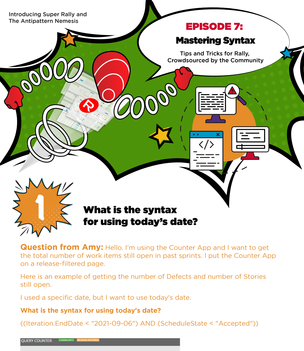
Introducing Super Rally and The Antipattern Nemesis

EPISODE 7:
Mastering Syntax

Tips and Tricks for Rally, Crowdsourced by the Community



1 What is the syntax for using today's date?

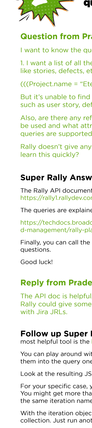
Question from Amy: Hello. I'm using the Counter App and I want to get the total number of work items still open in past sprints. I put the Counter App on a release-filtered page.

Here is an example of getting the number of Defects and number of Stories still open.

I used a specific date, but I want to use today's date.

What is the syntax for using today's date?

{{iteration.EndDate < "2021-09-06"} AND {ScheduleState < "Accepted"}}



Super Rally Answer:

You should be able to use the keyword "Today" like this:

{Iteration.EndDate < "Today"}

2 How to know which Rally query objects to use?

Question from Pradeep:

I want to know the query for the below scenario to be queried in Custom Apps.

1.I want a list of all the work products which are there in a particular iteration, like stories, defects, etc. I could write below:

{((Project.name = "Eternala") and (Iteration.name = "ET_21.2.2))....

But it's unable to find any class objects to use to query for work product type, such as user story, defect, etc.

Also, are there any reference materials to read to understand which objects can be used and what attributes to query to get data for custom apps where queries are supported?

Rally doesn't give any suggestions while writing these queries. How do you learn this quickly?

Super Rally Answer: Hi!

The Rally API documentation is one of the finest in the industry. You can start here: https://rally1.rallydev.com/slm/doc/webservice

The queries are explained in the doc,

https://techdocs.broadcom.com/us/en/ca-enterprise-software/agile-development-and-management/rally-platform-ca-agile-central/rally.html

Finally, you can call the support and get timely, accurate, and friendly answers to your questions.

Good luck!

Reply from Pradeep: Thank you for your reply!

The API doc is helpful, but not to the extent I'm looking for. It would be best if Rally could give some suggestions while writing the queries, like what happens with Jira JRLs.

Follow up Super Rally Answer: Besides the multiple query examples, the most helpful tool is the https://rally1.rallydev.com/slm/doc/webservice

You can play around with the sample queries from the links I posted earlier and put them into the query one by one and observe the resulting JSON REST URI.

Look at the resulting JSON carefully and you will figure it out in no time.

For your specific case, you may want to first query iteration by Name or Start Date. You might get more than one result, depending on the number of projects sharing the same iteration name.

With the iteration objects in hand, you can explore the Iteration Workproducts collection. Just run another 'curl' with this new URI and filter the results, as needed.

Good luck!

Update Super Rally Answer: it took me some time to collect queries I needed to use to get the data I needed. To test queries, you just need a web browser and an active Rally connection (log into Rally in a separate tab). Then, in the new tab, just copy and paste built URLs, starting from those below.

Here are some examples you can try:

Get opened Features with a specific Tag and from a specific project

https://rally1.rallydev.com/slm/webservice/v2.0/portfolioitem/feature?project=https://rally1.rallydev.com/slm/webservice/v2.0/project/{projectID}&fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500&query=((Tags.Name = {tag}) AND (State != Done) AND (State != Shipped))

Get all Rally projects

https://rally1.rallydev.com/slm/webservice/v2.0/project?fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500

Get Rally project by name

https://rally1.rallydev.com/slm/webservice/v2.0/project?fetch=true&query=(name contains {project})

Get Feature stories

https://rally1.rallydev.com/slm/webservice/v2.0/portfolioitem/feature/{featureID}/UserStories?fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500

Get Capability features

https://rally1.rallydev.com/slm/webservice/v2.0/portfolioitem/feature?fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500&query=(Parent.ObjectID = {CapabilityID})

Get Rally users (only these with Broadcom domain)

https://rally1.rallydev.com/slm/webservice/v2.0/UserStories?fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500&query=((UserName contains broadcom))

Get capabilities under certain project

https://rally1.rallydev.com/slm/webservice/v2.0/portfolioitem/capability?fetch=ObjectID,FormattedID,State,ScheduleState,Name,Owner,Project,Release,Iteration,PercentDoneByStoryCount,FirstName,LastName,EmailAddress,UserName&pagesize=500&project=https://rally1.rallydev.com/slm/webservice/v2.0/project/{projectID}&query=((State != Done)

The fetch parameters, if true, return all object properties. If you ask for a property which doesn't exist for a requested object, the property is simply returned back (but the query is still successful). This allows you to build universal fetch statements across most of the queries.

Have a great day!

Reply from Pradeep: Thanks for the reply. I will look at each link and gather these queries. I'm specifically looking out for filters to query work products, like stories, defects, etc.

Additional Super Rally Answer: It seems you've received two helpful replies, but I'll add another (hopefully more specific to your ask) regarding all work that is scheduled to an iteration.

There are two ways you can get this done:

1. You can query each artifact (work item) separately for a specific iteration or a set of iterations. That means you'll query for stories, then you'll query for defects, etc., where you'll apply a filter for the given iteration/iterations in that query. Then you can combine the results.

2. You can query the iteration object itself by the name/names that you need. Then, you'll have to look into the WorkProducts field, which has a combined list of all work items scheduled for this iteration.

You'll also need to remember that iterations are specific to projects. You may need to include a filter for project/projects if you'd like to limit this to a specific set of projects.

Choosing between the two ways would depend on your method of query and what is most comfortable to use. You could do it both ways, you'll have to implement a script. The first way is possible to get done using our Excel field add-in without a script. However, you'll need to export each work item type to a separate sheet and combine these sheets later (if you want).

If you'd like to provide more specific details, we can help further. I'll be happy to discuss this with you over the phone, if you'd like.

3 Why can't defects be split?

Question from Jason: I'm trying to understand why Defects are not set up like Sts, etc. that can be split at Iteration end if they weren't completed? We often find ourselves in this situation and our only choice is to move the Defect to the next Iteration, which negatively impacts our Velocity for both sprints. Can someone help me understand this? Are we using Rally wrong?

Super Rally Answer: Just my 2 cents =).

First let me say I can't answer your question, as I don't know why Rally chose not to provide this capability to the defect object.

If I had been the designer I might have made the same choice though. Right or wrong, I believe that defects are not user stories and should not be treated as such. If a defect is not completely resolved, then it is not considered done and should go back on the Product Backlog. The team gets no partial credit, but can bring the defect into the next Sprint to complete it (if it is still important to the Product Owner).

However, if the Product Owner sees true value in part of the resolution, they have the right to split it up. Ideally, that is something that should have been done when first identifying the defect or at the beginning of the Sprint, eliminating the need for a split in the first place.

As for affecting velocity, I can also say that defects should be sized in a way as to minimize effort thereby limiting the impact to velocity. This makes it harder at first, but enforces good behaviors which lead to better outcomes and increased velocity over the long term, in my experience.

Reply from Jason: Thanks for taking the time to reply.

IRL though, it's not uncommon to end an Iteration with work still in play. The work is fluid while the Iteration end date is just a marker. We try our best to anticipate how much effort all the work for the Iteration will take when planning the Iteration, but it's not an exact science and "stuff crops up".

It sounds like you have had to come up with a workaround for the current deficiency in Rally to properly accommodate for WIP at the end of the Iteration? I see no difference between a DE that the PO wanted worked and a US that the PO wanted worked. When planning the next Iteration, splitting incomplete US and DEs is not different. I can't imagine users being coached to put an incomplete US back on the Product Backlog simply because they didn't get to complete it through Acceptance.

Another way to look at it - I consider Defects to be of even higher priority than USes. The PO had previously identified a feature for the Product and it was important enough to get prioritized and worked so that it is now a part of the Product. If it is later found that this feature is not working as intended, then fixing it to initial expectations is an extension of the original work that had already been prioritized and worked. Why that correction is treated as one-off in Rally appears disjointed to me.

New Super Rally Answer: I'll be bold and suggest that yes, you may be using Rally "wrong". Or at least could use it better.

Firstly, if you're regularly splitting work at Iterations end, there's a problem for the team to address in Retrospective. It could be sizing, refinement, the team's capacity, etc. But splitting should be a short term problem that the team can mostly eradicate. When I hear a team say "we split work regularly," I urge a timeout or IP sprint to dive into RCA for this routine practice.

As far as the treatment of Defects like Stories, I would NOT. Stories are Stories. Defect records are Defects. Keep'em separate. This way, if the work to resolve a Defect exceeds the team's capacity for an Iteration, create two Stories as predecessor/successor (if you do that), or create a Feature with Stories as children to track the work. The Defect can be tied to the Stories or the Feature.

Another bonus to this approach: if you record the work in Stories, then someone decides to deprecate or defer the Defect, you still can take credit for the work in the Stories while the Defect record goes into the Backlog (or dust bin).

Does this make sense? ;)

Reply from Jason: We do 3 week iterations (that was a team decision, so not up for debate). Anticipating what all can be fit into an iteration is not realistic. Sizing a story or defect is not an exact science (the 'U' in CURSE). That said, I'm still not hearing why Defects shouldn't be able to be split the same as stories.

Work is work. If the work is not done at the end of the iteration, it doesn't really matter if it was work to introduce something new or work to do to fix something, it will still need to be continued on into the next sprint. The splitting feature allows you to create a parent to own both objects, so that could also apply for defects.

Super Rally Response: As an experienced software test/QA Analyst, I'd say that if I want to review work related to a bug report, I'm happy to look at Stories, Features, Tasks, whatever.

But, when I want to review a Bug Report, don't make me chase down multiple versions of that report. I want one Defect record to review the fix, the build version, when pushed through environments, validation results, etc.

I can't make it simpler than that.