

Enhanced querying with JQL

- [JQL Functions](#)
- [Custom Fields](#)

JQL Functions

The following JQL functions are available for querying Xray issues in the Issue Search Page. They enable you to query the relationships between Xray issue types.

JQL Function	Parameters	Description	Example
testTestSet	P1 - Test Issue Key	Returns a list of Test Set issues associated with the input Test issue key.	<pre>issuetype = 'Test Set' and key in testTestSet ('DEMO-1')</pre>
testSetTests	P1 - Test Set Issue Key/Filter of Test Sets	Returns a list of Test issues associated with the input Test Set issue key.	<pre>(1) issuetype = 'Test' and key in testSetTests ('DEMO-5') (2) issuetype = 'Test' and key in testSetTests ('Test sets saved filter')</pre>
testsWithNoTestSet	None	Returns a list of Test issues not associated with a Test Set .	<pre>key in testsWithNoTestSet()</pre>
testPreConditions	P1 - Test Issue Key	Returns the Pre-Condition issues associated with the input Test issue key.	<pre>issuetype = 'Pre-Condition' and key in testPreConditions ('DEMO-1')</pre>
preConditionTests	P1 - Pre-Condition Issue Key	Returns the Test issues associated with the input Pre-Condition issue key.	<pre>issuetype = 'Test' and key in preConditionTests ('DEMO-1')</pre>
testRequirements	P1 - Test Issue Key	Returns a list of Requirement issues associated with the input Test issue key.	<pre>issuetype = 'Feature' and key in testRequirements ('DEMO-1')</pre>

requirementTests	P1 - Requirement Issue Key/Filter of Requirement Issues	Returns a list of Test issues associated with the input Requirement issue key or saved filter with Requirements.	(1) issuetype = 'Test' and key in requirementTests ('DEMO-10') (2) issuetype = 'Test' and key in requirements ('Requirements saved filter')
testsWithReqVersion	P1 - Project Name/Key/Id P2 - Fix Version P3 - Fix Version (Optional) ... Pn - Fix Version (Optional)	Returns a list of Test issues associated with the Requirement issues of the input Fix Versions of the specified project.	issuetype = 'Test' and issue in testsWithReqVersion('DEMO', 'v1.0', 'v1.1')
testsWithTestSetVersion	P1 - Project Name/Key/Id P2 - Fix Version P3 - Fix Version (Optional) ... Pn - Fix Version (Optional)	Returns a List of Test issues associated with the Test Set issues of the input Fix Versions of the specified project.	issuetype = 'Test' and issue in testsWithTestSetVersion('DEMO', 'v1.0', 'v1.1')
testExecutionTests	P1 - Test Execution Issue Key/Id or Filter ID P2 - Test Run Status list separated by " " (pipe) (Optional) P3 - User assigned to execute Test Run (Optional). If you pass "" as argument then the function will look for unassigned Test Runs P4 - Defects Flag with value in true or false (optional). P5 - User who executed the Test Run (optional).	Returns a List of Test issues associated with the input Test Execution issues from P1 optionally filtered by the current test run status for each Test issue. Parameter P1 can either be a single Test Execution issue key, an ID or a saved filter containing multiple Test Execution issues. Possible Test Run Status values are: PASS, FAIL, EXECUTING, ABORTED, TODO and all custom statuses. P3 corresponds to the user assigned to execute the Test Run, while P5 corresponds to the one who actually executed it. If you pass true as the value for P4, the query returns all Tests from a particular set of Test Executions that have been failed but no Defects were created.	(1) issuetype = 'Test' and issue in testExecutionTests('DEMO-9') (2) issuetype = 'Test' and issue in testExecutionTests('DEMO-9', 'PASS') (3) issuetype = 'Test' and issue in testExecutionTests('DEMO-9', 'PASS', 'user A') (4) issuetype = 'Test' and issue in testExecutionTests(

			<pre> 'Saved Test Execution Filter', 'PASS') (5) issuetype = 'Test' and issue in testExecutionTes ts('Saved Test Execution Filter', '', 'user A') (6) issue in testExecutionTes ts('Saved Test Execution Filter', '', 'user A', 'true') (7) issue in testExecutionTes ts('Saved Test Execution Filter', '', '', 'false', 'admin') </pre>
testsWithoutTestExecution	None	Returns a list of Tests that are not associated with a Test Execution to be executed	<pre> (1) issuetype = Test and issue in testsWithoutTest Execution() </pre>

requirements	<p>P1 - Status list separated by " " (pipe)</p> <p>P2 - <i>Project</i> (Optional)</p> <p>P3 - <i>Version to calculate requirement status</i> (Optional)</p> <p>P4 - <i>Test Environment</i> (Optional)</p> <p>P5 - <i>Flat</i> (Optional)</p> <p>P6 - <i>ToDate</i> (Optional)</p>	<p>Returns a list of Requirement Issues with the provided coverage status.</p> <p>If a specific analysis version is required, then the <i>Project and Version</i> parameters must be filled.</p> <p><u>Optional</u> filters include:</p> <p><i>Test Environment</i>, for taking into account the Test Executions made for that environment. For analyzing the joint values of all environments, "" should be used. For taking into account the Test Executions without any Test Environment assigned, then "__NULL__" should be used.</p> <p><i>Flat</i> that indicates whether all Requirements (not only parents) should be searched. If "Flat" is not provided, the default value is 'false'.</p> <p><i>ToDate</i> considers only those requirements executions before a specific date/time (the date literal must follow the ISO8601 format).</p>	<pre>(1) issue in requirements ('OK') (2) priority = Major and fixVersion <= 'v3.0' and issue in requirements ('NOK', 'Calculator', 'V4.0') (3) issue in requirements ('NOK', '', '', '', '', '2014-01- 01') (4) issue in requirements ('OK', 'Calculator', 'v1.0', 'chrome' 'false' '2014-08-30') (5) issue in requirements ('NOK', 'Calculator', 'v2.0', '', 'true')</pre>
---------------------	--	---	---

requirementsWithStatusByTestPlan	<p>P1 - Status list separated by " " (pipe)</p> <p>P2 - Test Plan Issue Key</p> <p>P3 - Test Environment (Optional)</p> <p>P4 - Flat (Optional)</p> <p>P5 - ToDate (Optional)</p>	<p>Returns a list of Requirement Issues with the coverage status calculated for the given Test Plan issue.</p> <p><u>Optional</u> filters include:</p> <p><i>Test Environment</i>, for taking into account the Test Executions made for that environment. For analyzing the joint values of all environments, "" should be used. For taking into account the Test Executions without any Test Environment assigned, then "__NULL__" should be used.</p> <p><i>Flat</i> that indicates whether all Requirements (not only parents) should be searched. If "Flat" is not provided, the default value is 'false'.</p> <p><i>ToDate</i> considers only those requirements executions before a specific date/time (the date literal must follow the ISO8601 format).</p>	<p>(1)</p> <pre>issue in requirementsWith StatusByTestPlan ('OK', 'TP-123')</pre> <p>(2)</p> <pre>issue in requirementsWith StatusByTestPlan ('NOK', 'TP-123', '', 'true')</pre> <p>(3)</p> <pre>issue in requirementsWith StatusByTestPlan ('NOK', 'TP-123', 'Android', 'false', '2014-01-01')</pre>
defectsCreatedDuringTesting	<p>P1 - Test Issue Key/Filter of Test Issues</p>	<p>Return a list of defects created during the execution of Tests.</p>	<p>(1)</p> <pre>issue in defectsCreatedDu ringTesting()</pre> <p>(2)</p> <pre>issue in defectsCreatedDu ringTesting ("TEST-123")</pre> <p>(3)</p> <pre>issue in defectsCreatedDu ringTesting ("saved_filter")</pre>

defectsCreatedDuringTestExecution	<p>P1 - Test Execution issue Key or Test Execution based Filter</p> <p>P2 - List of users separated by " " (pipe). (Optional)</p>	<p>Returns a list of Defects created in the execution page and can optionally be filtered by the Defect Issue Assignee username.</p>	<pre>(1) issue in defectsCreatedDuringTestExecution(TEST-123) (2) issue in defectsCreatedDuringTestExecution(saved_filter) (3) issue in defectsCreatedDuringTestExecution(saved_filter, 'user1 user2') (4) issue in defectsCreatedDuringTestExecution(TEST-123, 'user1 user2')</pre>
defectsCreatedForRequirement	<p>P1 - Requirement key or saved filter</p>	<p>Returns a list of defects created during the testing of specific requirements.</p>	<pre>(1) issue in defectsCreatedForRequirement("REQ-123") (2) issue in defectsCreatedForRequirement("saved_filter")</pre>
manualTestsWithoutSteps	<p>None</p>	<p>Returns a list of manual tests that have no test steps.</p>	<pre>issue in manualTestsWithoutSteps()</pre>
testTestExecutions	<p>P1 - Test Issue Key/Id or Filter Name/Id</p> <p>P2 - Test Run Status list separated by " " (pipe) (Optional)</p>	<p>Returns a list of test executions associated with the input Test Issues from P1 optionally filtered by the current test run status for each Test issue.</p> <p>Parameter P1 can either be a single Test issue key or Id or a saved filter name or id containing multiple Test Execution issues.</p> <p>Possible Test Run Status values are: PASS, FAIL, EXECUTING, ABORTED, TODO and all custom statuses.</p>	<pre>(1) issuetype = 'Test Execution' and issue in testTestExecutions('DEMO-9') (2) issuetype = 'Test Execution' and issue in testTestExecutions('DEMO-9', 'PASS') (3) issuetype = 'Test Execution' and issue in testTestExecutions('Saved Test Filter', 'PASS')</pre>

testExecWithTestRunsAssignedToUser	<p><i>P1 - Username (Optional)</i></p> <p><i>P2 - Status (Optional) Username is required in case we use this parameter</i></p>	<p>Returns a list of test executions where a user has at least one test run assigned to him. You can optionally specify a user with P1, or if the user is omitted the current user will be used. Note that if you are not logged in to JIRA, a user must be specified.</p> <p>If you use status parameter then user is required</p>	<pre>(1) issuetype = 'Test Execution' and issue in testExecWithTest RunsAssignedToUser() (2) issuetype = 'Test Execution' and issue in testExecWithTest RunsAssignedToUser('userDPC') (3) issuetype = 'Test Execution' and issue in testExecWithTest RunsAssignedToUser('userDPC', "FAIL")</pre>
testSetPartiallyIn	<p>P1 - Test Execution Issue Key /Test Plan Issue Key/Id or Filter Id</p>	<p>Return a list of Test Sets that have at least one test in P1.</p>	<pre>(1) issuetype = 'Test Set' and issue in testSetPartiallyIn('DEMO-15') (2) issuetype = 'Test Set' and issue in testSetPartiallyIn('testExecList') (3) issuetype = 'Test Set' and issue in testSetPartiallyIn('testPlanList')</pre>
testSetFullyIn	<p>P1 - Test Execution Issue Key /Test Plan Issue Key/Id or Filter Id</p>	<p>Return a list of Test Sets that have all its tests in P1.</p>	<pre>(1) issuetype = 'Test Set' and issue in testSetFullyIn('DEMO-15') (2) issuetype = 'Test Set' and issue in testSetFullyIn('testExecList') (3) issuetype = 'Test Set' and issue in testSetFullyIn('testPlanList')</pre>

testPlanTests	<p>P1 - Test Plan Key or Filter Name</p> <p>P2 - Status (Optional)</p> <p>P3 - Environment (Optional)</p>	<p>Returns a list of tests that are associated with the test plan chosen or the JIRA Filter name.</p> <p>The "status" parameter is optional and allows to filter Test issues in a specific Plan with the specified execution status. If the "status" parameter is present, users might also pass the "environment" parameter. If this parameter is filled, Xray will provide all Tests in a Test Plan that are in the specified "status" and for the specified "environment".</p>	<p>(1)</p> <pre>issue in testPlanTests ("DEMO-10")</pre> <p>(2)</p> <pre>issue in testPlanTests ("DEMO-10", " TODO")</pre> <p>(3)</p> <pre>issue in testPlanTests ("DEMO-10", " TODO", "IOS")</pre>
testPlanTestExecutions	P1 - Test Plan Key	Returns a list of test executions that are associated with the test plan.	<pre>issue in testPlanTestExec utions("DEMO- 10")</pre>
testPlanRequirements	P1 - Test Plan Key/Filter of Test Plans	Returns the Requirement issues that are indirectly associated, through Test issues, with a Test Plan or a saved filter of Test Plans.	<p>(1)</p> <pre>issue in testPlanRequirem ents("DEMO-20")</pre> <p>(2)</p> <pre>issue in testPlanRequirem ents("Test Plans saved filter")</pre>
testTestPlan	P1 - Test Issue Key	Returns a List of Test Plan issues associated with the input Test issue key.	<pre>issuetype = 'Test Plan' and key in testTestPlan ('DEMO-1')</pre>
testRepositoryFolderTests	<p>P1 - Project Key</p> <p>P2 - Folder Path</p> <p>P3 - Flatten (Optional)</p>	<p>Returns the list of Tests contained in a folder (P2) of the Test Repository of a Project (P1)</p> <p>May optionally include the Tests in sub-folders by setting Flatten (P3) to "true".</p>	<p>(1)</p> <pre>issue in testRepositoryFolderT ests("CALC", 'Parent /Child')</pre> <p>(2)</p> <pre>issue in testRepositoryFolderT ests("CALC", 'Parent /Child', "true")</pre>
testPlanFolderTests	<p>P1 - Test Plan Key</p> <p>P2 - Folder Path</p> <p>P3 - Flatten (Optional)</p> <p>P4 - Test Run Status (Optional)</p> <p>P5 - Test Environment (Optional)</p>	<p>Returns the list of Tests contained in a folder (P2) of a Test Plan (P1).</p> <p>May optionally include the Tests in sub-folders by setting Flatten (P3) to "true".</p> <p>Can also filter by Tests Run Status (P4) for a given Test Environment (P5).</p> <p>To analyze the joint values of all Test Environments, "" should be used. To analyze the Test Executions without any Test Environment assigned, then "__NULL__" should be used.</p>	<p>(1)</p> <pre>issue in testPlanFolderTests (CALC-10, 'Parent /Child')</pre> <p>(2)</p> <pre>issue in testPlanFolderTests (CALC-10, 'Parent /Child', "true")</pre> <p>(3)</p> <pre>issue in testPlanFolderTests (CALC-10, 'Parent /Child', "true", "TODO FAIL", "windows")</pre>

Custom Fields

Xray also provides custom fields to allow more refined queries when searching for issues.

JQL Function	Issue Type	Description	Example
Test Type	Test	The Test type: Manual;Automated[Cucumber];Automated[Generic]	<pre>issuetype = 'Test' and "Test Type" = "Manual"</pre>
TestRunStatus	Test	<p>This is a calculated custom field that provides the latest Test Run status based on the current "Test Run Status Version Strategy" option configured in the Xray administration.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> Test Run Status</p> <p>The latest Test Run Status is calculated based on the latest final Test Run (i.e., latest finish date) or, in case there are no final Test Runs, the latest non-final Test Run (i.e., latest start date). Please see the custom fields preferences page.</p> </div>	<pre>issuetype = 'Test' and TestRunStatus in ("FAIL", "ABORTED") - issuetype = 'Test' and TestRunStatus = "PASS" - issuetype = 'Test' and TestRunStatus is EMPTY</pre>
Requirement Status	Requirement	<p>This is a calculated custom field that provides the requirement coverage status.</p> <p>Possible status values are:</p> <p>OK - All tests passed the requirement</p> <p>NOK - At least one test failed</p> <p>NOTRUN - At least one test did not run</p> <p>UNCOVERED - The requirements is not associated with tests</p> <p>testTestExecutions</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> Requirement Coverage</p> <p>For more information, please check out our page dedicated to requirements coverage.</p> <p>If the Requirements Coverage Strategy depends on the version, then you must also include the actual version name and the status when you do the search. The syntax: "[version name] - [status]"</p> </div>	<pre>issuetype = 'New Feature' and "Requirement Status" = "OK" - issuetype = 'New Feature' and "Requirement Status" in ("NOTRUN", "UNCOVERED") - issuetype = 'New Feature' and "Requirement Status" = "v1.0 - OK"</pre>
Steps Count	Test	The number of Steps in a Manual Test	<pre>issuetype = 'Test' and "Steps Count" = 3</pre>



The **Test Set Status** and **Test Plan Status** custom fields, mentioned in [Custom Fields Preferences](#), are not queryable.