It sounds like you've hit a common point of confusion when working with Keycloak's authorization management. The way permissions are represented and manipulated internally through the Java API differs significantly from the REST API representation. Let's break down how to correctly add a group to an existing scope-based permission using the Java API.
You're right, directly casting the result of policyStore.findByName to ScopePermissionRepresentation won't work for updates. The Policy object you get back is the internal representation, and modifications need to be done through its methods or by retrieving the specific permission representation for updates.
Here's a refined approach to add another group to an existing scope permission:

Java

```
import org.keycloak.authorization.admin.representation.ScopePermissionRepresentation;
import org.keycloak.authorization.model.Policy;
import org.keycloak.authorization.store.PolicyStore;
import org.keycloak.models.GroupModel;
import org.keycloak.models.RealmModel;
import org.keycloak.representations.idm.authorization.ResourceRepresentation;
import org.keycloak.representations.idm.authorization.ScopeRepresentation;
import org.keycloak.representations.idm.authorization.UserPolicyRepresentation;

import java.util.Set;
import java.util.stream.Collectors;

public void addAnotherGroupToExistingPermission(String permissionName, GroupModel
groupToAdd, PolicyStore policyStore, RealmModel realm,
org.keycloak.authorization.model.ResourceServer resourceServer) {
    Policy permission = policyStore.findByName(resourceServer, permissionName);

    if (permission == null) {
        System.out.println("Permission not found: " + permissionName);
        return;
    }

    if (!(permission instanceof org.keycloak.authorization.model.ScopePermission)) {
        System.out.println("Permission is not a scope-based permission: " + permissionName);
        return;
    }

    org.keycloak.authorization.model.ScopePermission scopePermission =
(org.keycloak.authorization.model.ScopePermission) permission;
```

```
    // Get the current resource IDs associated with the permission
    Set<String> resourceIds = scopePermission.getResources().stream()
            .map(org.keycloak.authorization.model.Resource::getId)
            .collect(Collectors.toSet());

    // Add the new group ID to the set of resource IDs
    resourceIds.add(groupToAdd.getId());

    // Create a new ScopePermissionRepresentation for the update
    ScopePermissionRepresentation updatedPermissionRep = new
ScopePermissionRepresentation();
    updatedPermissionRep.setName(scopePermission.getName());
    updatedPermissionRep.setDescription(scopePermission.getDescription());

updatedPermissionRep.setPolicies(scopePermission.getPolicies().stream().map(Policy::getId).
collect(Collectors.toSet()));

updatedPermissionRep.setScopes(scopePermission.getScopes().stream().map(org.keycloak.a
uthorization.model.Scope::getName).collect(Collectors.toSet()));
    updatedPermissionRep.setResourceType("Groups"); // Important: Set the correct resource
type
    updatedPermissionRep.setResources(resourceIds); // Set the updated set of resource IDs

    // Update the permission in the policy store
    policyStore.update(resourceServer, permission.getId(), updatedPermissionRep);

    System.out.println("Successfully added group '" + groupToAdd.getName() + "' to permission
'" + permissionName + "'.");
}
```

Explanation of the changes and why your initial approach didn't work:

1. **Casting to ScopePermission:** We first check if the retrieved Policy is actually a ScopePermission. This is crucial because permissions can be of different types (e.g., user-based, role-based).
2. **Retrieving Existing Resource IDs:** Instead of trying to create a ResourceWrapper manually, we retrieve the IDs of the resources currently associated with the permission. The scopePermission.getResources() method returns a set of org.keycloak.authorization.model.Resource objects. We extract their IDs.
3. **Adding the New Group ID:** We simply add the groupToAdd.getId() to the resourceIds set.
4. **Creating a ScopePermissionRepresentation for Update:** To update the permission, you need to create a ScopePermissionRepresentation that reflects the desired state. This is the same type of object you used for creation.

5. **Populating the ScopePermissionRepresentation:**
    - We copy the existing name, description, and associated policy IDs and scope names from the original scopePermission.
    - **Crucially, we set the resourceType to "Groups"**. This tells Keycloak that the resources refer to group IDs.
    - We set the resources field to the updated resourceIds set, which now includes the new group's ID.
6. **Using policyStore.update():** We use the policyStore.update() method, providing the resourceServer, the ID of the permission to update (permission.getId()), and the updatedPermissionRep.

Why ResourceWrapper was problematic:

- ResourceWrapper is an internal representation and isn't directly meant for updating permissions in this way.
- The addResource method on the internal Policy object likely expects specific internal Resource objects, not just wrappers with IDs and names. Keycloak manages the mapping between permissions and resources internally.

Why your "fictive example" doesn't directly translate:

The policyStore.findByName method returns the internal Policy object, not the ScopePermissionRepresentation. The internal object has methods like addPolicy, removePolicy, addScope, etc., but for modifying the resources (in this case, groups), you need to work with the representation used for creation and updates.

In summary, the correct approach involves:

1. Retrieving the existing ScopePermission object.
2. Getting the IDs of the currently associated resources (groups).
3. Adding the ID of the new group.
4. Creating a new ScopePermissionRepresentation with the updated set of resource IDs and the correct resourceType.
5. Using policyStore.update() with the permission ID and the updated representation.

This approach directly manipulates the permission data in a way that Keycloak understands for updates. Remember to handle potential null values for the permission and ensure the permission you're trying to update is indeed a scope-based permission.

Sources

1. https://github.com/keycloak/keycloak subject to license (Apache - 2.0)