Conflict Handling

Certainly! Let's dive deeper into solution numbers 4 (Conditional Logic and Flags) and 6 (Conflict-Aware Design) for managing conflicts between apps in Frappe/ERPNext environments.

4. Conditional Logic and Flags

This approach involves enhancing the logic within the hooks and possibly adding flag fields to help manage and resolve conflicts.

Enhancing Logic in Hooks

- **Checks Before Changes**: In each app's hook, include checks to see if another app has already modified the data. This can be done by checking the current value against expected values or using timestamp comparisons to see which change is more recent.
- **Sequential Logic**: Design the hooks' logic to work sequentially rather than independently. For example, if App1 sets a field to a certain value, App2's hook can first check this value before deciding on its action.

Using Flags

- **Flag Fields**: Add a custom field to your shared Doctype, such as `last_modified_by_app`. Each app, when making a change, sets this field to its own identifier (e.g., 'App1' or 'App2').
- **Logic Based on Flags**: Each app checks this flag field before making changes. If the flag indicates that the other app has recently made changes, the current app can decide whether to proceed with its own changes, merge them, or skip the update.

6. Conflict-Aware Design

This approach focuses on designing the applications and their interactions in a way that minimizes the potential for conflicts.

Design for Non-Interference

- **Separate Fields for Each App**: Instead of having both apps modify the same field, consider adding separate fields for each app. This way, each app has its own 'space' to work within, reducing direct conflicts.
- **Different Mechanisms for Similar Outcomes**: If both apps aim to achieve similar outcomes (like changing a status), consider implementing different mechanisms or workflows for each app. For example, one app could change a status directly, while another triggers a workflow that leads to the status change.

Loose Coupling

- **Independent Modules**: Design the apps to be as independent as possible. This reduces the chances of one app's changes affecting the other.
- **Event-Driven Architecture**: Implement an event-driven architecture where apps subscribe to and react to specific events rather than directly modifying shared resources. This way, one app's actions can emit events that the other app listens to, allowing for more controlled interactions.

Implementing These Approaches in Frappe

In Frappe, these approaches would involve customizing the Doctypes and their associated Python controller files. You would modify the hooks (`hooks.py`) and the methods that these hooks call to include the necessary logic for managing conflicts. This might include writing additional Python methods, creating new fields in the database schema, and perhaps adding custom scripts to handle complex interactions.

Final Thoughts

Both these approaches require careful planning and a deep understanding of how your applications interact within the Frappe framework. They also emphasize the importance of testing: you'll need to rigorously test your applications to ensure that the conflict resolution logic works as expected and that it doesn't introduce new issues.

Hooks

Hooks allow you to "hook" into functionality and events of core parts of the Frappe Framework. This page documents all of the hooks provided by the framework.

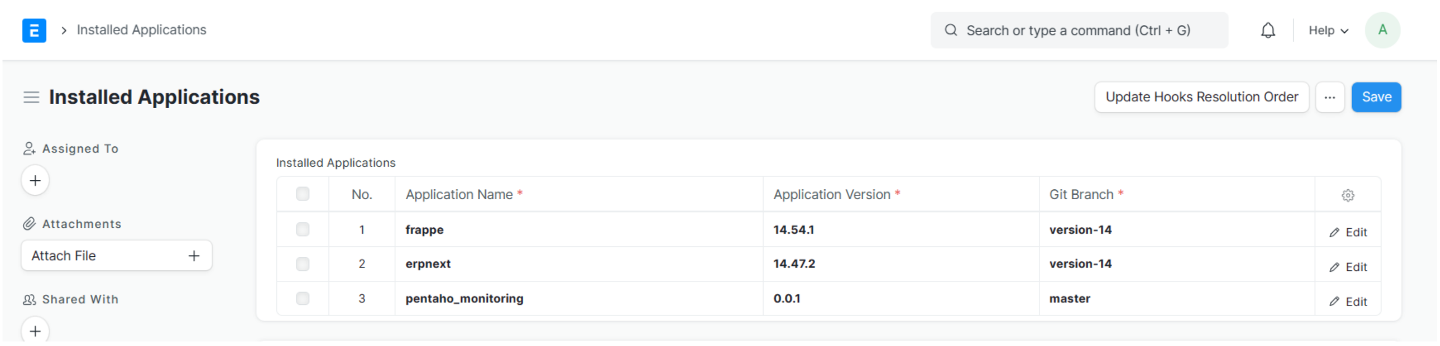
https://frappeframework.com/docs/user/en/python-api/hooks#how-are-conflicting-hooks-resolved-

How are conflicting hooks resolved?

Hooks are resolved using "last writer wins" strategy. Last installed app on site will have highest priority over others.

- When the hook overrides existing behaviour like overriding a class then only overrides from last app will work.
- When the hook extends behaviour then extensions will be applied in order of installation on the site.

> If you need to change this order you can do so by going to "Installed Applications" page and clicking on "Update Hooks Resolution Order"



Installed Applications

No.	Application Name *	Application Version *	Git Branch *	
1	frappe	14.54.1	version-14	Edit
2	erpnext	14.47.2	version-14	Edit
3	pentaho_monitoring	0.0.1	master	Edit