ETSI TS 128 572 V19.2.0 (2026-04)

TECHNICAL SPECIFICATION

5G;
Management and orchestration;
Management of planned configurations
(3GPP TS 28.572 version 19.2.0 Release 19)

Reference
RTS/TSGS-0528572vj20

Keywords
5G

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is (or any other verb in the indicative mood) indicates a statement of fact

is not (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

Introduction

Configuration management of mobile networks is a complex task. Configuration plans are typically generated by planning tools. This specification describes how these plans can be managed and activated.

1 Scope

The present document specifies the management of planned configurations.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 32.158: "Management and orchestration; Design rules for REpresentational State Transfer (REST) Solution Sets (SS)".

[3] IETF RFC 8072: " YANG Patch Media Type".

[4] 3GPP TS 28.532: "Management and orchestration; Generic management services".

[5] 3GPP TS 32.161: "Management and orchestration; JSON expressions (Jex)".

[6] IETF RFC 7951:" JSON Encoding of Data Modeled with YANG".

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] apply.

3.2 Symbols

None 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply.

4 Concept and overview

This specification introduces the concept of planned configurations that a MnS consumer can create on a MnS producer. In contrast to the one current configuration that refers to the configuration that is active in the managed system, multiple planned configurations can coexist. Multiple planned configurations can be grouped together in planned configuration groups.

Planned configurations and planned configuration groups can be validated and activated. Conditional activation is possible. Fallback configurations allow to revert to a previous state.

5 Requirements

Requirements related to planned configurations

Req-PCG-1: The 3GPP management system shall support creating, reading and deleting planned configurations for managed systems.

Req-PCG-2: The 3GPP management system should support updating planned configurations for managed systems.

Req-PCG-3: The 3GPP management system shall support creating, reading, updating, and deleting meta data for planned configurations.

Requirements related to control parameters for validation and activation of planned configurations

Req-PCC-1: The 3GPP management system shall support a capability allowing a MnS consumer to indicate to a MnS producer if validation and activation shall be processed in best effort or atomic mode.

Requirements related to planned configuration groups

Req-PGG-1: The 3GPP management system should allow MnS consumers to group planned configurations into planned configuration groups.

Req-PGG-2: The 3GPP management system should allow MnS consumers to group planned configuration groups into planned configuration groups.

Req-PGG-3: The 3GPP management system shall support creating, reading and deleting planned configuration groups.

Req-PGG-4: The 3GPP management system should support updating planned configuration groups.

Req-PGG-5: The 3GPP management system shall support creating, reading, updating, and deleting meta data for planned configuration groups.

Requirements related to control parameters for validation and activation of planned configuration groups

Req-PGC-1: The 3GPP management system shall support creating, reading, updating, and deleting control parameters for planned configuration groups.

General plan validation requirements

Req-PVG-1: The 3GPP management system shall support a capability allowing a MnS consumer to request a MnS producer to validate a planned configuration.

Req-PVG-2: The 3GPP management system should support a capability allowing a MnS consumer to retrieve the status of a plan validation.

Req-PVG-3: The 3GPP management system should support a capability allowing a MnS consumer to request a MnS producer to cancel an ongoing plan validation.

Req-PVG-4: The 3GPP management system shall support a capability allowing a MnS consumer to retrieve the result of a plan validation.

Req-PVG-5: The 3GPP management system should allow for multiple sequential validation of the same plan.

Req-PAG-6: The 3GPP management system shall support a capability allowing a MnS consumer to find out if a plan has been previously validated.

Requirements related to the validation of multiple plans

Req-PVM-1: The 3GPP management system should support a capability allowing a MnS consumer to request a MnS producer to validate multiple planned configurations with a single request.

Req-PVM-2: The 3GPP management system shall support a capability allowing a MnS consumer to indicate to a MnS producer the type of relationship between the plans, for the case where multiple plans are requested to be validated with a single request.

General plan activation requirements

Req-PAG-1: The 3GPP management system shall support a capability allowing a MnS consumer to request a MnS producer to activate a planned configuration.

Req-PAG-2: The 3GPP management system should support a capability allowing a MnS consumer to retrieve the status of a plan activation.

Req-PAG-3: The 3GPP management system should support a capability allowing a MnS consumer to request a MnS producer to cancel an ongoing plan activation.

Req-PAG-4: The 3GPP management system shall support a capability allowing a MnS consumer to retrieve the result of a plan activation.

Req-PAG-5: The 3GPP management system should allow for multiple sequential activations of the same plan.

Req-PAG-6: The 3GPP management system shall support a capability allowing a MnS consumer to find out if a plan has been previously activated.

Req-PAG-7: The 3GPP management system shall validate a planned configuration automatically when activation of the planned configuration is requested without prior validation.

Requirements related to control parameters for activation

Req-PAC-1: The 3GPP management system should support a capability allowing a MnS consumer to indicate to a MnS producer if activation should be done with the least possible service impact or within the shortest possible time.

Req-PAC-2: The 3GPP management system should support a capability allowing a MnS consumer to indicate to a MnS producer if fallback shall be enabled.

Requirements related to the activation of multiple plans

Req-PAM-1: The 3GPP management system should support a capability allowing a MnS consumer to request a MnS producer to activate multiple planned configurations with a single request.

Req-PAM-2: The 3GPP management system shall support a capability allowing a MnS consumer to indicate to a MnS producer the type of relationship between the plans, for the case where multiple plans are requested to be activated with a single request.

Req-PAM-3: The 3GPP management system shall support a capability allowing a MnS consumer to indicate to a MnS producer the behavior if plan conflicts are detected, for the case where multiple plans are requested to be activated with a single request.

Requirements related to conditional activation of plans

Req-CAP-1: The 3GPP management system should allow MnS consumers to create conditions on MnS producers that specify when planned configurations are activated.

6 Solution description

6.1 Planned configurations

6.1.1 Definition

The *current configuration* is the configuration currently used by the managed system. It is represented by (a tree of) configuration data nodes. State data nodes are not included. The format of the configuration data node tree is typically defined by (stage 2) NRM definitions and (stage 3) schema definitions.

Note that the data node tree representing a managed system typically comprises besides configuration data nodes also (read-only) state data nodes, that represent the current state of the managed system. Configuration data nodes and state data nodes may be separated in dedicated subtrees, but they may be also contained in one tree.

A *planned configuration* is a configuration that can be manipulated without impacting the current configuration of the system. It relates to configuration data nodes and not to state data nodes. It is maintained side-by-side with the current configuration.

6.1.2 Format of planned configurations

A planned configuration is represented by a set of operations to be applied to the current configuration. Each operation is described by three parameters: a modify operator, a target, and a value. An operation may be annotated with a description. The order of operations in the operation set is irrelevant.

The target parameter identifies a data node in the current configuration that is to be manipulated by the modify operator. This data node is referred to as target data node. The modify operarator specifies the modification to be applied to the target data node. The value parameter specifies the value that is used by the modify operator on the target data node.

The modify operator indicates, if a new data node is to be created in the current configuration, an existing data node is to be updated, or an existing data node is to be deleted. The value parameter is absent for data node deletions, in all other cases it needs to be present.

This specification defines a general format for planned configurations for which the following applies.

- The data node identified by the target parameter may be any kind of data node (managed objects, attributes, attribute fields, attribute elements).
- The following modify operations are defined:
  - **create**: creates the target data node in the current configuration with the representation specified by the value parameter, if the target data node does not exist in the current configuration. If the target data node already exists, an error is raised.
  - **merge**: merges the (partial) representation of the target data node specified in the value parameter into the current representation of the target data node, if the target data node exists in the current configuration. If the target data node does not exist, an error is raised. Deletion of data nodes is possible.
  - **merge-create**: merges the (partial) representation of the target data node specified in the value parameter into the current representation of the target data node, if the target data node exists in the current configuration. If the target data node does not exist, it is created with its representation set to the representation specified in the value parameter. Deletion of data nodes is possible.
  - **delete**: deletes the target data node (and all its contained data nodes), if the target data node exists in the current configuration. If the target data node does not exist, the MnS producer shall ignore the operation and shall not raise an error.
- In a valid planned configuration, the managed objects containg the target managed object shall exist either in the current configuration or as part of the planned configuration. The MnS producer is not required to implicitly create managed objects instances that are used as path components but which do not exist in the plan nor in the current configuration.

For the use with current configurations that are based on the managed object concept a special profile called Managed Object Plan is provided. For a Managed Object Plan the following applies:

- The data node identified by the target parameter shall be a managed object instance. No other data node types are allowed as target data nodes. The MnS producer shall reject a planned configuration if it contains target data data nodes that are not managed objects.
- All updates for an existing managed object should be grouped into a single "merge" or "merge-create" operation. The MnS producr may reject planned configurations where more than one operation targets the same managed object instance.

The producer shall indicate whether general plans or only Managed Object Plans are supported.

The MnS consumer may specify an identifier for each operation in the creation request for a planned configuration descriptor. The MnS producer may discard this identifier upon reception of the request and use the positional index of the operation in the operation set instead whenever a reference to an operation in the operation set is required. The positional indx of the first operation in the operation set is "0".

6.1.3 Meta data for planned configurations

Each planned configuration is described by meta data. This includes the name, the version, and a human readable textual description. It is also possible to specify additional MnS consumer defined properties (key value pairs) to further describe and qualify the planned configuration, for example to specify who created the planned configuration. These annotations are for usage by a (human) MnS consumer only. The are not processed by the MnS producer.

Furthermore, it includes the content type of the planned configuration it is specified if the operations of the operations set are an atomic set, where all operations need to be activated successfully, or if some operations are allowed to fail (non-atomic or best effort set). In other words, for atomic sets, all operations of the set must be processed successfully or no operation at all. Operations, that are already applied when an error occurs, must be rolled back. For best effort sets, the operations that can be processed successfully are processed, and those that for whatever reason cannot be processed successfully are not processed. In this mode the activation process continues on the occurrence of an error. In a third mode the activation process stops on the occurrence of an error.

For example, subscriptions to alarm notifications on multiple Network Functions may have value also in case the subscription on some Network Functions fails. When analytics are computed based on measurements collected at some well selected Network Functions, then all measurement collection jobs must be created. If one job cannot be created it does not make sense to collect the remaining measurements.

The date and time at which information in the planned configuration was modified the last time is provided as well.

6.1.4 Planned configuration descriptor

The operation set and related meta data are specified in a planned configuration descriptor. Each descriptor has a unique identifier.

6.2 Planned configuration groups

6.2.1 Definition

A planned configuration group is a set of planned configurations or planned configuration groups that shall be processed (validated and activated) together.

For example, one plan may create a new cell, and another plan may start the collection of measurements for the new cell, or one plan may configure one base station and another plan may configure another base station, when both base stations need to be consistently configured to enable handovers between them.

6.2.2 Planned configuration group members

A planned configuration (or planned configuration group) belonging to a planned configuration group is also called a member of that planned configuration group. A planned configuration (or planned configuration group) can be a member of more than one planned configuration group.

The members of a planned configuration group shall be considered as atomic or non-atomic (best effort), in the same way as the operations of an operation set may form an atomic or a non-atomic (best effort) set.

Furthermore, the members of a planned configuration group can be ordered or not ordered. If ordered, the planned configuration group members are processed in the order as specified in the planned configuration group. If not ordered, the members can be processed in any order or in parallel.

For example, when the configuration for a new Network Function is in one plan and the creation of a performance metric collection job in another plan, then the plans must be ordered (assuming the job object is contained under a configuration object).

Not ordered are for example plans adding performance metric collection jobs to already configured Network Functions.

There shall not be any circular membership relationships between two or more planned configuration groups. E.g. group-A is a member of group-B then group-B is not allowed to be a member of group-A.

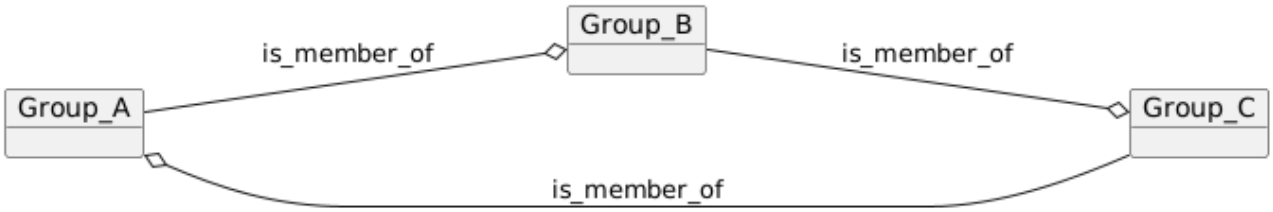

Figure 6.2.2-1: NOT allowed circular planed configuration group membership relation (with 3 groups)

A planned configuration group may not include any individual planned configuration more than once. The following is not allowed because Group_A cannot contain Plan2 twice, It is also invalid because Group_B contains Plan1 twice, once via Group_C and once via Group_A.

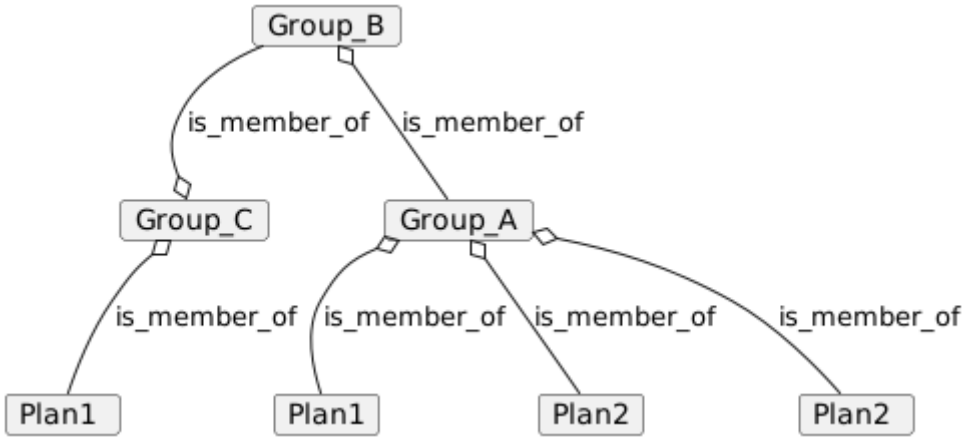

Figure 6.2.2-2: NOT allowed multiple containment of planned configurations

All member planned configurations shall have the same configuration content type.

6.2.3 Conflicts between planned configuration group members

Conflicting operations may be contained in planned configuration group members. For example, a planned configuration may set an attribute to some value and another planned configuration sets the same attribute to some other value.

If the configuration group members are ordered, the result is deterministic. The conflicting planned configurations are processed in the order as they appear in the planned configuration group. The last processed planned configuration determines the result. If the configuration group members are not ordered, the result is not guaranteed to be deterministic. Depending on which planned configuration group member is processed last, the result may be different.

Conflicting operations within a single planned configuration are not considered as conflicts.

It is configurable if a detected conflict invalidates the planned configuration group.

6.2.4 Meta data for planned configuration groups

Each planned configuration group is described by meta data. This includes annotations such as the name, the version, and a human readable textual description.

Furthermore, it includes the information if the group is atomic or non-atomic, if it is ordered or unordered, and if a conflict between planned configuration group members shall invalidate the planned configuration group and prevent its activation.

The date and time at which information in the planned configuration group was modified for the last time is provided as well.

6.2.5 Planned configuration group descriptor

The planned configuration group members and its meta data are specified in a planned configuration group descriptor. Each descriptor has a unique identifier.

6.3 Fallback configurations

6.3.1 Definition

Fallback is the process of returning to the configuration that was used by the managed system before the activation of a planned configuration or planned configuration group. A fallback configuration is the configuration describing this configuration.

Fallback configurations are generated by the system. Fallback may be applied when the activation of a planned configuration or a planned configuration group does not yield the desired results. Use cases include:

A wrong planned configuration was activated. The MnS consumer requests to return to the old configuration.

The new configuration does not perform as expected. The MnS consumer requests to return to the old configuration.

The new configuration was deployed on existing Network Functions in anticipation of some new Network Functions being rolled out. The roll out does not happen. The existing Network Functions are rolled back to the configuration state at the time the activation was triggered.

There are two types of fallback configurations:

- "undo": This type only reverses the changes resulting from the activation of a planned configuration or planned configuration group. It has no effect on configuration data nodes that are not affected by the activation of the planned configuration or planned configuration group. Therefore, changes to other data nodes, that occur after activation and before fallback, are not undone. The format of this type of fallback configuration is similar to a planned configuration.
- "complete restore": This type reverts all configuration changes done after the activation of a planned configuration or planned configuration group, i.e. those resulting from the activation of a planned configuration or planned configuration group and those done via some other means. This method restores the complete old configuration. Therefore, changes to data nodes, that occur after activation and before fallback, are undone as well. Note that the complete data node tree is restored in this mode. Therefore, this mode might not be supported in deployment scenarios with large data node trees. The format of this type of fallback configuration is implementation specific.

6.3.2 Fallback configuration preparation

A fallback configuration is created by the system upon the start of an activation process when enable fallback is requested. The format of the fallback configuration is dependent on the fallback method.

In case of "undo operations" the fallback configuration shall be visible to the consumer as a read-only planned configuration. A fallback configuration is always a single configuration (not a planned configuration group) and is atomic. For example, for a planned configuration that adds a new object to the current data node tree the fallback configuration simply removes this object. Note that the fallback operations are not always simply the opposite of the operations contained in the activated planned configuration. The fallback configuration is created on effective changes only, for example:

Only successful operations shall be included in the fallback configuration.

For removed objects, the fallback configuration needs to include the representation of the object in the current data node tree at the time of removal.

The removal of objects that do not exist cannot result in the addition of an object, leave alone that there would be no representation for that object.

The addition of an object that already exists shall not result in the removal of that object.

In case of "restore configuration" the fallback configuration is implementation specific.

6.3.3 Meta data for fallback configurations

Each fallback configuration is described by meta data. This includes annotations such as the name, the version, and a human readable textual description.

Furthermore, an identifier of the activation process whose result is to be rolled back must be provided.

A fallback is always atomic.

6.3.4 Fallback configuration descriptor

The fallback configuration and its meta data are specified in a fallback configuration descriptor. Each descriptor has a unique identifier.

A fallback configuration has the following properties:

- activation mode is always atomic (plan descriptor)
- must include the id of the activation job
- "configChangesContentType" is set by the MnS producer and is read-only (plan descriptor)
- cannot be a member of a planned configuration group (group descriptor)
- cannot be activated by trigger conditions (trigger conditions)
- fallback configurations (of the "complete restore" type) are not allowed to be validated (as they are always valid)
- no fallback is possible (activation)
- no cancelling is possible (activation)
- a pointer to the fallback configuration descriptor shall be included in the related activation job (activation)

6.4 Trigger conditions

6.4.1 Definition

A trigger condition is a set of definitions controlling the points in time when associated planned configurations and planned configuration groups are activated. The set of definitions typically includes expressions on data node trees that evaluate to true or false. Trigger conditions are evaluated permanently.

6.4.2 Usage

Trigger conditions are provided for conditional activation of planned configurations and planned configuration groups. The planned configuration (and/or group) is activated automatically when the provided conditions are met.

For example, additional performance metric collection jobs or assurance functions shall be created when metrics that are always collected cross certain thresholds.

6.4.3 Specification of trigger conditions

The trigger for activation is specified as follows:

a condition expression evaluating to true or false.

an evaluation period.

a hysteresis that may require the evaluation to yield true for a specified time or a specified number of times before the trigger is activated.

The condition expression works on the data node tree of the current configuration. The notation used to express the condition depends on the MnS protocol used. The transition of the evaluation result of the trigger condition from false to true is considered as a trigger. The transition from true to false is not considered as a trigger.

The evaluation period determines how often the trigger condition is evaluated.

The hysteresis may be used to ensure that the trigger is activated only when the trigger condition evaluates to true for a certain time or a certain number of times. This prevents the trigger from being activated also when the evaluation result toggles between false and true.

6.4.4 Meta data for trigger conditions

Each trigger condition is described by meta data. This includes annotations such as the name, the version, and a human readable textual description.

6.4.5 Control parameters for trigger conditions

The trigger may be activated each time the result of the trigger condition evaluation changes from false to true or only once. A control parameter allows to configure this behavior.

Furthermore, the evaluation of the trigger condition may be constrained in time by a start time and a stop time. Before the start time and after the stop time the evaluation result is set to false.

6.4.6 Monitor parameters for trigger conditions

Monitor parameters are provided that display the current result of the evaluation, the data and time at which the trigger was last activated, and a Boolean indicating if the trigger is active or inactive. The trigger is always inactive when the current time is before the start and after the stop time. After the start time and before the stop time the trigger is always active if the trigger is activated every time the condition evaluation result changes from false to true. If the trigger is activated only the at the first time the condition evaluation result changes from false to true, then the trigger becomes inactive upon the occurrence of this event.

6.4.7 Trigger condition descriptor

The trigger condition descriptor includes the specification of the trigger condition itself, and associated meta-data, control parameters and monitor parameters.

6.5 Validation of planned configurations

6.5.1 Definition

Planned configurations may have problems making it impossible to activate them. Validation is the process to discover potential problems with planned configurations before their activation.

The validation function has two input arguments: the planned configuration and a reference configuration which typically is a snapshot of the current configuration at a certain point of time. The planned configuration is applied to reference configuration. The resulting configuration data node tree must comply with the (stage 2) NRM definitions and (stage 3) NRM schema definitions.

6.5.2 Potential problems

Validation discovers problems of planned configurations. These problems may be classified as follows:

- Information model problems
    - Problems of the planned configuration itself
    - Problems of the planned configuration in relationship to the current configuration

- Application layer problems

Examples of problems with the planned configuration itself:

- Update of an attribute with a value that does not match the data type defined for the value of the attribute
- Creation of an attribute name value pair that is not defined
- Update of an existing attribute that is defined as read-only
- Creation of an object with an object class that is not defined
- Creation of an object with a representation that does not match the object class defined for the object
- Creation of an object under an object whose class does not support this containment

Examples of problems of the planned configuration in relationship to the current configuration:

- Creation of objects under parents that do not exist
- Violation of the cardinality property of an object
- Violation of the multiplicity property of an attribute

Examples of application layer problems

- An attribute value in the planned configuration is in the value range allowed by the attribute properties but the value is not allowed in the current state of the application.

6.5.3 Validation process

The purpose of validation is to detect problems of the planned configuration. As described in the previous clause, problems can be categorized into three groups.

Problems of the operations themselves are detected by analyzing the operations together with the NRM schema. The current configuration is not involved in this process. Problematic operations are always invalid. If an atomic operation set contains at least one invalid operation, the complete planned configuration is invalid. If a non-atomic operation set contains a bad operation, only that operation is invalid. The operation set with the other operations may still be valid, depending on the validation result for the other operations.

Problems of the planned configuration in relationship to the current configuration are detected by applying the operations of the operation set to a snapshot of the current configuration. This may result in intermediate states that do not comply to the constraints defined by the NRM schema. For example, assume the schema defines a constraint that an object has to contain exactly a certain number of child objects. When a plan contains an operation to remove one child object, and an operation to add a child object, there might be intermediate states at certain times during the execution of the validation with the constraint violated, as there can be too few or too many child objects. However, after validating all operations, the constraint is fulfilled again. The validation process needs to take this into account: Problems of the planned configuration in relationship to the current configuration may be temporary only and be healed later when more operations are processed. A final assessment on the validity of operations can be made only after processing all operations.

Problems of the planned configuration at application layer are detected by evaluating the updates specified by the operations in the context of the configuration that is produced by applying the planned configuration against the current configuration. As for problems of the planned configuration in relationship to the current configuration, a final assessment on the validity of operations can be made only after processing all operations.

All constraint violations shall be reported.

The result of a validation is valid only for the given reference configuration, i.e. in many use cases the conceptual snapshot of the current configuration at a certain point in time. The same validation could have a different result later for many reasons e.g. current configuration changes or network environment changes.

A successful validation does not guarantee a successful activation.

6.5.4 Validation of planned configuration groups

The planned configuration group members are validated in the order specified. If no order is specified, they can be validated in any order. Each member is validated as specified in this and the previous clause. The validation of atomic members concludes with either validated or failed. The validation of best effort members concludes with either validated, partially validated or failed.

An atomic group is considered as valid, if atomic members are valid, and best effort members are valid or partially valid. If the validation of one (atomic or best effort) member fails, the validation of the complete group shall fail.

A best effort group is considered as valid, if atomic members are valid, and best effort members are valid or partially valid. If the validation of at least one (atomic or best effort) member fails, the group is considered as partially valid. If the validation of all members fails, the validation of the complete group shall fail.

6.5.5 Validation of fallback configurations

When the fallback configuration is the complete old configuration, the complete current configuration is replaced when activating the fallback configuration. In this case the concept of validation has no meaning. The complete old configuration is always valid. Therefore, the fallback configuration should not be validated when the fallback configuration is represented by the complete old configuration.

When the fallback configuration is a set of undo operations, the fallback configuration should be validated like a planned configuration prior to activation.

6.5.6 Validation control parameters

The "validationMode" parameter is provided for specifying the validation mode, which can take one of two values: continue on error or stop on error.

In the stop on error mode the validation process stops upon the detection of the first invalid operation. In the continue on error mode the validation process continues upon the detection of invalid operations until all operations of the operation set are processed.

Validation jobs can be cancelled. A specific control parameter is provided for this purpose. A cancelled validation shall provide an (execution) summary and any problems found until cancellation.

6.6 Activation of planned configurations

6.6.1 Definition

Activation is the process of applying a planned configuration to the managed system.

6.6.2 Potential problems

A planned configuration is validated before activation. Therefore a planned configuration that is activated should not have any of the problems that are discovered by validation. Only valid or partially valid planned configurations or planned configuration groups should be activated. Therefore, other types of problems are encountered during activation. These include

A Network Function cannot be updated, because it is not reachable.

A Network Function cannot be updated, because it is locked.

A Network Function cannot be updated, because it does not exist.

A Network Function cannot be updated, because it lacks necessary resources.

A Network Function cannot be updated, because necessary access rights are not in place.

6.6.3 Activation process

Activation is the process of applying the updates specified by the operations contained in a planned configuration to the managed system. After the managed system is updated, the updates are reflected in the current configuration. The sequence of the operations while activating a planned configuration depends on the internal implementation of the MnS producer and on its internal state. This is an important benefit compared to command-by-command configuration, because implementation details remain private to the MnS producer, while the MnS consumer needs to know the target state only.

In atomic mode, the system may have to roll back to the current configuration that was effective when the activation process started. Therefore, the system needs to take and store a (conceptual) snapshot of the impacted part of the current configuration upon starting the activation process.

If not in atomic mode, the activation process may stop or continue when the first error occurs.

The activation process may run for quite some time, for example when large configurations are downloaded to Network Functions as part of the activation process.

The planned configuration to be activated should be validated first.

Problems related to the activation of a planned configuration shall be reported.

6.6.4 Activation of planned configuration groups

The planned configuration group members are activated in the order specified. If no order is specified, they can be activated in any order. Each member is activated as specified in the previous clause. The activation of atomic members concludes with either activated or failed. The activation of best effort members concludes with either activated, partially activated or failed.

An atomic group is considered as activated, if atomic members can be activated, and best effort members can be activated or partially activated. If the activation of one (atomic or best effort) member fails, the activation of the complete group shall fail (and all updates must be undone).

A best effort group is considered as activated, if atomic members can be activated, and best effort members can be activated or partially activated. If the activation of at least one (atomic or best effort) member fails, the group is considered as partly activated. If the activation of all members fails, the activation of the complete group shall fail (and all updates must be undone).

6.6.5 Activation control parameters

A couple of control parameters determine the behavior of an activation process.

Activation can be started immediately upon creation of the job, or when triggered by a trigger condition. The behaviour is set by a control parameter.

Activation is typically a trade-off between speed of activation and service impact. Both targets cannot be minimized at the same time. For example, to speed up the activation process many data node tree portions for Managed Elements may be handled in parallel, but if the planned configuration is bad the service provided by many Managed Elements is degraded or fails at the same time and must be rolled back to the old configuration. Activating the new configuration for the Managed Elements one by one slows down the speed of activation but limits the number of Managed Elements with bad configurations if the planned configuration is bad. The preference (speed or service impact) is indicated by a control parameter.

Another control parameter allows the MnS consumer to request the MnS producer to create a fallback configuration for potential later activations. The MnS producer will create a fallback configuration on explicit request only.

Activation jobs can be cancelled. A specific control parameter is provided for this purpose. Cancelling an activation job stops the activation of operations at the next point possible. Changes for atomic planned configurations are rolled back (undone).

6.7 Conditional activation of planned configurations

Planned configurations and planned configuration groups can be activated based on trigger conditions. A trigger condition holds the information, which activation job is triggered by it.

6.8 Asynchronous configuration updates

Updating the current configuration may take some time, more time than an update operation based on a synchronous request response pattern, where the request provides the desired updates and the response information about the outcome of the requested updates, may be able to cope with. In a synchronous request-response pattern, the response is sent shortly after reception of the request.

This problem is addressed by so-called asynchronous update operations, where the request response pattern just provides the desired updates to the MnS producer. The response does not contain information about the outcome but a confirmation of reception of the desired updates. The information about the outcome of the update operations is provided asynchronously, i.e. later when the information is available.

Management of planned configuration includes as a special case also an asynchronous update operation. It is not mandatory to create a planned configuration before validation or activation. A new configuration can be provided also together with the validation or activation request. This is conceptually equivalent to an asynchronous update operation.

6.9 Managing planned configurations

Planned configuration descriptors can be created, replaced completely and deleted by MnS consumers on MnS producers. The capability to partially update a planned configuration descriptor by a MnS consumer is an optional feature of an MnS producer, that is not required in all deployment scenarios. This might be the case when the operator updates the planned configuration descriptor on the MnS producer side using an appropriate GUI, or when the planned configurations are rather small.

Multiple planned configuration descriptors can be maintained on a MnS producer. Each planned configuration is identified by an identifier. The identifier is assigned by the MnS producer.

Planned configurations cannot be deleted when being validated or activated, and when being referenced in planned configuration groups or trigger conditions.

6.10 Managing planned configuration groups

Planned configuration group descriptors can be created, replaced completely and deleted by MnS consumers on MnS producers. The capability to partially update a planned configuration group descriptor by a MnS consumer is an optional feature of an MnS producer, that is not required in all deployment scenarios. This might be the case when the operator updates the planned configuration group descriptor on the MnS producer side using an appropriate GUI, or when the planned configurations are rather small.

The planned configurations and planned configuration groups referred to as members in a planned configuration group shall exist.

Multiple planned configuration group descriptors can be maintained on a MnS producer. Each planned configuration group is identified by an identifier. The identifier is assigned by the MnS producer.

6.11 Managing fallback configurations

Fallback configuration descriptors are not directly managed by MnS consumers. They are created automatically by MnS producers when the "enableFallback" attribute in an activation job is set to true. Only the "name", "version", and "description" can be set by MnS consumers once the object is created. Fallback configurations cannot be modified by MnS consumers. Each fallback configuration is identified by an identifier. The identifier is assigned by the MnS producer.

When many fallbacks are enabled, the situation can become soon complex and the risk for misconfigurations increases. MnS producers may therefore apply implementation specific strategies to reduce the risk for misconfigurations. For

example, they may allow only for one fallback configuration to exist; if another fallback is enabled, the existing fallback configuration is overwritten. Or they may allow for multiple fallback configurations only if they affect different parts of the network.

Fallback configurations can be deleted by MnS consumers, or by the MnS producer based on implementation specific policies.

No fallback on the activation of a fallback configuration is possible.

6.12 Managing trigger conditions

Conditions are created, replaced completely and deleted by MnS consumers on MnS producers. For each condition shall be specified which activation jobs are triggered by this condition. The capability to partially update a trigger condition by a MnS consumer is an optional feature of an MnS producer. Each trigger condition is identified by an identifier. The identifier is assigned by the MnS producer.

6.13 Managing validations

Validation of a planned configuration is requested by MnS consumers and performed by MnS producers. To request validation the MnS consumer needs to create a validation job on the MnS producer. Each job is identified by an identifier. The identifier is assigned by the MnS producer.

The MnS consumer may annotate the job with a "name", a textual human-readable "description" and his own identifier "mnsConsumerId".

The planned configuration to be validated is specified by including the identifier of the corresponding planned configuration descriptor in the job creation request. Alternatively, for validating a planned configuration group, a planned configuration group descriptor can be specified.

In case no planned configuration descriptor or no planned configuration group descriptor has been created for the planned configurations to be validated, a descriptor may be specified directly in the validation job creation request.

The validation mode meta data specifies whether the validation stops on the first error found or runs till the full planned configuration(group) is validated, potentially listing multiple errors.

The MnS producer adds the following information elements to the job: "jobState", "jobDetails", "currentConfigTime", "startedAt", "stoppedAt", "validationState", and "validationDetails".

The information element "error" is a structured data type with the following properties:

- The "type" property that provides high level error information.
- The "title" that provides a short, human-readable summary of the problem type. It shall not change from occurrence to occurrence of the problem.
- The "reason" property" that provides more details on the error conditions than the "type" property.
- The "detail" property that provides a human readable explanation specific to this occurrence of the problem.
- The "errorInfo" property that provides any additional error information.
- The "badDataNode" property that specifies the data node to which the operation could not be applied.

The "type" property is an enumeration of string values. A MnS producer should use the following values. Other values may be used as well if deemed more appropriate for specific errors.

- SCHEMA_VALIDATION_ERROR: Content in the operation request does not validate or the data node tree of the current configuration after applying the operation becomes invalid. Therefore, the operation cannot be processed.
- DATA_NODE_TREE_ERROR: The assumptions on the state of the data node tree of the current configuration as expressed in "path" do not match the real state (for example, a data node referred to in "path" that must exist does not exist). Therefore, the operation cannot be processed. The operation request is valid, though.

- MODIFICATION_NOT_ALLOWED: The operation is well formed and understood by the MnS producer but not allowed.
- ACCESS_CONTROL_CONFLICT: The MnS consumer is not allowed to perform the requested operation, because it does not have the required access rights.
- APPLICATION_LAYER_ERROR: The operation is well formed and understood by the MnS producer, but the MnS producer cannot satisfy the request due to application layer issues.
- SERVER_ERROR: The operation is well formed and understood by the MnS producer, but the MnS producer cannot process the operation due to an error in the server.

For each "type" value several reasons are provided:

SCHEMA_VALIDATION_ERROR

- NEW_DATA_NODE_NAME_INVALID: The data node cannot be created as requested, because its name, as specified as last segment of "path", is invalid at the target location.
- NEW_DATA_NODE_VALUE_INVALID: The data node specified in the "path" property cannot be created or cannot be updated as requested, because the value specified in "value" is invalid.
- NEW_DATA_NODE_CONTAINMENT_INVALID: The data node cannot be created under the specified parent data node as requested, because this containment is not allowed.
- FINAL_DATA_NODE_VALUE_INVALID: The current value of the target data node cannot be modified as requested, because this would result in an invalid value of the target data node. The value in “value” itself is valid.
- FINAL_DATA_NODE_UNIQUENESS_INVALID: The data node cannot be updated or created as requested, because this would result in violating uniqueness constraints. The value in “value” itself is valid. This error reason applies to multi-valued data nods only.
- FINAL_DATA_NODE_MULTIPLICITY_INVALID: The data node cannot be created or deleted as requested, because this would result in violating multiplicity constraints. The value in “value” itself is valid. This error reason applies to multi-valued data nods only.
- FINAL_DATA_NODE_CARDINALITY_INVALID: The data node cannot be created or deleted as requested, because this would result in violating cardinality constraints. The value in “value” itself is valid.

DATA_NODE_TREE_ERROR

- TARGET_DATA_NODE_NOT_FOUND: The target data node specified by “path” does not exist in the current configuration. Therefore, the “merge” operation cannot be applied as requested.
- TARGET_DATA_NODE_PARENT_NOT_FOUND: The parent of the target data node specified by “path” does not exist in the current configuration. Therefore, the target data node cannot be created as requested by the "create" operation.
- TARGET_DATA_NODE_FOUND: The target data node specified by “path” exists in the current configuration. Therefore, the target data node cannot be created as requested by the "create" operation. The value of the target data node in the planned configuration and the value of that node in the current configuration may or may not be identical.

MODIFICATION_NOT_ALLOWED

- TARGET_DATA_NODE_NOT_WRITABLE: The data node cannot be modified as requested, because the data node is not writable by MnS consumers.
- TARGET_DATA_NODE_INVARIANT: The data node cannot be modified as requested, because the data node is invariant.
- TARGET_DATA_NODE_CREATION_NOT_ALLOWED: The data node cannot be created by MnS consumers.

- TARGET_DATA_NODE_DELETION_NOT_ALLOWED: The data node cannot be deleted by MnS consumers.

ACCESS_CONTROL_CONFLICT

- ACCESS_DENIED: Access to the target data node is denied as a result of access control.

APPLICATION_LAYER_ERROR

SERVER_ERROR

Multiple validation jobs can run in parallel, even if the planned configurations target the same portions of the current configuration.

To provide a consistent result, a planned configuration or planned configuration group (incl. its members) cannot be modified during validation, or more precisely when the "jobState" of the validation job has either of the following values: "RUNNING", "CANCELLING". Modification requests shall be rejected.

6.14 Managing activations

Activation of a planned configuration is requested by MnS consumers and performed by MnS producers. To request activation the MnS consumer needs to create an activation job on the MnS producer. Each job is identified by an identifier. The identifier is assigned by the MnS producer.

The planned configuration to be activated is specified by including the identifier of the corresponding planned configuration descriptor in the job creation request. Alternatively, for activating a planned configuration group, a planned configuration group descriptor can be specified.

In case no planned configuration descriptor or no planned configuration group descriptor has been created for the planned configurations to be activated, a descriptor may be specified also directly in the activation request.

For falling back to an old configuration, the identifier of a fallback configuration descriptor needs to be specified.

Activation shall start immediately upon creation of the activation job, unless the "isImmediateActivation" parameter is set to false. In this case the activation is triggered by a trigger condition that points to the activation job. Note that the activation job does not know the related conditions.

For immediate activation an activation job can activate a planned configuration only once. If the planned configuration is to be activated a second time a new job needs to be created. For conditional activation, in contrast, an activation job can activate the same planned configuration multiple times. For example, if a planned configuration is for weekdays and another planned configuration is for weekends, two activation jobs can be used for activating both planned configurations. One job is triggered by a trigger condition triggering the activation when the weekend starts, the other job is triggered by a trigger condition when the weekdays start.

Furthermore, the following control parameters need to be specified in the job creation request: "isEnableFallback", and "serviceImpact", and "isImmediateActivation".

The MnS producer adds the following information elements to the job: "state", "startedAt", "stoppedAt", "result", and "errors".

Only planned configurations, that are successfully validated, should be activated. Therefor a MnS consumer should always create a validation job first. Only after successful validation of a planned configuration, a job for activating the planned configuration should be created. In case the planned configuration was not validated the MnS producer shall create a validation job for the planned configuration. The activation is started only if the planned configuration is valid. If the planned configuration is invalid the activation is not started and the result attribute of the activation indicates that the activation failed.

A MnS consumer can request a MnS producer to cancel a running activation process. The behavior depends on the application mode:

For "ATOMIC" the configuration is rolled back.

For "BEST_EFFORT" the activation is stopped.

For "STOP_ON_ERROR" the activation is stopped

As for validation, a planned configuration or planned configuration group (incl. its members) cannot be modified or deleted during its activation, or more precisely when the "jobState" of the activation job has either of the following values: "RUNNING", or "CANCELLING".

Multiple activation jobs can run simultaneously if the planned configurations do not affect the same portions of the current configuration. However, if they do affect the same portion, they cannot run in parallel and shall be queued.

If a new configuration is provided together with the activation request, the MnS producer shall create automatically a planned configuration descriptor or planned configuration group descriptor and replace the inline specification of the planned configuration or planned configuration group with a reference to the newly created descriptor.

6.15 Managing conditional activations

The MnS consumer shall create an activation job as described in clause 6.14 and a trigger condition as described in clause 6.12. The "isImmediateActivation" control parameter of the activation job shall be set to "FALSE". The "activationJobs" parameter in the trigger condition shall specify the activation jobs to be activated.

6.16 Notifications

Notifications are not specified.

7 Information model

7.1 PlannedConfigurationDescriptor

7.1.1 Definition

This definition represents a planned configuration descriptor.

7.1.2 Information Elements

The following table specifies the information elements of a planned configuration descriptor.

Table 7.1.2-1: Information elements of the "PlannedConfigurationDescriptor"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the planned configuration.	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the planned configuration.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
version	M	The version of the planned configuration. Its format is implementation specific.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description	M	The textual human-readable description of the planned configuration.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
customProperties	M	The container allowing to specify additional consumer defined properties (key value pairs) describing and qualifying the planned configuration.	type: String multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
configChangesContentType	M	The format of the planned configuration in "planConfig". The format depends on the solution set of the current configuration. allowedValues: - OPENAPI_BASED - YANG_BASED	type: ENUM multiplicity: 1 isInvariant: True isWritable: True
configChanges	M	The operation set specifying the planned configuration. The format of "configChanges" is specified in "configChangesContentType"	type: ConfigChange multiplicity: 1..* isInvariant: False isWritable: True
activationMode	M	Specifies if the operations in the operation set are treated for activation as atomic, best effort, or if processing shall stop on the occurrence of the first error. "ATOMIC": Either all or none of the operations are executed. In case of error the already executed operations should be rolled back. However, there is no guarantee that the rollback will succeed. "BEST_EFFORT": In case of an error all further operations that are possible to execute are attempted to be executed. "STOP_ON_ERROR": In case of an error no further operations are attempted. allowedValues: - ATOMIC - BEST_EFFORT - STOP_ON_ERROR	type: ENUM multiplicity: 1 isInvariant: False isWritable: True
lastModifiedAt	M	The date and time at which the planned configuration descriptor was modified the last time by a MnS consumer. Upon creation of the planned configuration descriptor the value of the information element is set to the date and time at which the descriptor is created.	type: DateTime multiplicity: 1 isInvariant: False isWritable: False
lastValidatedAt	M	The date and time at which the planned configuration was validated (the last time). The information element is absent, when the planned configuration has not been validated yet.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False

Information element name	S	Documentation and Allowed Values	Properties
validationState	M	The outcome of the last validation of the planned configuration ("VALID", "INVALID"), or "NOT_VALIDATED" if the planned configuration has not been validated yet. The state is reset to "NOT_VALIDATED", if in another state, when "configChanges" is updated. allowedValues: - NOT_VALIDATED - VALID - INVALID	type: ENUM multiplicity: 1 isInvariant: False isWritable: False

7.1.3 Data types

7.1.3.1 ConfigChange

Table 7.1.3.1-1: Information elements of the "ConfigChange"

Information element name	S	Documentation and Allowed Values	Properties
target	M	The target node of the operation.	type: String multiplicity: 1 isInvariant: False isWritable: False
modifyOperator	M	The modify operator applied to the target node. allowedValues: - create - merge - merge-create - delete	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
value	M	The value applied to the target node, which follows the NRM schema of the target. The "value" shall be present unless the "modifyOperator" is "DELETE".	type: Object multiplicity: 0..1 isInvariant: False isWritable: False
description	O	The textual human-readable description of the operation.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
changeId	O	The identifier of the operation. It may or may not be provided If provided, it shall be unique within an instance of "configChanges" and it shall be provided for all changes in "configChanges".	type: String multiplicity: 0..1 isInvariant: False isWritable: False

7.2 PlannedConfigurationGroupDescriptor

7.2.1 Definition

This definition represents a planned configuration group descriptor.

7.2.2 Information Elements

The following table specifies the information elements of a planned configuration group descriptor.

Table 7.2.2-1: Information elements of the "PlannedConfigurationGroupDescriptor"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the planned configuration group descriptor	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the planned configuration group.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
version	M	The version of the planned configuration group. Its format is implementation specific.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description	M	The textual human-readable description of the planned configuration group.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
customProperties	O	The container allowing to specify additional consumer defined properties (key value pairs) describing and qualifying the planned configuration.	type: Object multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
members	M	The identifiers of planned configurations or the planned configuration groups that are members of this planned configuration group.	type: Member multiplicity: 1..* isOrdered: True isUnique: True isInvariant: False isWritable: True
isOrdered	M	Specifies if the members of the planned configuration group are ordered. When ordered, the planned configuration group members shall be validated/activated in the specified order. When not ordered the planned configuration group members can be validated/activated in any order.	type: Boolean multiplicity: 1 isInvariant: False isWritable: True defaultValue: true
applyMode	M	Specifies if the members of the planned configuration group are treated as atomic, best effort, or if processing shall stop on the occurrence of the first error. allowedValues: - ATOMIC - BEST_EFFORT - STOP_ON_ERROR	type: ENUM multiplicity: 1 isInvariant: False isWritable: True defaultValue: BEST_EFFORT
isFailOnMemberConflicts	M	Specifies if the activation shall fail on detection of conflicts between planned configuration group members, or if the operations shall be processed as if there were no conflicts.	type: Boolean multiplicity: 1 isInvariant: False isWritable: True defaultValue: true
lastModifiedAt	M	The date and time at which the planned configuration group descriptor was modified the last time by a MnS consumer. Upon creation of the planned configuration group descriptor the value of the information element is set to the date and time at which the descriptor is created.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
lastValidatedAt	M	The date and time at which the planned configuration group was validated (the last time). The information element is absent, when the planned configuration has not been validated yet.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
validationState	M	The outcome of the last validation, or "NOT_VALIDATED" if the planned configuration group has not been validated yet. The state is reset to "NOT_VALIDATED", if in another state, when "members" is updated. allowedValues: - NOT_VALIDATED - VALID - PARTIALLY_VALID - INVALID	type: ENUM multiplicity: 1 isInvariant: False isWritable: False

7.2.3 Data types

7.2.3.1 Member

Information element name	S	Documentation and Allowed Values	Properties
planConfigDescrId	M	The identifiers of planned configuration descriptors.	type: String multiplicity: 1..* isInvariant: False isWritable: False
planConfigGroupDescrId	M	The identifiers of planned configuration descriptors.	type: String multiplicity: 1..* isInvariant: False isWritable: False

7.3 FallbackConfigurationDescriptor

7.3.1 Definition

This definition represents a fallback configuration descriptor.

7.3.2 Information Elements

The following table specifies the information elements of a fallback configuration descriptor.

Table 7.3.2-1: Information elements of the "FallbackConfigurationDescriptor"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the fallback configuration descriptor	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the fallback configuration.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
version	M	The version of the fallback configuration. Its format is implementation specific.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description	M	The textual human-readable description of the fallback configuration.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
customProperties	M	The container allowing to specify additional consumer defined properties (key value pairs) describing and qualifying the planned configuration.	type: Object multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
activationJob	M	The identifier of the related activation job.	type: String multiplicity: 1 isInvariant: True isWritable: False
configurationContentType	M	The format of the fallback configuration. For fallbacks of the "undo" type the same values shall be used as for PlannedConfigurationDescriptor.configChangesContentType.	type: String multiplicity: 1 isInvariant: True isWritable: False
fallbackConfig	M	The fallback configuration in case the fallback configuration is of the "complete restore" type. Exactly one of this element or the configChanges shall be present.	type: Any multiplicity: 0..1 isInvariant: True isWritable: False
configChanges	M	The operation set specifying the configuration in case the fallback confiiguration is of the "undo" type. The format of "configChanges" is specified in "configurationContentType". Exactly one of this element or the fallbackConfig shall be present.	type: ConfigChange multiplicity: 0..* isInvariant: False isWritable: True

7.4 TriggerConditionDescriptor

7.4.1 Definition

This definition represents a trigger condition descriptor.

7.4.2 Information Elements

The following table specifies the information elements of a trigger condition descriptor.

Table 7.4.2-1: Information elements of the "TriggerConditionDescriptor"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the trigger condition descriptor.	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the trigger condition.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
version	M	The version of the planned configuration. Its format is implementation specific.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description	M	The textual human-readable description of the trigger condition.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
customProperties	M	The container allowing to specify additional consumer defined properties (key value pairs) describing and qualifying the planned configuration.	type: String multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
conditionExpression	M	The condition expression.	type: String multiplicity: 1 isInvariant: False isWritable: True
evaluationPeriod	M	The evaluation period specifies the interval of time between two consecutive condition expression evaluations. The unit is seconds.	type: Integer multiplicity: 1 isInvariant: False isWritable: True
hysteresis	M	The hysteresis, when present, specifies that the trigger shall not be activated immediately, when the evaluation result changes from false to true, but only when the evaluation result is true for a specified number of times. allowedValues: Integers greater or equal to 1.	type: Integer multiplicity: 0..1 isInvariant: False isWritable: True
isTriggerOnce	M	The boolean indication, if the trigger is disarmed after the first firing.	type: Boolean multiplicity: 1 isInvariant: False isWritable: True defaultValue: true
activationJobs	M	The identifiers of one or more activation jobs that shall be triggered by this condition.	type: String multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
startEvaluationAt	M	The date and time at which the evaluation of the condition expression shall start. The evaluation result is set to "False" before that date and time. If the information element is not specified, evaluation of the trigger condition shall start immediately.	type: DateTime multiplicity: 0..1 isInvariant: True isWritable: True
stopEvaluationAt	M	The date and time at which the evaluation of the condition expression shall stop. The evaluation result is set to "False" after that date and time. If the information element is not specified, evaluation of the trigger condition shall continue until the deletion of the trigger condition descriptor.	type: DateTime multiplicity: 0..1 isInvariant: True isWritable: True
lastModifiedAt	M	The date and time at which the trigger condition was modified the last time by a MnS consumer. Upon creation of the trigger condition descriptor the value of the information element is set to the date and time at which the descriptor is created.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
currentEvaluationResult	M	The current result of evaluating the "condition-expression".	type: Boolean multiplicity: 1 isInvariant: False isWritable: False

Information element name	S	Documentation and Allowed Values	Properties
lastTriggeredAt	M	Th date and time at which the evaluation result of the trigger condition changed the last time from "False" to "True". The information element is absent or contains no information if the evaluation result never changed from "False" to "True".	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
isTriggerActive	M	The indication if the trigger can start activation jobs (trigger is active), or if the trigger cannot start activation jobs (trigger is inactive).	type: Boolean multiplicity: 1 isInvariant: False isWritable: False

7.4.3 Data types

7.4.3.1 Hysteresis

Information element name	S	Documentation and Allowed Values	Properties
timeOfTrueEvaluations	M	The hysteresis, when present, specifies that the trigger shall not be activated immediately, when the evaluation result changes from false to true, but only when the evaluation results is true for a specified time (which must be a multiple of the evaluation period). Unit is seconds	type: Integer multiplicity: 0..1 isInvariant: False isWritable: True
numberOfTrueEvaluations	M	This information element, when present, specifies that the trigger shall not be activated immediately, when the evaluation result changes from false to true, but only when the evaluation results is true for a specified time (which must be a multiple of the evaluation period) or a specified number of times.	type: Integer multiplicity: 0..1 isInvariant: False isWritable: True

7.5 ValidationJob

7.5.1 Definition

This definition represents a validation job.

7.5.2 Information Elements

The following table specifies the information elements of a validation job.

Table 7.5.2-1: Information elements of the "ValidationJob"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the validation job.	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the validation job.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description	M	The textual human-readable description of the validation job.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
mnsConsumerId	M	The consumer that created the job. It may indicate a human user and/or one or more applications, for example ["userid:janedoe", "appid:12314"].	type: String multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
planConfigDescrId	M	The identifier of the planned configuration descriptor, whose operation set is requested to be validated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigGroupDescrId	M	or, alternatively, the identifier of the planned configuration group descriptor, whose operation sets are requested to be validated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigDescr	O	or, alternatively, the planned configuration descriptor to be validated, if no planned configuration descriptor was created earlier.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigDescrGroup	O	or, alternatively, the planned configuration group descriptor to be validated, if no planned configuration group descriptor was created earlier.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
fallbackConfigDescrId	O	or, alternatively, the identifier of a fallback configuration. Only fallback configurations of the "undo" type can be validated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
validationMode		The validation mode. allowedValues: - CONTINUE_ON_ERROR - STOP_ON_ERROR	type: ENUM multiplicity: 1 isInvariant: True isWritable: True defaultValue: CONTINUE_ON_ERROR
cancelRequest	O	This boolean information element allows to request to cancel the activation process by setting its value to "True". Setting the value to "False" has no observable result. When the value is set to "True" it cannot be changed any more.	type: Boolean multiplicity: 1 isInvariant: False isWritable: True
jobState	M	The validation job state. allowedValues: - NOT_STARTED - RUNNING - CANCELLING - CANCELLED - COMPLETED - FAILED	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
jobDetails	M	Detailed information related to the job, including job related errors.	type: String multiplicity: * isInvariant: False isWritable: False
currentConfigTime	M	The date and time of the current configuration state against which the planned configuration or planned configuration group is validated.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False

Information element name	S	Documentation and Allowed Values	Properties
startedAt	M	The date and time at which the validation process started, i.e. the time when the job state transition from "NOT_STARTED" to "RUNNING" occurred. In the "NOT_STARTED" state the information element is absent or carries no information.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
stoppedAt	M	The date and time at which the validation process stopped, i.e. the job state transition from "RUNNING" to "COMPLETED" or "FAILED", or from "CANCELLING" to "CANCELLED". In the "NOT_STARTED", "RUNNING" or "CANCELLING" state the information element is absent or carries no information.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
validationState	M	The current validation state of the planned configuration or planned configuration group that is processed by the validation job. - UNKNOWN: validation result is unknown because the validation has not started yet or is still ongoing. The "jobState" provides further qualifications. - VALIDATION_SUCCEEDED: all operations are validated and are valid. VALIDATION_FAILED: at least one operation was not validated or is invalid. allowedValues: - UNKNOWN - VALIDATION_SUCCEEDED - VALIDATION_FAILED	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
validationDetails	M/	Details of the validation of the operations that are contained in the planned configuration or planned configuration group.	type: ExecutionDetails multiplicity: 1 isInvariant: False isWritable: False

7.5.3 Data types

7.5.3.1 JobDetails

Information element name	S	Documentation and Allowed Values	Properties
jobDetails		Detailed information related to the job.	type: String multiplicity: * isInvariant: False isWritable: False

7.5.3.2 ExecutionDetails

Information element name	S	Documentation and Allowed Values	Properties
summary	M	This information element provides a summary of the validation or activation results.	type: Summary multiplicity: 1 isInvariant: False isWritable: False
results	M	The validation or activation results for each operation.	type: Result multiplicity: * isInvariant: False isWritable: False
memberConflicts	M	If a planned configuration group is validated or activated, then this information element is used to report detected conflicts between members. If no conflicts are detected, this information element is absent or carries no information. If a planned configuration is validated or activated, then this information element is always absent or carries no information. For validations, this information element is a warning. Detected member conflicts are just reported and do not impact the validation process. The details of the validation result are reported in "summary" and "results". For activations, if "isFailOnMemberConflicts" is false, this information element is a warning. Detected member conflicts are just reported and do not impact the activation process. The details of the activation result are reported in "summary" and "results". For activations, if "isFailOnMemberConflicts" is true, and a member conflict is detected, the activation is not started. The "state" for each result in "results" is set to "NOT_STARTED", and consequently "notFinished" in "summary" is set to the total number of operations in "configChanges".	type: MemberConflict multiplicity: 0..* isInvariant: False isWritable: False

7.5.3.3 Summary

Information element name	S	Documentation and Allowed Values	Properties
notFinished	M	The number of operations not yet started to be processed or currently being processed.	type: Integer multiplicity: 1 isInvariant: False isWritable: False
succeeded	M	The number of successful operations.	type: Integer multiplicity: 1 isInvariant: False isWritable: False
failed	M	The number of failed operations.	type: Integer multiplicity: 1 isInvariant: False isWritable: False
rollBackSucceeded	M	The number of operations for which the roll back succeeded.	type: Integer multiplicity: 1 isInvariant: False isWritable: False
rollBackFailed	M	The number of operations for which the roll back failed.	type: Integer multiplicity: 1 isInvariant: False isWritable: False
conflicting	M	The number of operations in conflict. Each conflict involves at least two operations.	type: Integer multiplicity: 1 isInvariant: False isWritable: False

7.5.3.4 Result

Information element name	S	Documentation and Allowed Values	Properties
planConfigDescrId	M	If planned configuration groups are activated, this information elements specifies the planned configuration descriptor identifier, for which error details are reported. If a planned configuration is activated or validated, this information element is absent.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
changeId	M	The identification of the operation. It is the identifier ("changeId") provided by the MnS consumer. Exactly one of "changeId" or "changeIndex" shall be provided.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
changeIndex	M	The identification of the operation. It is the positional index of the operation in the operation set ("changeIndex"). The positional index of the leftmost element is "0". Exactly one of "changeId" or "changeIndex" shall be provided. allowedValues: non-negative integer	type: Integer multiplicity: 0..1 isInvariant: False isWritable: False
state	M	The state of the operation activation. allowedValues: - NOT_STARTED - PROCESSING - SUCCEEDED - FAILED	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
errors	M	The error details for the "FAILED" state. In all other states the information element is absent or carries no information.	type: Error multiplicity: * isInvariant: False isWritable: False

7.5.3.5 Error

Information element name	S	Documentation and Allowed Values	Properties
type	M	High level error information. allowedValues: - SCHEMA_VALIDATION_ERROR - DATA_NODE_TREE_ERROR - MODIFICATION_NOT_ALLOWED - ACCESS_CONTROL_CONFLICT - APPLICATION_LAYER_ERROR - SERVER_ERROR - OTHER	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
title	M	A short, human-readable summary of the problem type. It shall not change from occurrence to occurrence of the problem. In other words, each type is mapped to one and only one title.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
reason	M	Further qualification of the "type". allowedValues: - NEW_DATA_NODE_NAME_INVALID - NEW_DATA_NODE_VALUE_INVALID - NEW_DATA_NODE_CONTAINMENT_INVALID - FINAL_DATA_NODE_VALUE_INVALID - FINAL_DATA_NODE_UNIQUENESS_INVALID - FINAL_DATA_NODE_MULTIPLICITY_INVALID - FINAL_DATA_NODE_CARDINALITY_INVALID - TARGET_DATA_NODE_NOT_FOUND - TARGET_DATA_NODE_PARENT_NOT_FOUND - TARGET_DATA_NODE_FOUND - TARGET_DATA_NODE_NOT_WRITABLE - TARGET_DATA_NODE_INVARIANT - TARGET_DATA_NODE_CREATION_NOT_ALLOWED - TARGET_DATA_NODE_DELETION_NOT_ALLOWED - ACCESS_DENIED - OTHER	type: ENUM multiplicity: 0..1 isInvariant: False isWritable: False
detail	M	A human-readable explanation specific to this occurrence of the problem.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
errorInfo	M	Any additional error information.	type: Any multiplicity: 0..1 isInvariant: False isWritable: False
badDataNode	M	The path identifying the data node to which the operation could not be applied. The value of "path" may be identical to the value of the "path" property in the request or may contain additional components identifying data nodes in the value of the "value" property of the request.	type: String multiplicity: 1 isInvariant: False isWritable: False

7.5.3.6 MemberConflict

Information element name	S	Documentation and Allowed Values	Properties
memberConflict	M	The identification of two or more operations that have a conflict.	type: MemberOp multiplicity: 2..* isOrdered: False isUnique: True isInvariant: False isWritable: False

7.5.3.7 MemberOp

Information element name	S	Documentation and Allowed Values	Properties
planConfigDescrId	M	The identifier of a planned configuration descriptor that contains on operation involved in a conflict.	type: String multiplicity: 1 isInvariant: False isWritable: False
changeId	M	The identification of the operation. It is the identifier ("changeId") provided by the MnS consumer. Exactly one of "changeId" or "changeIndex" shall be provided.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
changeIndex	M	The identification of the operation. It is the positional index of the operation in the operation set ("changeIndex"). The positional index of the leftmost element is "0". Exactly one of "changeId" or "changeIndex" shall be provided. allowedValues: non-negative integer	type: Integer multiplicity: 0..1 isInvariant: False isWritable: False
target	M	Target of the change	type: String multiplicity: 1 isInvariant: False isWritable: False

7.6 ActivationJob

7.6.1 Definition

This definition represents an activation job.

7.6.2 Information Elements

The following table specifies the information elements of an activation job.

Table 7.6.2-1: Information elements of the "ActivationJob"

Information element name	S	Documentation and Allowed Values	Properties
id	M	The identifier of the activation job.	type: String multiplicity: 1 isInvariant: True isWritable: False
name	M	The name of the activation job.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
description.	M	The textual human-readable description of the activation job.	type: String multiplicity: 0..1 isInvariant: False isWritable: True
mnsConsumerId	M	The consumer that created the job. It may indicate a human user and/or one or more applications, for example ["userid:janedoe", "appid:12314"].	type: String multiplicity: * isOrdered: False isUnique: True isInvariant: False isWritable: True
planConfigDescrId	M	The identifier of the planned configuration descriptor, whose operation set is requested to be activated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigGroupDescrId	M	or, alternatively, the identifier of the planned configuration group descriptor, whose operation sets are requested to be activated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
fallbackConfigDescrId	M	or, alternatively, the identifier of the fallback configuration descriptor, whose operation sets are requested to be activated.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigDescr	M	or, alternatively, the planned configuration descriptor to be activated, if no planned configuration descriptor was created earlier.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
planConfigGroupDescr	M	or, alternatively, the planned configuration group descriptor to be activated, if no planned configuration group descriptor was created earlier.	type: String multiplicity: 0..1 isInvariant: True isWritable: True
createdFallbackConfigDescrId	O	Id of the created Fallback Plan Configuration Descriptor. If isFallbackEnabled is supported, this must be supported too.	type: String multiplicity: 0..1 isInvariant: False isWritable: False
isFallbackEnabled	O	This boolean attribute allows to specify if the MnS producer shall create a plan with the current configuration to allow for a later fallback. The plan-id of the fallback plan shall be returned to the MnS consumer.	type: Boolean multiplicity: 1 isInvariant: True isWritable: True defaultValue: false
serviceImpact	O	The service impact mode. allowedValues: - LEAST_SERVICE_IMPACT - SHORTEST_TIME	type: ENUM multiplicity: 1 isInvariant: True isWritable: True defaultValue: SHORTEST_TIME
isImmediateActivation	O	This boolean attribute specifies if the activation job shall start immediately (value is "True") or, alternatively, by conditional activation (value is "False").	type: Boolean multiplicity: 1 isInvariant: True isWritable: True defaultValue: true
cancelRequest	O	This boolean attribute allows to request to cancel the activation process by setting its value to "True". Setting the value to "False" has no observable result. When the value is set to "True" it cannot be changed any more.	type: Boolean multiplicity: 1 isInvariant: False isWritable: True

Information element name	S	Documentation and Allowed Values	Properties
jobState	M	The activation job state. allowedValues: - NOT_STARTED - QUEUED - RUNNING - CANCELLING - CANCELLED - COMPLETED - FAILED	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
jobDetails	M	Detailed information related to the job, including job related errors.	type: JodDetails multiplicity: 1 isInvariant: False isWritable: False
startedAt	M	The date and time at which the activation process started, i.e. the time when the job state transition from "NOT_STARTED" to "RUNNING" occurred. In the "NOT_STARTED" state the information element is absent or carries no information.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: FalseType: DateTime multiplicity: 0..1 isOrdered: NA isUnique: NA
stoppedAt	M	The date and time at which the activation process stopped, i.e. the job state transition from "RUNNING" to "COMPLETED" or "FAILED", or from "CANCELLING" to "CANCELLED". In the "NOT_STARTED", "RUNNING" or "CANCELLING" state the information element is absent or carries no information.	type: DateTime multiplicity: 0..1 isInvariant: False isWritable: False
activationState	M	The current activation state of the planned configuration or planned configuration group that is processed by the activation job. allowedValues: - NOT_STARTED - ACTIVATING - ACTIVATION_SUCCEEDED - ACTIVATION_SUCCEEDED_PARTIALLY - ACTIVATION_FAILED - ACTIVATION_FAILED_ROLLED_BACK - ACTIVATION_FAILED_ROLLBACK_FAILED	type: ENUM multiplicity: 1 isInvariant: False isWritable: False
activationDetails	M	Details of the activation of the operations that are contained in the planned configuration or planned configuration group.	type: ExecutionDetails multiplicity: 1 isInvariant: False isWritable: False

Annex A (normative): Solution sets

A.1 RESTful HTTP-based solution set

A.1.1 General Considerations

Provisions for mapping from stage 2 to stage 3

The naming of information elements in stage-3 generally is the same as in stage-2, except for the following cases:

A "changeId" may or may not be provided for each change by MnS consumers according to stage 2 definitions. The following snippet shows an example where the "changeId" is provided.

```
"configChanges": [
  {
    "changeId": "change-1",
    "modifyOperator": "create",
    "target": "example.org/3gpp/SubNetwork=SN1/ManagedElement=ME10",
    "value": {
      "id": "ME10",
      "attributes": {}
    }
  },
  {
    "changeId": "change-2",
    "modifyOperator": "create,
    "target": "example.org/3gpp/SubNetwork=SN1/ManagedElement=ME20",
    "value": {
      "id": "ME20",
      "attributes": {}
    }
  }
]
```

Provisions for OpenAPI defined data node trees

The following provisions apply when the planned configuration is for a current configuration that is defined with OpenAPI. The "configChangesContentType" is "OpenAPI". It is based on 3GPP JSON Patch defined in clause 6.4.3 of TS 32.158 [2].

The "target" parameter value is the concatenation of the canonical form of the target URI defined in clause 4.2.3 of TS 32.158 [2] with "scheme://" omitted, and the "path" parameter of 3GPP JSON Patch defined in clause 6.4.3 of TS 32.158 [2], for example:

```
"target": "example.org/SubNetwork=SN1/ManagedElement=ME1"
"target": "example.org/SubNetwork=SN1/ManagedElement=ME1#/attributes/userLabel"
"target": "example.org/SubNetwork=SN1/PerfMetricJob=PMJ1#/attributes/perfMetrics/2"
```

For Managed Object Plans the "target" parameter identifies always a manged object. Therefore, the fragment component (i.e. the part starting with "#") is always missing.

The "value" parameter value is either a JSON object, a JSON array, a string, a number or one of the following three literal names: false, null, true. It is exactly the JSON value of the node identified by "target" without any additional wrappers or containers, for example:

```
"value": 5
"value": "Bts10"
"value": {
  "id": "ME10",
  "attributes": {
    "userLabel": "Berlin NW 1",
    "vendorName": "Company XY",
    "location": "Castle Charlottenburg"
```

```
  }
```

For Managed Object Plans the value of the "value" parameter shall comply to the following JSON schema snippet.

```
type: object
  properties:
    id:
      type: string
    objectClass:
      type: string
    objectInstance:
      $ref: 'TS28623_ComDefs.yaml#/components/schemas/Dn'
    attributes:
      type: object
```

The allowed values of the "modifyOperator" are as defined in stage 2.

Provisions for YANG defined data node trees

The following provisions apply when the planned configuration is for a current configuration that is defined with YANG. The "configChangesContentType" is "YANG". It is based on YANG Patch defined in RFC 8072 [3].

The "target" parameter value is constructed based on clause 2.4 of RFC 8072 [3] with the following modifications:

- The datastore resource "{+restconf}/data" shall be omitted.
- The module prefix may be omitted, and the MnS producer shall accept it if the target without the module prefix is unambiguous.

The first example shows a target including the module prefix, and the second example shows the same target without module prefix. The third example shows the same target again without authority component.

```
"target": "example.org/3gpp-subnetwork:SubNetwork=SN1/3gpp-managed-element:ManagedElement=ME1"
"target": "example.org/SubNetwork=SN1/ManagedElement=ME1"
"target": "/SubNetwork=SN1/ManagedElement=ME1"
```

The "value" parameter value is encoded according to RFC 7951 [a]. The module prefix may also be omitted for the "value" parameter.

```
"value": {
  "count": 5
}
"value": {
  "name": "Bts10"
"value": {
  "ManagedElement": [
    {
      "id": "ME10",
      "attributes": {
        "userLabel": "Berlin NW 1",
        "vendorName": "Company XY",
        "location": "Castle Charlottenburg"
      }
    }
  ]
}
```

The allowed values of the "modifyOperator" are as defined in stage 2.

Provisions for mapping the planned configuration in (3GPP) JSON Patch

In case the activation process requires to map the planned configuration into a 3GPP JSON Patch document the following provisions apply:

The "target" parameter value is assigned to the "path" property without modification.

The "value" parameter value is assigned to the "value" property without modification.

The "modifyOperator" parameter value is mapped to the "op" property according to the following rules:

- The modify operator "create" is strict, the JSON Patch operation "add" is relaxed. Therefore, an "add" operation is only included in the JSON Patch document, if the node to be created does not exist in the current configuration. Otherwise, an error is raised.

- The modify operator "delete" is relaxed, the JSON Patch operation "remove" is strict. Therefore, a "remove" operation is only generated in the JSON Patch document, if the target node exists in the current configuration, otherwise the "deleteMoi" operation in the plan is not included in the 3GPP JSON Patch document.

- The modify operator "merge" is strict, the 3GPP JSON Patch operation "merge" is relaxed. Therefore, the "merge" operation is included in the 3GPP JSON Patch document only, if the target node exists in the current configuration. Otherwise an error is raised.

- The modify operator "merge-create" is relaxed, the 3GPP JSON Patch operation "merge" is relaxed. Therefore, the "merge" operation is always included in the 3GPP JSON Patch document.

Table A.1.1-1 summarizes the property names used in "configChanges" and the corresponding names in 3GPP JSON Patch.

Table A.1.1-1: Names in "configChanges" and 3GPP JSON Patch

configChanges property name	3GPP JSON Patch property name
target	path
modifyOperator	op
value	value

Provisions for mapping the planned configuration into NETCONF

In case the activation process requires to map the planned configuration into NETCONF the following provisions apply:

The "target" parameter value is mapped into a series of nested XML elements with the innermost XML element being the target data node.

The "value" parameter value (in JSON encoding according to RFC 7951 [a]) is mapped to its corresponding XML representation.

The "modifyOperator" parameter value is mapped to the "operation" attribute according to the following rules:

- The modify operator "create" is strict, NETCONF "create" operation is strict. Therefore, the "create" operation is always included in the NETCONF request.

- The modify operator "delete" is relaxed, NETCONF "remove" operation is relaxed. Therefore, the "remove" operation is always included in the NETCONF request.

- The modify operator "merge" is strict, NETCONF "merge" operation is relaxed. Therefore, the "merge" operation is included in the NETCONF request only, if the target data node exists in the current configuration. Otherwise an error is raised.

- The modify operator "merge-create" is relaxed, NETCONF "merge" is relaxed. Therefore, the "merge" operation is always included in the NETCONF request.

A.1.2 Resource structure

Table A.1.2-1 provides an overview of the resources on the MnS producer and applicable HTTP methods.

Table A.1.2-1: Resources and methods

Resource name	Resource URI	HTTP method	Description
	…/plan-management/{/MnSVersion}/\ plan-descriptors/	POST	Creates a new planned configuration descriptor
		GET	Reads all plannd configuration descriptors
	…/plan-management/{/MnSVersion}/\ plan-descriptors/{id}	GET	Reads one planned configuration descriptor
		PUT	Replaces one planned configuration. descriptor. HTTP method is optional.
		PATCH	Patches one planned configuration descriptor. HTTP method is optional.
		DELETE	Deletes one planned configuration descriptor
	…/plan-management/{/MnSVersion}/\ plan-group-descriptors	POST	Creates a new planned configuration group descriptor
		GET	Reads all plannd configuration group descriptors
	…/plan-management/{/MnSVersion}/\ plan-group-descriptors/{id}	GET	Read one planned configuration group descriptor
		PUT	Replace one planned configuration group descriptor
		DELETE	Delete one planned configuration group descriptor
	…/{plan-management}/{/MnSVersion}/\ fallback-descriptors	GET	Read all fallback descriptors
	…/{plan-management}/{/MnSVersion}/\ fallback-descriptors/{id}	GET	Read one fallback descriptor
		DELETE	Delete one fallback descriptor
	…/plan-management/{/MnSVersion}/\ trigger-descriptors	POST	Create a new trigger descriptor
		GET	Read all trigger descriptors
	…/plan-management/{/MnSVersion}/\ trigger-descriptors/{id}	GET	Read one trigger descriptors
		PUT	Replace one trigger descriptors
		DELETE	Delete one trigger descriptors
	…/{plan-management}/{/MnSVersion}/\ plan-validation-jobs	POST	Create a new plan validation job
		GET	Read all validation jobs
	…/{plan-management}/{/MnSVersion}/\ plan-validation-jobs/{id}	GET	Read on validation job
		DELETE	Delete one validation job
	…/{plan-management}/{/MnSVersion}/\ plan-validation-jobs/{id}/validation-details	GET	Read validation details of one validation job
	…/{plan-management}/{/MnSVersion}/\ plan-validation-jobs/{id}	PATCH	Cancel one validation job
	…/{plan-management}/{/MnSVersion}/\ plan-validation-jobs	POST	Create a new plan activation job
		GET	Read all activation jobs
	…/{plan-management}/{/MnSVersion}/\ plan-activation-jobs/{id}	GET	Read on activation job
		DELETE	Deleete one activation job
	…/{plan-management}/{/MnSVersion}/\ plan-activation-jobs/{id}/activation-details	GET	Read validation details of one activation job
	…/{plan-management}/{/MnSVersion}/\ plan-activation-jobs/{id}	PATCH	Cancel one activation job

Creating, reading, updating and deleting planned configurations

The data node tree on the MnS producer is as follows upon system start up.

```
{
    "plan-descriptors": {},
    "plan-group-descriptors": {},
    "fallback-descriptors": {},
    "trigger-descriptors": {},
    "plan-validation-jobs": {},
    "plan-activation-jobs": {}
}
```

A.1.3 OpenAPI definitions

OpenAPI definitions for the management of planned configurations are specified in Forge, refer to clause 4.3 of TS 28.623 [16] for the Forge location. An example of Forge location is: "https://forge.3gpp.org/rep/sa5/MnS/-/tree/Tag_Rel18_SA105/".

Directory: OpenAPI

Files:

TS28572_PlanManagement.yaml

A.1.4 Examples (informative)

A.1.4.1 Introduction

The examples in this section are grouped per configuration changes content type (OPENAPI_BASED / YANG_BASED).

A.1.4.2 OPENAPI_BASED

A new item of the collection resource "plan-descriptors" is created by MnS consumers using HTTP POST.

```
POST 3gpp/plan-management/v1/plan-descriptors HTTP/1.1
Host: exampleA.org
Content-Type: application/json

{
  "name": "NewBts10Plan",
  "version": "2.0",
  "description": "This is the plan for the new BTS 10.",
  "configChangesContentType": "OPENAPI_BASED",
 "activationMode": "ATOMIC",
  "configChanges": [
    {
      "modifyOperator": "create",
      "target": "example.org/3gpp/SubNetwork=SN1/ManagedElement=ME10",
      "value": {
        "id": "ME10",
        "attributes": {
          "userLabel": "Berlin NW 1",
          "vendorName": "Company XY",
          "location": "Castle Charlottenburg"
        }
      }
    }
  ]
}
```

The MnS producer allocates the identifier "p1" for the new resource and returns the response. The location header contains the URI of the new resource. The response body contains the representation of the new resource which is equal to the representation received in the request with the "lastModifiedAt" and "validationState" properties added.

```
HTTP/1.1 201 Created
Date: Wed, 21 Aug 2024 15:39:57 GMT
Location: http://exampleA.org/3gpp/plan-management/plan-descriptors/p1
Content-Type: application/json

{
  "name": "NewBts10Plan",
  "version": "2.0",
  "description": "This is the plan for the new BTS 10.",
  "activationMode": "ATOMIC",
  "configChangesContentType": "OPENAPI_BASED",
  "configChanges": [
    {
      "modifyOperator": "create",
      "target": "example.org/3gpp/SubNetwork=SN1/ManagedElement=ME10",
```

```
      "value": {
        "id": "ME10",
        "attributes": {
          "userLabel": "Berlin NW 1",
          "vendorName": "Company XY",
          "location": "Castle Charlottenburg"
        }
      }
    }
  ],
  "lastModifiedAt": "2025-03-06T16:50:26-08:00",
  "validationState": "NOT_VALIDATED"
}
```

The resource structure on the MnS producer contains the new resource.

```
{
  "plan-descriptors": {
    "p1": {
      "name": "NewBts10Plan",
      "version": "2.0",
      "description": "This is the plan for the new BTS 10.",
      "activationMode": "ATOMIC",
      "configChangesContentType": "OPENAPI_BASED",
      "configChanges": [
       {
          "modifyOperator": "create",
          "path": "example.org/3gpp/SubNetwork=SN1/ManagedElement=ME10",
          "value": {
            "id": "ME10",
            "attributes": {
              "userLabel": "Berlin NW 1",
              "vendorName": "Company XY",
              "location": "Castle Charlottenburg"
            }
          }
        }
      ],
      "lastModifiedAt": "2025-03-06T16:50:26-08:00",
      "validationState": "NOT_VALIDATED"
    }
  },
  "plan-group-descriptors": {},
  "fallback-config-descriptors": {},
  "trigger-condition-descriptors": {},
  "plan-validation-jobs": {},
  "plan-activation-jobs": {}
}
```

The next example shows how the value of the "location" attribute in the operation can be changed from "Castle Charlottenburg" to "Summer palace Charlottenburg" using JSON Patch.

```
PATCH 3gpp/plan-management/v1/plan-descriptors/p1 HTTP/1.1
Host: example.org
Content-Type: application/json-patch+json

[
  {
    "op": "replace",
    "path": "/configChanges/0/value/attributes/location",
    "value": "Summer palace Charlottenburg"
  }
]
```

The second example is for the case where a modified JSON Patch is used.

```
PATCH 3gpp/plan-management/v1/plan-descriptors/p1 HTTP/1.1
Host: example.org
Content-Type: application/json

[
  {
    "op": "replace",
```

```
    "path": "/configChanges[changeId='pc1']/value/attributes/location",
    "value": "Summer palace Charlottenburg"
  }
]
```

To read the planned configuration "p1" a MnS consumer might send.

```
GET 3gpp/plan-management/v1/plan-descriptors/p1 HTTP/1.1
Host: exampleA.org
Accept: application/json
```

The MnS producer may respond as follows.

```
HTTP/1.1 200 OK
Date: Tue, 06 Aug 2025 16:50:26 GMT
Content-Type: application/json

{
  "name": "NewBts10Plan",
  "version": "2.0",
  "description": "This is the plan for the new BTS 10.",
  "configChangesContentType": "OPENAPI_BASED",
  "currentConfigAddress": " example.org/3gpp ",
  "activationMode": "ATOMIC",
  "lastModifiedAt": "2025-03-06T16:50:26-08:00",
  "validationState": "NOT_VALIDATED",
  "configChanges": [
    {
      "modifyOperator": "CREATE",
      "path": "/SubNetwork=SN1/ManagedElement=ME10",
      "value": {
        "id": "ME10",
        "attributes": {
          "userLabel": "Berlin NW 1",
          "vendorName": "Company XY",
          "location": "Castle Charlottenburg"
        }
      }
    }
  ]
}
```

To delete the planned configuration "p1" a MnS consumer might send.

```
DELETE 3gpp/plan-management/v1/plan-descriptors/p1 HTTP/1.1
Host: exampleA.org
```

In case of success, the MnS producer returns the following message.

```
HTTP/1.1 204 No Content
Date: Wed, 21 Aug 2024 16:12:45 GMT
```

To read the planned configuration "p1" a MnS consumer might send.

```
GET 3gpp/plan-management/v1/plan-descriptors/p1 HTTP/1.1
Host: exampleA.org
Accept: application/json
```

The MnS producer may respond as follows.

```
HTTP/1.1 200 OK
Date: Tue, 06 Aug 2025 16:50:26 GMT
Content-Type: application/json

{
  "name": "NewBts10Plan",
  "version": "2.0",
  "description": "This is the plan for the new BTS 10.",
```

```
  "configChangesContentType": "OPENAPI_BASED",
  "currentConfigAddress": " example.org/3gpp ",
  "activationMode": "ATOMIC",
  "lastModifiedAt": "2025-03-06T16:50:26-08:00",
  "validationState": "NOT_VALIDATED",
  "configChanges": [
    {
      "modifyOperator": "create",
      "path": "/SubNetwork=SN1/ManagedElement=ME10",
      "value": {
        "id": "ME10",
        "attributes": {
          "userLabel": "Berlin NW 1",
          "vendorName": "Company XY",
          "location": "Castle Charlottenburg"
        }
      }
    }
  ]
}
```

The following shows a snippet reporting member conflicts.

```
"memberConflicts": [
  {
    "memberConflict": [
      {
        "planConfigDescrId": "pcd-1",
        "changeId": "change-4"
      },
      {
        "planConfigDescrId": "pcd-3",
        "changeId": "change-6"
      },
      {
        "planConfigDescrId": "pcd-4",
        "changeId": "change-7"
      }
    ]
  },
  {
    "memberConflict": [
      {
        "planConfigDescrId": "pcd-5",
        "changeId": "change-1"
      },
      {
        "planConfigDescrId": "pcd-6",
        "changeId": "change-9"
      }
    ]
  }
]
```

The following shows a snippet reporting activation details.

```
"activationDetails": {
  "summary": {
    "notfinished": 1,
    "succeeded": 1,
    "failed": 1,
    "rollBackSucceeded": 0,
    "rollBackFailed": 0
  },
  "results": [
    {
      "planConfigDescrId": "pcd1",
      "changeIndex": 0,
      "state": "SUCCEEDED"
    },
    {
      "planConfigDescrId": "pcd1",
      "changeIndex": 1,
      "state": "FAILED",
```

```
      "errors": {
        "type": "REQUEST_OBJECT_TREE_MISMATCH",
        "title": "The current state of the data node tree prevents activation.",
        "reason": "NEW_OBJECTS_ID_EXISTS",
        "detail": "The id of the new object to be created does exist already."
      }
    },
    {
      "planConfigDescrId": "pcd1",
      "changeIndex": 2,
      "state": "NOT_STARTED"
    }
  ]
}
```

A.1.4.3 YANG_BASED

The examples in this section show how an operator may make changes to their configuration based on a YANG based schemas and related json encoding (as per A.1.1 Provisions for YANG defined data node trees). Note that all the examples described in OPENAPI_BASED case above are compatible with the YANG_BASED option and will execute successfully. The YANG_BASED option will support a broader range of YANG capabilities (e.g. namespace validation and other yang based schema constructs).

Usecase : Create a plan configuration (plan-descriptor)

```
POST plan-management/v1/plan-descriptors HTTP/1.1
Host: example.org
Content-Type: application/json
Accept : application/json,application/problem+json


{
  "name": "Rollout-5G-Dublin-East",
  "version": "1.0.0",
  "description": "This is the plan for the new 5G rollout in Dublin east.",
  "customProperties": {
    "technology-type": "NR",
    "location": "Dublin"
  },
  "configChangesContentType": "YANG_BASED",
  "configChanges": {
    "modifyOperator": "create",
    "changeId": "add-nr-cell-001",
    "comment": "Add new NR cell for initial deployment in Dublin-4 area.",
    "target": "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/NRCellDU=4",
    "value": {
      "NRCellDU": [
        {
          "id": "4",
          "attributes": {
            "userLabel": "Dublin-1-Cell-4",
            "administrativeState": "UNLOCKED",
            …
          }
        }
      ]
    }
  }
}
```

A response for the above might look like below

```
Location: /plan-descriptors/Rollout-5G-Dublin-East-1.0.0
Content-Type: application/json
Date: Thu, 02 Oct 2025 19:21:37 GMT

{
  "id": "Rollout-5G-Dublin-East-1.0.0",
  "name": "Rollout-5G-Dublin-East",
  "version": "1.0.0",
  "description": "This is the plan for the new 5G rollout in Dublin east.",
  "creationTime": "2025-10-02T19:21:37Z",
```

```
  "status": "CREATED",
  "configChangesContentType": "YANG_BASED",
  "customProperties": {
    "technology-type": "NR",
    "location": "Dublin"
  },
  "configChanges": {
    "modifyOperator": "create",
    "changeId": "add-nr-cell-001",
    "comment": "Add new NR cell for initial deployment in Dublin-4 area.",
    "target": "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/NRCellDU=4",
    "value": {
      "NRCellDU": [
        {
          "id": "4",
          "attributes": {
            "userLabel": "Dublin-1-Cell-4",
            "administrativeState": "UNLOCKED"
          }
        }
      ]
    }
  }
}
```

An example request with yang semantics including the module name is shown below :

```
{
  "name": "Rollout-5G-Dublin-East",
  "version": "1.0.0",
  "description": "This is the plan for the new 5G rollout in Dublin east.",
  "customProperties": {
    "technology-type": "NR",
    "location": "Dublin"
  },
  "configChangesContentType": "YANG_BASED",
  "configChanges": {
    "modifyOperator": "create",
    "changeId": "add-nr-cell-001",
    "comment": "Add new NR cell for initial deployment in Dublin-4 area.",
    "target": "/3gpp-common-subnetwork:SubNetwork=Irl/3gpp-common-mecontext:MeContext=Dublin-
1/_3gpp-common-managed-element:ManagedElement=Dublin-1/3gpp-nr-nrm-
gnbdufunction:GNBDUFunction=1/NRCellDU=4",
    "value": {  # The value may contain any valid json encoded yang as per RFC 7951
      "3gpp-nr-nrm-nrcelldu:NRCellDU": [
        {
          "id": "4",
          "attributes": {
            "userLabel": "Dublin-1-Cell-4",
            "administrativeState": "UNLOCKED",
            …
          }
        }
      ]
    }
  }

}
```

Usecase : Plan Activation Job usecases (plan-activation-jobs)

This section shows examples for plan activation job related resources. This example uses the 'inline' form of the plan descriptor:

```
# Create plan-activation-job

POST {apiRoot}/plan-management/v1/plan-activation-jobs HTTP/1.1
Host: example.org
Content-Type: application/json
Accept : application/json,application/problem+json

{
```

```
  "name" : "act-plan-001",
  "mnsConsumerId" : ["user:joesoap","app:213"],
  "description": "Activation job for inline 5G cell creation in Dublin-5.",
  "customProperties": {
    "requester": "AutomationTool",
    "deployment-phase": "pilot"
  },

  "planConfigDescr": {
    "name": "Inline-Rollout-5G-Dublin-5",
    "version": "1.0.0",
    "description": "Plan defined inline for new cell deployment.",
    "customProperties": {
      "technology-type": "NR",
      "location": "Dublin-5"
    },
    "isFallbackEnabled" : true,
    "configChangesContentType": "YANG_BASED",
    "configChanges": [
      {
        "modifyOperator": "create",
        "changeId": "add-nr-cell-005",
        "comment": "Inline job: Add new NR cell 5 to GNBDUFunction 1.",
        "target": "/SubNetwork=Irl/MeContext=Dublin-
1/ManagedElement=1/GNBDUFunction=1/NRCellDU=5",
        "value": {
          "NRCellDU": [
            {
              "id": "5",
              "attributes": {
               "userLabel": "Dublin-5-Cell-5",
               "administrativeState": "UNLOCKED"
              }
            }
          ]
        },
      {
        "modifyOperator": "create",
        "changeId": "add-nr-cell-006",
        "comment": "Inline job: Add new NR cell 6 to GNBDUFunction 1.",
        "target": "/SubNetwork=Irl/MeContext=Dublin-
1/ManagedElement=1/GNBDUFunction=1/NRCellDU=6",
        "value": {
          "NRCellDU": [
            {
              "id": "6",
              "attributes": {
               "userLabel": "Dublin-6-Cell-6",
               "administrativeState": "UNLOCKED"
              }
            }
          ]
        },
      {
        "modifyOperator": "create",
        "changeId": "add-nr-cell-007",
        "comment": "Inline job: Add new NR cell 7 to GNBDUFunction 1.",
        "target": "/SubNetwork=Irl/MeContext=Dublin-
1/ManagedElement=1/GNBDUFunction=1/NRCellDU=7",
        "value": {
          "NRCellDU": [
            {
              "id": "7",
              "attributes": {
               "userLabel": "Dublin-7-Cell-7",
               "administrativeState": "UNLOCKED"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Response

```
HTTP/1.1 201 Created
Location: /plan-activation-jobs/job-23451235153465vz5254
```

```
Content-Type: application/json
{
  "id": "job-23451235153465vz5254",
  "name" : " act-plan-001",
  "description" : "Activation job for inline 5G cell creation in Dublin-5.",
  "isFallbackEnabled" : true,
  "planConfigDescrId" : "planxyz",
  "mnsConsumerId" : ["user:joesoap","app:213"],
  "jobState" : "RUNNING",
  "startedAt" : "<some-time>",
  "activationState" : "NOT_STARTED",
  "activationDetails": {
     "href" : "{apiRoot}/ProvMnS/1900/plan-activation-jobs/myjob-111/activation-details"
  },
  "cancelRequest" : false,
  "links" : {
    "self" : { "href" : "{apiRoot}/ProvMnS/1900/plan-activation-jobs/myjob-111"},
    "descriptor" : "{apiRoot}/ProvMnS/1900/plan-descriptors/planxyz"},
    "status": { "href" : "{apiRoot}/ProvMnS/1900/plan-activation-jobs/myjob-111/status"},
    "fallback": { "href" : "{apiRoot}/ProvMnS/1900/fallback-descriptors/fb-001"}
   }
}


# GET plan-activation-job status.  Report all the relevant status information
# related to the plan activation job

GET {apiRoot}/plan-management/v1/plan-activation-jobs/myjob-111/status

Response
{
  "jobState": "COMPLETE",
  "activationState": "ACTIVATED",
  "startedAt": "2024-12-02T13:16:54.088Z",
  "stoppedAt": "2024-12-02T13:16:58.088Z"
}


#  GET plan-activation-job activation-details
#       - jobState=COMPLETED / activationState=ACTIVATED
#       - All configuration edits/operations are successfully activated
#       - no detailed info on failed edits/operations are reported (none failed)

GET {apiRoot}/ plan-management/v1/plan-activation-jobs/myjob-111/activation-details

Response
{
  "summary" : {
    "notfinished": 0,
    "succeeded": 3,
    "failed": 0
    “rollbackSucceeded”: 0
    “rollbackFailed”: 0
    “conflicting” : 0
  }
}


#  GET plan-activation-job activation-details with verbose option (expand=all)
#       - jobState=COMPLETED / activationState=ACTIVATED
#       - All configuration edits/operations are successfully activated
#       - no detailed info on failed edits/operations are reported (none failed)

GET {apiRoot}/plan-management/v1/plan-activation-jobs/myjob-111/activation-details?expand=all

Response
{
  "results" : [
    {
      "planDescriptorId" : "plan-desc-001",
      "configChanges" : [
        {
          "changeId" : "add-nr-cell-005",
          "state" : "SUCCEEDED",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=5"
        },
        {
```

```
          "changeId" : "add-nr-cell-004",
          "state" : "SUCCEEDED",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=7"

        },
        {
          "changeId" : "add-nr-cell-007",
          "state" : "SUCCEEDED",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=7"

        }
      ]
    }
  ],
  "summary" : {
    "nofinished": 0,
    "succeeded": 3,
    "failed": 0,
    “rollbackSucceeded”: 0
    “rollbackFailed”: 0
    “conflicting” : 0

  }
}


#  GET plan-activation-job result-details with complete configuration edit/operation failure
#       - jobState=COMPLETED / activationState=ACTIVATION_FAILED
#       - All configuration edits/operations failed to activate
#       - Activation details on failed edits/operations are reported

GET {apiRoot}/plan-management/v1/plan-activation-jobs/myjob-111/activation-details

Response
{
  "results" : [
    {
          "planConfigDescrId" : "planxyz",
          "changeId" : "opId-001",
          "state" : "FAILED",",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=1"
          "errors" :: [
              {
                "type": "ACCESS_CONTROL_CONFLICT",
                "badDataNode": "/_3gpp-common-subnetwork:SubNetwork=Irl/_3gpp-common-
mecontext:MeContext=Dublin-1/_3gpp-common-managed-element:ManagedElement=Dublin-1/_3gpp-nr-nrm-
gnbdufunction:GNBDUFunction=1/_3gpp-nr-nrm-nrcelldu:NRCellDU=1",
                "message": "Data already exists; cannot be created"
              }
          }
        },
        {
          "planConfigDescrId" : "planxyz",
          "changeId" : "opId-002"",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=2",
          "state" : "FAILED",
          "errors": [
              {
                "type": "ACCESS_CONTROL_CONFLICT",
                "badDataNode": "/_3gpp-common-subnetwork:SubNetwork=Irl/_3gpp-common-
mecontext:MeContext=Dublin-1/_3gpp-common-managed-element:ManagedElement=Dublin-1/_3gpp-nr-nrm-
gnbdufunction:GNBDUFunction=1/_3gpp-nr-nrm-nrcelldu:NRCellDU=2
              }
            ]
          }
        },
        {
          "planConfigDescrId" : "planxyz",
          "changeId" : "opId-003",
          "state" : "FAILED"",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=3",
          "errors": [
              {
```

```
                "type": "ACCESS_CONTROL_CONFLICT",

                "badDataNode": "/_3gpp-common-subnetwork:SubNetwork=Irl/_3gpp-common-
mecontext:MeContext=Dublin-1/_3gpp-common-managed-element:ManagedElement=Dublin-1/_3gpp-nr-nrm-
gnbdufunction:GNBDUFunction=1/_3gpp-nr-nrm-nrcelldu:NRCellDU=3
              }
            ]
          }
        }
      ]
    }
  ],
  "summary" : {
    "notfinished": 0,
    "succeeded": 0,
    "failed": 3,
    “rollbackSucceeded”: 0
    “rollbackFailed”: 0
    “conflicting” : 0

   }
}


#  GET plan-activation-job result-details with failure (with expand=all option)
#       - jobState=COMPLETED / activationState=PARTIALLY_ACTIVATED
#       - Some configuration edits/operations failed to activate
#       - Activation details on failed edits/operations are reported
#       - Activation details on successful edits/operations are reported (expand=all)

GET {apiRoot}/plan-management/v1/plan-activation-jobs/myjob-111/activation-details?expand=all

Response
{
  "results" : [
    {
          "planDescriptorId" : "planxyz",
          "changeId" : "opId-001"",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=1",
          "state" : "FAILED",
          "errors": [
              {
                "type": "DATA_NODE_TREE_ERROR",
                "badDataNode": "/_3gpp-common-subnetwork:SubNetwork=Irl/_3gpp-common-
mecontext:MeContext=Dublin-1/_3gpp-common-managed-element:ManagedElement=Dublin-1/3gpp-nr-nrm-
gnbdufunction:GNBDUFunction=1/_3gpp-nr-nrm-nrcelldu:NRCellDU=1",
                "message": "Data already exists; cannot be created"
              }
          }
        },
        {
          "planDescriptorId" : "planxyz",
          "changeId" : "opId-002"",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=2",
          "state" : "SUCCEEDED"
        },
        {
          "planDescriptorId" : "planxyz",
          "changeId" : "opId-003"",
          "target" : "/SubNetwork=Irl/MeContext=Dublin-1/ManagedElement=1/GNBDUFunction=1/
                        NRCellDU=3",
          "state" : "SUCCEEDED"
        }
      ]
    }
  ],
  "summary" : {
    "notfinished": 0,
    "succeeded": 2,
    "failed": 1,
    “rollbackSucceeded”: 0
    “rollbackFailed”: 0
    “conflicting” : 0

   }
}
```

Annex B (informative): PlantUML for figures

```
@startuml
left to right direction
object Group_A
object Group_B
object Group_C
Group_A ---o Group_B : is_member_of
Group_B ---o Group_C : is_member_of
Group_C ---o Group_A : is_member_of
@enduml
```

Figure 6.2.2-1: NOT allowed circular planned configuration group membership relation (with 3 groups)

```
@startuml
hide empty members

object Group_A
object Group_B
object Group_C

object Plan1
object "Plan1 "
Object Plan2
Object "Plan2 "

Group_A -up-o Group_B : is_member_of
Group_B -down-o Group_C : is_member_of
Plan1 -up-o Group_A : is_member_of
Plan2 -up-o Group_A : is_member_of
"Plan2 " -up-o Group_A : is_member_of
"Plan1 " -up-o Group_C : is_member_of
@enduml
```

Figure 6.2.2-2: NOT allowed multiple containment of planned configuration

Annex C (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2024-10	SA5#157	S5-245426				Input Draft TS 28.572 Management of planned configurations	0.0.0
2024-10	SA5#157	S5-246080				pCR 28.572 Add skeleton	0.1.0
2024-10	SA5#157	S5-246081				pCR 28.572 Add requirements for plans	0.1.0
2024-10	SA5#157	S5-246082				pCR 28.572 Add requirements for transaction	0.1.0
2024-10	SA5#157	S5-246083				pCR 28.572 Add requirements for validation	0.1.0
2024-10	SA5#157	S5-246084				pCR 28.572 Add requirements for activation	0.1.0
2024-10	SA5#157	S5-246085				pCR 28.572 Add content for plans	0.1.0
2024-10	SA5#157	S5-246086				pCR 28.572 Add content for transactions	0.1.0
2024-10	SA5#157	S5-246087				pCR 28.572 Add content for validation	0.1.0
2024-10	SA5#157	S5-246088				pCR 28.572 Add content for activation	0.1.0
2024-11	SA5#158	S5-247183				pCR 28.572 Add conditional activation of plans	0.2.0
2024-11	SA5#158	S5-247184				pCR 28.572 Improve description of clause Planned configurations	0.2.0
2024-11	SA5#158	S5-247185				pCR 28.572 Improve description of clause Planned configuration groups	0.2.0
2024-11	SA5#158	S5-247186				pCR 28.572 Improve description of clause Validation of planned configurations	0.2.0
2024-11	SA5#158	S5-247187				pCR 28.572 Improve description of clause Activation of planned configurations	0.2.0
2024-11	SA5#158	S5-247188				pCR 28.572 Improve description of clauses about managing planned configurations	0.2.0
2024-11	SA5#158	S5-247189				pCR 28.572 Add examples for REST	0.2.0
2025-02	SA5#159	S5-250908				pCR TS 28.572 Improve clause Planned configuration groups	0.3.0
2025-02	SA5#159	S5-250909				pCR TS 28.572 Add clause Fallback configurations and Managing fallback configurations	0.3.0
2025-02	SA5#159	S5-251126				pCR TS 28.572 Improve clause Trigger conditions	0.3.0
2025-02	SA5#159	S5-250911				pCR TS 28.572 Improve clause Validation	0.3.0
2025-02	SA5#159	S5-250912				pCR TS 28.572 Improve clause Activation	0.3.0
2025-02	SA5#159	S5-250913				pCR TS 28.572 Add clause Asynchronous configuration updates	0.3.0
2025-02	SA5#159	S5-250914				pCR TS 28.572 Improve clauses on Managing planned configurations	0.3.0
2025-02	SA5#159	S5-250915				pCR TS 28.572 Add information model	0.3.0
2025-04	SA5#160	S5-251950				Rel-19 pCR TS 28.572 Improve clauses on Planned configuration concepts	0.4.0
2025-04	SA5#160	S5-251951				Rel-19 pCR TS 28.572 Improve clauses on Managing planned configurations	0.4.0
2025-04	SA5#160	S5-251952				Rel-19 pCR TS 28.572 Improve clause Information model	0.4.0
2025-04	SA5#160	S5-251953				Rel-19 pCR TS 28.572 Improve clause RESTful HTTP-based solution set	0.4.0
2025-04	SA5#160	S5-251954				Rel-19 pCR TS 28.572 Allow recursive Planned configuration groups	0.4.0
2025-05	SA5#161	S5-252897				Rel-19 pCR TS 28.572 Improve clause Information model	0.5.0
2025-05	SA5#161	S5-252898				Rel-19 pCR TS 28.572 Plan Configuration Examples	0.5.0
2025-07						MCC: the previous version seems to contain an empty file.	0.5.1
2025-08	SA5#162	S5-253895				Rel-19 pCR 28.572 Enhance Draft TS on Management of planned configurations	0.6.0
2025-08	SA5#162	S5-253896				Rel-19 pCR TS 28.572 Plan Configuration Stage-3	0.6.0
2025-09	SA#109	SP-251037				Presented for information and approval	1.0.0
2025-09	SA#109					Upgrade to change control version	19.0.0
2025-12	SA#110	SP-251401	0001	1	C	Rel-19 CR 28.572 Plan management stage3 updates	19.1.0
2025-12	SA#110	SP-251401	0002	2	F	Rel-19 CR 28.572 Correct multiple errors in stage 2	19.1.0
2025-12	SA#110	SP-251401	0003		F	CR 28.572 Correct multiple errors in stage 3	19.1.0
2026-03	SA#111	SP-260071	0004	1	F	Rel-19 CR TS 28.572 Plan management corrections	19.2.0

History

Version	Date	Status
V19.0.0	October 2025	Publication
V19.1.0	February 2026	Publication
V19.2.0	April 2026	Publication