Shared Notification Service

This page is about the design of a notification system that could be shared between TTAT and Workspaces

Workspaces Notification Service

The current (WS sprint 5-6) version of our notification service is quite simple. There is a REST API for creating notification templates, which consist of a name and a Mustache template. When you want to send a notification, you post to a REST API with the name of the notification you want to send and JSON arguments. One of the JSON arguments is `receiver_email` and that becomes the recipient. The JSON arguments in total are the context for the Mustache template render. The result of that render is emailed to the recipient.

This system has a significant benefit, in that notification templates can change while the application is online, without waiting for redeployments. This comes with the extra cost that one must be sure that the templates are going to render without error.

Future directions

The following are ideas for directions that the notification service should probably evolve towards.

Notification previews

If we allow system managers to modify the templates, we are going to need to provide them with some way of knowing if the template is valid and going to produce the expected result when the expected notifications arrive. A preview mechanism, perhaps with the addition of a JSON schema, could work for this.

Authentication

The notification service is currently protected by being hidden behind other services in a Docker configuration. If we want to open this up to sharing, we will want to arrange for *application authorization*, so that random users are not able to send REST calls to cause notifications to be sent. Authentication and authorization is a larger and separate issue (see Authentication, Authorization and Allocation (A3) System) but having well-defined and in-common concepts of principal and whatever else might go with it, would be a useful thing to make a formal part of an API here. In other words, "I am *some application*, please send this notification on behalf of *some user*."

Non-email protocols

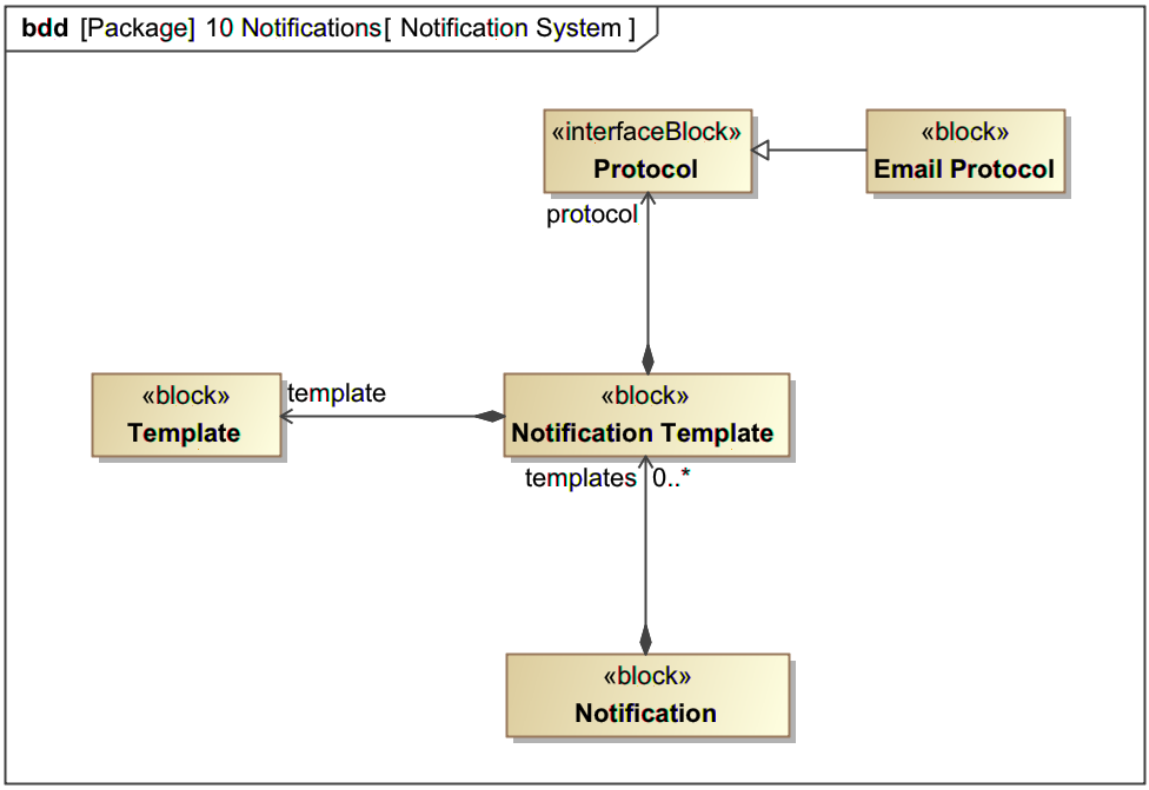
For internal stuff, Slack notifications are useful. Other possible protocols for notifications include SMS, or server-sent events for HTTP. The current system assumes it is sending email, but that fact is mostly hidden from the notification sender. Maintaining that separation will make it easy to start sending notification over other protocols, as configured in the notification service.

Keeping this configuration in the notification service and outside the applications that need to send notifications opens up the possibility of changing how notifications are sent at runtime, without waiting for an application deployment.

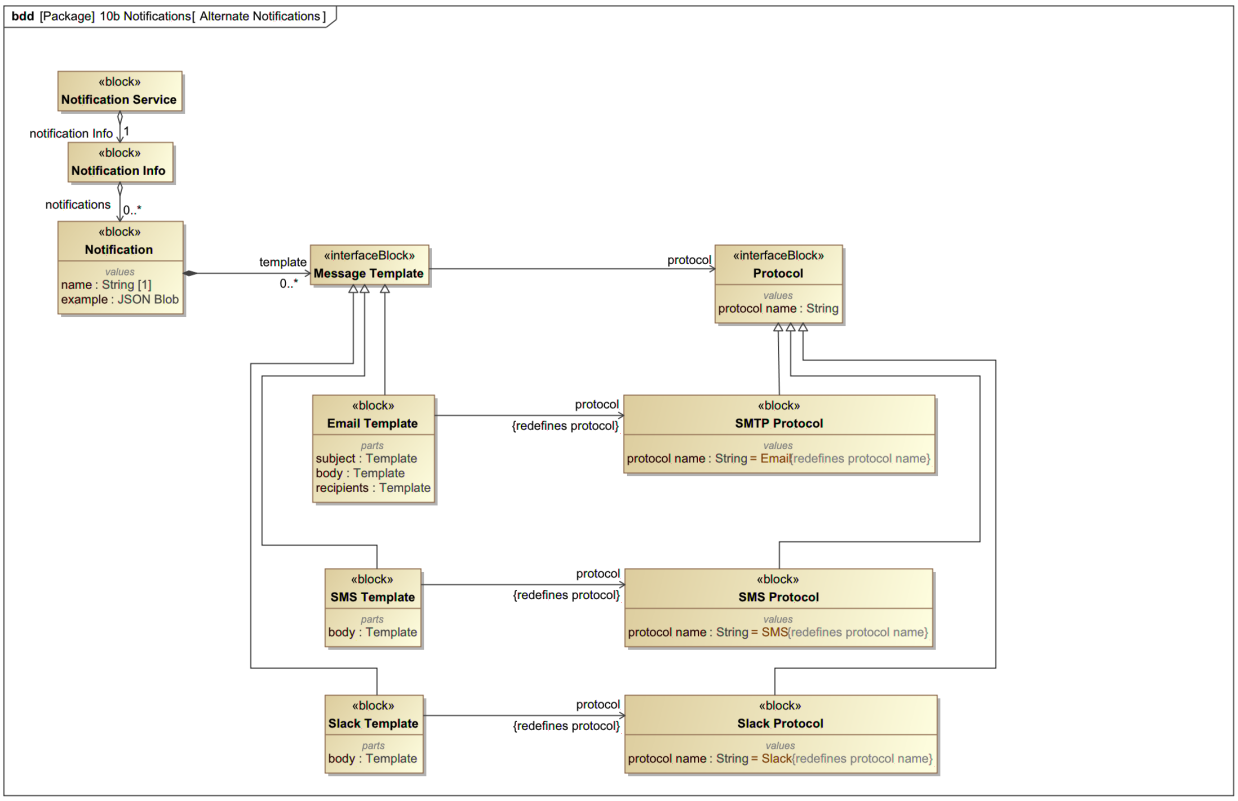
Multiple notifications

It would not be hard to introduce a 1:Many relationship between notifications and templates/messages. This could enable sending multiple notifications in response to one notification event. In workspaces, an example of this might be sending one note to the QA team for a capability reaching that step while also sending a notification to the user that the processing has entered the quality assurance state. The workspaces application wouldn't need to intentionally send two notifications, but two would get sent out by the notification system.

Implementation Notes



A more rigorous conception:



Design questions

Consider the three protocols SMS, Email and Slack. Of these, one is simply body text (SMS); the other two have additional message attributes. Email has arbitrary headers, but some of those headers are essentially mandatory (From, Subject). Slack has a mandatory text field but can actually accept a complex JSON object with many "blocks" and create a high level of interactivity. On top of this, two of these protocols need a recipient to be explicitly identified (SMS and Email) and the other does not.

How do we set up a UI that knows something about these protocols, on top of a rendering system that does not? Or do we need to make the templates protocol-specific? Or do something simple like, have each notification template be a map of string Mustache template, and the UI just shows N templates per notification template?