

Actions and Links

Summary:

In Frappe's framework, "Actions and Links" (also known as Connections) are mechanisms that enhance user interaction with documents. They provide ways to execute specific actions on a DocType or navigate between related documents.

Key Insights:

1. **Actions:** Actions in a DocType result in button creation on the DocType View. There are two primary types of actions:
 - **Server Action:** Triggers a whitelisted server action.
 - **Route:** Redirects to a specified route.

To implement an action in a custom app, a Python function decorated with

`frappe.whitelist` is required. This function will be executed when the corresponding action button is clicked.

To call an Action in you own app, you will need a python function decorated with `frappe.whitelist` :

```
import frappe

@frappe.whitelist()
def execute_function(*args,**kwargs):
    """
    This function will be executed when the Execute Action Button will be clicked
    """
    print('Hello World')
    # The data is transmitted via keyword argument
    print(kwargs)
```

This code should go somewhere inside you app, typically in a file like `apps/my_app/my_app/api.py`

And then, configure the correspondent Action path :

Actions ^						
Actions						
<input type="checkbox"/>	No.	Label *	Action Type *	Action / Route *	Group	
<input type="checkbox"/>	1	Execute	Server Action	my_app.api.execute_function		Edit

2. **Connections (Linked Documents):** The Connections section on the DocType dashboard aids navigation by showing which document types are related to the current DocType. This feature allows users to quickly identify and create new related documents. These links can also support internal links, i.e., links to DocType in child tables.
3. **Configuration:**
 - **Via Script:** Connections for a DocType can be configured by creating a `get_data()` function inside `<doctype>_dashboard.py`. This function defines the connections, their groups, and the fieldnames to be searched for external links.
 - **Customization:** Both DocType Actions and Links can be extended and customized via the "Customize Form".

More Explanation

Actions in Frappe's DocTypes can be visualized as the spell cards in a Yu-Gi-Oh duel. They are tools that allow you to perform specific operations or maneuvers on the data stored within a DocType. **When you invoke an action, you command the**

system to execute a particular operation, altering the state of your data or triggering specific processes.

Examples: Save, Submit, Cancel.

In Frappe, **Links** establish connections between different DocTypes, allowing them to reference and interact with one another.

For instance, consider two DocTypes: `Duelist` and `Deck`. A Duelist may have a Link field connecting to a Deck, indicating which set of cards (data) they are using in battles (operations). This Link allows data from the `Deck` DocType to be referenced and utilized within the `Duelist` DocType.

Example:

```
# DocType: Dark Magician's Insight
class DarkMagiciansInsight(Document):
    def on_submit(self):
        # Action: When the spell is activated (DocType is submitted)
        self.peek_into_deck() # Invoke the power of the spell (a method)

    def peek_into_deck(self):
        # Logic to select a card from the deck (query data from another DocType)
        selected_card = frappe.get_doc("Deck", {"monster": "Dark Magician"})
        # Use the selected card in some operation...
```

In this example:

- **Action:** `on_submit` is an action that triggers when the "Dark Magician's Insight" spell card (DocType) is activated (submitted). It invokes the method `peek_into_deck`, which contains the logic (operations) of the spell.
- **Link:** If we had a Link field connecting to another DocType (e.g., `Deck`), we could directly interact with and utilize data from that linked DocType.