

Table of Contents

| | |
|--------------------------------|----|
| Model Detail | 2 |
| Orchestration | 2 |
| BasicPipelineConfigurator | 4 |
| BasicProductionRunConfigurator | 5 |
| DatabaseConfigurator | 5 |
| ProductionRun | 7 |
| ProductionRunConfigurator | 7 |
| ProductionRunManager | 9 |
| ProvenanceRecorder | 11 |
| ProvenanceSetup | 11 |
| StatusListener | 13 |
| Workflow | 15 |
| WorkflowConfigurator | 15 |
| WorkflowLauncher | 16 |
| WorkflowManager | 17 |
| WorkflowMonitor | 19 |

Model Documentation

Model Detail

This document provides a complete overview of all element details. For simpler and more focused reports, simply copy this initial template and turn off the sections not required.

Orchestration

Type: **Package**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Control
Detail: Created on 2/2/2010. Last modified on 2/2/2010
GUID: {335E00E5-E8A4-4a6d-B9A7-7B8387414744}

Orca - (Logical diagram)

Created By: rplante on 7/17/2009
Last Modified: 2/8/2010
Version: 1.0. *Locked:* False
GUID: {CBC8BE46-DEE1-4118-A617-F6C78105877F}

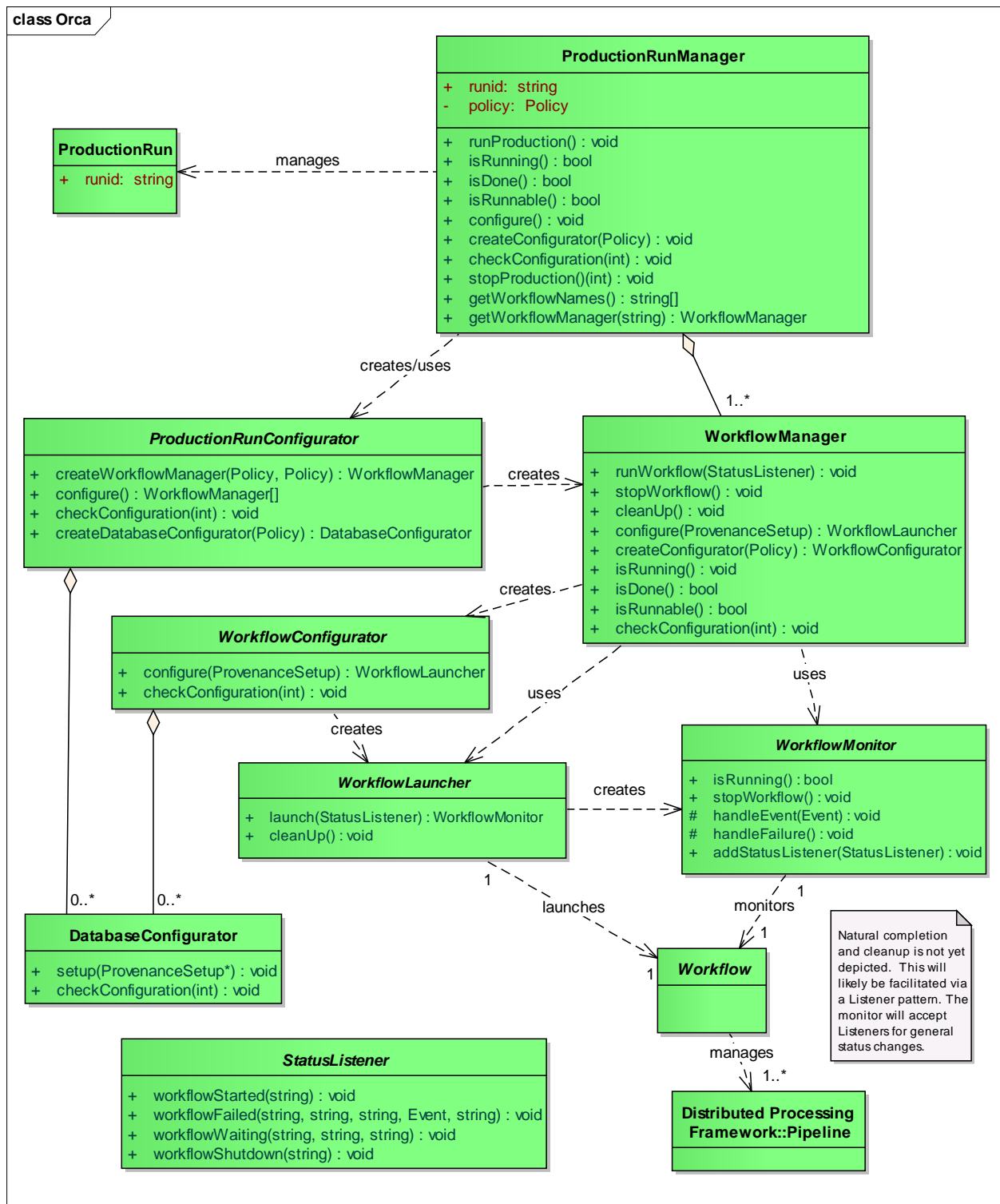


Figure: 1

Provenance - (Logical diagram)

Created By: Ray Plante on 2/5/2010

Last Modified: 2/5/2010

Version: 1.0. Locked: False
 GUID: {DEAC91DD-5B44-4e3e-98FF-22E9D55A4EBE}

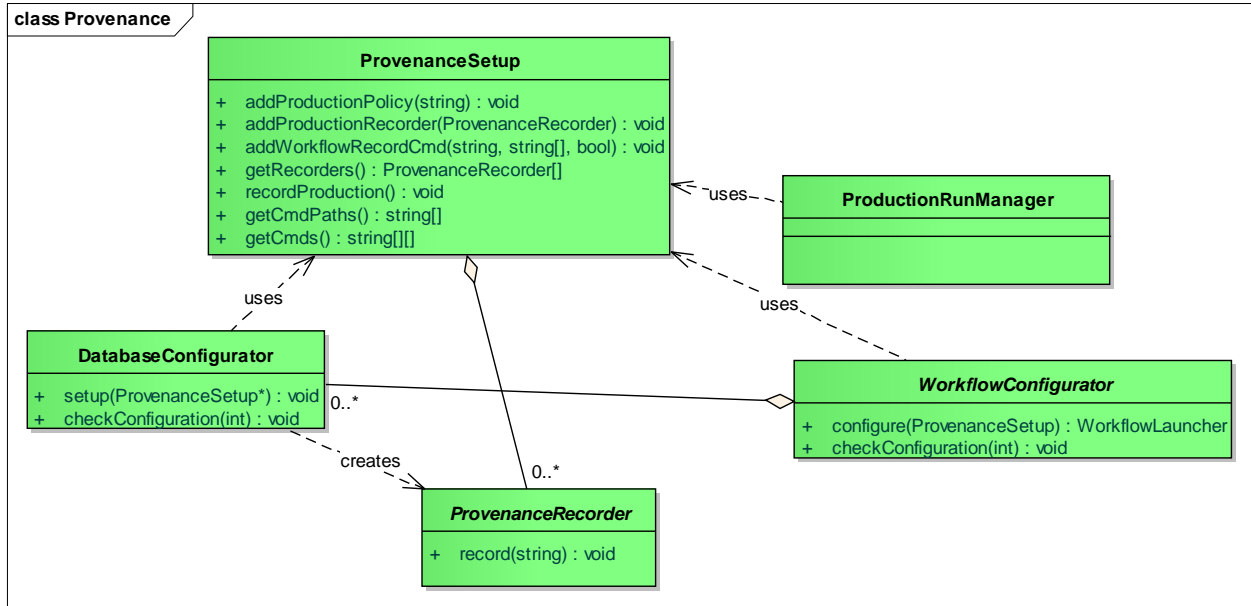


Figure: 2

BasicPipelineConfigurator

Type: **Class WorkflowConfigurator**
 Status: Proposed. Version 1.0. Phase 1.0.
 Package: Orchestration *Keywords:*
 Detail: Created on 7/20/2009. Last modified on 2/2/2010.
 GUID: {D71F075C-035E-4051-8E86-0D608F22A7D3}

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|----------------------------------|----------------------------------|-------|
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public BasicPipelineConfigurator | |
| Generalization Source -> Destination | Public BasicPipelineConfigurator | Public WorkflowConfigurator | |

Operations

| Method | Notes | Parameters |
|--|--|------------|
| configure() WorkflowLauncher Public | Setup as much as possible in preparation to execute the pipeline and return a PipelineLauncher object that will launch the configure pipeline. | |
| createNodeList() void Public | | |
| prepPlatform() void Public | | |
| deploySetup() void Public | | |
| createDirs() void Public | | |
| setupDatabase() void Public | initialize any databases required by this pipeline. | |

BasicProductionRunConfigurator

Type: **Class** ProductionRunConfigurator
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/20/2009. Last modified on 2/2/2010.
GUID: {C78EBDA4-EB03-4acf-8DFC-87ADF80F9369}

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|--|--|-------|
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public BasicProductionRunCo nfigurator | |
| Generalization Source -> Destination | Public BasicProductionRunCo nfigurator | Public ProductionRunConfigur ator | |

DatabaseConfigurator

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/20/2009. Last modified on 2/8/2010.

GUID: {DBD1245C-CC09-4dc0-9F52-768498FCEE43}

A class for setting up a database used for an individual workflow or for entire production run.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|-----------------------------|---------------------------------------|--|
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public BasicPipelineConfigurator | |
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public BasicProductionRunConfigurator | |
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public ProductionRunConfigurator | |
| Dependency creates Source -> Destination | Public DatabaseConfigurator | Public ProvenanceRecorder | The DatabaseConfigurator creates a ProvenanceRecorder that understands how to talk to its database. |
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public WorkflowConfigurator | |
| Dependency uses Source -> Destination | Public DatabaseConfigurator | Public ProvenanceSetup | The DatabaseConfigurator may use the ProvenanceSetup instance passed to it via setup() to add a ProvenanceRecorder for its database. |

Operations

| Method | Notes | Parameters |
|---|---|---|
| setup() void Public | setup the required databases. If this database is to receive provenance information, this function will use the ProvenanceSetup instance to register this intent. | ProvenanceSetup [inout] provSetup A visiting ProvenanceSetup that this configurator should use to register its interest in receiving provenance data. If null, the configurator is not allowed to receive provenance data. |
| checkConfiguration() void Public | | int [in] care the thoroughness of the checks where a higher number implies more checks |

| Method | Notes | Parameters |
|--------|-------|------------|
| | | |

ProductionRun

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/17/2009. Last modified on 2/2/2010.
GUID: {F2F4DA6B-7834-41fd-AEC3-190D20FFFAA9}

A set of coupled Pipelines.

It is not expected that this domain class will be realized as a software class.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|--------------------------------|-------------------------|-------|
| Dependency manages Source -> Destination | Public ProductionRunManager | Public ProductionRun | |

Attributes

| Attribute | Notes | Constraints and tags |
|-------------------------------|--|----------------------|
| runid string Public | A name identifying the execution context of this production run. It is a way of distinguishing one ProductionRun from another. | <i>Default:</i> |

ProductionRunConfigurator

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/20/2009. Last modified on 2/4/2010.
GUID: {9976D80F-172B-4dcb-B9B2-233838DD899C}

a class for configuring a production run as a whole.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|--|-------------------------------------|-------|
| Dependency creates/uses Source -> Destination | Public ProductionRunManager | Public ProductionRunConfigurator | |
| Generalization Source -> Destination | Public BasicProductionRunConfigurator | Public ProductionRunConfigurator | |
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public ProductionRunConfigurator | |
| Dependency creates Source -> Destination | Public ProductionRunConfigurator | Public WorkflowManager | |

Operations

| Method | Notes | Parameters |
|---|---|--|
| createWorkflowManager() WorkflowManager Public | create the WorkflowManager for the workflow described by the given short name. | Policy [in] wfPolicy The policy that describes the particular workflow to be created. Policy [in] prodPolicy the policy that describes this production run as a whole. This is used to get data that needs to be passed on down into the pipelines (e.g. event broker host) |
| configure() WorkflowManager[] Public | carry out production-level configuration and setup. To complete the configuration, the createPipelineManager() function must be called to create each the pipeline managers on which the configure() function must be called. | |
| checkConfiguration() void Public | run checks to test whether this production run is setup and ready to run. A ConfigurationError exception is raised if any check fails. When care=0 (or less) this will only raise an exception if the workflow managers not been created yet. | int [in] care a measure of how careful or thorough a check to do where higher numbers imply more checks. A value of zero or less implies minimal or no checks. |
| createDatabaseConfigurator() DatabaseConfigurator Public | | Policy [in] policy the database usage policy. This is the item "database" in the production-level policy |

ProductionRunManager

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration **Keywords:**
Detail: Created on 7/17/2009. Last modified on 2/2/2010.
GUID: {DA5667A3-33EF-46a0-AFA7-035F8F48F71A}

A class in charge of launching, monitoring, managing, and stopping a ProductionRun.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|--------------------------------|-------------------------------------|--|
| Dependency creates/uses Source -> Destination | Public ProductionRunManager | Public ProductionRunConfigurator | |
| Dependency manages Source -> Destination | Public ProductionRunManager | Public ProductionRun | |
| Dependency uses Source -> Destination | Public ProductionRunManager | Public ProvenanceSetup | The ProductionRunManager uses the ProvenanceSetup that it gets back from the ProductionConfigurator to actually record provenance just before launching all the workflows. |
| Aggregation Source -> Destination | Public WorkflowManager | Public ProductionRunManager | |

Attributes

| Attribute | Notes | Constraints and tags |
|---------------------------------|---|----------------------|
| runid string Public | a name that identifies the execution context of this productions run. | <i>Default:</i> |
| policy Policy Private | the policy that describes the production run as a whole | <i>Default:</i> |

| Attribute | Notes | Constraints and tags |
|-----------|-------|----------------------|
| | | |

Operations

| Method | Notes | Parameters |
|--|--|---|
| runProduction() void Public | run all pipeines to completion. | |
| isRunning() bool Public | return true if pipelines are currently running. | |
| isDone() bool Public | return true if the pipelines have all completed. | |
| isRunnable() bool Public | return true if the possible to call runProduction() to execute the production. It may return false if it is already running or if | |
| configure() void Public | setup this production run according to the production policy. This will use the createConfigurator() method to create the convfigurator. | |
| createConfigurator() void Public | | Policy [in] prodPolicy the production run policy |
| checkConfiguration() void Public | run checks that ensure that all pipelines are properly setup. This will call checkConfiguration() on each of the workflows. | int [in] care The level of care in checking the configuration to take. In general, the higher the number, the more checks that are made. |
| stopProduction() () void Public | stop all pipelines | int [in] urgency an indicator of how urgently to carry out the shutdown. Recognized values are: FINISH_PENDING_DATA end after all currently available data has been processed END_ITERATION end after the current data ooping iteration CHECKPOINT end at next checkpoint opportunity (typically between stages) NOW end as soon as possible, forgoing any checkpointing |
| getWorkflowNames() string[] Public | return the "short" name for each workflow in this production. These are names that can be passed to getWorkflowManager(). | |
| getWorkflowManager() WorkflowManager Public | return the WorkflowManager identified by the given "short" name. None is returned if the name is either not recognized or the associated WorkflowManager has not be created yet. | string [in] name the "short" name associated with the desired workflow |

ProvenanceRecorder

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 2/5/2010. Last modified on 2/5/2010.
GUID: {65ED5A52-329F-4af9-91BE-B72A05007856}

an abstract interface for recording production-level policy data as provenance into a particular database. A DatabaseConfigurator instance will instantiate a subclass that is wired for that particular database.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|--------------------------------|------------------------------|---|
| Dependency creates Source -> Destination | Public DatabaseConfigurator | Public ProvenanceRecorder | The DatabaseConfigurator creates a ProvenanceRecorder that understands how to talk to its database. |
| Aggregation Source -> Destination | Public ProvenanceRecorder | Public ProvenanceSetup | |

Operations

| Method | Notes | Parameters |
|--------------------------------|-------|---|
| record() void Public | | string [in] filename the path to a local policy file to load into record into provenance. |

ProvenanceSetup

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 2/5/2010. Last modified on 2/5/2010.
GUID: {20BEBBC49-6721-4977-80D2-1C13C1D052D7}

a container for collecting and bringing together providers and consumers of provenance information in order to record provenance for a production run.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|--------------------------------|---------------------------|--|
| Dependency uses Source -> Destination | Public ProductionRunManager | Public ProvenanceSetup | The ProductionRunManager uses the ProvenanceSetup that it gets back from the ProductionConfigurator to actually record provenance just before launching all the workflows. |
| Dependency uses Source -> Destination | Public DatabaseConfigurator | Public ProvenanceSetup | The DatabaseConfigurator may use the ProvenanceSetup instance passed to it via setup() to add a ProvenanceRecorder for its database. |
| Aggregation Source -> Destination | Public ProvenanceRecorder | Public ProvenanceSetup | |
| Dependency uses Source -> Destination | Public WorkflowConfigurator | Public ProvenanceSetup | The WorkflowConfigurator uses the ProvenanceSetup to get the commands that should be inserted into the launch scripts that get run on the remote platform. |

Operations

| Method | Notes | Parameters |
|--|---|--|
| addProductionPolicy() void Public | | string [in] policyfile Add the local path to a policy file that contains some portion of the production-level pipeline. Production-level policy data is all policy data consumed by the orchestration layer in order to remotely launch the workflows. It generally does not include the policy data consumed by the pipeline harness to run the pipeline (as this is usually record on the remote platform. |
| addProductionRecorder() void Public | | ProvenanceRecorder [in] recorder the ProvenanceRecorder object to use to record the production-level provenance. |
| addWorkflowRecordCommand() void Public | register the desire to receive workflow-level provenance by providing shell-executable command to record that provenance. The expectation is that this command will be executed on the remove execution platform when the workflow is launched. The cmd | string [in] cmd the name of the command (without arguments) to execute. string[] [in] args the arguments to pass to command (prior to the list of policy files) |

| Method | Notes | Parameters |
|---|--|--|
| | argument must not include any path as part of its name. The command must accept one or more arguments (after the provided args) representing filenames of policies to record. | bool [in] path the path to executable on the launch platform. This path allows the executable to be copied over to the execution platform where the workflow will run. |
| getRecorders() ProvenanceRecorder[] Public | return the recorders that have been added to this setup thus far. | |
| recordProduction() void Public | record the production-level policy provenance to all interested databases. This will do this by looping through each ProvenanceRecorder that it has and call its record() function. | |
| getCmdPaths() string[] Public | return the paths to the registered provenance-recording scripts. This is used to find the scripts that must be transferred to the remote workflow platform where they will be executed. | |
| getCmds() string[][] Public | Return the list of commands that should be run on each workflow platform to record provenance for that workflow. The returned value is a list of list of strings. Each element of the outer list represents the command to execute. The first element of the inner list is the name of the executable command (sans path); the remaining elements are the arguments to pass, in order. The expected algorithm for forming a complete command line for each command (i.i. element in the outer list) is: <ol style="list-style-type: none"> 1. prepend to the first element (of inner list) the directory path for the location of the executable on the remote platform. 2. append as extra arguments the pathnames to the policy files to record. 3. join and execute the command word list. | |

StatusListener

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration **Keywords:**
Detail: Created on 2/6/2010. Last modified on 2/6/2010.
GUID: {FABD2295-EA3E-4268-B393-675268282EEE}

An interface for getting notified about changes in the status of a workflow: when it has started and when it has finished.

Custom Properties

- isActive = False

Operations

| Method | Notes | Parameters |
|---|--|---|
| workflowStarted() void Public | Called when a workflow has started up correctly and is ready to process data. Note that if a pipeline is waiting for an event, the listener should be notified via a workflowWaiting() message. | string [in] name the short name for the workflow that has successfully started |
| workflowFailed() void Public | indicate that a workflow has experienced an as-yet unhandled failure that prevents further processing. | string [in] name the name of the failed workflow. string [in] errorName a logical name for the failure that maps to a specific meaning. "Unknown" is reserved to mean that the failure does not map to a defined error. string [in] errmsg a human-readable description of the error that caused the failure. Event [in] event An event that reported the failure, if one did so. This can provide information about where and why it occurred. Null indicates that no event was involved in the reporting. Note that the source of such notifications won't typically report on errors or failures it can handle itself. string [in] pipelineName The name of the pipeline where the failure occurred. This may be null if the failure did not occur within a specific pipeline or if the originating pipeline is otherwise not known. |
| workflowWaiting() void Public | indicate that a workflow is waiting for an event to proceed. This should only be called only for events that are expected from outside the workflow. Events that are meant to travel between Pipelines within a workflow should not trigger this notification. | string [in] name the name of the waiting workflow string [in] eventTopic a event topic (referring to the semantics of the event message) for the event being waited on. If null, the topic is known by the notifier. string [in] pipelineName the name of the specific pipeline waiting for the event. If null, it is not a Pipeline that is waiting. |

| Method | Notes | Parameters |
|---|--|--|
| workflowShutdown() void Public | the workflow has successfully shutdown and ready to be cleaned up. | string [in] name the name of the workflow that has been shutdown |

Workflow

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 2/2/2010. Last modified on 2/2/2010.
GUID: {0940F8FD-8DF0-453e-B589-E1E60A2BA359}

A set of pipelines configured to run on a given platform in a certain order and/or with certain dependencies. Specific implementations may use different workflow engines to manage the flow.

In simple modes, a workflow will be a single pipeline.

Note that it is not expected that there will be a Workflow class in code. This domain class exists primarily to capture the definition of a workflow.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|----------------------------|--------------------|--|
| Dependency manages Source -> Destination | Public Workflow | Public Pipeline | Through a workflow engine, a workflow will launch and manage one or more Pipelines |
| Dependency launches Source -> Destination | Public WorkflowLauncher | Public Workflow | |
| Dependency monitors Source -> Destination | Public WorkflowMonitor | Public Workflow | |

WorkflowConfigurator

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/17/2009. Last modified on 2/4/2010.
GUID: {5B38FD54-9234-47fe-AE06-5434CAE24717}

A class that configures a workflow to run on a platform. This can include copying all files (including policy files) to the execution platform and prepping the database.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|-------------------------------------|--------------------------------|--|
| Aggregation Source -> Destination | Public DatabaseConfigurator | Public WorkflowConfigurator | |
| Dependency creates Source -> Destination | Public WorkflowConfigurator | Public WorkflowLauncher | |
| Dependency uses Source -> Destination | Public WorkflowConfigurator | Public ProvenanceSetup | The WorkflowConfigurator uses the ProvenanceSetup to get the commands that should be inserted into the launch scripts that get run on the remote platform. |
| Dependency creates Source -> Destination | Public WorkflowManager | Public WorkflowConfigurator | |
| Generalization Source -> Destination | Public BasicPipelineConfigurator | Public WorkflowConfigurator | |

Operations

| Method | Notes | Parameters |
|--|--|--|
| configure() WorkflowLauncher Public | Setup as much as possible in preparation to execute the pipeline and return a PipelineLauncher object that will launch the configure pipeline. | ProvenanceSetup [in] provSetup the ProvenanceSetup instance to visit to DatabaseConfigurators for this workflow. |
| checkConfiguration() void Public | run checks on the production-level wide setup, and raise a ConfigurationError exception if any problems are found | int [in] care a measure of the thoroughness of the check where a higher number means more checks. |

WorkflowLauncher

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/17/2009. Last modified on 2/6/2010.
GUID: {3F92BCF0-A8F7-48b1-AD8C-E992318D54EF}

A class that actually starts a workflow on the configured platform

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|--------------------------------|----------------------------|-------|
| Dependency creates Source -> Destination | Public WorkflowLauncher | Public WorkflowMonitor | |
| Dependency creates Source -> Destination | Public WorkflowConfigurator | Public WorkflowLauncher | |
| Dependency uses Source -> Destination | Public WorkflowManager | Public WorkflowLauncher | |
| Dependency launches Source -> Destination | Public WorkflowLauncher | Public Workflow | |

Operations

| Method | Notes | Parameters |
|--|---|--|
| launch() WorkflowMonitor Public | launch the pipeline on the (remote) execution platform and return a PipelineMonitor that can be used to monitor it. | StatusListener [in] listener if provided, this listener will be notified whenever the run status of this workflow changes. |
| cleanUp() void Public | clean up after a deployed and possible executed Pipeline. If called before launching, it will just clean-up the environment put into place as part of the configuration phase. If called after launching, it also cleans up the products as well. | |

WorkflowManager

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/17/2009. Last modified on 2/2/2010.
GUID: {1E5CBF4C-0836-491e-BF34-47B9309BA592}

A manager class for managing a workflow deployed and executed on a platform.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|--|-------------------------------------|--------------------------------|-------|
| Dependency creates Source -> Destination | Public ProductionRunConfigurator | Public WorkflowManager | |
| Aggregation Source -> Destination | Public WorkflowManager | Public ProductionRunManager | |
| Dependency uses Source -> Destination | Public WorkflowManager | Public WorkflowMonitor | |
| Dependency uses Source -> Destination | Public WorkflowManager | Public WorkflowLauncher | |
| Dependency creates Source -> Destination | Public WorkflowManager | Public WorkflowConfigurator | |

Operations

| Method | Notes | Parameters |
|---|--|--|
| runWorkflow() void Public | setup, launch, and monitor a workflow to its completion, and then clean-up. | StatusListener [in] listener if provided, this StatusListener will be notified whenever the run status changes for this workflow. |
| stopWorkflow() void Public | stop the running workflow as soon as possible. If successful, this call should be followed by a call to cleanUp(). | |
| cleanUp() void Public | Carry out post-execution tasks for removing pipeline data and state from the platform and archiving/ingesting products as needed. | |
| configure() WorkflowLauncher Public | prepare a pipeline for launching. This calls createConfigurator() and calls the returned configurator's configure() function. | ProvenanceSetup [in] provSetup the ProvenanceSetup instance to visit to DatabaseConfigurators for this workflow |
| createConfigurator() WorkflowConfigurator Public | create and return a PipelineConfigurator instance for the Pipeline. | Policy [in] wfPolicy the workflow policy for this workflow |
| isRunning() void Public | return True if the workflow is currently running. | |
| isDone() bool Public | return true if the pipeline has been run to completion. This will be true if the pipeline has run normally through cleaned up or if it was stopped and clean-up has been called. | |
| isRunnable() bool Public | return true if runPipeline() can still be called. This may return false because the pipeline has already been run and cannot be rerun. | |

| Method | Notes | Parameters |
|---|---|---|
| checkConfiguration() void Public | Runs checks that ensure that the Workflow has been properly setup. A ConfigurationError exception is raised if any check fails. When care=0 (or less) this will only raise an exception if the workflow configuration not been created yet. | int [in] care an indication of how thoroughly to check. In general, a higher number will result in more checks being run. |

WorkflowMonitor

Type: **Class**
Status: Proposed. Version 1.0. Phase 1.0.
Package: Orchestration *Keywords:*
Detail: Created on 7/17/2009. Last modified on 2/6/2010.
GUID: {A0AD543D-F3FD-42c5-8CAF-1F54992D793D}

A class in charge of monitoring and/or controlling the progress of a running pipeline.

Custom Properties

- isActive = False

Connections

| Connector | Source | Target | Notes |
|---|----------------------------|---------------------------|-------|
| Dependency creates Source -> Destination | Public WorkflowLauncher | Public WorkflowMonitor | |
| Dependency uses Source -> Destination | Public WorkflowManager | Public WorkflowMonitor | |
| Dependency monitors Source -> Destination | Public WorkflowMonitor | Public Workflow | |

Operations

| Method | Notes | Parameters |
|---|---|---|
| isRunning() bool Public | return true if the pipeline being monitored appears to still be running | |
| stopWorkflow() void Public | stop the workflow. | |
| handleEvent() void Protected | | Event [in] event an event delivered to this pipeline |
| handleFailure() void Protected | handle a failure | |
| addStatusListener() void Public | Add a StatusListener to this monitor. The listener will be notified when the Workflow | StatusListener [in] listener the listener interested in receiving |

| Method | Notes | Parameters |
|---------------|---|-----------------------|
| | has stopped running, either due to an unhandled failure or successful completion. | status notifications. |