

**FO1.4.9 Architecture des messages, mécanismes de sécurité et protocole d’échanges**

Version : 0.4.2

Date de version : 11/05/2025

|  |  |
| --- | --- |
| Une image contenant signe, extérieur, ciel, rue  Description générée automatiquement | Opération réalisée avec le concours des Investissements d’avenir de l’Etat confiés à l’ADEME |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Projet** | **Zone** | **Lot** | **Phase** | **Générations** | **Type de document** | **Emetteur** | **Numéro Chrono** | **Version** | **Indice** |
| FM | TZ | L4 | TTP | G2 | LIV | SIC | 00118 | 0 | 4.2 |

Informations du document

Périmètre de diffusion : Interne

Type : Initial

Date prévue de livraison : xxx

Statut : en cours

**Auteurs :**

|  |  |  |
| --- | --- | --- |
| **Pilote(s)** | **Organisation** | **Rôle dans le projet** |
|  |  |  |
|  |  |  |
|  |  |  |
| **Contributeurs** | **Organisation** | **Rôle dans le projet** |
| MAHMOUD Mohamed Ali | SICEF | Architect |
|  |  |  |
|  |  |  |

**Table de révision :**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Contenu de la modification** |
| 0.1 | 16/09/2024 | Création de document pour la version POC Simulink |
| 0.2 | 01/12/2024 | Ajout de chapitre : Restructuration de protocole d’échange entre supervision et l’AD |
| 0.3 | 23/12/2024 | Correctif au niveau 3.3.1  Ajouter de 3.3.2 Règles pour les champs  Modification ISAuthorizationRequest ajout de direction  Modification de vehicleTrackingInfo ajout d’un champ pour le nom de systemMetric  ~~MissionAlignement 🡺~~ MissionAssignement page 18  Ajout d’un exemple pour le message ISLMTVCommand page 20 2.3.2. Structure des messages de l’AD mise à jour des tableaux  2.3.1. Structure des messages de supervision mise à jour des tableaux  Les valeurs de ‘systemMetric’ mise à jour de tableau page 26 |
| 0.3.1 | 24/12/2024 | 1. Prise en compte de remarques de l’équipe safety dans l’email de 24 decembre 12 :53PM 2. Simplification de la représentation de mission (Notion de matrice comme représentation est clarifié) 3. Clarification de l’utilisation de notion [] 🡺 indique un séquence des valeurs. |
| 0.3.2 | 26/12/2024 | Correctifs soulignés en jaune |
| 0.3.3 | 30/12/2024 | Correctifs en dorées |
| 0.3.4 | 19/01/2025 | Correctifs en saumon  sequenceNumber sur 3 octets  Les valeurs booléennes codé sur 1 bit au lieu d’un octet  Autres corrections |
| 0.4 | 24/02/2025 | * Première approche supprimée * Approche finale sera la deuxième approche en full numérique * Ajout d’une note à la fin de point 1.2. Contenu des messages * Corrections de quelques fautes d’orthographes sur les noms de champ. * Mise à jour de la structure des message checkStatus/ MissionAssignement/IsUpdateMission/vehicleTrackingInfo * ToDo prochaine version :  - review et mise à niveau de point 4. |
| 0.4.1 | 19/03/2025 | * Format d’encodage de localisation dans vehicleTrackingInfo PKM et Odo sont en UInt32 * Taille de message envoyé par supervision toujours sur 1460 octets * sectionId dans tous les message est codé sur 2 octets * Ajout de message ISKnowledge * Ajout de message ResumeMission * Mécanismes de sécurité et sûreté d’échange à jour * Structure de message à envoyer * Séquencement d’envoi * Messages ponctuels et séquencement * Terminologie à jour Triplet à la place de polyline * Les réponses (valeurs booléennes) sont sur 1 byte au lieu d’un bit |
| 0.4.2 | 16/05/2025 | * Ajout de message ISLMTVRelease   Ajout des états au misisionStatus de message "missionStatus": "int", [inProgress, Stopped,emergencyStop,updatedMission,LTVTINCharge,Canceled]  Ajout de champ lastSectionReleased "lastSectionReleased": "int"  Ajout de champ BatteryStatus "BatteryStatus": "int|int", [batChInProgress (0,1) |AutonLevel (int)] |

|  |  |
| --- | --- |
| Liste des exigences | Version |
| EX- FP1.4.9.1 | 1 |
| EX- FS1.4.9.1.1 | 1 |
| EX- FS1.4.9.1.2 | 1 |
| EX- FS1.4.9.1.3 | 1 |
| EX- FS1.4.9.1.4 | 1 |
| EX- FS1.4.9.1.5 | 1 |
| EX- FS1.4.9.1.6 | 1 |
| EX- FP1.4.9.2 | 1 |
| EX- FS1.4.9.2.1 | 1 |
| EX- FS1.4.9.2.2 | 1 |
| EX- FS1.4.9.2.3 | 1 |
| EX- FS1.4.9.2.4 | 1 |
| EX- FS1.4.9.2.5 | 1 |
| EX- FS1.4.9.2.6 | 1 |
| EX- FS1.4.9.2.7 | 1 |
| EX- FS1.4.9.2.8 | 1 |
| EX- FS1.4.9.2.9 | 1 |
| EX- FS1.4.9.2.10 | 1 |

Table des matières

[Informations du document 2](#_Toc193750204)

[0. Généralités 6](#_Toc193750205)

[0.1. Objet du document 6](#_Toc193750206)

[1. Les messages échangés entre le véhicule et la supervision 6](#_Toc193750207)

[1.1. Synthèse des messages 6](#_Toc193750208)

[1.2. Contenu des messages : 7](#_Toc193750209)

[1.2.1. Messages envoyés par la supervision 7](#_Toc193750210)

[1.2.2. Messages émis par le véhicule 10](#_Toc193750211)

[2. Restructuration de protocole de communication 12](#_Toc193750212)

[2.1. Approche numérique 12](#_Toc193750213)

[2.1.1. Protocole de mappage entre le format string et numérique 12](#_Toc193750214)

[2.2. Catalogue des codes 13](#_Toc193750215)

[2.2.1. Règles de plages numérique des types de messages 13](#_Toc193750216)

[2.2.2. Règles de type 13](#_Toc193750217)

[2.2.3. Règles pour les champs 13](#_Toc193750218)

[2.2.1. Structure des messages de supervision 14](#_Toc193750219)

[2.2.2. Structure des messages de l’AD 23](#_Toc193750220)

[3. Exigences 31](#_Toc193750221)

[4. Mécanismes de sécurité et sûreté d’échange 47](#_Toc193750222)

[4.1.1. Techniques de cryptographique : 48](#_Toc193750223)

[4.2. Protocoles d’échange entre AD et Supervision : 55](#_Toc193750224)

[4.2.1. Messages périodiques 55](#_Toc193750225)

[4.2.2. Message ponctuels envoyés par les véhicules : 56](#_Toc193750226)

[4.2.3. Message ponctuels envoyés par la supervision : 56](#_Toc193750227)

[4.2.4. Messages Ponctuels et séquencement : 57](#_Toc193750228)

[4.3. Tests à réaliser 58](#_Toc193750229)

# Généralités

## Objet du document

Le but de ce document est de détailler la structure des différents messages échangés entre le véhicule et la supervision.

Définir les exigences et les mécanismes de contrôle et de sécurité des données échangées, ainsi que les protocoles d’échanges.

# Les messages échangés entre le véhicule et la supervision

## Synthèse des messages

Les messages émis par la supervision sont :

* InscriptionMessageResponse : la réponse envoyée par la supervision suite à un message d’inscription envoyé par le véhicule
* CheckStatus : c’est le message envoyé par la régulation vers le véhicule pour demander si le véhicule a la capacité de faire une mission
* MissionAssignment : c’est le message envoyé par la régulation vers le véhicule pour lui confier une mission, la mission complète est indiquée au véhicule en lui indiquant la prochaine mission partielle.
* ISAuthorization : c’est la réponse de l’ATP au véhicule par suite d’une demande d’autorisation envoyé par ce dernier, les valeurs possibles pour ce message sont [YES, NO].
* ISUpdateMission : c’est le message envoyé par la régulation en cas de changement de mission pour la mettre à jours au niveau de l’AD.
* IsStopCommand : message envoyé par le système de supervision pour ordonner l'arrêt du véhicule.
* ISLMTVCommand : message envoyé par le système de supervision pour imposer une limitation de vitesse.
* IsAnnulationReservationSection : message envoyé par le système de supervision pour notifier un véhicule de l'annulation d'une réservation préalablement confirmée.
* ISAcknolege : message envoyé par le système de supervision au véhicule afin de confirmer le maintien actif de la communication entre les deux entités dans le sens supervision vers l’AD.
* ResumeMission : message envoyé par le système de supervision au véhicule a fin de le demander de reprendre la mission.

Les messages émis par les véhicules sont :

* InscriptionMessage : C’est le message envoyé par le véhicule pour s’inscrire à la supervision.
* CheckStatusResponse : C’est la réponse envoyée par le véhicule à la suite d’une demande de la régulation.
* MissionAssignmentResponse: C’est la réponse envoyée par le véhicule à la suite d’attribution de mission avec le message ‘MissionAssignment ‘.
* ISAuthorizationRequest : C’est le message envoyé par le véhicule pour demander l’autorisation d’une section.
* VehicleTrackingInfo : C’est le message envoyé par le véhicule chaque 100 ms contenant différentes informations nécessaires pour la supervision.
* IsUpdateMissionResponse : C’est le message envoyé par le véhicule pour confirmer la prise en charge de la mise à jour de la mission.
* IsReleaseRequest : c’est le message envoyé par l’AD pour demander la libération d’une section.

## Contenu des messages :

### Messages envoyés par la supervision

***InscriptionMessageResponse***

"InscriptionMessageResponse": {

"messageId": "int",

"timeStamp": "int[7]",

"sequenceNumber": "int",

"response" : "int"

}

***CheckStatus:***

"CheckStatus": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"idTopo": "int",

"mission": {

"missionId": "int",

"missionType": "int",

"TripNumber": "int",

"NodesByTrip": "int",

"Triplets": [

{

"Triplet": "int [3]",

},

{

"Triplet": "int [3]",

}

]

}

}

***MissionAssignment:***

"MissionAssignment": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"checkStatusResponseMessageId": "int",

"mission": {

"missionId": "int",

"missionType": "int",

"TripNumber": "int",

"NodesByTrip": "int",

"Triplets": [

{

"Triplet": "int [3]",

},

{

"Triplet": "int [3]",

}

{

"Triplet": "int [3]",

….

}

]

}

}

***ISAuthorization***

"ISAuthorization": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"requestMessageId": "int",

"sectionId": "int",

"response": "int"

}

***ISStopCommand***

"ISStopCommand": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"StopCommand": "int", # ‘Soft’ ou ‘Hard’ à vérifier si un arrêt brutale pourrait être nécessaire

}

***ISLMTVCommand***

"ISLMTVCommand": {

"messageId": "int",

"timeStamp": "int[7]",

"sequenceNumber": "int",

"LTVNumber": "int",

"LTVRequest": [

{

"velocityCmd": "int",

"zone": {

"startNodeId": "int",

"endNodeId": "int"

}

},

{

"velocityCmd": "int",

"zone": {

"startNodeId": "int",

"endNodeId": "int"

}

}

]

}

***UpdateMission***

"ISUpdateMission": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"idTopo": "int",

"mission": {

"missionId": "int",

"TripNumber": "int",

"NodesByTrip": "int",

"Triplets": [

{

"Triplet": "int [3]",  
 },....

{

"Triplet": "int [3]",

}

]

}

}

***ISAnnulationReservationSection***

"ISAnnulationReservation": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"sectionId": "int" ~~// en cas d’une réservation pour plusieurs sections de voie en envoie les ids de toutes les sections id1|id2~~

}

~~Il reste à préciser le contenu lors d’une mise à jour, doit-on envoyer seulement le reste de chemin à parcourir ou la totalité du chemin (je préfère envoyer le reste, mais c’est à décider avec C&A)~~

***ISAcknowledge***

"ISAcknowledge": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

}

***ResumeMission***

"ResumeMission": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"missionId": "int",

}

### Messages émis par le véhicule

***InscriptionMessage***

"InscriptionMessage": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"vehicleId" : "int"

}

***CheckStatusResponse***

"CheckStatusResponse": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"response": "int",

"vehicleId": "int"

}

***MissionAssignmentResponse***

"MissionAssignmentResponse": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"response": "int",

"vehicleId": "int"

}

***ISAuthorizationRequest***

"ISAuthorizationRequest": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"sectionId": "int [3]",

"Direction": "int",

"vehicleId": "int"

}

***ISReleaseRequest***

"ISReleaseRequest ": {

"messageId": "int",

"timeStamp": "int [7]",

"sequenceNumber": "int",

"sectionId": "int",

"vehicleId": "int"

}

***VehicleTrackingInfo***

"VehicleTrackingInfo": {

          "messageId": "int",

          "timeStamp": "int",

          "sequenceNumber": "int",

          "vehicleId": "int",

          "globalVehicleStatus": "int", [vehicle OK, vehicle NOK]

          "missionId"": "int",

"missionStatus": "int", [inProgress, Stopped,emergencyStop,updatedMission,LTVTINCharge,Canceled]

"executedRequestId": "int",

    "modeUpdateMissionStatus": "int|int", ~~[état d’exécution, identifiant de mission]~~ [idMission,int(0,1)]

"sectionId": "int",

SectionReleased

          "currentVitesse": "int",

"localization": "int [4]", // [PKM|SystemMetric|odo|isAscendant Or isDescendant]  
 "globalStatusLoc": "int", [AtHub, AtPC,AtLine,AtPA,outSecurityZone,inSecurityZone]

"BatteryStatus": "int|int", [batChInProgress (0,1) |AutonLevel (int)]

      }

P.S : Le caractère de fin message choisi est ∅ (code ASCII 237)   
  
[http ://www.table-ascii.com/](http://www.table-ascii.com/)

***Note :***

Cette description n’est pas une mise en place d’une solution technique se basant sur la structuration des données JSON coté supervision ou AD. Mais une description de la structure des données et ces types des différents messages.

Dans le périmètre de système AD les messages sont en version numérique.

L’utilisation de JSON dans le périmètre de système supervision reste un point à évaluer. Car dans une approche standard le JSON est typiquement manipulé avec une allocation dynamique dans la plupart des bibliothèques. Avec des approches spécifiques il est possible de d’utiliser les allocations de mémoire statique avec de JSON.

Soit avec des bibliothèque JSON avec des allocations statiques comme JSMN ou Frozen pour le C.

Pour le C#, l'utilisation d'une allocation de mémoire statique pour manipuler du JSON est plus complexe que dans des langages comme C, car C# repose principalement sur le Garbage Collector et l'allocation dynamique via le tas (heap). Cependant il y a des approches possibles permettant d’éviter l’utilisation de l’allocation dynamique de mémoire.

Pour manipuler du JSON sans allocation dynamique, il est possible je pense d’utiliser des approches plus spécifiques comme le Span<T> qui permettent de gérer des buffers alloués sur la pile. (à vérifier).

En revanche, le problème lié à l'utilisation de bibliothèques JSON non certifiées ne peut pas être résolu par une simple mise en œuvre ces approches.

Pour cette raison l’approche la plus simple et d’utiliser des structures des données prédéfinies en mémoire et au lieu de manipuler des objets JSON sous forme de chaines, on peut directement les manipuler dans des structures des données statiques.

# Restructuration de protocole de communication

## Approche numérique

### Protocole de mappage entre le format string et numérique

Ce protocole vise à simplifier les implémentations Simulink en évitant l’utilisation de chaînes de caractères. Un mapper (encodeur/décodeur) intégré au niveau de la supervision appliquera les règles de mappage pour transmettre les messages au format numérique vers l’AD ou les convertir en format représentatif lors de la réception de messages provenant du véhicule.

Toutes les opérations d’enregistrement de l’historique des événements au niveau de la supervision seront effectuées sous forme textuelle, sans conversion en format numérique.

Dans une seconde phase, une procédure similaire sera mise en œuvre au niveau du bloc de monitoring de l’AD. Celui-ci permettra soit de conserver l’historique sous un format textuel représentatif, soit de l’extraire en texte pour les analyses.

## Catalogue des codes

Une première version est de rester sur de JSON mais avec des types plus adaptés à l’environnement de développement Simulink

### Règles de plages numérique des types de messages

|  |  |
| --- | --- |
| Type message | Intervalle types |
| Messages de Supervision | [1-150] |
| Messages de l’AD | [160-255] |

### Règles de type

Dans l’environnement Simulink :

* Les entiers sur 1 octet sont stockés dans des variables de type Uint8.
* Les entiers sur 2 octets sont stockés dans des variables de type Uint16.
* Les entiers sur 3 octets sont stockés dans des variables de type Uint24.

### Règles pour les champs

|  |  |
| --- | --- |
| Champs | Format |
| Séquence de fin message | Int [4] “0000” |
| Version de protocole | Séquencement de 2 entiers « 41 » Le premier chiffre indique l’année, tandis que le dernier correspond à la version du protocole pour cette année. cette valeur est codé sur 1 octet. |

\*D’autres règles à définir dans le cas de besoin

**Identifiants de type des messages Supervision :**

|  |  |
| --- | --- |
| Message | Id type |
| InscriptionMessageResponse | 1 |
| CheckStatus | 2 |
| MissionAssignment | 3 |
| ISAuthorization | 4 |
| ISStopCommand | 5 |
| ISLMTVCommand | 6 |
| ISUpdateMission | 7 |
| ISAnnulationReservation | 8 |
| ISKnowledge | 9 |
| ResumeMission | 10 |
| LMTVRelease | 11 |

**Identifiants de type des messages AD :**

|  |  |
| --- | --- |
| Message | Id type |
| InscriptionMessage | 201 |
| CheckStatusResponse | 202 |
| MissionAssignmentResponse | 203 |
| ISAuthorizationRequest | 204 |
| ISReleaseRequest | 205 |
| VehicleTrackingInfo | 206 |

*La taille de l’identifiant de type est de 1 octet. Cet identifiant se situe dans l’intervalle [1, 255], correspondant aux valeurs représentées par un octet non signé 28.*

### Structure des messages de supervision

**Message InscriptionMessageResponse**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| Type message (Name) | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| Response | Int | 1 octet ~~bit~~ |

**Exemple de message**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | Response [0,1] | Version de protocole | Séquence de fin de message |
| 1 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**CheckStatus**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| TypeMessage (Name) | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| idTopo | int | 2 octets |
| missionId | Int | 2 octets |
| TripNumber | Int | 1 octet |
| NodesByTrip | Int | 1 octets |
| Mission   |  |  |  |  | | --- | --- | --- | --- | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | Séquence des Triplets représentant les trips de la mission  Chaque Triplet est sous forme de 3 ids des nœuds | Les identifiants des nœuds sont codé sur 2 octets. |

*Règle de configuration à respecter :*

Une mission doit contenir au maximum 32 trips, et chaque trip ne doit pas dépasser 32 nœuds.

Les données de la mission sont transmises sous un format où chaque trip est représenté par une Triplet. Plutôt que d’envoyer la liste complète des nœuds d’un trip, seuls trois nœuds définissant la Triplet sont transmis. Dans le cas où le un nœud représentera une Bifurcation ou trifu

L’objectif de cette règle est d’optimiser la gestion de la mémoire au niveau du calculateur.

- En appliquant la règle de simplification, seuls 96 nœuds (32 trips × 3 nœuds par trip) seront effectivement envoyés, ce qui réduit la charge mémoire et optimise le traitement des données.

Les informations TripNumber et NodesByTrip sont conservées dans le message afin d’exporter les paramètres de configuration vers le véhicule.

Si le protocole de messagerie évolue en raison d’un changement des règles de configuration sur la taille de la mission, l’AD sera en mesure de détecter toute incohérence entre la version de configuration prise en charge et la version du logiciel installé sur le calculateur.

**Exemple de message**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | idTopo | missionId | TripNumber | NodesByTrip | Mission | Version protocole | Séquence fine de message |  |
| 2 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 123 | 105 | 32 | 32 | Voir M | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |  |

**M** représente les Triplets représentant la mission.

|  |  |  |
| --- | --- | --- |
| 1 | 19 | 32 |
| 50 | 56 | 59 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Cette représentation correspond au même exemple que dans la version précédente du document, où le message ne contient que les deux Triplets représentant les deux trips.

Il revient à l’AD de reconstruire le containeur statique en associant les identifiants des nœuds aux deux trips, puis d’initialiser les valeurs restantes avec la valeur par défaut 65535.

**MissionAssignement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Champ | | Type | Taille | |
| TypeMessage (Name) | | Int | 1 octet | |
| messageId | | Int | 1 octet | |
| timeStamp | | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | | |
| sequenceNumber | | Int | 3 octets | |
| checkStatusResponseMessageId | | int | 1 octet | |
| missionId | | Int | 2 octets | |
| TripNumber | Int | | | 1 octet | |
| NodesByTrip | Int | | | 1 octets | |
| Mission   |  |  |  |  | | --- | --- | --- | --- | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | Séquence des Triplets représentant les trips de la mission  Chaque Triplet est sous forme de 3 ids des nœuds | | | Les identifiants des nœuds sont codé sur 2 octets. | |

**Exemple de message**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | checkStatusResponseMessageId | missionId | TripNumber | NodesByTrip | Mission | Version de protocole | Séquence fine de message |
| 3 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 1 | 32 | 32 | Voir M | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISAuthorization**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISAuthorization(Name) | int | 1 octet |
| messageId | int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| requestMessageId | Int | 2 octets |
| sectionId | Int | 2 octets |
| response | int | 1 octet |

**Exemple de message**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | requestMessageId | sectionId | reponse | Version Protocole | Séquence de fin de message |
| 4 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 1 | 1 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISStopCommand**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISStopCommand(Name) | Int | 1 octets |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| StopCommand | Int (0,1), # 0 : ‘Soft’ ou 1 : ‘Hard’ | 1 octet |

**Exemple de message**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | stopCommand | Version de protocole | Séquence de fin de message |
| 5 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 0 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISLMTVCommand**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISLMTVCommand(Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| LTVNumber | Int | 1 octet |
| velocityCmd | ~~Float~~ Int | 2 octets |
| Zone | Un séquencement de 2 entiers | |  |  | | --- | --- | | 2octest | 2octets | |

**Exemple de message**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | LTVNumber | velocityCmd1 | Zone1 | velocityCmd2 | Zone2 | Version de protocole | Séquence de fin de message |
| 5 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 2 | 15.0 | |  |  | | --- | --- | | 102 | 118 | | 20 | |  |  | | --- | --- | | 180 | 200 | | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISUpdateMission**

|  |  |  |  |
| --- | --- | --- | --- |
| Champ | Type | | Taille |
| ISUpdateMission(Name) | Int | | 1 octet |
| messageId | Int | | 1 octet |
| timeStamp | Séquencement de 7 valeurs de type int | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | | 3 octets |
| idTopo | Int | | 2 octets |
| missionId | int | | 2 octets |
| TripNumber | | Int | 1 octet |
| NodesByTrip | | Int | 1 octets |
| Mission   |  |  |  |  | | --- | --- | --- | --- | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | Triplet   |  |  |  | | --- | --- | --- | |  |  |  | | | | Séquence des Triplets représentant les trips de la mission  Chaque Triplet est sous forme de 3 ids des nœuds | Les identifiants des nœuds sont codé sur 2 octets. |

La solution proposée pour la mise à jour de la mission consiste à transmettre l’ensemble des données de la mission à l’AD, incluant les Triplets représentant les trips.

**Exemple de message**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | idTopo | missionId | TripNumber | NodesByTrip | Mission | Version de protocole | Séquence de fin de message |
| 6 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 105 | 123 | 32 | 32 | Voir M | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

|  |  |  |
| --- | --- | --- |
| 1 | 29 | 32 |
| 50 | 51 | 59 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |
| 65535 | 65535 | 65535 |

**ISAnnulationReservation**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISAnnulationReservation(Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| sectionId | Séquencement de n entier codé chacun sur 2 octets. Chaque entier représente l’identifiant de la section à annuler. | - Chaque identifiant de section est codé sur 2 octets.  - Nombre maximum : Jusqu'à 3 sections peuvent être définies.  - Par défaut, les identifiants sont initialisés à zéro.  - Les identifiants égaux à zéro sont ignorés. |

**Exemple de message**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | sectionId | Version de protocole | Séquence de fin de message |
| 7 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISAcknowledge**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISAcknowledge (Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |

**ResumeMission**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISResumeMisssion(Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| missionId | Int | 2 octets |

**ISLMTVRelease**



|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISLMTVRelease(Name) | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 valeurs de type int | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| LMTVNumberToRelease | Int | 1 octet |
| LMTVToRelease | Sequencement de N int (avec N = LMTVToRelease) | N octet |
| LMTVNumber | Int | 1 octet |

* + - 1. Règles d’envoi pour la supervision

La supervision envoie des messages de taille fixe de 1460 octets. Ainsi, si la taille du message à transmettre est inférieure à 1460 octets, l’émetteur doit le compléter avec une série de données par défaut, encodées sur 2 octets et ayant la valeur 65 535.

### Structure des messages de l’AD

**InscriptionMessage**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| InscriptionMessage(Name) | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| vehicleId | Int | 2 octets |

**Exemple de message**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | vehicleId | Version de protocole | Séquence de fin de message |
| 201 | **1** | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | **1** | **1** | **41** | |  |  |  | | --- | --- | --- | | 0 | 0 | 0 | |

**checkStatusResponse**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| CheckStatusResponse(Name) | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| response | Int (0,1) | 1 octet |
| vehicleId | Int | 2 octets |

**Exemple de message**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | response | vehicleId | Version de protocole | Séquence de fin de message |
| 202 | **1** | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | **1** | **1** | **1** | **41** | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**MissionAssignmentResponse**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| MissionAssignmentResponse | Int | 1 octet |
| messageId | Int | 1 octet |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| response | Int (0,1) | 1 octet |
| vehicleId | Int | 2 octets |

**Exemple de message**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | Response | vehicleId | Version de protocole | Séquence de fin de message |
| 203 | **1** | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | **1** | **1** | **1** | **41** | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISAuthorizationRequest**

|  |  |  |
| --- | --- | --- |
| Champ | Type |  |
| ISAuthorizationRequest(Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| sectionId | Séquencement de 3 entiers | |  |  |  | | --- | --- | --- | | 2 octets | 2octets | 2 octets | |
| Direction | Int (0,1) | 1 octet |
| vehicleId | Int | 2 octets |

**Exemple de message**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | sectionId | Direction | vehicleId | Version de protocole | Séquence de fin de message |
| 204 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 1 | 1 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**ISReleaseRequest**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| ISReleaseRequest(Name) | Int | 1 octet |
| messageId | Int | 2 octets |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| sectionId | Int | 2 octets |
| vehicleId | Int | 2 octets |

**Exemple de message**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | timeStamp | sequenceNumber | sectionId | vehicleId | Version de protocole | Séquence de fin de message |
| 205 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 1 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

**VehicleTrackingInfo**

|  |  |  |
| --- | --- | --- |
| Champ | Type | Taille |
| VehicleTrackingInfo(Name) | Int | 1 octet |
| messageId | Int | 3 octets |
| timeStamp | Séquencement de 7 entiers | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets | |
| sequenceNumber | Int | 3 octets |
| vehicleId | Int | 2 octets |
| globalVehicleStatus | Int (0,1) Ok or NOk | 1 octet |
| missionStatus | Int | 1 octet |
| executedRequestId | Int | 2 octets |
| modeUpdateMissionStatus | Un séquencement de 2 entiers ([Etat d’exécution, id de mission]) | |  |  | | --- | --- | | 1 octet | 1 octet | |
| currentVitesse | int | 2 octets |
| sectionId | Int | 2 octets |
| lastSectionReleased | Int | 2 octets |
| localisation | Un séquencement de 4 valeurs. Entier,Entier,Entier,Entier | |  |  |  |  | | --- | --- | --- | --- | | **4 octets** | **2 octet** | **4 octet** | **1 octets** | |
| globalStatusLoc | Int (soit AtHub ou AtPC ou AtLine ou AtPA) | 1 octet |
| BatteryStatus | Cette séquence se compose de deux octets : le premier indique si la batterie est en cours de charge (0 = non, 1 = oui), et le second correspond au niveau d’autonomie. | |  |  | | --- | --- | | 1 octet | 1 octet | |

**Exemple de message**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | messageId | TimeStamp | sequenceNumber | vehicleId | globalVehicleStatus | missionStatus | executedRequestId | modeUpdateMissionStatus | currentVitesse | Localisation | gloablaStausLoc | Version de protocole | Séquence de fin de message |
| 206 | 1 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2024 | 11 | 28 | 5 | 14 | 59 | 998 | | 1 | 1 | 1 | 1 | 26 | |  |  | | --- | --- | | 1 | 120 | | 40.2 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **100.0** | **25.0** | **1.0** | **289.3** | **1** | | 2 | 41 | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | |

Les valeurs de sens de roulage par rapport au système métrique

|  |  |
| --- | --- |
| Status | Valeur |
| Ascendant | 1 |
| Descendant | 0 |

Les valeurs de ‘systemMetric’

|  |  |  |  |
| --- | --- | --- | --- |
| Charactère | Valeur de charactère | Numéro de système métrique | Nom de système métrique |
| A | 1 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| B | 2 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| C | 3 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| D | 4 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| E | 5 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| F | 6 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| G | 7 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| H | 8 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| I | 9 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| J | 10 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| K | 11 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| L | 12 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| M | 13 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| N | 14 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| O | 15 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| P | 16 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| Q | 17 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| R | 18 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| S | 19 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| T | 20 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| U | 21 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| V | 22 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| W | 23 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| X | 24 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| Y | 25 | [1...n] | Séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |
| Z | 26 | [1...n] | séquencement de deux valeurs (1 octet par valeur) 1 et x avec x ∈ [1…n] |

Les différentes valeurs d’état « GlobalStatusLocalisation »

|  |  |
| --- | --- |
| Status | Valeur |
| AtHub | 0 |
| AtPC (plateforme de croisement) | 3 |
| AtLine | 2 |
| AtPA (plateforme d’alignement) | 1 |

\*\*Information version protocole à ne pas oublier

\*\*\*\*

# Exigences

|  |
| --- |
| **Id : EX-** FP1.4.9.1#1 |
| **Exigence : Les messages envoyés par les véhicules doivent respecter les structures définis dans la spécification de protocole d’échange.** |
| **Description :** Les messages :   * ‘InscriptionMessage’ * ‘CheckStatusResponse’ * ‘MissionAssignmentResponse’ * ‘ISAuthorizationRequest’ * ‘ISReleaseRequest’ * ‘VehicleTrackingInfo’   Doivent respecter les structures définies corresponds à chaque message. |
| **Inputs :** Messages à vérifier |
| **Outputs :** Messages vérifiés |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.1#1 |
| **Exigence : Le message ‘InscriptionMessage’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘InscriptionMessage’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 201 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : Un entier codé sur 3 octets * vehicleId : n entier codé sur 2 octets |
| **Inputs :** Données composant le message ‘insciptionMessage’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.2#1 |
| **Exigence : Le message ‘** **CheckStatusResponse’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ CheckStatusResponse’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 202 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : Un entier codé sur 3 octets * response : Un entier codé sur 1 octet (1 ou 0) * vehicleId : Un entier codé sur 2 octets |
| **Inputs :** Données composant le message ‘CheckStatusResponse’ à vérifier. |
| **Outputs :** Les données vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.3#1 |
| **Exigence : Le message ‘MissionAssignmentResponse’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ MissionAssignmentResponse’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 203 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : Un entier codé sur 3 octets * response : Un entier codé sur 1 octet (1 ou 0) * vehicleId : Un entier codé sur 2 octets |
| **Inputs :** Données composant le message ‘MissionAssignmentResponse’ à vérifier. |
| **Outputs :** Les données vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.4#1 |
| **Exigence : Le message ‘ISAuthorizationRequest’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISAuthorizationRequest’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 204 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : un entier codé sur 3 octets * sectionId : un entier codé sur 2 octets * direction : 1 octet (1 ou 0) * vehicleId : un entier codé sur 2 octets |
| **Inputs :** Données composant le message ‘ISAuthorizationRequest’ à vérifier. |
| **Outputs :** Les données vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.5#1 |
| **Exigence : Le message ‘ISReleaseRequest’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISReleaseRequest’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 205 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : Un entier codé sur 3 octets * sectionId : Un entier codé sur 2 octets * vehicleId : Un entier codé sur 2 octets |
| **Inputs :** Données composant le message ‘ISReleaseRequest’ à vérifier. |
| **Outputs :** Les données vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.1.6#1 |
| **Exigence : Le message ‘VehicleTrackingInfo doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘VehicleTrackingInfo’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 205 codé sur 1 octet * ‘messageId’ : Un entier codé sur 3 octets * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequencNumber : Un entier codé sur 3 octets * vehicleId : Un entier codé sur 2 octets * globalVehicleStatus : Un entier pouvant avoir la valeur 0 ou 1, codé sur 1 octet. ( vehicleOK, vehicleNotOK) * missionStatus : Un entier pouvant avoir la valeur 0 ou 1, codé sur 1 octet. (missionOk in progress = 1, missionStopped = 0) * executedRequestId : Un entier codé sur 2 octets. * modeUpdateMissionStatus : Une séquence de deux entiers, chacun codé sur 1 octet. * sectionId : l’identifiant de la section où le véhicule * lastSectionReleased : l’identifiant de la dernière section libéré codé sur 2 octets. * currentVitesse : Un entier codé sur 2 octets (Une vérification du format si une exigence spécifique est définie par la norme ferroviaire doit être faite). * Localisation : Une séquence de 4 entiers, codés respectivement en 2,2,3,1 octet. * globalStatusLoc : Un entier codé sur 1 octet (une correction orthographique sera appliquée dans la version vérifiée 0.4). |
| **Inputs :** Données composant le message ‘VehicleTrackingInfo’ à vérifier. |
| **Outputs :** Les données vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FP1.4.9.2#1 |
| **Exigence : Les messages envoyés par la supervision doivent respecter les structures définis dans la spécification de protocole d’échange.** |
| **Description :** Les messages :   * InscriptionMessageResponse * CheckStatus * MissionAssignment * ISAuthorization * ISStopCommand * ISLMTVCommand * ISUpdateMission * ISAnnulationReservation * ISAcknowledge * ResumeMission   Doivent respecter les structures définies corresponds à chaque message. |
| **Inputs :** Messages à vérifier |
| **Outputs :** Messages vérifiés |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.1#1 |
| **Exigence : Le message ‘InscriptionMessageResponse’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ InscriptionMessageResponse’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 1 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet (L'hypothèse est que le véhicule ne peut pas effectuer plus de 255 inscriptions par jour d'exploitation.) * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : Un entier codé sur 3 octets * Response : Un entier pouvant avoir la valeur 0 ou 1, codé sur 1 octet. |
| **Inputs :** Données composant le message ‘InscriptionMessageResponse’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.2#1 |
| **Exigence : Le message ‘** **CheckStatus’ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘CheckStatus’ doit respecter ce format :   * ‘Name’ = la valeur 2 codé sur 1 octet * ‘messageId’ un entier codé sur 1 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets * IdTopo : un entier codé sur 2 octets * missionId : un entier codé sur 2 octets * TripNumber : un entier codé sur 1 octets * NodesByTrip : un entier codé sur 2 octets * Liste des nœuds : Un séquencement des entiers (TripNumber\*NodesByTrip) chaque entier est codé sur 2 octets. |
| **Inputs :** Données composant le message ‘ CheckStatus’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.3#1 |
| **Exigence : Le message ‘MissionAssignment‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘MissionAssignment’ doit respecter ce format :   * ‘Name’ = Un entier dont la valeur 3 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet (L'hypothèse est que le véhicule ne peut pas effectuer plus de 255 missions par jour d'exploitation.) * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : Un entier codé sur 3 octets * checkStatusResponseMessageId : Un entier codé sur 1 octet * missionId : Un entier codé sur 2 octets * TripNumber : Un entier codé sur 1 octet * NodesByTrip : Un entier codé sur 2 octets * Liste des nœuds : Un séquencement des entiers (TripNumber\*NodesByTrip) chaque entier est codé sur 2 octets. |
| **Inputs :** Données composant le message ‘ MissionAssignment’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.4#1 |
| **Exigence : Le message ‘ISAuthorization‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISAuthorization’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 4 codé sur 1 octet * ‘messageId’ : Un entier codé sur 2 octet * Timestamp : Un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : Un entier codé sur 3 octets * requestMessageId : Un entier codé sur 2 octets * sectionId : Un entier codé sur 2 octets * response : Un entier codé sur 1 octet |
| **Inputs :** Données composant le message ‘ ISAuthorization’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.5#1 |
| **Exigence : Le message ‘ISStopCommand‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISStopCommand’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 5 codé sur 1 octet * ‘messageId’ : Un entier codé sur 2 octet * Timestamp : une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : Un entier codé sur 3 octets * stopCommand : Un entier codé sur 1 octet |
| **Inputs :** Données composant le message ‘ISStopCommand’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.6#1 |
| **Exigence : Le message ‘ISLMTVCommand‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISLMTVCommand’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 6 codé sur 1 octet * ‘messageId’ : Un entier codé sur 2 octet * Timestamp : Une séquence de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : Un entier codé sur 3 octets * velocityCmd : Un entier codé sur 2 octets * Zone : Une séquence de deux entiers, chacun codé sur 2 octets. |
| **Inputs :** Données composant le message ‘ISLMTVCommand’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.7#1 |
| **Exigence : Le message ‘ISUpdateMission‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ ISUpdateMission’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 7 codé sur 1 octet * ‘messageId’ : Un entier codé sur 1 octet * Timestamp : Une séquence de 7 entiers codés sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets. * idTopo : un entier codé sur 2 octets * missionId : un entier codé sur 2 octets * TripNumber : un entier codé sur 1 octet * NodesByTrip : un entier codé sur 2 octets * Liste des nœuds : une séquence des entiers (TripNumber \* NodesByTrip) codés sur 2 octets |
| **Inputs :** Données composant le message ‘ ISUpdateMission’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.8#1 |
| **Exigence : Le message ‘ISAnnulationReservation‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ISAnnulationReservation’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 8 codé sur 1 octet * ‘messageId’ un entier codé sur 2 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets. * sectionId : Une séquence de trois entiers, chacun codé successivement sur 2 octets. |
| **Inputs :** Données composant le message ‘ISAnnulationReservation’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.9#1 |
| **Exigence : Le message ‘ISAcknowledge‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ ISAcknowledge’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 8 codé sur 1 octet * ‘messageId’ un entier codé sur 3 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets. |
| **Inputs :** Données composant le message ‘ ISAcknowledge’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.10#1 |
| **Exigence : Le message ‘ResumeMission‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ ResumeMission’ doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 8 codé sur 1 octet * ‘messageId’ un entier codé sur 1 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets. |
| **Inputs :** Données composant le message ‘ ResumeMission à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

|  |
| --- |
| **Id : EX-** FS1.4.9.2.11#1 |
| **Exigence : Le message ‘ISLMTVRelease‘ doit respecter la structure définie dans la spécification de protocole d’échange de messagerie.** |
| **Description :** Le message ‘ ISLMTVRelease doit respecter ce format :   * ‘Name’ : Un entier dont la valeur 8 codé sur 1 octet * ‘messageId’ un entier codé sur 1 octet * Timestamp : représenté par un séquencement de 7 entiers codé sous ce format  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 2octets | 1octet | 1octet | 1octet | 1 octet | 1octet | 2 octets |  * sequenceNumber : un entier codé sur 3 octets. * LMTVNumberToRelease : un entier codé sur 1 octet. * LMTVToRelease : un séquencement d’entier codés sur 1 octet (nombre d’octet = LMTVNumberToRelease) |
| **Inputs :** Données composant le message ‘ISLMTVRelease’ à vérifier. |
| **Outputs :** Les données de message vérifiées |
| **Sub-exigences :**   * **Fonctionnelles :** * **Non fonctionnelles :** * **De performance : temps de traitement** |
| **Contraintes : à ajouter par l’équipe ‘safety’** |

# Mécanismes de sécurité et sûreté d’échange

***Les mécanismes de sécurité feront l'objet d'une révision dans la prochaine version de document.***

Les champs possédant des informations sécuritaires, impose l’utilisation des mécanismes de sécurité pour assurer le niveau de sécurité exigé.

Les messages doivent contenir :

* MessageID : identifiant unique de message, cette information permet de différencier chaque message individuellement, aidant à capter les duplications et à suivre les messages spécifiques. Cette information sera incluse dans la réponse dans le cas d’une réponse à une requête.
* TimeStamp : Horodatage indiquant le moment où le message était créé, cette information assure au récepteur de vérifier la fraîcheur des informations. Le TimeStamp doit être conforme au norme ISO 8601 sous cette format YYYY-MM-DDTHH:MM:SS.sssZ.
* SequenceNumber : Un numéro de séquence ajouté à chaque message envoyé par l’émetteur. Ceci permet au récepteur de vérifier la séquence des messages provenant de l’émetteur.
* Version : Version de message ou de protocole utilisé, cette information permet de gérer les évolutions du format ou des spécifications du message.

En plus d’autres données utiles de message.

La norme EN50159, précisément les exigences de classe 3 définies 3 types des mécanismes de sécurité et de technique de cryptographiques (B0, B1, A1).

### Techniques de cryptographique :

B0 exige une structure de message avec :

* + Un code de sécurité non cryptographique pour vérifier l’intégrité des données à calculer avant le chiffrement à partir des données de base, dans notre cas sont les données de l’AD à envoyer
  + Le chiffrement est appliqué sur les données AD brutes avec le code non-cryptographique. Ce chiffrement assure la confidentialité des données (dans notre cas informations AD + code non-cryptographique) .
  + Une signature numérique pour garantir l’authenticité et l’intégrité des données chiffrées.

Le code de sécurité non-cryptographique est utilisé pour vérifier l’intégrité des données transmises, différents algorithmes utilisés pour calculer ces codes :

* + - CRC (Cyclic Redundancy Check)
    - Checksum
    - Luhn Algorithm
    - Fletcher
    - Adler-32

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type de Code** | **Principe de Fonctionnement** | **Avantages** | **Limites** | **Applications courantes** |
| **Checksum** | Addition des valeurs des bits/octets | Simple et rapide à calculer | Faible détection des erreurs complexes | Transmission de données simple |
| **CRC (Cyclic Redundancy Check)** | Division polynomiale des données | Très bonne détection des erreurs | Plus complexe à implémenter | Réseaux, télécommunications |
| **LRC (Longitudinal Redundancy Check)** | Addition en colonnes des valeurs | Bonne pour les données structurées | Moins courante, limité aux matrices | Mémoire, stockage de données |
| **Parity Bits** | Ajout d'un bit pour parité paire/impair | Simple à implémenter | Ne détecte que les erreurs sur un seul bit | Mémoire, bus de données |
| **Hamming Code** | Ajout de bits de redondance | Détecte et corrige les erreurs | Plus de redondance nécessaire | Mémoire, communication critique |
| **Simple Arithmetic Codes** | Opérations arithmétiques simples | Facile à calculer et vérifier | Moins robuste que CRC | Applications spécifiques peu complexes |

Tableau 1 : Algorithme de calcul de code non-cryptographique

Le chiffrement, permet de chiffrer les données sensibles de message, différents algorithmes de chiffrement permet de l’assurer, des algorithmes de cryptographiques symétrique ou asymétrique.

* Le chiffrement symétrique utilise la même clé pour chiffrer et déchiffrer les données, il est utilisé pour ses performances rapides et son efficacité.
  + - AES algorithme de chiffrement largement utilisé et approuvé par les normes, avec différents logeurs de clés 128, 192 ou 256 bits
* Le chiffrement asymétrique utilise une paire des clés, une clé publique pour chiffrer et une clé privée pour les déchiffrer.
  + - RSA algorithme de chiffrement asymétrique couramment utilisé pour sécuriser les communications. Avec différentes clés de différentes longueurs typiquement 2048 et 4096.

Voici une liste plus complète des algorithmes de chiffrement :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithme** | **Type** | **Taille des Clés** | **Usage Principal** | **Avantages** | **Inconvénients** |
| **AES** | Symétrique | 128, 192, 256 bits | Données sensibles, réseaux sans fil, fichiers | Très sécurisé, rapide | Complexe comparé à DES |
| **DES** | Symétrique | 56 bits | Historique | Simple à implémenter | Facilement cassable, clé trop courte |
| **3DES** | Symétrique | 112 ou 168 bits | Successeur de DES, historique | Plus sécurisé que DES | Plus lent que AES |
| **RSA** | Asymétrique | 2048 bits ou plus | Échanges de clés, signatures numériques | Très sécurisé | Lourd et lent pour grandes données |
| **ECC** | Asymétrique | 256 bits typiquement | Données, signatures, mobiles | Sécurisé avec des clés plus courtes, performant | Complexe à implémenter |
| **Blowfish** | Symétrique | 32 à 448 bits | Fichiers, bases de données | Flexible, rapide | Vulnérable aux attaques par force brute avec clés courtes |
| **Twofish** | Symétrique | 128, 192, 256 bits | Données, alternative à AES | Très sécurisé, rapide | Moins adopté que AES |

Tableau 2 : Algorithme de chiffrement

***Structure de message final (B0)***

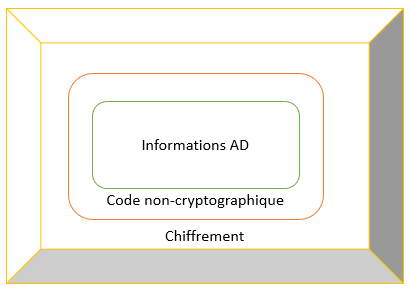


Figure 1 : Structure de message (type B0)

{

"encryptedData": "-----BEGIN ENCRYPTED MESSAGE-----" ………………"-----END ENCRYPTED MESSAGE-----",

}

La partie encryptedData inclut les données AD et le code non-cryptographique. Cela signifie que le paquet des données chiffrées contient le code non-cryptographique, ce qui permet de vérifier l’intégrité des données transmises ((informations brutes de l’AD ou consignes de Supervision).

"encryptedData" : {

"nonCryptographicSecurityCode": "6789ABCD",

“data AD” : {

……

}

}

B1 exige un code de sécurité cryptographique plus que le code non-cryptographique.

Pour le type B1 le chiffrement est optionnelle mais recommandé.

***Structure de message final (B1)***

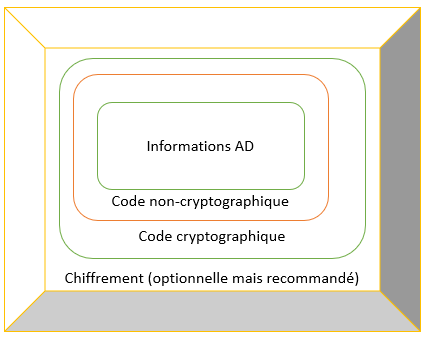


Figure 2 : Structure de message (type B1)

{

"encryptedData": "-----BEGIN ENCRYPTED MESSAGE-----……………-----END ENCRYPTED MESSAGE-----", // (optionnel)

}

L’encryptedData inclus le code non-cryptographique et le code cryptographique en plus des informations brutes de l’AD.

"encryptedData" : {

"nonCryptographicSecurityCode": "6789ABCD",

"cryptographicSecurityCode» : "A1B2C3D4EFGH0012",

“data AD” : {

……

}

}

Le code cryptographique est utilisé pour assurer l’intégrité et l’authenticité des messages. Voici une liste des algorithmes pour générer les codes cryptographiques.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type de Code** | **Principe de Fonctionnement** | **Avantages** | **Limites** | **Applications Courantes** |
| **HMAC (Hash-based Message Authentication Code)** | Utilise une fonction de hachage et une clé secrète | Facile à implémenter, protège contre modifications et attaques par rejeu | Nécessite une clé secrète partagée | Authentification des messages, API, communications sécurisées |
| **CMAC (Cipher-based Message Authentication Code)** | Utilise un chiffrement par bloc (ex : AES) et une clé secrète | Assure intégrité et authenticité des messages | Nécessite une clé secrète partagée | Communications sécurisées, systèmes embarqués |
| **Digital Signatures (Signatures numériques)** | Utilise des algorithmes asymétriques (ex : RSA, ECDSA) | Vérifie l'authenticité et l'intégrité des données, non-répudiation | Plus complexe et computationnellement coûteux | Transactions financières, documents légaux, certificats numériques |
| **Message Authentication Code (MAC)** | Utilise une clé secrète et un algorithme de chiffrement | Assure l'intégrité et l'authenticité des messages | Nécessite une clé secrète partagée | Protocoles de communication, stockage sécurisé |
| **Authenticated Encryption (AE)** | Combine chiffrement et authentification (ex : AES-GCM) | Protège confidentialité et intégrité simultanément | Complexité d'implémentation plus élevée | Communications réseau, stockage de données sensibles |
| **SHA-256 with Secret Key (HMAC-SHA-256)** | Utilise SHA-256 avec une clé secrète | Haute sécurité contre-attaques cryptographiques | Nécessite une clé secrète partagée | Authentification des messages, intégrité des fichiers |

Tableau 3 : Algorithme de code cryptographique

Le type A1 n’exige pas une couche de protection d’accès séparée. Il emploie une protection intégrée, ce qui signifie que la sécurité des messages est garantie par le chiffrement et la signature numérique directement appliqués aux données.

* La signature a pour rôle de garantir l’intégrité et l’authenticité des données chiffrés.

Il existe différents algorithmes de génération de signature, ci-dessous une liste plus exhaustive des algorithmes de signature numérique

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algo** | **Type** | **Usage Principal** | **Sécurité** | **Performance** | **Clés Typiques** | **Limitations** |
| **RSA** | Asymét | Signatures, certificats SSL/TLS | Très sécurisé avec des clés longues | Lourd pour grandes données | 2048 bits ou plus | Lenteur pour grandes quantités de données ; clés longues nécessaires |
| **DSA** | Asymét | Signatures (US Gov standard) | Très sécurisé | Plus rapide pour les signatures | 1024 à 3072 bits | Moins efficace pour la vérification ; vulnérable à des attaques spécifiques si mal implémenté |
| **ECDSA** | Asymét | Signatures numériques, mobiles | Très sécurisé avec clés courtes | Rapide, faible consommation | 256 bits | Complexe à implémenter correctement ; support limité |
| **EdDSA** | Asymét | Signatures haute performance | Très sécurisé, résistant aux attaques | Très rapide, même sur faibles ressources | 256 bits (Ed25519) | Moins de support et adoption comparé à RSA et ECDSA |
| **HMAC** | Symét | Authentification de messages | Très sécurisé avec bon hash | Très rapide, faible consommation | Dépend de la fonction de hachage | Nécessite le partage préalable d'une clé secrète, ne fournit pas de non-répudiation |

*Nécessite le partage préalable d’une clé secrète, ne fournit pas de non-répudiation*

* Étant donné que la clé secrète est partagée entre les parties, toute personne ayant accès à cette clé peut générer un HMAC valide. Cela signifie qu'aucune des parties ne peut prouver de manière exclusive qu'elle est l'auteur du message. ①
* Parce que la clé est partagée, il est impossible de prouver qu'un message a été envoyé par une partie spécifique. Par exemple, si Véhicule 1 et Supervision utilisent la même clé secrète, Véhicule 1 ne peut pas prouver que Supervision a envoyé un message, et vice versa. ②

① Cette faiblesse ne pose pas un problème dans notre cas car les données brutes contenant l’identifiant de l’émetteur et un séquencement de message, ainsi un code cryptographique chiffré avec les données permettent de vérifier l’intégrité et l’authenticité des informations dont l’id de l’émetteur et le numéro de séquence.

② Couvert par le premier argument !

PS : Il faut vérifier si Supervision est capable de gérer plusieurs clés (JE PENSE OUI à vérifier ave C&A). Si c’est le cas il est important de déterminer si le fait de charger une clé différente à chaque itération d’échange pour chaque véhicule impact le temps de calcul.

Dans le cas d’utiliser la même clé pour tous les véhicules, même si l’un des véhicules est devenu vénérables les informations erronées envoyés avec l’id d’un autre véhicule seront divulgués par la supervision par un check des cohérences en se basant sur le numéro de séquence et le timestamp.

***Structure de message final (A1)***

{

"encryptedData » :"-----BEGIN ENCRYPTED MESSAGE---……----END ENCRYPTED MESSAGE-----», // Données chiffrées

}

L’encryptedData représente les données brutes signées, en utilisant une clé privée par l’émetteur. Le destinataire doit disposer de la clé publique pour déchiffrer le message.

Cette approche est applicable lorsque la taille de message est modérée, ce qui permet un temps de déchiffrement raisonnable.

Cependant si les messages à transmettre sont volumineux ce qui augmente le temps de déchiffrement, une deuxième approche plus efficace consiste à calculer un hash des données brutes à l’aide d’une fonction de hachage. Cette empreinte sera signée et envoyée avec les données brutes en clair. Le destinataire calculera le hach des données brutes reçues, déchiffra la signature du hash et il comparera les deux valeurs. Si le hash calculé correspond au hash reçu, les données considérées comme authentiques.

A tester pour notre cas.

***Structure de message à appliquer***

Afin de garantir que les communications entre l’AD et la supervision respectent la norme EN50159 onj va appliquer le type de B1.

La structure de notre message est la suivante :

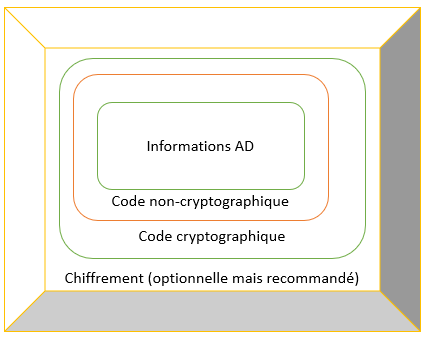


Figure 3 : Structure de Message respectant le type B1

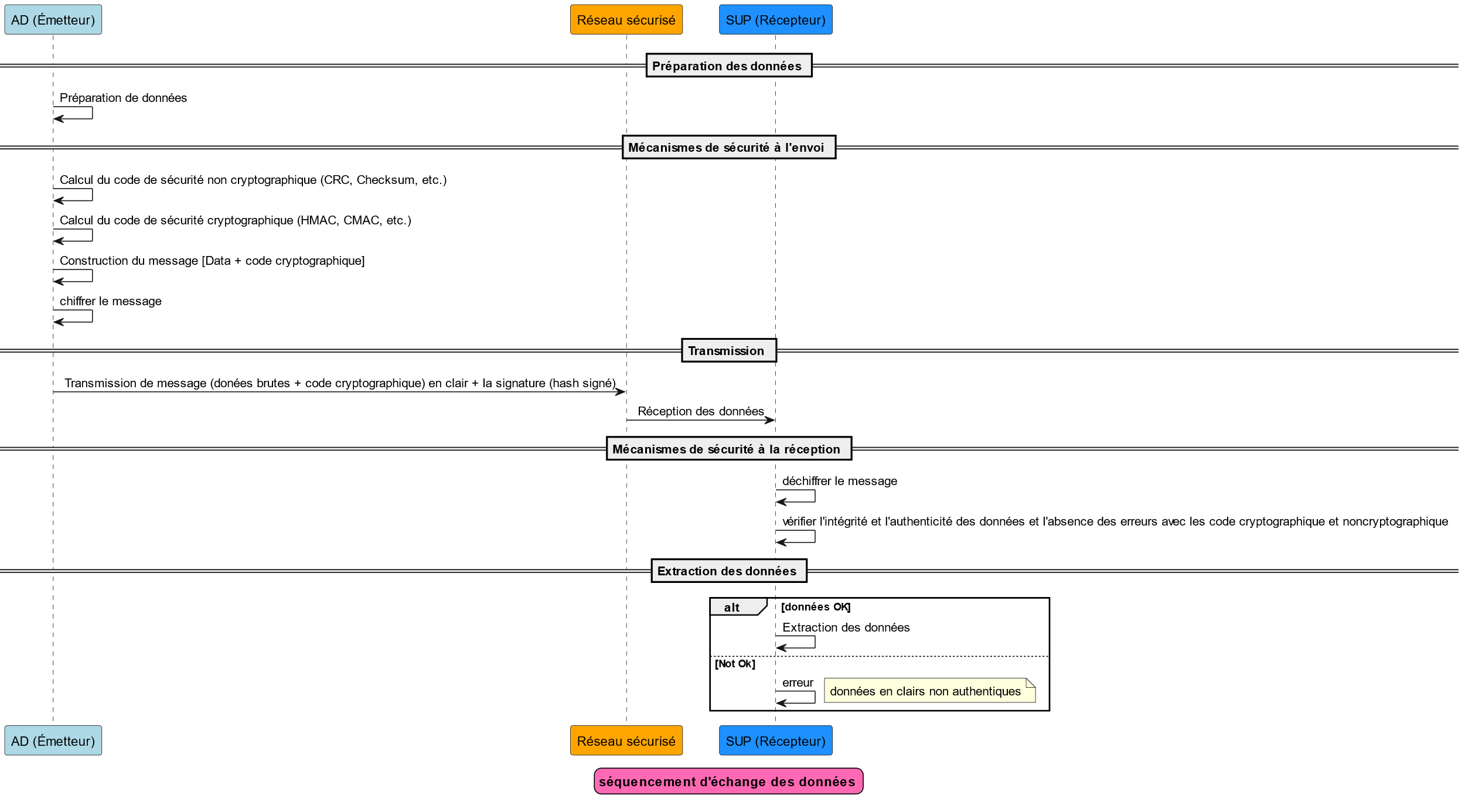
***Structure de message à envoyer :***

"encryptedData": {

[AD + code non-cryptographique + code cryptographique]

},

***Séquencement de transmission***



## Protocoles d’échange entre AD et Supervision :

L’échange entre la supervision et l’AD se fait avec une fréquence déterminée et sous des conditions spécifiques d’événement.

### Messages périodiques

* Le seul message à envoyer d’une façon périodique et avec une fréquence de 10 Hz (un envoi chaque 100 ms), c’est le message VehicleTrackingInfo.

***Messages Ponctuels (non-Périodiques)***

Les autres messages sont des messages échangés entre le véhicule et la supervision sont des messages à envoyer en réaction à des évènements ou à d’autres messages.

### Message ponctuels envoyés par les véhicules :

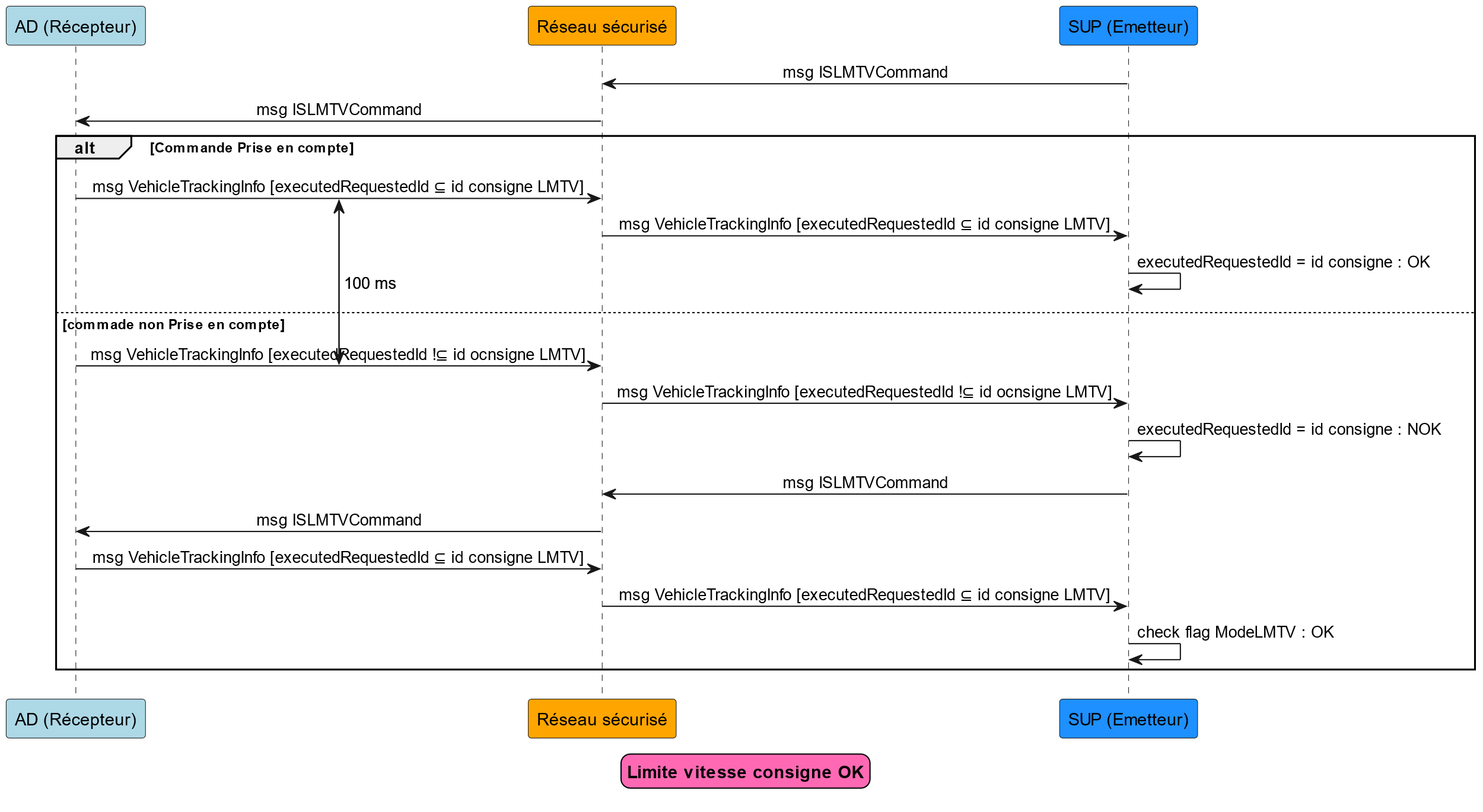
* + checkStatusResponse :envoyé en réponse au message de supevision checkStatus, ce message contient des informations sur l’état du véhicule et sa capacité à accomplir une mission.
  + MissionAssignmentReponse : envoyé par le véhicule en réponse au message de supervision MissionAssignement, ce message contient la confirmation de la prise de mission par le véhicule.
  + ISAuthorizationRequest : envoyé au moment de demande d’autorisation.
    - * Dans le cas d’un problème de sécurité La supervision pourra vérifier si le message est envoyé par le véhicule approprié grâce :
        + Le vehicleId : c’est l’identifiant de véhicule. Cette information permet à la supervision de déterminer la source de message.
        + timeStamp : permet à la supervision l’ordre chronologique de message de d’assurer la cohérence.
        + sequenceNumber : permet à la supervision de contrôler l’ordre spécifique des messages reçus de véhicule et de les traiter dans le bon ordre et de détecter les messages manquants.

### Message ponctuels envoyés par la supervision :

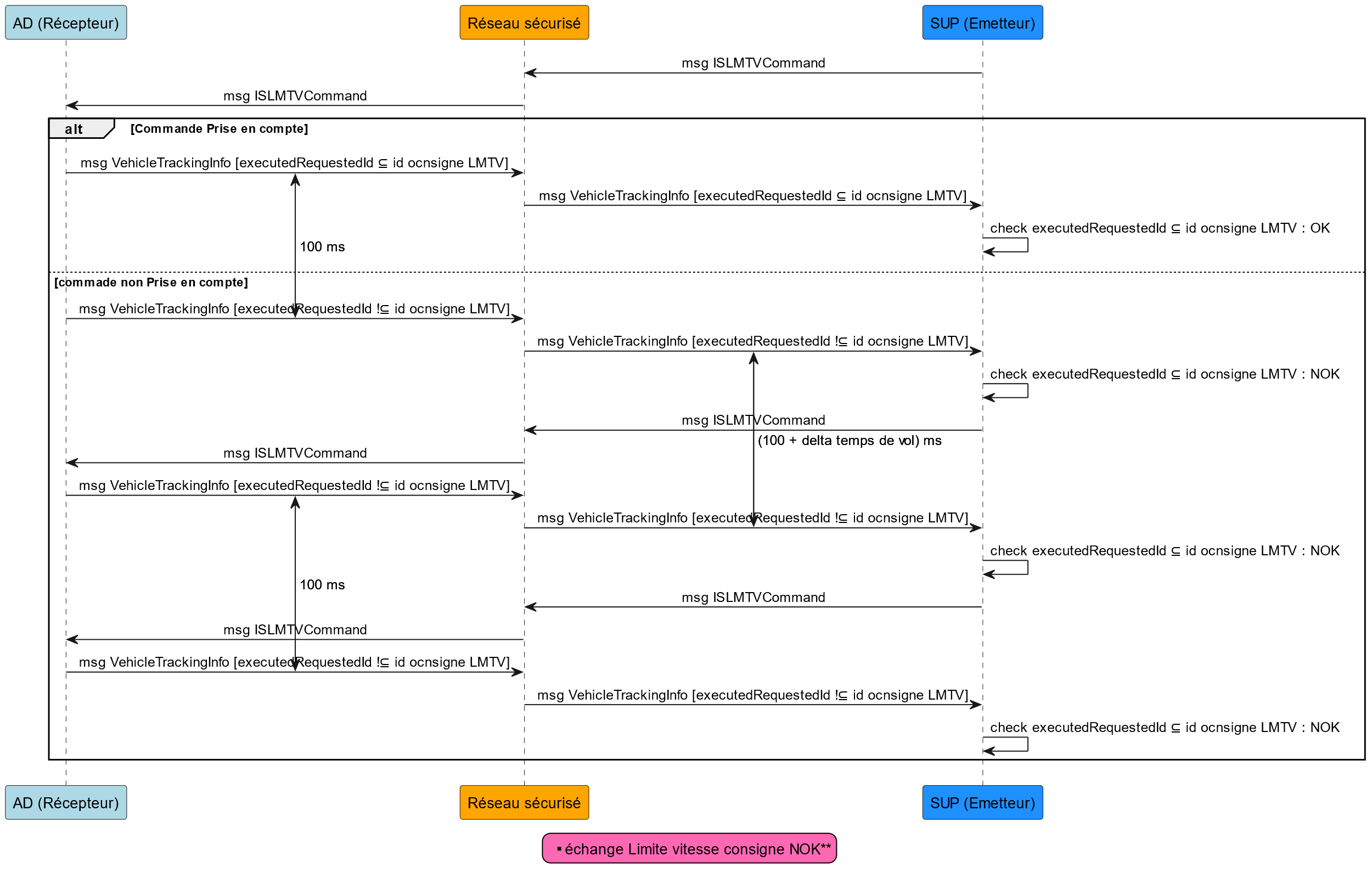
* + Le message CheckStatus est envoyé par la supervision à chaque fois, y a une mission à faire. Ce message est envoyé vers le véhicule pour vérifier son état et sa capacité d’exécuter la mission.
  + Le message MissionAssignment est envoyé pour assigner une mission au véhicule, suite à des premiers échanges.
  + Le message ISAuthorization est envoyé suite à une demande d’autorisation envoyé par le véhicule, via le message ISAuthorizationRequest.
  + Le message ISUpdateMission est envoyé vers le véhicule pour changer le chemin de la mission.
  + Le message ISLMTVCommand est envoyé vers le véhicule pour appliquer une limite de vitesse sur une zone.
  + Le message ISStopCommand est envoyé vers le véhicule pour exécuter un arrêt.

### Messages Ponctuels et séquencement :

*Scénario d’envoi d’une limite de vitesse*

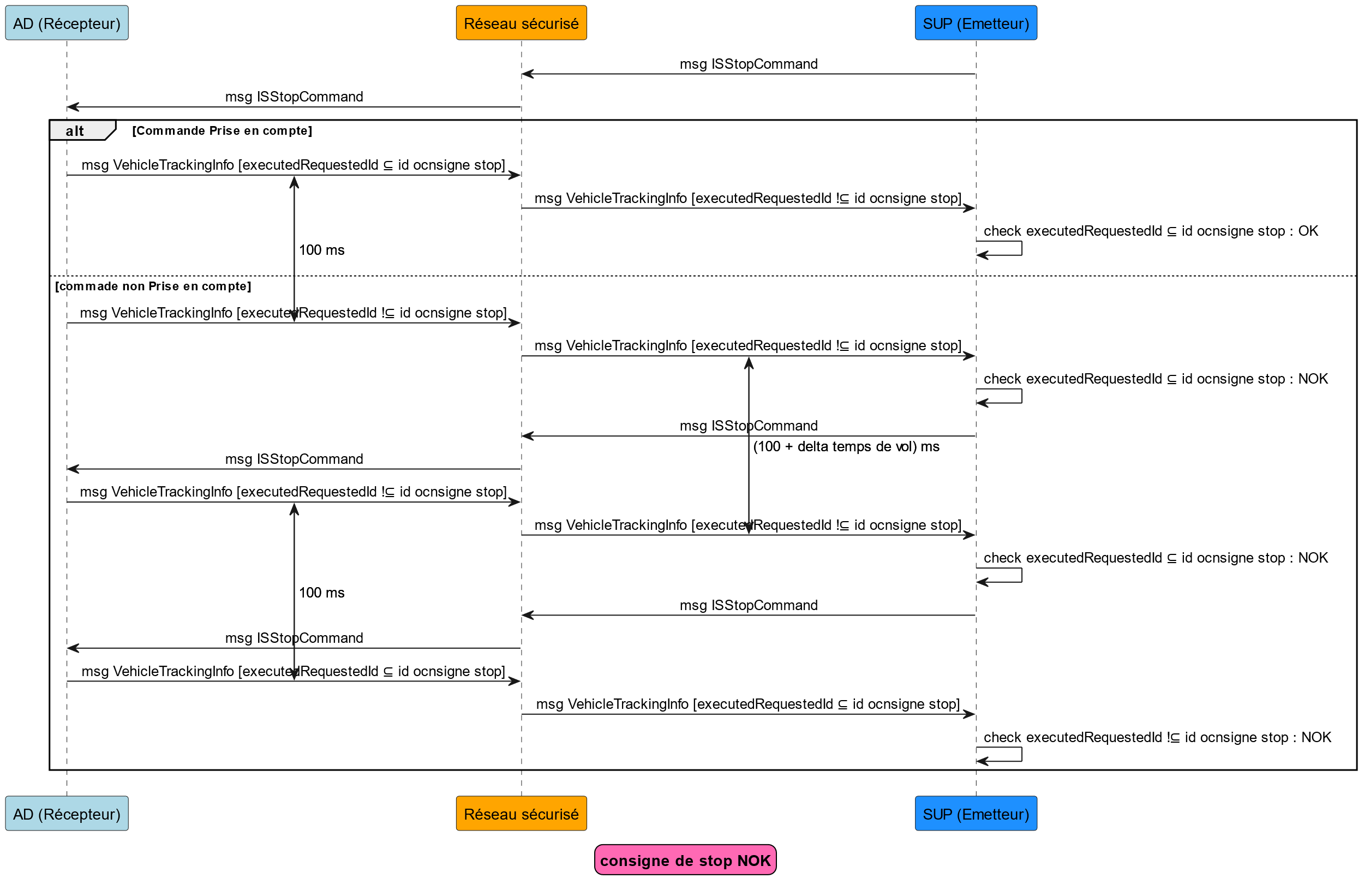


À la suite de l’envoi du message ‘ISLMTVCommand’ par la supervision, le message ‘vehicleTrackingInfo’ sera vérifié. Si le champ ‘executedRequestedId’ contient l’id de la consigne de ‘ISLMTVCommand’, la supervision considère que le véhicule a pris en charge cette consigne de limite de vitesse.



En revanche, si après N tentatives (N à préciser) la consigne LMTCommand n’est pas exécuté par le véhicule la supervision doit lancer le mode dégradé.

*Scénario d’envoi de la consigne stop*



À la suite de l’envoi du message ISStopCommand par la supervision, le message vehicleTrackingInfo sera vérifié. Si le champ executedRequestedId contient l’id de consigne stop, la supervision considère que le véhicule a pris en charge cette consigne stop.

En revanche, la supervision enchaine à envoyer la consigne stop vers le véhicule.

## Tests à réaliser

Un ensemble des tests sera mis en place, pour tester les performances des échanges avec les tailles réelle des messages ainsi que pour tester le niveau de sécurité (à réfléchir).