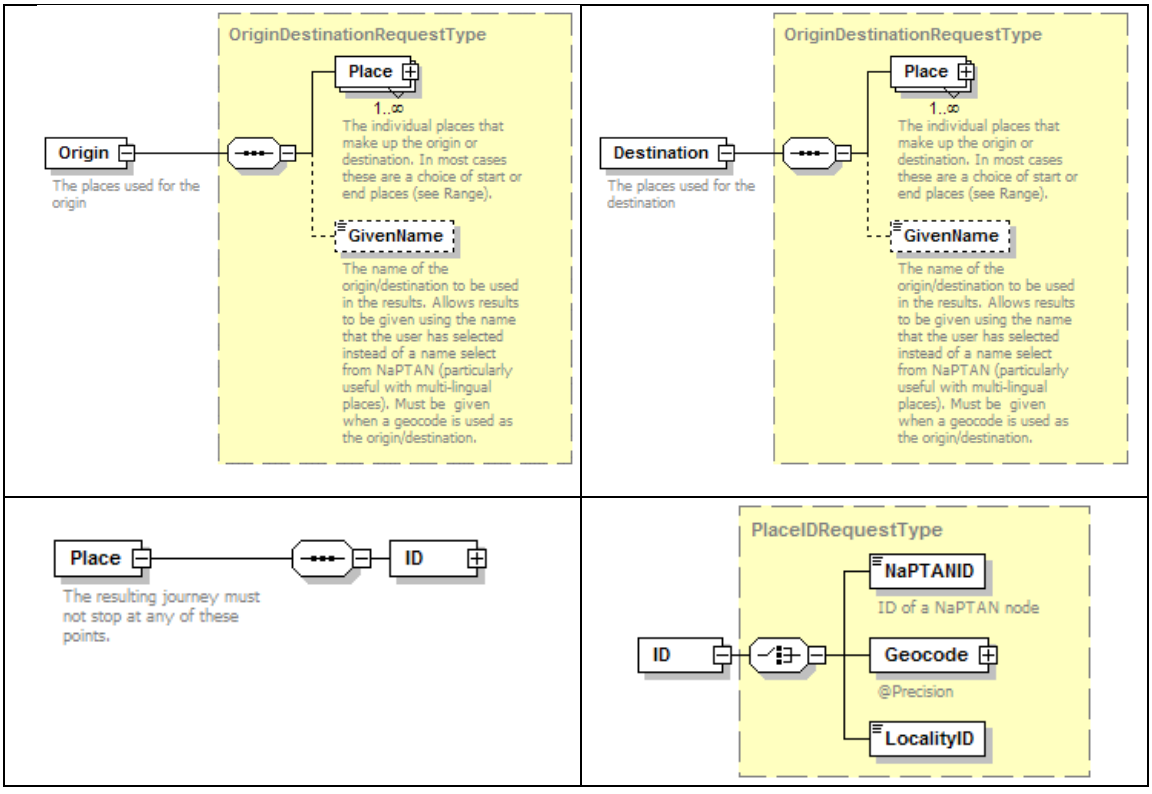


Figure 9 - Diagrammes de localisation

Le tableau suivant illustre les limitations que l'on peut fixer afin de restreindre l'étendue d'une recherche d'itinéraire, d'en fixer les bornes en terme de durée et/ou de localisation. Cette limitation est d'autant plus importante si le service de calcul d'itinéraire est peu performant lorsque une requête est trop ambiguë.

Paramètre	Définition
Sequence	On peut fixer le nombre de résultats souhaités
Interval	Tous les itinéraires seront évalués selon l'intervalle de temps
LimitedInterval	On peut limiter l'intervalle afin de borner le nombre de recherche d'itinéraires

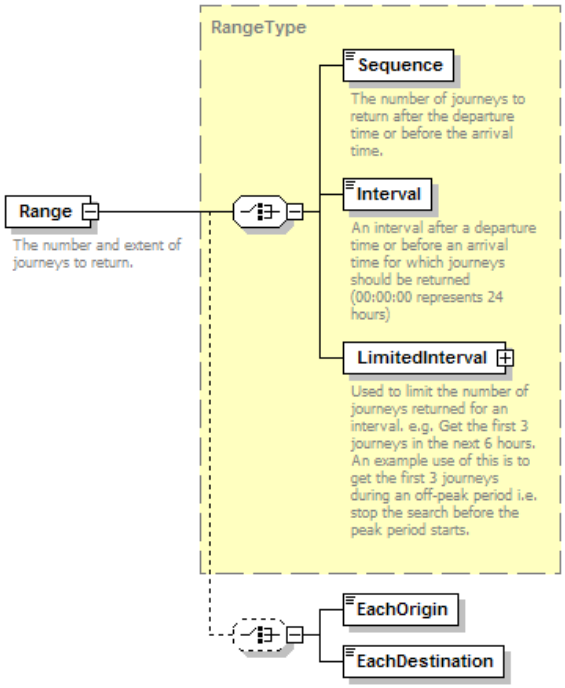


Figure 10 - Diagramme de limitation de recherche

La limitation à l'aide d'un intervalle de temps peut être utilisée pour se fixer par exemple un nombre maximum de 5 solutions de recherche d'itinéraires, dans l'heure suit celle recherchée, ou limiter à une solution si on étend la période de temps à une demi-journée (12 heures).

Les paramètres suivants sont optionnels. Ils peuvent être implémentés par exemple pour fournir au service des critères particuliers.

Paramètre	Définition
Via	On peut préciser un point de passage obligatoire
Not Via	On peut exclure une liste de points par lesquels on ne souhaite pas passer
Modes	Une liste de modes de transport à inclure ou à exclure dans la recherche
Operators	Une liste d'opérateurs à inclure ou à exclure dans la recherche
Services	Une liste de services à inclure ou à exclure dans la recherche
Algorithm	Le type d'algorithme à utiliser, le plus rapide, le moins de marche à pied,…
Walk Speed	La vitesse de marche à pied
Max Walk Distance	La distance maximum de marche à pied en mètres
Interchange Speed	La vitesse de changement correspondance
Real Time	On doit recherche les temps réels si possible
Intermediate Stops	

Une demande d'horaires, de passage, de départ et d'arrivée peut être formulée à l'aide de la requête *StopTimetableRequest*. On peut préciser un groupe de points à l'origine ou à la destination. Par

exemple on peut préciser pour tous les bus d'une gare routière, ou encore pour tous les arrêts à proximité d'une localité.

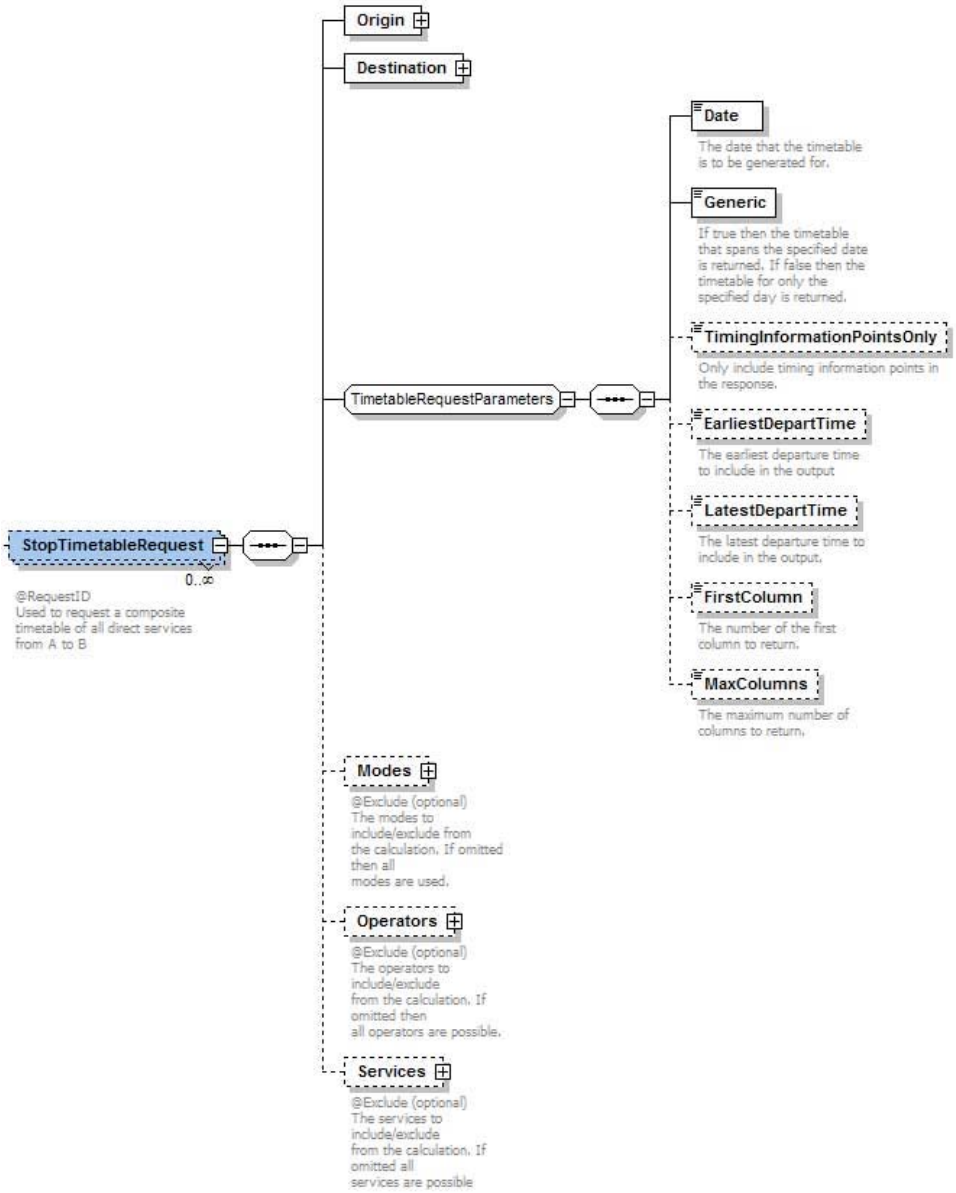


Figure 11 - Diagramme de StopTimetableRequest

La matrice des horaires résultante de cette requête sera un ensemble comprenant tous les services qui relient directement les arrêts précisés, ceci quelque soit le mode indiqué. Cette table des horaires sera triée et paginée pour permettre un affichage par exemple.

L'intervalle de temps de l'horaire est défini en utilisant la date et des paramètres génériques. Par exemple, si *Generic* est positionné à *false* alors l'horaire portera sur le jour de la date indiquée. Si *Generic* est *true* alors on retournera un horaire qui sera à cheval sur la date indiquée. Ceci est par exemple utile lorsque un horaire est valable du lundi au vendredi, du lundi au samedi, ou le dimanche, les jours fériés.

Les paramètres suivants sont optionnels. Ils peuvent être implémentés par exemple pour fournir au service des critères supplémentaires et permettant de limiter le volume de données dans le résultat de la requête.

Paramètre	Définition
TimingInformationPointsOnly	On fournit seulement les informations horaires
EarliestDepartTime	Inclure dans le résultat le premier départ
LatestDepartTime	Inclure dans le résultat le dernier départ
FirstColumn	L'indice de la première colonne
MaxColumns	Le nombre maximum de colonnes
Modes	Une liste de modes de transport à inclure ou à exclure dans la recherche
Operators	Une liste d'opérateurs à inclure ou à exclure dans la recherche
Services	Une liste de services à inclure ou à exclure dans la recherche

Une requête vers un service de recherche d'horaires peut renvoyer un très grand nombre de données. Aussi des mécanismes doivent être fournis pour contraindre et limiter le champ d'exécution de la requête. Par exemple, l'horaire du premier départ, du dernier, le nombre maximum de colonnes peuvent être implémentés pour réduire considérablement le volume de données des résultats.

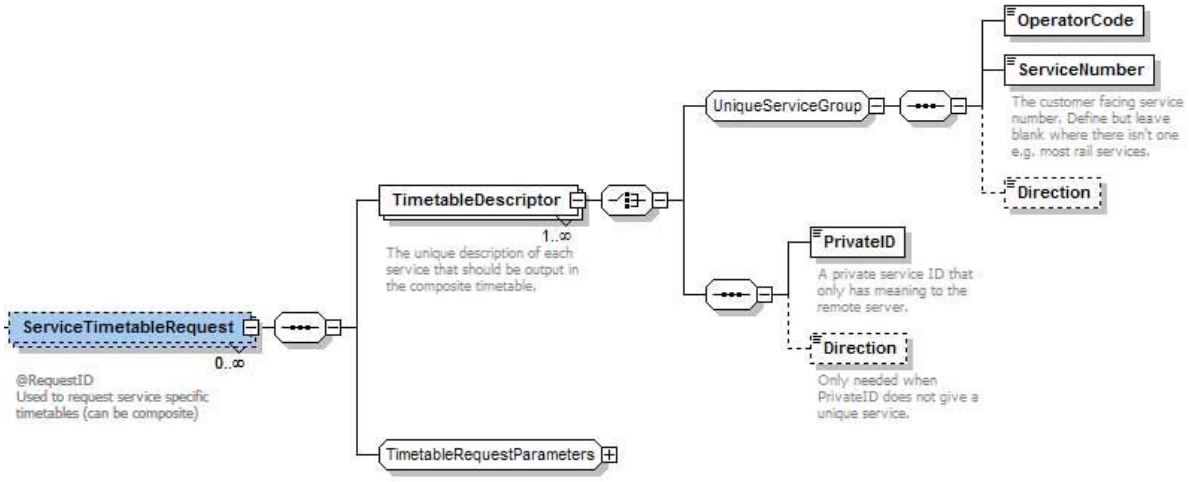


Figure 12 - Diagramme ServiceTimetableRequest

On retrouve dans cette requête la plupart des paramètres *TimeTableRequestParameters* que l'on a décrit dans la requête précédente.

Ce type de requête peut être utilisé pour rechercher quels sont les services qui sont assurés relativement aux itinéraires et aux horaires qui nous intéressent.

On peut formuler ce type de requête de deux façons :

- En indiquant le code d'un opérateur et un numéro de service,
- En indiquant un identifiant de service propre au système

La requête suivante permet d'interroger le système pour obtenir une liste des **événements** qui sont déclarés sur des arrêts. La figure suivante illustre le diagramme de cette requête.

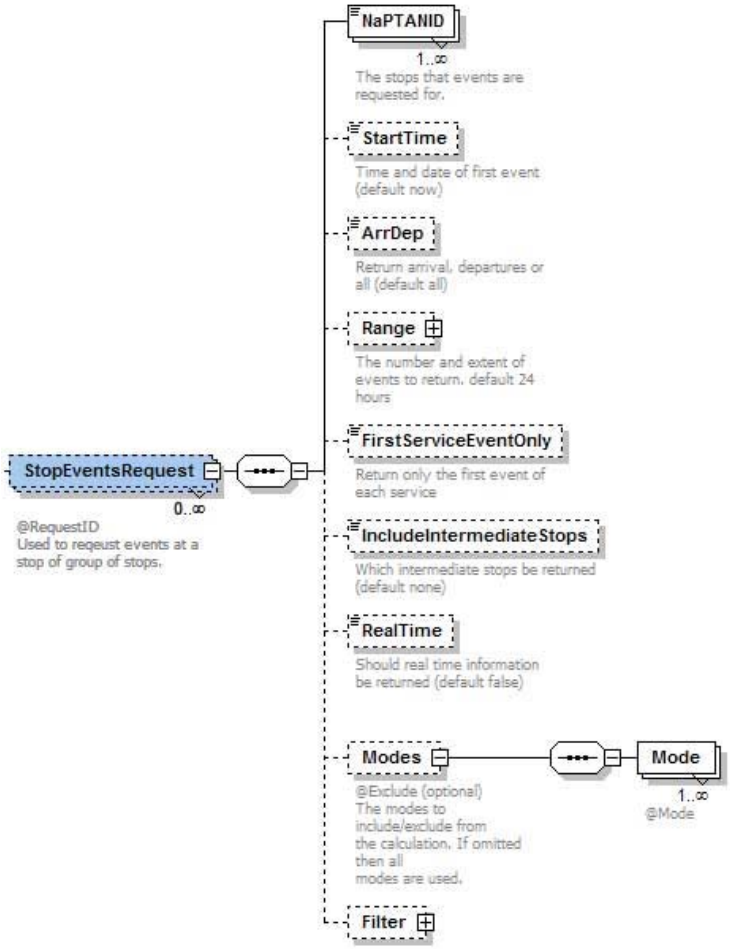


Figure 13 - Diagramme de StopEventsRequest

Une heure de début et un intervalle de temps peuvent être précisés. Par défaut, une période de temps de 24h à partir de l'heure actuelle sera utilisée.

Si le paramètre *FirstServiceEventOnly* est positionné à true, alors seulement le premier événement de chaque service sera retourné par le service.

Le paramètre *ArrDep* peut être utilisé pour indiquer si l'on souhaite obtenir seulement les événements au départ ou à l'arrivée.

On retrouve dans cette requête des paramètres optionnels dont le rôle et l'usage est identique à la requête StopTimetableRequest étudiée précédemment.

La requête suivante permet d'obtenir une liste de **services** selon les critères suivants :

- Le code d'un opérateur
- Un mnémonique, un début de numéro approchant le numéro exact du service
- Une direction
- Inclure ou exclure un ou plusieurs modes de transport

La figure suivante illustre le diagramme de cette requête.

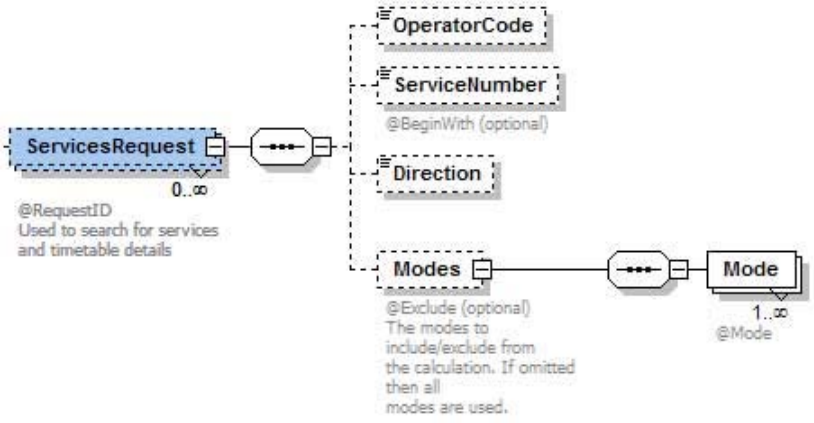


Figure 14 - Diagramme de ServiceRequest

La requête suivante permet de rechercher les **opérateurs** qui assurent un mode de transport.

On peut préciser comme dans la requête précédente si on exclut ou si l'on inclut certains modes de transport.

La figure suivante illustre le diagramme de cette requête.

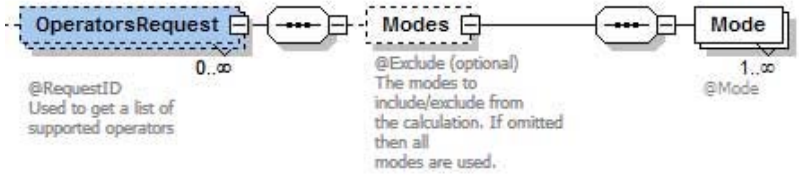


Figure 15 - Diagramme de OperatorsRequest

3.6 LES REPONSES DES SERVICES DE CALCUL D'ITINERAIRE

Comme nous l'avons indiqué dans le paragraphe précédent, la réponse est aussi un message du protocole JourneyWeb.

On trouve plusieurs types de réponse possible en fonction du ou des types de requêtes que l'on a formulé dans le message du protocole.

La figure suivante illustre le diagramme de cette réponse générale.

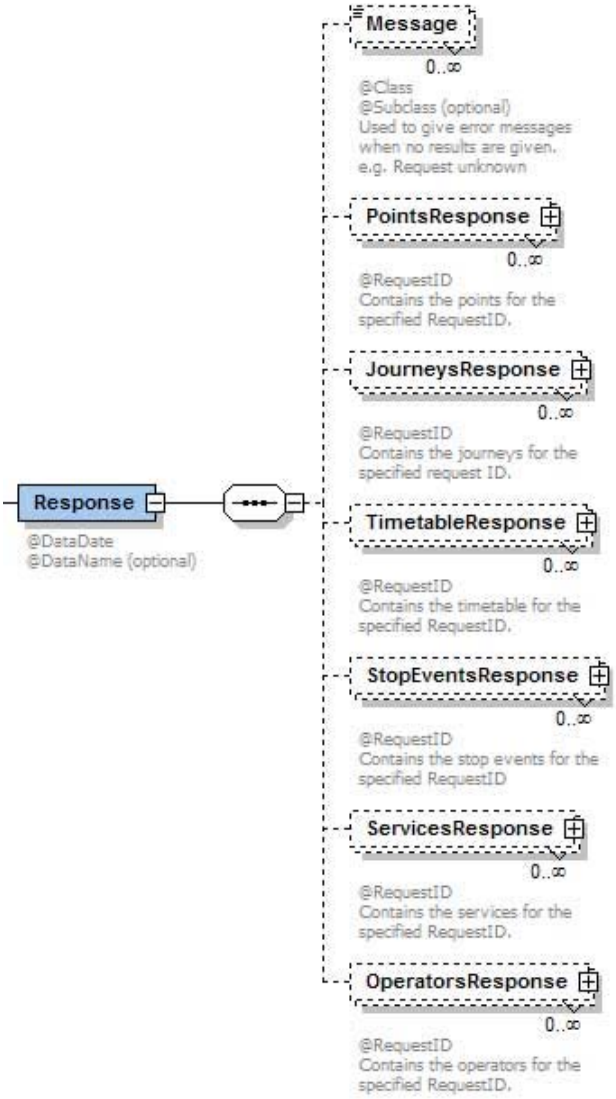


Figure 16 - Diagramme de Response

On peut remarquer que chaque type de réponse correspond à un type de requête étudié précédemment. Excepté pour *TimetableResponse*, qui est une réponse relative à deux requêtes, *StopTimetableRequest* et *ServiceTimetableRequest*.

Chaque réponse est donc caractérisée par un identifiant unique de requête. On remarque aussi que dans un seul message de type *Response* on peut avoir plusieurs résultats relatifs à différentes requêtes. Une réponse peut contenir un champ Message, qui indiquera un message d'erreurs.

La première réponse concerne la liste résultat des points obtenus avec la requête *PointsRequest*.

La figure suivante illustre le diagramme de cette réponse.

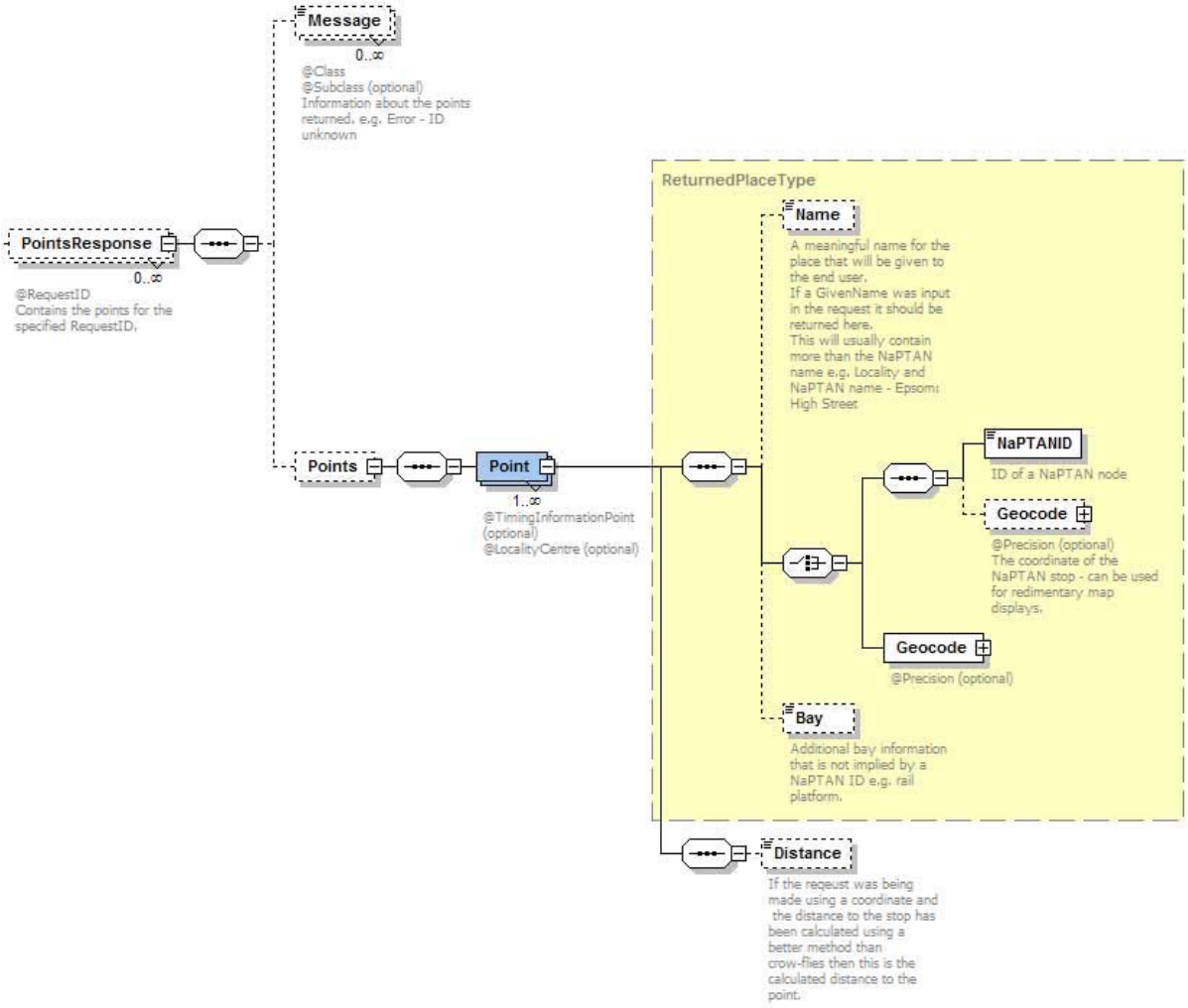


Figure 17 - Diagramme de PointsResponse

Tout point se trouvant dans la localité précisée dans la requête ou à proximité du point géodésique sera retourné dans la liste résultat. Le type de réponse relatif à l'emplacement du point sera utilisé dans les autres types de message.

Chaque point est défini par un identifiant du référentiel NaPTAN, et les coordonnées du point. Sont optionnels les champs comme le nom de l'emplacement, la distance entre le point et le point d'arrêt.

Une requête de calcul d'itinéraire peut donner plusieurs réponses possibles comme nous l'avons indiqué dans les chapitres précédents. Toutes ces réponses sont décrites dans le message, *JourneysResponse*.

La figure suivante illustre le diagramme de cette réponse.

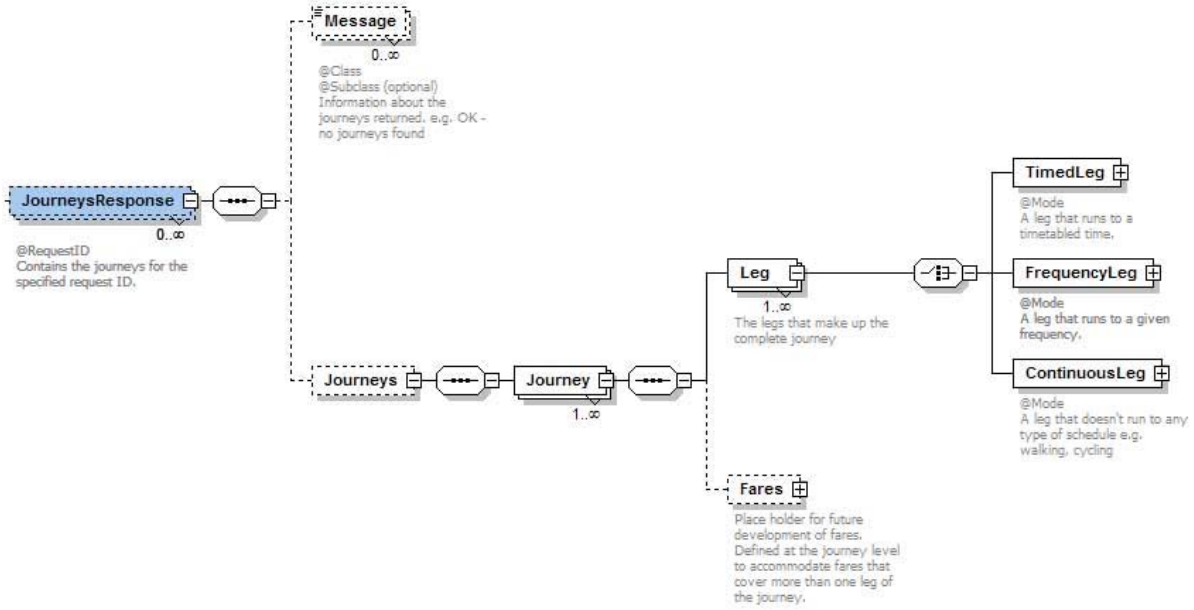


Figure 18 - Diagramme de JourneyResponse

Le résultat contenu dans ce message est un tableau décrivant des itinéraires. Chaque itinéraire contient un tableau de segments, on peut les comparer à des tronçons. On le nomme dans JourneyWeb, un **Leg**.

Chaque Leg se définit selon trois types :

- **Timed** Leg : si lds véhicules empruntent cet itinéraire, à une heure précise,
- **Frequency** Leg : si des véhicules empruntent plusieurs fois par jour cet itinéraire,
- **Continuous** Leg : si des véhicules empruntent cet itinéraire en boucle.

Ainsi une réponse d'un itinéraire pourra être constitué successivement de segments ou de tronçons d'itinéraires ayant chacun un des trois types définis plus haut.

On ne va décrire plus en détail les autres diagrammes mais on doit noter que chaque segment peut décrire un mode de transport sur un tronçon de réseau particulier et pour un horaire ou un service précis. Chaque réponse d'itinéraire contient les extrémités de chaque segment afin de bien identifier lorsqu'on doit effectuer un changement de correspondance, soit entre deux modes de transport, soit entre deux lignes. L'itinéraire doit donc inclure tous les temps de changement entre ces correspondances ainsi que les temps d'attente en ces points.

Tous ces paramètres sont illustrés dans les figures suivantes.

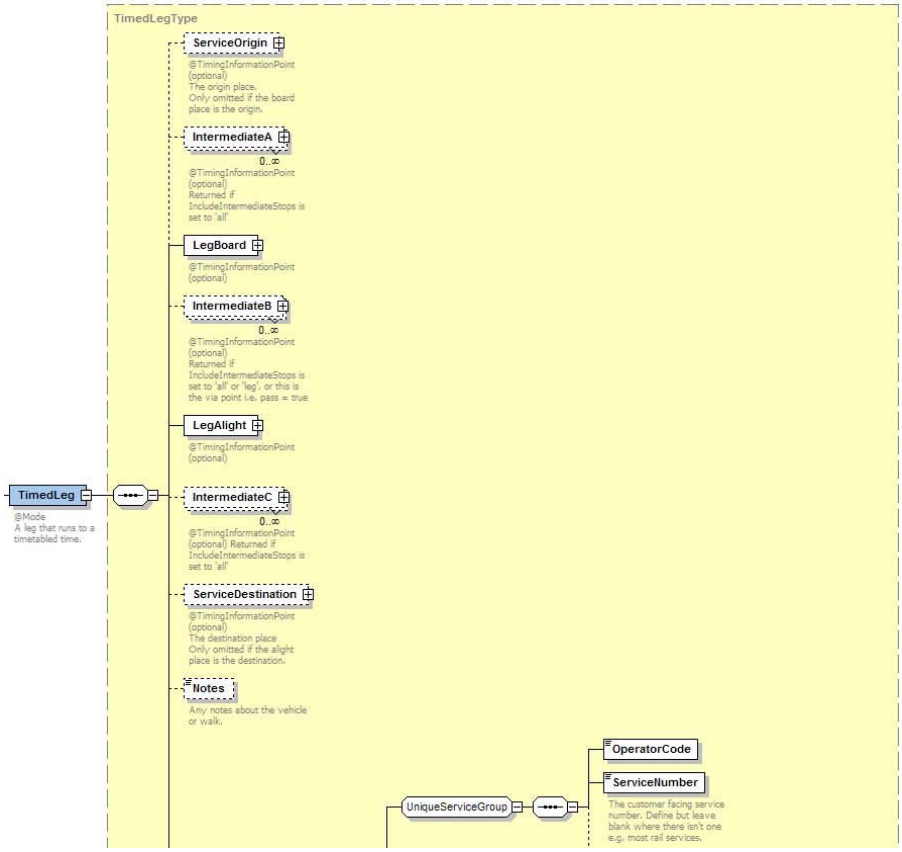


Figure 19 - Diagramme de TimedLeg

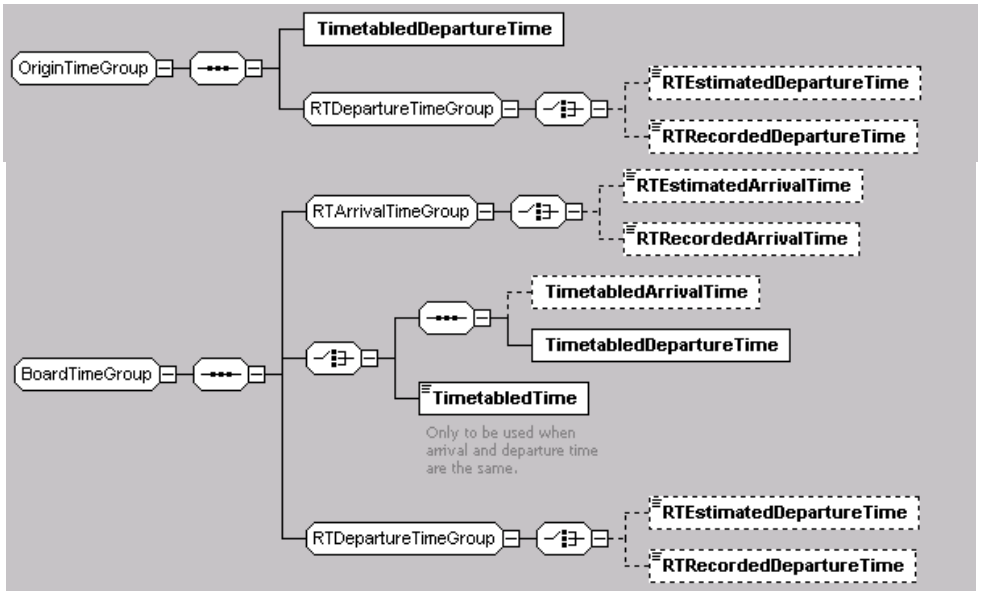


Figure 20 - Diagrammes de OriginTimeGroup et BoardTimeGroup

La figure suivante illustre le diagramme de la réponse à une demande d'horaires.

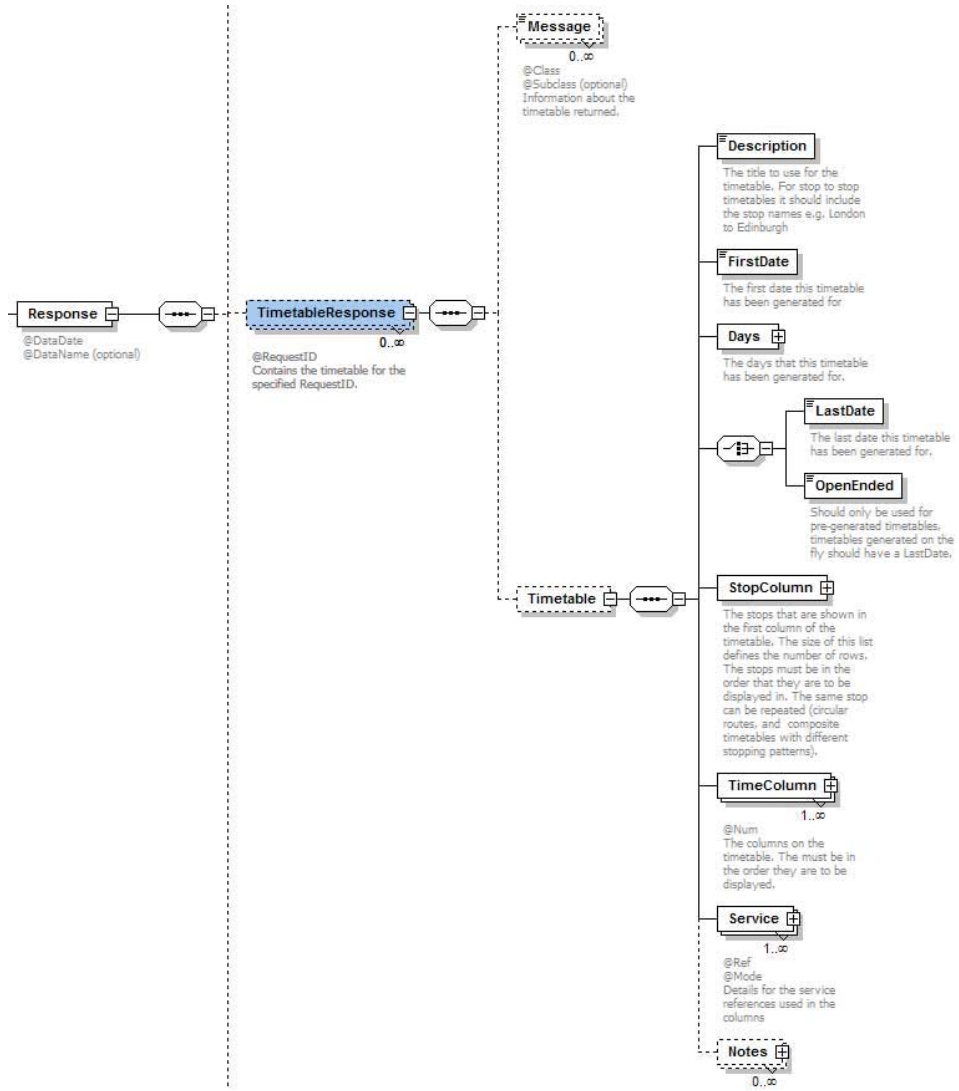


Figure 21 - Diagramme de TimetableResponse

Un message de réponse, TimetableResponse représente une matrice d'horaires. Chaque ligne correspond à un arrêt, et chaque colonne représente les horaires de passage. Ainsi dans une intersection ligne, colonne de cette matrice, on a l'horaire de passage à l'arrêt.

Si la matrice d'horaires est le résultat d'une requête de type ServiceTimeTable, la réponse doit contenir les détails du service correspondant. Si elle répond à une requête de type StopTimetable, la réponse doit alors contenir la description des points qui été utilisés pour composer cette requête.

Pour chaque cellule de cette matrice on va trouver deux types d'horaires, l'heure d'arrivée et l'heure de départ, ainsi qu'un attribut décrivant si l'arrêt est l'origine ou la destination d'un trajet. On trouvera aussi des étiquettes de colonnes permettant d'identifier les services rendus par ces colonnes. Toutes ces particularités sont mises en place afin de faciliter la diffusion de ces informations résultats, par exemple dans les applications Web en l'occurrence.

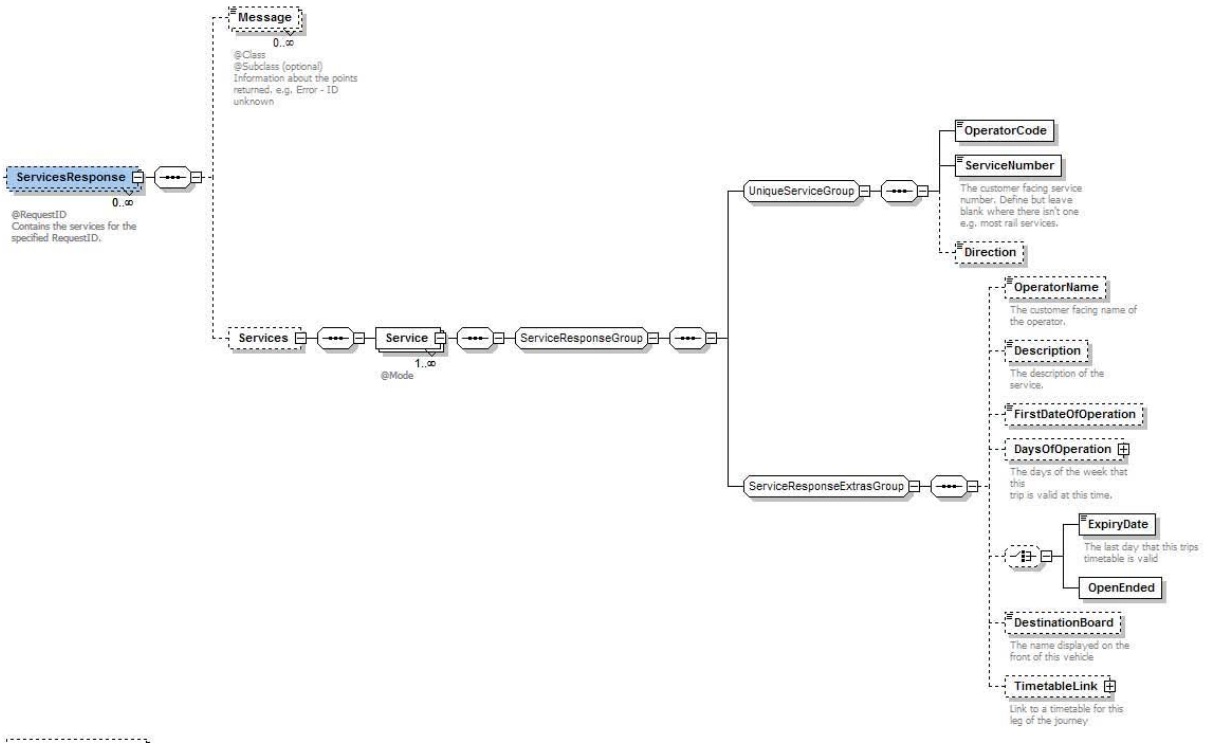


Figure 22 - Diagramme de ServiceResponse

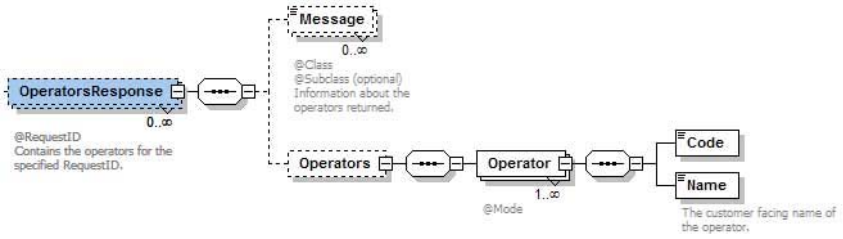


Figure 23 - Diagramme de OperatorsResponse

3.7 DIAGNOSTIC

Ce protocole est un excellent support à notre proposition de standardisation. Le cas anglais répond à nos attentes et aux besoins exprimés en matière de calcul d'itinéraire global à l'échelle d'un pays comme la France. Certes nous ne disposons pas actuellement d'une organisation similaire. Nous n'avons pas de référentiel national des lieux (équivalent NPTG) et des points d'arrêts (NaPTAN). Dans le cadre de l'étude nous serons amenés à envisager cette lacune et à la combler, car strictement nécessaire à l'expérimentation.

On retrouve dans la spécification de JourneyWeb que l'on vient d'étudier, des similitudes avec la pré-norme Trident, notamment sur les types de données et objets manipulés. Il conviendra donc de ne pas s'écarter de cette pré-norme et chaque fois qu'elle proposera dans son modèle une solution, c'est celle-ci qui sera utilisée. Nous allons dans le chapitre suivant examiner les solutions proposées par cette prénorme.

4 UTILISATION DE TRIDENT POUR LA SPECIFICATION

4.1 LE CALCUL D'ITINERAIRE

Nous allons étudier dans ce chapitre ce que nous apporte la prénorme Trident en matière de calcul d'itinéraire. Un calcul d'itinéraire est décrit dans les spécifications de Trident comme une fonction particulière de recherche d'information au même titre que la recherche d'un objet, d'un réseau, d'une situation de trafic ou encore la recherche d'un point d'arrêt à proximité d'un lieu donné. Le calcul d'itinéraire est ainsi classé et spécifié dans une famille de fonctions de recherche intitulée, *Request and Response*.

4.2 L'OUTIL CHOUETTE

La prénorme Trident ne sera pas décrite ici. De nombreuses publications ou présentations illustrent et décrivent à la fois l'approche et les bienfaits de cette prénorme. Aujourd'hui le CERTU a mis dans le domaine de l'Open Source un outil, CHOUETTE qui implémente une partie très réduite de cette norme.

CHOUETTE couvre la description des réseaux de transport en commun en termes de lignes, missions, courses et horaires. Le concept d'itinéraire correspond à un itinéraire particulier d'une ligne de réseau et permettant de relier deux points. CHOUETTE est capable de renseigner un client sur les horaires de transport en commun, et les services assurés sur un segment de réseau particulier.

Mais CHOUETTE ne dit pas quel chemin peut-on prendre pour se rendre d'un point à un autre, et n'effectue pas de calcul de correspondance et encore moins une quelconque optimisation de recherche d'itinéraires selon des critères classiques comme moins de temps de marche, le plus rapide, le plus court ou encore avec un minimum de perturbations. Ces fonctions existent dans quasiment tout logiciel de calcul d'itinéraire.

Cette étude doit proposer d'étendre le périmètre fonctionnel et technique de l'outil CHOUETTE, aussi bien relativement à son modèle de données qu'à son architecture et ses composants logiciels. Cet objectif doit lui permettre d'une part de pouvoir initialiser tout logiciel de calcul d'itinéraire à l'aide de ses données réseaux, lignes et horaires théoriques. D'autre part de pouvoir compléter un service de recherche d'itinéraire dans une finalité de calcul d'itinéraire multimodal et global.

4.3 REUTILISER TRIDENT COMME BASE DE SPECIFICATIONS

Avant de développer de nouveaux services de calcul d'itinéraire et d'étendre le modèle de données de CHOUETTE, nous allons étudier ce que nous apporte la prénorme Trident en termes de modèle de données et de protocole d'échange dans ce contexte de calcul d'itinéraire.

Nous allons autant que possible se référer à la première partie de notre étude pour analyser une carte de complétude entre JourneyWeb et Trident.

Trident définit un vaste domaine d'information. A la lecture des spécifications on constate assez rapidement qu'Il couvre quasiment tous les aspects des transports que l'on souhaite étudier, c'est à dire la recherche d'itinéraires.

Nous allons donc étudier dans ce chapitre comment les spécifications de Trident peuvent répondre à l'implémentation de notre protocole. Nous illustrerons chacun des messages à l'aide de diagrammes UML.

4.4 COMPARAISON DE JOURNEYWEB ET DE TRIDENT

JourneyWeb	Trident
Localisation : JourneyWeb utilise les paramètres *Easting* et *Northing* pour localiser un point ou lieu plus généralement.	Trident est plus riche dans la description de la localisation d'un point dans l'espace, et propose plusieurs systèmes de coordonnées dont le système X Y similaire à celui de JourneyWeb.
Itinéraire : JourneyWeb permet aux clients de fournir des informations plus précises pour effectuer des requêtes de calcul d'itinéraire. Les informations supportées dans une demande de calcul d'itinéraire sont le point de départ, le point d'arrivée, L'instance de départ et/ou l'instance d'arrivée (notons que les 2 peuvent co-exister !) les points intermédiaires, une période de temps pour limiter le nombre d'itinéraires retournés, l'algorithme souhaité pour calculer l'itinéraire, la vitesse de marche à pied, la distance maximale de marche à pied, la vitesse de changement de correspondances, si des informations temps réels seront retournées ou pas, Les modes de transport voulus (tram, bus …) Les opérateurs voulus Les services voulus (quelle ligne …). La réponse d'une demande de calcul d'itinéraire est un ensemble d'itinéraires. Un itinéraire est un ensemble ordonné de segments (*leg*) dans la spécification JourneyWeb). Un segment est une partie de l'itinéraire dont le type de transport n'est pas changé. Chaque segment contient les informations comme : L'origine et la fin du service avec les temps correspondants (temps de départ prévu, temps enregistré …)	Les informations supportées par Trident pour ce service sont moins détaillées on retrouve cependant : le point de départ et le point d'arrivée l'instance de départ ou l'instance d'arrivée (une seule peut être précisée), les points intermédiaires, les modes de transport souhaités et l'algorithme préconisé pour calculer l'itinéraire. L'itinéraire est un type défini dans Trident. Dans la version 2.0 des spécifications, un itinéraire (*Itinerary*) contient seulement un temps de départ. L'itinéraire est composé de plusieurs segments (*TripSegment*). Un segment peut être un lien qui n'appartient pas à un réseau de transport (marche à pied, véhicule personnel…) ou un lien qui appartient à un réseau de transport (on parle alors de *Ride*). Notons qu'il n'y a pas d'informations explicites sur le temps de départ ni sur le temps d'arrivée pour chaque segment (comme c'est le cas dans JourneyWeb). A la place de cela, chaque segment Trident définit un intervalle du temps pour le parcourir. On peut donc, à partir du temps de départ d'un itinéraire et des intervalles de temps des segments qui le composent, en déduire le temps de départ et d'arrivée pour chaque segment. Ces informations sont donc, implicites.

Après l'analyse d'une telle comparaison, on peut affirmer que Trident apporte l'essentiel de notre besoin de spécifications en matière de recherche d'itinéraires tant au niveau des messages devant être échangés qu'au niveau du protocole de communication. Ce dernier point fera l'objet d'un chapitre particulier dans lequel nous décrirons et argumenterons le choix technique que nous avons fait.

Nous allons illustrer dans les paragraphes suivants l'essentiel des spécifications élaborées à l'aide de Trident.

4.5 TYPES DE MESSAGES

La table suivante énumère la liste des types de messages que nous avons spécifié pour le protocole de communication qui sera décrit dans le chapitre suivant.

Type de message	Signification
RequestResponseType	Le type racine de toutes les requêtes et toutes les réponses
ObjectRequestType	Demander un objet en indiquant son identifiant **OID**.
ObjectResponseType	L'ensemble des objets qui possèdent cet OID défini dans la requête.
PtStopPointsRequest	Demander une liste des StopPoints satisfaisant un critère
PtStopPointsResponse	Contient une liste des StopPoints
ItineraryCalculationRequest	Demander des itinéraires en indiquant le point de départ, d'arrivée et les critères de recherche optionnels.
ItineraryCalculationResponse	Les itinéraires trouvés
TimeTableRequestType	Demander l'horaire de passage sur un point d'arrêt ou d'une ligne.
TimeTableResponseType	La liste des horaires demandé.

4.6 DESCRIPTION DES MESSAGES

RequestResponseType

RequestResponseType possède un attribut, l'identification du message (requête ou réponse). Cela permet d'identifier et de synchroniser toutes les requêtes et les réponses. Tous les messages ci-dessus héritent du type *RequestResponseType*.

ObjectRequestType

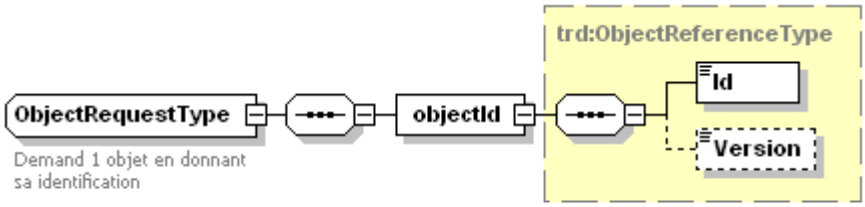


Figure 24 - Diagramme ObjectRequest

Dans Trident, un objet est entièrement défini par un couple de paramètres objectId, **OID** (une chaîne de caractères String, de la forme **{PeerId}:{ClassId}:{progressiveInt}** et un numéro de version (un entier Int).

ObjectResponseType

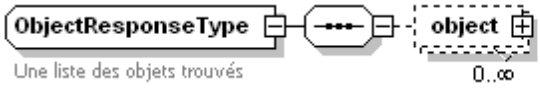


Figure 25 - Le type ObjectResponse

- object (trd:TridenObjectType) : Contient une liste des objets de type TridentObjectType, qui est la racine de tous objets Trident. Concrètement, il peut être n'importe quel objet (un point d'arrêt, une Ligne, un réseau de transport ...).

PtStopPointsRequestType

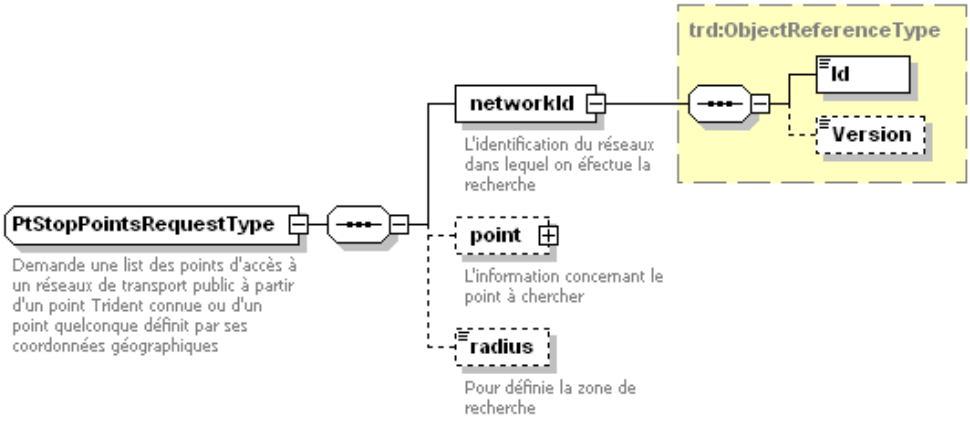


Figure 26 - Le type PtStopPointsRequest

- networkId (trd:ObjectReferenceType) : l'identification du réseau.
- point (trd:PointType) : le point à partir duquel on effectue la recherche
- radius (xsd:decimal) : le rayon définit un cercle au tour de ce point.

PtStopPointsResponseType

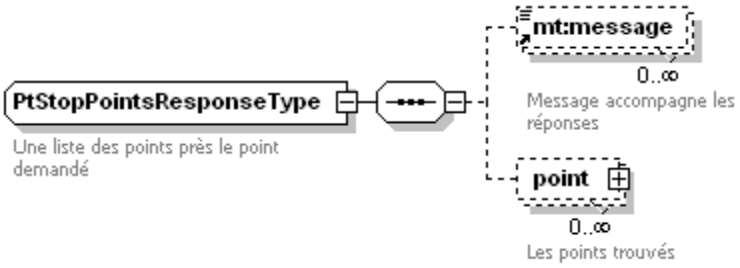


Figure 27 - Le type PtStopPointsResponse

- point (trd:StopPointType) : Liste des points d'arrêt trouvés

ItineraryCalculationRequestType

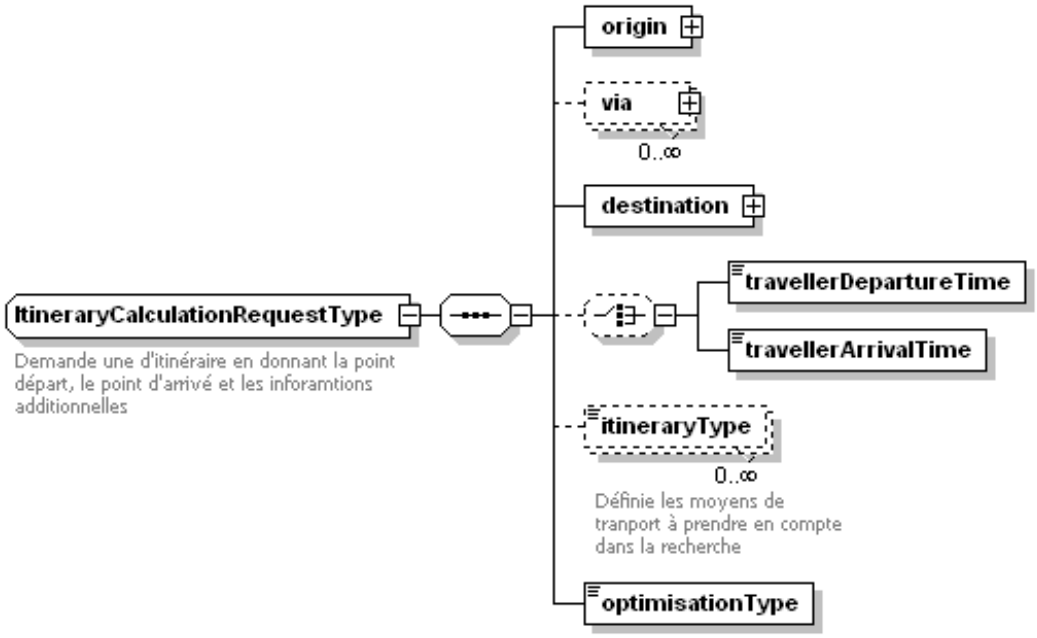


Figure 28 - Le type ItineraryCalculationRequest

- origine & destination (trd:PointType) : Points qui définissent les emplacements de départ et d'arrivée.
- via (trd:PointType) : Les points qu'on veut inclure dans la recherche d'itinéraire (optionnel).
- travellerDepartureTime & travellerArrivalTime (xsd:dateTime) : date du départ **ou** d'arrivée.
- itineraryType (trd:TransportModeNameType) : Préciser les modes de transport utilisés (à pied, voiture personnelle, métro, tram ...)
- optimisationType (trd:OptimisationType) : Critère d'optimisation. Les valeurs permises sont :
- MinTravelDuration : Minimiser le temps du voyage.
- MinTravelDistance : Minimiser la distance du voyage.
- MinWalkingDuration : Minimiser le temps de marche à pied.
- MinNumOfTransfers : Minimiser le nombre de correspondances sur l'itinéraire.
- MinTravelCost : Minimiser le coût de transport.

ItineraryCalculationResponseType

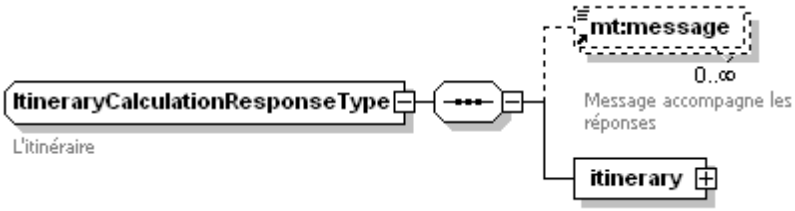


Figure 29 - Le type ItineraryCalculationResponse

- itinerary (trd:ItineraryType) : l'ensemble des itinéraires possibles.

Le type « itinéraire » est décrit selon le schéma suivant.

ItineraryType

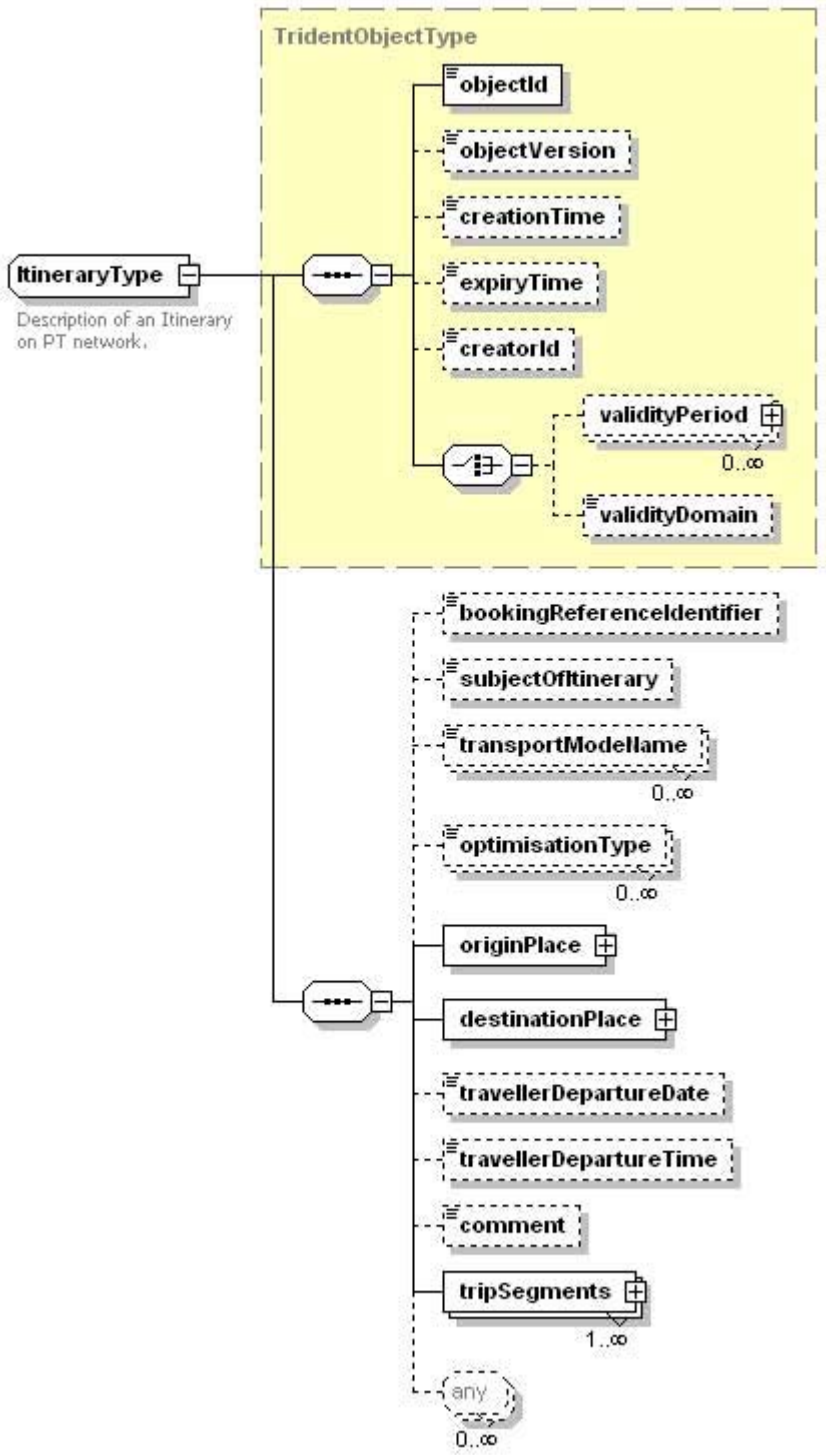


Figure 30 – Le type Itinerary

TripSegmentType

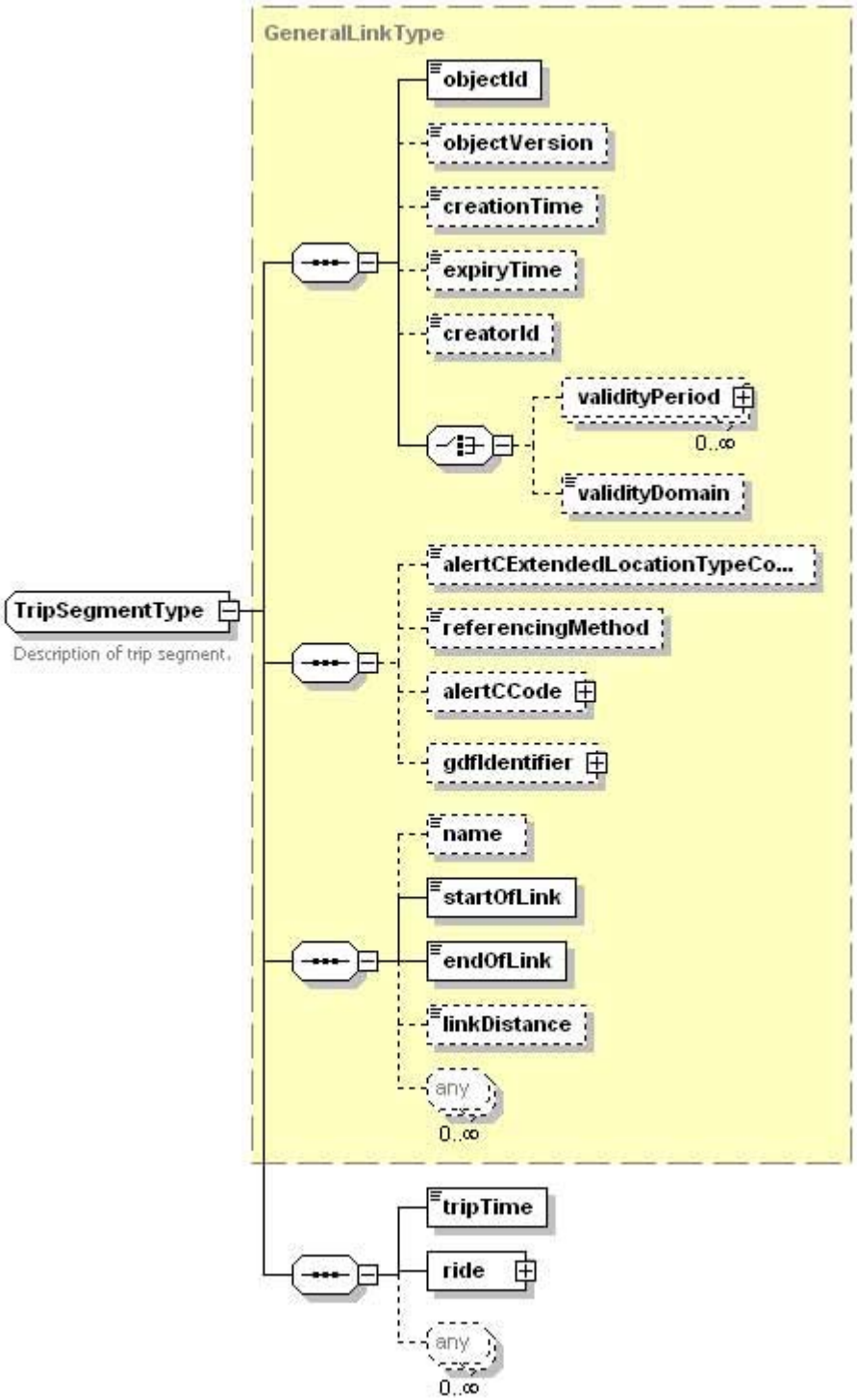


Figure 31 - Le type TripSegment

TimeTableRequestType

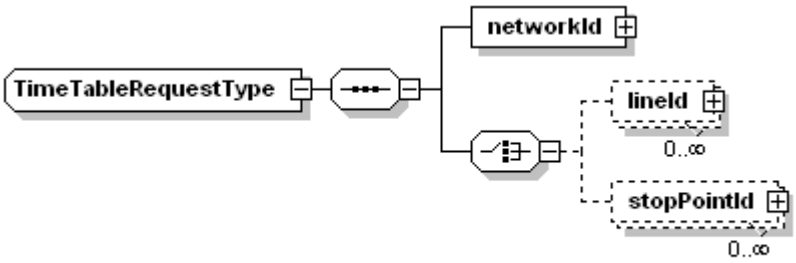


Figure 32 – Le type TimeTableRequest

- networkId (trd:ObjectReferenceType) : identifie le réseau.
- lineId (trd:ObjectReferneceType) : identifie une ligne, dans ce cas, la réponse contiendra tous les horaires de cette ligne.
- stopPointId (trd:ObjectReferneceType) : définit un point d'arrêt, dans ce cas, la réponse contiendra l'horaire sur ce point.

TimeTableResponseType

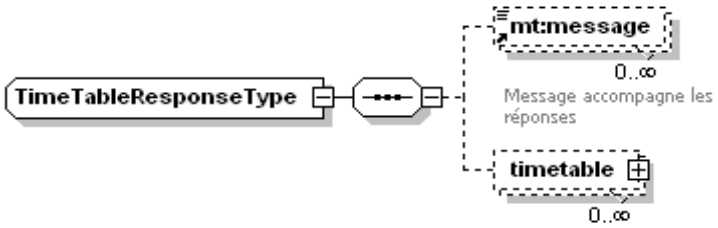


Figure 33 - Le type TimeTableResponse

- timetable (trd:TimeTableType) : contient les horaires d'un réseau, d'une ligne, ou d'un point d'arrêt.

TimeTableType

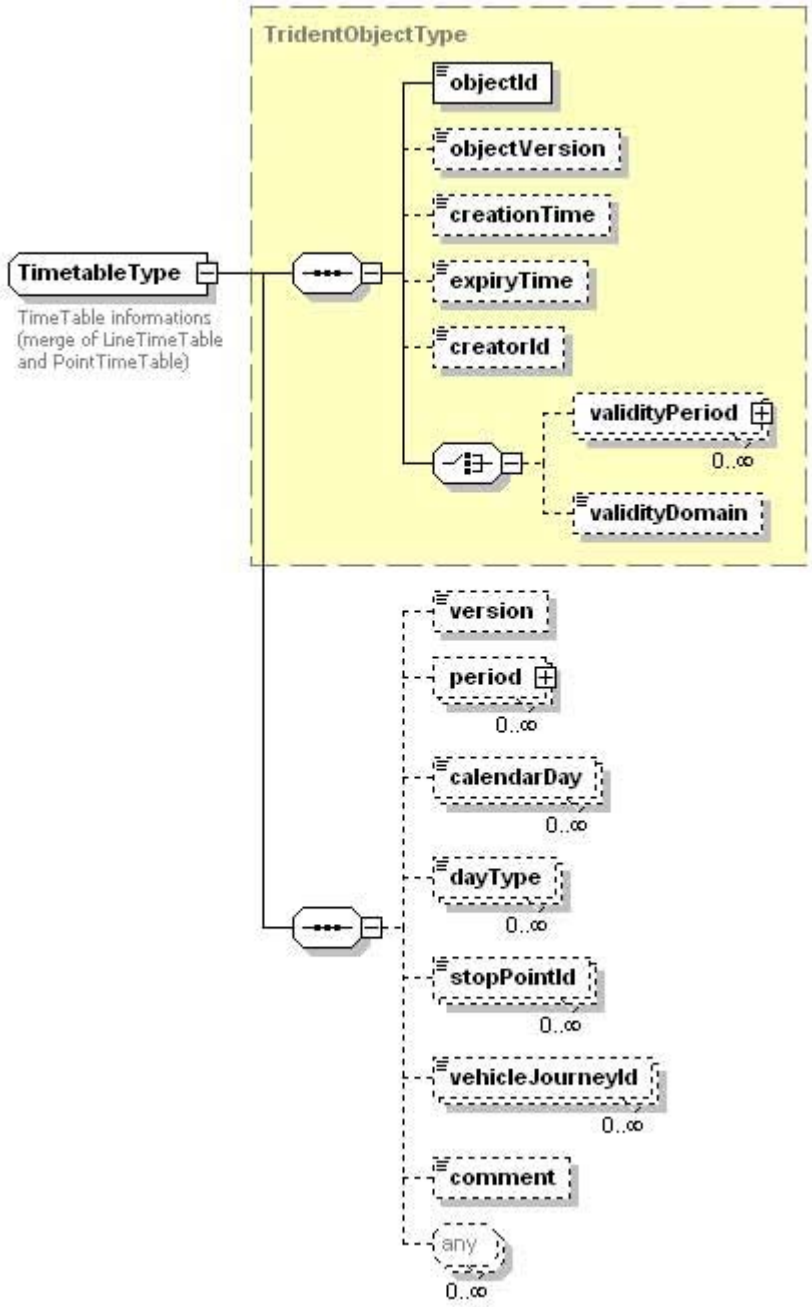


Figure 34 - Le type Timetable

4.7 MISE EN ŒUVRE D'UN CALCUL D'ITINERAIRE

On va détailler dans ce paragraphe un scénario type d'utilisation de notre protocole pour la recherche d'un itinéraire.

Enoncé : un usager se connecte à une page Web pour rechercher un itinéraire lui permettant de se rendre à la station de métro « Hénon ». Il désire partir de la station de Tramway "INSA – Albert Einstein".

ItineraryCalculation

1. Utilisateur fournit un nom d'arrêt pour le départ de sa recherche d'itinéraire	Si le nom entré est suffisamment précis pour localiser un endroit ou un point stocké dans la base des données, le serveur Web effectue un *PtPointsRequest* pour récupérer les point d'arrêt situés à proximité. Si non, il consulte un moteur lexicographie pour proposer les noms possibles à l'utilisateur.
2. Utilisateur choisit un point d'arrêt pour le départ	
3. Répéter les opérations 1, 2 pour le point d'arrivée, les points d'intermédiaires.	
4. Utilisateur définit les critères de recherche optionnels (le type d'optimisation, le temps de départ ou d'arrivée...)	En ayant ces informations fournies par l'utilisateur, le serveur Web construit un *ItineraryCalculationRequest* et l'envoie au noeud d'échange qui s'adresse au service de calcul d'itinéraire.
	La réponse *ItineraryCalculationResponse* contient une liste des segments qui définissent l'itinéraire. Chaque segment contient un OID qui définit le départ du segment et un OID qui définit la fin du segment. Notons que le départ et la fin d'un segment peut être un *Location*, un *BusStopPoint*, un *VehicleJourneyAtStop* ... Le serveur Web effectue des *ObjectRequest* pour récupérer les objets représentés par ces OIDS, met en forme la page Web et la renvoie au client.
5. Utilisateur visualise la page résultante.	

4.8 MISE EN ŒUVRE D'UNE RECHERCHE D'HORAIRES

Cette fonction sera décrite dans le cadre lot 2 et du démonstrateur.

5 CHOIX DES SERVICES WEB POUR LE PROTOCOLE DE COMMUNICATION

5.1 MODELE DE COMMUNICATION

On va s'intéresser dans ce chapitre aux problématiques de communication de données en termes de protocole et de type de données échangées entre les composants logiciels intervenant dans notre calcul d'itinéraire global. Nous détaillerons l'argumentaire en faveur du choix des Services Web comme socle logiciel de notre spécification et de notre implémentation.

La prénorme TRIDENT fournit à cet effet des lignes directrices et un cadre de développement des couches de transport de l'information et des interfaces d'accès aux protocoles de communication. Trident recommande d'implémenter au minimum un modèle de communication de type PEER-TO-PEER entre les applications jouant le rôle de fournisseur ou de client. Trident recommande par ailleurs de s'orienter vers un modèle de communication à objets distribués. Un tel modèle peut être par exemple un middleware orienté messages. Enfin, Trident recommande de supporter les flux d'échange de données XML et en particulier les protocoles qui permettent de véhiculer ce type de flux. En l'occurrence, le protocole SOAP est en parfaite adéquation avec ses exigences.

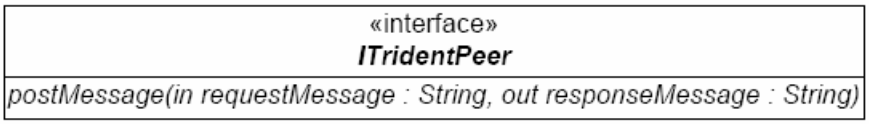
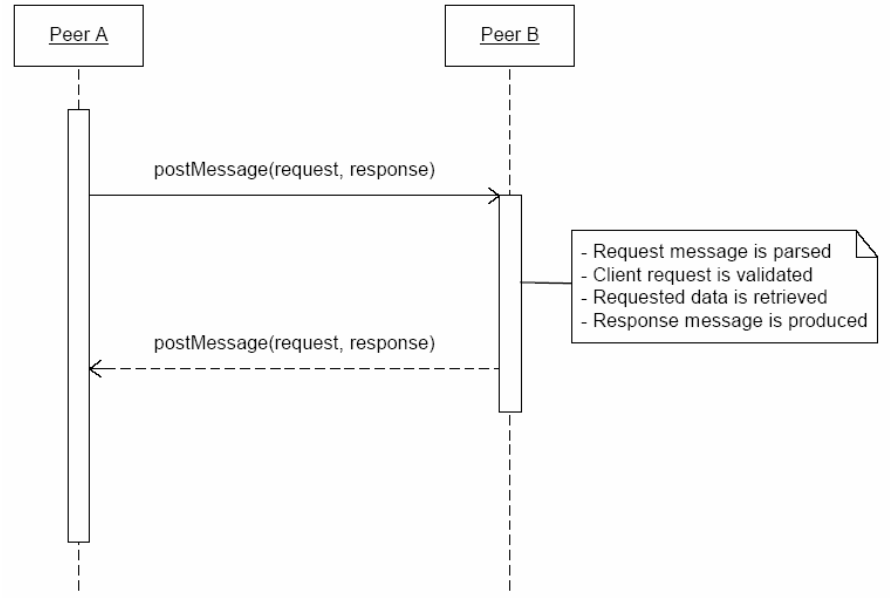


Figure 35 - Diagramme de classe de ITridentPeer

Ce modèle issu de la spécification de Trident représente l'interface de la couche de transport qui devra être implémentée par tout client « Peer A » désirant établir une communication avec le fournisseur « Peer B ».



Dans le diagramme ci-dessus est illustré une interaction entre ces deux « Peers ». A poste son message et attend la réponse du fournisseur B. Ce scénario est dans plus simple expression. Dans la réalité un modèle de programmation doit être implémenté afin de rendre indépendant à la fois le client et le fournisseur, mais aussi de permettre aux clients et aux fournisseurs de traiter plusieurs clients simultanément et aux client d'interroger plusieurs fournisseurs. Comme cela est indiqué dans Trident,

il n'y a pas à proprement parler, un client ou un serveur distant. Il y a des systèmes qui sont déclarés comme fournisseurs d'information. A ce titre, ils doivent publier des interfaces d'accès à leurs informations.

Dans la pratique cela consiste pour un logiciel à publier les principales méthodes d'accès à son protocole, par exemple à l'aide de Java RMI, des interface IDL CORBA ou DCOM ou encore à publier la description de son service à l'aide du langage XML. Ce dernier mécanisme est mis en œuvre à l'aide d'un protocole **SOAP** et du langage **WSDL** (Services Web) et à déclarer son service à un annuaire UDDI. (équivalent des *pages jaunes* pour les Services Web). La figure suivante illustre les principales phases d'une telle communication avec les Services Web.

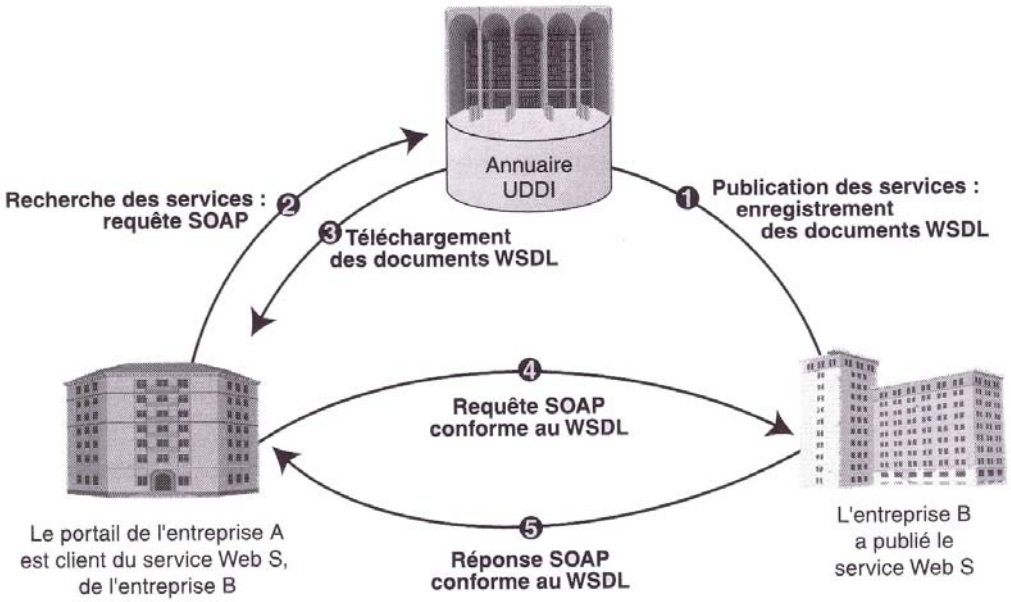


Figure 36 - Annuaire des Services Web

Le point crucial est ici l'étanchéité entre les clients et les serveurs qui ignorent tout de l'implémentation, dite privée, des opérations respectives de leurs correspondants. Ainsi une application Web qui désire utiliser les services d'un logiciel de calcul d'itinéraire n'aura jamais à connaître les détails du procédé, d'un algorithme ou d'une technologie protégée. Les Services Web sont donc une brique essentielle dans la mise en œuvre d'une proposition d'un standard en matière de calcul d'itinéraire global sans dévoiler les technologies de chacun. Chacun d'entre eux ne pourra se reconnaître qu'à travers une description publique, le Service Web et son descripteur, le WSDL.

Un autre point tout aussi important, concerne l'assemblage des composants entre eux pour former des services plus complets et plus globaux. En effet les Services Web peuvent s'assembler entre eux pour former de nouveaux Services Web. Enfin le mécanisme de communication permettant cette circulation de requêtes et de résultats autorise la mise en relation de plusieurs clients et de plusieurs serveurs. Un Service Web donné peut être suivant le contexte aussi bien un client qu'un serveur d'un autre Service Web.

Ce principe de communication répond parfaitement aux attentes de la prénorme TRIDENT relatives aux principes d'un réseau d'échange de données entre des systèmes d'information multimodale sans imposer à tel ou tel système de jouer un rôle particulier de client ou de serveur. Avec les Services Web, on peut donc parler de « Bus de Requêtes ». Les spécifications sont complètement décrites dans le descripteur WSDL du Service Web correspondant.

5.2 INTEROPERABILITE

Nous avons donc retenu cette dernière solution à la fois plus récente, plus facile à déployer et surtout transparente aux technologies que nous devons interfacer comme l'illustre la figure suivante.

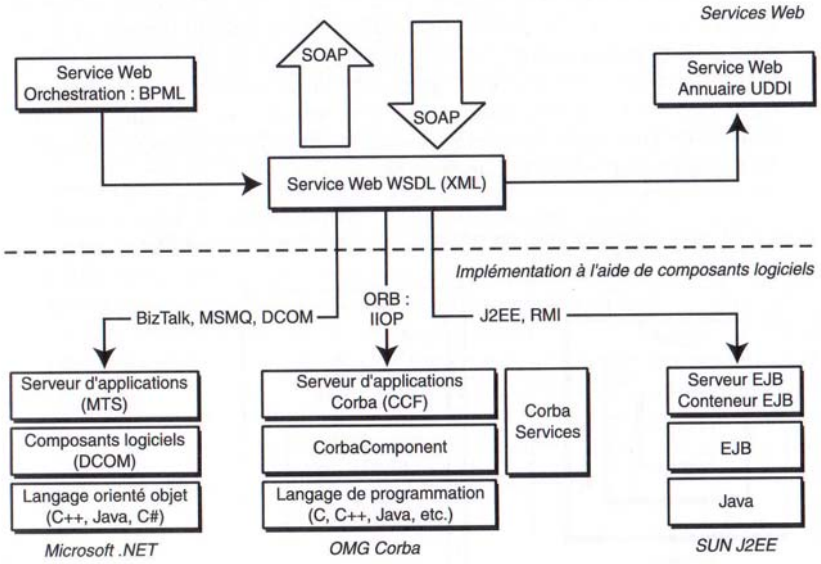


Figure 37 - Transparence de SOAP et des Services Web

SOAP se classe parmi les protocoles de communication orientés objet, reprenant les principes traditionnels de RPC (Remote Procedure Call) des protocoles antérieurs comme DCOM ou IIOP de CORBA. SOAP ne bouleverse donc pas l'architecture de communication éventuellement préexistante dans le réseau de l'entreprise. Bien au contraire, le rapprochement des principes de SOAP de ceux de RPC facilité l'intégration des différents protocoles et assure une excellente interopérabilité à moindre coût à la jonction de l'Intranet et de l'Internet.

5.3 DECRIRE UNE SPECIFICATION D'INTERFACE AVEC WSDL

Les Services Web sont représentés dans le langage WSDL et représentés comme des terminaisons d'un réseau maillé. indépendant de la couche de communication. WSDL permet de spécifier comment employer les protocoles de communication comme HTTP, SOAP ou MIME comme couche de communication de transport des messages des Services Web. (requêtes et réponses).

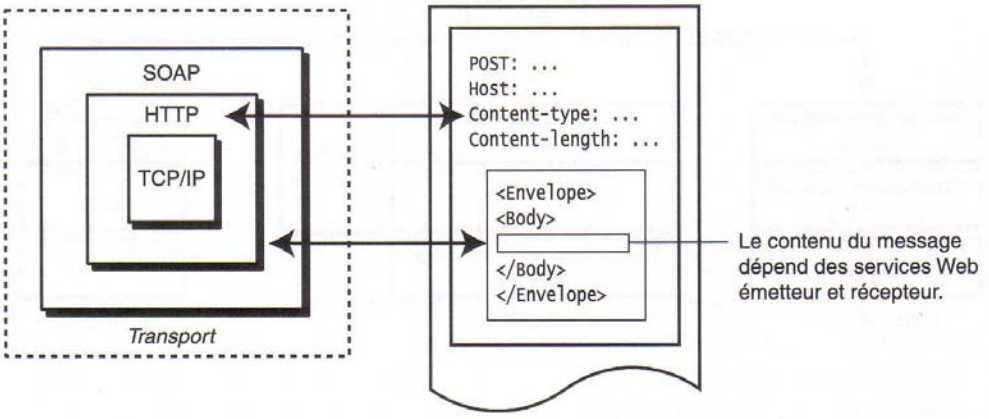


Figure 38 – Structure générale d'un message

Avec WSDL, les Services Web communiquent par échange de messages entre des terminaisons, constituées d'un ou de plusieurs ports, chacun doté d'un type – au sens des langages de programmation -. La spécification WSDL 1.0 définit les entités suivantes illustrées dans la figure suivante.

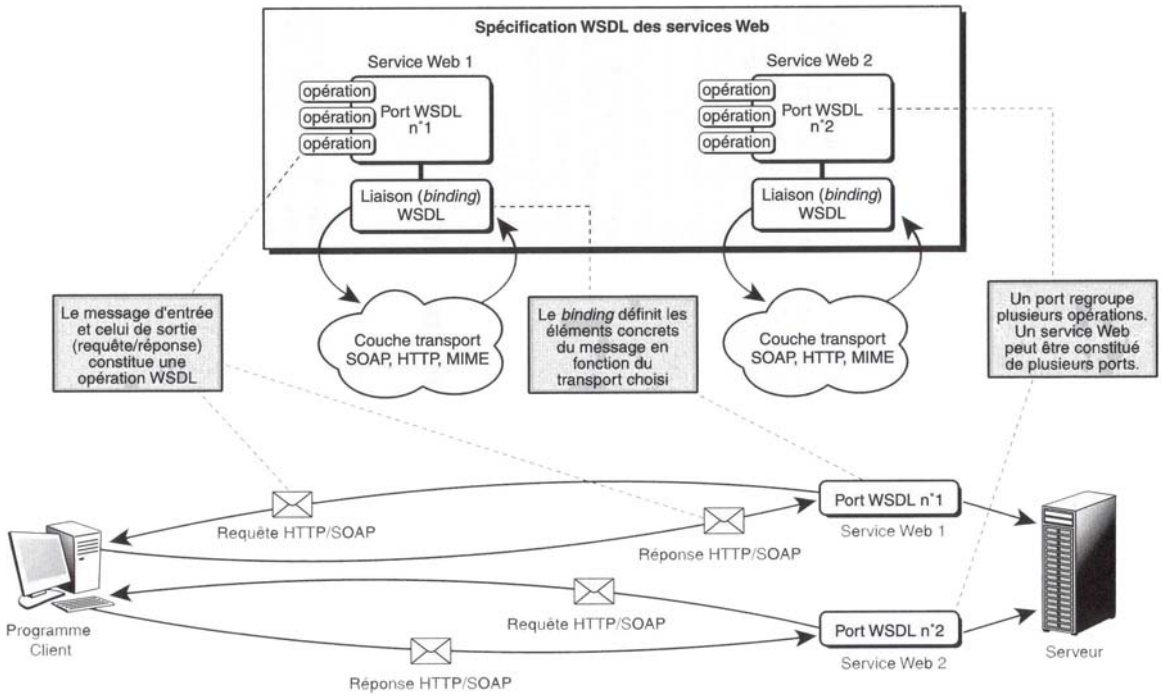


Figure 39 - Description d'un Service Web avec WSDL

Définissons ces entités :

- Message : abstraction décrivant les données échangées entre les Services Web.
- Opération : abstraction décrivant une action implémentée par un Service Web.
- Type de port : un ensemble d'opérations implémenté par une terminaison donnée.
- Liaison : un protocole concret d'accès à un port et un format de spécification des messages.
- Port : un point de terminaison identifié de manière unique par la combinaison d'uen adresse Internet et d'une liaison.
- Service Web : une collection de ports.

Dans cette vision des Services Web, ceux-ci sont des « boites noires », présentant l'apparence d'un certain nombre de ports d'entrée et de sorties, chacun doté d'un type lié à l'implémentation d'un ensemble d'opérations. Ces ports sont mis bout à bout, reliés par un protocole de transport comme HTTP, SOAP ou MIME.

```
<wsdl:portType name="MTServicePortType">
		<wsdl:operation name="ObjectRequest">
			<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return an Object in
function of its ObjectId</wsdl:documentation>
			<wsdl:input message="tns:ObjectRequestMessage"/>
			<wsdl:output message="tns:ObjectResponseMessage"/>
		</wsdl:operation>
		<wsdl:operation name="PtStopPointsRequest">
			<wsdl:input message="tns:PtStopPointsRequestMessage"/>
			<wsdl:output message="tns:PtStopPointsResponseMessage"/>
		</wsdl:operation>
		<wsdl:operation name="ItineraryCalculationRequest">
			<wsdl:input message="tns:ItineraryCalculationRequestMessage"/>
```

```
                        <wsdl:output message="tns:ItineraryCalculationResponseMessage"/>
                </wsdl:operation>
                <wsdl:operation name="TimeTableRequest">
                        <wsdl:input message="tns:TimeTableRequestMessage"/>
                        <wsdl:output message="tns:TimeTableResponseMessage"/>
                </wsdl:operation>
</wsdl:portType>
```

Figure 40 - Exemple de description d'un port d'un Service Web de calcul d'itinéraire

En WSDL, les messages sont spécifiés sans référence explicite au protocole de transport retenu. Cette indépendance permet de réutiliser cette description pour plusieurs systèmes. Ainsi un WSDL servira de spécification aussi bien pour une implémentation en langage Java que des langages plus légers comme PHP, PERL ou autre.

5.4 PROPOSITION DE MISE EN ŒUVRE

Après avoir argumenté notre choix technique des Services Web comme base technique et comme formalisme de notre spécification, nous allons présenté un principe de mise en œuvre d'un calcul d'itinéraire global reposant sur l'assemblage de Services Web.

Chaque fois qu'il sera nécessaire d'interfacer tel ou tel logiciel de calcul d'itinéraire, nous devons développer une nouvelle interface d'accès à ce logiciel. Cette interface doit exposer à l'ensemble des applications désirant utiliser ce logiciel, à la fois les méthodes réalisant par exemple, une recherche de points d'arrêts, ou une recherche d'un trajet le plus court possible entre deux points. La plupart du temps les informations nécessaires à la réalisation d'un calcul d'itinéraire global vont être localisées dans des bases de données particulières et souvent géographiquement éloignées. Là aussi il sera nécessaire de tenir compte d'une part de la situation géographique de ces données, et de masquer aux applications devant utiliser ces données, la complexité de ces dernières.

Notre proposition n'imposera pas un modèle stricte d'implémentation mais plutôt un cadre de travail pour une implémentation. Les Services Web, comme l'illustre la figure suivante, vont permettre de réaliser par exemple l'interfaçage entre l'outil Chouette et un logiciel de calcul d'itinéraire. Tout logiciel de calcul d'itinéraire aura besoin des données descriptives d'un réseau de transport pour pouvoir opérer un calcul d'itinéraire efficace. Le Service Web va permettre à ce logiciel de calcul d'accéder aux données de Chouette en utilisant seulement une description WSDL.

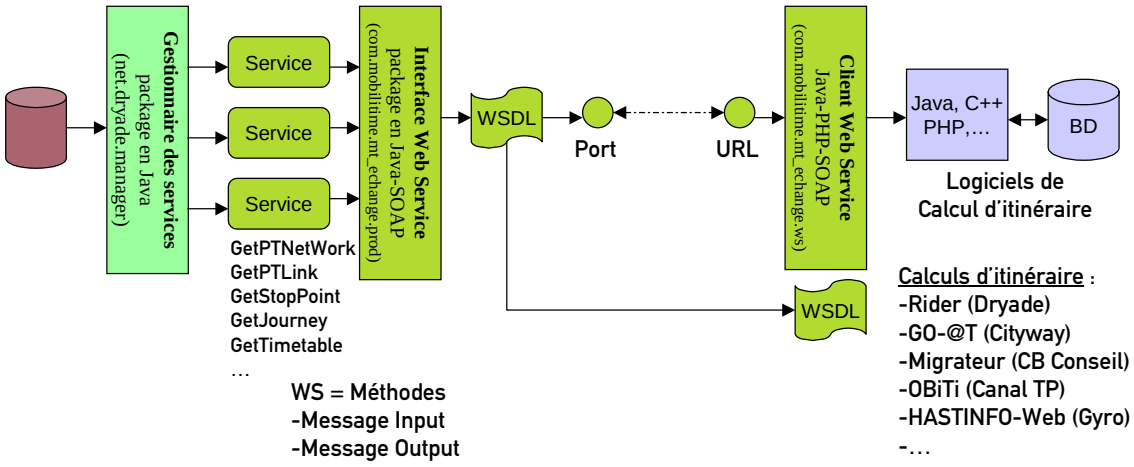
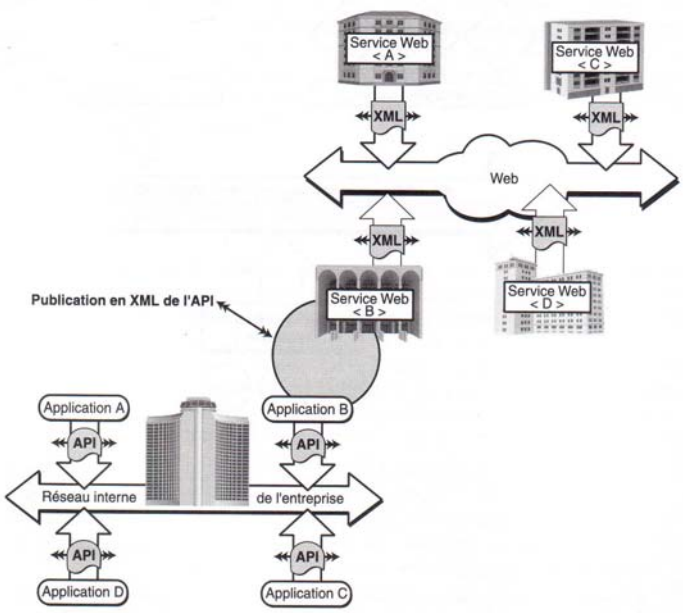


Figure 41 - Service Web comme interface logicielle

Quand on souhaite effectuer un calcul d'itinéraire global par exemple mettant en œuvre une coopération de plusieurs outils de calcul d'itinéraires appliqués chacun à une région particulière ou à

un réseau de transport particulier ou à un mode de transport spécifique, il va être nécessaire d'orchestrer l'enchaînement de ces processus, et autant que possible d'assurer une transparence vis à vis des données échangées et de l'ensemble des réseaux d'entreprises devant être traversés, Intranet ou Internet comme l'illustre la figure suivante.



Les Services Web vont nous fournir le moyen d'assembler ces processus en un assemblage de plusieurs Services Web. La figure ci-après illustre cette orchestration. Les Services Web WS1, WS2 et WS3 servent d'interfaces et de spécifications d'accès pour une application de calcul d'itinéraire global, vers les différentes applications ou logiciels de calcul réparties.

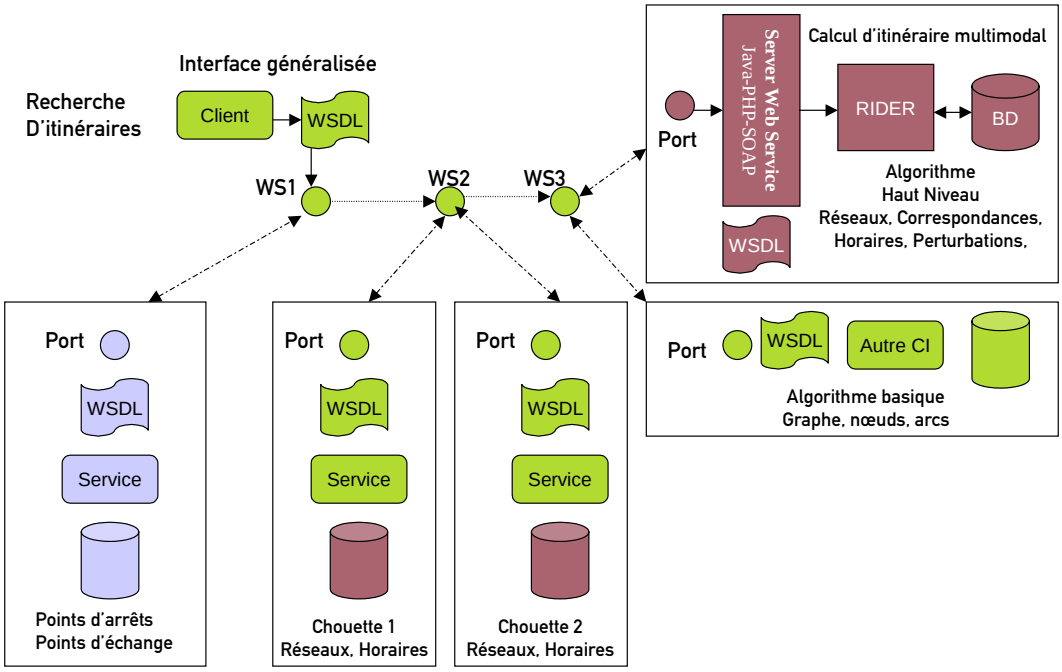


Figure 42 - Assemblage des Services Web pour un calcul d'itinéraire global

6 INTERFAÇAGE DE RIDER COMME EXEMPLE DE LOGICIEL DE CALCUL D'ITINERAIRE

6.1 INTRODUCTION

Dans cette dernière partie de notre étude nous allons examiner un exemple d'interface d'accès au service de calcul d'itinéraire RIDER.

6.2 INTERFACE D'ACCES VIA SOAP

Les différents services d'accès au logiciel de calcul d'itinéraire, RIDER, sont regroupés dans un Web Service et via l'interface SOAP. Ces services sont basés sur ceux du service « Embouteillages », déjà développé pour le site **citefutee.com** et pour la RATP.

6.3 GESTION DES DROITS D'ACCES

L'accès à l'interface SOAP est géré par un mécanisme de contrôle basé sur un couple :

- nom d'utilisateur
- un mot de passe

Ces paramètres sont toujours inclus dans chacune des requêtes qui seront transmises au service.

6.4 RECHERCHE D'ADRESSE

La première des fonctionnalités offertes par ce service est la recherche d'adresse en Ile de France. Un fichier au format *XML Schema* décrit l'ensemble de l'interface SOAP offerte aux services distants. Le contenu XML ne sera pas décrit dans cette étude mais on illustrera les diagrammes des principales méthodes exposées par ce Web Service.

Voici ci-dessous une illustration du diagramme de la méthode ***getStreetList***.

Question :

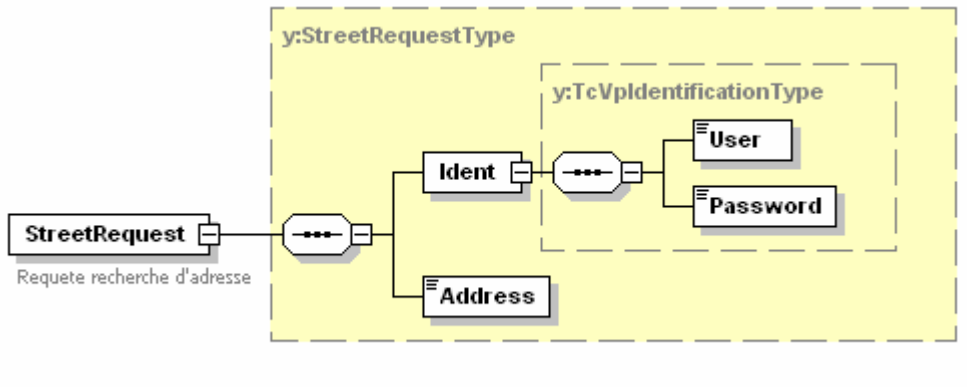


Figure 43 - Diagramme de la méthode StreetRequest

Réponse :

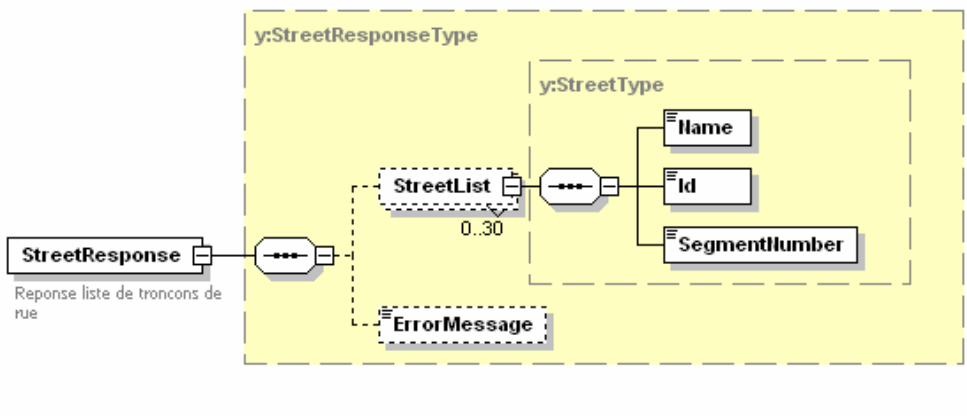


Figure 44 - Diagramme de la méthode StreetResponse

6.5 RECHERCHE DES ARRETS

La méthode **getTcVpPointList** permet de récupérer l'ensemble des pôles TC/VP connus du calculateur d'itinéraire.

Question :

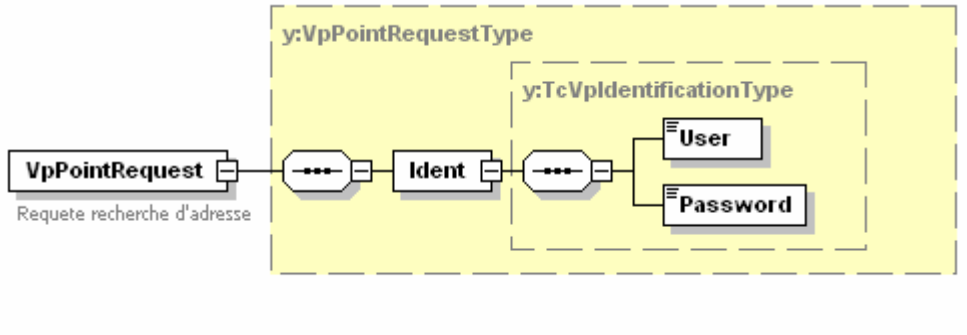


Figure 45 - Diagramme de la méthode VpPointRequest

Réponse :

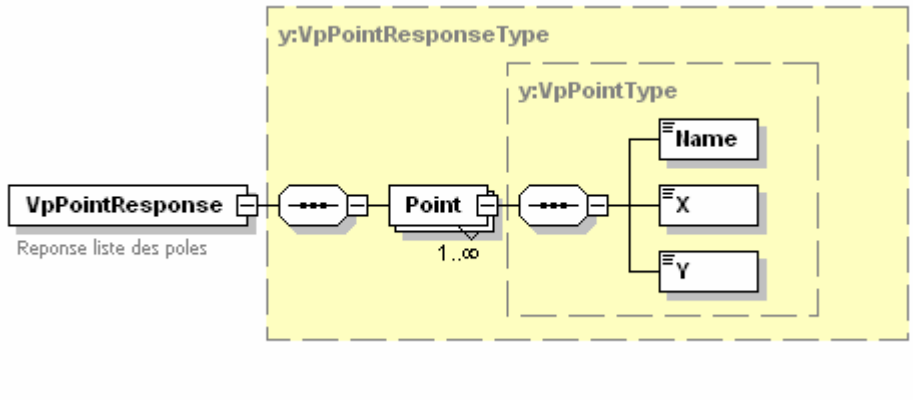


Figure 46 - Diagramme de la méthode VpPointResponse

6.6 CALCUL D'ITINERAIRE

La méthode **getTcVpRequest** permet d'interroger le calcul d'itinéraires en lui indiquant les points de départ et d'arrivée. On peut affiner cette recherche en indiquant les modes de transport que l'on préfère, ou le critère de trajet plus rapide ou plus court.

Question :

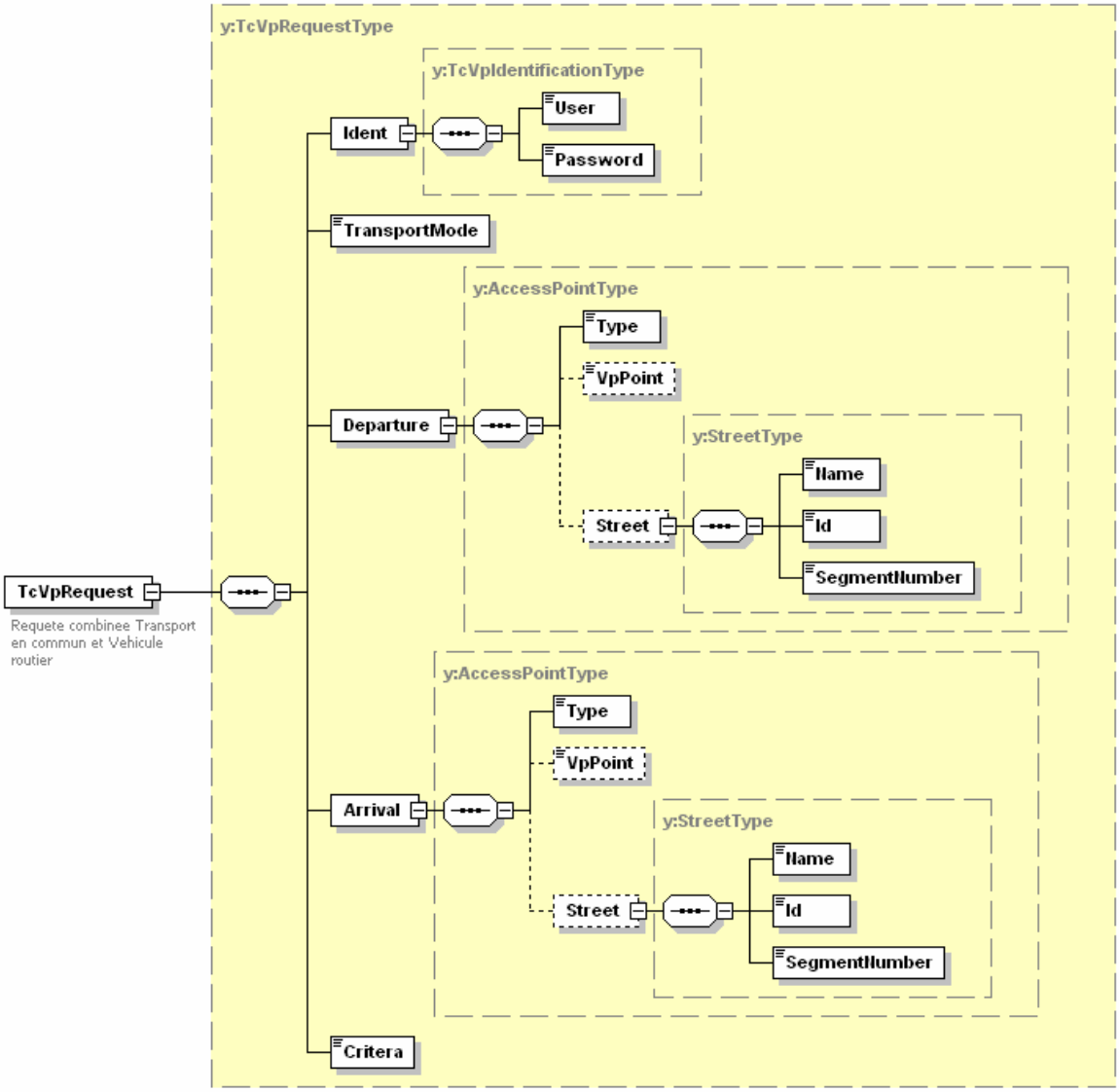


Figure 47 - Diagramme de la méthode TcVpRequest

En réponse à notre précédente question, le calcul d'itinéraire va nous retourner le message *TcVpResponse* comme illustré dans le diagramme suivant. Il sera constitué de deux sous-réponses, une concernant les transports en commun, TC et une autre concernant le mode routier, VP. A l'origine ce service a servi à comparer les services offerts par les transports en commun de la RATP avec l'alternative de l'utilisation de son véhicule personnel.

Réponse VP :

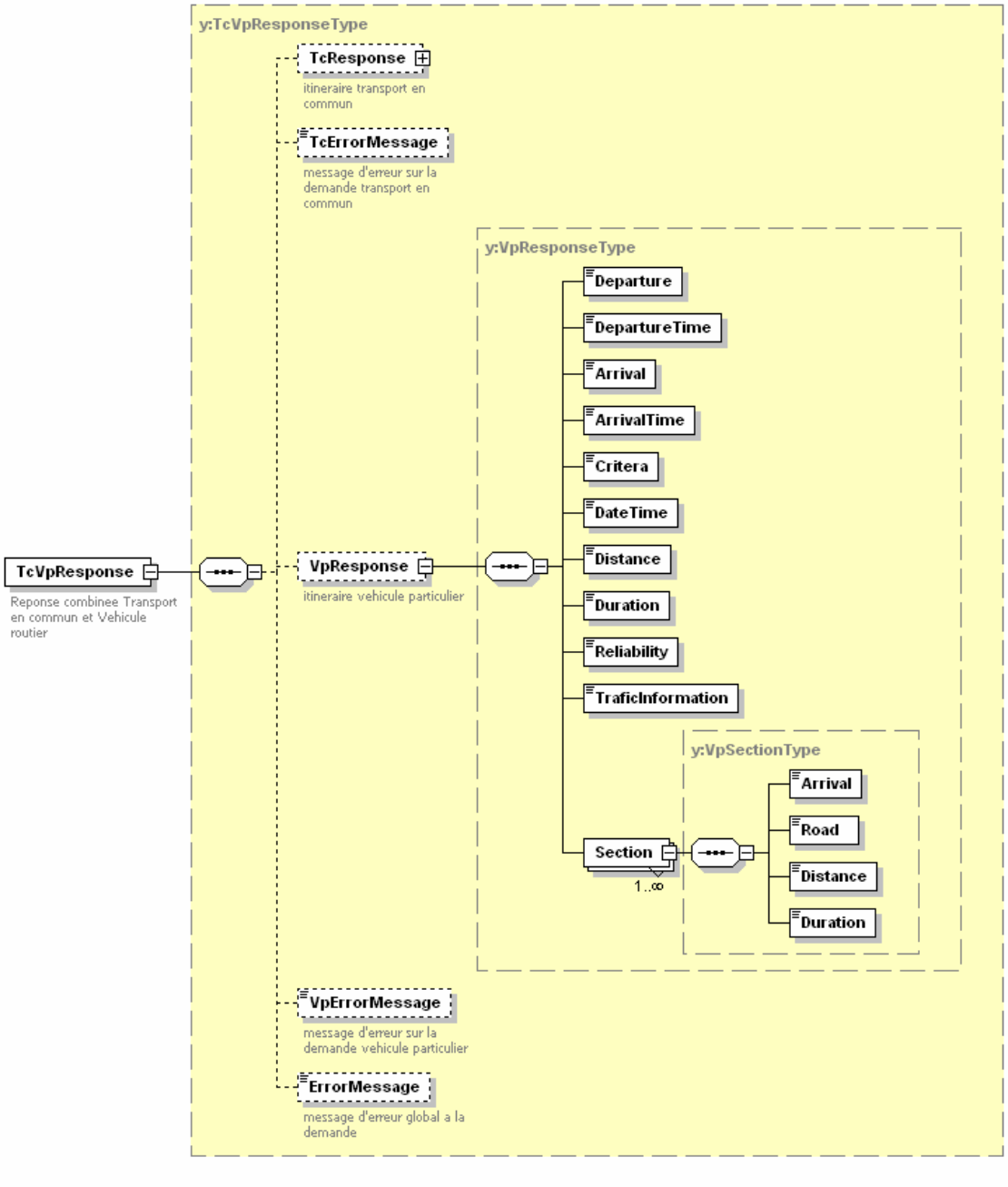


Figure 48 - Diagramme de la méthode TcVpResponse

Nous disposons dans ce message d'une première réponse relatif à l'usage d'un véhicule personnel et de la route. On a donc une réponse avec une liste de tous les segments d'itinéraire routier proposés.

Dans le schéma suivant nous allons illustrer la réponse liée à la solution trouvée par RIDER relative au transport en commun. On remarquera que pour chaque segment constituant un itinéraire, on indique les horaires de départ et d'arrivée ainsi que les temps d'attente ou de marche pour les correspondances.

Réponse TC :

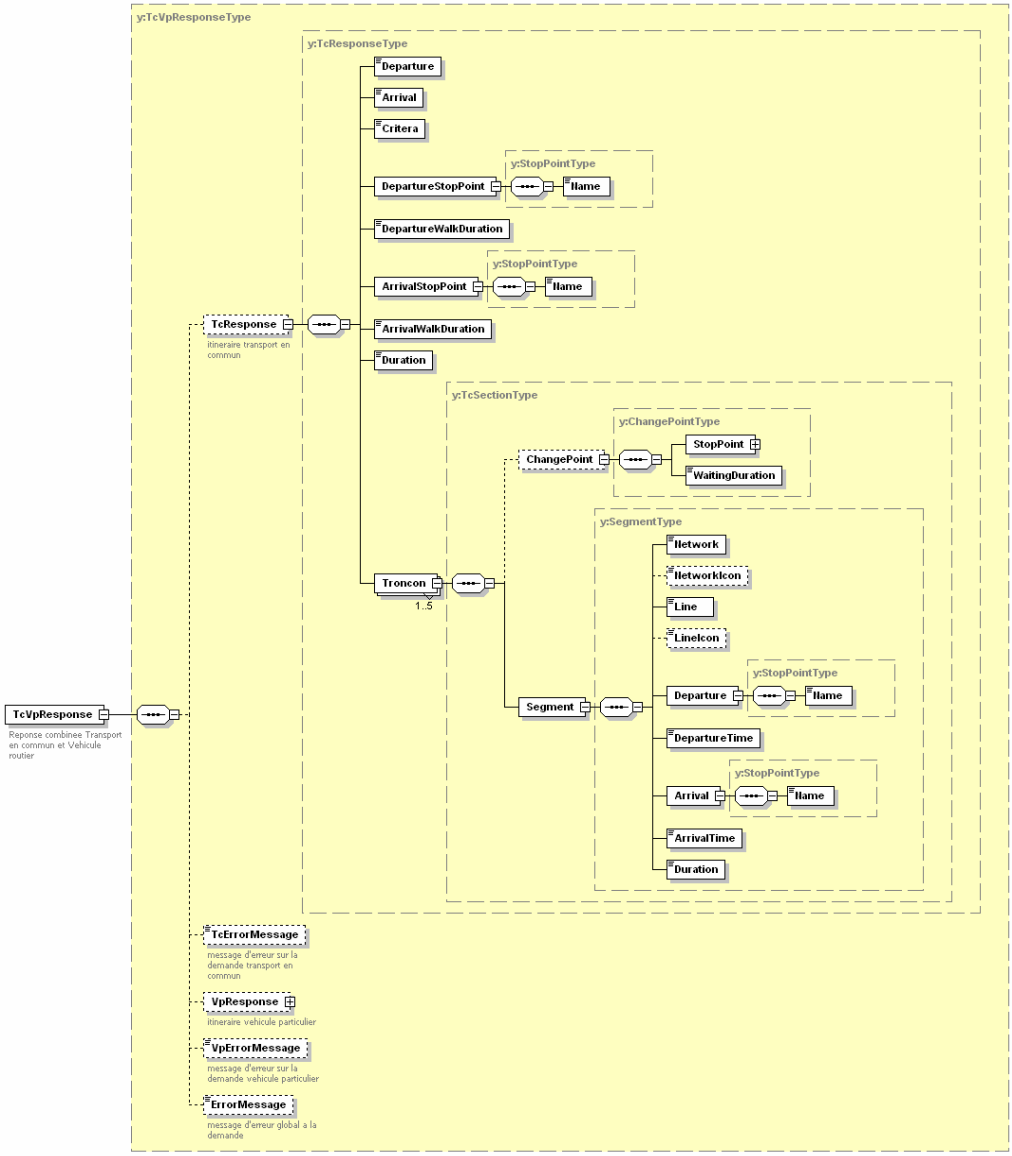


Figure 49 - Diagramme de la méthode TcVpResponse

6.7 DIAGNOSTIC

Nous venons de présenter un exemple d'interfaçage avec le logiciel de calcul d'itinéraire RIDER. Il se limite au calcul d'itinéraire et aux horaires des services correspondants aux itinéraires trouvés. Pour utiliser cette interface il n'a pas été nécessaire de connaître à priori comment RIDER effectuait son calcul en terme d'algorithme ou encore comment il applique les critères de recherche que nous pouvons spécifier à l'aide de cette interface. Néanmoins nous pouvons communiquer avec ce logiciel à l'aide de ce protocole et des méthodes qui ont été présentées. Le document WSDL qui va suivre illustre la spécification d'un Service Web s'interfaçant avec le calcul d'itinéraire RIDER et exposant de manière publique, selon la sémantique de ce langage, une interface d'accès conforme à Trident.

C'est ce service Web qui sera implémenté dans le cadre du démonstrateur. Il s'agit en fait d'une façade pouvant être réutilisée avec d'autres logiciels de calcul d'itinéraire.

Exemple de spécification en WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
   name="MTService"
   targetNamespace="http://www.mobilitime.com/schema/wsdl/mtws"
   xmlns:mt="http://www.mobilitime.com/schema/wsdl/mtws"
   xmlns:mtxsd="http://www.mobilitime.com/schema/xml/mtsystem"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.mobilitime.com/schema/wsdl/mtws"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Created by NGUYEN Tuan Anh using
Cape Clear Studio SOA Editor - http://www.capeclear.com</wsdl:documentation>
   <wsdl:import
      location="../xml/mobilitime/MT_Global_schema.xsd"
      namespace="http://www.mobilitime.com/schema/xml/mtsystem"/>
   <wsdl:types>
      <xsd:schema
         targetNamespace="http://www.mobilitime.com/schema/wsdl/MTService"
         xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:mtxsd="http://www.mobilitime.com/schema/xml/mtsystem"> </xsd:schema>
   </wsdl:types>
   <wsdl:message name="ObjectResponseMessage">
      <wsdl:part name="objectResponse" type="mtxsd:ObjectResponseType"/>
   </wsdl:message>
   <wsdl:message name="ItineraryCalculationResponseMessage">
      <wsdl:part
         name="itineraryCalculationResponse"
         type="mtxsd:ItineraryCalculationResponseType"/>
   </wsdl:message>
   <wsdl:message name="PtStopPointsResponseMessage">
      <wsdl:part name="ptStopPointsResponse" type="mtxsd:PtStopPointsResponseType"/>
   </wsdl:message>
   <wsdl:message name="ObjectRequestMessage">
      <wsdl:part name="objectRequest" type="mtxsd:ObjectRequestType"/>
   </wsdl:message>
   <wsdl:message name="PtStopPointsRequestMessage">
      <wsdl:part name="ptStopPointsRequest" type="mtxsd:PtStopPointsRequestType"/>
   </wsdl:message>
   <wsdl:message name="TimeTableResponseMessage">
      <wsdl:part name="timeTableResponse" type="mtxsd:TimeTableResponseType"/>
   </wsdl:message>
   <wsdl:message name="TimeTableRequestMessage">
      <wsdl:part name="timeTableRequest" type="mtxsd:TimeTableRequestType"/>
   </wsdl:message>
   <wsdl:message name="ItineraryCalculationRequestMessage">
      <wsdl:part
         name="itineraryCalculationRequest"
         type="mtxsd:ItineraryCalculationRequestType"/>
   </wsdl:message>
   <wsdl:portType name="MTServicePortType">
      <wsdl:operation name="ObjectRequest">
         <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Return an Object in function of its
ObjectId</wsdl:documentation>
         <wsdl:input message="tns:ObjectRequestMessage"/>
         <wsdl:output message="tns:ObjectResponseMessage"/>
      </wsdl:operation>
      <wsdl:operation name="PtStopPointsRequest">
         <wsdl:input message="tns:PtStopPointsRequestMessage"/>
         <wsdl:output message="tns:PtStopPointsResponseMessage"/>
      </wsdl:operation>
```

```
      <wsdl:operation name="ItineraryCalculationRequest">
         <wsdl:input message="tns:ItineraryCalculationRequestMessage"/>
         <wsdl:output message="tns:ItineraryCalculationResponseMessage"/>
      </wsdl:operation>
      <wsdl:operation name="TimeTableRequest">
         <wsdl:input message="tns:TimeTableRequestMessage"/>
         <wsdl:output message="tns:TimeTableResponseMessage"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MTServiceBinding" type="tns:MTServicePortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="ObjectRequest">
         <soap:operation
            soapAction="capeconnect:MTService:MTServicePortType#ObjectRequest"/>
         <wsdl:input>
            <soap:body parts="objectRequest" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="objectResponse" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="PtStopPointsRequest">
         <soap:operation
            soapAction="capeconnect:MTService:MTServicePortType#PtStopPointsRequest"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ItineraryCalculationRequest">
         <soap:operation
            soapAction="capeconnect:MTService:MTServicePortType#ItineraryCalculationRequest"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="TimeTableRequest">
         <soap:operation
            soapAction="capeconnect:MTService:MTServicePortType#TimeTableRequest"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="MTService">
      <wsdl:port binding="tns:MTServiceBinding" name="MTServicePort">
         <soap:address location="http://localhost:9000/ccx/MTService"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```