

Deployit Trigger Plugin Manual

Version 3.8.0

Table of Contents

Table of Contents	2
Preface	3
Overview	3
Features	3
Requirements	3
Actions	3
Email Action	3
State Transitions	4
Task state transitions	4
Step state transitions	4
CI Reference	5
Configuration Item Overview	5
Other Configuration Items	5
Configuration Item Details	5
mail.SmtpServer	5
trigger.EmailNotification	5
trigger.StepTrigger	6
trigger.TaskTrigger	6
trigger.Trigger	6

Preface

This document describes the functionality provided by the Trigger Plugin.

Refer to the *Deployit Reference Manual* for background information on Deployit and deployment concepts.

Overview

When Deployit resolves a deployment plan, a task comprising of multiple steps is created and subsequently executed. The task, along with the steps, transition through various states before finally completing. The Trigger Plugin allows a user to associate actions, example sending an email, to be triggered for one of these state transitions.

Features

- Ability to send emails for state transitions of Steps or Task.

Requirements

This plugin requires:

- **Deployit:** version 3.8+

Actions

The following Deployit objects are available to actions :

- **deployedApplication:** The entire deployed application containing application and environment configuration items. See UDM Ci reference.
- **task:** Contains information about the task. The following properties are available,
 - id
 - state
 - description
 - startDate
 - completionDate
 - nrSteps : number of steps in the task.
 - currentStepNr : the current step been executed.
 - failureCount : number of times the task has failed
 - owner
 - steps : List of steps in the task. Not available when action triggered from StepTrigger. See step information below for properties.
- **step:** Contains information about a step. Not available when action triggered from TaskTrigger. The following properties are available,
 - description
 - state
 - log
 - startDate
 - completionDate
 - failureCount
- **action:** Reference to the executing action.

Email Action

First, you will need to define an [SMTPServer](#) under the *Infrastructure* root.

```
mailServer = factory.configurationItem("Infrastructure/MailServer", "mail.SMTPServer")
mailServer.host = "smtp.mycompany.com"
mailServer.username = "mymailuser"
mailServer.password = "secret"
mailServer.fromAddress="noreply@mycompany.com"
repository.create(mailServer)
```

The SMTPServer uses Java Mail to send emails. You can specify additional Java Mail properties in the *smtpProperties* attribute. Refer to <http://javamail.kenai.com/nonav/javadocs/com/sun/mail/smtp/package-summary.html> for a list of all properties.

The [EmailNotification](#) captures the email details that will be sent.

Under the *Configuration* root, define an [EmailNotification](#) configuration item.

```
myEmailAction = factory.configurationItem("Configuration/MyFailedDeploymentNotification",
    "trigger.EmailNotification")
myEmailAction.mailServer = "Infrastructure/MailServer"
myEmailAction.subject("Application ${deployedApplication.version.application.name} failed.")
myEmailAction.toAddresses = ["support@mycompany.com"]
myEmailAction.body = "Deployment of ${deployedApplication.version.application.name} was cancelled on environment
    $ {deployedApplicaiton.environment.name}"
repository.create(myEmailAction)
```

The *subject*, *toAddresses*, *fromAddress*, *body* properties can contain FreeMarker template scripting that can access the Deployit objects described above.

If preferred, the definition of the email body can be defined in an external template file. The path to the file can be set in the *bodyTemplatePath* property. This can either be an absolute path, or a relative path that will be resolved via Deployit's classpath.

State Transitions

When an environment has triggers defined in its *triggers* property, the Trigger plugin listens to state transitions in tasks and steps that occur during a deployment. When the state transition described by the trigger matches, the associated actions are executed. Currently the plugin has an email action.

Task state transitions

From	To	Description
Queued	Pending	Thread becomes available to execute task.
Pending	Executing	Task starts to execute.
Executing	Stopped	Task is stopped due to step failure, pause step or user aborted step.
Executing	Executed	Task has executed successfully.
Executed	Done	Task is stored in Deployit's archive.
Stopped	Executing	User continues the execution of a failed, paused or aborted step.
Stopped	Cancelled	User cancels task.

A [TaskTrigger](#) can be defined under the *Configuration* root and associated with the environment on which it should be triggered.

```
taskTrigger = factory.configurationItem("Infrastructure/TriggerOnCancel", "trigger.TaskTrigger")
taskTrigger.fromState = "ANY"
taskTrigger.toState = "CANCELLED"
taskTrigger.actions = [myEmailAction]
repository.create(taskTrigger)

env = repository.read("Environments/Dev")
env.triggers = ["Configuration/TriggerOnCancel"]
repository.update(env)
```

Step state transitions

From	To	Description
Pending	Executing	Step starts to execute.
Pending	Skipped	User skips pending step.
Skipped	Pending	Step is unskipped.
Executing	Done	Step completed executing successfully.
Executing	Failed	Step failed during execution.
Executing	Paused	Step paused its execution.
Paused	Skipped	User skips paused step.
Paused	Executing	User continues the execution of paused step.
Failed	Skipped	User skips failed step.

Failed Executing User retries the execution of failed step.

A [StepTrigger](#) can be defined under the *Configuration* root and associated with the environment on which it should be triggered.

```
stepTrigger = factory.configurationItem("Infrastructure/TriggerOnFailure","trigger.StepTrigger")
stepTrigger.fromState = "EXECUTING"
stepTrigger.toState = "FAILED"
stepTrigger.actions = [myEmailAction]
repository.create(stepTrigger)

env = repository.read("Environments/Dev")
env.triggers = ["Configuration/TriggerOnFailure"]
repository.update(env)
```

CI Reference

Configuration Item Overview

Other Configuration Items

CI	Description
mail.SmtplibServer	SMTP Mail Server Configuration
trigger.EmailNotification	Email Action
trigger.StepTrigger	Defines actions to executed for the specified state transition of a Step
trigger.TaskTrigger	Defines actions to executed for the specified state transition of a Task
trigger.Trigger	Trigger with associated actions

Configuration Item Details

mail.SmtplibServer

Type Hierarchy udm.BaseConfigurationItem

Interfaces udm.ConfigurationItem

SMTP Mail Server Configuration

Public Properties	
* fromAddress : STRING	Default from address to use for messages sent with this server.
* host : STRING	SMTP host
* port : INTEGER = 25	SMTP port
password : STRING	Password to authenticate with host
smtpProperties : MAP_STRING_STRING	Refer to http://javamail.kenai.com/nonav/javadocs/com/sun/mail/smtp/package-summary.html for all properties that can be used.
username : STRING	Username to authenticate with host

trigger.EmailNotification

Type Hierarchy udm.BaseConfigurationItem

Interfaces trigger.Action, udm.ConfigurationItem

Email Action

Public Properties
* mailServer : <code>CI<mail.SmtpServer ></code> The mail server used to send the email.
* subject : <code>STRING</code> Mail subject
* toAddresses : <code>LIST_OF_STRING</code> Mail addresses of recipients.
body : <code>STRING</code> Mail body content in the form of a Freemarker template.
bodyTemplatePath : <code>STRING</code> Freemarker template used to render mail body content. Path can be absolute or relative to Deployit's classpath.
fromAddress : <code>STRING</code> From mail address. Defaults to SMTPServer fromAddress.

trigger.StepTrigger

Type Hierarchy [trigger.Trigger](#) >> `udm.BaseConfigurationItem`

Interfaces `udm.ConfigurationItem`

Defines actions to executed for the specified state transition of a Step.

Public Properties
* actions : <code>LIST_OF_CI<trigger.Action></code> Actions to execute when specified state transition occurs.
* fromState : <code>ENUM [ANY, PENDING, EXECUTING, DONE, FAILED, PAUSED, SKIPPED] = ANY</code> Trigger actions when the Step transitions from this state.
* toState : <code>ENUM [ANY, PENDING, EXECUTING, DONE, FAILED, PAUSED, SKIPPED]</code> Trigger actions when the Step transitions to this state.

trigger.TaskTrigger

Type Hierarchy [trigger.Trigger](#) >> `udm.BaseConfigurationItem`

Interfaces `udm.ConfigurationItem`

Defines actions to executed for the specified state transition of a Task.

Public Properties
* actions : <code>LIST_OF_CI<trigger.Action></code> Actions to execute when specified state transition occurs.
* fromState : <code>ENUM [ANY, QUEUED, PENDING, EXECUTING, DONE, STOPPED, EXECUTED, CANCELLED] = ANY</code> Trigger actions when the Task transitions from this state.
* toState : <code>ENUM [ANY, QUEUED, PENDING, EXECUTING, DONE, STOPPED, EXECUTED, CANCELLED]</code> Trigger actions when the Task transitions to this state.

trigger.Trigger

Virtual Type

Type Hierarchy `udm.BaseConfigurationItem`

Interfaces `udm.ConfigurationItem`

Trigger with associated actions.

Public Properties
* actions : <code>LIST_OF_CI<trigger.Action></code> Actions to execute when specified state transition occurs.