OrientDB API Scripts

[**REST APIs for KPI assessment**](#_nmdbgk6s9zhj) **2**

[requirementsMappingToTestCasesRate()](#_18wgcvijgxdb) 2

[testCaseExecutedRate()](#_i321qqjw4h6j) 3

[testCasePassRate()](#_3liw66j4hhm2) 4

[troubleTicketsFromOlderReleases()](#_aumdwwlcupl5) 5

[progressMetricsForRiskPrediction()](#_onybodh5zz3p) 7

[featureDocumentationComplete()](#_edc67dstrjri) 9

[meanTimeToRepair()](#_mwexii1vb0kl) 11

[preliminaryCapacityAndMemoryAnalysis()](#_afq9m5d4yu4w) 12

[regressionCompleted()](#_sa5t6rn141xl) 13

[regressionPassRate()](#_s28mcsccadv3) 14

[upgradePathsValidated()](#_7w81knzgv2wd) 15

[**Triggers**](#_hhj79a5o9by5) **17**

[onCreated()](#_1wdz9ecpjlmv) 17

[onUpdated()](#_jctnvxxm6mf7) 18

"Executed Nodal Testcases": "QualityManagement/testCaseExecutedRate/71e1820c-5db9-4429-965c-b7164a00e265",

"Nodal Testcases Pass Rate": "QualityManagement/testCasePassRate/71e1820c-5db9-4429-965c-b7164a00e265",

"Feature Documentation": "Test/featureDocumentationComplete/97616dab-48c8-4b1b-a129-56e0fa9a589c",

"Requirements-Testcases Mapping":"QualityManagement/requirementsMappingToTestCasesRate/9fe0d9e7-92dc-4b4c-9906-314c49c0bd64",

"Carry-over Trouble Ticket Resolution": "QualityManagement/troubleTicketsFromOlderReleases/a0da7296-042b-4a50-aa81-2a1271f5020b",

"Preliminary Capacity and Memory Analysis":"Test/preliminaryCapacityAndMemoryAnalysis/97616dab-48c8-4b1b-a129-56e0fa9a589c",

"Completed Regression": "Test/regressionCompleted/97616dab-48c8-4b1b-a129-56e0fa9a589c",

"Regression Pass Rate": "Test/regressionPassRate/97616dab-48c8-4b1b-a129-56e0fa9a589c",

"Validated Upgrade Paths": "Test/upgradePathsValidated/97616dab-48c8-4b1b-a129-56e0fa9a589c",

"Meantime to Repair": "Test/meanTimeToRepair/97616dab-48c8-4b1b-a129-56e0fa9a589c"

# REST APIs for KPI assessment

## requirementsMappingToTestCasesRate()

**Name**: requirementsMappingToTestCasesRate

**Description**: calculate Requirements-Testcases Mapping.

**Language**: sql

**Parameters**: requirementID

**URL**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/requirementsMappingToTestCasesRate/{requirementID}

**Example**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/requirementsMappingToTestCasesRate/9fe0d9e7-92dc-4b4c-9906-314c49c0bd64

**Code**:

|  |
| --- |
| /\*  requirementsMappingToTestCasesRate  Search Path:  DesignRequirement >> containsFeatures >> Feature >> testedByTestPlan    Calculation: Coverage of test plan to features  Note: A DesignRequirement has many Features; A Feature is tested by many testPlans;    \*/  begin  let features = SELECT identifier FROM ( SELECT EXPAND( OUT("ContainsFeatures") ) FROM DesignRequirement WHERE identifier=$requirementID)  let testPlans = SELECT outV().identifier AS feature, inV().identifier AS testPlan FROM TestedByTestPlan WHERE outV().identifier in $features GROUP BY outV().identifier  if($features.size() == 0) { return 0; }  return $testPlans.size() \* 1.0 / $features.size();  /\*  TestCase:  // DesignRequirement1 0.5  9fe0d9e7-92dc-4b4c-9906-314c49c0bd64  // DesignRequirement2 0  ac33f8a5-ccd5-4669-9556-5a3870093590  \*/ |

## testCaseExecutedRate()

**Name**: testCaseExecutedRate

**Description**: calculate Executed Nodal Testcases.

**Language**: sql

**Parameters**: projectID

**URL**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/testCaseExecutedRate/{projectID}

**Example**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/testCaseExecutedRate/71e1820c-5db9-4429-965c-b7164a00e265/

**Code**:

|  |
| --- |
| /\*  testCaseExecutedRate  Search Path:  Project << belongsToProject << DesignRequirement >> testedByTestPlan >> TestPlan >> usesTestCase >> TestCase << runsTestCase << TestExecutionRecord >> Calculate testcases with at least one TestExectionRecord  Calculation: # of TestPlan.usesTestCases / # of TestExecutionRecord.runsTestCase  Note: Different testPlans may share a testCase, so also make sure the testExecutionRecord reports on the correct testPlan.    \*/  begin  let testPlans = SELECT identifier FROM ( SELECT EXPAND( OUT("TestedByTestPlan") ) FROM DesignRequirement WHERE identifier in (SELECT identifier FROM ( SELECT EXPAND( IN("BelongsToProject") ) FROM Project WHERE identifier=$projectID)))  let testCases = SELECT identifier FROM ( SELECT EXPAND( OUT('UsesTestCases') ) FROM TestPlan WHERE identifier in $testPlans)  let executionRecordsRunsTestCase = SELECT identifier, out("RunsTestCase").identifier AS testCase FROM ( SELECT EXPAND( IN("RunsTestCase") ) FROM TestCase WHERE identifier in $testCases)  let executionRecordsReportsTestPlan = SELECT identifier FROM ( SELECT EXPAND( IN("ReportsOnTestPlan") ) FROM TestPlan WHERE identifier in $testPlans) /\* Bo: Also gets TestResult since they share the same EDGE "ReportsOnTestPlan" \*/  let intersectionExecutionRecords = SELECT \* FROM $executionRecordsRunsTestCase where identifier in $executionRecordsReportsTestPlan GROUP BY testCase  if($testCases.size() == 0) { return 0; }  return $intersectionExecutionRecords.size() \* 1.0 / $testCases.size()  /\*  TestCase:  // Project1: 6 out of 7  71e1820c-5db9-4429-965c-b7164a00e265  \*/ |

## 

## testCasePassRate()

**Name**: testCasePassRate

**Description**: calculate Nodal Testcases Pass Rate.

**Language**: sql

**Parameters**: projectID

**URL**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/testCasePassRate/{projectID}

**Example**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/testCasePassRate/71e1820c-5db9-4429-965c-b7164a00e265/

**Code**:

|  |
| --- |
| /\*  testCasePassRate  Search Path:  Project << belongsToProject << DesignRequirement >> testedByTestPlan >> TestPlan >> usesTestCase >> TestCase << reportsOnTestCase << TestResult << Calculate passed test result rate  Calculation: # of "Pass" result with MOST RECENT timestamp / total # of TestCases  Note: total # of TestCases = Pass + No Pass + No Executed  \*/  begin  let designRequirements = SELECT identifier FROM ( SELECT EXPAND( IN("BelongsToProject") ) FROM Project WHERE identifier=$projectID)  let testPlans = SELECT identifier FROM ( SELECT EXPAND( OUT("TestedByTestPlan") ) FROM DesignRequirement WHERE identifier in $designRequirements)  let testCases = SELECT identifier FROM ( SELECT EXPAND( OUT('UsesTestCases') ) FROM TestPlan WHERE identifier in $testPlans)  let testResult = SELECT outV().title AS testResult, outV().modified AS modified, outV().status AS status, inV().identifier AS testCase FROM ReportsOnTestCase WHERE inV().identifier in $testCases  let latestTimeStamp = SELECT testCase, MAX(modified) AS modified FROM $testResult GROUP BY testCase  let passResults = SELECT testCase, result, modified FROM $testResult WHERE status="Pass" AND modified in (SELECT modified FROM $latestTimeStamp)  if($testCases.size() == 0) { return 0; }  return $passResults.size() \* 1.0 / $testCases.size()  /\*  TestCase:  // Project1: 2 out of 7  71e1820c-5db9-4429-965c-b7164a00e265  \*/ |

## troubleTicketsFromOlderReleases()

**Name**: troubleTicketsFromOlderReleases

**Description**: calculate Carry-over Trouble Ticket Resolution.

**Language**: javascript

**Parameters**: projectID

**URL**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/troubleTicketsFromOlderReleases/{projectID}

**Example**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/troubleTicketsFromOlderReleases/a0da7296-042b-4a50-aa81-2a1271f5020b/

**Code**:

|  |
| --- |
| /\*  troubleTicketsFromOlderReleases  Search Path:  Project >> productVersion >> productVersion (1 smaller than the referenced) << reportsOnProject << Defect << status is not closed or solved    Note: return the # of unsolved defects  \*/  // Bo: A simple versionNumber converter  function olderReleaseNumber(currentReleaseNumber) {  var currVer = currentReleaseNumber.split(".");  currVer[0] = (parseInt(currVer[0]) - 1).toString();  return currVer.join(".");  }  var mydb = orient.getGraph();  try{  var currVersion = mydb.command('sql', 'SELECT title, productVersion FROM Project WHERE identifier="' + projectID + '"');  var olderVersion = mydb.command('sql', 'SELECT identifier FROM Project WHERE title="' + currVersion[0].getProperty("title") + '" and productVersion="' + olderReleaseNumber((String)(currVersion[0].getProperty('productVersion'))) + '"');  if (olderVersion.length == 0) {return 1.0;}  var defects = mydb.command('sql', 'SELECT identifier, title, status FROM ( SELECT EXPAND( IN("ReportsOnProject") ) FROM Project WHERE identifier="' + olderVersion[0].getProperty('identifier') + '")');  var unsolvedDefects = mydb.command('sql', 'SELECT identifier, title, status FROM ( SELECT EXPAND( IN("ReportsOnProject") ) FROM Project WHERE identifier="' + olderVersion[0].getProperty('identifier') + '") WHERE status NOT IN ["Closed", "Solved", "closed", "solved"]');  } catch(err) {  return err;  }  if (typeof defects === "undefined" || typeof unsolvedDefects === "undefined") {return 1.0;}  return (defects.length - unsolvedDefects.length) \* 1.0 / defects.length;  // TestCase:  // 1.0 No older release  // 71e1820c-5db9-4429-965c-b7164a00e265  // 2.0 1 unsolved defects out of 2  // a0da7296-042b-4a50-aa81-2a1271f5020b  // 3.0 2 unsolved defects out of 2  // 1bed9bb8-569e-455b-b20e-bbad74fe91a5 |

## progressMetricsForRiskPrediction()

**Name**: progressMetricsForRiskPrediction

**Description**: calculate 5 metrics for risk Prediction

\* FCR (feature completion rate)

\* CD (number of code defects)

\* ISS (number of reported issues at the feature level)

\* ISRR (issue removal rate)

\* DRR (defect removal rate)

**Language**: javascript

**Parameters**: projectID

**URL**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/progressMetricsForRiskPrediction/{projectID}

**Example**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/QualityManagement/progressMetricsForRiskPrediction/71e1820c-5db9-4429-965c-b7164a00e265/

**Code**:

|  |
| --- |
| /\* five metrics are create to measure the progress of the project.  \*  \* FCR (feature completion rate)  \* CD (number of code defects)  \* ISS (number of reported issues at the feature level)  \* ISRR (issue removal rate)  \* DRR (defect removal rate)  \*  \*/  //projectID = "71e1820c-5db9-4429-965c-b7164a00e265"  if(!projectID){  return {  "msg": "error"  }  }  var mydb = orient.getGraph();  var FCR = 0;  var CD = 0;  var ISS = 0;  var ISRR = 0;  var DRR = 0;  try{  //var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  //var dr = mydb.command('sql', 'SELECT title FROM DesignRequirement LET $project=OUT("BelongsToProject") WHERE $project.title="Project1"');  var features = mydb.command('sql', 'SELECT title,status,created FROM ( SELECT EXPAND( OUT("ContainsFeatures") ) FROM DesignRequirement LET $project=OUT("BelongsToProject")) GROUP BY title ORDER BY created ASC');  var codeDefects = mydb.command('sql', 'SELECT title,status,created,type FROM Defect LET $project=OUT("reportsOnProject") WHERE $project.identifier="' + projectID + '" AND type="Code" GROUP BY title ORDER BY created ASC');  var featureLevelIssues = mydb.command('sql', 'SELECT title,status,created,type FROM Defect LET $project=OUT("reportsOnProject") WHERE $project.identifier="' + projectID + '" AND type="Functional" GROUP BY title ORDER BY created ASC');    //var passedRegressionTestCases = mydb.command('sql', 'SELECT title FROM ( SELECT EXPAND( OUT("RunsTestCase") ) FROM TestExecutionRecord LET $testPlan=OUT("ReportsOnTestPlan") WHERE $testPlan.testPlanType="regression" AND $testPlan.OUT("TestForProject").identifier="' + projectID + '" AND OUT("Result").status="pass") GROUP BY title');  var completedFeatures = 0;  for(var i in features){  var feature = features[i];  if(feature.getProperties().get("status") == "Completed"){  completedFeatures ++;  }  }  FCR = features.length == 0 ? 0 : completedFeatures/features.length;  // FCR = features.length;  // FCR = dr.length;  CD = codeDefects.length;  ISS = featureLevelIssues.length;    var removedCodeDefects = 0;  for(var i in codeDefects){  var codeDefect = codeDefects[i];  if(codeDefect.getProperties().get("status") == "Closed"){  removedCodeDefects ++;  }  }    DRR = CD == 0 ? 0 : removedCodeDefects/CD;    var removedFeatureIssues = 0;  for(var i in featureLevelIssues){  var issue = featureLevelIssues[i];  if(issue.getProperties().get("status") == "Closed"){  removedFeatureIssues ++;  }  }    ISRR = ISS == 0 ? 0 : removedFeatureIssues/ISS;  } catch(err) {  return err;  }  //return projectID;  //return executedRegressionTestCases;  //return passedRegressionTestCases.length/executedRegressionTestCases.length ;  return {  //dr: dr.length,  FCR: FCR,  CD: CD,  ISS: ISS,  DRR: DRR,  ISRR: ISRR  }; |

## featureDocumentationComplete()

**Name**: featureDocumentationComplete

**Description**: calculate feature documentation completion rate.

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/featureDocumentationComplete/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/featureDocumentationComplete/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  var featureDocumentationCompletionRate = 0;  try{  //var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  var documentationItems = mydb.command('sql', 'SELECT title,OUT("TestedByTestPlan").title AS testPlanTitle,OUT("RelatedFeatureDesign").title AS featureDesignTitle,OUT("RelatedFeatureSpec").title AS featureSpecificationTitle FROM Feature WHERE IN("ContainsFeatures").IN("CurrentDesignRequirement").identifier = "'+projectID+'"');  //results = defects;  var n = 0;    for(var i in documentationItems){  var testPlanTitles = documentationItems[i].getProperties().get('testPlanTitle');  var featureDesignTitles = documentationItems[i].getProperties().get('featureDesignTitle');  var featureSpecificationTitles = documentationItems[i].getProperties().get('featureSpecificationTitle');    if(testPlanTitles.size() > 0 && featureDesignTitles.size() > 0 && featureSpecificationTitles.size() > 0){  n++;  }    }  featureDocumentationCompletionRate = n/documentationItems.length;  //featureDocumentationCompletionRate = testPlanTitles.size();  //results = MTTR;  } catch(err) {  return err;  }  //return projectID;  return featureDocumentationCompletionRate;  //return featureDocumentationCompletionRate;  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

## meanTimeToRepair()

**Name**: meanTimeToRepair

**Description**: calculate the mean time to repair the sytem

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/meanTimeToRepair/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/meanTimeToRepair/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  var MTTR = 0; //implemented in weeks.  //var oneAndHalfWeek = 907200;  //var TTRRate = 0;  try{  //var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  var defects = mydb.command('sql', 'SELECT created,fixedDate FROM ( SELECT EXPAND( OUT("RelatedDefect") ) FROM Feature WHERE IN("ContainsFeatures").IN("CurrentDesignRequirement").identifier = "'+projectID+'")');  //results = defects;  var tttr = 0; //implemented in weeks.  //var m = 0; //defects repaired within 1.5 weeks.  var n = 0; //total number of defects.  for(var i in defects){  var created = defects[i].getProperties().get('created');  var fixedDate = defects[i].getProperties().get('fixedDate');    if(created == ""){  continue;  }    var createdDateTime = new Date(created);  var fixedDateTime = Date.now();  if(fixedDate != ""){  fixedDateTime = new Date(fixedDate);  }  tttr += (fixedDateTime.getTime() - createdDateTime.getTime())/1000/60/60/24/7;  //if(tttr <= oneAndHalfWeek) {  //m++;  //}  n++;  }  MTTR = tttr/n;  //TTRRate = m/n;  //results = MTTR;  } catch(err) {  return err;  }  //return projectID;  //return regressionTestCases.length;  return MTTR;  //return TTRRate;  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

## preliminaryCapacityAndMemoryAnalysis()

**Name**: preliminaryCapacityAndMemoryAnalysis

**Description**: analyse preliminary capacity and memory

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/preliminaryCapacityAndMemoryAnalysis/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/preliminaryCapacityAndMemoryAnalysis/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  var analysisRate = 0;  try{  //var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  var capacityAndMemoryAnalysisTestCaseResults = mydb.command('sql', 'SELECT title,status FROM ( SELECT EXPAND( OUT("Result") ) FROM TestExecutionRecord LET $testPlan=OUT("ReportsOnTestPlan") WHERE $testPlan.subject IN ["capacity", "memory"] AND $testPlan.OUT("TestForProject").identifier="' + projectID + '") GROUP BY title');  var numberOfPasses = 0;  for(var i in capacityAndMemoryAnalysisTestCaseResults){  var result = capacityAndMemoryAnalysisTestCaseResults[i];  if(result.getProperties().get("status") == "Pass"){  numberOfPasses ++;  }  }  analysisRate = numberOfPasses/capacityAndMemoryAnalysisTestCaseResults.length;  } catch(err) {  return err;  }  //return projectID;  //return regressionTestCases.length;  return analysisRate;  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

## regressionCompleted()

**Name**: regressionCompleted

**Description**: calculate regression completion rate

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/regressionCompleted/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/regressionCompleted/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  try{  var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  var executedRegressionTestCases = mydb.command('sql', 'SELECT title FROM ( SELECT EXPAND( OUT("RunsTestCase") ) FROM TestExecutionRecord LET $testPlan=OUT("ReportsOnTestPlan") WHERE $testPlan.testPlanType="regression" AND $testPlan.OUT("TestForProject").identifier="' + projectID + '") GROUP BY title');  } catch(err) {  return err;  }  //return projectID;  //return regressionTestCases.length;  return regressionTestCases.length / executedRegressionTestCases.length;  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

## regressionPassRate()

**Name**: regressionPassRate

**Description**: calculate regression pass rate.

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/regressionPassRate/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/regressionPassRate/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  var passRate = 0;  try{  //var regressionTestCases = mydb.command('sql','SELECT title FROM ( SELECT EXPAND( OUT("UsesTestCases") ) FROM TestPlan WHERE testPlanType = "regression" AND OUT("TestForProject").identifier="' + projectID + '")');  var executedRegressionTestCases = mydb.command('sql', 'SELECT title,status,created FROM ( SELECT EXPAND( OUT("Result") ) FROM TestExecutionRecord LET $testPlan=OUT("ReportsOnTestPlan") WHERE $testPlan.testPlanType="regression" AND $testPlan.OUT("TestForProject").identifier="' + projectID + '") GROUP BY title ORDER BY created ASC');  //var passedRegressionTestCases = mydb.command('sql', 'SELECT title FROM ( SELECT EXPAND( OUT("RunsTestCase") ) FROM TestExecutionRecord LET $testPlan=OUT("ReportsOnTestPlan") WHERE $testPlan.testPlanType="regression" AND $testPlan.OUT("TestForProject").identifier="' + projectID + '" AND OUT("Result").status="pass") GROUP BY title');  var numberOfPasses = 0;  for(var i in executedRegressionTestCases){  var result = executedRegressionTestCases[i];  if(result.getProperties().get("status") == "Pass"){  numberOfPasses ++;  }  }  passRate = numberOfPasses/executedRegressionTestCases.length;  } catch(err) {  return err;  }  //return projectID;  //return executedRegressionTestCases;  //return passedRegressionTestCases.length/executedRegressionTestCases.length ;  return passRate;  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

## upgradePathsValidated()

**Name**: upgradePathsValidated

**Description**: calculate upgrade paths validation rate

**Language**: javascript

**Parameters**: projectID

**URL**: http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/upgradePathsValidated/{projectID}

**Example**:

http://ec2-52-53-163-218.us-west-1.compute.amazonaws.com:2486/function/Test/upgradePathsValidated/97616dab-48c8-4b1b-a129-56e0fa9a589c

**Code**:

|  |
| --- |
| var mydb = orient.getGraph();  var validatedRate = 0;  try{  var currentProductVersion = mydb.command('sql', 'SELECT productVersion FROM Project WHERE identifier="' + projectID + '"');  var currentProductVersionNumber = currentProductVersion[0].getProperties().get('productVersion');  var previousReleases = mydb.command('sql','SELECT releaseNumber FROM releaseRecord WHERE releaseNumber < "'+currentProductVersionNumber+'"');  var validatedNumber = 0;  for(var i in previousReleases){  var previousReleaseNumber = previousReleases[i].getProperties().get('releaseNumber');  var validationTestPlans = mydb.command('sql', 'SELECT title FROM ( SELECT EXPAND( IN("ValidateUpgradePath") ) FROM UpgradeConfiguration WHERE OUT("FromReleaseNumber").releaseNumber="'+previousReleaseNumber+'" AND OUT("ToReleaseNumber").releaseNumber="'+currentProductVersionNumber+'") GROUP BY title');  if(validationTestPlans.length != 0){  validatedNumber++;  }  }  validatedRate = validatedNumber/previousReleases.length;  } catch(err) {  return err;  }  //return projectID;  //return regressionTestCases.length;  //var result1 = result['result'][0];  return validatedRate;  //response.send(500, "Error on creating new user", "text/plain", result.toString());  //97616dab-48c8-4b1b-a129-56e0fa9a589c |

# 

# 

# 

# Triggers

## onCreated()

Name: onCreated

Description: Used for assigning uuid and timestamp when a new record is either created or modified.

Language: javascript

Code:

|  |
| --- |
| // OnCreated  // 1. Assign a uuid to identifier  // 2. Update 'created' and 'modified' fields  // Bo: Temporarily used for generating the uuid  function uuid(seed){  var uuid = 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {  var r = (seed + Math.random()\*16)%16 | 0;  dt = Math.floor(seed/16);  return (c=='x' ? r :(r&0x3|0x8)).toString(16);  });  return uuid;  }  var now = new Date().getTime();  doc.field("created", now);  doc.field("modified", now);  doc.field("identifier", uuid(now));  doc.save(); |

## onUpdated()

Name: onUpdated

Description: Used for re-assigning timestamp when a record is modified.

Language: javascript

Code:

|  |
| --- |
| // onUpdated  // 1. Update 'modified' field  doc.field("modified", new Date().getTime());  doc.save(); |