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.	`issuetype = 'Test Set' and key in testTestSet ('DEMO-1')`
testSetTests	P1 - Test Set Issue Key/Filter of Test Sets	Returns a list of Test issues associated with the input Test Set issue key.	`(1) issuetype = 'Test' and key in testSetTests ('DEMO-5') (2) issuetype = 'Test' and key in testSetTests ('Test sets saved filter')`
testsWithNoTestSet	None	Returns a list of Test issues not associated with a Test Set.	`key in testsWithNoTestSet()`
testPreConditions	P1 - Test Issue Key	Returns the Pre-Condition issues associated with the input Test issue key.	`issuetype = 'Pre-Condition' and key in testPreConditions('DEMO-1')`
preConditionTests	P1 - Pre-Condition Issue Key	Returns the Test issues associated with the input Pre-Condition issue key.	`issuetype = 'Test' and key in preConditionTests('DEMO-1')`
testRequirements	P1 - Test Issue Key/Filter name of Tests	Returns a list of Requirement issues associated with the input Test issue key/Filter of tests.	`(1) issuetype = 'Feature' and key in testRequirements ('DEMO-1') (2) issuetype = 'Feature' and key in testRequirements ('Tests saved filter')`

requiremen tTests	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 requirementTests ('Requirements saved filter')`

testsWithR eqVersion	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`
			
			`testsWithReqVers ion('DEMO',`
			
			`'v1.0', 'v1.1')`

testsWithT estSetVersi on	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`
			
			`testsWithTestSet Version( 'DEMO',`
			
			`'v1.0', 'v1.1')`

testExecuti onTests	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* optiona lly 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 testExecutionTes ts('DEMO-9')`
			
			`(2)`
			
			`issuetype = 'Test'`
			
			`    and issue in testExecutionTes ts('DEMO-9',`
			
			`'PASS')`
			
			`(3)`
			
			`issuetype = 'Test'`
			
			`    and issue in testExecutionTes ts('DEMO-9',`
			
			`'PASS',`
			
			`'user A')`
			
			`(4)`
			
			`issuetype = 'Test'`
			
			`    and issue in testExecutionTes ts(`

			`'Saved Test Execution Filter',`
			
			`'PASS')`
			
			(5)
			
			`issuetype = 'Test'`
			
			`   and issue in testExecutionTests(`
			
			`'Saved Test Execution Filter',`
			
			`'',`
			
			`'user A')`
			
			(6)
			
			`issue in testExecutionTests(`
			
			`'Saved Test Execution Filter',`
			
			`'',`
			
			`'user A', 'true')`
			
			(7)
			
			`issue in testExecutionTests(`
			
			`'Saved Test Execution Filter',`
			
			`'','', 'false', 'admin')`

testsWithoutTestExecution	None	Returns a list of Tests that are not associated with a Test Execution to be executed	(1) `issuetype = Test and issue in testsWithoutTestExecution()`

| requirements | P1 - Status list separated by "\|" (pipe)

P2 - *Project (Optional)*

P3 - *Version to calculate requirement status (Optional)*

P4 - *Test Environment (Optional)*

P5 - *Flat (Optional)*

P6 - *ToDate (Optional)* | Returns a list of Requirement Issues with the provided coverage status.

If a specific analysis version is required, then the *Project and Version* parameters must be filled.

Optional filters include:

Test Environment, 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.

Flat that indicates whether all Requirements (not only parents) should be searched. If "Flat" is not provided, the default value is 'false'.

ToDate considers only those requirements executions before a specific date/time (the date literal must follow the ISO8601 format). | (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')` |

requirementsWithStatusByTestPlan	P1 - Status list separated by "|" (pipe)
	
	P2 - Test Plan Issue Key
	
	P3 - Test Environment (Optional)
	
	P4 - Flat (Optional)
	
	P5 - ToDate (Optional)
		Returns a list of Requirement Issues with the coverage status calculated for the given Test Plan issue.
		
		Optional filters include:
		
		Test Environment, 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.
		
		Flat that indicates whether all Requirements (not only parents) should be searched. If "Flat" is not provided, the default value is 'false'.
		
		ToDate considers only those requirements executions before a specific date/time (the date literal must follow the ISO8601 format).
			`(1)`
			
			`issue in`
			
			`requirementsWith StatusByTestPlan ('OK', 'TP-123')`
			
			`(2)`
			
			`issue in`
			
			`requirementsWith StatusByTestPlan ('NOK',`
			
			`'TP-123',`
			
			`'',`
			
			`'true')`
			
			`(3)`
			
			`issue in`
			
			`requirementsWith StatusByTestPlan ('NOK',`
			
			`'TP-123',`
			
			`'Android',`
			
			`'false',`
			
			`'2014-01-01')`

defectsCreatedDuringTesting	P1 - Test Issue Key/Filter of Test Issues	Return a list of defects created during the execution of the specified Tests.	`(1)`
			
			`issue in defectsCreatedDu ringTesting()`
			
			`(2)`
			
			`issue in defectsCreatedDu ringTesting ("TEST-123")`
			
			`(3)`
			
			`issue in defectsCreatedDu ringTesting ("saved_filter")`

defectsCreatedDuringTestExecution	P1 - Test Execution issue Key or Test Execution based Filter
	
	P2 - List of users separated by "|" (pipe). (Optional)
		Returns a list of Defects created during the execution of the specified Test Executions; can optionally be filtered by the Defect Issue Assignee username.	`(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')`

defectsCreatedForRequirement	P1 - Requirement key or saved filter	Returns a list of defects created during the execution of Tests covering the specified requirements.	`(1)`
			
			`issue in defectsCreatedForRequirement ("REQ-123")`
			
			`(2)`
			
			`issue in defectsCreatedForRequirement ("saved_filter")`

manualTestsWithoutSteps	None	Returns a list of manual tests that have no test steps.	`issue in manualTestsWithoutSteps()`

testTestExecutions	P1 - Test Issue Key/Id or Filter Name/Id
	
	P2 - Test Run Status list separated by "|"(pipe) (Optional)
		Returns a list of test executions associated with the input Test Issues from *P1* optionally filtered by the current Test status in each Test Execution issue.
		
		Parameter *P1* can either be a single Test issue key or Id or a saved filter name or id containing multiple Test issues.
		
		Possible Test Run Status values are: PASS, FAIL, EXECUTING, ABORTED, TODO and all custom statuses.
			`(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')`

testExecWi thTestRuns AssignedT oUser	*P1 - Username (Optional)*
	
	P2 - Status (Optional) Username is required in case we use this parameter
		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.
		
		If you use status parameter then user is required
			(1)
			
			`issuetype = 'Test Execution' and`
			
			`  issue in testExecWithTest RunsAssignedToUs er()`
			
			(2)
			
			`issuetype = 'Test Execution' and`
			
			`  issue in testExecWithTest RunsAssignedToUs er('userDPC')`
			
			(3)
			
			`issuetype = 'Test Execution' and`
			
			`  issue in testExecWithTest RunsAssignedToUs er('userDPC', "FAIL")`

testSetPa rtiallyIn	P1 - Test Execution Issue Key /Test Plan Issue Key/Id or Filter Id	Return a list of Test Sets that have at least one test in *P1*.	(1)
			
			`issuetype = 'Test Set' and`
			
			`  issue in testSetPartially In('DEMO-15')`
			
			(2)
			
			`issuetype = 'Test Set' and`
			
			`  issue in testSetPartially In ('testExecList')`
			
			(3)
			
			`issuetype = 'Test Set' and`
			
			`  issue in testSetPartially In ('testPlanList')`

testSetFu llyIn	P1 - Test Execution Issue Key /Test Plan Issue Key/Id or Filter Id	Return a list of Test Sets that have all its tests in *P1*.	(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')`

testPlanTests	P1 - Test Plan Key/Filter of Test Plans
	
	P2 - Status (Optional)
	
	P3 - Environment (Optional)
		Returns a list of tests that are associated with the test plan.
		
		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".
			(1)
			
			`issue in testPlanTests ("DEMO-10")`
			
			(2)
			
			`issue in testPlanTests ("Test Plans saved filter"," TODO")`
			
			(3)
			
			`issue in testPlanTests ("DEMO-10"," TODO")`
			
			(4)
			
			`issue in testPlanTests ("DEMO-10"," TODO","IOS")`

testPlanTestExecutions	P1 - Test Plan Key/Filter of Test Plans	Returns a list of test executions that are associated with a Test Plan or a saved filter of Test Plans.	(1)
			
			`issue in testPlanTestExec utions("DEMO-10")`
			
			(2)
			
			`issue in testPlanTestExec utions("Test Plans saved filter")`

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.	(1)
			
			`issue in testPlanRequirem ents("DEMO-20")`
			
			(2)
			
			`issue in testPlanRequirem ents("Test Plans saved filter")`

testTestPlan	P1 - Test Issue Key	Returns a List of Test Plan issues associated with the input Test issue key.	`issuetype = 'Test Plan'`
			
			`    and key in testTestPlan ('DEMO-1')`

testRepositoryFolderTests	P1 - Project Key
	
	P2 - Folder Path
	
	P3 - Flatten (Optional)
		Returns the list of Tests contained in a folder (P2) of the Test Repository of a Project (P1)
		
		May optionally include the Tests in sub-folders by setting Flatten (P3) to "true".
			(1)
			
			`issue in testRepositoryFolderT ests("CALC", 'Parent /Child')`
			
			(2)
			
			`issue in testRepositoryFolderT ests("CALC", 'Parent /Child', "true")`

testPlanFolderTests	P1 - Test Plan Key P2 - Folder Path *P3 - Flatten (Optional)* *P4 - Test Run Status (Optional)* *P5 - Test Environment (Optional)*	Returns the list of Tests contained in a folder (P2) of a Test Plan (P1). May optionally include the Tests in sub-folders by setting Flatten (P3) to "true". Can also filter by Tests Run Status (P4) for a given Test Environment (P5). 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.	(1) issue in testPlanFolderTests (CALC-10, 'Parent /Child') (2) issue in testPlanFolderTests (CALC-10, 'Parent /Child', "true") (3) issue in testPlanFolderTests (CALC-10, 'Parent /Child', "true", "TODO|FAIL", "windows")
projectParentRequirements	P1 - Project Key	Returns the list of Requirement issues, from a given Project, which are not Sub-requirements	(1) issue in projectParentRequirements("CALC")

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]	`issuetype = 'Test'` `    and "Test Type" = "Manual"`

TestRunStatus	Test	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.
		
		ⓘ **Test Run Status**
		
		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.
			`issuetype = 'Test'`
			`    and TestRunStatus in ("FAIL", "ABORTED")`
			
			—
			
			`issuetype = 'Test'`
			`    and TestRunStatus = "PASS"`
			
			—
			
			`issuetype = 'Test'`
			`    and TestRunStatus is EMPTY`

| Requirement Status | Requirement | This is a calculated custom field that provides the **requirement coverage status.**

Possible status values are:

OK - All tests passed the requirement

NOK - At least one test failed

NOTRUN - At least one test did not run

UNCOVERED - The requirements is not associated with tests

testTestExecutions

ⓘ **Requirement Coverage**

For more information, please check out our page dedicated to requirements coverage.

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]"** | ```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"``` |
| **Steps Count** | Test | The number of Steps in a Manual Test | ```issuetype = 'Test'```

```   and "Steps Count" = 3``` |

⚠ The **Test Set Status** and **Test Plan Status** custom fields, mentioned in Custom Fields, are not queryable.