# Common Code Patterns

The following examples demonstrate common patterns for implementing Workflow Assignment Rules.

#### Pattern 1: Assign Single Identity to Activity

Simplest pattern for assigning a specific Identity to an activity:

```
// Target = "Activity"
// this = Activity item
Innovator inn = this.getInnovator();
string activityId = this.getID();

// Define the Identity to assign (can be user Identity or group Identity)
string identityId = "IDENTITY_ID_HERE";

// Create assignment
Item assignment = inn.newItem("Activity Assignment", "add");
assignment.setProperty("source_id", activityId);
assignment.setProperty("related_id", identityId);
assignment.setProperty("voting_weight", "100");
assignment = assignment.apply();

if (assignment.isError())
{
return inn.newError("Error creating assignment: " +
assignment.getErrorDetail());
}

return this;
```

#### Pattern 2: Assign Manager to All Workflow Activities

Assigns the manager from the context item to all non-start/non-end activities:

```
// Target = "Workflow"
// this = Workflow Process item
Innovator inn = this.getInnovator();
string workflowId = this.getID();

// Resolve context and get manager
string contextTypeId = this.getAttribute("context_type_id");
string contextItemId = this.getAttribute("context_item_id");

Item contextItem = inn.newItem();
contextItem.setAction("get");
contextItem.setAttribute("typeId", contextTypeId);
contextItem.setID(contextItemId);
contextItem.setAttribute("select", "managed_by_id");
contextItem = contextItem.apply();

if (contextItem == null || contextItem.isError())
{
return inn.newError("Failed to load context item");
}

string managerId = contextItem.getProperty("managed_by_id");
if (string.IsNullOrEmpty(managerId))
{
return inn.newResult("No manager found; no assignments created");
}

// Query workflow activities
Item wf = inn.newItem("Workflow Process", "get");
wf.setID(workflowId);
wf.setAttribute("select", "id");
Item wfPA = wf.createRelationship("Workflow Process Activity", "get");
wfPA.setAttribute("select", "related_id");
wf = wf.apply();
```

```
if (wf == null || wf.isError())
{
return inn.newError("Failed to load workflow activities");
}
```

```
// Get activities
var activities = wf.getItemsByXPath("//Item[@type='Activity']");
if (activities == null || activities.getItemCount() == 0)
{
return inn.newResult("No activities found; no assignments created");
}
```

```
// Create assignments for non-automatic activities
int created = 0;
for (int i = 0; i < activities.getItemCount(); i++)
{
var activity = activities.getItemByIndex(i);
```

```
// Skip automatic activities
if (activity.getProperty("is_auto") == "1")
{
continue;
}
```

```
string activityId = activity.getID();
```

```
Item assignment = inn.newItem("Activity Assignment", "add");
assignment.setProperty("source_id", activityId);
assignment.setProperty("related_id", managerId);
assignment.setProperty("voting_weight", "100");
assignment = assignment.apply();
```

```
if (assignment.isError())
{
return inn.newError("Failed to create assignment for activity " +
```

```
activityId);
 }
 created++;
 }
 return inn.newResult("Created " + created + " assignments");
```

**Pattern 3: Assign Part Owner Based on Affected Items**

Queries affected items and assigns Part Owners to the activity:

```
// Target = "Activity"
// this = Activity item
 Innovator inn = this.getInnovator();
 string activityId = this.getID();

// Resolve context
 string contextTypeId = this.getAttribute("context_type_id");
 string contextItemId = this.getAttribute("context_item_id");

 if (string.IsNullOrEmpty(contextTypeId) ||
string.IsNullOrEmpty(contextItemId))
 {
 return inn.newError("Unable to determine context item");
 }

// Query affected items through Change Order
 Item changeOrder = inn.newItem();
 changeOrder.setAction("get");
 changeOrder.setAttribute("typeId", contextTypeId);
 changeOrder.setID(contextItemId);
 changeOrder.setAttribute("select", "id");

// Get affected items
 Item affectedRel =
```

```
changeOrder.createRelationship("ucm ChangeOrder AffectedItem", "get");
 affectedRel.setAttribute("select", "related_id(owned_by_id)");

 changeOrder = changeOrder.apply();

 if (changeOrder == null || changeOrder.isError())
 {
 return inn.newError("Failed to load affected items");
 }

// Collect unique part owners
 var owners = new HashSet<string>();
 var affectedItemRels =
changeOrder.getRelationships("ucm_ChangeOrder_AffectedItem");

 if (affectedItemRels != null && affectedItemRels.getItemCount() > 0)
 {
 for (int i = 0; i < affectedItemRels.getItemCount(); i++)
 {
 var rel = affectedItemRels.getItemByIndex(i);
 var affItem = rel.getRelatedItem();

 if (affItem != null && !affItem.isError())
 {
 string ownerId = affItem.getProperty("owned_by_id");
 if (!string.IsNullOrEmpty(ownerId))
 {
 owners.Add(ownerId);
 }
 }
 }
 }

 if (owners.Count == 0)
 {
```

```
return inn.newResult("No part owners found; no assignments created");
}

// Create assignment for each unique owner
int created = 0;
foreach (string ownerId in owners)
{
Item assignment = inn.newItem("Activity Assignment", "add");
assignment.setProperty("source_id", activityId);
assignment.setProperty("related_id", ownerId);
assignment.setProperty("voting_weight", "100");
assignment = assignment.apply();

if (assignment.isError())
{
return inn.newError("Failed to create assignment for owner " + ownerId);
}

created++;
}

return inn.newResult("Created " + created + " assignments for part owners");
```

**Pattern 4: Query Existing Assignments Before Modification**

Checks for existing assignments to avoid duplicates or to modify assignment properties:

```
// Target = "Activity"
// this = Activity item
Innovator inn = this.getInnovator();
string activityId = this.getID();

// Query existing assignments
Item existingAssignments = inn.newItem("Activity Assignment", "get");
existingAssignments.setProperty("source_id", activityId);
```

```
existingAssignments.setAttribute("select", "related_id,voting_weight");
existingAssignments = existingAssignments.apply();

// Check if specific identity is already assigned
string targetIdentityId = "IDENTITY_ID";
bool alreadyAssigned = false;

if (!existingAssignments.isError() && existingAssignments.getItemCount()
> 0)
{
for (int i = 0; i < existingAssignments.getItemCount(); i++)
{
var assignment = existingAssignments.getItemByIndex(i);
if (assignment.getProperty("related_id") == targetIdentityId)
{
alreadyAssigned = true;
break;
}
}
}

if (alreadyAssigned)
{
return inn.newResult("Identity already assigned; no action taken");
}

// Create new assignment
Item newAssignment = inn.newItem("Activity Assignment", "add");
newAssignment.setProperty("source_id", activityId);
newAssignment.setProperty("related_id", targetIdentityId);
newAssignment.setProperty("voting_weight", "100");
newAssignment = newAssignment.apply();

if (newAssignment.isError())
{
```

```
 return inn.newError("Error creating assignment: " +
newAssignment.getErrorDetail());
 }
 return this;
```