COYOTECREEK™

DATA SHEET

REDMINE TO JIRA

By Matt Starr
Senior Systems Engineer

THE MANY BENEFITS OF MOVING TO JIRA

If you've been using Redmine for project management and issue tracking, you're undoubtedly aware that this tool has quite a few limitations. You've probably also heard that Atlassian's JIRA is the number one development tool for agile teams, and that it has a reputation of being able to improve a team's performance at every step of the development process.

Unfortunately, migrating from Redmine to JIRA is not a quick and simple task. Which raises the question: is it worth the bother to switch to JIRA? Based on our experience in the field, the answer is a resounding "yes!" Here are some of the reasons why:

- **Excellent performance** – Redmine has been known to be buggy and present issues when run on certain servers and operating systems. And the larger your organization grows, the slower Redmine gets. You'll see a serious degradation in the software's performance, and you won't be able to do anything about it. With Redmine you've got the product and that's what you get; there's not much you can to do improve performance as you grow.

- JIRA provides excellent performance without any of these problems. Atlassian has been strong on performance since day one, and continues to grow in this area. So not only is JIRA not buggy, there are modifications available that will enhance performance as your demands increase.

- **Easy integration with other apps** – JIRA easily integrates with other apps, whether they're from the Atlassian product suite or from other platforms. For example, JIRA provides great integration techniques for integration with JIRA Service Desk, Atlassian's Portfolio (another reporting tool), Bamboo, Hudson and many others. You simply don't get this with Redmine.

- **Clean user interface** – Redmine looks like you're dealing with something made in 1980. JIRA's layout and functionality is clean, up to date and easy to use.

- **Better workflows** – It's no secret that Redmine's workflows are terrible. Because they're not particularly customizable, all you can do is choose tasks from start to end. In contrast, with JIRA you can not only have your tasks from start to end, you can also flow in other actions and requirements for different types of transitions. In addition, JIRA is also very customizable on a per-project basis; Redmine is not.

- **In-depth reporting** – With Redmine you're very limited on the reports that you can pull for issues and projects. JIRA gives you in-depth reporting, charting and graphing, with the choice of running reports on one project, multiple projects or all projects.

- **Strong online help community** – Because Redmine's online help community is so limited, you're often left hanging in the dark, trying to figure things out by yourself. With Atlassian's very strong community backing and extensive issues documentation, resolving questions with JIRA is quite a bit easier.

- **Robust add-on marketplace** – Redmine doesn't even have a marketplace per se, and a very limited choice of add-ons is available. Atlassian has a strong add-on marketplace, with many options available that can extend JIRA's usefulness to your organization.

PREPARING TO MIGRATE FROM REDMINE TO JIRA

If you have not done a Redmine migration before, you'll find that it can be a very taxing process. Why? Because there are so many areas to evaluate—including wiki content, build information and source code repository data—and so many things that must be mapped out ahead of time.

JIRA comes bundled with a Redmine importer that helps make the migration a little less painful. But before you dive in and start your migration, you really should evaluate a few areas of concern before proceeding. What we will be focusing on here is simply (or not so simply!) migrating Redmine issues and data into JIRA. I recommend that you review this entire whitepaper before you perform the import.

User Mapping

One major hurdle you should handle before any migration is the user base inside Redmine. If your JIRA instance is not a fresh and empty install or if the Active Directory (AD) connection you plan to use for JIRA is not the same as the one used for Redmine, you will need to take extra care reviewing the users. If both JIRA and Redmine have the identical user base and email addresses, you are in luck! But, in most cases, you will need to do a little user mapping. For example, if you are using different AD connections for JIRA and Redmine, you will need to plan a method to bring all users and their associated data over successfully.

It's safe practice to evaluate both sets of users from JIRA and Redmine. You should determine if all users need to be imported as is. If you are just importing all users as is into JIRA, you can skip past the user mapping section. But if you are going to sync users from Redmine to their respective JIRA user account and either the user names and/or email addresses of the users differ between systems, you will need to work on first setting up a 1-to-1 mapping of the users. This can be a little time consuming, but very necessary. After you have your mapping set up, you can use that information to update the users as needed. It is best to get the users in sync before the import is performed. This will require some scripts to update all users in Redmine and their associated data.

Some points to note regarding the user import:

- **Users who have interacted in Redmine** will be created as active accounts in JIRA.
- **Users who have not interacted in Redmine** will be placed into a group called "Redmine-import-unused-users" and deactivated.

- **Passwords are not imported.** Imported Redmine users will need to have their passwords emailed to them the first time they login to JIRA.
- **If you are using External User Management**, users cannot be imported. Instead you will receive a list of new users that will need to be created and added prior to the import.
- **If you exceed your user license**, the import will stop and you will be given a list of users that cannot be created.

Statuses and Priorities

Statuses and Priorities will need to be mapped during the import. So now is a good time to evaluate the statuses and priorities used in Redmine and compare them to the ones in JIRA.

Since these are global elements, you may want to consider mapping to what exists in JIRA before simply adding all of the statuses and priorities used in Redmine. If you have several statuses and priorities in Redmine that do not exist in JIRA, you must determine if it is worthwhile adding more to JIRA, or if you can map the Redmine statuses and priorities to what already exists in JIRA without cluttering up your system with unnecessary items. Be sure to document the mapping before the import. You will need this information for the workflow migration!

Workflows

Workflows are another area of concern that must be evaluated prior to the import. You will need to get all workflows used in Redmine over to JIRA.

Using your status mapping data you can safely create all the workflows used by Redmine. Ideally you will rebuild your workflows in the Atlassian paradigm. If that is the case, you can just add your statuses to the workflow but not worry too much about the steps of the workflow, since these steps will change after the import. Just make sure you use all of the proper statuses. If you have just one workflow with all the proper statuses, it should be safe for you to use that for all Redmine projects to be imported.

Custom Fields

Another time-consuming task is the custom field evaluation. You will need to set up the custom fields used in Redmine in your JIRA instance. This can be a bit redundant, but it is best to have all fields set up prior to the import. This allows you to breeze through the mapping of fields during the import.

Tracker/Issue Type

Redmine "Trackers" are the equivalent of JIRA "Issue Types." Check to see if you will be able to map to what exists in JIRA. If not, you will need to create the missing tracker/issue types.

Schemes and Projects

Now is a good time to start setting up the empty projects into which you will be importing all this data. To keep things clean, it is best to set up Redmine-specific schemes to be used during the import process. You can set these up like you would for any other JIRA project. After all of your schemes are created, go through the list of projects you plan to import from Redmine and start setting up the empty projects with their proper schemes. Do not add components or versions, because doing so would turn the empty projects into non-empty projects—and the importer cannot be run for non-empty projects. Let the importer handle adding the components and versions.

Links

Since Redmine supports hierarchical issues, you will need to duplicate this using links in JIRA. You may need to configure a custom issue link to replicate this.

MIGRATING FROM REDMINE TO JIRA

At this point you should be in a safe place to begin the import. Hopefully you are doing this in your Development Environment first and making plenty of backups. In fact, now is a good time to make one more backup before the import.

Before you can start the import you must first make sure you are an admin in both JIRA and Redmine. Then enable the REST service in Redmine, or you will not be able to connect for the import.

Once you are properly prepared, the migration itself is straightforward. Here's what you need to do:

1. **Get connected** – Connect to Redmine through JIRA.

2. **Map the projects** – You will be prompted to map the projects from Redmine to JIRA. You will have three options: Create project, Map To, or Do Not Import. Since you already set up your empty projects, you can easily map from Redmine to JIRA.

3. **Map the custom fields** – Since all of your custom fields are set up at this point, just find the matching fields in the drop down and map them out. There have been bugs reported when the importer creates the custom field versus when you manually create them prior to import. Save yourself the struggle and create them first!

4. **Map other fields** – After you complete the custom field mapping, you will be prompted to choose to map fields such as priorities. Trackers and statuses always get mapped. During this step you will also need to select the Redmine workflow that you already configured in JIRA.

5. **Map the values** – This can be a lot of clicking and selecting, but if you did all your prep work, it should go smoothly. This is where you get to refer to all your mapping documentation and map out priorities (if selected in the previous step), issue types, statuses and such.

6. **Map the links** – The last step before the import runs is mapping out the links which you should have already configured. Once this is mapped out successfully, you can start the import.

7. **Do the import** – If you are not importing every project that is in Redmine, you may see errors during the import due to links between an imported and not imported project being broken.

8. **Save the configuration file** – After the import runs, SAVE YOUR CONFIGURATION FILE! This comes in handy so you don't have to go through all the mapping again.

9. **Review your logs** – Check for any errors or issues.

10. **Repeat as necessary** – It may take a few sweeps to get everything imported as expected. But since you are in your Development Environment with plenty of backups, this should not be an issue.

CONCLUSION

While migrating from Redmine to JIRA is a significant project that requires careful planning and preparation, the project is well worth the effort. Once the migration is complete you'll have a project management and bug tracking tool that's superior in all ways—performance, reporting, user interface, workflows, extendibility and online help community.

Of course, if you need help making this migration happen, give us a call. As Atlassian Platinum Experts, Coyote Creek has extensive experience with the entire Atlassian product line.

ABOUT COYOTE CREEK

Founded in 1998 by a team of former corporate IT professionals, Silicon Valley-based Coyote Creek understands the issues and complexities that are part of large-scale and high-growth IT environments. Coyote Creek brings deep expertise to help you manage technology and project risk, inspiring confidence and creating peace of mind.

The hallmarks of our work are practicality, flexibility, integrity and a commitment to your success that goes well beyond anything you can write in a contract. From project work such as Atlassian JIRA support consulting and Microsoft consulting to managed services and staffing services, Coyote Creek can meet your business process automation and IT needs.