Service Public
Fédéral
FINANCES

AM 29/04/2024

SYSTÈME DE CAISSE ENREGISTREUSE - RÈGLEMENTATION TECHNIQUE -

DESCRIPTIONS DÉTAILLÉES DU FONCTIONNEMENT ET DE LA COMMUNICATION ENTRE LE SYSTÈME DE CAISSE ET LE FISCAL DATA MODULE

VERSION 1.1 – 18/07/2024

CHANGELOG

Version	Date	Résumé
1.0	26/06/2024	Texte original
1.1	18/07/2024	**<u>Corrections d'erreurs</u>**: - Input SaleInput: fdmRefs (Chap. 2, 2.3.4) - lineTotal: remarque composite prodct – formule a envoyer vers FDM (Chap. 2, 2.2.2) - type MessageItem: message (Chap. 2, 2.3.2) - input ReportUserXInput: reportBookingDate deux fois (Chap. 2, 2.3.4) - input InvoiceInput: insérer costCenter (Chap. 2, 2.3.4) - whitelist remplacer par allowlist (Chap. 2, 2.2.4 en 2.3.3.) - enum Code – mise à jour (Chap. 2, 2.2.4.) **<u>Clarifications</u>** - Connexion physique POS – FDM (Chap. 2, 1.) - Object fdmRefs – (Chap. 2, 2.2.2) - lineTotal – (Chap. 2, 2.2.2)

INTRODUCTION

Les descriptions détaillées reflètent de manière plus technique et détaillée les dispositions techniques de l'arrêté ministériel (AM) du 29/04/2024 relatif aux aspects techniques en ce qui concerne la certification d'un système de caisse enregistreuse (SCE).

Ce document est la description technique détaillée concernant le fonctionnement du système de caisse enregistreuse et sa communication avec le Fiscal Data Module.

Pour le fonctionnement du Fiscal Data Module, veuillez-vous référer à la description détaillée correspondante.

Ce document peut faire l'objet de modifications mineures, en raison d'éventuelles décisions réglementaires, afin de corriger d'éventuelles erreurs.

Des informations complémentaires peuvent être obtenues auprès du service compétent, CNR division SCE via secr.gksce@minfin.fed.be.

CHAPITRE 1 – EXIGENCES TECHNIQUES POUR LE SYSTÈME DE CAISSE

Référence AM : Titre II, Chapitre 1

1. ENREGISTREMENT DES EVENTS

Référence AM : Section1, sous-section 1,7 et 8

1.1. VENTE DIRECTE

L'event N est immédiatement enregistré.

La transaction (y compris le message GraphQL) contient toujours les lignes de produits enregistrées dans l'ordre chronologique. Les changements de prix sont liés à ces lignes de produits. Les corrections du nombre de produits sont toujours des lignes séparées, qui prennent leur place dans la chronologie.

L'impression finale du ticket de caisse TVA peut être consolidée et la chronologie peut également être modifiée (ex : pour un regroupement par département). Le ticket de caisse TVA peut également contenir plus d'informations que celles requises par la loi (par exemple Gtin, prix unitaire, etc.).

L'imprimé contient toujours la mention TICKET DE CAISSE TVA.

Exemples dans les Use Cases (UC).

1.2. ENREGISTREMENT DE VENTE AVEC REMBOURSEMENT COMPLET POUR UN TICKET DE CAISSE TVA DÉJÀ DELIVRÉ

Si un event N précédemment créé est ENTIEREMENT repris dans un nouvel event N, **la référence** du ticket de caisse TVA original sera mentionnée dans le message GraphQL ainsi que sur l'impression du ticket de caisse TVA. Pour cela, on utilise l'array 'fdmRefs'. Ce nouvel event N peut contenir des lignes de produits supplémentaires. Par event N, un seul event N précédemment établi peut être repris.

Si le ticket de caisse TVA contient uniquement une reprise complète d'un event N précédent (et contient donc uniquement des lignes de produits négatives), l'impression de ce ticket portera également la mention 'REFUND' (article 5, 3° de AM).

Exemples dans les UC.

1.3. VENTE INTERROMPUE

Dans ce type de vente, l'enregistrement des entrées est interrompu dans le temps et un ou plusieurs enregistrements partiels (réservations) sont effectués. Ex : la gestion des tables, la gestion des clients, le transfert vers le compte de l'hôtel, commandes par Internet, commandes par kiosque, la mise en attente d'un enregistrement (on-hold).

L'event P est utilisé à cette fin. Après (un ou plusieurs de) ces events P, l'enregistrement complet des ventes doit toujours donner lieu à un event final N pour l'ensemble de ces events P (à l'exception des transferts vers le compte de l'hôtel).

Une ‘reference est utilisée pour conserver toutes les entrées d'une transaction de vente.

Chaque réservation individuelle suit les mêmes règles pour le message GraphQL que pour les ventes directes (corrections, impressions, etc.).

Chaque impression d'une telle entrée contient toujours la mention PRO FORMA et, en bas, la mention CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE.

Exemples dans les UC.

1.4. APERÇU D’ADDITION (ADDITION PROVISOIRE)

Cet event P peut être créé lors d'une fonctionnalité 'ventes interrompues', où un aperçu des commandes enregistrées et/ou du montant à payer est affiché, avant que le paiement ne soit enregistré, l'event 'normal' est créé et le ticket de caisse TVA est livré.

Cela inclut également l'appel et l'affichage de l'état d'une table sur une tablette, un smartphone ou tout autre appareil connecté ou faisant partie du système de caisse enregistreuse.

Conformément à l'art. 34 du MB, les lignes de produits peuvent être consolidées aussi bien sur l'imprimé que dans le message GraphQL au FDM de cet extrait de compte. Ajouter de nouvelles lignes de produits ou apporter des modifications aux lignes de produits existantes n'est pas autorisé lors de la création de l’aperçu d’addition.

Un imprimé indique toujours PRO FORMA et FACTURE PROVISOIRE. En bas, il y aura également la mention CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE.

Exemples dans les UC.

1.5. ENREGISTREMENTS EN MODE TRAINING

Toutes les inscriptions créées lorsque l'ensemble du système POS est en mode formation ou par un utilisateur en mode formation sont transmises au FDM en tant qu'event de formation. Ils portent donc toujours l'étiquette ‘T’.

Un imprimé d'un tel event mentionne toujours le nom TRAINING. En bas, il y aura également la mention CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE.

Exemples dans les UC.

1.6. ENREGISTREMENT DES TRANSACTIONS FINANCIÈRES PURES

Exemples dans les UC.

1.7. ÉTABLISSEMENT DE FACTURES BASÉES SUR DES EVENTS N PRÉCÉDEMMENT ÉTABLIS

Si un système de caisse permet l’établissement de facture, quel que soit son format, sur la base d'un ou plusieurs tickets de caisse TVA émis précédemment, cette facture sera incluse dans l'event ‘INVOICE’. Les lignes de produits de ces tickets de caisse TVA sont incluses dans l'impression de cet event (mais **pas** dans le message GraphQL envoyé au FDM).

Les lignes de paiement des tickets de caisse TVA originaux **ne sont pas** copiées.

L'impression porte toujours la mention FACTURE et en bas se trouve également la mention CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE.

Exemples dans les UC.

1.8. COPIES D'EVENTS PRÉCÉDEMMENT ÉTABLIS

Si vous souhaitez imprimer une **copie** d'un ticket de caisse TVA établi précédemment (event “NORMAL”) (par exemple après une panne d'imprimante ou un ticket devenu illisible), un event de copie doit être établi à cet effet. Une référence au ticket de caisse TVA original est toujours fournie.

Cet event peut également être utilisé pour des copies d'events ‘P’, ‘I’, ‘S’, ‘F’ ou ‘R’ créés précédemment.

Toutes les impressions contiennent toujours les mots COPIE et en bas se trouve également la mention CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE.

1.9. ENREGISTREMENT DE LA PRÉSENCE DU PERSONNEL

Chaque système de caisse enregistreuse doit fournir cette fonctionnalité ; l'utilisation effective n'est pas obligatoire, sauf si l'employeur opte pour certains avantages de l’ONSS ou autres.

L'enregistrement de la présence (début/fin) des ‘utilisateurs’ peut être transmis au FDM via l'event ‘SOCIAL’. N'importe quelle routine d'inscription peut être utilisée à cette fin, à condition que la caisse enregistreuse crée les données correctes pour l'event et les envoie au FDM via le message GraphQL. Cet event **ne sera pas** imprimé et **ne constitue pas** une transaction de vente.

2. NUMEROTATION DES EVENTS

Art. 13. Nécessite une numérotation consécutive pour chaque event. Cela signifie un des éléments suivants :

- une numérotation consécutive distincte par type d'event (label) ;
- une numérotation générale continue des events ;
- une numérotation générale continue sur l'ensemble de l'event par terminal ;
- une numérotation continue distincte par type d'event (label) par terminal.

3. CONDITIONS D'ENREGISTREMENT DES EVENTS

1) Un utilisateur qui souhaite enregistrer des events doit d'abord se connecter au système de caisse enregistreuse. Aucune opération ne peut être effectuée sur le système de caisse sans qu'un utilisateur soit connecté. Une procédure d'inscription distincte peut être prévue pour l'enregistrement d'events de type ‘S’ (Social).

Les utilisateurs du système de caisse, quelle que soit leur fonction au sein de l'entreprise, doivent être clairement identifiables par leur numéro NISS, qui, selon les cas, est constitué du numéro de registre national ou du numéro BIS. Ce numéro est stocké dans le logiciel serveur ou dans la base de données ‘utilisateurs’ du système de caisse. Le numéro NISS se compose de 11 caractères numériques.

Un utilisateur du système de caisse, étranger à l'entreprise (par exemple un technicien), qui enregistre des actions à l'aide du système de caisse, est toujours identifié dans le système de caisse avec le numéro ‘00000000097’.

Pour les commandes en ligne et en kiosque et toutes les options de commande futures qui ne nécessitent pas l'intervention d'une personne physique, mais relèvent de l'entière responsabilité de l'opérateur, le numéro ‘00000000029’ doit être utilisé comme ‘utilisateur du robot’.

Un aperçu de ces paramètres de programme ou du tableau de la base de données doit pouvoir être mis à la disposition du responsable du contrôle de manière simple, sur demande.

2) Le système de caisse enregistreuse ne peut enregistrer les livraisons de biens et/ou de services que lorsque le FDM est connecté et pleinement opérationnel. Une transaction dont l'enregistrement a commencé ne peut être terminée qu'après réception d'une signature du FDM.

4. CONSERVATION DES DONNEES

Référence AM : Section 1, sous-section 2

Le contribuable utilisateur d'un système de caisse enregistreuse est responsable du stockage des données créées par le système de caisse, conformément à la législation TVA (et par extension à la législation comptable). Par exemple, le contribuable utilisateur est principalement responsable du stockage des données sur le système de caisse et sur le FDM. Les sauvegardes des données de caisse font partie du système de caisse et sont donc soumises aux mêmes durées de conservation.

Les données créées par la caisse elle-même (base de données, messages JSON, les données du GraphQL) et reçues du FDM (JSON) sont stockées sous leur forme originale sur le système de caisse, conformément à la durée de conservation fiscale en vigueur. Il est rappelé que, même si aucun format spécifique n'est prescrit, toutes les données créées par le système de caisse doivent pouvoir être présentées sous une forme lisible et compréhensible conformément à l'article 61, §1 du code TVA. Pour faciliter la copie de ces données, au moins un port (d'un type courant) de la caisse enregistreuse doit être accessible/activé pour un support de données externe.

Lors de la demande de certification, le fabricant du système de caisse enregistreuse fournit une description détaillée de la base de données et, plus particulièrement, tous les tableaux contenant des données d'events ou une description complète de toute autre méthode de stockage des données mentionnée dans l'arrêté ministériel.

Il est expressément rappelé aux fabricants de systèmes de caisse qu'ils sont responsables de la manière dont sont sécurisées les données de la base de données relatives aux events susmentionnés. Dans sa demande de certification, le fabricant indique les mécanismes de sécurité utilisés (qu'ils soient intégrés au logiciel lui-même ou fournis par des tiers).

5. CODES TVA

Référence AM : Section 1, sous-section 3

Le système de caisse enregistreuse doit fournir les codes TVA suivants

CODE TVA	DESCRIPTION DU TAUX TVA	TAUX DE TVA
A	Haut	21 %
B	Moyen	12 %
C	Bas	6 %
D	Taux nul	0 %
X	Hors du champ d'application TVA	Aucun

Le logiciel de caisse peut calculer lui-même les montants imposables et la TVA due. Cependant, ce résultat **n'est pas** transmis au FDM.

Le firmware du FDM calcule lui-même le montant imposable et la TVA due, sur la base des lignes de produits transmises avec le code TVA associé. Les tarifs pris en compte par le FDM sont tenus à jour grâce à la connexion en ligne avec les serveurs du SPF Finances. En cas de changement de taux, aucune action manuelle n'est requise sur le FDM.

Le ticket de caisse TVA indique les montants imposables et les montants de TVA reçus du FDM. Ceci s'applique également à la création des rapports.

Utilisation du code TVA X : pour l'enregistrement de la 'vente' de biens et de services qui échappent totalement au champ d'application de la TVA.

Cela signifie : l'imputation et la reprise des vidanges, l'imputation et le remboursement des cautions, la vente d'un bon d'achat Multi-Purpose-Voucher (MPV).

Remarque importante : articles avec plusieurs codes TVA.

Dans des cas exceptionnels, des articles peuvent avoir deux codes TVA ou plus. Des exemples typiques incluent : un magazine vendu avec un CD (hors restauration) ou un menu tout compris (restauration).

Les structures de données utilisées permettent d'envoyer correctement de tels articles au FDM en tant que produit unique avec plusieurs codes TVA ; chaque code TVA est alors associé à la partie du prix de vente (TVA incluse) à laquelle il se rapporte, complété par les éventuelles modifications de prix applicables à cette partie du prix de vente.

Ces articles peuvent être imprimés comme une seule ligne de produits sur le ticket de caisse TVA. De préférence avec un code TVA combiné (par exemple AB). Si cela n'est techniquement pas possible, le code TVA de la partie ayant la plus grande valeur peut être imprimé, à condition que la répartition réelle soit envoyée au FDM.

Il est bien entendu possible d'opter pour une ligne de produits distincte par code TVA, avec la valeur correspondante.

Attention : si un menu est disponible avec un forfait de boissons dont la valeur est affichée séparément, il faut toujours opter pour deux lignes de produit distinctes (à la fois lors de l'impression et lors de l'envoi à la FDM).

6. LE TICKET DE CAISSE TVA

Référence AM : Section 1, sous-section 4

Voir aussi plus haut dans ce document, inscription de l'event N.

6.1. IDENTIFICATION DU SYSTÈME DE CAISSE

Il s'agit du numéro de série attribué selon l'art. 46 prévu dans l'Arrêté ministériel. Ce numéro est affiché dans le message GraphQL adressé au FDM dans le champ posId.

6.2. IDENTIFICATION DU TERMINAL

Si un SCE comprend plusieurs caisses enregistreuses sous le même posId, un identifiant de terminal unique doit être utilisé pour chaque caisse enregistreuse. Des exemples sont un nom logique (bar, terrasse, …) ou toute autre identification (numéro, ...). Ceci est affiché dans le message GraphQL envoyé au FDM dans le champ terminalId.

6.3. IDENTIFICATION DU DISPOSITIF DE SAISIE (INPUT DEVICE)

Un périphérique d'entrée est le périphérique physique sur lequel l'event est enregistré. Il peut s'agir d'un système de caisse autonome, d'un terminal, d'un kiosque, d'un handheld ou d'un smartphone, ...

Pour les webshops situés dans le cloud ou proposées par un tiers, le nom de ce tiers ou l'URL du webshop est utilisé.

Dans le message GraphQL entre le système de caisse et FDM, l'identification est incluse dans le champ deviceId

6.4. IDENTIFICATION DE L'UTILISATEUR

Le numéro NISS de l'utilisateur n'est **JAMAIS** imprimé sur le ticket de caisse TVA. L'utilisateur doit pouvoir être lié, en interne dans le système de caisse, via l'identification indiquée sur le ticket, à ce numéro NISS, qui est envoyé au FDM dans le message GraphQL.

6.5. QR CODE

Celui-ci est généré par le logiciel de caisse lui-même sur base de l'URL reçue du FDM.

L'URL se compose de 38 caractères alphanumériques, ceux-ci s'insèrent dans un module 25x25 de type 2 QR avec un medium error correction feature (ECC).

Ce code QR est imprimé sur le ticket de caisse papier TVA. En cas d'impression numérique, l'URL est indiquée sous forme de lien sur le ticket de caisse TVA.

6.6. GÉNÉRAL

Le ticket de caisse TVA peut contenir plus d'informations que ce qui est légalement déterminé.

7. ARRONDISSEMENT DES PAIEMENTS

Référence AM : Section 1, sous-section 9

Les nouveaux articles VI.7/1 et VI.7/2 du Code de droit économique obligent le commerçant à arrondir le montant total du paiement en espèces par le consommateur au multiple inférieur ou supérieur de 5 centimes.

Concrètement, cela signifie que les versements se terminant par 1, 2, 6 ou 7 centimes sont arrondis au multiple inférieur de 5 centimes, tandis que les versements se terminant par 3, 4, 8 ou 9 centimes sont arrondis au multiple supérieur de 5 centimes. Les paiements inférieurs à 5 centimes ne sont jamais arrondis.

Si le commerçant souhaite effectuer tous les paiements, quel que soit le mode de paiement, SCE doit proposer cette option. Compte tenu du caractère actuellement facultatif de l'arrondi pour les paiements autres qu'en espèces, le SCE n'applique pas l'arrondi pour les paiements autres qu'en espèces si l'opérateur ne choisit pas de le faire.

Il est précisé que ce choix ne s'effectue pas par action, mais pour tous les paiements autres qu'en espèces. L'arrondi est obligatoire pour les paiements en espèces.

Attention : ce qui précède ne s'applique JAMAIS aux chèques-repas, aux écochèques et aux bons de valeur, qui ont toujours une valeur fixe. Lors du paiement avec des chèques-repas, des écochèques et des bons de valeur, ceux-ci doivent toujours être traités en premier.

Pour les types de lignes de paiement à utiliser, veuillez-vous référer au chapitre 2, plus loin dans ce document.

Remarque importante : Un ticket de caisse TVA peut comporter un arrondi maximal de 2 centimes d'euro au total (en plus ou en moins).

Si le SCE est utilisé dans un environnement HoReCa, il peut arriver que des groupes plus importants effectuent des paiements par personne ou par groupe de personnes. Selon l'organisation de l'entreprise et/ou le système de caisse utilisé, la 'vente directe par consommateur', la 'division de table' ou la 'division du paiement' sera utilisée.

En 'vente directe', le montant exact consommé 'par consommateur', est enregistré dans la caisse et selon le mode de paiement (et le choix du commerçant), arrondi ou non. Un ticket de caisse TVA est délivré pour chaque consommateur. En 'division de table', un ticket de caisse avec TVA est également délivré à chaque consommateur (soit avec le montant exact de ce qu'il a consommé, soit avec sa part proportionnelle). L'arrondi des paiements est identique à celui de la 'vente directe'.

Cependant, avec la 'division du paiement' un seul ticket de caisse avec TVA est émis pour l'ensemble des consommateurs. Dans ce cas, l'exploitant doit travailler méthodiquement pour répartir correctement les paiements (ceci ne s'applique d'ailleurs pas uniquement au secteur de la restauration et au SCE).

Compte tenu de la remarque importante mentionnée ci-dessus, il convient de procéder comme suit :

7.1. L'EXPLOITANT ARRONDI UNIQUEMENT LES PAIEMENTS EN ESPÈCES

Méthodologie :

1. Les paiements par chèques-repas et bons de valeur (globaux ou séparés, selon choix) sont enregistrés. Ceux-ci apparaîtront alors sous forme de ligne de paiement sur le ticket. La caisse enregistreuse calcule le solde.
2. Ensuite, les paiements en espèces sont traités. Ceux-ci apparaissent comme une seule ligne de paiement sur le ticket, ARRONDIE. Le système de caisse recalcule le solde.
3. En dernier lieu, les paiements électroniques (NON ARRONDIS) sont traités pour régler le solde. Il est possible, si souhaité, d'utiliser plusieurs lignes de paiement par type de paiement.

Le total des arrondis ne peut être supérieur à 2 centimes d'euro (en moins ou en plus).

7.2. L'EXPLOITANT ARRONDI TOUS LES MOYENS DE PAIEMENT

Méthodologie :

1. Les paiements par chèques-repas et bons de valeur (globaux ou séparés, selon choix) sont enregistrés. Ceux-ci apparaîtront alors sous forme d'une ligne de paiement sur le ticket. Le système de caisse calcule le solde.
2. LA CAISSE ARRONDI CE SOLDE.
3. Ensuite, les paiements en espèces sont traités. Ceux-ci apparaissent comme une seule ligne de paiement sur le ticket, ARRONDIE. Le système de caisse recalcule le solde.
4. En dernier lieu, les paiements électroniques sont traités pour régler le solde, qui sera forcément un montant ARRONDI. Il est possible, si souhaité, d'utiliser plusieurs lignes de paiement par type de paiement.

Le total des arrondis ne peut être supérieur à 2 centimes d'euro (en moins ou en plus).

Le SPF Finances met à la disposition des développeurs de systèmes de caisse intéressés des exemples de situations concrètes via le site Internet www.systemedecaisseenregistreuse.be.

8. REGLES DE CONSOLIDATION ET D'IMPRESSION

Référence AM : Section 1, sous-sections 7 et 8

8.1. IMPRESSIONS

Pour l'application de l'arrêté, une distinction claire doit toujours être maintenue entre les notions suivantes : EVENT <> IMPRESSION (ticket) <> MESSAGE au FDM.

CHAQUE event donne TOUJOURS lieu à un message vers le FDM.

Seuls les events NORMAL (N) et INVOICE (I) donnent TOUJOURS lieu à une impression, sur papier, numériquement ou des deux manières. En effet, le ticket de caisse TVA doit être délivré au client final conformément à l'article 21bis de l'arrêté royal n°1. Il en va de même pour la facture émise conformément à l'article 53 §2 de la loi TVA.

L'event REPORT (R) entraîne TOUJOURS la création d'un fichier, MAIS celui-ci doit également être imprimable facilement par l'opérateur, soit sur papier, soit sous forme numérique.

Tous les autres events PEUVENT (mais ce n'est pas obligatoire) être imprimés. Ces impressions contiennent toujours, de manière clairement lisible en bas, la mention 'CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE'.

Un aperçu de compte porte également de manière claire et lisible la mention ‘ADDITION PROVISOIRE’.

De plus, les impressions des events TRAINING (T) et PRO FORMA (P), outre la mention ‘CECI N'EST PAS UN TICKET DE CAISSE TVA VALABLE’, en haut et clairement lisible, contiennent respectivement la mention ‘TRAINING TICKET’ et ‘PRO FORMA TICKET’. Pour l'impression de l'event COPY (C), la mention ‘COPY TICKET’ est également indiquée en plus de ce qui précède.

Les tickets de cuisine et de bar ne mentionnent JAMAIS de montants. S'ils sont imprimés suite à l’établissement d'un event PRO FORMA, ils doivent contenir cette mention.

8.2. CONSOLIDATION

Afin d'optimiser la lisibilité des tickets, des consolidations sont autorisées dans certains cas, y compris toutes les corrections (lignes de produits négatives ou positives) et les modifications de prix (augmentation ou diminution, associées à la ligne de produit correspondante). Le logiciel de caisse ne doit pas créer d’event 'P' distinct à cet effet.

REMARQUE IMPORTANTE : Ces consolidations ne sont jamais appliquées aux messages que le système POS envoie au FDM. Concrètement, cela signifie qu'un ticket correspond substantiellement au message, mais qu'il peut y avoir des différences formelles.

La consolidation n'est pas obligatoire, chaque constructeur détermine lui-même si son logiciel de caisse consolide ou non. Si la consolidation est utilisée, les règles ci-dessous doivent être respectées.

Ventes directes (event 'N') :

Le message au FDM contient toutes les données d'entrée, par ordre chronologique, y compris les corrections (lignes de produits négatives ou positives) et les changements de prix (plus ou moins liés à la bonne ligne de produits).

Exemple:

1	Cola	2.50	A
1	Eau	3.00	A
2	Spaghetti	20.00	B
-1	Cola	-2.50	A
1	Eau	3.00	A

Lorsqu'elles sont imprimées sur le ticket de TVA, les corrections peuvent être consolidées et même la commande peut être ajustée selon les paramètres du logiciel de caisse (par exemple regroupement par départements d'articles ou par taux de TVA).

Dans l'exemple précédent, cela donnerait :

2	Eau	6.00	A
2	Spaghetti	20.00	B

Vente interrompue (gestion des tables)

Les mêmes règles s’appliquent comme celles décrites ci-dessus. Concrètement, l'event P (qui contient la commande **originale**) sera transmis chronologiquement et incluant les corrections. L'imprimé associé (facture provisoire ou ticket cuisine/bar) peut être consolidé et regroupé.

A la clôture de l'event N, le message et l'impression du ticket de caisse final TVA peuvent être consolidés.

CHAPITRE 2 – INTERACTION POS – FDM

Lorsque le FDM est connecté au système de caisse enregistreuse :

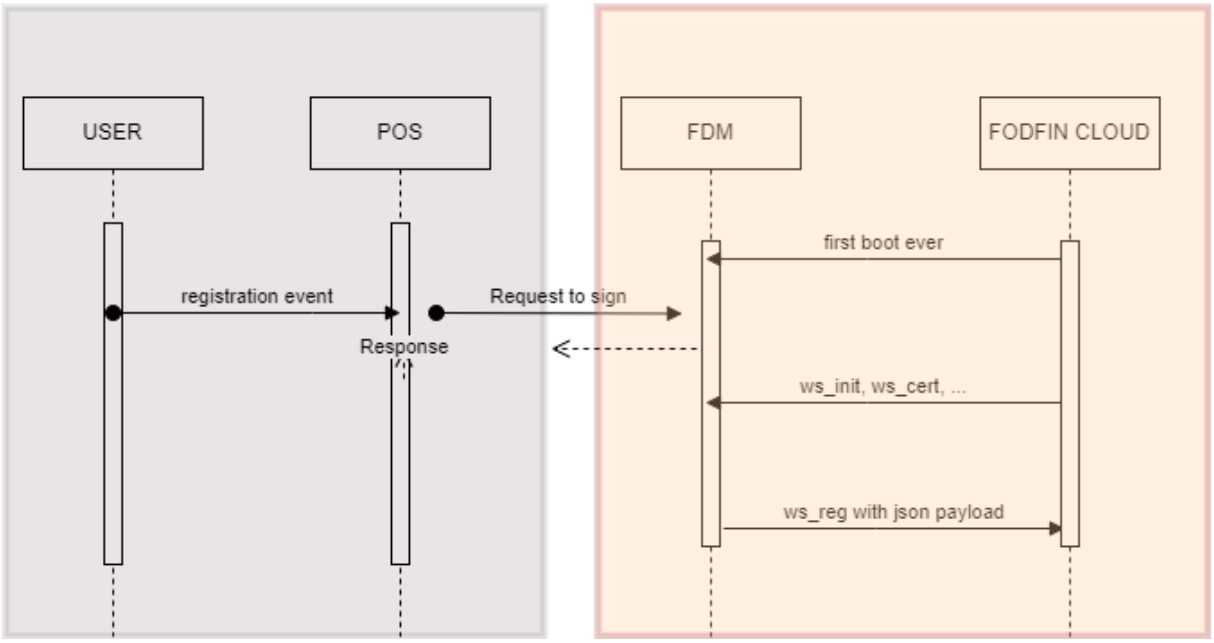
- le FDM reçoit les données d'event spécifiques du système de caisse ;
- le FDM met à jour ses compteurs concernés ;
- le FDM crée les données de contrôle ;
- le FDM stocke ces messages et réponses ;
- le FDM renvoie les données de contrôle au système de caisse ;
- le système de caisse peut imprimer le ticket avec les données de contrôle reçues, à condition qu'il reçoive une telle réponse du FDM.

Tous les systèmes de caisse enregistreuse, visés par l'arrêté ministériel, doivent communiquer ces données en utilisant le protocole de communication et selon les formats de données, tels que décrits dans le présent document dans le cadre de l'arrêté ministériel susvisé.

DEFINITIONS ET ABREVIATIONS

API	Application Programming Interface
EFT	Electronic Funds Transfer
ENUM	Énumération
FDM	Fiscal Data Module
GRAPHQL	Graph Query Language
JSON	JavaScript Object Notation
LAN	Local Area Network
POS	Point Of Sale
RTC	Real Time Clock
SCE	Système de Caisse Enregistreuse
TCP/IP	Transmission Control Protocol/Internet Protocol
TRANSACTIONS	tous les events pour lesquels des données sont enregistrées
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format
WiFi	Wireless Fidelity

Le flux de traitement des données peut être brièvement résumé comme suit :



Chronologique :

1. Le FDM démarre pour la première fois et reçoit les paramètres nécessaires du cloud SPFFIN.
2. L'utilisateur enregistre les events dans le système de caisse.
3. La caisse transmet immédiatement ces events au FDM sous forme de messages JSON, qui effectue un certain nombre de calculs en fonction du type d'event et appose sa signature sur le JSON enrichi.
4. Le FDM renvoie la réponse avec les données de contrôle à la caisse enregistreuse.
5. Le FDM stocke dans sa mémoire tampon le JSON enrichi des events.
6. Le FDM contacte régulièrement le service web SPFFIN pour mettre à jour ses paramètres ou certificats.
7. Le FDM transmet les messages JSON mis en mémoire tampon au cloud SPFFIN à intervalles réguliers.
8. Les messages JSON reçus sont stockés dans le cloud SPFFIN dans un environnement sécurisé, où ils sont disponibles pour analyse.
9. Le système de caisse affiche les messages du FDM et/ou du cloud SPFFIN via son interface utilisateur.

1. CONNEXION PHYSIQUE

Le système de caisse et le FDM sont connectés par câble (LAN) ou sans fil (WiFi). Ceci est configuré dans le back-office de la caisse et du FDM.

IMPORTANT : En théorie, la connexion sans fil peut également se faire via Internet. En cas de perte de connexion Internet, la connexion système de caisse - FDM sera automatiquement perdue et le système de caisse cessera d'enregistrer.

Le fabricant du FDM doit fournir suffisamment d'options pour permettre une sécurité adéquate de la connexion entre la caisse et le FDM. Des exemples typiques sont : HTTP avec token, HTTPS avec TLS avec certificat auto-sginé, HTTPS avec mTLS utilisant un certificat auto-signé basé sur le certificat intermédiaire du fabricant de FDM.

La responsabilité ultime de l'installation et de la configuration correctes et sécurisées incombe au distributeur/installateur.

Le système de caisse et le FDM utilisent le protocole TCP/IP pour échanger leurs messages. Ces messages sont décrits plus en détail ci-dessous. D'autres messages ne sont autorisés que dans la mesure où ils n'affectent pas les fonctionnalités obligatoires et contribuent au bon fonctionnement de l'ensemble. Le FDM met son service GraphQL à disposition du POS via HTTP(S).

Le FDM est connecté à l'API cloud SPFFIN via Internet et reçoit également régulièrement des instructions via cette voie. Dans certains cas, voir la description détaillée correspondante, ces instructions peuvent entraîner l'affichage d'un message sur le système de caisse et/ou une intervention manuelle via le système de caisse.

2. COMMUNICATION ENTRE POS ET FDM

Les requêtes sont envoyées par la caisse au service GraphQL du FDM. Les données JSON sont utilisées comme payload pour envoyer les données d'event. Ce payload est envoyée au chemin d'accès '/graphql' sur le FDM avec le verbe HTTP 'POST' et Content-Type : application/json.

Tous les objets JSON sont conformes à la norme RFC 8259.

Les champs de type chaîne ne contiennent **jamais** d'espaces blancs en début ou en fin de chaîne.

Les données envoyées comprennent

- des données de transaction (events P et N);
- des données financières (event F);
- des données sociales (event S);
- les copies (event C);
- les events de formation (event T);
- les factures (event I);
- les rapports (event R).

Ces objets et leurs conditions sont décrits plus en détail dans le schéma GraphQL complet.

2.1. GÉNÉRAL

Les events suivants ont été déterminés dans l'Arrêté ministériel :

- NORMAL (label N);
- PRO FORMA (label P);
- TRAINING (label T);
- COPY (label C);
- FINANCIAL (label F);
- SOCIAL (label S);
- INVOICE (label I);
- REPORT (label R).

Ces events sont tous envoyés au FDM, afin d'être signés numériquement, comme protection contre la manipulation des données. À cette fin, diverses mutations sont mises à disposition par le service GraphQL du FDM. Ces mutations peuvent être utilisées pour remplir correctement la structure JSON des enregistrements sur le FDM comme décrit ci-dessous. La structure JSON du FDM peut contenir tout type d'event ; via les différentes mutations, le FDM n'accepte pour chaque type d'event que les champs et les objets autorisés.

Les mutations suivantes sont définies.

Pour l'event S:

- signWorkIn
- signWorkOut

Pour l'event I:

- signInvoice

Pour l'event N:

- signSale

Pour l'event P:

- signCostCenterChange
- signOrder
- signPreBill

Pour l'event F:

- signMoneyInOut
- signDrawerOpen
- signPaymentCorrection

Pour l'event R:

- signReportTurnoverX
- signReportTurnoverZ
- signReportUserX
- signReportUserZ

Pour l'event C:

- signCopy

2.2. CONTENU DE LA COMMUNICATION POS ⇗ FDM

Le payload du FDM vers le SPFFIN (voir la description détaillée du FDM) est entièrement basée sur la structure et le schéma de validation associé décrits ci-dessous.

Par conséquent, les données d'event des mutations doivent également respecter cette structure et ces règles de validation. La section 2.3. décrit la conversion vers le schéma GraphQL.

2.2.1. POS ➔ FDM – aperçu complet

{

language (scalar value)

ticketMedium (scalar value)

posId (scalar value)

posFiscalTicketNo (value)

posSwVersion (scalar value)

terminalId (scalar value)

deviceId (scalar value)

posDateTime (scalar value)

bookingPeriodId (scalar value)

bookingDate (scalar value)

vatNo (scalar value)

estNo (scalar value)

employeeId (scalar value)

customerVatNo (scalar value)

invoiceNo (scalar value)

fdmRefs (array de l'objet fdmRef)

costCenter (objet)

transfer (objet)

transaction (objet)

drawer (objet)

financials (array de paymentLine ou de l'objet moneyInOutLine)

reportNo (value)

reportBookingDate (scalar value)

posDevices (array de l'objet posDevice)

fdmDevices (array de l'objet fdmDevice)

turnover (objet)

users (array de l'objet user)

}

2.2.2. POS ➔ FDM – descriptions détaillées

Langue

language (enum, obligatoire)

Langue dans laquelle le système de caisse souhaite recevoir les messages du FDM, à choisir parmi les valeurs ci-dessous :

language	Description
EN	Anglais
NL	Néerlandais
FR	Français
DE	Allemand

Méthode de délivrance

ticketMedium (enum, obligatoire)

Sorte de ticket, à choisir parmi les valeurs ci-dessous :

ticketMedium	Description
NONE	Pas d'impression
PAPER	Impression sur papier
DIGITAL	Délivrance numérique
PAPER_DIGITAL	Impression sur papier et de manière numérique

Données d'identification du POS

posId (string, obligatoire, 14 caractères en majuscules)

Le numéro de série du POS, tel qu'enregistré dans l'e-service SCE. Ex : CFOD0061234567

posFiscalTicketNo (numérique, obligatoire, valeur entre 1 et 999999999)

Numéro de ticket interne du POS. Ce numéro unique peut être généré pour tous les events, générale pour tous les terminaux, par type d'event ou par terminal, tant que la combinaison de ce numéro avec le type d'event ou l'ID du terminal est unique. Lorsque la valeur maximale est atteinte, le POS recommence avec le numéro "1".

posSwVersion (string, obligatoire, longueur min. 1 et max. 36 caractères) Version actuelle du logiciel de caisse. Exemple : 1.8.3

terminalId (string, obligatoire, longueur min. 1 et max. 600 caractères)

Si une installation SCE comprend plusieurs caisses sous le même posId, un terminalId unique doit être utilisé pour chaque caisse. Il peut s'agir par exemple d'un nom logique (bar, terrasse, ...) ou d'un autre identifiant (numéro, ...). Le concept de terminal est totalement indépendant de tout composant matériel.

Si le paramétrage du GKS ne l'utilise pas, la même valeur doit toujours être saisie dans ce champ. Exemple : “1”, “n/a”, GUID fixe.

deviceId (string, obligatoire, longueur min. 1 et max. 600 caractères)

L'identification matérielle de l'appareil sur lequel la transaction a été enregistrée.

Date et heure du POS

posDateTime (string, obligatoire)

Date et heure du système de caisse, en heure locale (Brussel) suivant le format ISO 8601 comme ci-dessous.

Exemple: 2022-10-20T15:01:25+02:00 (heure d’été), 2022-11-03T15:01:25+01:00 (heure d’hiver)

bookingPeriodId (string, obligatoire, format = GUID avec des tirets et des minuscules)

Période de fonctionnement de la caisse : il s'agit généralement d'un jour d'ouverture, ce qui signifie que cette période coïncide avec la période d'un rapport journalier fiscal. Peut également être un shift comme partie de la période d'ouverture. Dans ce dernier cas, un rapport fiscal quotidien contiendra les données de plusieurs périodes de réservation.

Exemple: dffcd829-a0e5-41ca-a0ae-9eb887f95637

bookingDate (string, obligatoire)

Date comptable à laquelle la transaction est comptabilisée, au format date ISO 8601 comme ci-dessous. Cette valeur fait référence à la date à laquelle le bookingPeriodId fait référence. Par exemple, une période de réservation de 00h30 à 04h30 du matin peut en effet être incluse dans le chiffre d'affaires journalier de la veille (date).

Exemple: 2023-10-20

Données d’identification de l’Entreprise et des Utilisateurs

vatNo (string, obligatoire, format: BE suivi de 10 chiffres, longueur totale = 12)

Numéro de TVA de l’entreprise. La partie numérique commence toujours par un '0' ou un '1'. Le FDM vérifiera la validité de ce numéro (en appliquant le modulo 97).

Exemple: BE0499999960

estNo (string, obligatoire, longueur = 10)

Numéro d'unité d'établissement de l'exploitation. Un numéro d'unité d'établissement commence toujours par un numéro qui va de ‘2’ à ‘8. Le FDM vérifiera la validité de ce numéro (en appliquant le modulo 97).

Exemple: 8789456149

employeeId (string, obligatoire, longueur = 11)

Numéro d'immatriculation à la sécurité sociale (NISS), composé soit du numéro de registre national, soit du numéro bis de l'exploitant ou du membre du personnel. Le FDM effectuera un contrôle de la validité de ce numéro (en appliquant le modulo 97).

Exemple: 75061189731

Références

customerVatNo (string, conditionnel, longueur min. 1 et max. 600 caractères)

Pour l'event 'I' (facture), l'utilisation de ce champ est obligatoire et le numéro de TVA du co-contractant y est indiqué. Le format décrit dans VatNo est utilisé pour ce numéro ou l'équivalent étranger si le client n'est pas une entreprise belge.

invoiceNo (string, conditionnel, longueur min. 1 en max. 600 caractères)

Numéro de facture, à utiliser seulement pour l'event de type 'I'.

fdmRefs (optionnel, array de l'objet fdmRef, voir fdmRef)

Cet array contient les références attribuées par le FDM aux events précédents auxquels il est fait référence. Cela peut être un ticket de caisse TVA (event N) qui est **entièrement** annulé, un ou plusieurs tickets de caisse TVA faisant partie d'une facture (event I), le ticket de caisse TVA dont le mode de paiement est corrigé (event F) ou la création d'une copie d'un event précédent (event C).

L'objet fdmRef est décrit dans chapitre 4 - RESPONSE

Enregistrement des transactions – utilisation des possibilités de gestion

Les enregistrements intermédiaires des transactions de ventes peuvent être attribués à des tables, des chaises, des clients, des chambres… Ces enregistrements peuvent survenir fréquemment ou changer d'affectation avant de se finaliser par un ticket de caisse TVA. Les objets 'costCenter' et 'transfer' sont utilisés pour les enregistrer correctement.

costCenter

costCenter (objet, conditionnel)

Identifier 'l'endroit' de la transaction. Il est obligatoire de l'utiliser pour des transactions où l'enregistrement est interrompu (transfert de table, gestion de client, …). L'objet est construit de la façon suivante :

{

id (string, obligatoire, longueur min. 1 et max. 600 caractères)

Description du costCenter. Exemple: "T17", "K08"

type (enum, obligatoire)

Valeur à choisir dans la table tb_costCenterType.

tb_costCenterType

type
TABLE
CHAIR
ROOM
CUSTOMER
ON_HOLD
OTHER

Remarque : La valeur **CHAIR** ne peut être utilisée que si la valeur **TABLE** a été utilisée au même niveau ou à un niveau supérieur.

reference (string, obligatoire, longueur min. 1 en max. 600 caractères)

Session d'un costCenter qui conserve ensemble toutes les transactions liées.

Exemple : "e4eb2509-dce9-44db-8322-e445aaeb19c4"

costCenter (objet costCenter, optionnel, max. 2 niveaux de profondeur)

Cette structure imbriquée permet, par exemple, d'effectuer des enregistrements simultanément au niveau de la table et des chaises.

}

Transfer

transfer (objet, conditionnel)

L'objet transfer est utilisé pour déplacer des livraisons de biens/services déjà enregistrés d'un costCenter où elles ont déjà été réservées vers un autre costCenter. Les exemples ici sont le transfert de table, la division de table, la fusion de table, le transfert de chambre, le transfert vers le compte client, …

{

from (array d'un ou plusieurs objets transferItem)

Origine(s) du costCenter(s), *d'où part le transfert.*

to (array d'un ou plusieurs objets transferItem)

Nouveau(x) costCenter(s), *où le transfert arrive.*

}

Les objets transferItem dans from et to peuvent avoir une relation 'one-to-one', 'multiple-to-one' ou 'one-to-multiple'. Une relation 'multiple-to-multiple' dans laquelle des biens ou des services sont déplacés de plusieurs costCenter vers plusieurs costCenter est interdite.

transferItem

{

costCenter (objet, obligatoire, voir costCenter)

Identification du costCenter d'où part le transfert ou vers lequel arrive un transfert.

transaction (objet, obligatoire, voir transaction)

Les biens/services livrés qui sont déplacés du costCenter.

}

Remarque importante

Lors de l'utilisation de la gestion des tables, une table est normalement fermée lorsque tous les enregistrements ont abouti à l'event 'N', avec la création d'un ticket de caisse TVA.

Il existe une exception à cette règle, à savoir lorsque le contenu de la table est transféré à la chambre d'hôtel, où ce contenu est transféré à la facture globale de l'hôtel. Il existe deux façons de clôturer le tableau :

- soit la clôture avec un événement 'N' et le mode de paiement 'ROOM_CREDIT' (dans ce cas, le contenu de la table figure dans le chiffre d'affaires journalier du rapport de chiffre d'affaires),
- soit la clôture avec un événement 'P', enregistrant un transfert d'un costCenter TABLE vers un costCentre ROOM (dans ce cas, le contenu de la table NE figure PAS dans le chiffre d'affaires journalier du système de caisse).

Enregistrement des transactions – lignes de produit

transaction (objet, conditionnel)

Cet objet contient les enregistrements des livraisons de biens et/ou de services dans le système POS.

Cet objet est **obligatoire** pour les eventOperations suivants :

- SALE;
- PRE_BILL;
- ORDER;
- COST_CENTER_CHANGE.

Cet objet est **interdit** pour toutes les autres eventOperations.

L'objet transaction contient toujours un array transactiontLines. Il peut être vide dans le cas d'un ticket zéro.

{

transactionLines (array des objets transactionLine, voir description transactionLine)

Lignes de produits, y compris tous les changements (de prix).

transactionTotal (numérique, deux décimales)

Prix total de l'ensemble de l'event (somme de toutes les lineTotal). Ce montant peut être positif, négatif ou nul (0).

}

transactionLine

Cet objet contient une ligne de transaction dans laquelle maximum un produit est inclus.

{

lineType (enum, obligatoire)

Valeur à sélectionner dans le tableau ci-dessous.

tb_lineType

lineType
SINGLE_PRODUCT
COMPOSITE_PRODUCT

mainProduct (product object, obligatoire)

Contient toutes les informations sur le produit contenu dans la transactionLine. S'il s'agit d'un produit composite (par exemple un menu), l'array des objets subProduct doit toujours être utilisé également.
Voir la description ci-dessous.

subProducts (array des objets product, conditionnel).

Contient toutes les informations relatives aux produits contenus dans le produit composite mentionné dans l'objet mainProduct.
Voir la description ci-dessous.

costCenter (object, optionnel)

Est utilisé si la caisse permet d'affecter des produits à un costCenter jusqu'au niveau le plus bas (par exemple : chair). L'objet costCenter a été décrit précédemment.

Remarque : la valeur CHAIR ne peut être utilisée que si la valeur TABLE a été utilisée au même niveau ou à un niveau supérieur.

lineTotal (numérique, obligatoire)

Prix total de la ligne de transaction, TVA inclus, après application des modifications de prix. Peut être positif, négatif ou nul.
Dans le cas d'un SINGLE_PRODUCT, il n'y a qu'un seul objet mainProduct et la formule est : [quantity * unitPrice] + Somme[priceChangeAmount].

}

Remarque importante : composite_product

L'utilisation d'un composite_product permet de combiner plusieurs produits en un seul nouveau produit, avec ou sans application de remises pour arriver à un prix unique.

Un composite_product peut être soumis à plusieurs taux de TVA. Les directives suivantes doivent être respectées lors de l'utilisation d'un composite_product :

➔ ***Calcul*** de la base d'imposition et du montant de la TVA par le FDM :

La FDM se basera sur les informations fournies par la caisse concernant les sous-produits.

La base sera alors, par TVA et par sous-produit : somme(vatPrice + somme([priceChangeAmount])).

➔ ***Transmission*** du lineTotal au FDM :

Pour le mainProduct, la somme(vatPrice + somme([priceChangeAmount])) de tous les sous-produits concernés est toujours transmise.

➔ ***Impression*** de la ligne d'article du composite_product :

Sur l'impression du ticket de caisse TVA, seule la ligne de produit composite_product apparaît, avec tous les taux de TVA concernés, le prix total étant la somme(vatPrice + somme([priceChangeAmount])) de tous les sous-produits concernés.

product

Fait partie d'une ligne de transaction où il contient les détails de la livraison d'un bien ou d'un service, ou fait partie d'un produit composite.

{

gtin (string, optionnel, longueur min. 8 et max. 20 caractères)

Global Trade Item Number.

Exemple: “0811571013579”

productId (string, obligatoire, longueur min. 1 et max. 600 caractères)

Numéro de production interne du système de caisse.

productName (string, obligatoire, longueur min. 1 et max. 600 caractères)

Nom du produit tel que sauvegardé dans le système de caisse.

departmentId (string, obligatoire, longueur min. 1 et max. 600 caractères)

Id interne du département du système de caisse, auquel le produit concerné est lié.

departmentName (string, obligatoire, longueur min. 1 et max. 600 caractères)

Nom du département auquel le produit concerné est lié.

quantity (numérique, obligatoire, format = multiple de 0.0001)

Nombre d’unités du produit compris dans cet ligne d’article. Le nombre peut être positif, négatif ou nul.

Exemples: 1245.4789, 2.5, -17.23.

negQuantityReason (enum, conditionnel).

Raison pour laquelle une quantité négative est enregistrée, à choisir parmi les valeurs ci-dessous.

negQuantityReason
REFUND
CORRECTION
PRICE_CHANGE
VOUCHER
COST_CENTER_CHANGE
PRODUCT_SUBSTITUTION
VOUCHER
OTHER

- *refund* désigne : un ‘bien’ ou ‘service’ faisant partie d’une transaction de vente précédemment effectuée qui est repris.
- *correction* désigne : tout ajustement d'une quantité dans une transaction de vente quinon encore définitivement clôturée (exemple : annulation de ligne, correction, annulation de ticket).
- *price_change* désigne : une ligne de produit ‘artificielle’ est ajoutée pour enregistrer correctement un changement de prix sur une ligne de produit déjà comptabilisée dans une transaction qui n'a pas encore été finalisée.
  Exemple : la commande initiale contenait 2 bières ; après avoir transmis la commande, une remise est accordée ultérieurement sur 1 de ces 2 bières ; pour cela, 1 bière sera d'abord corrigée et ensuite 1 bière sera à nouveau enregistrée avec une remise.

quantityType (enum, obligatoire)

L'unité dans laquelle la quantité est exprimée, valeur à choisir dans la liste tb_quantityType

quantityType
PIECE
KILOGRAM
METER
LITRE
HOUR

unitPrice (numérique, deux décimales, obligatoire)

Prix normal d'une unité du produit, inclus dans la ligne de produits, avant application des changements de prix. Peut être positif, négatif ou nul (0).

vats (array de l’objet vat, obligatoire).

Pour chaque label TVA liée au produit concerné, un objet vat est ajouté. Cet array peut contenir au maximum 5 objets vat.

Remarque : Pour un composite_product, les arrays vats du mainProduct sera vide. Toutes les valeurs seront en effet incluses dans les subProducts.

L’objet vat est décrit dans vat.

}

vat

Chaque ligne d'une transaction est soumise à au moins un taux de TVA (voir n° 18 de l'arrêté ministériel). À cela peuvent également être associées une ou plusieurs modifications de prix.

{

label (enum, obligatoire)

Le code TVA, tel que prévu au num. 18 de l’arrêté ministériel qui s’applique à l’article concerné.

label	DESCRIPTION DU TAUX TVA	TARIF TVA
A	Haut	21 %
B	Moyen	12 %
C	Bas	6 %
D	Tarif nul	0 %
X	Hors du champ d’application TVA	Aucun

price (numérique, deux décimales, obligatoire)

Partie du prix de vente (TVA incluse) de l'article concerné soumise au taux de TVA susmentionné, avant application des changements de prix. Ce montant peut être positif, négatif ou nul.

priceChanges (array des objets priceChange)

Cet array n'est utilisé que si des changements de prix ont été appliqués à cette partie du prix de vente de l'article concerné au cours de cet event. Cet array peut contenir plusieurs objets si plusieurs changements de prix s'appliquent. Un maximum de 99 objets est autorisé.

L'objet priceChange est décrit dans priceChange.

}

priceChange

Un objet est utilisé pour chaque changement de prix appliqué. Ce changement peut être en + ou en -

{

groupingId (numérique, interdit pour le type scope PRODUCT, obligatoire pour scope LINE et EVENT, valeur entre 1 et 99)

Ce numéro regroupe une modification de prix applicable à plusieurs produits. Il peut s'agir de plusieurs produits dans le scope d'une seule ligne de transaction ou dans le scope de la transaction entière.

id (string, obligatoire, longueur min. 1 et max. 600 caractères).

Id du changement de prix, tel qu'indiqué dans le système de caisse, quelle que soit la langue.

name (string, obligatoire, longueur min. 1 et max. 600 caractères)

Nom du changement de prix, tel qu'indiqué dans le système de caisse. Exemple : 'réduction menu', 'happy hour', 'on the house'.

scope (enum, obligatoire)

Indique à quoi la modification de prix s'applique. Valeur à choisir dans la table tb_priceChangeScope.

Scope	Description
PRODUCT	product price change
LINE	transactionLine price change
EVENT	Event price change

type (enum, obligatoire)

Identification du type de changement de prix, à choisir dans la table tb_priceChangeType.

type	Description
PUBLIC	Visible pour le client
INTERNAL	Recalcul interne

amount (numérique, deux décimales, obligatoire)

Montant du changement de prix. Il peut être positif, négatif ou nul.

}

Enregistrement des transactions – mouvements financiers

Drawer

drawer (objet, optionnel)

Ne peut être utilisé qu'à la racine de la structure JSON avec un eventOperation DRAWER_OPEN, lorsque le tiroir-caisse s'ouvre sans transaction et est enregistré comme tel dans le système de caisse. Le tiroir-caisse peut être utilisé si l'entreprise souhaite indiquer exactement quel tiroir-caisse/portefeuille est/a été utilisé et dans la mesure que le système de caisse prenne en charge cette fonctionnalité.

Pour l'utilisation de l'objet drawer dans les autres types de transactions, voir financialLine.

{

id (string, obligatoire, longueur min. 1 et max. 600 caractères)

Identification du tiroir-caisse/portefeuille.

name (string, obligatoire, longueur min. 1 et max. 600 caractères)

Nom du tiroir-caisse/portefeuille, tel que prévu dans le système de caisse.

}

Mouvements d'argent

financials (array de paymentLine ou de l'objet moneyInOutLine)

Cet array contient toutes les lignes financières d'un event, en EUR ou en devise. Cet array peut être utilisé pour les events 'N' lorsqu'ils concernent une transaction de vente ou 'F' lorsqu'ils impliquent un simple mouvement monétaire.

Les objets paymentLine et moneyInOutLine incluent un mouvement d'argent d'un moyen de paiement et la signification du paiement.

Lignes de paiement

Pour un événement 'N', cela signifie l'enregistrement d'un paiement ou d'un paiement différé. Si le montant à payer est nul, un financials vide est autorisé.

Dans le cas d'un paiement différé, le mode de paiement 'CUSTOMER_CREDIT' est utilisé, et pour un transfert vers une chambre d'hôtel dans un event N, le mode de paiement 'ROOM_CREDIT' est utilisé.

Simples mouvements d'argent

Pour un event 'F', cela signifie l'enregistrement de mouvements financiers sans lien avec des ventes ou une modification du mode de paiement d'une transaction de vente déjà clôturée.

paymentLine (objet)

L'objet paymentLine permet d'enregistrer correctement un **paiement** pour permettre une parfaite gestion des encaissements

{

Id (string, obligatoire, longueur min. 1 et max. 600 caractères)

ID du changement de prix, tel qu'indiqué dans le système de caisse, quelle que soit la langue.

name (string, obligatoire, longueur min. 1 et max. 600 caractères)

Nom du moyen de paiement utilisé, tel qu'encodé dans le système de caisse. Exemples : 'Cash', 'Carte de débit'.

type (enum, obligatoire)

Type du moyen de paiement, à choisir dans la table tb_paymentTypes.

tb_paymentTypes

type	Remarque
UNKNOWN	Type non enregistré
CASH	En liquide
CARD_DEBIT	Paiement par carte avec carte de débit
CARD_UNKNOWN	Paiement par carte avec carte non précisée
CARD_CREDIT	Paiement par carte avec carte de crédit
CARD_OTHER	Paiement par carte non mentionné précédemment
CHEQUE_MEAL	Paiement avec chèque restaurant
CHEQUE_OTHER	Chèque de paiement (éco, culture, ...)
APP	Paiement avec l'application de paiement
ONLINE	Paiement en ligne
CUSTOMER_CREDIT	Paiement à crédit/différé
ROOM_CREDIT	Paiement via la facture globale de l'hôtel
LOYALTY_REWARDS	Paiement avec points de fidélité
VOUCHER_STORE	Paiement avec bon d'entreprise
VOUCHER_SUPPLIER	Paiement avec bon du fournisseur
VOUCHER_OTHER	Paiement avec bon non mentionné précédemment
OTHER	Tout autre non mentionné précédemment

provider (string, optionnel, longueur max. = 600 caractères)

Émetteur du moyen de paiement (Exemple : Visa, MasterCard, Edenred, ...)

inputMethod (enum, obligatoire)

Une des valeurs de la table tb_inputMethod :

Type	Description
MANUAL	Enregistré manuellement
AUTOMATIC	EFT, autres liens

Si le système de caisse est directement connecté au terminal de paiement (et enregistre donc le paiement via EFT), le terminal de paiement renvoie lui-même le type de carte utilisé.

amount (numérique, deux décimales, obligatoire)

Montant de la ligne de paiement ou du mouvement d'argent, exprimé en euros. Ce montant peut être positif, négatif ou nul (0).

amountType (enum, obligatoire)

Décrit à quoi fait référence le montant, type à choisir dans tb_mountType

Type	Description
PAYMENT	Paiement
TIP	Pourboire
ROUNDING	Arrondissement du paiement

- Tip: est utilisé pour enregistrer les pourboires reçus. Un pourboire peut être reçu dans le cadre d'une transaction ou être enregistré comme un mouvement d'argent
- Rounding: obligatoire pour enregistrer l'arrondi des moyens de paiement.

Exemple: montant à payer, 9,97 et le client paie en espèces. La caisse doit arrondir à 9,95. Cela signifie que deux lignes de paiement doivent être répertoriées.

[{"name" : "CONTANT", "type" : "CASH", , "inputMethod":"MANUAL", "amount" : 9.97, "amountType": "PAYMENT"},

{"name" : "CONTANT", "type" : "CASH", "inputMethod":"MANUAL", "amount": -0.02, "amountType": "ROUNDING"}]

foreignCurrency (objet, optionnel, voir foreignCurrency)

Cet objet n'est utilisé que si le système de caisse prend en charge les paiements en devises étrangères.

Attention: cet objet ne peut pas être utilisé en combinaison avec le montantType ROUNDING ;

{

amount (numérique, obligatoire)

Le montant en devise étrangère. Ce montant peut être positif, négatif ou nul (0).

iso (string, obligatoire, longueur = 3)

Notation ISO 4217 (code alphabétique en majuscules) de la monnaie concernée. Exemples : : 'USD', 'GBP', 'SEK'.

}

reference (string, optionnel, longueur min. 1 en max. 600 caractères)

Si elle est utilisée par le système de caisse, la référence du paiement peut être saisie ici.

drawer (objet, optionnel, description voir plus haut)

}

moneyInOutLine (objet)

L'objet moneyInOutLine permet d'enregistrer correctement un **mouvement d'argent** pour permettre une parfaite gestion des encaissements.

{

Id (string, obligatoire, longueur min. 1 et max. 600 caractères)

ID du changement de prix, tel qu'indiqué dans le système de caisse, quelle que soit la langue.

name (voir paymentLine.name)

type (voir paymentLine.type)

provider (voir paymentLine.provider)

inputMethod (voir paymentLine.inputMethod)

amount (voir paymentLine.amount)

amountType (enum, obligatoire)

Décrit à quoi fait référence le montant, à choisir dans tb_amountType

Type	Description
MONEY_IN_OUT	Changements en + ou en – du contenu du tiroir-caisse.
TIP	Pourboire
DRAWER_DECLARATION	Comptage du contenu du tiroir-caisse

Tip: Utilisé pour enregistrer les pourboires reçus. Un pourboire peut être reçu dans le cadre d'une transaction ou être enregistré avec un mouvement d'argent.

foreignCurrency (voir paymentLine.foreignCurrency)

reference (voir paymentLine.reference)

drawer (voir paymentLine.drawer)

}

Rapports

Ceux-ci sont construits avec le contenu des valeurs, objets ou arrays ci-dessous.

reportNo (numérique, requis uniquement pour Z, valeur comprise entre 1 et 999999999)

Numérotation continue du rapport (Z uniquement), voir l'article 27 de l'arrêté ministériel. Fait partie à la fois du rapport chiffre d'affaires et du rapport sur les utilisateurs.

reportBookingDate (string, obligatoire)

La bookingDate concernée par ce rapport, dans le format ISO 8601 mentionné plus haut.

posDevices (array des objets posDevice, obligatoire)

Contient les informations de tous les systèmes de caisse dont les chiffres sont inclus dans ce rapport. Il fait partie à la fois du rapport chiffre d'affaires ainsi que du rapport utilisateurs.

fdmDevices (array d'objets fdmDevice, obligatoire)

Contient les informations de tous les FDM liés à la caisse et dont les chiffres sont inclus dans ce rapport. Il fait partie à la fois du rapport chiffre d'affaires ainsi que du rapport utilisateurs.

turnover (objet, partie obligatoire du rapport chiffre d'affaires)

Contient toutes les informations relatives aux revenus pour la reportBookingDate concernée.

users (array des objets, partie obligatoire du rapport utilisateurs)

Contient toutes les informations pertinentes sur les utilisateurs du système de caisse pour la reportBookingDate concernée.

posDevice

Cet objet contient l'identité et la période de tous les POS dont les données sont incluses dans ce rapport.

{

posId (string, obligatoire)

Le posId tel qu'il est utilisé lors de l'enregistrement des events.

terminalId (string, obligatoire)

Le terminalId tel qu'il est utilisé lors de l'enregistrement des events.

firstPosDateTime (string, obligatoire)

L'horodatage du premier enregistrement avec la combinaison posId et terminalId pour la bookingDate concernée.

lastPosDateTime (string, obligatoire)

L'horodatage du dernier enregistrement avec la combinaison posId et terminalId pour la bookingDate concernée.

}

fdmDevice

Cet objet contient les détails de tous les FDM dont les chiffres sont inclus dans le rapport et la référence à la première et à la dernière transaction incluse dans le rapport.

{

fdmId (string, obligatoire)

firstFdmDateTime (string, obligatoire)

firstTotalCounter (numérique, obligatoire, min. 1 et max. 999999999)

lastFdmDateTime (string, obligatoire)

lastTotalCounter (numérique, obligatoire, min. 1 et max. 999999999)

}

turnover

Cet objet conditionnel contient le chiffre d'affaires total réalisé, le chiffre d'affaires par département et par taux de TVA. Il n'est utilisé que dans le rapport chiffre d'affaires.

{

transactions (array de tous les eventLabels utilisés avec leurs totaux)

[

{

eventLabel (string, obligatoire, voir description ci-dessus)

ticketCount (intégral, obligatoire)

Total du nombre d'events de l'eventLabel concerné pendant la bookingDate correspondante.

amount (numérique, obligatoire, deux décimales)

Som(VatInput.Price)+Som(VatInput.PricesChanges.Amount) de chaque ProductInput avec ce departementId

}

]

departements (array des objets contenant les données relatives au chiffre d'affaires de tous les départements utilisés, obligatoire)

[

{

departmentId (string, obligatoire, voir description ci-dessus)

departmentName (string, obligatoire, voir description ci-dessus)

amount (numérique, deux décimales, obligatoire)

Somme(VatInPut Price)+Som(VatInPut PricesChanges Amount) van elke ProductInput met dit depaartmentId **}**

}

]

vats (array des objets contenant les données relatives au chiffre d'affaires pour tous les taux de TVA utilisés, obligatoire)

[

{

label (string, obligatoire, voir description ci-dessus)

rate (numérique, obligatoire, valeur min. = 0 et valeur max. = 100, deux décimales, voir description au chapitre 2.2.4)

taxableAmount (numérique, deux décimales, obligatoire)

Le montant total des valeurs taxableAmount (voir description au chapitre 2.2.4) pour le label concerné.

vatAmount (numérique, deux décimales, obligatoire)

Le montant total des valeurs vatAmount (voir description au chapitre 2.2.4) pour le label concerné.

totalAmount (numérique, deux décimales, obligatoire)

Le montant total des valeurs totalAmount (voir description au chapitre 2.2.4) pour le label concerné.

}

]

payments (array des objets contenant les montants reçus en lien avec les chiffres d'affaires mentionnés ci-dessus par type de mode de paiement, obligatoire)

[

{

id (string, obligatoire, voir description ci-dessus)

name (string, obligatoire, voir description ci-dessus)

type (enum, obligatoire, voir description ci-dessus)

amount (array des objets PaymentTotalAmount, obligatoire)

[

{

type (enum PaymentLineType, obligatoire)

normalAmount (numérique, deux décimales, obligatoire)

Contient le montant total de tous les paiements relatifs au chiffre d'affaires réalisé, à déclarer en euros (Sale).

negativeCorrections (numérique, deux décimales, obligatoire)

Contient le montant total des corrections de paiement négatives.

positiveCorrections (numérique, deux décimales, obligatoire)

Contient le montant total des corrections de paiement positives.

correctionsCount (entier, obligatoire)

Le nombre total de corrections de paiement.

totalAmount (numérique, deux décimales, obligatoire)

Le total de normalAmount, negativeCorrections et positiveCorrections

normalForeign (array des objets foreignCurrency, obligatoire)

Contient le montant total par devise de tous les paiements en devises liés au chiffre d'affaires réalisé (Sale).

correctionsForeign (array des objets foreignCurrency, conditionnel)

Contient le montant total des corrections de paiement en devise étrangère. Ces montants sont déjà inclus en euros dans les autres champs (negativeCorrections, positiveCorrections et totalAmount).

}

]

}

]

drawersOpenCount (numérique, min. 0 et max. 999999999, obligatoire)

Contient le nombre de fois où un (ou plusieurs) tiroirs-caisse(s) ont été ouverts (via le système de caisse) sans enregistrement de vente.

negQuantities (array des objets, obligatoire)

Contient la quantité et le montant total de toutes les quantités négatives utilisées.

[

{

reason (enum, obligatoire, voir description ci-dessus)

count (numérique, obligatoire, min. 1 et max. 999999999)

Le nombre de fois où un enregistrement négatif a été utilisé pendant la bookingDate.

ticketCount (numérique, obligatoire, min. 1 et max. 999999999)

Le nombre de tickets pour cette reason.

amount (numérique, obligatoire, deux décimales)

Le montant total de ces enregistrements négatifs pour ce type.

}

]

priceChanges (array des objets, obligatoire)

Contient l'aperçu complet de tous les changements de prix comptabilisés.

[

{

id (string, obligatoire, longueur min. 1 et max. 600 caractères)

Id du changement de prix, tel que prévu dans le système de caisse, indépendamment de la langue.

name (string, obligatoire, voir description ci-dessus de priceChange.name)

type (enum, obligatoire, voir description ci-dessus de priceChange.type)

À limiter au type priceChange.type PUBLIC.

amount (array des objets, obligatoire)

Contient le montant total de tous les changements de prix pendant la bookingDate concernée, ventilé par taux de TVA utilisé et selon qu'ils s'agissent de changements positifs ou négatifs)

[

{

label: (string, obligatoire, voir description ci-dessus)

negative: (numérique, deux décimales, obligatoire)

Est la somme de tous les changements de prix **négatifs** pour le tarif TVA concerné.

positive (numérique, deux décimales, obligatoire)

Est la somme de tous les changements de prix **positifs** pour le tarif TVA concerné.

}

]

}

]

invoices (array des objets, conditionnel)

Aperçu de toutes les factures réalisées à l'aide du POS pendant la bookingDate concernée.

[

{

invoiceNo (string, obligatoire, voir description ci-dessus)

amount (numérique, obligatoire, voir description ci-dessus de transactionTotal)

}

]

}

users

Cet array des objets contient les informations du rapport de l'utilisateur, ventilées par utilisateur.

[

{

employeeId (string, obligatoire, voir description ci-dessus)

totalAmount (numérique, deux décimales, obligatoire).

Le montant total des tous les transactionTotal qui ont été enregistrées via les events 'N' par l'utilisateur concerné.

firstPosDateTime (string, obligatoire, format tel que décrit pour posDateTime)

L'horodatage du **premier** enregistrement d'un event par l'utilisateur concerné pendant cette période.

lastPosDateTime (string, obligatoire, format tel que décrit pour posDateTime)

L'horodatage du **dernier** enregistrement d'un event par l'utilisateur concerné pendant cette période.

socialEvents (array des objets, obligatoire)

Contient les horodatages de la connexion et de la déconnexion de l'utilisateur concerné par l'utilisation de l'event 'S'.

[

{

posDateTime (string, obligatoire, voir description ci-dessus)

Horodatage de l'enregistrement de l'event 'S' concerné.

inOut (enum, obligatoire)

Valeur à choisir dans tb_inOut

inOut	Déscription
IN	Work In
OUT	Work Out

}

]

payments (array des objets, obligatoire)

Cet array contient le résumé de tous les enregistrements de tous les paiements de l'utilisateur concerné pour cette bookingDate, ventilés par type utilisé). Il a la même structure que pour l'objet turnover.

Règles de communication

Afin de garantir un traitement correct des requests et de les distinguer de manière unique, les champs clés suivants sont déterminés :

- **posId**
- **posDateTime**
- **terminalId**
- **eventLabel**
- **posFiscalTicketNo**
-

Lorsqu'un POS transmet une request au FDM dans un délai de 10 minutes, où ces 5 champs clés sont identiques à ceux envoyés dans un message précédent, le FDM doit vérifier s'il s'agit ou non d'une request à laquelle il a déjà été répondu (y compris en augmentant les compteurs concernés dans le FDM).

Les situations suivantes peuvent se présenter :

1. La combinaison des champs clés est différente : le FDM traite cette demande de manière normale.
2. Le contenu de la JSON request est totalement identique : le FDM traite cette requête comme un 'double' envoi et renvoie dans la réponse les mêmes données que dans la request originale; le FDM n'augmente pas non plus ses compteurs ; cette transaction en double n'est pas suivie dans le FDM.
3. La combinaison des champs clés est identique, mais les autres valeurs de la JSON request sont différentes de celles de la request précédemment traitée : le FDM refuse de traiter cette request et envoie un code d'erreur dans sa réponse.

2.2.3. FDM ➔ POS – aperçu complet

Après avoir reçu les requests décrites ci-dessus, le FDM effectue les calculs nécessaires, ajuste les compteurs nécessaires, crée l'objet JSON canonique, signe les données.

Le service GraphQL du FDM renvoie les champs demandés au POS.

Vous trouverez ci-dessous l'aperçu complet des réponses :

posId (scalar value)
posFiscalTicketNo (scalar value)
posDateTime (scalar value)
terminalId (scalar value)
deviceId (scalar value)
eventOperation (scalar value)
fdmRef (object)
fdmSwVersion (scalar value)
digitalSignature (scalar value)
shortSignature (scalar value)
verificationUrl (scalar value)
vatCalc (array des objets)
bufferCapacityUsed (scalar value)
warnings (array des objets)
informations (array des objets)

2.2.4. FDM ➔ POS – aperçu détaillé

Un certain nombre d'arrays, d'objets ou de valeurs qui n'ont pas été décrits auparavant sont décrits dans ce point.

eventOperation (enum, obligatoire)

Valeur à choisir dans la table tb_eventOperation.

tb_eventOperation

eventOperation	eventLabel	déscription
WORK_IN	S	Pointage de début sur la caisse
WORK_OUT	S	Pointage de fin sur la caisse
SALE	N	Clôture de la transaction de vente
INVOICE	I	Facture d'un ticket de caisse TVA émis précédemment
COST_CENTER_CHANGE	P	Modification de l'affectation du costcenter
ORDER	P	Enregistrement de l'ordre du costCenter
PRE_BILL	P	Création d'une addition provisoire
MONEY_IN_OUT	F	Enregistrement de mouvements d'argent en dehors des ventes
DRAWER_OPEN	F	Ouverture du tiroir-caisse en dehors des ventes
PAYMENT_CORRECTION	F	Modification du mode de paiement déjà enregistré
REPORT_TURNOVER_X	R	Rapport du chiffre d'affaires X
REPORT_TURNOVER_Z	R	Rapport du chiffre d'affaires Z
REPORT_USER_X	R	Rapport des utilisateurs X
REPORT_USER_Z	R	Rapport des utilisateurs Z
COPY	C	Copie d'une eventOperation énumérée ci-dessus
(*eventOperation*)	T	Tous les eventOperation en mode training

fdmRef (objet, obligatoire)

Contient l'identification et l'horodatage du FDM et, si une signature a pu être fournie, les relevés du compteur associés. Cela permet d'identifier de manière unique la transaction signée.

{

fdmId (string, obligatoire)

Le numéro de série du FDM qui répond à la request. Longueur = 11.

Exemple: SPF01987654

fdmDateTime (string, obligatoire)

La date et l'heure de la réponse provenant du RTC du FDM. **UTC+00** est toujours utilisé (donc : pas d'utilisation de l'heure d'été ou d'hiver). Le format ISO 8601 suivant est obligatoire.

Exemple: 2022-10-20T15:01:26Z

eventLabel (enum, obligatoire, longueur = 1)

L'eventLabel qui correspond à la mutation envoyée par le POS au service GraphQL du FDM, sur la base de laquelle le FDM a incrémenté le bon compteur. L'eventtype lui-même est donc renvoyé dans cette réponse (voir chapitre 3).

Exemple: 'N'.

eventCounter (numérique, obligatoire, min. 1 et max. 999999999)

Le compteur d'event. Le FDM possède un certain nombre de compteurs intégrés, voir n° 64 de l'arrêté ministériel. Le FDM fournit la dernière la valeur (après ajustement en fonction du contenu de la request concernée) du compteur de l'event concerné.

Exemple: 46895

totalCounter (numérique, obligatoire, min. 1 et max. 999999999)

Le compteur total d'event. Le FDM possède un certain nombre de compteurs intégrés, voir n° 64 de l'arrêté ministériel. Le FDM fournit la dernière valeur (après ajustement en fonction du contenu de la request concernée) du compteur total.

Exemple: 53896

}

digitalSignature (string, obligatoire, min 1 caractère)

Ce champ contient la signature numérique placée par le certificat de signature du FDM. Cette signature est apposée selon les règles décrites au point 2.4.

shortSignature (string, conditionnel, min 1 caractère)

Ce champ contient la signature abrégée. C'est cette signature qui sera **imprimée** sur le ticket de caisse TVA. Par conséquent, il est envoyé **uniquement** dans le cas d'un event 'NORMAL'. Elle est calculée comme suit :

shortSignature = Hex(SHA1(Base64Decode(digitalSignature))).

verificationurl (string, conditionnel, min. 1, max. 60)

Cette URL est générée par le FDM sous de code QR sur le ticket de caisse TVA (voir chapitre 3). Le préfixe est transmis par le SPFFIN au FDM, qui crée ensuite l'URL sur la base de certaines données d'event.

Ce string est ajouté uniquement à la réponse JSON pour les events de type 'N'

fdmSwVersion (string, obligatoire, min. 1 et max. 10 caractères)

La version du firmware du FDM au moment de la réponse. Exemple : '1.2.0'.

bufferCapacityUsed (numérique, obligatoire, 2 décimales)

La capacité du buffer utilisée, exprimée en pourcentage.

vatCalc (array de l'objet vatCalcItem, obligatoire si des objets vat sont contenus dans la request)

Le FDM est chargé de calculer correctement le montant imposable et la TVA pour chaque taux applicable. Ce calcul n'est effectué que pour le type de transaction/eventlabel 'N'.

Pour chaque code TVA, le montant imposable et le montant de la TVA sont calculés.

Le contenu des champs vatId et rate est déterminé en fonction des données fournies par le SPF Finances (art. 66 de l'arrêté ministériel). Le contenu des champs taxableAmount et vatAmount est calculé par le FDM.

Un objet n'est introduit que pour les vatLabels pour lesquels des lignes de produits ont été transmises dans la request.

[

{

label (enum, obligatoire)

Le label TVA tel que reçu du SPF Finances. Exemple : 'B'.

rate (numérique, obligatoire, valeur min. = 0 et valeur max. = 100, 2 décimales)

Le pourcentage du label tva concerné, tel que reçu par le FDM du SPF Finances.

Exemple: 12

taxableAmount (numérique, deux décimales, obligatoire)

Montant imposable, pour le taux de TVA applicable. Pour le calcul, il est fait référence au numéro 66 de l'arrêté ministériel. Cette valeur peut être positive, négative ou nulle (0).

Exemple: 8.93

vatAmount (numérique, deux décimales, obligatoire)

Montant de la TVA, pour le taux de TVA concerné et calculé sur la base de la base imposable communiquée ci-dessus. Pour le calcul, il est fait référence au numéro 66 de l'arrêté ministériel. Cette valeur peut être positive, négative ou nulle (0).

Exemple: 1.07

totalAmount: (numérique, deux décimales, obligatoire)

outOfScope: (true of false, obligatoire)

}

]

warnings (array des objets message, optionnel, max. 50 objets, voir description)

Liste des avertissements sur lesquels le FDM veut/doit attirer l'attention de la caisse.

Message

{

category (enum, obligatoire)

Valeur à choisir dans la table tb_messageCategory.

tb.messageCategory

categorie	omschrijving
SPF_FOD	Messages du SPF Finances envoyés au FDM via l'API. Les codes de tb_Codes sont obligatoirement utilisés
FDM	Messages du FDM – complétés librement par le producteur
OTHER	Autres messages.

code (enum, obligatoire)

Le contenu de l'avertissement ou des informations fournies à la caisse par le FDM. Pour la catégorie SPF_FOD, seuls les codes de tb_messages sont utilisés. Pour les autres catégories, ils sont librement choisis par les producteurs.

message (string, obligatoire)

Description du message, tel que défini dans le présent arrêté ou par le producteur du FDM. La langue utilisée correspond à la langue indiquée dans le champ langue de la request. Pour les messages associés à la catégorie SPF_FOD, ceux-ci doivent être basés sur les messages de tb_messages ci-dessous.

tb_messages

Code	Message
CLIENT_CERT_NEAR_EXPIRATION	Client certificate expires in less than 4 months
CLIENT_CERT_EXPIRED	Client certificate has expired
RTC_SYNC_FAILED	FDM could not synchronize real-time clock.
UPDATE_URLS_FAILED	FDM could not update URLs.
UPDATE_TRUST_CERT_FAILED	FDM fails to update trust certificates
UPDATE_TASK_LIST_FAILED	FDM could not download the task list.
TASK_FEEDBACK_FAILED	FDM could not report the task result
NOP_FAILED	FDM could not post a NOP message
BUFFER_NEAR_FULL	FDM buffer usage exceeds 70 %
SERVER_CERT_RESOLVE_FAILED	FDM could not resolve server certificate
TRANSACTION_UPLOAD_FAILED	FDM has more than 10 failed attempts to post business transaction
INITIALIZATION_FAILED	FDM initialization has failed
RTC_NOT_INITIALIZED	FDM Real Time Clock not synchronized
CORRUPT_RECORD_ENCOUNTERED	FDM memory contains corrupt transaction
UPDATE_PARAMS_FAILED	FDM could not update parameters
UPDATE_CLIENT_CERT_FAILED	FDM could not update the client certificate
UPDATE_VAT_RATES_FAILED	FDM could not update the vat rates
UDPATE_POS_ALLOWLIST_FAILED	FDM could not update the POS allowlist

data (array des objets data, optionnel)

Peut contenir des informations utiles supplémentaires destinées aux développeurs ou au personnel de support. N'est pas destiné à être affiché via l'interface utilisateur de la caisse.

showPos (booléen, obligatoire)

Cette valeur (vrai, faux) détermine si le message est affiché à l'utilisateur via l'interface utilisateur du POS. Pour la catégorie SPF_FOD, la valeur par défaut est 'true'.

}

Exemple:


```
{
"category": "SPF_FOD",
"code": "BUFFER_OVER_70_PCT",
"message": "FDM buffer usage exceeds 70 %",
"data": [ ],
'showPos': true
}
```


Data

{

name (string, obligatoire)

Contient le nom de la valeur. Par exemple : Offset (indique où se trouve l'erreur), Attendu (ce qui était attendu), Rencontré (ce qui a été envoyé)

value (string, obligatoire)

La valeur supplémentaire donnée à un message.

}

Informations (array des objets Message, optionnel, voir description ci-dessus)

Liste des messages supplémentaires dont le FDM souhaite informer la caisse.

2.2.5. FDM ➔ POS – error handling

Les erreurs sont renvoyées dans les arrays des erreurs selon le format GraphQL. Chaque objet d'erreur de l'array contient obligatoirement une string 'message', qui fournit une description de l'erreur, éventuellement un array 'locations' indiquant où dans la requête l'erreur s'est produite et éventuellement un array 'path' contenant le chemin d'accès à l'erreur.

Toutes les erreurs liées à la syntaxe, à la validation et à l'exécution sont déterminées par le FDM.

Exemple:


```
{
 'errors': [
  {
  'message': 'Variable '$data' got invalid value 'N' at 'data.language'; Value 'N' does not exist in 'Language' enum. Did you mean the enum value 'EN' or 'NL'?',
  'locations': [
   {
```

```
    'line': 1,
    'column': 20
    }
   ]
   }
 ]
}
```


Les erreurs déterminées par l'administration sont ajoutées à l'objet d'erreur en tant qu'objet 'message' dans le array des extensions.
L'objet message a la même structure que l'objet message dans le array des avertissements et des informations.

{

category (enum, obligatoire)

Valeur à sélectionner dans tb_messageCategory (voir ci-dessus sous message)

code (enum, obligatoire)

Le contenu de l'erreur fournie à la caisse par le FDM. Si la demande en est faite, c'est ce contenu qui est affiché sur l'écran de caisse. Pour la catégorie SPF_FOD, seuls les codes de tb_error_messages sont utilisés. Pour les autres catégories, ils sont librement choisis par les producteurs.

message (string, obligatoire)

Description du message, telle que définie dans le présent arrêté ou par le producteur du FDM. La langue utilisée correspond à la langue indiquée dans le champ langue de la request. Pour les messages associés à la catégorie SPF_FOD, ceux-ci doivent être basés sur les messages de tb_error_messages ci-dessous.

tb_error_messages

Code	Message
BUFFER_FULL	FDM buffer is full
UNKOWN_POS	POS is not registered for use with this FDM
FDM_LOCKED	FDM is locked
TOO_MANY_MEMORY_ERRORS	FDM's memory contains too many corrupt transactions

data (array de data objects, optionnel)

Peut contenir des informations utiles supplémentaires destinées aux développeurs ou au personnel de support. N'est pas destiné à être affiché via l'interface utilisateur de la caisse enregistreuse.

(Voir description de l'objet de données au point 2.2.4).

showPos (boolean, obligatoire)

Cette valeur (true, false) détermine si le message d'erreur est affiché à l'utilisateur via l'interface utilisateur du POS. Pour la catégorie SPF_FOD, la valeur par défaut est 'true'.

}

Exemple error response:


```
{
 'errors': [
  {
   'message': 'FDM error',
```

```
    'extensions':
     {
      'category': 'SPF_FOD',
      'code': 'BUFFER_FULL',
      'message': 'FDM buffer is full.',
      'showPos': true
     }
   }
 ]
}
```

Note générale concernant le traitement des erreurs

Certaines erreurs et avertissements sont affichés à l'utilisateur via l'interface utilisateur du POS. En plus des dispositions ci-dessus, le fabricant du FDM fournit à l'utilisateur les informations supplémentaires nécessaires afin qu'il puisse prendre les mesures appropriées pour résoudre l'erreur.

Ceci est décrit en détail dans la documentation fournie avec la demande de certification.

2.3. GRAPHQL SCHÉMA POUR LA COMMUNICATION POS VERS FDM

Le schéma complet est décrit ci-dessous.

2.3.1. Mutations

Mutations disponibles sur le FDM :

- signWorkIn(data: WorkInOutInput!, isTraining: Boolean! = false): SignResult
- signWorkOut(data: WorkInOutInput!, isTraining: Boolean! = false): SignResult
- signInvoice(data: InvoiceInput!, isTraining: Boolean! = false): SignResult
- signSale(data: SaleInput!, isTraining: Boolean! = false): SignResult
- signCostCenterChange(data: CostCenterChangeInput!, isTraining: Boolean! = false): SignResult
- signOrder(data: OrderInput!, isTraining: Boolean! = false): SignResult
- signPreBill(data: PreBillInput!, isTraining: Boolean! = false): SignResult
- signMoneyInOut(data: MoneyInOutInput!, isTraining: Boolean! = false): SignResult
- signDrawerOpen(data: DrawerOpenInput!, isTraining: Boolean! = false): SignResult
- signPaymentCorrection(data: PaymentCorrectionInput!, isTraining: Boolean! = false): SignResult
- signReportTurnoverX(data: ReportTurnoverXInput!, isTraining: Boolean! = false): SignResult
- signReportTurnoverZ(data: ReportTurnoverZInput!, isTraining: Boolean! = false): SignResult
- signReportUserX(data: ReportUserXInput!, isTraining: Boolean! = false): SignResult
- signReportUserZ(data: ReportUserZInput!, isTraining: Boolean! = false): SignResult
- signCopy(data: CopyInput!, isTraining: Boolean! = false): SignResult

2.3.2. Types

```
type SignResult {
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  terminalId: String
  deviceId: String!
  eventOperation: EventOperation!
  fdmRef: FdmReference!
  fdmSwVersion: String!
  digitalSignature: String!
  shortSignature: String
  verificationUrl: String
  vatCalc: [VatCalcItem!]
  bufferCapacityUsed: Float!
  warnings: [MessageItem!]
  informations: [MessageItem!]
}

type FdmReference {
  fdmId: String!
  fdmDateTime: String!
  eventLabel: EventLabel!
  eventCounter: Int!
  totalCounter: Int!
}

type VatCalcItem {
  label: VatLabel!
  rate: Float!
  taxableAmount: Float!
  vatAmount: Float!
  totalAmount: Float!
  outOfScope: Boolean!
}


type MessageItem {
  category: Category!
  code: Code!
  message: String!
  data: [DataItem!]
  showPos: Boolean!
}

type DataItem {
  name: String!
  value: String!
}
```

2.3.3. Enums

Cette section décrit les différents enums. Consultez toujours la version la plus récente via le site web https://www.systemedecaisseenregistreuse.be/fr.

```
enum Language  {
    EN
    NL
    FR
    DE
}

enum TicketMedium  {
  NONE
  PAPER
  DIGITAL
  PAPER_DIGITAL
}

enum InOut  {
  IN
  OUT
}

enum PriceChangeType  {
  PUBLIC
  INTERNAL
}

enum PriceChangeScope  {
  PRODUCT
  LINE
  EVENT
}

enum VatLabel  {
  A
  B
  C
  D
  X
}

enum EventLabel  {
  N
  P
  F
  S
  I
  R
  C
  T
}


enum QuantityType  {
  PIECE
  KILOGRAM
  METER
  LITRE
  HOUR
}
```

```
enum TransactionLineType {
  SINGLE_PRODUCT
  COMPOSITE_PRODUCT
}

enum NegQuantityReason {
  REFUND
  CORRECTION
  PRICE_CHANGE
  COST_CENTER_CHANGE
  PRODUCT_SUBSTITUTION
  VOUCHER
  OTHER
}

enum PaymentType {
  UNKNOWN
  CASH
  CARD_UNKNOWN
  CARD_DEBIT
  CARD_CREDIT
  CARD_OTHER
  CHEQUE_MEAL
  CHEQUE_OTHER
  APP
  ONLINE
  CUSTOMER_CREDIT
  ROOM_CREDIT
  LOYALTY_REWARDS
  VOUCHER_STORE
  VOUCHER_SUPPLIER
  VOUCHER_OTHER
  OTHER
}

enum InputMethod {
  MANUAL
  AUTOMATIC
}

enum PaymentLineType {
  PAYMENT
  TIP
  ROUNDING
}

enum MoneyInOutLineType {
  MONEY_IN_OUT
  TIP
  DRAWER_DECLARATION
}

enum CostCenterType {
  TABLE
  CHAIR
  ROOM
  CUSTOMER
  ON HOLD
  OTHER
```

```
}

enum EventOperation  {
  WORK_IN
  WORK_OUT
  SALE
  INVOICE
  COST_CENTER_CHANGE
  ORDER
  PRE_BILL
  MONEY_IN_OUT
  DRAWER_OPEN
  PAYMENT_CORRECTION
  COPY
  REPORT_TURNOVER_X
  REPORT_TURNOVER_Z
  REPORT_USER_X
  REPORT_USER_Z
}

enum Category  {
  SPF_FOD
  FDM
  OTHER
}

enum Code  {
  CLIENT_CERT_NEAR_EXPIRATION
  CLIENT_CERT_EXPIRED
  RTC_SYNC_FAILED
  UPDATE_URLS_FAILED
  UPDATE_TRUST_CERT_FAILED
  UPDATE_TASK_LIST_FAILED
  TASK_FEEDBACK_FAILED
  NOP_FAILED
  BUFFER_NEAR_FULL
  SERVER_CERT_RESOLVE_FAILED
  TRANSACTION_UPLOAD_FAILED
  INITIALIZATION_FAILED
  RTC_NOT_INITIALIZED
  CORRUPT_RECORD_ENCOUNTERED
  UPDATE_PARAMS_FAILED
  UPDATE_CLIENT_CERT_FAILED
  UPDATE_VAT_RATES_FAILED
  UDPATE_POS_ALLOWLIST_FAILED
  BUFFER_FULL
  UNKOWN_POS
  FDM_LOCKED
  TOO_MANY_MEMORY_ERRORS
}
```

2.3.4. input objects

```
input FdmReferenceInput {
  fdmId: String!
  fdmDateTime: String!
  eventLabel: EventLabel!
  eventCounter: Int!
  totalCounter: Int!
```

```
}

input DrawerInput {
  id: String!
  name: String!
}

input CostCenterInput {
  id: String!
  type: CostCenterType!
  reference: String!
  costCenter: CostCenterInput
}

input PriceChangeInput {
  groupingId: Int
  id: String!
  name: String!
  scope: PriceChangeScope!
  type: PriceChangeType!
  amount: Float!
}

input VatInput {
  label: VatLabel!
  price: Float!
  priceChanges: [PriceChangeInput!]
}

input ProductInput {
  gtin: String
  productId: String!
  productName: String!
  departmentId: String!
  departmentName: String!
  quantity: Float!
  quantityType: QuantityType!
  negQuantityReason: NegQuantityReason
  unitPrice: Float!
  vats: [VatInput!]!
}

input ForeignCurrencyInput {
  amount: Float!url
  iso: String!
}

input TransactionLineInput {
  lineType: TransactionLineType!
  mainProduct: ProductInput!
  subProducts: [ProductInput!]
  costCenter: CostCenterInput
  lineTotal: Float!
}

input TransactionInput {
  transactionLines: [TransactionLineInput!]!
  transactionTotal: Float!
```

```
}

input PaymentLineInput {
  id: String!
  name: String!
  type: PaymentType!
  provider: String
  inputMethod: InputMethod!
  amount: Float!
  amountType: PaymentLineType!
  foreignCurrency: ForeignCurrencyInput
  reference: String
  drawer: DrawerInput
}

input TransferInput {
  from: [TransferItemInput!]!
  to: [TransferItemInput!]!
}

input TransferItemInput {
  costCenter: CostCenterInput!
  transaction: TransactionInput!
}

input MoneyInOutInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  financials: [MoneyInOutLineInput!]!
}

input MoneyInOutLineInput {
  id: String!
  name: String!
  type: PaymentType!
  provider: String
  inputMethod: InputMethod!
  amount: Float!
  amountType: MoneyInOutLineType!
  foreignCurrency: ForeignCurrencyInput
  reference: String
  drawer: DrawerInput
}

input WorkInOutInput  {
  language: Language!
  vatNo: String!
```

```
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
}

input InvoiceInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  invoiceNo: String!
  costCenter: costCenterInput
  customerVatNo: String!
  fdmRefs: [FdmReferenceInput!]!
}

input SaleInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  fdmRef: FdmReferenceInput
  costCenter: CostCenterInput
  transaction: TransactionInput!
  financials: [PaymentLineInput!]!
}

input CostCenterChangeInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
```

```
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  transfer: TransferInput!
}

input OrderInput  {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  costCenter: CostCenterInput
  transaction: TransactionInput!

input PreBillInput  {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  costCenter: CostCenterInput
  transaction: TransactionInput!
}

input DrawerOpenInput  {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  drawer: DrawerInput
```

```
}

input PaymentCorrectionInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  financials: [PaymentLineInput!]!
  fdmRef: FdmReferenceInput!
}

input CopyInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  fdmRef: FdmReferenceInput!
}

input ReportTurnoverXInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  reportBookingDate: String!
  posDevices: [PosDeviceInput!]!
  fdmDevices: [FdmDeviceInput!]!
  turnover: TurnoverInput!
}

input PosDeviceInput {
  posId: String!
```

```
  terminalId: String
  firstPosDateTime: String!
  lastPosDateTime: String!
}

input FdmDeviceInput  {
  fdmId: String!
  firstFdmDateTime: String!
  firstTotalCounter: Int!
  lastFdmDateTime: String!
  lastTotalCounter: Int!
}

input TurnoverInput  {
  transactions: [EventTotalInput!]!
  departments: [DeparmentTotalInput!]!
  vats: [VatTotalInput!]!
  payments: [PaymentTotalInput!]!
  drawersOpenCount: Int! = 0
  negQuantities: [NegQuantityTotalInput!]!
  priceChanges: [PriceChangeTotalInput!]!
  invoices: [InvoiceTotalInput!]
}

input EventTotalInput  {
  eventLabel: EventLabel!
  ticketCount: Int!
  amount: Float!
}

input DepartmentTotalInput  {
  departmentId: String!
  departmentName: String!
  amount: Float!
}

input VatTotalInput  {
  label: VatLabel!
  rate: Float!
  taxableAmount: Float!
  vatAmount: Float!
  totalAmount: Float!
}

input PaymentTotalInput  {
  id: String!
  name: String!
  type: PaymentType!
  amount: [PaymentTotalAmountInput!]!
}

input PaymentTotalAmountInput {
  type: PaymentLineType!
  normalAmount: Float!
  negativeCorrections: Float!
  positiveCorrections: Float!
  correctionsCount: Int!
  totalAmount: Float!
  normalForeign: [ForeignCurrencyInput!]
```

```
  correctionsForeign: [ForeignCurrencyInput!]
}

input NegQuantityTotalInput {
  negQuantityReason: NegQuantityReason!
  negQuantityCount: Int!
  ticketCount: Int!
  amount: Float!
}

input PriceChangeTotalInput {
  id: String!
  name: String!
  type: PriceChangeType!
  amount: [PriceChangeVatTotalInput!]!
}

input PriceChangeVatTotalInput {
  label: VatLabel!
  negative: Float!
  positive: Float!
}

input InvoiceTotalInput {
  invoiceNo: String!
  amount: Float!
}

input ReportTurnoverZInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  reportNo: Int!
  reportBookingDate: String!
  posDevices: [PosDeviceInput]!
  fdmDevices: [FdmDeviceInput]!
  turnover: TurnoverInput!
}

input ReportUserXInput {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String!
  deviceId: String!
```

```
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  reportBookingDate: String!
  posDevices: [PosDeviceInput!]!
  fdmDevices: [FdmDeviceInput!]!
  users: [UserItemInput!]!
}


input UserItemInput  {
  employeeId: String!
  totalAmount: Float!
  firstPosDateTime: String!
  lastPosDateTime: String!
  socialEvents: [InOutItemInput!]!
  payments: [PaymentTotalInput!]!
}

input InOutItemInput  {
  posDateTime: String!
  inOut: InOut!
}

input ReportUserZInput  {
  language: Language!
  vatNo: String!
  estNo: String!
  posId: String!
  posFiscalTicketNo: Int!
  posDateTime: String!
  posSwVersion: String!
  terminalId: String
  deviceId: String!
  bookingPeriodId: String!
  bookingDate: String!
  ticketMedium: TicketMedium!
  employeeId: String!
  reportNo: Int!
  reportBookingDate: String!
  posDevices: [PosDeviceInput!]!
  fdmDevices: [FdmDeviceInput!]!
  users: [UserItemInput!]!
}
```

2.4. SIGNATURE NUMERIQUE

La signature mentionnée au point 4.2 est apposée sur le 'JSON enrichi' décrit plus loin. La signature est apposée à l'aide du certificat privé, après application du recodage JSON également décrit plus loin. Cela permet de garantir la reproductibilité de la signature.

2.4.1. JSON canonique pour le hachage reproductible

La flexibilité offerte par JSON permet d'encoder les mêmes informations de différentes manières. Pour obtenir toujours la même signature pour les mêmes données sources, un certain nombre de règles sont appliquées. Ces règles conduisent à un JSON canonique.

Les règles suivantes doivent être strictement appliquées par le FDM aux champs JSON mentionnés plus loin au point 2.4.2.

- Les **nombres** ne peuvent contenir que les caractères 0 à 9, le signe moins (-) et un point décimal. Cela signifie, entre autres, que les valeurs contenues dans les notes scientifiques doivent être recodées sous leur forme la plus élémentaire. Les zéros après une virgule ne sont pas autorisés.
- Tous les **espaces**, comme décrit dans les spécifications JSON, sont supprimés, aucun espace n'est ajouté.
- Les caractères avec un **point de code > U+001F et < U+007F** ne sont **jamais** échappés, à l'exception des points de code **U+0022** (guillemets doubles) et **U+005C** (barre oblique inverse) qui sont toujours échappés conformément aux spécifications JSON. Ces deux-là doivent être échappés avec la note la plus courte possible (\' et \\ respectivement).
- Les caractères avec un **point de code <U+0020 ou > U+007E** sont **toujours** échappés et toujours dans la notation la plus courte possible. Backspace, formfeed, linefeed, carriage return et horizontale tab ont de telles notations courtes (\b, \f, \n, \r et \t respectivement).
- Lors de l'échappement d'un point de code sous la forme d'un ou plusieurs ensembles de 4 caractères hexadécimaux, les caractères hexadécimaux sont toujours en **majuscule** (par exemple : \uAAFF).
- Le décodeur JSON doit respecter l'interprétation **stricte** des spécifications JSON. Par exemple : les littéraux vrai, faux, nul sont toujours en minuscules et sans guillemets, les noms des couples nom/valeur dans les objets doivent toujours être entre guillemets, ...
- Les **paires nom/valeur** de chaque objet en JSON sont **triées**, par ordre croissant, par la valeur numérique des points de code dans le nom. Il est supposé, à titre de bonne pratique, qu'aucun nom double n'est utilisé, car il n'existe pas de norme JSON à cet égard.

2.4.2. Le 'JSON enrichi' dont le contenu est signe numériquement

L'objet JSON ‘enrichi’ (**enrichedEventData**) est constitué des valeurs et objets ci-dessous, selon qu'ils ont été utilisés ou non dans le message.

{
language
posId
vatNo
estNo
terminalId
deviceId
posDateTime
posFiscalTicketNo
ticketMedium
eventOperation
copyOfEvent
employeeId
customerVatNo
invoiceNo
fdmRefs
bookingPeriodId
bookingDate
costCenter
transfer
transaction
vatCalc
financials
drawer
reportNo
reportBookingDate
posDevices
fdmDevices
turnover
users
fdmSwVersion
posSwVersion
bufferCapacityUsed
verificationUrl
fdmId
fdmDateTime
eventLabel
eventCounter
totalCounter
}

2.4.3. Certificat à utiliser pour la signature

La signature doit être apposée à l'aide du certificat matériel dont les détails sont décrits dans la description détaillée du fonctionnement et de la communication entre le module de données fiscales et le service cloud du SPF.