**3GPP TSG-SA5 Meeting #132e S5-204411**

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

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

**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 28.541, stage 3 code of TS 28.532 and TS 28.623.**

1. 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 (OpenAPI)
	+ Balázs Lengyel (Yang)
* TS 28.532
	+ Anatoly Andrianov / Olaf Pollakowski /Xuruiyue (OpenAPI)
	+ (any volunteer? ) (PM file format XML)
* TS 28.550
	+ Yizhi Yao
* TS 28.536
	+ Jan Groenendijk
* TS 32.423
	+ (any volunteer?) (Trace file format XML)
1. 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 have many related contributions. It’s recommended that the contribution authors take the offline initiative before the meeting in case there is potential conflict in its own multiple contributions.

## Scenario 1: NRM related CR

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

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

Note: Looks like XML solution sets is seen by SA5 as outdated and redundant (XML can be automatically generated from YAML/YANG) - there was a proposal to \_slowly\_ move away from XML as XML is not used by any organization for 5G network configuration.

2. Create 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, the spaces in tDoc title is 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 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, contribution author should solve potential conflicts before submitting the stage 3 code.

Note: 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. (this check is optional)

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

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

## Scenario 2: OpenAPI 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 check, (e.g. merged all CRs in a integration test branch and make sure there’s no compilation error on the merged code), especially conflict check, before the SA plenary. In case of error 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 cannot be implements.

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

Once the changes are confirmed by SA, Code Master (MCC) merges the agreed contributions into corresponding specifications after each SA meeting. The code Moderator extracts the final content from the TS and copies to code file, then creates branch in 3GPP forge and commits the code in the branch, If the code is validated, Code Moderator submits the merge request to 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 to correct the TS (in which case it is needed to indicate somewhere 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.