**3GPP TSG-SA5 Meeting #132e S5-204449**

**E-meeting, 17-28 August 2020**

**Source: Nokia, Nokia Shanghai Bell, Huawei, Ericsson**

**Title: 3GPP Forge process**

**Document for: Discussion and Endorsement**

**Agenda Item: 6.4.3**

***Abstract of the contribution:***

*This contribution proposes how SA5 could proceed stage 3 related changes, especially considering quality of stage 3 codes.*

# 1 Decision/action requested

***Please discuss and endorse***

# 2 References

[1] 3GPP TS 28.541: "Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and Stage 3".

# 3 Rationale

Rel16 eNRM exception was agreed in last SA5 meeting, the justification was that the stage 3 is not fully aligned with stage 2. To complete the NRM WI in next plenary, and assure the quality of the NRM specification, this TD proposed process to handle the NRM related CRs, especially regarding stage 3 code in TS 28.541. Generally, the stage 2 CR can be finally agreed in SA5 after the corresponding stage 3 code created, validated/compiled in Forge (committed into CR branch of Forge). Further the conflicts between CRs must be solved before the SA plenary. Finally, the merged code should be validated and published in Forge in the corresponding release branch.

SA5 lead team is driving Forge process discussion, and there’re both short-term and long-term solutions. The proposal in this TD is based on short-term solution with some tailoring.

# 4 Detailed proposal

**Please endorse the process proposed for stage 3 code of all TSs listed in clause 1 below.**

## Stage 3 Solution sets:

The following solution sets are recommended to be provided as stage 3 code and verified in 3GPP Forge for the management services:

* YAML
* YANG
* XML (For PM/Trace File schema)

## Roles in the 3GPP Forge process

There are three different roles related to this activity, and they coordinate with each other to achieve the goal.

* Contribution author
* Code Moderator
* Code Master

The following persons are assigned as Code Moderator for relevant stage 3 specifications:

* TS 28.541
	+ Jing Ping (YAML)
	+ Balázs Lengyel (YANG )
* TS 28.623
	+ Olaf Pollakowski (YAML)
	+ Balázs Lengyel (YANG)
* TS 28.532
	+ Anatoly Andrianov / Olaf Pollakowski /Xuruiyue (YAML)
	+ (any volunteer? ) (PM file format XML)
* TS 28.550
	+ Yizhi Yao (YAML)
* TS 28.536
	+ Jan Groenendijk (YAML)
* TS 32.423
	+ (any volunteer?) (Trace file format XML)

## 3GPP Forge process

### Step 0 - Preparing for an SA5 meeting

* Contribution author is recommended to setup his local verification environment with the latest source code from the 3GPP Forge.
* Contribution author prepares for contribution and make sure the stage 3 source code proposed in the contribution is compiled successfully in the local verification environment.
* Contribution author submits contribution to a meeting. It is recommended that the author provides a link in the contribution to the stage 3 source code in Forge, which can be used for verification.
* There may be many related contributions. It’s recommended that the contribution authors take the offline initiative before the meeting in case there is a potential conflict in multiple contributions from different contributors.

#### Scenario 1: NRM related CR

After a stage 2 CR has been conditionally agreed, the contribution author is required to present the corresponding stage 3 code, either as part of same stage 2 CR, or separate stage 3 CR. Contribution author should make sure of the following:

1. Provide at least one of: YAML or YANG code.

2. Create a tdoc branch in Forge for the corresponding stage 3 code, and provide a link in the contribution to the stage 3 source code in Forge, which can be used for verification.

Note: the naming rule of the branch is: tDoc number\_tDoc title, and the spaces in tDoc title replaced with “\_”, e.g. S5-203390\_new\_NRM\_fragment\_to\_support\_RIM.

3. The stage 3 source code proposed in the CR passes Forge validation (compiles successfully) and is committed in corresponding tDoc branch in Forge.

Note: Forge validates the code automatically as part of the commit.

4. If one contribution author has multiple contributions impacting stage 3 code, the contribution author should solve potential conflicts before submitting the stage 3 code.

**The stage 2 and 3 CRs can be finally agreed in SA5 \_ONLY\_after all 4 steps have been completed by the contribution author (or co-signer).**

**Note: the stage 2 definition for a feature would be removed from the specification before freezing of the release if there’s no corresponding stage 3 to satisfy the release criteria.**

#### Scenario 2: MnS Operation related CR

To be added.

### Step 1 – Consideration of the contribution at the SA5 meeting

It’s recommended that the contribution authors merge the related contributions which may be potentially in conflict as much as possible during the meeting. (i.e. author needs to ensure there is no conflict)

### Step 2: Code cross check after SA5 meeting and before SA meeting

The Code Moderator, with appropriate assistance from the relevant Contribution authors, is responsible for taking care of overall code checking, (e.g. merged all CRs in a integration test branch and make sure there’s no compilation error on the merged code), especially conflict checking, before the SA plenary. In case of errors being found during the checking process, the code moderator or corresponding contribution author (depends on the error type, complexity, and severity) shall provide contributions to SA plenary for the error correction. This check needs to be done after each SA5 meeting. if errors => correction to SA Plenary.

Note: conflicts in code must be resolved before the CR approval at the SA plenary... otherwise all conflicting CRs must bewithdrawn/not pursued.

### Step 3: Agreement of the contributions, after the SA meeting

Once the CRs are approved by the SA plenary, Code Master (MCC) implements the approved CRs into corresponding TSs after each SA meeting. The Code Moderator extracts the final stage3 content from the TS and copies it to the code file, then creates a branch in 3GPP Forge and commits the code in the branch. If the code is validated, Code Moderator submits the merge request to the Code Master.

In case of errors, this can be corrected by the Code Moderator in the source code in 3GPP Forge. The Code Moderator could evaluate whether this correction can be incorporated in the TS with a new TS iteration (z) of the version Vx.y.z to be created by MCC, or whether a CR is needed to the next SA5 meeting to correct the TS (in which case it is needed to indicate somewhere that there is a mismatch between the source code in 3GPP Forge and the TS, or alternatively we live with a wrong source code).

Code Master takes care of the merge requests, ensures that commits are squashed, and the original branch deleted.