Understanding coverage and the calculation of Test and requirement statuses
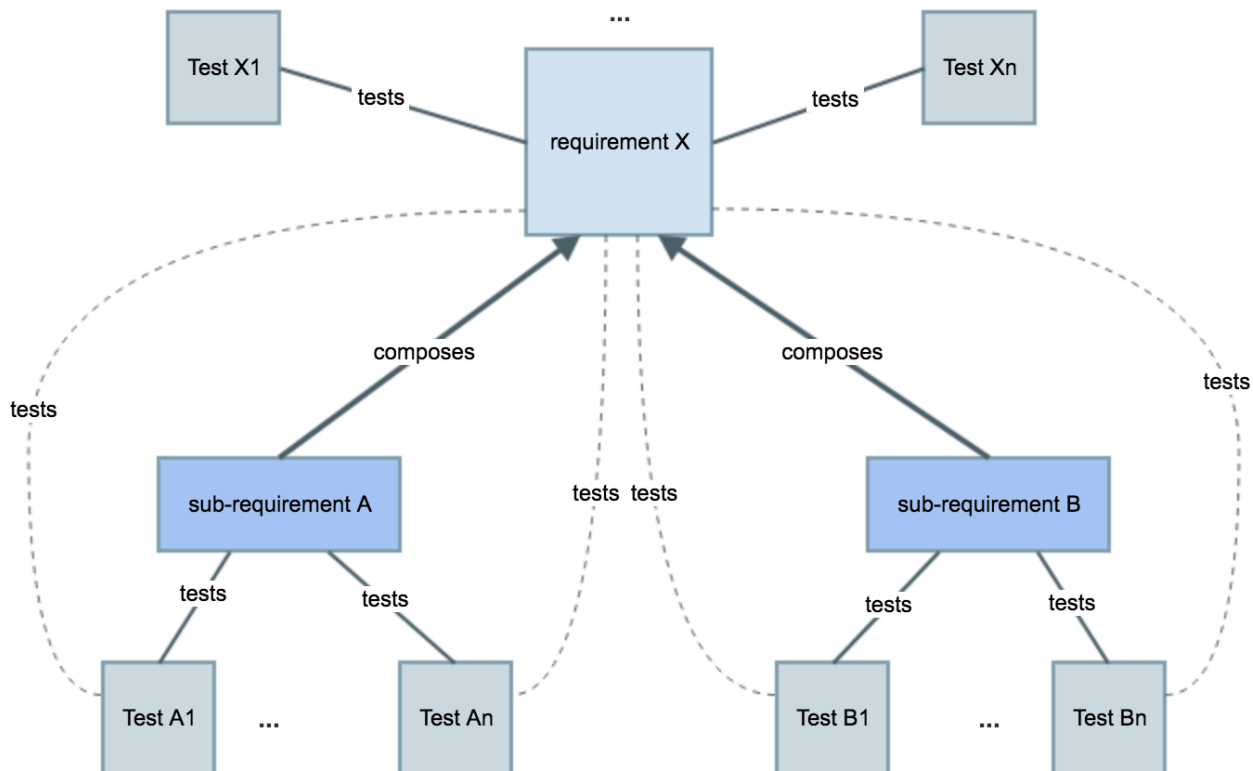
A requirement may be either covered by one or multiple Tests. In fact, the status of a given requirement goes way further than the basic covered/not covered information: it will take into account your test results.

As soon as you start running your Tests, the individual test result may be one of many and be very specific to your use case.

To make your analysis even more complex, you may be using sub-requirements and executing related Tests.

Requirements may be validated directly or indirectly, through the related sub-requirements and associated Test cases.

Thus, how do all these factors contribute to the calculation of a requirement status? How is evaluated the status of some Test?

Let's start by detailing the different possible values Test, Test Step and requirement statuses. In the end, we'll see how they'll impact on the calculation of the coverage status of a requirement in some specific version or Test Plan.

Overview of statuses

Whenever talking about statuses, we may be talking about statuses of requirements, Tests, Test Runs and Test Steps.

The status of a requirement depends on the status of the "related" Tests.

The status of a Test depends on the status of the "related" Test Runs, which in turn depend on the recorded Test Step statuses for each one of them.

> (i) **Please note**
>
> Whenever we're speaking about the status associated with a requirement or with a Test (and even with a Test Set), we may be talking about different things:
>
> - (coverage) **status** for some version, taking into account executions made for that version of the Tests that validate the requirement
> - workflow status associated to the requirement issue (e.g., "New", "In Progress", "Closed")
>
> - specific custom fields for those issue types (e.g. "Requirement Status", "TestRunStatus" and "Test Set Status")
>
> The current page details the first one, i.e. the status of the entities based on the executions made in some context.
>
> The "Requirement Status", "TestRunStatus" and "Test Set Status" custom fields are **calculated** fields, usable in the context of requirement, Test and Test Set issues, respectively, that calculate the "status" (not the workflow status) of the requirement/Test/Test Set for a specific version, considering the executions (i.e., test runs) for that version. The version for which it calculates the status depends on the behavior defined in a global configuration (**see configuration details** here).
>
> These fields will show the calculated status for the respective entity, for some specific version; they're just used to have a quick glimpse of the status of each entity, for example right in the view issue screen; therefore their usage is limited.
>
> More information on these and other custom fields can be found here .

Test status

The status of a Test tells you information about its current consolidated state (e.g. latest record result, if existent). *Was it is executed? Successfully? In which version?*

Thus, whenever speaking about the "status of a Test" we need to give it some additional context (e.g. "In which version?") since it depends on "where" and how you want to analyze it.

> (i) The status of a Test indicates its "latest state" in some given context (e.g. for some version, some Test Plan and/or in some Test Environment).

Xray provides some built-in Test statuses (which can't be modified nor deleted):

- TODO – Test is pending execution; this is a non-final status;
- EXECUTING – Test is being executed; this is a non-final status; at least one step is mapped to a non-final Test Run status
- FAIL – Test failed
- ABORTED – Test was aborted
- PASS – Test passed successfully

Each of this status maps to a requirement status, accordingly with the following table.

Test status	Final status?	Requirement status mapped to
PASS	yes	OK
FAIL	yes	NOK
TODO	no	NOTRUN

ABORTED	yes	NOTRUN
EXECUTING	no	NOTRUN
custom	*custom*	*OK, NOK, NOTRUN or UNKNOWN*

The status (i.e. result) of a Test Run is an attribute of the Test Run (a "Test Run" is an instance of a Test and is not a Jira issue) and is the one taken into account to assess the status of the requirement.

> **ⓘ Please note**
>
> Do not mix up the status of a given test with the "TestRunStatus" custom field, which shows the status of a given Tests for a specific version, depending on the configuration under Configuring Statuses Custom Fields. More info on this custom field here.
>
> The "TestRunStatus" custom field is a calculated field that belongs to the Test issue and that takes into account several Test Runs; the "TestRunStatus" does not affect the calculation of the status of requirements.

Managing Test Statuses

Creating new Test (Run) statuses may be done in the Manage Test Statuses configuration section of Xray.

Whenever creating/editing a Test status, we have to identify the Requirement status we want this Test status to map to.

Test Statuses

Manage custom Test Run statuses for Xray

Name	Description	Final	Color	Requirement Status
PASS	The test run has passed	✓	🟩	OK
TODO	The test run has not started	☐	⬜	NOTRUN
EXECUTING	The test run is currently being executed	☐	🟨	NOTRUN
FAIL	The test run has failed	✓	🟥	NOK
ABORTED	The test run was aborted	✓	⬛	NOTRUN
BLOCKED		☐	🟦	NOTRUN
PENDING		✓	🟥	UNKNOWN
FAIL_DISCARDABLE	FAIL_DISCARDABLE	☐	🟧	UNKNOWN
MYFAIL	MYFAIL	✓	🟪	NOK
MYPASS2	MYPASS2	✓	🟩	OK

Create

Preferences

☑ Final statuses have precedence over non-final statuses

When this option is enabled, the latest Test Issue Status is calculated based on the latest Test Run with a **final** status. Otherwise, this calculation will always co status is non-final.

One important attribute of a Test status is the "final" attribute. If "Final Statuses have precedence over non-final" flag is enabled, then Xray will give priority to final statuses whenever calculating the status of a Test. In other words, if you have a Test currently in some final status (e.g. PASS, FAIL) and you schedule a new Test Run for it, then this Test Run won't affect the calculation of the status of the Test.

This may be used, when users prefer to take into account only the last final/complete recorded result and want to discard Test Runs that are in an intermediate status (e.g. EXECUTING, TODO).

Test Step statuses

The status of a Test Step indicates the result obtained for that step for some Test Run.

> ⓘ Statuses reported at Test Step level will contribute to the overall calculation of the status of the related Test Run.

Xray provides some built-in Test Step statuses (which can't be modified nor deleted).

Test Step status	Test status
PASS	PASS
TODO	TODO
EXECUTING	EXECUTING
FAIL	FAIL
custom	*custom*

Managing Test Step Statuses

Creating new Test Step statuses may be done in the Manage Test Step Statuses configuration section of Xray.

Test Step Statuses

Manage custom Test Step statuses for Xray

Name	Description	Color	Test Status	Operations
PASS	The test step has passed	🟩	PASS	
TODO	The test step has not started	⬜	TODO	
EXECUTING	The test step is currently being executed	🟨	EXECUTING	
FAIL	The test step has failed	🟥	FAIL	

Create

Whenever creating/editing a Test Step status, we have to identify the Test status we want this step status to map to.

Create New Test Step Status

Create a new custom Test Step status

Name* [   ]

The Test Step Status name

Description [   ]

The Test Step Status description

Color* [#FFFFFF]

This is the color used for the Test Step Status

Test Status* [TODO   ▾]

This is Test Status that will be used in the requirement coverage calculation for Test Runs with this status.

[**Save**]   Cancel

Note that native Test Step statuses can't be modified nor deleted.

Requirement statuses

The status of a requirement tells you information about its current state, from a quality perspective. Is it covered with test cases? If so, has it been validated successfully? In which version?

Thus, whenever speaking about the "status of a requirement" we need to give it some additional context (e.g. "In which version?") since it depends on "where" and how you want to analyze it.

> ⓘ The status of a requirement indicates its coverage information along with its "state", depending on the results recorded for the Tests that do validate it.
>
> The status of a requirement is evaluated in some given context (e.g. for some version, some Test Plan and/or in some Test Environment).

In Xray, for a given requirement, considering the default settings, its coverage status may be:

- OK – requirement has been successfully and fully validated; all the Tests associated with the Requirement are PASSED
- NOK – requirement is unsuccessfully validated; at least one Test associated with the Requirement is FAILED
- NOTRUN – requirement has not been validated completely; at least one Test associated with the Requirement is TODO or ABORTED and there are no Tests with status FAILED
- UNKNOWN – requirement is in an unknown state; at least one Test associated with the Requirement is UNKNOWN and there are no Tests with status FAILED
- UNCOVERED – requirement is not covered with tests; the Requirement has no Tests associated to it

It's not possible to create custom requirement statuses.

You can see that in order to calculate a requirement coverage status, for some specific system version, we "just" need to take into account the status of the related Tests for that same version. We'll come back to this later on.

> ⓘ **Please note**
>
> Do not mix up the status of a given requirement with the "Requirement Status" custom field, which shows the status of a given requirement for a specific version, depending on the configuration under Custom Fields. More info on this custom field here.

Calculation of the status for a given Test Run

The status of a given Test Run is an attribute that is often calculated automatically based on the respective recorded step statuses. You can also enforce a specific status for a Test Run, which in turn may implicitly enforce specific step statuses (e.g., setting a Test Run as "FAIL" can set all steps as "FAIL").

This calculation is made by following these rules:

1. Obtain the test status mapped to each reported test step status; this is important as the actual test step statuses are not directly compared
2. Compare all the the previously mapped test statuses together
   a. if any of these statuses (e.g., "PASS") is in turn mapped to the coverage status "OK", then the other status wins; if both are mapped to "OK" then the highest ranked wins
   b. if any of these statuses is "FAIL", then the Test Run status will be "FAIL"
   c. if any of these statuses is in turn mapped to the coverage status "NOK", then the Test Run status will be that one
   d. if any of theses statuses is final, then wins over non-final ones
   e. of these statuses, the status with the highest ranking wins

The order of the steps is irrelevant for the purpose of the overall Test Run status value.

<u>Consequences</u>:

- if any test step status is "FAIL" then the calculated status for the Test Run will be "FAIL"
- if any of the test steps "contributes negatively" (i.e., is mapped to a Test status associated with the NOK coverage), then the status of Test Run will correspond to the mapped Test status of that step
- the Test Run will have status "PASS" if all the steps are marked as "PASS"
- the calculated status for the Test Run will only be "EXECUTING" if there is at least one step in "EXECUTING" or "TODO" (or a similar custom test step status) and all other steps are in "PASS" or equivalent (i.e., associated to the "OK" coverage status)

Configuration Example 1

Test Step Statuses

Manage custom Test Step statuses for Xray

Name	Description	Color	Test Status	Operations
PASS	The test step has passed		PASS	
TODO	The test step has not started		TODO	
EXECUTING	The test step is currently being executed		EXECUTING	
FAIL	The test step has failed		FAIL	
SKIP	The test step has been skipped.		PASS	Edit Delete
MYFAIL	FAIL but continue		FAIL_DISCARDABLE	Edit Delete
XTODO	XTODO		TODO	Edit Delete
XPASS2	XPASS2		MYFAIL	Edit Delete
XFAIL	XFAIL		MYFAIL	Edit Delete
XPASS	XPASS		FAIL	Edit Delete

The following table provides some examples given the Test Step Statuses configuration shown above.

Example #	Statuses of the steps /contexts (the order of the steps /contexts is irrelevant)	Calculated value for the status of the Test Run	Why?
1	• PASS • PASS • PASS	PASS	All steps are PASS, thus the joint value is PASS
2	• PASS • TODO • PASS	EXECUTING	At least one step status (i.e. TODO) is mapped to a non-final Test status
3	• PASS • FAIL • PASS	FAIL	One of the step statuses (i.e. FAIL) has higher ranking than the other ones

4	• XPASS • FAIL • PASS	FAIL	Since one of the steps is FAIL, then the run will be marked as FAIL.
5	• FAIL • XPASS • FAIL	FAIL	Since one of the steps is FAIL, then the run will be marked as FAIL.

6	• XFAIL
	  (=>MYFAIL=>NOK)
	• XPASS2
	  (=>MYFAIL=>NOK)
	• XPASS (=>FAIL=>NOK)
		FAIL	All mapped statuses map to a test status that in turn is associated to "NOK". Since one of them is FAIL, then the run will be marked as FAIL.

Configuration Example 2

Let's consider the following configuration.

Test Statuses

Manage custom Test Run statuses for Xray

| Name	Description	Final	Color	Requirement Status
PASS	The test run/iteration has passed	☑		OK
TODO	The test run/iteration has not started	☐		NOTRUN
EXECUTING	The test run/iteration is currently being executed	☐		NOTRUN
FAIL	The test run/iteration has failed	☑		NOK
ABORTED	The test run/iteration was aborted	☑		NOTRUN
BLOCKED		☐		NOTRUN
PENDING		☑		UNKNOWN
CUSTOM_PASS		☑		OK
CUSTOM_UNK		☐		UNKNOWN
CUSTOM_FAIL		☑		NOK
CUSTOM_FAIL2		☑		NOK
CUSTOM_PASS2		☑		OK

Test Step Statuses

Manage custom Test Step statuses for Xray

Name	Description	Color	Test Status
PASS	The test step has passed		PASS
TODO	The test step has not started		TODO
EXECUTING	The test step is currently being executed		EXECUTING
FAIL	The test step has failed		FAIL
SKIP	The test step has been skipped.		PASS
DUMMY_E			EXECUTING
DUMMY_F2			CUSTOM_FAIL2
DUMMY_F1			CUSTOM_FAIL
DUMMY_P1			CUSTOM_PASS
DUMMY_P2			CUSTOM_PASS2

Example #	Statuses of the steps/contexts (the order of the steps/contexts is irrelevant)	Calculated value for the status of the Test Run	Why?

1	• DUMMY_P2 (=>CUSTOM_PASS2=>OK) • DUMMY_P1 (=>CUSTOM_PASS=>OK)	CUSTOM_PASS2	We can see that both steps contribute in a "positive way" (i.e., they were successful as ultimately they are linked to successful coverage impact). Both statuses mapped to these test step statuses are associated with the "OK" coverage; as CUSTOM_PASS2 has higher ranking than CUSTOM_PASS, the run will be marked as "CUSTOM_PASS2".
2	• DUMMY_P2 (=>CUSTOM_PASS2=>OK) • DUMMY_P1 (=>CUSTOM_PASS=>OK) • PASS (=>PASS=>OK)	CUSTOM_PASS2	Similarly to the previous example. Any status wins the "PASS" status.
3	• DUMMY_F2 (=>CUSTOM_FAIL2=>NOK) • DUMMY_F1 (=>CUSTOM_FAIL=>NOK)	CUSTOM_FAIL2	We can see that both steps contribute in a "negative way" (i.e., they were not successful as ultimately they are linked to unsucessful coverage impact). Both statuses mapped to these test step statuses are associated with the "NOK" coverage; as CUSTOM_FAIL2 has higher ranking than CUSTOM_FAIL, the run will be marked as "CUSTOM_FAIL2".

Calculation of the status for a given Test

It is possible to calculate the status of a Test either by Version or Test Plan, in a specific Test Environment or globally, taking into account the results obtained for all Test Environments.

Analysis:

- **By Version:** For a given Test X, in order to calculate the coverage status for version V, we need to evaluate the related Tests Runs that were executed on that same version V. A special case is whenever you don't have versions or simply don't want to calculate the status based on a version (i.e. "None (latest execution)")
- **By Test Plan:** For a given Test X, in order to calculate the coverage status for Test Plan TP, we need to evaluate the related Tests Runs that were executed on Test Executions associated with Test Plan TP.
- **On a specific Test Environment:** For a given Test X, if a specific Environment is also chosen, then only Test Runs from Test Executions with this Environment will be considered. In case no Environment is specified then all Test Executions are considered (more info here).

What affects the calculation:

- the setting "Final statuses have precedence over non-final statuses" in Xray administration settings, in the tab "Test Statuses" (enabled by default)
- the existence of Test Runs for different Test Environments, in case the analysis is made for "All Environments"

Calculate the status of some Test, in version V or Test Plan TP, for Test Environment TE

1. This takes into account Test Runs in version V (as a result of Test Executions in version V) or Test Runs in Test Plan TP (within Test Executions associated with Test Plan TP)
2. If Test Environment is chosen, then only Tests Runs on that Environment (e.g. TE) will be considered
3. If "Final statuses have precedence over non-final statuses" is true, then:
    - final Test Run statuses will have higher ranking than non-final ones
    - only the latest Test Run is taken into account based on its "finished on" date
4. If "Final statuses have precedence over non-final statuses" is false, then:
    - only the latest Test Run is taken into account based on its "created" date (i.e. the creation date of the related Test Run entity - this happens when a Test Run is first executed, or when the user navigates into the execution screen)

Calculate the status of some Test, in version V or Test Plan TP, for "All Environments"

1. calculate the Test status for each Test Environment, based on all the implicit Test Environments from the relevant Test Executions (i.e. Test Executions in version V or Test Executions associated with Test Plan TP)
2. calculate the joint value for the Test status
    a. PASS has lowest ranking (i.e. for the calculated to be PASS, all calculated statuses must be PASS in the different Test Environments)
    b. if one is FAIL, then the calculated value will be FAIL
    c. otherwise, use the ranking of Test statuses

Examples

The following table provides some examples given the Test Statuses configuration shown above in the Managing Test Statuses section.

Example #	Statuses of the Test Runs (ordered by time of execution /creation, ascending)	Final statuses have precedence over non-final statuses	Calculated value for the status of the Test	Why?
1a	1. PASS 2. **PASS** 3. TODO	true	PASS	Latest **executed** Test Run (2) having a final status was PASS.
1b	1. PASS 2. PASS 3. **TODO**	false	TODO	Latest **created** Test Run (3) was TODO.
2a	1. **PASS** (env1) 2. **MYPASS2** (env2) 3. TODO (env2) 4. **PASS** (env3)	true	MYPASS2	Latest **executed** final Test Runs on each environment were PASS, MYPASS2, and PASS respectively. Since MYPASS2 (2) has a higher ranking then the calculated status will be MYPASS2.
2b	1. **PASS** (env1) 2. MYPASS2 (env2) 3. **TODO** (env2) 4. **PASS** (env3)	false	TODO	Latest **created** Test Runs on each environment were PASS, TODO, and PASS respectively. Since PASS has the lowest ranking, then TODO (3) will "win" and then the calculated status will be TODO
3	1. **PASS** (env1) 2. **TODO** (env2) 3. **PASS** (env3)	true	TODO	Latest **created** Test Runs on each environment were PASS, TODO, and PASS respectively. Although Test Environment "env2" has only a non-final Test Run, since there is no other Run for that environment, then it will be considered as the calculated status for that environment. Since PASS has the lowest ranking, then TODO (2) will "win" and then the calculated status will be TODO.
4	1. PASS (env1) 2. FAIL (env2) 3. PASS (env3)	true (or false)	FAIL	Latest **executed (or created)** final Test Runs on each environment were PASS, FAIL, and PASS respectively. Since the calculated status for one of the environments is FAIL, then the calculated status will be FAIL.
5	1. **PASS** (env1) 2. **MYPASS2** (env2) 3. TODO (env2) 4. **MYFAIL** (env3)	true	MYPASS2	Latest **executed** final Test Runs on each environment were PASS, MYPASS2, and MYFAIL respectively. MYPASS2 has a higher ranking than the other ones, thus the overall calculated value will be MYPASS2.
6	1. **PASS** (env1) 2. MYPASS2 (env2) 3. **TODO** (env2) 4. **MYFAIL** (env3)	false	MYFAIL	Latest **created** Test Runs on each environment were PASS, TODO, and MYFAIL respectively. MYFAIL has a higher ranking than the other ones, thus the overall calculated value will be MYFAIL.

Calculation of the status for a given requirement

It is possible to calculate the status of a Requirement either by Version or Test Plan, in a specific Test Environment or globally, taking into account the results obtained for all Test Environments.

Analysis :

- **By Version:** For a given requirement X, in order to calculate the coverage status for version V, we need to evaluate the related Tests statuses that were executed on that same version V.
- **By Test Plan:** For a given requirement X, in order to calculate the coverage status for Test Plan TP, we need to evaluate the related Tests statuses that were executed on Test Executions associated with Test Plan TP.
- **On a specific Test Environment:** For a given Test X, if a specific Environment is also chosen, then only Test Runs from Test Executions with this Environment will be considered. In case no Environment is specified then all Test Executions are considered (more info here).

The algorithm is similar to the overall calculation of the Test status taking into account the results obtained for different Test Environments.

In other words, the status for each linked and "relevant" Test case is calculated and in the end a joint calculation is done for a virtual Test case. The requirement status will correspond to the mapped value for the status that was calculated for this virtual Test.

The Tests that will be considered as covering the requirement are not just the ones directly linked to the requirement. In fact, they may either be direct ones or ones linked to sub-requirements. This list can be further restricted if Test Sets are being used for defining the scope of coverage (i.e. the list of Tests relevant for the coverage calculation for some version).

Algorithm :

1. Obtain the list of Tests that directly or indirectly through Sub-Requirements (info here) cover the requirement
    - This depends on the Requirement Coverage configuration
2. Calculate the Test status for all the Tests individually, in version V or Test Plan TP
    - This takes into account Test Runs in version V (as a result of Test Executions in version V) or Test Runs in Test Plan TP (within Test Executions associated with Test Plan TP)
    - If a specific Environment is also chosen, then only Test Runs from Test Executions with this Environment will be considered. In case no Environment is specified then all Test Executions are considered (more info here).
3. Calculate the "joint" status of all the previous Test statuses (i.e. by comparing together each Test status)
4. Calculate the requirement status mapped to the previous calculated Test status

What affects the calculation:

- the requirement coverage strategy defined in the respective setting, in Xray administration settings, in the tab "Requirement Coverage" (more info here)
    - the strategy is used to find out the relevant Tests considered for the calculation
- the separation of concerns setting, in Xray administration settings, in the tab "Requirement Coverage" (more info here)
    - this affects how the status of a Test Run contributes to the status of each linked requirement; if disabled, users can enforce a specific mapping for each requirement, otherwise the Test Run status will contribute in the same way to all the linked requirements
- indirectly, the setting "Final statuses have precedence over non-final statuses" in Xray administration settings, in the tab "Test Statuses" (enabled by default)
- the existence of Test Runs for different Test Environments, in case the analysis is made for "All Environments"

Requirements and sub-requirements conjunction

When a requirement has some sub-requirements, then the calculated status for the parent requirement depends not only on its status calculated per-si but also on the status of each individual sub-requirement.

The calculation follows the rules described in the following table.

REQ \ SUB-REQ	OK	NOK	NOT RUN	UNKNOWN	UNCOVERED
OK	OK	NOK	NOT RUN	UNKNOWN	OK
NOK	NOK	NOK	NOK	NOK	NOK
NOT RUN	NOT RUN	NOK	NOT RUN	UNKNOWN	NOT RUN
UNKNOWN	UNKNOWN	NOK	UNKNOWN	UNKNOWN	UNKNOWN
UNCOVERED	OK	NOK	NOT RUN	UNKNOWN	UNCOVERED

From another perspective, you would obtain the same value for the calculation of the status of the parent requirement if you consider that it is being covered by all the explicitly linked Tests and also the ones linked to sub-requirements.

Consequences :

- the parent requirement is OK if it is OK *per se* and the sub-requirements are either UNCOVERED or also OK
- the parent requirement is NOK if it is NOK *per se* or if any of the sub-requirements is NOK
- the parent requirement is only UNCOVERED if neither the parent requirement is covered *per se* nor the sub-requirements are covered

---

ⓘ **Rationale**

Even if you have sub-requirements, when you have tests that are directly linked to the parent requirement, Xray assumes that you are validating the requirement directly. Thus, it's irrelevant if the sub-requirements are uncovered by tests.

Examples

The following table provides some examples given the Test Statuses configuration shown above in the Managing Test Statuses section.

Example #	Statuses of the related Tests (sub-requirements, whenever are present using the notation subReqX)	Calculated value for the status of the requirement	Why?
1	1. PASS 2. PASS 3. PASS	OK	All Tests are passed (it is similar to having just one virtual test that would be considered PASS and thus mapped to the OK status of the requirement)
2	1. PASS 2. PASS 3. **TODO**	NOT RUN	One of the Tests (3) is TODO, which has higher ranking than PASS.
3	1. PASS 2. PASS 3. **FAIL**	NOK	One of the Tests (3) is FAIL, which has higher ranking than PASS.

4	1. PASS
	2. subReq1 => OK
	   a. PASS
	3. subReq2 => NOK
	   a. PASS
	   b. **FAIL**
		NOK	One of the Tests (3b) is FAIL, thus subReq2 will be considered as NOK. Since it is NOK, then the parent requirement status will be NOK.

5	1. PASS
	2. subReq1 => NOTRUN
	   a. TODO
	3. subReq2 => OK
	   a. PASS
	   b. PASS
		NOT RUN	One of the subRequirements (subReq1) is NOT RUN, thus the calculated status, whenever doing the conjunction with the parent requirement status, will be NOT RUN.

6	1. PASS
	2. subReq1 => UNCOVERED
	   a. (no Tests associated)
	3. subReq2 => UNCOVERED
	   a. (no Tests associated)
		OK	Since all sub-requirements are uncovered and the parent requirement is covered directly by one Test (1), which is currently PASS, then the calculated "OK" status will be based on that Test.

Setup information for some possible use cases

1. I want to skip some steps and proceed as they didn't exist
   a. create a "Test Step Status" (e.g. "SKIP"), mapped to the Test Status "PASS"
2. I want to fail a Test Run but I <u>don't</u> want to mark the requirement as being NOT OK because this failure can be discarded
   a. create a "Test Status" (e.g. "FAIL_DISCARD") , non-final and mapped to the requirement status "UNKNOWN"; setting the status as non-final will give priority to other Test Runs you may have for that Test, If "Final Statuses have precedence over non-final" flag is enabled
   b. create a "Test Step Status" (e.g. "IRRELEVANT_FAIL") and map it to the Test Status created in the previous step
3. I want to always see, for a given Test, the status of Test based on the last run scheduled for it, no matter if it was completed (i.e. in a final status) or not
   a. just uncheck the setting "Final Statuses have precedence over non-final"
4. I want to execute some steps, set them as failed or passed, but I don't want them to reflect immediately in the status of the Test Run

a. create custom, non-final, Test statuses for passing and failure (e.g. "MYPASS", "MYFAIL"), mapped to the OK and NOK requirement statuses, respectively
b. create your own custom Test Step statuses for passed and failure (e.g. "PASS_CONTINUE" and FAIL_CONTINUE"), mapped to the previously created Test statuses