Federale
Overheidsdienst
FINANCIEN

MB 29/04/2024
GEREGISTREERD KASSASYSTEEM – TECHNISCHE VOORSCHRIFTEN –

DETAILBESCHRIJVING VAN DE WERKING VAN DE FISCAL DATA MODULE EN DE COMMUNICATIE MET HET KASSASYSTEEM EN DE FODFIN CLOUDSERVICE

VERSIE 1.2 – 20/09/2024

CHANGELOG

Versie	Datum	Samenvatting
1.0	05/08/2024	Originele tekst
1.1	06/09/2024	**Verduidelijking** - Fysieke verbinding POS - FDM
1.2	20/09/2024	**Correctie:** - Vervanging tabel in hoofdstuk 3, punt 3.3

INLEIDING

De detailbeschrijvingen geven op meer technische en gedetailleerde wijze de technische bepalingen van het ministerieel besluit van 29/04/2024 met betrekking tot de technische aspecten en de certificatie van de Fiscale Data Module (FDM) weer.

Dit document is de technische detailbeschrijving met betrekking tot de werking van de FDM, de communicatie met een gecertificeerd kassasysteem en de communicatie met de cloudservice van de FOD Financiën.

Voor de werking van een gecertificeerd kassasysteem wordt verwezen naar de desbetreffende detailbeschrijving.

Dit document kan kleine wijzigingen ondergaan, wegens eventuele reglementaire beslissingen, voor het rechtzetten van eventuele fouten.

Bijkomende informatie kan ingewonnen worden bij de bevoegde dienst, NCO afdeling GKS via secr.gksce@minfin.fed.be .

GEBRUIKTE AFKORTINGEN

AES	Advanced Encryption Standard
CBC	Cipher Block Chain
ECDSA	Elliptic Curve Digital Signature Algorithm
FDM	Fiscal Data Module
GKS	Geregistreerd KassaSysteem
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON web token
LAN	Local Area Network
(m)TLS	(mutual) Transport Layer Security
NOP	No Operation
NTP	Network Time Protocol
PEM	Privacy Enhanced Mail
PKCS	Public Key Cryptography Standards
POS	Point Of Sale
QR code	Quick Response code
RTC	Real Time Clock
SECP256r1	Specific Elliptic Curve Based on Prime Fields
TCP/IP	Transmission Control Protocol/Internet Protocol
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format

HOOFDSTUK 1 – TECHNISCHE EISEN FISCAL DATA MODULE

Referentie MB: Titel II, Hoofdstuk 2

1.1. ALGEMENE VOORSCHRIFTEN

Principieel mag een FDM geen andere functionaliteiten bevatten dan deze beschreven in het MB. Hetzelfde geldt voor de communicatie tussen POS en FDM en tussen FDM en POS.

Indien een producent extra functionaliteit en/of communicatie wenst bij te voegen dan dient deze:

- bij te dragen tot de goede (en bijgevolg verbeterde) werking van het toestel;
- uitgebreid gedocumenteerd te worden bij de certificatieaanvraag.

Eventuele extra functionaliteit en/of communicatie moet steeds ondergeschikt zijn aan de functionaliteiten en communicatie zowel met POS als met de cloudservice van de FOD Financiën.

Het ontvangen en versturen van berichten en de behandeling ervan mag nooit voor deze extra functionaliteit worden onderbroken.

1.1.1. TELLERS

De in art. 64 vermelde tellers starten opnieuw bij 1 indien zij hun maximum waarde van 999999999 hebben bereikt.

1.1.2. BEREKENING BTW EN MAATSTAF VAN HEFFING

Voorbeeld van de berekening van de maatstaf van heffing en van het btw bedrag (zoals beschreven in art. 66):

								btw bedrag	btw bedrag	maatstaf
aantal	omschrijving	bedrag	btw code		btw code	totaal		niet afgerond	afgerond	van heffing
2	Food	41,50	B		A	10,00		1,73553719	1,74	8,26
2	Drinks	10,00	A		B	54,25		5,8125	5,81	48,44
1	Food	12,75	B		C	13,00		0,735849057	0,74	12,26
1	Take-away	13,00	C		D	15,00		0	0,00	15,00
5	Caution	2,00	X		X	2,00		0	0,00	2,00
1	Tabac	15,00	D							
		94,25				94,25			8,28	

Belangrijke opmerking: btw-tarieven kunnen wijzigen. Via de cloudservice van de FODFIN zal de FDM steeds ruim op tijd de nieuwe tarieven en hun geldigheidsperiode ontvangen. Indien een bookingPeriod over twee kalenderdagen heen loopt (bijvoorbeeld van 30 juni 2024 16:00 tot en met 1 juli 2024 12:00) zal de FDM steeds de btw tarieven hanteren die van kracht zijn/waren op de bookingDate die de POS heeft doorgegeven. Indien in bovenstaand voorbeeld het btw tarief van code B op 1 juli 2024 zou wijzigen naar 9 %, dan moet de FDM toch de berekening uitvoeren met het tarief dat van kracht is op 30 juni 2024 (bookingDate).

1.1.3. BUFFERING EN AANPASBAARHEID INSTELLINGEN

Hiervoor wordt verwezen naar het communicatieprotocol in hoofdstuk 2.

1.1.4. CE-MARKERING

De fabrikant moet ervoor zorgen dat zijn producten voldoen aan de eisen van alle geldende wetgeving. Hij voert een conformiteitsbeoordeling uit en schakelt daarbij – als de EU-wetgeving dat eist – een aangemelde instantie in.

Verder houdt de fabrikant een technisch dossier bij van zijn producten en toont dat op verzoek aan de marktautoriteiten van het land waar de producten worden verhandeld. In een verklaring van overeenstemming verklaart de fabrikant dat zijn producten voldoen aan alle geldende wetgeving. Tot slot brengt de fabrikant het CE-merkteken aan op het product.

Meer info op: Veelgestelde vragen over CE-markering | FOD Economie (fgov.be)

1.2.HARDWARE EISEN

Referentie MB: Titel II, Hoofdstuk 2, afdeling 2.

1.2.1 OPSLAG

De opslag bedoeld in art. 76 en art. 81 zijn niet dezelfde. Er dient effectief minimaal 2 GB capaciteit te zijn om transactiegegevens (enriched JSON) te kunnen opslaan (bufferen).

Deze staat volledig los van het interne geheugen, zoals beschreven in art. 76.

1.3. USER INTERFACE

Referentie MB: Titel II, Hoofdstuk 2, afdeling 3.

Een minimum aantal gezondheidsindicatoren moet aanwezig zijn **op het toestel** zelf. Deze staan beschreven in art. 83 van het MB. De fabrikant heeft hierbij de keuze tussen een display of led.

Er dient een gedetailleerde gebruikershandleiding/troubleshooting sheet beschikbaar te zijn voor de eindgebruiker, installateur en de bevoegde dienst bij de FOD Financiën.

De toegang tot de backoffice van de FDM moet voldoende beschermd zijn. De producent documenteert dit uitgebreid bij zijn certificatieaanvraag.

Via deze user interface moet de volledige configuratie van de netwerkverbinding met de POS en die met de cloudservice van de FOD Financiën mogelijk zijn.

Artikel 85 van het MB laat de producent de vrije keuze tussen:

- user interface op het toestel zelf (display);
- via een geconnecteerd toestel (laptop, tablet, …);

- via de aangesloten POS (scherm);
- een combinatie van bovenstaande.

De gebruikers- en/of installatiehandleiding beschrijft gedetailleerd de procedures. Deze handleiding wordt mee overhandigd tijdens de certificatieprocedure.

1.4. FIRMWARE

De firmware draagt steeds een versienummer. Het certificaat van conformiteit vermeldt steeds dit versienummer.

De firmware moet in staat zijn **alle** functionaliteiten gevraagd in het MB en in deze detailbeschrijving te ondersteunen.

De firmware zal bijgevolg:

- de correctheid (volgens het JSON en GraphQL schema) van de ontvangen berichten van de POS controleren; niet-correcte berichten worden niet behandeld (digitale handtekening, berekeningen, verhogen van tellers, …). De POS wordt hiervan in kennis gesteld via de errorcodes in het antwoord;
- bij correcte berichten de nodige berekeningen, verhoging van tellers, omzettingen, plaatsen van digitale handtekening uitvoeren;
- de inhoud van de ondertekende berichten stockeren en klaarmaken voor verzending naar de FODFIN cloudservice;
- de transactiegegevens volgens de verder beschreven procedures doorsturen naar de FODFIN cloudservice;
- in staat zijn om de berichten, zoals voorzien in het verder beschreven protocol, van de FODFIN cloudservice te ontvangen, correct te interpreteren en het voorziene gevolg eraan te geven.

Het staat FDM producenten vrij om meer functionaliteiten in te bouwen. Deze mogen evenwel geen inbreuk doen op de goede fiscale werking van het toestel en moeten bijdragen aan de goede werking/gebruiksvriendelijkheid van het toestel. Ze worden gedetailleerd beschreven in de meegeleverde documentatie.

De firmware kan noodzakelijke of opgelegde upgrades/patches krijgen. Na goedkeuring door de bevoegde dienst bij de FOD Financiën gebeurt de installatie op de toestellen door de producent of zijn gedelegeerde, op een beveiligde manier. Deze wordt uitvoering beschreven in de documentatie die bij de certificatieaanvraag wordt meegeleverd en vormt een onderdeel van het certificatieproces.

HOOFDSTUK 2 – COMMUNICATIE MET HET AANGESLOTEN KASSASYSTEEM

2.1. FYSIEKE VERBINDING

Kassa en FDM worden met elkaar verbonden met een kabel (LAN) of draadloos (WiFi). De configuratie hiervan gebeurt in de backoffice van zowel kassa als FDM.

BELANGRIJK: In theorie kan de draadloze verbinding ook via het internet verlopen. Bij wegvallen van het internet valt dan automatisch ook de verbinding kassa-FDM weg en stopt de kassa met registreren.

De FDM producent voorziet voldoende mogelijkheden om een goede beveiliging van de verbinding kassa – FDM mogelijk te maken. Typische voorbeelden zijn: HTTP met token, HTTPS met TLS met self signed certificaat, HTTPS met mTLS met self signed certificaat gebaseerd op het intermediate certificaat van de FDM producent.

De eindverantwoordelijkheid voor de correcte en beveiligde installatie en configuratie ligt bij de verdeler/installateur.

Kassa en FDM gebruiken het TCP/IP protocol voor het uitwisselen van hun boodschappen. Deze boodschappen worden hieronder verder beschreven. Andere boodschappen zijn enkel toegelaten voor zover zij de verplichte functionaliteiten niet beïnvloeden en bijdragen tot een goede werking van het geheel. De FDM stelt zijn GraphQL service aan de POS beschikbaar via http(s).

De FDM is via het internet verbonden met de FODFIN cloud API en ontvangt via deze weg ook regelmatig instructies. In bepaalde gevallen, zie betreffende detailbeschrijving, kan dergelijke instructie leiden tot het tonen van een boodschap op het kassasysteem en/of een manuele interventie via het kassasysteem.

De FDM mag niet verbonden worden met toepassingen van derde partijen. Enkel de POS toepassingen, de FODFIN toepassingen en de toepassingen van de FDM-producent mogen verbonden worden met de FDM.

2.2 GRAPHQL SERVICE

Het uitwisselen van berichten tussen POS en FDM gebeurt via de GraphQL service van de FDM.

De GraphQL standaard van 27 oktober 2021 wordt hierbij volledig gevolgd.

Meer informatie kan gevonden worden op: Release October 2021 · graphql/graphql-spec · GitHub.

De requests worden door de kassa naar de GraphQL service van de FDM gestuurd. Voor het versturen van de event gegevens wordt JSON data als payload gebruikt. Deze payload wordt naar het pad **"/graphql"** op de FDM verstuurd met HTTP verb "POST" en Content-Type: application/json.

Alle JSON objecten volgen de RFC 8259 standaard.

String velden bevatten **nooit** voorloop- of volgwitruimte.

De verstuurde gegevens omvatten onder andere:

- transactiegegevens (events P en N);
- financiële gegevens (event F);
- sociale gegevens (event S);
- kopieën (event C);
- training events (event T);

- facturen (event I);
- rapporten (event R).

Deze objecten en hun voorwaarden worden verder bij het volledige GraphQL schema beschreven.

2.2.1. ALGEMEEN

In het ministerieel besluit werden de volgende events bepaald:

- 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).

Deze events worden allemaal naar de FDM gestuurd, om – als beveiliging tegen data manipulatie – digitaal ondertekend te worden. Hiervoor worden er diverse mutaties beschikbaar gesteld door de GraphQL service van de FDM. Deze mutaties kunnen gebruikt worden om de hieronder beschreven JSON structuur van de registraties op de FDM correct op te vullen. De JSON structuur van de FDM kan elk type event bevatten; via de verschillende mutaties aanvaardt de FDM voor elke soort event enkel die velden en objecten die toegelaten zijn.

2.2.2. VERPLICHT TE GEBRUIKEN MUTATIES

Volgende mutaties worden gedefinieerd:

Voor het event S:

- signWorkIn
- signWorkOut

Voor het event I:

- signInvoice

Voor het event N:

- signSale

Voor het event P:

- signCostCenterChange
- signOrder
- signPreBill

Voor het event F:

- signMoneyInOut
- signDrawerOpen
- signPaymentCorrection

Voor het event R:

- signReportTurnoverX
- signReportTurnoverZ
- signReportUserX
- signReportUserZ

Voor het event C:

- signCopy

2.2.3. INHOUD VAN DE COMMUNICATIE TUSSEN POS EN FDM

Hiervoor wordt verwezen naar de DETAILBESCHRIJVING POS, die eveneens ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

Communicatieregels

Om een correcte afhandeling van de requests te kunnen garanderen en deze uniek te kunnen onderscheiden worden volgende sleutelvelden bepaald:

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

Wanneer een POS binnen een tijdspanne van 10 minuten een request verstuurt naar de FDM, waarbij deze 5 sleutelvelden identiek zijn aan deze verstuurd in een vorig bericht, dan moet de FDM verifiëren of dit een request betreft waarop al eerder werd geantwoord (inclusief het verhogen van de betrokken tellers op de FDM).

Volgende situaties kunnen zich voordoen:

1. De combinatie van sleutelvelden is verschillend: de FDM behandelt dit request op de normale manier.
2. De inhoud van het JSON request is volledig identiek: de FDM behandelt dit request als een “dubbele” zending en stuurt in het response dezelfde gegevens terug als in het oorspronkelijke request; de FDM verhoogt zijn tellers ook niet; deze dubbele transactie wordt niet bijgehouden op de FDM.
3. De combinatie van sleutelvelden is identiek, maar de andere waarden in het JSON request zijn verschillend dan die in het eerder verwerkte request: de FDM weigert dit request te behandelen en stuurt een errorcode mee in zijn antwoord.

2.2.4. INHOUD VAN DE COMMUNICATIE FDM ➔ POS

Hiervoor wordt verwezen naar de DETAILBESCHRIJVING POS, die eveneens ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

2.2.5. FDM ➔ POS ERRORHANDLING

Hiervoor wordt verwezen naar de DETAILBESCHRIJVING POS, die eveneens ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

Algemene opmerking omtrent de behandeling van errors.

Bepaalde fouten en waarschuwingen worden getoond aan de gebruiker via de user interface van de POS. De FDM producent voorziet, bijkomend aan de bepalingen hierboven, de nodige bijkomende informatie voor de gebruiker zodat die gepast kan handelen om de fout op te lossen.

Dit wordt uitvoerig beschreven in de bij de certificatieaanvraag meegeleverde documentatie.

2.3. GRAPHQL SCHEMA VOOR DE COMMUNICATIE TUSSEN POS EN FDM

Hiervoor wordt verwezen naar de DETAILBESCHRIJVING POS, die eveneens ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

2.4. DIGITALE HANDTEKENING

Hiervoor wordt verwezen naar de DETAILBESCHRIJVING POS, die eveneens ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

HOOFDSTUK 3 – VERIFICATION URL – QR-CODE

De POS moet op zijn btw-kasticket een QR-code afdrukken. De FDM maakt hiervoor de url aan. Deze url bestaat uit twee grote onderdelen:

- de vaste prefix: https://www.gs2.be/
- de gecodeerde waarde.

De prefix is vast en wordt bepaald door de FOD Financiën.

De gecodeerde waarde wordt volledig berekend door de FDM en is uniek voor elk event N. Deze waarde bevat:

- Een indicatie omtrent het aantal bits van de digitalSignature dat in de berekening werd meegenomen;
- De bits van de digitalSignature (met een maximum van 5 bytes);
- Het fdmId;
- De totaalteller van de events;
- Geen of meerdere padding bits.

Voorbeeld:

-

-
- Prefix: gecodeerde waarde:
- https://www.gs2.be/ viXnmc1vvqGaMAS8MJV
-
-

3.1. VASTE LENGTE

De gecodeerde waarde is **altijd** 19 base-62 karakters lang.

Aangezien base-62 characters vijf of zes bits kunnen zijn is het mogelijk dat padding noodzakelijk is. Anderzijds is het ook mogelijk dat de gecodeerde waarde meer dan 19 karakters zou worden. In dat geval dient de hoeveelheid data in de gecodeerde waarde te worden verminderd.

Standaard zal de encoder 40 bits van de digitalSignature meenemen; indien nodig, kan dit verminderd worden tot 36, 32 of 28 bits, met dien verstande dat de encoder steeds het maximaal mogelijk aantal bits van de digitalSignature moet meenemen.

3.2. TE GEBRUIKEN BASE-62 CHARACTER SET

Decimal	Binary	Base62
0	00000	0
1	11111	1
2	000010	2
3	000011	3
4	000100	4
5	000101	5
6	000110	6
7	000111	7
8	001000	8
9	001001	9
10	001010	A
11	001011	B
12	001100	C
13	001101	D
14	001110	E
15	001111	F

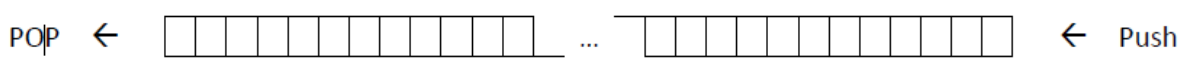
Decimal	Binary	Base62
16	010000	G
17	010001	H
18	010010	I
19	010011	J
20	010100	K
21	010101	L
22	010110	M
23	010111	N
24	011000	O
25	011001	P
26	011010	Q
27	011011	R
28	011100	S
29	011101	T
30	011110	U
31	011111	V

Decimal	Binary	Base62
32	100000	W
33	100001	X
34	100010	Y
35	100011	Z
36	100100	a
37	100101	b
38	100110	c
39	100111	d
40	101000	e
41	101001	f
42	101010	g
43	101011	h
44	101100	i
45	101101	j
46	101110	k
47	101111	l

Decimal	Binary	Base62
48	110000	m
49	110001	n
50	110010	o
51	110011	p
52	110100	q
53	110101	r
54	110110	s
55	110111	t
56	111000	u
57	111001	v
58	111010	w
59	111011	x
60	111100	y
61	111101	z

3.3. BIT STREAM

Zowel de encoder als de decoder routines gebruiken een 64-bit buffer die handelt als een bit stream. Bits worden in de stream geduwd van de minst significante tot de meest significante, en worden door de stream behandeld van de meest significante tot de minst significante.



De langste waarde die in de stream moet worden geduwd is 41 bits lang. De bitstream verwerking start pas nadat elk veld erin werd geduwd (om een overflow te vermijden). De verwerkte bits worden onmiddellijk uit de stream verwijderd.

Een teller houdt continu bij hoeveel bits in de stream werden geduwd. Het aantal behandelde bits doet de teller telkens afnemen.

Van zodra de vijf oudste bits 00000 of 11111 voorstellen worden deze verwerkt als een vijf bit eenheid, alle andere waarden worden als zes bit eenheid verwerkt.

De encoder duwt de bit voorstelling van de originele data in de bit stream en gebruikt hiervoor de tabel uit sectie 3.2 om het correcte base-62 karakter bepalen voor de bits die hij behandelt.

De decoder gebruikt dan weer het base-62 karakter om te bepalen hoeveel bits en in welk patroon hij die in de stroom moet duwen en reconstrueert zo de origineel gecodeerde data uit de bits die hij verwerkt.

3.4. XOR MASK

Om een goede verdeling van de gecodeerde waarden te bekomen wordt een XOR masker gebruikt op de waarden van fdmId en totalCounter.

Zonder dit masker zouden de steeds terugkerende waarden van fdmId en de oplopende totalCounter een herkenbaar patroon kunnen vormen in de gecodeerde waarden.

Het masker wordt gegenereerd op basis van de eerste twee bytes van de digitalSignature (hierna B1 en B2 genoemd.

set mask equal to B2
shift the bits in the mask to the left by 8 positions
add B1 to mask
shift the bits in the mask to the left by 8 positions
add the inverse of B1 to mask
shift the bits in the mask to the left by 8 positions
add the inverse of B2 to mask
shift the bits in the mask to the left by 8 positions
add B1 to mask
shift the bits in the mask to the left by 8 positions
add B2 to mask

Van het bekomen 48 bit masker worden de minst significante 41 bits genomen als masker voor het fdmId en van de inverse van dit 48 bit masker worden de minst significante 30 bits gebruikt als masker voor de totalCounter.

3.5. DIGITALSIGNATURE

Tot vijf bytes van de digitalSignature worden opgenomen in de gecodeerde waarde. Byte één, twee en drie van de digitalSignature worden **altijd** opgenomen. Byte vier en vijf kunnen geheel of gedeeltelijk weggelaten worden indien de gecodeerde waarde langer zou worden dan 19 base 62-karakters.

Deze worden meegenomen of weggelaten als **halve bytes**, gebaseerd op de sequentie 40, 36, 32, 28 zoals hieronder beschreven.

B1	B2	B3	B4		B5	
Always	**Always**	**Always**	**Always (28)**	**>= 32**	**>= 36**	**= 40**
0x00 – 0xFF	**0x00 – 0xFF**	**0x00 – 0xFF**	**0x0 – 0xF**	**0x0 – 0xF**	**0x0 – 0xF**	**0x0 – 0xF**

3.6. SIGNATURE BIT TELLER EN SIGNATURE BITS

De signature bit teller is het **eerste veld** dat in de bit stream wordt geduwd.

Deze teller bevat het aantal bits waaruit de digitale handtekening bestaat.

Dit veld bevat steeds **twee** bits.

1	**1**	**40 bits**
1	**0**	**36 bits**
0	**1**	**32 bits**
0	**0**	**28 bits**

Het signature bit teller veld wordt onmiddellijk gevolgd door het betrokken aantal signature bits (**tweede veld**).

3.7. FDMID

Het **derde veld** dat in de bit stream moet worden geduwd is het fdmId. Dit is een string die bestaat uit drie karakters uit de range A tot en met Z, gevolgd door acht digits uit de range 0 tot en met 9.

De drie karakters worden geconverteerd naar integer waarden (c1, c2, c3) uit de range van 0 tot en met 25, waarbij A = 0, B = 1, C = 2, en zo verder.

De acht digits uit de string worden getransformeerd naar een integer waarde (ms). Deze vier waardes worden tot slot gecombineerd naar één 41 bit integer waarde door onderstaande formule te gebruiken:

v = ((((((ms * 26) + c3) * 26) + c2) * 26) + c1)

Vooraleer deze waarde in de bit stream wordt geduwd, wordt eerst het 41 bit XOR masker erop toegepast.

3.8. TOTALCOUNTER

Het **vierde veld** dat in de bit stream wordt geduwd is de totalCounter. Dit is een 30 bit integer waarde.

Vooraleer deze waarde in de bit stream wordt geduwd, wordt eerst het 41 bit XOR masker erop toegepast.

3.9. PADDING (OPVULLING)

De eventueel toe te passen padding heeft twee grote doelstellingen:

- Het garandeert een goede werking van de bit stream en dat alle bits gecodeerd worden als base 62 values;
- Het garandeert dat het finale resultaat 19 base 62 karakters lang is.

Padding bits hebben altijd de waarde 0. Non-zero padding is niet toegestaan en gecodeerde waardes die dergelijke waardes bevatten moeten als invalid beschouwd worden.

3.10. REFERENTIEWAARDEN

Onderstaande waarden kunnen als referentie worden gebruikt om de encoding en decoding implementaties te verifiëren.

FDM serial	TotalCounter	Original signature bytes (hex)					Encoded Signature bits	
FDM01000001	1	9B	21	C7	09	BF	viXnmc1vvqGaMAS8MJV	40
FDM01000001	2	26	8D	D5	E5	F4	oQDrUNqaEVtV0cdZTbm	40
FDM01000001	3	B4	8B	E6	23	67	xIBvYDdswxqDH0rYqjt	40
FDM01000001	999999997	F8	4B	C9	FF	27	lXBoVylc8b0cu0MZd98	36
FDM01000001	999999998	20	26	5B	00	3C	Y0JBO0CDvAB1ergy5YS	36
FDM01000001	999999999	75	E1	F9	FD	97	tNX1FxBt2UWgjADUqFX	40
AAA01000001	1	42	A6	98	AD	4A	qAccArA8jNY1mE3fhrO	40
AAA01000001	999999999	B1	5E	58	9A	4C	x5UM9fCMYqUX1s9n8TU	40
AAA99999999	1	5B	4C	86	12	4C	rjCXX9CuSrM70KSJAIo	40
AAA99999999	999999999	F7	40	CD	C4	62	lT0Pk8rlFg2K60UrWe0	36
ZZZ01000001	1	96	E1	D0	9B	C5	vRXq9l5oB1WvMkNuMaV	40
ZZZ01000001	999999999	74	C6	C6	CC	49	tJ6nin9EyxIEyISk8MC	40
ZZZ99999999	1	45	94	D8	4D	B1	qMKs4snxi0TE9h6bBfg	40
ZZZ99999999	999999999	0D	F3	38	D8	B5	mtpEDYrIkG9rGPgqdVc	40
XYZ98123456	33751	99	F2	2F	B2	CB	vdoBxBBXFSOzRorvSkq	40
XYZ98123456	33766	D4	A0	48	B4	45	zIWIBH5eL2nqnMhGL5o	40
XYZ98123456	33788	35	13	34	9A	01	pKJD9e0gL2q4UOs4qaG	40

3.11. ENCODER

Deze implementatie gaat er vanuit dat alle inputs geldige waarden zijn.

```
public static byte[] Encode(string fdmSerial, long totalCounter, byte[] digitalSignature)
{
        // State machine states:
        // 0. initialize
        // 1. push signature bitcount modifier
        // 2. push signature bits, apply bitcount modifier
        // 3. push FDM manufacturer, model, and serial number
        // 4. push TotalCounter
        // 5. flush the stream by adding padding if need be
        // 6. verify length of the output (return the result)

        byte[] bytes = new byte[19];
        long bitStream = 0;
        long mask = digitalSignature[1];
        mask = (mask << 8) + digitalSignature[0];
        mask = (mask << 8) + ~digitalSignature[0];
        mask = (mask << 8) + ~digitalSignature[1];
        mask = (mask << 8) + digitalSignature[0];
        mask = (mask << 8) + digitalSignature[1];
        for (int state = 0, signatureBitCountModifier = 4, byteCount = 0, streamLen = 0; state < 7 &&
        signatureBitCountModifier > 0; state++)
        {
                switch (state)
                {
```

```
case 0:
        bitStream = 0;
        streamLen = 0;
        byteCount = 0;
        break;

case 1:
        bitStream = (bitStream << 2) | (byte)(signatureBitCountModifier - 1);
        streamLen += 2;
        break;

case 2:
        bitStream = (bitStream << 8) | digitalSignature[0];
        bitStream = (bitStream << 8) | digitalSignature[1];
        bitStream = (bitStream << 8) | digitalSignature[2];
        bitStream = (bitStream << 8) | digitalSignature[3];
        bitStream = (bitStream << 8) | digitalSignature[4];
        streamLen += 40;

        int undo = (4 * (4 - signatureBitCountModifier));
        bitStream = (bitStream >> undo);
        streamLen -= undo;
        break;

case 3:
        int c1 = fdmSerial[0] - 65;
        int c2 = fdmSerial[1] - 65;
        int c3 = fdmSerial[2] - 65;
        long ms = int.Parse(fdmSerial.Substring(3, 8));
        long mask41 = mask & (((long)1 << 41) - 1);
        long v = ((((((ms * 26) + c3) * 26) + c2) * 26) + c1) ^ mask41;
        bitStream = (bitStream << 41) | v;
        streamLen += 41;
        break;

case 4:
        long mask30 = (~mask) & (((long)1 << 30) - 1);
        bitStream = (bitStream << 30) | (totalCounter ^ mask30);
        streamLen += 30;
        break;

case 5:
        if (0 != streamLen)
        {
                int n = 6 - streamLen;
                bitStream = bitStream << n;
                streamLen += n;
        }
        break;

case 6:
        while (bytes.Length > byteCount)
```

```
                                bytes[byteCount++] = 48; /* padding with "0" */
                                return bytes;
        }
        /* try to read from the bit stream */while (6 <= streamLen)
        {
        if (bytes.Length <= byteCount)
                {
                        /* we are ending up with too many characters, drop half a signature byte and start
                                over */
                        signatureBitCountModifier--;
                        state = -1;
                        break;
                }

                int read = 6;
                int bits = (int)((bitStream >> (streamLen - read)) & 63);
                switch (bits)
                {
                        case 0:
                                read = 5;
                                break;

                        case 1:
                                bits = 0;
                                read = 5;
                                break;

                        case 62:
                        case 63:
                                bits = 1;
                                read = 5;
                                break;
                }
                bytes[byteCount++] = (byte)(bits + ((10 > bits) ? 48 : (35 < bits) ?
                        61 : 55));
                streamLen -= read;
                bitStream &= (((long)1) << streamLen) - 1;
        }
}
/* if we got here the encoding failed */return null;
}
```

3.12. DECODER

Deze implementatie gaat er vanuit dat de parameter bytes een array is met een lengte gelijk of groter dan 19.

```
public static bool Decode(byte[] bytes)
{
        // State machine states:
        // 0. get signatureBitCount
```

```
// 1. get signature byte #1
// 2. get signature byte #2
// 3. get signature byte #3
// 4. get signature byte #4 MSB (according to the signatureBitCount)
// 5. get signature byte #4 LSB (according to the signatureBitCount)
// 6. get signature byte #5 MSB (according to the signatureBitCount)
// 7. get signature byte #5 LSB (according to the signatureBitCount)
// 8. get FDM manufacturer, model, and serial number
// 9. get TotalCounter
// 10. verify padding (must be zeroes)

int state = 0;
long bitStream = 0;
int streamLen = 0;
int nextByteIndex = 0;
long mask = 0;

/* output variables */
StringBuilder fdm = new StringBuilder();
long totalCounter = 0;
byte[] signatureBytes = new byte[5];
int signatureBitCount = 40;

while (state <= 10)
{
        if (nextByteIndex < bytes.Length)
        {
                int b = bytes[nextByteIndex];
bitStream &= (((long)1) << streamLen) - 1;

/* convert the byte to the range (-1, 63), where -1 indicates illegal character */
b -= (48 <= b && 57 >= b) ? 48 : (65 <= b && 90 >= b) ?
      55 : (97 <= b && 122 >= b) ? 61 : b + 1;

switch (b)
{
        case -1:
                return false; /* invalid character in input */

        case 0:
                bitStream = (bitStream << 5);
                streamLen += 5;
                break;

        case 1:
                bitStream = (bitStream << 5) | 31;
                streamLen += 5;
                break;

                default:
                        bitStream = (bitStream << 6) | (byte)b;
                        streamLen += 6;
```

```
                        break;
                }
                nextByteIndex++;
        }
        else if (nextByteIndex++ > bytes.Length)
                return false; /* we ran out of characters but the state machine is still waiting for more bits */

        switch (state)
        {
                case 0:
                        if (2 <= streamLen)
                        {
                                int modifier = (int)(((bitStream >> (streamLen - 2)) & 3) + 1);
                                signatureBitCount -= (4 * (4 - modifier));
                                streamLen -= 2;
                                state++;
                        }

                        break;
                case 1:
                case 2:
                case 3:
                        if (8 <= streamLen)
                        {
                                signatureBytes[state - 1] = (byte)(bitStream >> (streamLen - 8));
                                streamLen -= 8;
                                state++;
                        }
                        break;

                case 4:
                        if (4 <= streamLen)
                        {
                                if (28 <= signatureBitCount)
                                {
                                        signatureBytes[state - 1] |=
                                                (byte)(((bitStream >> (streamLen - 4)) & 15) << 4);
                                        streamLen -= 4;
                                }
                                state++;
                        }
                        break;

                case 5:
                        if (4 <= streamLen)
                        {
                                if (32 <= signatureBitCount)
                                {
                                        signatureBytes[state - 2] |=
                                                (byte)((bitStream >> (streamLen - 4)) & 15);
                                        streamLen -= 4;
                                }
```

```
                state++;
            }
            break;

        case 6:
            if (4 <= streamLen)
                {
                    if (36 <= signatureBitCount)
                    {
                        signatureBytes[state - 2] |=
                            (byte)(((bitStream >> (streamLen - 4)) & 15) << 4);
                        streamLen -= 4;
                    }
                    state++;
                }
                break;

        case 7:
            if (4 <= streamLen)
            {
                if (40 <= signatureBitCount)
                {
                    signatureBytes[state - 3] |=
                        (byte)((bitStream >> (streamLen - 4)) & 15);
                    streamLen -= 4;
                }
                state++;
            }
            break;

        case 8:
            if (41 <= streamLen)
            {
                mask = signatureBytes[1];
                mask = (mask << 8) + signatureBytes[0];
                mask = (mask << 8) + ~signatureBytes[0];
                mask = (mask << 8) + ~signatureBytes[1];
                mask = (mask << 8) + signatureBytes[0];
                mask = (mask << 8) + signatureBytes[1];

                long mask41 = mask & ((long)1 << 41) - 1;
                long v = ((bitStream >> (streamLen - 41)) & 0x1FFFFFFFFF) ^ mask41;
                fdm.Append((char)((v % 26) + 65));
                fdm.Append((char)(((v / 26 % 26)) + 65));
                fdm.Append((char)((v / 676 % 26) + 65));
                fdm.Append((v / 17576).ToString("00000000"));
                streamLen -= 41;
                state++;
            }
            break;

        case 9:
```

```
                if (30 <= streamLen)
                {
                        long mask30 = (~mask) & ((long)1 << 30) - 1;
                        totalCounter =
                                ((long)((bitStream >> (streamLen - 30)) & 0x1FFFFFFFF)) ^
                                mask30;
                        streamLen -= 30;
                        state++;
                }
                break;

        case 10:
                if (0 != (bitStream & ((long)1 << streamLen) - 1))
                        return false; /* non-zero padding encountered */
                return true; /* output variables are set */
        }
    }
    return false;
}
```

HOOFDSTUK 4 – TIJDELIJKE OPSLAG VAN DE GEGEVENS

De FDM beschikt over een data storage geheugen van minimaal 2 GB. Dit laat de FDM toe om de enriched JSON berichten te bufferen, vooraleer ze doorgestuurd worden naar de FODFIN cloudservice.

Deze enriched JSON vormen, aangevuld met enkele bijkomende velden (zie verder) de payload van de berichten die naar ws_reg zullen worden gestuurd.
De FDM zorgt ervoor dat deze berichten niet kunnen gewist of gewijzigd worden.
Enkel via de backoffice (of via de ws_query) kunnen opgeslagen berichten gekopieerd/geraadpleegd worden.
Doorgestuurde enriched JSON berichten mogen enkel volgens de bepalingen van hoofdstuk 5 worden verwijderd na verzending.

HOOFDSTUK 5 – COMMUNICATIE FDM ↔ FODFIN CLOUDSERVICE

5.1. VERBINDING EN BEVEILIGING

5.1.1. VERBINDING

De verbinding via het internet wordt geconfigureerd via de backoffice van de FDM. Deze backoffice is producenteigen, waarbij de gebruikersinterface verschillend kan zijn. Minimaal biedt de backoffice de volgende configuratiemogelijkheden aan:

- de url's van de FODFIN cloudservice;
- de url van de time server.

5.1.2. BEVEILIGING

Elke communicatie wordt beveiligd door het gebruik van:

- het authenticatiecertificaat;
- het mTLS protocol dat de zendingen encrypteert.

5.2. FIRST TIME EVER

Conform de bepalingen van het ministerieel besluit van 29/04/2024 worden zowel het kassasysteem als de Fiscal Data Module door de betrokken stakeholders in de diverse fases (productie, levering) geregistreerd in de online GKS e-service van de FOD Financiën.

Hierbij wordt op ondernemings- en vestigingsniveau bepaald welke FDM aan welk kassasysteem is gekoppeld .

Bij de indienstneming van het GKS worden kassa en FDM aan elkaar gekoppeld. Bij een correcte opstart (unieke verantwoordelijkheid van de horecaonderneming en haar leverancier(s)) zal de FDM contact maken met de cloudservice van de FOD Financiën via een beveiligde internetverbinding. De FODFIN cloudservice:

1. verifieert het serienummer van de FDM, kent het unieke en gepersonaliseerde authenticatiecertificaat toe en stuurt het via de beveiligde verbinding naar de FDM die het opslaat op de beveiligde omgeving (art. 77 van dit ministerieel besluit);

stuurt gelijktijdig het initialisatiepakket (overzicht geldende btw-tarieven, gepersonaliseerde frequentiesettings,…) door.De volledige detailbeschrijving is opgenomen in het document FPSFIN_API_PROTOCOL_FDM_FINCLOUD dat ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0 .

5.3. API EN PROTOCOL VOOR COMMUNICATIE MET CLOUD SERVICE FOD FINANCIËN

Referentie MB: Titel II, Hoofdstuk 2, afdeling 4.

De FDM communiceert met de (cloud)servers van de FODFIN via het protocol, beschreven in het document FPSFIN_API_PROTOCOL_FDM_FINCLOUD dat ter beschikking wordt gesteld via www.geregistreerdkassasysteem.be/nl/gks-2-0,. **Geen enkele andere communicatie is toegelaten.**

HOOFDSTUK 6 - ONLINE DOORSTUREN TRANSACTIEGEGEVENS

Afhankelijk van de geconfigureerde zendfrequentie stuurt de FDM pakketten met transactiegegevens van de FDM door naar de FODFIN cloudservice. Dergelijke pakketten bevatten minimaal de inhoud van één volledig event of een NOP.

De FODFIN cloudservice bevestigt (of ontkent) de goede ontvangst van de pakketten. De FDM houdt deze pakketten nog 10 dagen bij, waarna deze mogen verwijderd worden uit de buffer.

Deze pakketten worden door de FODFIN cloudservice minimaal 3 jaar bijgehouden.

Deze bewaring doet geen afbreuk aan de wettelijke fiscale bewaar- en voorleggingsverplichtingen van de originele gegevens op het kassasysteem (artikelen 315bis en 315ter van het WIB92 en artikelen 60 en 63 van het WBTW).

Andere dan in dit ministerieel besluit opgenomen gegevens kunnen en mogen niet via deze verbinding doorgestuurd worden.

6.1. JSON BERICHT NAAR FODFIN

Het JSON bericht dat wordt doorgestuurd naar de servers van de FODFIN via de ws_reg bevat de arrays, objecten en values zoals door het kassasysteem doorgestuurd naar de FDM en deze door de FDM zelf aangemaakt. We noemen dit verder de enriched JSON.

{

unsentEventCount (numeriek, verplicht)

Aantal niet-verzonden berichten in de buffer. Steeds 0 in geval van een No Operation (NOP).

fdmId (string, verplicht)

Het serienummer van de FDM.

events (array, verplicht)

Array van enriched events. Verplicht, in geval van een NOP is de array leeg.

[

{

enrichedEventData (object, verplicht)

Object met event-gegevens van de kassa, aangevuld met berekeningen van de FDM. De velden opgenomen in dit object staan beschreven in de Detailbeschrijving POS.

digitalSignature (string, verplicht)

Digitale handtekening in base64 op basis van de canonical JSON hercodering van `enrichedEventData`

shortSignature (string, conditioneel)

SHA-1 checksum in hex van de digitale handtekening in bytes. Wordt enkel berekend voor het event N.

fdmLocalisation (object, verplicht)

}

In uitzonderlijke gevallen, wanneer fouten optreden bij het uitlezen van een transactie, mag dit object vervangen worden door:

{

rawCounter (numeriek, verplicht)

Deze teller wordt verhoogd elke keer een corrupte transactie wordt doorgestuurd.

rawBytes (string, optioneel)

De ruwe bytes in base64 van de corrupte transactie. Wordt ingevuld wanneer de transactie (deels) leesbaar is.

rawInfo (string, verplicht)

Bijkomende info van de FDM, bijvoorbeeld foutmelding.

}

]

}

fdmLocalisation

Verplicht object in verband met de lokalisatie van de FDM op het moment van de communicatie. Indien deze FDM een 4G/5G-module bevat, worden deze coördinaten verplicht doorgestuurd.

{

geoLocation (object, optioneel)

Bevat lokalisatiegegevens van de optionele GPS-module van de FDM

{

latitude (numeriek, verplicht)

Bevat de breedtegraad gegevens, in decimaal formaat. Min waarde = -90, max. waarde = 90. Formaat = 2,6. Voorbeeld: 50,851415.

longitude (numeriek, verplicht)

Bevat de lengtegraad gegevens, in decimaal formaat. Min waarde = -180, max. waarde = 180. Formaat = 3,6. Voorbeeld: 4,354365.

}

}

}