Optin Sensor Protocol

Kommunikációs protokoll Specifikáció

Szerzők:
Optin Team
hwdev@optin.hu

Ellenőrizte:
Almási Dénes Attila

2015. március 19.
Dokumentum verziószáma: HU-1.1.1042

Tartalomjegyzék

1. Bevezető

Az **Optin Sensor Protocol** egy kis overheaddel működő, payload-agnosztikus kliens-szerver protokoll.

- Egyszerűsége miatt kitűnően alkalmas csekély erőforrással rendelkező eszközökön történő használatra is.
- Számos opcionálisan alkalmazható megoldást biztosít a kritikus csomagok célba érésének garantálására, az elveszett csomagok számának csökkentésére.
- A kevésbé kritikus csomagok kezelését nem terheli a biztonsági megoldások alkalmazásával járó megkötések betartása.

Ez a specifikáció két lényeges fejezetre bontható:

- Általános üzenetformátum, amely minden üzenetre érvényes.
- Üzenettípusok speciális elemei.

Figyelem: A *DATA* típusú üzenetek eszközspecifikus tartalma a mellékletekben található.

2. Szerzői nyilatkozat

Az Optin Kft. (továbbiakban mint Szerző) engedélyezi bárki számára az OSP felhasználását, implementációját, a protokoll-specifikáció másolását vagy közzétételét, azzal a feltétellel, hogy a protokoll-specifikáció MINDEN másolata vagy annak része ami más dokumentációkban, specifikációkban, cikkekben feltűnik, illetve minden rendszer ami implementálja OSP-t, tartalmazza a következőket:

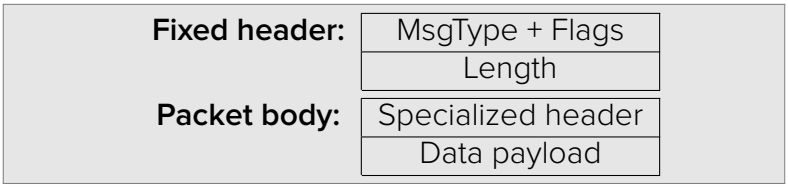
- A Szerző nevét
  (Optin Kft.).
- Egy linket vagy URL-t a OSP specifikációra a Szerző weboldalán.
  www.optin.hu/products/optin-sensor-protocol/

3. A kommunikáció alapjai

A protokollon történő kommunikáció mindegyike packeteken (csomagokon) keresztül történik. A packetek atomiak a protokoll szempontjából. (A legkisebb értelmes, jelentést hordozó egységek) A packetek beágyazhatóak más protokollba, és a packetekbe legfeljebb egy ponton beágyazható tetszőleges másik protokoll. (Lásd később: *DATA* és *COMMAND* packet).
Jelen specifikáció nem definiálja, hogy a packetek egészben, vagy tetszőlegesen feldarabolva továbbítandóak-e.

3.1. Packet struktúra

Minden egyes packet a következő összetevőkből áll: egy állandó fejléc, ezt követően pedig a packet törzse. Az állandó fejléc azonosítja a packet típusát, megadja a hosszát, valamint flageket tartalmaz, amelyek meghatározzák a csomag státuszát. A packet törzs opcionális, tartalmazhat specializált fejlécet, ami egy packetben egyszer szerepel, és adattartalmat ami egy vagy több adatot tartalmazhat.



3.2. Definíciók

3.2.1. In-flight-egyértelműség

Egy X szemantikájú y érték 'in-flight-egyértelmű', ha nincs olyan packet adott pillanatban és adott irányban a kommunikációs csatornán, valamint kommunikációs cachekben, memóriákban, amelynek X értéke szintén y volna.
Erre példa a *MessageID* vagy a *CommandID*. Az in-flight-egyértelmű értékek segítenek a packet egyértelmű azonosításában egy, a fogadó által értelmezett időablakon belül.

3.2.2. Azonosítók

Az azonosítók pozitív számok. A nullás érték minden esetben foglalt, és hibát vagy érvénytelen bejegyzést jelez.

3.2.3. String/binary típusok

String a kommunikációban csak a packetek végén fordulhat elő, hossza korlátlan, és nincs záróbájt. A string végét a packet hossz mezője (*Length*) alapján lehet számítani.
A protokoll szempontjából a stringek és a binary-k azonosak: Tetszőleges hosszúságú bájt-sorozatok.

3.2.4. Kliens és Szerver

A Kliens az, aki az OSP kapcsolatot kezdeményezi, és kapcsolódik a Szerverhez. A protokoll korábbi és jelenlegi, **1.2**-es verziója megköti, hogy DATA típusú packet-et csak a Kliens állíthat össze (mint például szenzor vagy szenzorok halmaza), és küldhet a Szervernek, ami adatgyűjtőként működik.

3.2.5. Byte sorrend

Az OSP-ben, mint a legtöbb hálózati protokollban, big-endian kódolás használatos a több byte-os értékek esetén. Ez azt jelenti, hogy a legnagyobb helyiértékű bájt – angolul ”most significant byte” (rövidítve MSB) – van legelöl és az LSB leghátul. *Ez alól a Length mező kivétel, amelynek speciális kódolása van, lásd lentebb.*

3.3. Garanciák

A specifikáció garantálja a következőket:

3.3.1. MessageID garanciák

Amennyiben a MessageID jelen van egy csomagban:

- Mindig 1 byte hosszú.
- Mindig a Fixed Header utáni első byte.
- A Kliens csak olyan packetekre kapcsolhat (A) AckReq bitet, amely tartalmaz MessageID-t.
- A Szerver visszakérheti az utolsó in-flight K MessageID-hoz tartozó packetet a RESEND packet segítségével. K értékében a felek impliciten megegyeznek. (A szerver pl. ModuleID vagy DeviceType alapján tudhatja)

4. Állandó fejléc (Fixed header)

Minden packet az állandó fejléccel kezdődik, ez legalább 2 byte, legfeljebb 5 byte lehet, függően a változó hossz mezőtől.

A fix fejléc formátuma a következő:

	8-5.bit	**4.bit**	**3.bit**	**2.bit**	**1.bit**
1.byte	MsgType	Cached	Saved	AckReq	CRC
2.byte	Length ([1-4] byte, chaining)				

1. táblázat. A fix fejléc felépítése

1.byte:
Tartalmazza a packet típusát (MsgType) és az állapot flageket (Cached, Saved, AckReq és CRC).

2.byte:
A packet méretét meghatározó mező (legalább 1 byte).

4.1. Message Type (MsgType)

Pozíció: 1. byte, 8-5.bit

A packet típusa, ez egy 4 bites előjel nélküli, felsorolás típusú érték, mely jelentése a 2. táblázat szerinti.

MsgType értéke	Rövidítés	Leírás
0x00	[RESERVED]	*Nem használt érték!*
0x01	CONECT/DISCONNECT	Kapcsolódást vagy kapcsolat bontást kezdeményező packet a Szerver felé
0x02	COMMAND	Parancs típusú packet
0x03	ACKNOWLEDGE	Nyugta típusú packet egy korábbi nyugtázandó packetre
0x04	PINGREQ	Ping kérés
0x05	PINGRESP	Ping válasz
0x06	FIRMWARE	Firmware típusú packet
0x07	RESEND	Újraküldést kérő packet típus
0x08	DATA	Általános adat tartalmú packet
[0x09 - 0x0F]	[RESERVED]	*További fejlesztésre*

2. táblázat. Az MsgType értékek jelentése

4.2. Cached bit (C)

Pozíció: 1. byte, 4. bit

Ha ez a bit **1-es** értékű, akkor olyan csomagról van szó, amit az eszköz legalább egyszer már megpróbált elküldeni. Ez a következő esetekben lehetséges:

- A szerver újrakér egy csomagot a *RESEND* paranccsal.
- A nyugtázandó üzenetre nem érkezett meg időben a nyugta és automatikusan újraküldésre került.
- A Kliens úgy érzékelte, hogy sikertelen volt egy csomag küldése, így újra próbálkozik.

4.3. Saved bit (S)

Pozíció: 1. byte, 3. bit

Ha ez a bit **1-es** értékű, akkor olyan csomagról van szó, amely valamilyen perzisztens tárolóból (belső Flash vagy SD-kártya) került visszaolvasásra. Ez mindig múltbéli adat küldését eredményezi. Felhasználástól függően például a következő esetekben indokolt a bit beállítása:

- Hosszú ideig nem sikerült kapcsolatot létesíteni a szerverrel.
- Roaming módban le volt tiltva, vagy ritkítva az adatküldés, és az eszköz hazai hálózatra kapcsolódott újra.
- Le volt tiltva a mobil adatküldés, majd újra engedélyezve lett.

A fenti, vagy ahhoz hasonló hálózati kimaradások során keletkező csomagok küldésekor a csomagok Saved bitjét be kell állítani.

4.4. AckReq bit (A)

Pozíció: 1. byte, 2.bit

Ha ez a bit **1-es** értékű, akkor olyan csomagról van szó, amit a szervernek nyugtáznia kell. A nyugtázás egy ACKNOWLEDGE típusú csomag küldésével történhet.

4.5. CRC bit (CRC)

Pozíció: 1. byte, 1.bit

Ha ez a bit **1-es** értékű, akkor olyan csomagról van szó, amely a data payload után tartalmaz egy 1 byte hosszú ellenőrző összeget (checksum). Ennek számítási módja lentebb látható, és a data payload mezőre értelmezett.

```
// Pseudo C code

for( index = 0; index < count; ++index )
{
  crc_byte = (crc_byte + payload[index]) % 255;
}
```

4.6. Length mező

Pozíció: 2. byte

A teljes üzenet hossza, beleértve a fix header szekciót is. Ez egy változó hosszúságú mező, amely legalább 1, legfeljebb 4 byte. Az első byte legfelső bitje (8.bit) nem része a reprezentált számnak. Amennyiben ez be van kapcsolva, a következő byte is a length részét képezi, és így tovább. Így minden byte hét bittel járul hozzá a length-hez.

Például: egy 64 byte-os üzenet length mezője egyszerűen egy byte-ból áll, 0x40 értékkel. Ha viszont az üzenet 321 byte hosszú $(2 * 128 + 65)$, akkor a length mező két byte-os lesz. LSB first kódolás szerint az első byte $65 + 128 = 193$ értékű. Itt a legfelső bit azt jelöli, hogy legalább egy byte folytatás van még. A második byte értéke 2.

Mivel a protokoll négy byte-ban maximalizálja a length mezőt, az alkalmazások maximum 268 435 455 byte (256 MB) adatot küldhetnek egy csomagban.

Az 1. ábrán a length mező dekódoló algoritmusa látható. Amikor az algoritmus végez, a *value* változó tartalmazza az üzenet hosszát byte-ban.

```
// Pseudo C code
int multiplier = 1;
int value = 0;
do {
  digit = next_byte_from_stream;
  value += (digit & 127) * multiplier;
  multiplier *= 128;
} while ((digit & 128) != 0)
```

1. ábra. A *Length* mező dekódolásának algoritmusa.

5. Packet típusok

A továbbiakban tömör formában jelezzük a **C** = Cached, **S** = Saved és **A** = AckReq bitek, illetve az **R** = Reserved bit állásait.
Az X szimbólum tetszőleges értéket, a 0 fix nullás, az 1 fix egyes értéket jelent.

Pl.:

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	0	0	1	0	0	0	X
2.byte	Length							

Ez a következőt jelenti:
A MessageType alapján ez egy CONNECT packet, amiben a *Cached*, a *Saved* és az *AckReq* flag is fix 0 értéket kell hogy felvegyen mindig.

A flag bitek szabályainak megsértése a kapcsolat azonnali lezárásával kell, hogy járjon. A három bit mindegyike olyan logikát indíthat be bármelyik félben, amely helytelen működése további kommunikációs hibák sorozatát eredményezné.

Példa erre az AckReq bit felelőtlen használata: ACKNOWLEDGE-ot kérni értelmetlen egy olyan csomagra, amelynek nincsen MessageID-ja.

5.1. CONNECT/DISCONNECT

A kapcsolódási szándék jelzésére, autentikációra és a kapcsolat visszaigazolására vagy elutasítására szolgáló packet.
A fix fejléc azonos, de a packet törzs függ attól, hogy a Kliens vagy a Szerver küldi.
A CONNECT packet nem lehet *Cached*, *Saved* és *AckReq* flaggel jelölt.
Kapcsolódást mindig a Kliens kezdeményez a Szerver felé.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	0	0	1	0	0	0	X
2.byte	Length							

5.1.1. Kliens ⇒ Szerver

A Kliens kapcsolódási szándékot jelez a szerver felé, a TCP csatorna létrejötte utáni első packet CONNECT kell, hogy legyen. Minden további CONNECT-re a szerver kötelessége bontani a kapcsolatot: ez egyben a lekapcsolódási szándék jelzése a Kliens felől.

Packet body:

	Leírás
1.byte	DeviceType (MSB)
2.byte	DeviceType (LSB)
3.byte	ModuleID (MSB)
4.byte	ModuleID (...)
5.byte	ModuleID (...)
6.byte	ModuleID (LSB)
7.byte	ProtocolVersion
8.byte	Password (string)
...	
n.byte	Password (string)

DeviceType: A Kliens típusazonosítója.

Érték	Jelentés
0x0000	Hiba / Invalid
0x0001	IRIS.base FW 2.0

ModuleID: A Kliens beállított azonosítója. 4 byte-os érték.

ProtocolVersion: A Kliens által használni kívánt OSP verzió.

5.1.2. Szerver ⇒ Kliens

A Szerver válasza a TCP csatornán először megjelenő CONNECT üzenetre. A DISCONNECT állapotra (élő kapcsolat közben érkező CONNECT üzenet) a Szerver nem válaszol.

Packet body:

	Leírás
1.byte	ResponseCode
2.byte	Timestamp (MSB)
3.byte	Timestamp (...)
4.byte	Timestamp (...)
5.byte	Timestamp (LSB)

ResponseCode: Válasz a kapcsolódási kérésre.

Érték	Jelentés
0x00	Elutasítva
0x01	Sikeres
0x02	Hibás ModuleID
0x03	Hibás DeviceType
0x04	Hibás ProtocolVersion
0x05	Hibás autentikáció
...	*Fenntartott értékek*

Timestamp: A kapcsolódás időpillanata a Szerver oldalon, Unix Time formátumban. A Kliens használhatja óráinak beállítására.

5.2. PINGREQ

Bármelyik fél küldheti a másiknak. Szerepe pusztán a kapcsolat integritásának ellenőrzése.
Egy fél pontosan egy PINGREQ packetet küldhet ki egy pillanatban, és nem küldhet továbbiakat, amíg nem érkezett pontosan egy PINGRESP csomagja.
Az irányelv megszegésére a másik fél implementációtól függő döntést hozhat. (Pl. kidobja a szabálysértő felet, ignorálja a felesleges PINGREQ csomagokat, esetleg válaszol mindegyikre.)
Egy PINGREQ packet csak az alábbi fix fejlécből áll, azaz 2 byte.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	1	0	0	0	0	0	X
2.byte	Length							

5.3. PINGRESP

Bármelyik fél küldheti a másiknak. Egy PINGRESP csomagot mindig meg kell, hogy előzzön pontosan egy, ellenkező irányba közlekedő PINGREQ csomag.
A váratlan PINGRESP csomagok érkezésére a fogadó fél implementációtól függően reagálhat.

Egy PINGRESP packet csak az alábbi fix fejlécből áll, azaz 2 byte.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	1	0	1	0	0	0	X
2.byte	Length							

5.4. COMMAND

Távoli parancsvégrehajtás kezdeményezése. A packet magába foglalja a végrehajtandó utasítást akár szöveges, akár bináris formában. (A protokoll szempontjából lényegtelen.)
A COMMAND packet mindig tartalmaz egy in-flight-egyértelmű command azonosítót.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	0	1	0	0	0	0	X
2.byte	Length							

5.4.1. Szerver ⇒ Kliens

Megjegyzés: A Szerver felelőssége olyan utasításokat küldeni, amiket a kliens meg is ért. Az ilyen utasítások az OSP-t implementáló konkrét eszköz dokumentációjában találhatóak.

Packet body:

	Leírás
1.byte	CommandID
2.byte	Script (string)
	...
n.byte	Script (string)

CommandID: A parancs in-flight-egyértelmű azonosítója. Megadása azért szükséges, hogy az egymás utáni parancsokra később érkező válaszok egyértelműen azonosíthatók legyenek.

Script: A parancs és paraméterei akár szöveges, akár bináris formában.

5.4.2. Kliens ⇒ Szerver

A Kliens válasza egy, a Szerver által küldött parancsra.

Packet body:

	Leírás
1.byte	CommandID
2.byte	ExitCode
3.byte	Response (string)
	...
n.byte	Response (string)

CommandID: A kiváltó parancs in-flight-egyértelmű azonosítója. Az ugyanilyen CommandID-vel kiadott parancsra adott válasz. *Ha a kliens érvénytelen parancs azonosítót küld vissza, a szerver ignorálja azt.*

ExitCode: A parancs eszközfüggő visszatérési kódja a Kliens rendszerében.

Response: A kliens válasza a parancsra. (Lehet üres string is.)

5.5. FIRMWARE

A firmware frissítés kommunikációs folyamata. A Kliens a letöltendő firmware-t annak nevével azonosítja, és adott méretű darabokban (chunk) kéri el a Szervertől.

Fixed header:

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1.byte	MessageType				C	S	A	R
	0	1	1	0	0	X	0	X
2.byte	Length							

5.5.1. Kliens ⇒ Szerver

A kliens jelzi a szerver felé, hogy egy adott firmware melyik darabját (chunk) kéri.

Packet body:

	Leírás
1.byte	ChunkID (MSB)
2.byte	ChunkID (LSB)
3.byte	FirmwareName (string)
...	
22.byte	FirmwareName (string)

ChunkID: A firmware darabka azonosítója. 2 byteos, előjel nélküli érték, azaz maximum 65535 csomag lehetséges.
A protokoll nem definiálja az egyes chunkok méretét.
FirmwareName: A firmware egyedi azonosítására szolgáló 20 karakteres szöveg mező.

5.5.2. Szerver ⇒ Kliens

Firmware frissítés során egy firmware darabka (chunk) küldése. A packet tartalmazza a firmware globálisan egyértelmű megnevezését (verzióját), valamint a chunk darabkáját.
Amennyiben a Kliens olyan verziót vagy chunkot kért, amely nem létezik, akkor a szerver üres nevű és 0-s chunk id-t tartalmazó FIRMWARE csomagot küld vissza. (Üres ChunkData-val)

Packet body:

	Leírás
1.byte	ChunkID (MSB)
2.byte	ChunkID (LSB)
3.byte	FirmwareName (string)
...	
22.byte	FirmwareName (string)
23.byte	ChunkData (string)
...	
n.byte	ChunkData (string)

ChunkID: A firmware darabka azonosítója. 2 byteos, előjel nélküli érték, azaz maximum 65535 chunk lehetséges.

FirmwareName: A firmware egyedi azonosítására szolgáló 20 karakteres szöveg mező.

ChunkData: A firmware darabka binárisa. A protokoll nem definiálja az egyes chunkok méretét, csak a fix fejléc Length mezőjében megadható 256 MB ad abszolút korlátot.

5.6. DATA

Általános adatcsomagok, amelyeket mindig a Kliens küld a Szervernek. Ezekre a csomagokra igazak a következők:

- A DATA csomagok tárolhatóak (Cache flag) kliensoldalon és újraküldhetőek.
- A DATA csomagok menthetőek (Saved flag) kliensoldalon.
- A DATA csomagok (és csak azok) kérhetnek ACKNOWLEDGE-ot.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	1	0	0	0	X	X	X	X
2.byte	Length							

5.6.1. Kliens ⇒ Szerver

Adatcsomag küldése. Az adatcsomag rendelkezik egy, a Kliens által generált in-flight-egyértelmű azonosítóval. A csomag továbbá rendelkezik egy DataType mezővel, amely a csomag tartalmának szemantikájára utal, és a Kliens és a Szerver közös megegyezés alapján használja.

Packet body:

	Leírás
1.byte	MessageID
2.byte	DataType (MSB)
3.byte	DataType (LSB)
4.byte	Payload (string)
	...
n.byte	Payload (string)

MessageID: A Kliens által a DATA packetnek generált in-flight-egyértelmű azonosító. A MessageID-k kiosztása folytonos, azaz a Szerver észlelhet egy esetleges adatkiesést, és RESEND kérést küldhet a Kliens felé a hiányzó MessageID-vel.

DataType: A Payload struktúráját és tartalmát meghatározó paraméter. 2 byteos, előjel nélküli érték.
Foglalt DataType értékek:

Érték	Leírás
0	RESEND packetre adott hibajelzés, lásd lentebb
1-9	Fejlesztésre, tesztelésre fenntartott értékek
>10	Eszközspecifikus adatpacket, lásd a konkrét eszköz OSP protokoll mellékletében

A 0-s DataType érték: A Kliens olyan esetekben küldheti a Szervernek ezeket a packeteket, amikor a Szerver egy érvénytelen azonosítót tartalmazó RESEND csomagot küldött. A Szerver ilyen packet fogadásakor nyugtázza az azonosító érvénytelenségét és továbblép.
Minden további 0-s DataType értékű csomagot a szerver köteles ignorálni.
Payload: Tetszőleges hosszú, DataType-nak megfelelő adattartalom, struktúrája a protokoll szempontjából lényegtelen. A definiált DataType-ok struktúrájának részletes leírása a mellékletekben található.

5.6.2. Szerver ⇒ Kliens

Soha nem küldheti a szerver a kliens felé.
Implementációfüggő, hogy a Kliens miképpen kezeli ezen szabály megsértését.

5.7. RESEND

Egy in-flight-egyértelmű MessageID azonosítóval ellátott DATA packet újraküldésének kérelmezése.
A protokoll korábbbi és jelenlegi, **1.2** verziója DATA packetet csak Kliens ⇒ Szerver irányban enged meg, ezért RESEND packetet csak Szerver ⇒ Kliens irányban értelmezünk.

Fixed header:

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1.byte	MessageType				C	S	A	R
	0	1	1	1	0	0	0	X
2.byte	Length							

5.7.1. Szerver ⇒ Kliens

Packet body:

	Leírás
1.byte	MessageID

MessageID: Azon DATA packet azonosítója, amely újraküldését a Szerver kérelmezni szeretné.

5.7.2. Kliens ⇒ Szerver

Jelen specifikáció nem határoz meg olyan körülményeket, melyek hatására a kliens ezt a csomagot a szervernek küldené.
Implementációfüggő, hogy a szerver miképpen reagál az ilyen packetek érkezésére.

5.8. ACKNOWLEDGE

Egy adott, in-flight-egyértelmű MessageID-vel ellátott DATA packet nyugtázására szolgáló packet típus. A protokoll korábbi és jelenlegi, **1.2** verziója DATA packetet csak Kliens ⇒ Szerver irányban enged meg, ezért nyugtát, ACKNOWLEDGE packetet csak Szerver ⇒ Kliens irányban értelmezünk.

Fixed header:

bit	7	6	5	4	3	2	1	0
1.byte	MessageType				C	S	A	R
	0	0	1	1	0	0	0	X
2.byte	Length							

5.8.1. Szerver ⇒ Kliens

A szerver nyugtázza egy DATA packet megérkezését. Az ACKNOWLEDGE tartalmazza a nyugtázott üzenet in-flight-egyértelmű azonosítóját.

Packet body:

	Leírás
1.byte	MessageID

MessageID: A nyugtázott csomag in-flight-egyértelmű azonosítója. Érvénytelen azonosító esetén a kliens ignorálja a csomagot.

5.8.2. Kliens ⇒ Szerver

A korábbi és jelenlegi, **`1.2`** verzióban nem megengedett.
Implementációfüggő, hogy a szerver miképpen reagál az ilyen packetek érkezésére.

5.9. Protokoll verziók

Az aktuális OSP verzió: v **`1.2`**

Előzmények:

- **`1.0`** - Koncepció (2013)
- **`1.1`** - Első implementált verzió (2014)
- **`1.2`** - CRC flag kiegészítés (2015)