

Taskwarrior - Feature # 359: Cross referencing task IDs

Status:	New	Priority:	Low
By:	John Florian	Category:	
Created:	12/21/2009	To:	David Patrick
Updated:	12/23/2009	Due date:	
Subject:	Cross referencing task IDs		
Description:	<p>It would be useful if it were possible to do something like:</p> <pre><pre> \$ task add Put out house fire. Created task 42 \$ task add Fix fire hydrant. Created task 43 \$ task annotate 42 See also #43. Annotated 42 with 'See also #43.' \$ task annotate 43 See also #42. Annotated 43 with 'See also #42.' </pre></pre> <p>The example is kind of lame because it suggests a dependency, but that's not really the intent. Obviously this can be done now, but if the IDs shift, the annotations will become incorrect. The gist of the feature would be that these somehow get adjusted behind the scenes. Present workarounds might involve a project or tags, but I can see cases where the projects are distinct. I suppose the annotation could suggest something like 'See tag foo'.</p> <p>No pressing need now, just thought the idea warranted recording for future consideration.</p>		

History

12/21/2009 03:31 PM - David Patrick

- Priority changed from Normal to Low

- Target version set to 2.2.x - Tanto

this is directly analogous to RedMine's Related Issue links (look up)

How about this ? if an annotation is a single numeric entry, and that entry matches an active task, then those tasks are related.

or even, following the redmine example above, annotations could consist of

ann:related to 78

ann:duplicates 78

ann:blocks 78

ann:precedes 78

What happens afterwards is the interesting question, as to use this new link, reporting will have to be enhanced, again, and much discussion will be needed of how related tasks should behave and be displayed.

12/23/2009 11:08 AM - Paul Beckingham

I love the idea of connected tasks. This feature keeps cropping up, and is highly desirable. There are problems though - while everyone agrees that task hierarchies, linkage, and dependencies are all things we want, no one has yet been able to put forth a sufficiently complete definition of what they are, what they do, and how they are used. We all want them, but only see them in a vague and ill-defined way. I would jump at the chance of implementing this, but not without a coherent and complete specification. That said:

Linking tasks would be straightforward to implement. For example, there could be a command like:

```
<pre>
```

\$ task 4 link 23

Task 4 "foo" linked to task 23 "bar".

</pre>

Task would resolve the "23" to the UUID of the task, and create some link using the UUID. When displayed, the UUID would be replaced by a numeric ID for clarity. There would need to be a corresponding unlink command.

But before implementing something like this, we need to do a lot of analysis, and answer some very hard questions. They don't look hard, but just try answering them in a way that is sufficiently detailed and complete. Here they are:

Cardinality: Is linkage a one-to-one or one-to-many connection?

Commutative: Is linkage a two-way connection? For example, if 4 is linked to 23, is 23 implicitly linked to 4?

Transitive: If 4 is linked to 23, and 23 is linked to 30, is 4 linked to 30?

Differentiation: Do we need linkage types (related, duplicates, blocks, precedes for example)? Or is just one type (link) all that we need? The desirability of multiple connection types is balanced by the complexity of having different types.

What actions taken against task 4 would now affect task 23, and how?

Does 23 show up in reports that would otherwise show only 4?

Does completing 4 imply 23 is completed?

Does completing a linked task break the link?

How is linking tasks any different from assigning tasks to a common project, or applying a common tag?

Then there is the old dependency paradox I like to bring out:

* If task 6 is dependent on task 7, and I complete 6, is 7 automatically completed, or is completing 6 prevented?

* If completing 6 does not automatically complete 7, then what exactly does "dependent" mean?

* If completing 6 before 7 is not rejected, then what exactly does "dependent" mean?

* If completing 6 before 7 is rejected, then is task truly representing reality, when it says "7 depends on 6, you can't do that"?

This is our killer feature. Solving all this and coming up with a good linkage plan would be a major and dramatic enhancement to task, and would satisfy the bulk of all the (big) feature requests ever made. We could identify critical path. We could have estimates and estimated completion. We could make a really good "next" algorithm without the need for heuristics. We could have reports with natural groupings. The list goes on...

Can we do it?

12/23/2009 11:51 AM - David Patrick

Paul Beckingham wrote:

> I love the idea of connected tasks. This feature keeps cropping up, and is highly desirable. There are problems though - while everyone agrees that task hierarchies, linkage, and dependencies are all things we want, no one has yet been able to put forth a sufficiently complete definition of what they are, what they do, and how they are used. We all want them, but only see them in a vague and ill-defined way. I would jump at the chance of implementing this, but not without a coherent and complete specification.

[snip]

>

> Then there is the old dependency paradox I like to bring out:

[snip]

>

> Can we do it?

We can and will!

In order for task to be truly relevant, and to reflect the actualities of task-doing, we will have to address linking and dependencies (aspects of the same thing) and it's going to be way hairy to reduce the lists, wants, wishes, use-cases, reconcile the project management dialects and best-practices, but we're not going to do it now, or even in the near-future.

Task 2.0.x +, with interactive ncurses interface, is the next thing up. It will quintuple the potential user-base overnight.

In 2.1.x, the fuller feature-set of what-we-know-now of task, will find it's place in the interactive interface.

These are foundational aspects for the future of the application, and contain lots of goodies, and will require lots of hard work and deep-thought. The dependency/linking subject is big enough to consume is all for months, just establishing the key specs, and if we are having that discussion with an interactive taskwarrior in place, we can focus on the nitty-gritty.

So let's keep the subject alive in the forums, and keep thinking and discussing how you think that should work, and let's get 2.0.x out the door.