3GPP TSG RAN WG2 #125 R2-24XXXXX

**Athens, Greece, February 26 – March 1, 2024**

**Agenda Item:** **7.0.3**
**Source:** **Toyota InfoTechnology Center**
**Title:** **[Y003] Discussion on the extensibility of SL-TxProfiles**

**Document for:** **Discussion and Decision**

# Introduction

This contribution discusses the extensibility of SL-TxProfiles and addresses R18 ASN.1 review issue Y003.

# Discussion

## Background

The latest version of the NR Radio Resource Control (RRC) specification [1] defines the following *SL-QoS-Info-v1800* information element in the *SidelinkUEInformationNR* message:

SL-QoS-Info-v1800 ::= SEQUENCE {

 sl-QoS-FlowIdentity-r18 SL-QoS-FlowIdentity-r16,

 sl-TxInterestedFreqList-v1800 SL-TxInterestedFreqList-r16 OPTIONAL,

 sl-TxProfile-v1800 ENUMERATED {backwardsCompatible, backwardsIncompatible} OPTIONAL,

 ...

}

The information element contains a field *sl-TxProfile-v1800* which is defined with a build-in enumerated data type that is composed of two code-points. The information element *SL-QoS-Info-v1800* is further extensible with an extension marker.

In addition, the specification defines a separate *SL-TxProfile* module which is composed of an information element *SL-Tx-Profile-v1800.* The information element has the same code-points as above, but it includes also multiple spares for the sake of extensibility.

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= ENUMERATED {backwardsCompatible, backwardsIncompatible, spare6, spare5, spare4, spare3, spare2, spare1}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

So far, the *SL-TxProfile-v1800* information element is neither referenced anywhere in the ASN.1 code nor imported to any module. It seems that the new module is either accidentally omitted in the CR drafting or the intention is to specify these profiles in another way. Nevertheless, there should not be any unreferenced information elements or unused modules in the code.

**Observation 1: *SL-TxProfile-v1800* is defined but neither referenced nor imported to any module.**

There are many possible options to resolve this issue as discussed below.

## Option 1

Remove the unreferenced information element *SL-TxProfile-v1800* and the *SL-TxProfile* module because its content is not imported to other modules.

~~-- ASN1START~~

~~-- TAG-SL-TXPROFILE-START~~

~~SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=~~

~~BEGIN~~

~~SL-TxProfile-v1800 ::= ENUMERATED {backwardsCompatible, backwardsIncompatible, spare6, spare5, spare4, spare3, spare2, spare1}~~

~~END~~

~~-- TAG-SL-TXPROFILE-STOP~~

~~-- ASN1STOP~~

Future extensions to SL-TxProfiles can be added in the *SL-QoS-Info-v1800* information element after the extension marker as exemplified below:

SL-QoS-Info-v1800 ::= SEQUENCE {

 sl-QoS-FlowIdentity-r18 SL-QoS-FlowIdentity-r16,

 sl-TxInterestedFreqList-v1800 SL-TxInterestedFreqList-r16 OPTIONAL,

 sl-TxProfile-v1800 ENUMERATED {backwardsCompatible, backwardsIncompatible} OPTIONAL,

 ...,

 [[

 sl-TxProfile-vXYZ ENUMERATED {lorem, ipsum} OPTIONAL,

 ]]

}

The advantage of the solution is that the unreferenced information element and the unused module are removed. The downside is that the enumerated type needs to be introduced in every place where new profiles are used which means unnecessary duplication.

## Option 2

Import the information element *SL-TxProfile-v1800* from *SL-TxProfile* module to *NR-RRC-Definitions* module and replace the build-in enumerated type with the named type *SL-TxProfile-v1800.* The import is needed because the *SidelinkUEInformationNR* message that conveys the *sl-TxProfile* field is defined in the *NR-RRC-Definitions* module.

-- ASN1START

-- TAG-NR-RRC-DEFINITIONS-START

NR-RRC-Definitions DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS

 SL-TxProfile-v1800

FROM SL-TxProfile;

-- TAG-NR-RRC-DEFINITIONS-STOP

-- ASN1STOP

The build-in enumerated type should be replaced with the named type as shown below:

SL-QoS-Info-v1800 ::= SEQUENCE {

 sl-QoS-FlowIdentity-r18 SL-QoS-FlowIdentity-r16,

 sl-TxInterestedFreqList-v1800 SL-TxInterestedFreqList-r16 OPTIONAL,

 sl-TxProfile-v1800 SL-TxProfile-v1800~~ENUMERATED {backwardsCompatible, backwardsIncompatible}~~ OPTIONAL,

 ...

}

The extensibility of the solution raises questions because the named type contains spare values, but the *SidelinkUEInformationNR* message is transferred from the UE to the network. If the UE and the network implements the same ASN.1 version, spare values should not be an issue. A problem however arises since this not necessarily the case. Therefore, the UE may use a code-point that is non-comprehended by the network. For example, if *spare6* is defined a new meaning in Rel-19 but the network does not implement Rel-19 ASN.1, the network will receive a spare value. In general, the reception of a spare value is defined as an error case for the UE. The standard does not specify network behavior, but it is plausible to assume that a received spare value throws an exception/error also at the network side. One possible solution is to remove the spare values as shown below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1~~}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

The removal of spare values does not impede extensibility because the information element can be extended by defining a new information element whenever new values are needed as exemplified below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1~~}

SL-TxProfile-vXYZ ::= ENUMERATED {lorem, ipsum}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

Now, the profiles are defined in a separate module and therefore the future extensions of these profiles need to be imported as well. This is shown below for the example extension *SL-TxProfile-vXYZ*:

-- ASN1START

-- TAG-NR-RRC-DEFINITIONS-START

NR-RRC-Definitions DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS

 SL-TxProfile-v1800,

 SL-TxProfile-vXYZ

FROM SL-TxProfile;

-- TAG-NR-RRC-DEFINITIONS-STOP

-- ASN1STOP

The extension can be added to the SL-QoS-Info-v1800 information element after the extension marker in the same manner as in the option 1. The only difference compared to option 1 is that an imported named type from *SL-TxProfile* module is referenced instead of the build-in enumerated type, as shown below:

SL-QoS-Info-v1800 ::= SEQUENCE {

 sl-QoS-FlowIdentity-r18 SL-QoS-FlowIdentity-r16,

 sl-TxInterestedFreqList-v1800 SL-TxInterestedFreqList-r16 OPTIONAL,

 sl-TxProfile-v1800 SL-TxProfile-v1800~~ENUMERATED {backwardsCompatible, backwardsIncompatible}~~ OPTIONAL,

 ...,

 [[

 sl-TxProfile-vXYZ SL-TxProfile-vXYZ OPTIONAL

 ]]

}

The advantage of the solution is that the profiles and their extensions can be shared between modules. The disadvantage is that references and imports of the information element needs to be added in all modules where they are used every time when new profiles are introduced, i.e. for all extensions.

## Option 3

A variant of option 2 is to redefine the information element as a sequence type that is extensible with an extension marker as shown below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= SEQUENCE {

 sl-TxProfile-v1800 ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1~~},

 ...

}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

The option 3 requires imports like option 2. However, the information element needs to be imported only once instead of adding new imports every time when new profiles are introduced because the information element can be extended by defining a new information element after the extension marker whenever new values are needed as exemplified below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= SEQUENCE {

 sl-TxProfile-v1800 ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1~~},

 ...,

 [[

 sl-TxProfile-vXYZ ENUMERATED {lorem, ipsum} OPTIONAL

 ]]

}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

The advantage of this solution is that the extensions can be added to one place only and no new references to this information are needed in the modules where the information element is imported. The downside is that the *sl-TxProfile-v1800* field is within the *SL-TxProfile* module which may complicate its usage with respect to field descriptions and procedures because SL profiles have already been used in legacy.

Another possibility is to redefine *SL-TxProfile-v1800* as a non-critically extensible sequence type where an empty optional sequence is added in the end of the information element as shown below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= SEQUENCE {

 sl-TxProfile-v1800 ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1~~},

 nonCriticalExtension SEQUENCE {} OPTIONAL

}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

The information element can be extended by defining a new information element in the non-critical extension whenever new values are needed. Even though this kind of variant of option 3 have the same advantage as the extension marker approach, the information element does not then contain a length indicator. Therefore, the extension should be contained as an octet string every time it is referenced. The overhead of an extension marker is about the same as that of an extension marker, but the octet string complicates the usage. Therefore, it is difficult to see any added value compared to the extension marker approach above.

## Other extension mechanisms

It should be noted that there are also other possible extension mechanisms as discussed below.

One possibility could be making the build-in enumerated type extensible with an extension marker instead of spare values or by changing the enumerated type to a choice type that is likewise extensible with an extension marker. Observe, the difference between enumerated and choice types is only visible in the ASN.1 syntax whereas the encoded outcome/octets is the same. These alternatives are exemplified below:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= ENUMERATED {backwardsCompatible, backwardsIncompatible~~, spare6, spare5, spare4, spare3, spare2, spare1,~~ ...}

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

and the same with a choice type:

-- ASN1START

-- TAG-SL-TXPROFILE-START

SL-TxProfile DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

SL-TxProfile-v1800 ::= CHOICE {

 backwardsCompatible NULL,

 backwardsIncompatible NULL,

 ...

} ~~ENUMERATED {backwardsCompatible, backwardsIncompatible, spare6, spare5, spare4, spare3, spare2, spare1}~~

END

-- TAG-SL-TXPROFILE-STOP

-- ASN1STOP

None of these extension mechanisms however resolves the problem of non-comprehended extensions because in both cases it is possible that the receiver does not comprehend the extension whenever the sender is implemented according to a later Release than the receiver. If so happens, the receiver does not have any information about the profile.

It is therefore concluded that these other extensions mechanisms are unsuitable for this information elements, and hence they are not discussed further in this contribution.

## Comparison

There is no difference with respect to overhead between option 1, option 2, and option 3 because all of them make use of an extension marker. Options 2 and 3 are somewhat more convoluted than option 1 due to the imports. Option 3 however requires only imports once because the extensions are within the information element whereas option 2 requires new imports for all extended versions. One the other hand, option 3 may complicate field descriptions and procedures because the profiles are defined and extended within the new module unlike the legacy profiles.

One important advantage of options 2 and 3 compared to option 1 is that the profiles are defined in one place, and they can be imported to other modules. It means that there is a mechanism for keeping them consistent. It should be noted that imports have not been used in the *NR-RRC-Definitions* module before and, so far, the SL-TxProfile is not imported to the *NR-RRC-Definitions* module at all.

It is therefore proposed to choose between the options and discuss the following. One of the three proposals below should be agreed:

**Proposal 1-1: Remove the unreferenced information element *SL-TxProfile-v1800* and the unused *SL-TxProfile* (option 1) unless there are reasons to share the same profile definition between different modules (option 1).**

**Proposal 1-2: If there are reasons to share the same profile definition between different modules, remove the spare values and add the named type to imports (option 2).**

**Proposal 1-3: If there are reasons to anticipate many future extensions to SL TX profiles, remove the spare values, add the named type to imports, and redefine the named type as a sequence type that is extensible with an extension marker (option 3).**

# Conclusion

This contribution discusses the extensibility of *SL-TxProfile* information element and offers the following observations and proposals:

**Observation 1: *SL-TxProfile-v1800* is defined but neither referenced nor imported to any module.**

**Proposal 1-1: Remove the unreferenced information element *SL-TxProfile-v1800* and the unused *SL-TxProfile* (option 1) unless there are reasons to share the same profile definition between different modules (option 1).**

**Proposal 1-2: If there are reasons to share the same profile definition between different modules, remove the spare values and add the named type to imports (option 2).**

**Proposal 1-3: If there are reasons to anticipate many future extensions to SL TX profiles, remove the spare values, add the named type to imports, and redefine the named type as a sequence type that is extensible with an extension marker (option 3).**

# References

1. TS 38.331, v18.0.0