BLOG

A sneak peek behind the scenes of our MS Teams integration

The combination of Teams and Conductor is great: you can have unstructured conversations, idea generation, and collaboration within Teams, and then formalize and add structure to the work within Conductor by creating tasks, assigning them to the right person, and setting goals and timelines.

Al Torreno August 25, 2020

As a business, we're very closely aligned with our partners at Microsoft. Our collaborative work management solution, Conductor, is built on .NET, deployed on Azure, and has tight integrations with the entire Microsoft 365 suite. For Sensei Labs, going all-in on Microsoft has led us to efficiency and cost benefits, as well as opening new opportunities for growth via the Microsoft Partner Network.

It was a no-brainer to continue this momentum when we decided to build our Microsoft Teams integration for Conductor. Teams is the fastest growing workplace communications platform. Along with many of our customers, we've also made the switch to Teams for our internal communications. The combination of Teams and Conductor is great: you can have unstructured conversations, idea generation, and collaboration within Teams, and then formalize and add structure to the work within Conductor by creating tasks, assigning them to the right person, and setting goals and timelines.

Where do we start in creating a Teams integration?

There are lots of possible integration options between Teams and third-party applications:

- **App tabs:** Add your application as a tab within Teams. Allows you to pass information from Teams (e.g. current user, current channel) into your application. If you render your app without any of your typical chrome (i.e. navigation, footer) you can create a seamless UI between Teams and your app.
- **Messaging extensions:** Add extra functionality for users when they are posting messages in Teams. For example, starting an ad-hoc poll
- **Cards:** Richly formatted snippets of content that can be added into chats
- **Bots:** Interacting with your app via conversations

We believe that reducing the friction of opening yet another web application in a browser will lead to increased engagement. For project team members, they can manage tasks, update project schedules, or add new data values without leaving Teams. For the executive sponsor, they can take a quick peek at the Conductor dashboard for an up-to-the-minute initiative summary, and then immediately jump into a Teams chat.

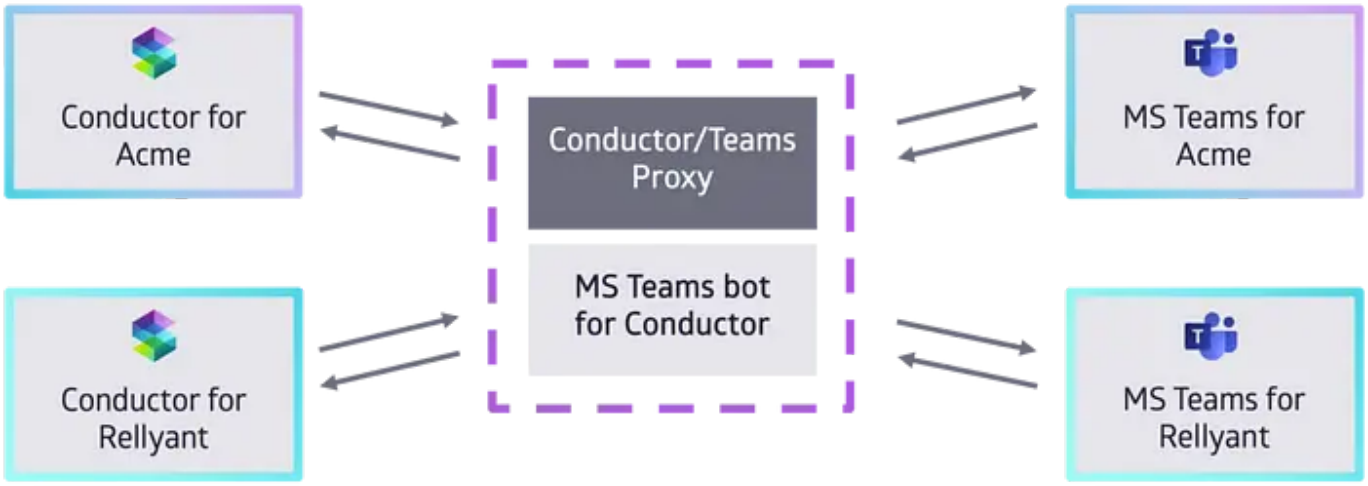
The more data we have about an initiative in Conductor, the more Conductor can do to help inform decision making, track project health, and ultimately lead to better outcomes. Our dashboard widgets instantly identify trends, areas of focus, and progress towards initiative goals. This is the key differentiator when Conductor is your single source of truth.

In the end, we created an MVP integration that included:

- Ability to view Conductor task boards (and other pages within Conductor) inside a Teams tab
- Ability to complete initial configuration via Bot conversation
- Microsoft Teams as a Conductor notification channel

As a major integration into a relatively new third-party application, a lot of questions were raised along the way. I wanted to share a few of those and how we resolved them!

How can we connect single tenant instances? Our application is single tenant. That means for each customer we set up a separate instance of Conductor. As we add more customers, and all of them are adding our app in Teams, how do we connect those Teams instances to all our Conductor instances? The answer: a proxy application.



The proxy directs traffic between our customers' Teams instances and their Conductor instances. It will allow us in the future to connect a Teams instance to multiple Conductor instances, which will be important for our partners that leverage Conductor as part of their service offering. It also aids us in the other direction: as we push more content from Conductor into Teams, the proxy maps Conductor instances and end users to the Teams counterparts.

How can we accelerate our development?

We had an advantage that accelerated our development: access to Microsoft technical contacts through our partner status with Microsoft. We often ran into moments where we were trying to accomplish something atypical. Or moments when there was unexpected behavior and existing documentation didn't help us narrow down what we might be doing wrong. It was invaluable to have the support from Microsoft to point us in the right direction. Our ISV support contact Ben Cheng directed us to useful docs and informed us of serendipitously timed MS Teams educational workshops. Nikola Metulev with the Teams product team helped us understand Teams development best practices and walked us through debugging our most stubborn bugs.

What is best practice for authenticating into our application within Teams?

Here's an example of the support we received. Some of our bot interactions require the Teams user to be authenticated into Conductor, since we needed a mapping between Teams user and Conductor user in order to execute some commands. Accessing Conductor pages within a Teams tab also requires authentication to verify the account and to maintain access controls and permissioning. What is the correct way to handle this in Teams? We started down one path and hit technical roadblocks. During our regular regroup, we described the problem, and was immediately pointed to documentation that clearly described a better method. We tried it out the next day and got things up and running!

What gaps exist in our internal processes?

We had another major lesson learned: involving our DevOps team closely throughout this process. At the onset, we took for granted how much infrastructure was needed to facilitate Conductor development: version control system and processes, non-production environments, deployment pipelines, logging, caching, and the list goes on and on. All of this has been in place forever for the main Conductor application, but none of this was in place for our new proxy web application. It left us scrambling a few times. Thankfully our Platform team is one of the best and quickly spun up environments as needed. Next time we will keep them closely engaged from the beginning to ensure we have all the necessary environments and internal tooling!

How do we retain and spread our learnings?

A good reminder for all of us in software development: throw everything into the wiki! Based on our learnings from this project, we've created tons of new documentation, including all of our initial Teams API R&D, a guide for setting up developer local environments, deployment guides, links to helpful Microsoft documentation, troubleshooting guides, and end user guides.

Final thoughts

In the end, we were able to move from a proposal to an MVP in roughly 12 weeks and our app is now approved and listed in the Teams marketplace! We're very excited about how this new Teams integration brings Conductor into to our customers' Teams workflow and makes it more convenient for them to create a seamless collaborative work experience!