

Workflow

Fundamentals of IdentityIQ Implementation
IdentityIQ

Overview

Workflow

- What is a workflow?
- Workflows in IdentityIQ
- Key workflow concepts
- Workflows in detail
 - Workflow case
 - Workflow variables
 - Workflow steps
 - Approvals
- Workflow troubleshooting

What is a Workflow?

- A sequence of steps with transition logic between them
- Each step performs one action
 - For example, execution of code, or user-interaction
- Defined with Business Process Editor or by creating XML definitions
- Referred to in the SailPoint UI as “Business Processes”



LCM and Workflow

- Workflows are intimately tied to LCM
 - Requests
 - Entitlement/Role
 - Account Requests
 - Passwords
 - Management of users passwords
 - Reactions to Native Password changes
 - Identity
 - Create/Edit
 - Identity Change Events
 - Mover
 - Joiner
 - Leaver
 - Other Attribute Changes

Lifecycle Manager Configuration

Lifecycle Actions Business Processes Additional Options

Action		Business Process
Request Access	?	LCM Provisioning
Manage Accounts	?	LCM Provisioning
Manage Passwords	?	LCM Manage Passwords
Edit Identity	?	LCM Create and Update
Create Identity	?	LCM Create and Update

Lifecycle Events

Lifecycle Events

Filter by Lifecycle Event Name



Add New Lifecycle Event

Name	Type	Attribute Name	Owner
Department Transfer	Attribute Change	Department	The
Joiner	Create		
Leaver	Attribute Change	Inactive	

Workflows throughout IdentityIQ

- Workflows are also used for...
 - Role Creation
 - Identity Update
 - Identity Refresh
 - Identity Correlation
 - Sunrise/Sunset Role Assignment
 - Sunrise/Sunset Role Activation
 - Policy Violations

Configure IdentityIQ Settings

Mail Settings Work Items **Identities** Roles Password Policy **Miscellaneous**

Identity Risk

? Number of Bands

Label	Range	Indicator
<input type="text" value="risk_band_low"/>	<input type="text" value="0 - 333"/>	<input type="radio"/>
<input type="text" value="risk_band_med"/>	<input type="text" value="334 - 667"/>	<input type="radio"/>
<input type="text" value="risk_band_high"/>	<input type="text" value="668 - 1000"/>	<input type="radio"/>

Identity Attributes

Number of searchable attributes

Identity Snapshots

Snapshot frequency in days(2 equals every second day)

Business Processes

Identity update ?

Identity refresh ?

Identity Correlation ?

Edit Policy

Entitlement SOD Policy

Name

Owner

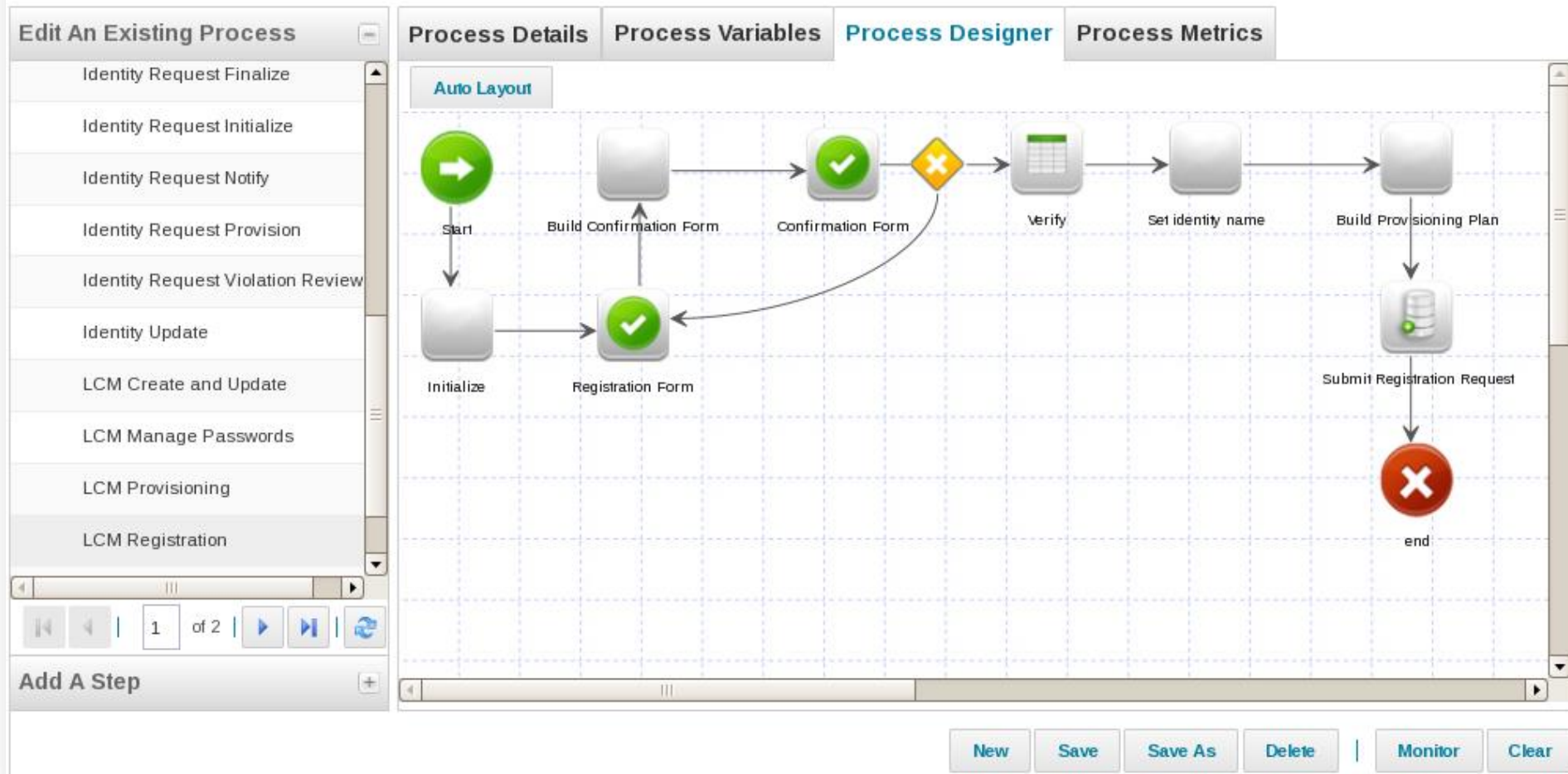
Violation business process

What can Workflows do?

- Check for Policy Violations
- Gather information from users
- Get Approvals/Generate Workitems
- Send Emails
- Audit
- Adjust Provisioning Plan
- Call Subprocesses (child workflows)
- Run other Workflows
- Really anything you can code in Java/Beanshell

What Do Workflows Look Like in the GUI?

Business Process Editor



What Do Workflows Look Like in the XML?

```
<Workflow name='My Workflow'>
```

```
  <Variable name='identity' input='true'/>
  <Variable name='newRoles' input='true'/>
  <Variable initializer='string:spadmin'
input='true' name='approveOwner'/>
```

```
  <Step name='start'>
    <Transition to='Approve'/>
  </Step>
```

Step

```
  <Step name='Approve'>
```

```
    <Approval owner='ref:approveOwner'
      send='identity,newRoles'
      renderer='myApproval.xhtml'/>
```

```
    <Transition to='Approved'
when='approved'/>
    <Transition to='Rejected'/>
```

```
  </Step>
```

Step

```
    <Step name='Approved'
      action='call:saveObject'>
      <Arg name='object' value='ref:identity'/>
      <Transition to='end'/>
    </Step>
```

Step

```
    <Step name='Rejected' action='call:audit'>
      <Arg name='source' value='ref:launcher'/>
      <Arg name='action' value='rejected'/>
      <Arg name='target'
value='script:identity.getName()'/>
      <Arg name='string1' value='failure'/>
      <Transition to='end'/>
    </Step>
```

Step

```
  <Step name='end'/>
```

Step

```
</Workflow>
```

Workflow

Key Workflow Concepts

- **Workflow**
 - Definition of a workflow process
 - **Variable**
 - Named data holder within a workflow process
 - **Step**
 - Named unit of work within the process
 - **Transition**
 - Rules for moving between steps
 - **Approval**
 - Special type of step that allows user interaction
- Static elements of the workflow
-
- **WorkflowCase**
 - Instance of a running workflow process
 - **WorkItem**
 - Object representing the state of one user interaction
- Dynamic, runtime objects

WorkflowCase

- In traditional workflow terminology a “case” is an instance of a running workflow process
- There can be many active cases for a single workflow process
- In IdentityIQ a case is represented with a WorkflowCase object that is removed when the workflow completes
- A WorkflowCase contains a complete copy of the Workflow object that defines the process
 - This makes it immune to changes in the process definition

Key Workflow Concept – Scriptlets

- Flexible way to specify values using a single line of text
- Scriptlets make the XML smaller and easier to read
- 5 types of scriptlets
 - Supply a literal string
 - Reference a workflow variable
 - Run a BeanShell script to return a value
 - Call a rule to return a value
 - Call a method to return a value
- Specified in the UI or directly in the XML
- Used extensively throughout our workflows

Key Workflow Concept – Scriptlets

- In UI, you select scriptlet type and specify source

Initial Value

If you would like this variable to be initialized with an initial value, you can configure how this will be calculated using the options below. Each of the options will determine how the value will be produced.

String ☐ Reference ☐ Script ☒ Rule ☐ Call Method ☐

Source: `event.getcause();`

- In XML, scriptlets begin with type, followed by a colon, followed by the “source”

string:spadmin	supplies a literal string
ref:approver	references a workflow variable
script:identity.getManager().getName()	runs a piece of BeanShell
rule:My Approver Rule	calls a rule
call:buildOwnerApproval	calls a method in the registered WorkflowHandler class or Library

Key Workflow Concepts – XML

```
<Workflow name='My Workflow'>
```

Variables

```
<Variable name='identity' input='true'/>
```

```
<Variable name='newRoles' input='true'/>
```

```
<Variable initializer='string:spadmin'  
input='true' name='approvalOwner'/>
```

Scriptlet - string

```
<Step name='start'>
```

```
<Transition to='Approve'/>
```

```
</Step>
```

Step

```
<Step name='Approve'>
```

Approval

```
<Approval owner='ref:approvalOwner'  
send='identity,newRoles'  
renderer='myApproval.xhtml'/>
```

Transition

```
<Transition to='Approved'  
when='approved'/>
```

```
<Transition to='Rejected'/>
```

```
</Step>
```

Step

```
<Step name='Approved'
```

```
action='call:saveObject'
```

```
<Arg name='object' value='ref:identity'/>
```

```
<Transition to='end'/>
```

```
</Step>
```

Scriptlet - call

Scriptlet - ref

```
<Step name='Rejected' action=call:'audit'>
```

```
<Arg name='source' value='ref:launcher'/>
```

```
<Arg name='action' value='rejected'/>
```

```
<Arg name='target'  
value='script:identity.getName()'/>
```

```
<Arg name='string1' value='failure'/>
```

```
<Transition to='end'/>
```

```
</Step>
```

Step

```
<Step name='end'/>
```

Step

```
</Workflow>
```

Workflow

Workflow Details

Workflow Variables

```
<Variable name='approver' input='true' initializer='script:identity.getManager().getName();'>
```

```
<Description>
```

Holds the name of the identity to approve the request.

```
</Description>
```

```
</Variable>
```

- Variables are global to the workflow and available to all steps/approvals/transitions/etc.
- Input
 - Option means it may be passed into the workflow
- Initializer
 - Used to provide a value if one is not passed in
- Description
 - Should be meaningful

The screenshot shows the 'Process Variables' tab in the SailPoint interface. It features a list of variables, each with a dropdown arrow, a name, and a description. The variables listed are: 'identitizer' (Instance of sailpoint.object.Identitizer that la...), 'identity' (Identity object we're refreshing. This is a tr...), 'refreshOptions' (Map of options to configure Identitizer when...), 'identityName' (The name of the identity object being refres...), 'project' (A ProvisioningProject object describing the...), and 'changeEvents' (A list of IdentityChangeEvent objects gene...).

Variable Name	Description
identitizer	Instance of sailpoint.object.Identitizer that la...
identity	Identity object we're refreshing. This is a tr...
refreshOptions	Map of options to configure Identitizer when...
identityName	The name of the identity object being refres...
project	A ProvisioningProject object describing the...
changeEvents	A list of IdentityChangeEvent objects gene...

Workflow Variables – Internal Variables

```
<Variable name='roleNames' >  
  <Description>List of strings representing the role names to be  
  approved.</Description>  
  <Script><Source>  
    // An initialization script written in Beanshell  
    // ask the ProvisioningProject to generate the names list  
    return project.getNewRoleNames();  
  </Source></Script>  
</Variable>
```

- Lack of the “input” option means it is an internal variable
- Variables are initialized only once when the case is launched but may be changed by Beanshell later in the workflow

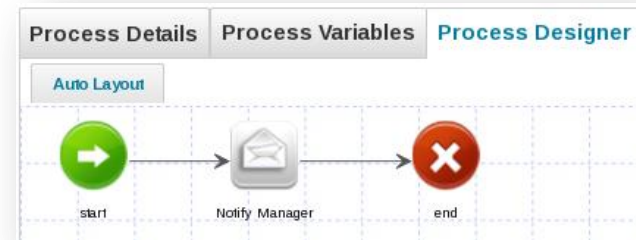
Global Variables (Reserved Words)

Variable Name	Purpose
context	SailPoint Context
log	Log object for log4J
wfcontext	An instance of WorkflowContext containing details about the state of the workflow
handler	An instance of the registered WorkflowHandler
step	The Step object currently being evaluated
approval	The Approval object currently being evaluated
item	The WorkItem being opened or assimilated
trace	Whether to trace or not (if set to true, we trace the workflow process)

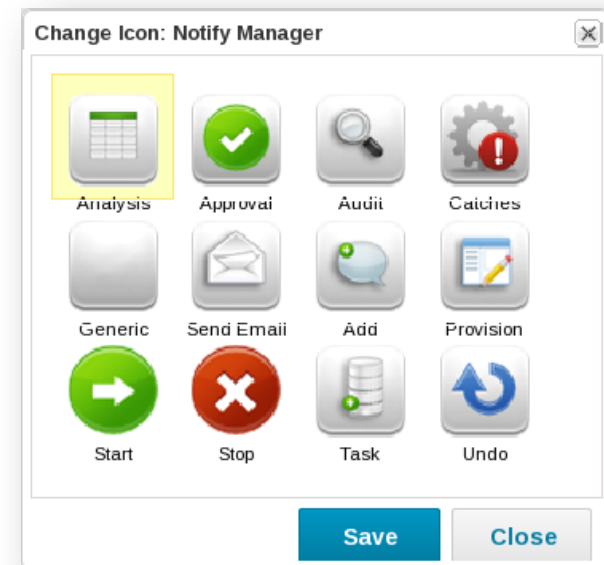
NOTE: Additional variables are added to specific types of LCM requests and vary from request to request

Workflow Steps – Overview

```
<Step name='Audit Failure' action='call:audit' posX="588" posY="7" icon='Audit'>  
  <Arg name='source' value='ref:launcher'/>  
  <Transition to='stop'/>  
</Step>
```



- **Name**
 - Steps normally have names for logging
- **Action**
 - What the step does
 - script, call to a handler method, subprocess, or approval
- **posX, PosY**
 - Step location in GUI drawing
- **Description**
 - Brief description for documentation
- **Icon**
 - Icon to display in GUI
- **Arguments**
 - Additional values to be passed to the action
- **Transition**
 - Which step to move to after the action finishes



Step Details

Coming Up...

- Arguments
- Actions
 - Script
 - Call
 - Subprocess
 - Approval
- Special purpose
 - Catches
 - Wait
- Transition

Debug Step

Details **Arguments**

Name:

Description:

Result Variable:

Enable Monitoring: ☐

Action

The action determines what action a step performs. You can configure the behavior of this step by using the fields below.

Script ☐ **Rule** ☐ **Subprocess** ☐ **Call Method** ☐

Steps

Arguments

```
<Arg name='foo' value='literal string'/>
<Arg name='bar'>
  <Script>
    <Source>
      // value calculated by Beanshell script
      identity.getAttribute("jobCode");
    </Source>
  </Script>
</Arg>
<Arg name='baz' value='I can see the other
args! bar is $(bar)'/>
```

Debug Step

Details Arguments

Add A New Argument

Name: foo

Value:

Specify the value for this by using the fields below.

String ☒ Reference ☐ Script ☐ Rule ☐ Call Method ☐

Value: literal string

Remove

- Arguments are found inside Steps and inside Approvals
- Step arguments are passed to the action scripts or called handler method
- Approval arguments are included in the work item

Step Action: Script

```
<Step name='Trace Project'>
  <Script>
    <Source>
      System.out.println('Provisioning identity: ' + identity);
      String xml = project.toXml();
      System.out.println('Project xml: ' + xml);
    </Source>
  </Script>
</Step>
```

Action

The action determines what action a step performs. You can configure the behavior of this step by using the fields:

Script ☒ Rule ☐ Subprocess ☐ Call Method ☐

Source

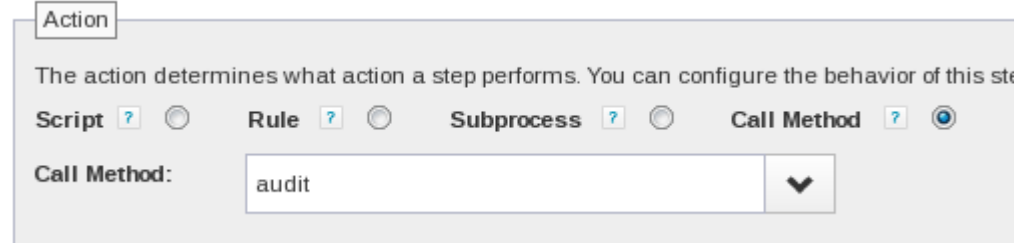
```
System.out.println('Provisioning identity: ' + identity);
String xml = project.toXml();
System.out.println('Project xml: ' + xml);
```

Open Editor

- Step action is a Beanshell script of any complexity
- All current workflow variables are accessible as top-level symbols
 - “identity” and “project” are both workflow variables

Step Action: Call

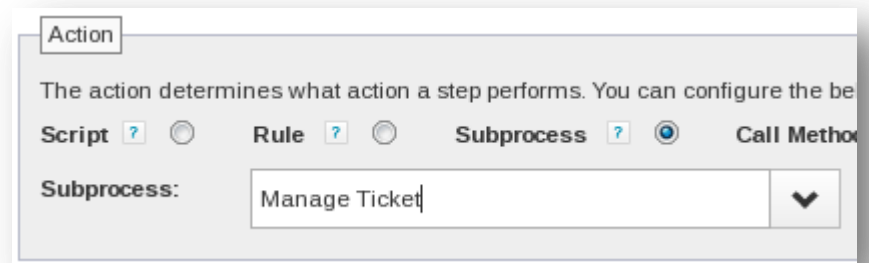
```
<Step name='Audit Failure' action='call:audit'  
posX="588" posY="7" icon='Audit'>  
  <Arg name='source' value='ref:launcher'/>  
  <Arg name='action'  
value='call:getApprovalAuditAction'/>  
  <Arg name='target' value='Role: $(roleName)'/>  
  <Arg name='string1' value='failure'/>  
  <Transition to='stop'/>  
</Step>
```



- Step action is to call a method in the StandardWorkflowHandler
 - Method “audit” is called with arguments
- Argument values can be specified using scriptlets
- See “target” argument value for an example of using \$() in a string literal to substitute workflow variable values

Step Action: Subprocess

```
<Step icon="Task" name="Initialize" posX="320" posY="126">
  <Arg name="flow" value="ref:flow"/>
  ...
  <Description>
    Description Here
  </Description>
  <Return name="project" to="project"/>
  <Return name="approvalSet" to="approvalSet"/>
  ...
  <WorkflowRef>
    <Reference class="sailpoint.object.Workflow" name="Manage Ticket"/>
  </WorkflowRef>
  <Transition to="end">
</Step>
```



Action

The action determines what action a step performs. You can configure the behavior of the action using the following options:

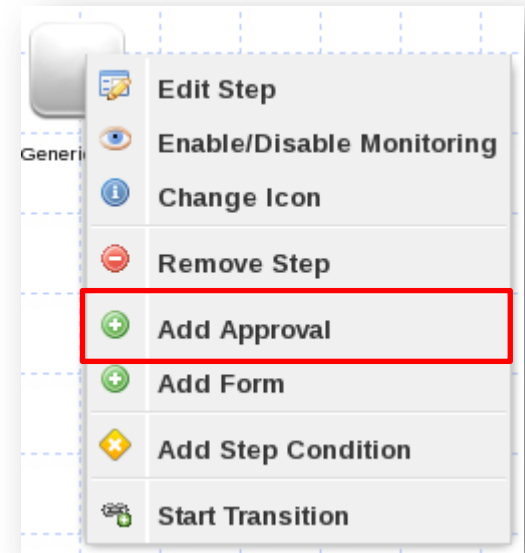
Script ☐ **Rule** ☐ **Subprocess** ☒ **Call Method** ☐

Subprocess:

- Calls another workflow
- Defines parameter passing behavior to map subprocess variable using Return tags

Step Action: Approval

```
<Step name='Do Approval'>
  <Approval owner='spadmin'>
    <Arg name='workItemDescription'
      value='Approve change to user $(identity)'/>
    </Approval>
  </Step>
```



- Approval element is special kind of step that interacts with users
- Approval steps cannot have call or script actions or call subprocesses
- Approval objects may be nested in a complex hierarchy

Step Action: Approval

Approval Work Items

- WorkItem Creation

- Each Approval generates a Workitem
- Workflow variables may be passed into the WorkItem
- WorkItem is rendered in UI by a form or renderer (JSF)
- User sees the rendered WorkItem from their Inbox or from the WorkItem interface

- WorkItem Completion

- User must complete WorkItem for workflow to continue
- Validation can occur on entered data and WorkItem can be returned
 - Messages can indicate what was missing
Example: *Email address has incorrect format* or *Approver Comments Missing*
- Submitted data may be used for future transition logic or saved in workflow variables
Example: Approval or Rejection, Approver comments

Step Action: Approval

Approval Groups

```
<Approval name='Group Approval' mode='serial'>  
  <Arg name='workItemDescription' value='Approve change to user $(identity)'/>  
  <Approval owner='billy'/>  
  <Approval owner='dusty'/>  
  <Approval owner='frank'/>  
</Approval>
```

- Child approvals must complete before the parent approval completes
- Child approvals inherit the Args and other options of the parent approval
- Parent approval has a “mode” that determines how child approvals are processed
- The default mode is “serial”

Step Action: Approval

Simple Approval Expansion

All of these examples are effectively the same

```
<Approval name='Approve'>  
  <Approval owner='billy'/>  
  <Approval owner='dusty'/>  
</Approval>
```

```
<Approval name='Approve' owner='billy,dusty'/>
```

```
<Variable name='approvers' initializer='billy,dusty'/>  
<Approval name='Approve' owner='ref:approvers'/>
```

Step Action: Approval

Approval Modes

- **Serial**
 - Approvals processed one at a time, the first reject terminates the parent approval
- **SerialPoll**
 - Approvals are processed one at a time, all are allowed to either approve or reject
- **Parallel**
 - Approvals are processed concurrently, the first reject terminates the parent approval
- **ParallelPoll**
 - Approvals processed concurrently, all are allowed to approve or reject
- **Any**
 - Approvals processed concurrently, first responder makes the decision for the group

Step Action: Approval

Approval Options

```
<Approval name='Manager Approval' owner='ref:manager'
  send='identity,newRoles' return='managerComments'
  renderer='customManagerApproval.xhtml'>
  <Arg name='workItemDescription' value='Approve change to user
$(identity)'/>
</Approval>
```

- **send** and **return** attributes have a CSV of workflow variable names to send in and return out of the approval
- **renderer** – specifies a JSF fragment to render the work item contents (alternatively can be done with a form)
- **workItemDescription** (Reserved Arg) – used to set the description of the work item that will appear in the inbox
 - If not set, the Approval name is used

Step Action: Approval

Forms – Work Item Rendering

- Forms are used to layout HTML form objects to gather data from the user or ask for approvals
- Defined within Approval element
- Values are passed in/passed out from workflow
- Results determine what to do with work item:
 - back – reject the work item
 - next – approve the work item
 - refresh – update the form in place
 - cancel – do nothing, return, leaving work item in working state
- **AfterScript** can operate on information after it is entered
 - place values back into workflow variables
 - manipulate objects based on entered data
 - validate and refuse changes, forcing user to adjust/re-enter

Step Action: Approval

Form Basics

```
<Approval owner='spadmin' send='giftwrap' return='giftwrap'>
```

```
<Form id='foo'>
```

```
<Attributes>
```

```
<Map>
```

```
<entry key='pageTitle' value='A Form Is Born'>
```

```
<entry key='title' value='Giftwrap?'>
```

```
</Map>
```

```
</Attributes>
```

```
<Section>
```

```
<Field name='giftwrap' displayName='Select giftwrap' type='string'>
```

```
<AllowedValues>
```

```
<String>Plain Offwhite</String>
```

```
<String>Masculine Stripes</String>
```

```
</AllowedValues>
```

```
</Field>
```

```
</Section>
```

```
<Button label='Reject' action='back'>
```

```
<Button label='Approve' action='next'>
```

```
<Button label='Do Nothing' action='cancel'>
```

```
</Form>
```

```
</Approval>
```

Step Action: Approval

Work Item Escalation

```
<Approval name='Approve' owner='billy'>
  <WorkItemConfig hoursTillEscalation='96' hoursBetweenReminders='24'
maxReminders='3'>
    <NotificationEmail>
      <Reference class='EmailTemplate' value='Approval Notification'/>
    </NotificationEmail>
    <ReminderEmail>
      <Reference class='EmailTemplate' value='Approval Reminder'/>
    </ReminderEmail>
    <EscalationEmail>
      <Reference class='EmailTemplate' value='Approval Escalation'/>
    </EscalationEmail>
    <EscalationRule>
      <Reference class='Rule' value='Escalation Rule'/>
    </EscalationRule>
  </WorkItemConfig>
</Approval>
```

- WorkItemConfig may be at any level of the Approval hierarchy and may also appear at the top-level of the Workflow
- WorkItemConfigs are inherited, those lower in the hierarchy have priority over those higher

Step Action: Catches

```
<Step name='Finalize' catches="complete">
```

```
<Description>
```

 This step will get called last

```
</Description>
```

```
</Step>
```

- Option catches="complete"
- Step that gets called when a workflow is complete
- Can handle error handling or marking a request as complete
- Used in our "LCM Provisioning" workflow to mark identity request item as complete or failed when workflow ends.

Step Action: Wait

```
<Step name="Wait for next retry" wait="ref:retryInterval">
```

```
<Description>
```

This is a sleep step and just waits to execute the next step.

Typically only called when we are retrying to give us some delay between calls down to the PE. Skipped during the first loop.

```
</Description>
```

```
<Transition to="Provision Retry"/>
```

```
</Step>
```

- Step uses wait="**<Value>**" to specify a sleep time in minutes

Transitions

- Transitions are evaluated when the step action or approval completes
- Evaluation is in the order written
- The first transition whose condition evaluates to true, or which has no condition, is taken
- If there are no transitions, or if all of the transition conditions are false, the workflow normally ends
 - Word of caution, be specific about transitions

Transition Examples

```
<Transition to='Some Step' when='isTrue(approved)'/>
<Transition to='Another Step'>
  <Script><Source>
    identity.getManager().getName().equals("dave");
  </Source></Script>
</Transition>
<Transition to='end'/'>
```

- **to**
 - Specifies the name of the next step
- **when**
 - Has a scriptlet that tests the value of the built-in “approved” variable
- Second transition has a complex script condition rather than a single line “when” scriptlet
- Third transition is unconditional = fallback transition

Workflow Troubleshooting

Debugging Workflows

■ Tracing

- Turn on “trace” to “true” in workflows – Output goes to stdout for the app server

■ Email Redirection

- redirect email to a file (System Setup → IdentityIQ Settings → Miscellaneous)

■ Logging

- `system.out.println()` or `log4j` are your friends
- `log4j.logger.sailpoint.api.Workflower=debug`
- `log4j.logger.sailpoint.workflow.WorkflowHandler=debug`
- `log4j.logger.sailpoint.workflow.WorkflowLibrary=debug`

■ Console

- `list workitem`
- `list workflowcase`
- The “workflow” console command can be used to launch a workflow case with an XML file containing initial workflow variables

Workflow Status

Task Result

Details

Name	Update Identity Tammy.Daniels AccessRequest	Started By	Catherine.Simmons
Type	LCM	Started	2/21/14 2:32:26 PM
Description	Workflow Case	Completed	
Status	pending...		

[Return to Tasks](#)

Current Workflow Step: Approve

Interactions

Owner	Type	Request	Status	Started	Completed
Randy.Knight	Approval	Owner Approval - Account Changes for User: Tammy.Daniels	Open	2/21/14 2:32:28 PM	

Debug Pages

Object Browser

WorkflowCase



Filter by Name or ID



[Configuration Objects](#)

<input type="checkbox"/>	Id	Name	Created	Modified
<input type="checkbox"/>	ff80808143db395f0143e482d258033e	Native Change Detection LDAP: Allen.Bu...	1/30/14 12:56 PM	1/30/14 12:56 PM
<input type="checkbox"/>	ff8080814451375f0144562626d10071	Update Identity Denise.Hunt AccessReq...	2/21/14 2:32 PM	2/21/14 2:32 PM
<input type="checkbox"/>	ff8080814451375f0144562612790061	Update Identity Irene.Mills AccessRequest	2/21/14 2:32 PM	2/21/14 2:32 PM
<input type="checkbox"/>	ff8080814451375f014456261def0069	Update Identity Jeremy.Palmer AccessR...	2/21/14 2:32 PM	2/21/14 2:32 PM
<input type="checkbox"/>	ff8080814451375f0144562604460059	Update Identity Louis.Black AccessRequest	2/21/14 2:32 PM	2/21/14 2:32 PM
<input type="checkbox"/>	ff8080814451375f01445626313a0079	Update Identity Tammy.Daniels AccessR...	2/21/14 2:32 PM	2/21/14 2:32 PM
<input type="checkbox"/>	ff8080814451375f014465fa6e4100d7	Update Identity Tammy.Daniels Accounts...	2/24/14 4:18 PM	2/24/14 4:18 PM

Workflow Metrics

- Capture analytics for entire workflow
 - Avg/Max/Min Execution Time, # of Executions
- Drill into individual steps and identify execution bottlenecks
- View metrics
 - Process Metrics tab
 - Advanced Analytics → Process Metrics Search

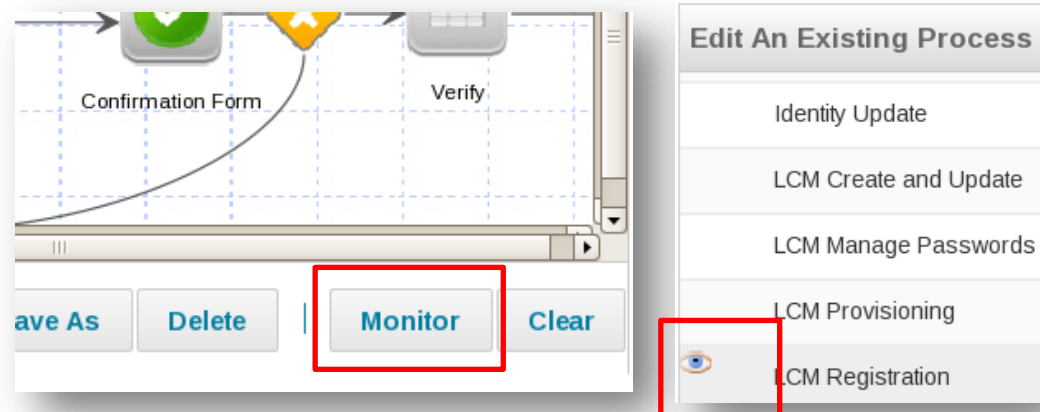
Advanced Analytics

Identity Search	Access Review Search	Role Search	Account Group Search	Activity Search	Audit Search	Process Metrics Search
Process Metrics Search ►► Process Metrics Search Results						
Filter						
Name LCM Registration						
Result Status All						
Process Metrics Search Results						
Show times in Minutes ▼						
Process	Average Execution Time	Maximum Execution Time	Minimum Execution Time	Number Of Executions		
LCM Registration	11.27	15.23	8.65	3		

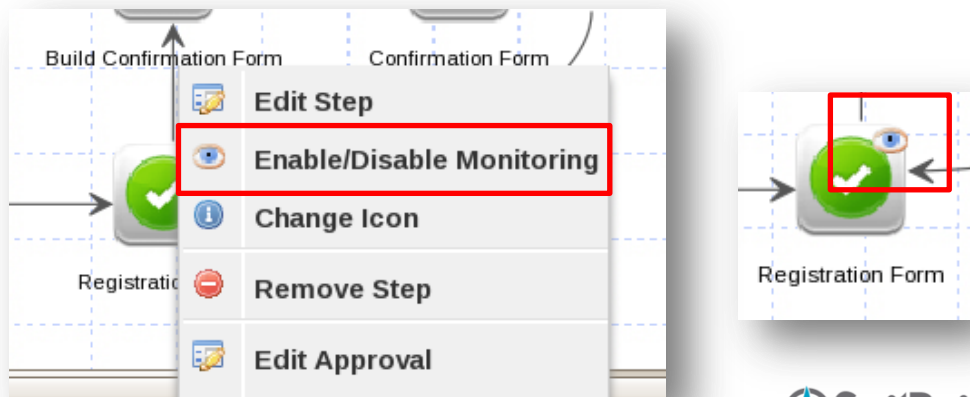
Workflow Monitoring

- Monitoring for whole process or individual steps

- Monitor Process



- Monitor step



Questions?

Exercise Preview

Section 4, Exercise 6

- Exercise 6: Use Lifecycle Manager to Create a Lifecycle Event
 - Configure a new Business Process for an event