**GESTION DU SAV FTTH**

**Présentation technique**

**du protocole FTTH v3.0**

Version 0.4

|  |  |  |  |
| --- | --- | --- | --- |
| SUIVI DES VERSIONS | | | |
| Version | Date | Nom du rédacteur | Nature de la modification |
| 0.0 | 24/06/2019 | C.Bertin | Création du draft |
| 0.1 | 01/10/2019 | M.Amraoui | Mise à jour de la matrice des codes retours |
| 0.2 | 21/10/2019 | GT SAV | Validation suite à relecture |
| 0.3 | 28/01/2021 | APNF | Correction incohérences et rajout sur sytème d’authentification (chapitre 3.3) |
| 0.4 | 30/08/2021 | C. Gerard | Correction coquille dans le numéro de code erreur page 17 (Exemple d’exception Fault) |
|  |  |  |  |
|  |  |  |  |

Sommaire

**1.** Contexte et objectifs 3

* 1.1. Objet du document 3
* 1.2. Lexique 4

**2.** Protocole SAV FTTH v3.0 5

* 2.1. Généralités 5
* 2.2. Cycle de vie du ticket 7
* 2.3. Règles des échanges 8
* 2.4. Principe des échanges 8

**3.** Protocole 9

* 3.1. Explication des onglets du protocole 9
* 3.2. Sécurité des échanges 10
* 3.3. Système d’authentification 10
* 3.4. Liste des Webméthodes présentées par l’OI 10
* 3.5. Liste des Webméthodes présentées par l’OC 11

**4.** Description méthode « getTroubleTicketByKey » 12

* 4.1. Partie Requête 12
* 4.2. Partie réponse 12

**5.** Description Codes Erreur 15

* 5.1. Erreur fonctionnelle 15
* 5.2. Erreur technique 17

**6.** Annexe 18

* 6.1. WSDL et XSD 18
* 6.2. Exemple 18
* 6.3. Modalité des tests entre l’OI et l’OC 18

# Contexte et objectifs

Après deux versions du protocole inter opérateurs SAV FTTH, dont l’objectif est de disposer de flux d’échange simplifiés et normalisés pour l’activité SAV, le Groupe de Travail inter-opérateurs Processus & SI SAV FTTH a défini en 2019 un nouveau protocole normalisé pour assurer l’interopérabilité et l’industrialisation des Processus et des Systèmes d’Informations du SAV FTTH des opérateurs.

Ce protocole, Machine To Machine, repose sur l’échange de fichiers XML en mode web service. Il a été défini à partir de la norme internationale JSR-91 (Trouble Ticket) du TM Forum.

Cette présente version du protocole sera finalisée, pour intégration chez les différents opérateurs selon les modalités convenues par le Groupe.

## Objet du document

L’objet du document est de présenter le protocole inter-opérateurs SAV FTTH v3.0 d’un point de vue technique et permettre de présenter les grands principes et de décrire les échanges afin de le partager avec l’ensemble des opérateurs dans l’objectif d’harmoniser les pratiques chez l’ensemble des opérateurs d’immeuble et commerciaux.

## Lexique

Lien normalisation ARCEP :

<http://www.arcep.fr/fileadmin/reprise/dossiers/fibre/20120213-cefibre_schemas_glossaire_FttH.pdf>

| **Sigle ou concept** | **Signification** |
| --- | --- |
| Contact Client | Numéro de téléphone de contact du client |
| ZTD | Zone Très Dense. Liste de 148 communes définie par l'ARCEP (décision N°2009-1106). |
| ZMD | Zone Moins Dense |
| Signalisation | Ticket d’incident déposé par l’OC auprès de l’OI |
| XML | Langage informatique qui sert à échanger des données textuelles |
| WSDL | Langage qui permet de définir les services proposés |
| XSD | Langage informatique qui permet de définir et de contrôler un fichier XML |
| WS | Web service |
| TroubleTicket | Ticket d’incident |
| Web service | Un web service est un programme informatique permettant la communication via le web et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. |
| SOAP | Protocole de communication permettant l’échande de données xml |
| WSDL | WSDL est une grammaire XML permettant de décrire un Service Web |
| guichet | Application d’un opérateur comprenant plusieurs web services |
| SU | Service Supplier = OI, Opérateur d’Immeuble |
| SP | Service Provider = OC, Operateur Commercial |

# Protocole SAV FTTH v3.0

Le principe de ce protocole est de permettre l’échange entre l’OC et l’OI sur un ticket d’incident. Chacun utilisant une interface web service. Cette technique permet de valider immédiatement chaque échange et d’éviter des désynchronisations des informations entre les différents systèmes d’information de l’OC et de l’OI.

## Généralités

Le protocole SAV FTTH inter-opérateurs v 3.0 repose le principe d’un échange par web service entre un OI et un OC. L’OI pourra présenter un guichet unique de web service à l’ensemble des OC et de même l’OC pourra présenter un guichet unique de web service à l’ensemble des OI. Le principal intérêt est de pouvoir intégrer un OI ou un OC supplémentaire simplement, sans avoir à recréer des guichets pour chacun.

(voir Convention\_SAV\_FTTH\_v3.0@Diagramme d'états).

Exemple d’architecture du point de vue d’un OI :

**Webservice SAV de l’OI**

Comprenant les méthodes :

* **createTroubleTicketByValue**
* **setTroubleTicketByValue**
* **getTroubleTicketByKey**
* getSupportedOptionalOperations
* getManagedEntityTypes
* getUpdateProcedureTypes
* getTroubleTicketTypes
* getTroubleTicketItemTypes
* getNamedQueryTypes
* getEventDescriptor
* getEventTypes

**OC2**

**OC1**

Exemple d’architecture du point de vue d’un OC :

**Webservice SAV de l’OC**

Comprenant les méthodes :

* **setTroubleTicketByValue**
* getSupportedOptionalOperations
* getManagedEntityTypes
* getUpdateProcedureTypes
* getTroubleTicketTypes
* getTroubleTicketItemTypes
* getNamedQueryTypes
* getEventDescriptor
* getEventTypes

**OI2**

**OI1**

## Cycle de vie du ticket

Le cycle de vie d’un ticket d’incident commence à la création du ticket avec l’appel de l’OC à l’OI via la méthode **createTroubleTicketByValue**. Si celui-ci est acquitté, le ticket est créé et son cycle de vie commence. La suite des échanges entre les deux opérateurs s’effectue par la méthode **setTroubleTicketByValue** qui permet le changement d’état du ticket (champ « troubleTicketState ») ou des échanges de notification d’information.

Le cycle de vie du ticket se termine lors de l’échange de l’OC vers l’OI par la méthode setTroubleTicketByValue avec le changement d’état définitif CLOSED. Une fois le ticket passé dans cet état, plus aucun changement d’état n’est possible sur ce ticket. Seules certaines modifications d’information peuvent être rediffusées par l’OI post-clôture du ticket, mais l’état du ticket ne changera pas.

La liste des statuts d’un ticket est décrit dans le protocole ainsi que le tableau de changement d’état (voir Convention\_SAV\_FTTH\_v3.0@Diagramme d'états).



## Règles des échanges

Les règles des échanges sont définies par le diagramme de changement d’état. Si le changement n’est pas autorisé, un opérateur peut refuser ce changement.

L’envoi par l’OI ou l’OC d’un setTroubleTicketByValue peut recevoir un refus définitif pour les raisons suivantes :

* Violation des règles de changement d’état
* Violation des règles de gestion des actions ou des règles de gestion des rendez-vous
* Absence de paramètre obligatoire au changement
* Tout autre raison (ticket fermé, ticket inexistant, ligne hors du parc de l’OC, ligne inconnu de l’OI)

L’envoi par l’OI ou l’OC d’un setTroubleTicketByValue peut recevoir un refus provisoire pour les raisons suivantes :

* Maintenance ou indisponibilité momentanée du système

Dans ce cas, la charge est à l’émetteur qu’il soit OI ou OC, de réessayer ultérieurement. Cela impose à l’OI et à l’OC

* Soit de gérer les émissions de manière asynchrone et de stocker la requête pour pouvoir l’émettre ultérieurement en cas d’indisponibilité du web service
* Soit de retraiter le ticket manuellement ultérieurement.

## Principe des échanges

Cette notion asynchrone peut générer un conflit entre l’OC et l’OI. C’est pourquoi, un paramètre de gestion lastUpdateVersionNumber a été intégré dans l’échange. Celui-ci s’incrémente à chaque requête (quelle que soit la requête de l’OI) effectuée par l’OI. Dès qu’un traitement est effectué par l’OI, ce paramètre s’incrémente d’une valeur égale à 1, avant de l’émettre à l’OC. L’OC lorsqu’il effectue une requête doit reprendre la dernière valeur, connue par lui, du paramètre lastUpdateVersionNumber. L’OI n’accepte la transaction que si cette valeur est identique à sa valeur actuelle.

Cela permet d’éviter à l’OI d’accepter des demandes obsolètes du fait que l’OC n’a pas encore intégré les dernières requêtes de l’OI. L’OC peut savoir immédiatement lors d’un échange qu’il n’a pas pris en compte les derniers changements de l’OI soit à cause d’une indisponibilité de son interface, soit par un décalage de traitement.

# Protocole

## Explication des onglets du protocole

|  |  |
| --- | --- |
| Sommaire | Sommaire et description des onglets |
| Nomenclature | Références techniques et nomenclature utilisée dans les protocoles |
| Modèle de données TT | Définition du modèle de données permettant les échanges  le modèle TroubleTicket est utilisé dans   * createTroubleTicketByValueRequest * setTroubleTicketByValueRequest * getTroubleTicketByKeyResponse   le modèle troubleTicketResponse est utilisé pour   * createTroubleTicketByValueResponse * setTroubleTicketByValueResponse   les autres modèles ont le nom de leurs utilisations   * getTroubleTicketByKeyRequest |
| Diagramme d’états | Diagramme d’état et règle de changement de troubleTicketState |
| Valeurs possibles | Liste des valeurs possibles et/ou autorisées. La liste de valeur est liée à un modèle de donnée. L’entête de la colonne définit le nom du champ |
| Mapping actions OC | Liste des actions possibles de l’OC en précisant les valeurs optionnelles/obligatoires des modèles de données (TroubleTicket…) |
| Mapping actions OI | Liste des actions possibles de l’OI en précisant les valeurs optionnelles/obligatoires des modèles de données (TroubleTicket…) |
| Operations Méta | Liste des opérations de type méta permettant au guichet de décrire la liste des opérations disponibles et les versions.. |
| Mapping Action Lecture Ticket | Description de la méthode getTroubleTicketByKey |

## Sécurité des échanges

Les données échangées doivent être sécurisées (cryptées) et ne doivent pas être lisibles de tous. Une gestion d’authentification doit être mise en place par chaque OI et OC.

Il appartient à chaque OI et OC de définir et de mettre en œuvre leur politique de sécurité « commune » dans le cadre de leurs relations contractuelles.

## Système d’authentification

Pour chaque appel à un web service, l’opérateur émetteur doit s’authentifier à travers un couple login/mot de passe à fournir par l’opérateur destinataire et à renseigner au sein d’un en-tête du message SOAP :

<soapenv:Envelope

<soapenv:Header>

<v1:**TroubleTicketHeaderRequest**>

<v1:LoginName>?</v1:**LoginName**>

<v1:**Password**>?</v1:**Password**>

</v1:TroubleTicketHeaderRequest>

</soapenv:Header>

<soapenv:Body>

...

</soapenv:Body>

</soapenv:Envelope>

## Liste des Webméthodes présentées par l’OI

|  |  |  |
| --- | --- | --- |
| **Type d'opération** | **Méthode** | **Remarque** |
| Action | createTroubleTicketByValue | Permet de créer le ticket d'incident |
| Action | setTroubleTicketByValue | Permet de modifier le ticket d'incident |
| Action | getTroubleTicketByKey | Récupère les informations du ticket d'incident |
| Introspection | getSupportedOptionalOperations | Fournit liste des opérations optionnelles disponibles. |
| Introspection | getManagedEntityTypes | Fournit liste des objets disponibles |
| Introspection | getUpdateProcedureTypes | Fournit la liste des méthodes de modification du ticket :   * setTroubleTicketByValue |
| Introspection | getTroubleTicketTypes | Fournit la liste des objets TroubleTicket disponibles, un seul actuellement :   * troubleTicket |
| Introspection | getTroubleTicketItemTypes | La réponse sera vide par convention |
| Introspection | getNamedQueryTypes | La réponse sera vide par convention |
| Introspection | getEventDescriptor | La réponse sera vide par convention |
| Introspection | getEventTypes | La réponse sera vide par convention |

## Liste des Webméthodes présentées par l’OC

|  |  |  |
| --- | --- | --- |
| **Type d'opération** | **Méthode** | **Remarque** |
| Action | setTroubleTicketByValue | Permet de modifier le ticket d'incident |
| Introspection | getSupportedOptionalOperations | Fournit liste des opérations optionnelles disponibles. |
| Introspection | getManagedEntityTypes | Fournit liste des objets disponibles |
| Introspection | getUpdateProcedureTypes | Fournit la liste des méthodes de modification du ticket :   * setTroubleTicketByValue |
| Introspection | getTroubleTicketTypes | Fournit la liste des objets TroubleTicket disponibles, un seul actuellement :   * troubleTicket |
| Introspection | getTroubleTicketItemTypes | La réponse sera vide par convention |
| Introspection | getNamedQueryTypes | La réponse sera vide par convention |
| Introspection | getEventDescriptor | La réponse sera vide par convention |
| Introspection | getEventTypes | La réponse sera vide par convention |

\*ces méthodes sont optionnelles, mais elles sont vivement conseillées pour faciliter la gestion par l’OC.

## 

# Description méthode « getTroubleTicketByKey »

Cette méthode, exposé par l’OI, permet la récupération de toutes les informations à tout moment par l’OC d’un ticket.

Les informations d’un ticket sont disponibles par l’OC tant que le ticket est ouvert chez l’OI. En outre, lorsque le ticket est fermé chez l’OI (CLOSED) les informations de ce ticket restent disponibles par l’OC tant que l’accès est détenu par l’OC concerné.

## Partie Requête

Pour déterminer le ticket d’incident, deux paramètres sont possibles :

Soit le troubleTicketKey

Les autres paramètres permettent de définir l’OI et l’OC lors de l’échange.

Les paramètres composant la requête :

|  |  |  |
| --- | --- | --- |
| Paramètres | Description | Présence |
| troubleTicketKey | Référence du Ticket créé dans le System de l'OI | oui |
| serviceProviderID | SIREN de l'OC | vide |
| technicalId | Référence de la prise FTTH (PTO) | vide |
| supplierID | SIREN de l'OI | vide |

## Partie réponse

La partie réponse effectuée par l’OI doit comprendre les dernières informations de l’OI.

La partie réponse se décompose en trois parties :

Les champs du protocole permettant de savoir l’état du ticket.

Les champs décrivant le ticket d’incident émis par l’OC, ces champs n’évoluent pas lors de l’avancement du ticket.

Les champs d’information sur le ticket d’incident remplis par l’OI ou l’OC au fur et à mesure de l’avancement du ticket.

La partie protocole

|  |  |
| --- | --- |
| Paramètres | Description |
| troubleTicketKey | Identifiant du ticket |
| lastUpdateVersionNumber | Dernier numéro de mise à jour de l’OI |
| interactionDateComplete | Date de cloture du ticket dans le SI de l’OI |
| messageType | Dernière action venant de l’OI ou de l’OC enregistré par l’OI |
| troubleTicketState | Dernier état du ticket d’incident |

La partie description de l’incident

|  |  |
| --- | --- |
| Paramètres | Description |
| troubleDescription | Information déposée par l’OC lors de la création du ticket |
| troubleDetectionTime | Information déposée par l’OC lors de la création du ticket |
| serviceProviderID | Information déposée par l’OC lors de la création du ticket |
| serviceProviderName | Information déposée par l’OC lors de la création du ticket |
| serviceProviderUnit | Information déposée par l’OC lors de la création du ticket |
| serviceProviderPhoneNumber | Information déposée par l’OC lors de la création du ticket |
| supplierID | Information déposée par l’OC lors de la création du ticket |
| troubleTicketType | Information déposée par l’OC lors de la création du ticket |
| offerId | Information déposée par l’OC lors de la création du ticket |
| commercialId | Information déposée par l’OC lors de la création du ticket |
| technicalId | Information déposée par l’OC lors de la création du ticket |
| serviceProviderTroubleTicketKey | Information déposée par l’OC lors de la création du ticket |
| defectClassId | Information déposée par l’OC lors de la création du ticket |
| defectDescription | Information déposée par l’OC lors de la création du ticket |
| defectPrelocalization | Information déposée par l’OC lors de la création du ticket |
| alreadyOperated | Information déposée par l’OC lors de la création du ticket |

La partie des informations échangée entre l’OC et l’OI

|  |  |
| --- | --- |
| Paramètres | Description |
| serviceRestoredTime | Information présente lorsque le ticket est clos |
| supplierName | Dernière Information présente renseignée |
| supplierUnit | Dernière Information présente renseigné |
| supplierPhoneNumber | Dernière Information présente renseignée |
| customerLastName | Dernière Information présente renseignée |
| customerFirstName | Dernière Information présente renseignée |
| contactPhoneNumber1 | Dernière Information présente renseignée |
| contactPhoneNumber2 | Dernière Information présente renseignée |
| customerAccessibilityHours | Dernière Information présente renseignée |
| appointmentMode | Dernière Information présente renseignée |
| SUAppointmentManagementId | Dernière Information présente renseignée |
| messageContent | Dernière Information présente renseignée |
| supplierPlannedActionDate | Dernière Information présente renseignée |
| supplierResolutionAction | Dernière Information présente renseignée |
| supplierResolutionState | Dernière Information présente renseignée |
| troubleTicketClosureCode | Information présente lorsque le ticket est clos |
| troubleTicketClosureLabel | Information présente lorsque le ticket est clos |
| defectLocalization | Information présente lorsque le ticket est clos par l’OI |
| defectResponsibility | Information présente lorsque le ticket est clos |
| reprovisionningId | Dernière Information présente renseignée |
| troubleTicketClosureComment | Information présente lorsque le ticket est clos par l’OI |
| closureDuration | Information présente lorsque le ticket est clos par l’OI |
| linkLengthPMPRDM | Dernière Information présente renseignée |
| delivryPointPRDM | Dernière Information présente renseignée |
| delivryPointPM | Dernière Information présente renseignée |
| offerLinkValue | Dernière Information présente renseignée |
| offerOptionalId | Dernière Information présente renseignée |

# Description Codes Erreur

Les erreurs renvoyées par la méthode peuvent être de deux types : fonctionnelles ou technique.

## Erreur fonctionnelle

Le format de ce type d’erreur est décrit dans le document « convention » dans la partie

FTTHMutTicketResponse.

Le champ « troubleTicketKey » est obligatoire lors de la réponse à la méthode CreateTroubleTicketByValue si la valeur du returnCode est 0.

Dans les cas de l’erreur 1 ou 2, le champ, dont la valeur du champs provoque l’erreur, doît être indiqué dans le returnLabel.

La liste des valeurs est disponible dans le document Convention\_SAV\_FTTH\_3.0@valeurs possibles

Champ composant une erreur fonctionnelle

|  |  |
| --- | --- |
| Paramètres | Description |
| troubleTicketKey | Identifiant du ticket dans le SI de l’OI s’il a été crée |
| returnCode | Code retourné |
| returnLabel | Description de l’erreur |

Pour la méthode CreateTroubleTicketByValue

<?xml version="1.0" encoding="UTF-8" ?>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Envelope xmlns:soapenv="**http://schemas.xmlsoap.org/soap/envelope/**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Body>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:CreateTroubleTicketByValueResponse xmlns:ns7="**http://ossj.org/xml/TroubleTicket/v1-2**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:troubleTicketResponse>

  <ns5:troubleTicketKey xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**" xsi:nil="**1**" />

  <ns5:ReturnCode xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**">8</ns5:ReturnCode>

  <ns5:ReturnLabel xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**"> PTO INCONNUE</ns5:ReturnLabel>

  </ns7:troubleTicketResponse>

  </ns7:CreateTroubleTicketByValueResponse>

  </soapenv:Body>

  </soapenv:Envelope>

Pour la méthode SetTroubleTicketByValue

<?xml version="1.0" encoding="UTF-8" ?>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Envelope xmlns:soapenv="**http://schemas.xmlsoap.org/soap/envelope/**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Body>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:CreateTroubleTicketByValueResponse xmlns:ns7="**http://ossj.org/xml/TroubleTicket/v1-2**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:troubleTicketResponse>

  <ns5:troubleTicketKey xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**" xsi:nil="**1**" />

 <ns5:ReturnCode xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**">**14**</ns5:ReturnCode>

<ns5:ReturnLabel xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**">**TROUBLETICKETKEY INCONNU**</ns5:ReturnLabel>

  </ns7:troubleTicketResponse>

  </ns7:CreateTroubleTicketByValueResponse>

  </soapenv:Body>

  </soapenv:Envelope>

Pour la méthode getTroubleTicketByKey

<?xml version="1.0" encoding="UTF-8" ?>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Envelope xmlns:soapenv="**http://schemas.xmlsoap.org/soap/envelope/**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <soapenv:Body>

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:getTroubleTicketByKeyResponse xmlns:ns7="**http://ossj.org/xml/TroubleTicket/v1-2**">

[**-**](file:///C:\Users\CBAUDOUI\AppData\Local\Temp\createTroubleTicketByValue_reponse_KO.xml) <ns7:troubleTicketResponse>

  <ns5:troubleTicketKey xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**" xsi:nil="**1**" />

 <ns5:ReturnCode xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**">**14**</ns5:ReturnCode>

<ns5:ReturnLabel xmlns:ns5="**http://ossj.org/xml/TroubleTicket-CBETrouble/v1-2**">**TROUBLETICKETKEY INCONNU**</ns5:ReturnLabel>

  </ns7:troubleTicketResponse>

  </ns7: getTroubleTicketByKeyResponse >

  </soapenv:Body>

  </soapenv:Envelope>

## Erreur technique

Cette erreur intervient lorsque lorsqu’une erreur technique intervient. L’erreur est sous la forme d’une exception soap. La liste des soap fault sont décrites dans le WSDL/XSD.

Ce type d’erreur indique une erreur fatale liée au format xml.

Sous la forme d’une exception soap :

|  |  |
| --- | --- |
| Borne XML | Description |
| soapenv:Fault | Borne XML |
| faultcode | soapenv:Server |
| faultstring | Nom de l’exception  Ex :createTroubleTicketByValueException |
| detail | Borne XML |
| Nom exception | Borne XML |
| Type exception | Borne XML |
| message | La valeur contient « erreurCode => signification », voir liste des valeurs possibles |

Exemple d’exception Fault :

<soapenv:Fault>

<faultcode>soapenv:Server</faultcode>

<faultstring>CreateTroubleTicketByValueException</faultstring>

<detail>

<ns6:createTroubleTicketByValueException xmlns:ns6="http://ossj.org/xml/TroubleTicket/v1-2">

**<ns6:**createException xmlns:ns1="http://ossj.org/xml/Common/v1-5" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:RemoteException">

<ns1:message> 15 => Accès non autorisé </ns1:message>

</ns6:createException>

</ns6:createTroubleTicketByValueException>

</detail>

</soapenv:Fault>

# Annexe

## WSDL et XSD

La wsdl et la XSD sont présentées sur GIT

## Exemple

Les exemples xml des flux envoyés sont diffusés dans un document séparé

## Modalité des tests entre l’OI et l’OC

Des tests de cette version de protocole peuvent être initiés sur l’outil de tests géré par l’APNF.