ETSI - CPS Implementation Task Force
UPER bit incompatibility of different ANS.1 Compilers

DLR

Document properties

Title	UPER bit incompatibility of different ANS.1 Compilers
Subject	ETSI - CPS Implementation Task Force
Institute	Institut für Verkehrssystemtechnik
Created by	Jan-Elric Neumann
Participant	Jan-Elric Neumann
Reviewed by	Sten Ruppe
Approved by	Sten Ruppe
Date	17.05.2024
Version	1.0
File path	etsi-cpm-itf-report_UPER bit incompatibility_2024-05-17

Table of Contents

1. Management Summary

The industry does not yet consider CPS v2.1.1 to be production ready. Further development is being processed in work item RTS/ITS-001985, aiming to "update […] the Collective Perception Service specification to achieve full compliance with the Release 2 ITS framework and increase the maturity of its available functionalities." [1]

At the same time, the implementation task force (WG1 ETSI TS 103 324 ITF) is examining the practicability of the current standard. It provides support work for the three groups RTS/ITS-001985 (CPS Release v2.2.1), DTS/ITS-001235 (CPS Interoperability Test Specification), DTS/ITS-001978 (CPS Conformance Test Specification), by implementing individual components via various ETSI members. The goal is to find defects in the standard or to identify features that are over-engineered. Extensions of the CPS are not part of the work.

As part of the Implementation Taskforce, a working group was set up with VW, Bosch and DLR to carry out initial tests. The first objective was to check whether a CPM message v2.1.1 can be read with different compilers. In short, can we understand each other fundamentally?

We noticed that the UPER coded message from the generated decoder of one compiler decodes the message correctly and is not decoded correctly by the other compiler. These compilers are in widespread use in the ITS world.

The CPM V2.1.1 employs an ASN.1 construct (further constraining a type defined with an extensible constraint)[3, p. 96] which has not been used yet in an ITS V2X message yet.

This deviation of compilers for this construct poses an interoperability issue for V2X communication if the use of this construct is not avoided. Avoiding this or similar constructs does not represent a limitation of functionality.

2. The Problem

During the CPM Implementation Task Force activity, we noticed a discrepancy of encoding a known CPM Payload with different ASN.1 compilers.

Payload in ASN.1 Value Notation

```
{
  header {protocolVersion 2, messageId cpm, stationId 12345678},
  payload {
    managementContainer {
      referenceTime 0,
      referencePosition {
        latitude 521133819,
        longitude 98909254,
        positionConfidenceEllipse {semiMajorConfidence 0, semiMinorConfidence 0,
semiMajorOrientation 0},
        altitude {altitudeValue 23300, altitudeConfidence alt-000-05}
      }
    },
    cpmContainers {
      {
        containerId 5,
        containerData PerceivedObjectContainer : {
          numberOfPerceivedObjects 1,
          perceivedObjects {
            {
              objectId 1,
              measurementDeltaTime -100,
              position {
                xCoordinate {value 1000, confidence 10},
                yCoordinate {value 1200, confidence 20},
                zCoordinate {value 100, confidence 30}
              },
              velocity cartesianVelocity : {
                xVelocity {value 200, confidence 10},
                yVelocity {value 100, confidence 10},
                zVelocity {value 10, confidence 10}
              },
              acceleration cartesianAcceleration : {
                xAcceleration {value 11, confidence 2},
                yAcceleration {value 5, confidence 1},
                zAcceleration {value 3, confidence 1}
              },
              objectDimensionY {value 22, confidence 1},
              objectDimensionX {value 50, confidence 1},
              objectPerceptionQuality 2
            }
          }
        }
      }
    }
  }
}
```

UPER Encoding of Compiler #1

```
02 0E 00 BC 61 4E 00 00 00 00 00 02 A5 A6 3F DB 89 78 72 30 00 00 00 00 0F 0D 21 02
14 00 40 2E 3A 00 00 5E 73 01 F4 00 4C 09 60 02 70 03 20 0E F0 31 C4 C0 63 13 00 24
4E AB 04 A5 02 A3 02 26 01 50 18 80 80 00
```

UPER Encoding of Compiler #2

```
02 0E 00 BC 61 4E 00 00 00 00 00 02 A5 A6 3F DB 89 78 72 30 00 00 00 00 0F 0D 21 04
28 00 80 5C 74 00 00 BC E6 03 E8 00 98 12 C0 04 E0 06 40 1D E0 63 89 80 C6 26 00 48
9D 56 09 4A 05 46 04 4C 02 A0 31 01 00
```

It is obvious that the resulting UPER Encodings are different, starting with the 221th bit (0-indexed):

Compiler #1	…**00100001000001000010100000000000010000000010111000111010**…
Compiler #2	…**00100001000001000010100000000000010000000010111000111010**…

Table 1 Excerpt of the UPER encoding

Compiler #1 inserts an extra 0-bit at bit-position 217 (0-indexed). All previous and subsequent bits are equal (excluding trailing bits necessary to fill the last byte).

Compiler #1	…**001000010 0 0000100001010000000000010000000010111000111010**…
Compiler #2	…**001000010 x 0000100001010000000000010000000010111000111010**…

Table 2 Excerpt of the UPER encoding, realigned; x = Not present

x = Not Present

3. The culprit

During investigation the culprit could be identified:

“

```
/**
 * This DF represents a list of CPM containers, each with their type
identifier.
*/
WrappedCpmContainers::= SEQUENCE SIZE(1..8,...) OF WrappedCpmContainer

/**
 * This DF represents a list of CPM containers, each with their type identifier
with an additional constraint.
*/
ConstraintWrappedCpmContainers ::= WrappedCpmContainers
    ((WITH COMPONENT (WITH COMPONENTS {..., containerId (ALL EXCEPT 1)})) |
    (WITH COMPONENT (WITH COMPONENTS {..., containerId (ALL EXCEPT 2)})))
```

” [2]

The ETSI CPM V2.1.1 defines five data-containers [cf. 2]:

- Originating Vehicle Container

- Originating Rsu Container
- Sensor Information Container
- Perception Region Container
- Perceived Object Container

At the time of writing up to 8 of these containers can be placed inside a CPM (‘WrappedCpmContainers’). For backwards and forwards combability reasons, the ‘WrappedCpmContainer’ constrained is declared extendable.

To prevent a CPM from both containing an Originating Vehicle Container and Originating Rsu Container, the Type ‘ConstraintWrappedCpmContainers’ is defined as a subtype of ‘WrappedCpmContainers’. The ‘ConstraintWrappedCpmContainer’s restricts its containing data-containers to have all but ‘containerId’ 1 OR all but ‘containerId’ 2.

The problem with this construct arises when considering the rules for extension marker inheritance for constrained types and subtypes:

“If a type defined with an extensible constraint is further constrained with an "ElementSetSpecs", the resulting type does not inherit the extension marker nor any extension additions that may be present in the former constraint (see 50.11). For example:

A ::= INTEGER (0..10, ...) -- A is extensible.
B ::= A (2..5) -- B is inextensible. [Emphasis by Author]
C ::= A -- C is extensible. “[3, p. 96]

By restricting the subtype ‘ConstraintWrappedCpmContainers’ of Type ‘WrappedCpmContainers’ further, the extendibility of ‘WrappedCpmContainers’ is dropped.

Compiler #1 assumes ‘ConstraintWrappedCpmContainers’ to be extendable and inserts the extension-bit at bit-position 217 (0-indexed) before encoding the ‘ConstraintWrappedCpmContainers’s length and its elements.

Compiler #2 assumes ‘ConstraintWrappedCpmContainers’ to be non-extendable and directly encodes the length and its element without first placing the extension bit.

	ManagementContainer		ConstraintWrappedCpmConstainers		
Field	…	*AltitudeConfidence*	Extension Bit	Length	Elements
Encoding #1	…	*0010*	**0**	000	0100001010000000000…
Encoding #2	…	*0010*	**x**	000	0100001010000000000…

Table 3 Excerpt of the UPER encoding, with corresponding fields, realigned; x = Not present

4. Minimal Test Case

The following ASN.1 Module presents a minimal test case

```
Defect DEFINITIONS AUTOMATIC TAGS ::= BEGIN

Content ::=  INTEGER(1..16)
ContentSequence ::= SEQUENCE SIZE(1..8, ...) OF Content
UnconstrainedContentSequence ::= ContentSequence
ConstrainedContentSequence ::= ContentSequence (WITH COMPONENT (1..8))

END
```

Test value: `{1,2,3,4}`

	UnconstrainedContentSequence (UPER)	**ConstrainedContentSequence (UPER)**
Compiler #1	00110000000100100011	0**0**110000000100100011
Compiler #2	00110000000100100011	0110000000100100011

Table 4 Full UPER encoding without trailing 0-bits.

The table shows the UPER bit-encoding (omitting the trailing 0-bits) of the test value using the Subtypes ‘UnconstrainedContentSequence’ and ‘ConstrainedContentSequence’.

Both compilers (correctly) identify ‘UnconstrainedContentSequence’ as extendable and insert the extension-bit at bit-position 0 (0-indexed).

For the ‘ConstrainedContentSequence’ case, we can observe the same behaviour as with the ‘ConstraintWrappedCpmContainers’: Compiler #1 inserts the extension-bit, while Compiler #2 omits the extension-bit.

5. Who does it affect?

Compiler	**Behaviour**
Objective Systems ASN.1C V6.6.9	Inserts Extension-Bit for ‘ConstrainedContentSequence’
Erik Moqvist’s asn1tools V0.166.0	Inserts Extension-Bit for ‘ConstrainedContentSequence’
OSS Nokalva ASN1Step V10.2.0 (from Asn1Studio V10.2.0)	No Extension-Bit for ‘ConstrainedContentSequence’
Fabrice Bellard’s ffasn1dump	Appears to ignore extension markers
Lev Walkin’s asn1c	Could not compile ‘defect’ module nor ‘cpm’ module

Table 5 Overview of the behavior of different ASN.1 compilers [4][5][6][7][8]

6. How to mitigate?

The inconsistent handling of the extendibility of further constrained subtypes represents a major risk to the interoperability of different V2X operators. Therefore, we advise against the usage of this or similar ‘tricky’ ASN.1 constructs on the Message Definition Level and propose adding the necessary constraints to the respecting Technical Standards.

Further we advise against the usage of ASN.1 constraints at the message definition level which have no effect on the bit encoding ("per-visible"). These constraints should instead be specified in the Technical Specifications of the facilities.

List of Tables

Bibliography

[1] RTS/ITS-001985, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=69494, 17.05.2024
[2] CPM-PDU-Descriptions, https://forge.etsi.org/rep/ITS/asn1/cpm_ts103324/-/blob/v2.1.1/asn/CPM-PDU-Descriptions.asn, 17.05.2024.
[3] ITU X.680 52.5, https://www.itu.int/rec/T-REC-X.680-202102-I/en, 17.05.2024.
[4] Objective Systems ASN.1C, https://obj-sys.com/, 17.05.2024.
[5] Moqvist, Erik, asn1tools, https://github.com/eerimoq/asn1tools/, 17.05.2024.
[6] OSS Nokalva ASN1Step (from Asn1Studio), https://www.oss.com/, 17.05.2024.
[7] Bellard, Fabrice, ffasn1dump, https://bellard.org/ffasn1/,17.05.2024.
[8] Walkin, Lev, asn1c, https://github.com/vlm/asn1c, 17.05.2024.