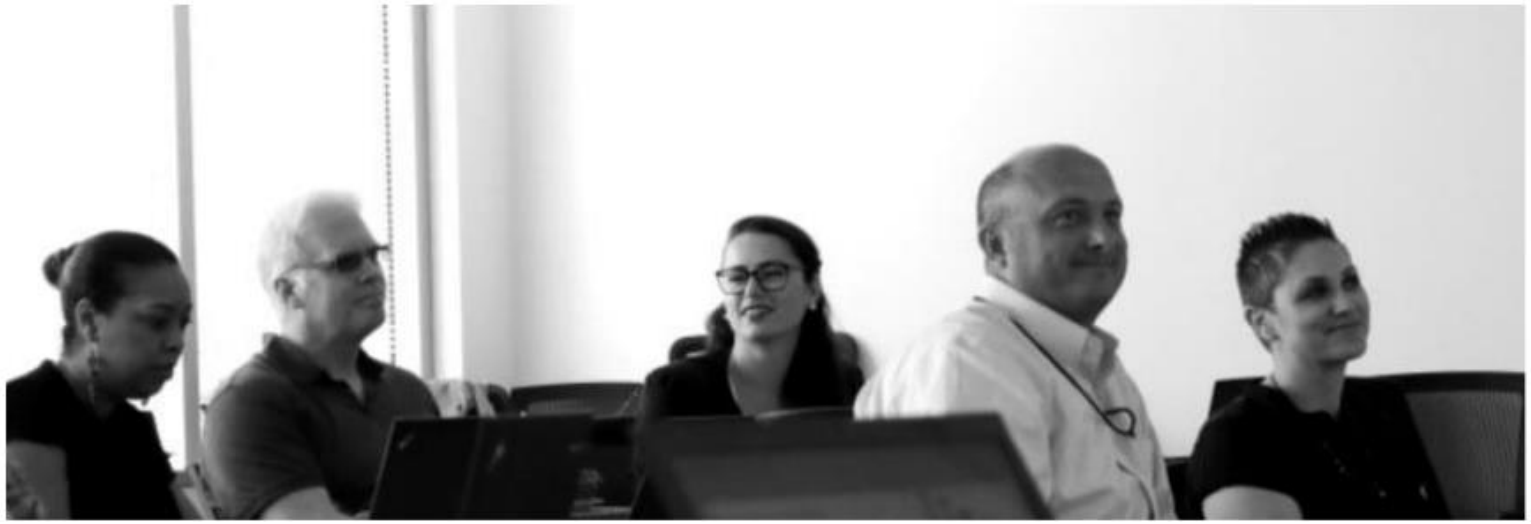
THE JOB LIFECYCLE

Source : Guidewire Educational Platform

The Job Lifecycle



Source : Guidewire Educational Platform

Lesson objectives

By the end of this lesson, you will be able to:

- Describe how the special job components function during the lifecycle of a job
- Configure a job process

Source : Guidewire Educational Platform

Special job components

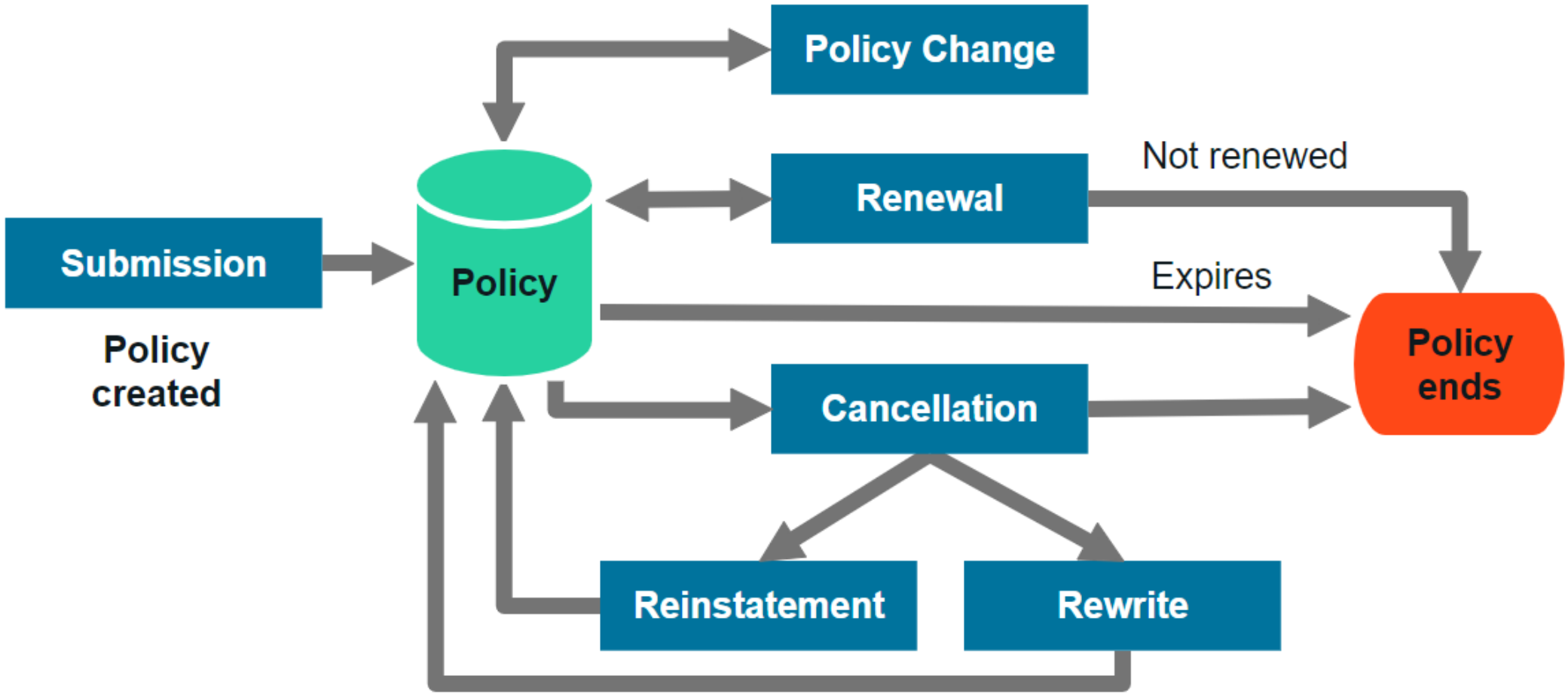
Source : Guidewire Educational Platform

Review: Jobs

- **Policy Transactions** are represented in PolicyCenter by Jobs
  - "Policy Transaction" and "Transaction" are used in the user interface
  - "Job" is used throughout the configuration
- Jobs coordinate all of the work associated with creating a new policy and modifying the policy

Source : Guidewire Educational Platform

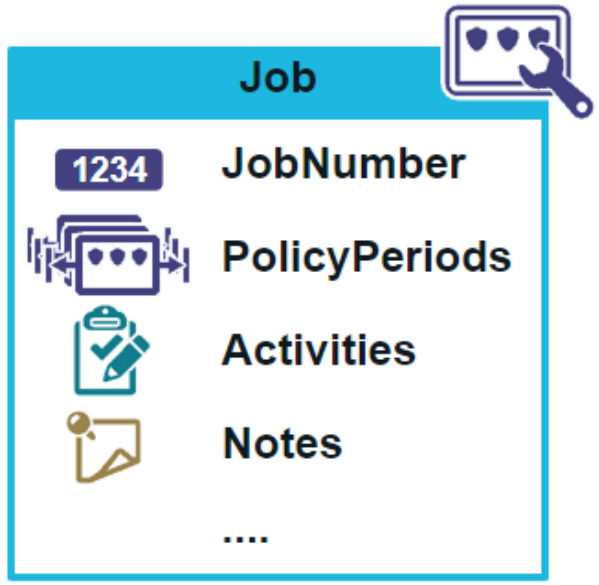
Review: Policy transactions



Source : Guidewire Educational Platform

<Jobtype> entities

- **Job** is the parent entity
  - Subtypes include Submission, PolicyChange, Cancellation, Renewal, Audit, etc.
- <Jobtype> object
  - Is instantiated when the job is started
  - Executes a transaction on the policy
  - Contains required data
  - Can be enhanced or extended

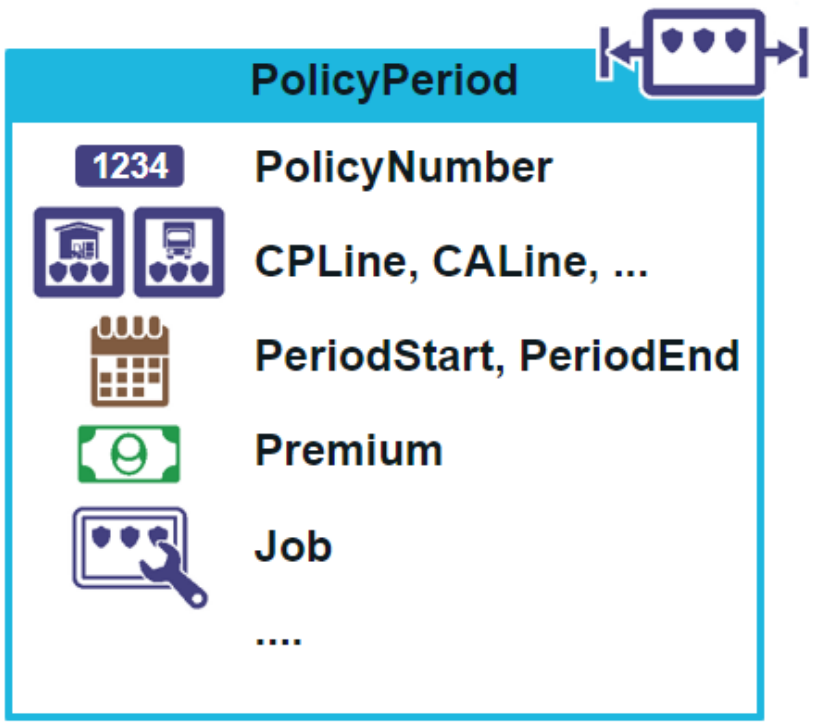


Submission	Cancellation	Audit
SubmissionDate	CancelProcessDate	PaymentReceived
RejectReason	NotificationDate	AuditInformation
QuoteType	CancelReasonCode	

Source : Guidewire Educational Platform

Branches

- A **branch** is a set of objects related to one PolicyPeriod object
- A branch is one version of the Policy
- A branch is at the core of a job
  - The job creates a branch

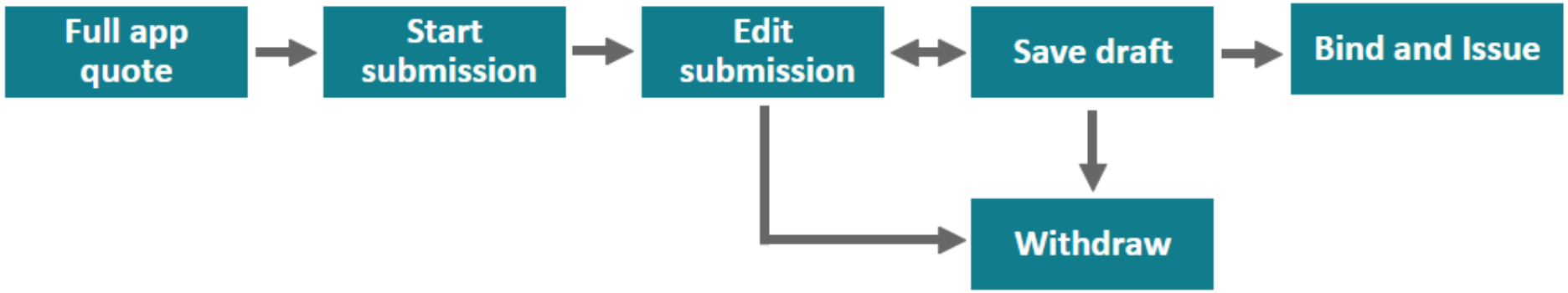


Branch

Source : Guidewire Educational Platform

Job processes

Submission process snippet



- A **job process** controls the flow of a job by using:
  - Gosu classes such as SubmissionProcess.gs
  - Wizard actions that can directly interact with the job and policy revisions
  - Workflows
- Job processes are flexible for user-driven logic

Source : Guidewire Educational Platform

<Jobtype>Process classes

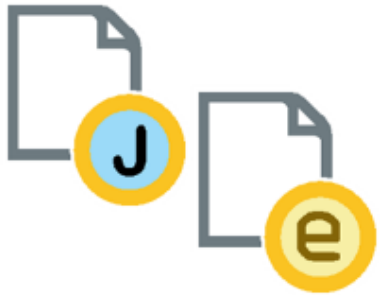
- Contain methods required for processing the job
  - Sub classes of JobProcess include SubmissionProcess, RenewalProcess, AuditProcess, CancellationProcess, etc.
- In Studio, **configuration → gsrc → gw → job**
- Instantiated when methods are required
  - Maintained in cache

Source : Guidewire Educational Platform

<Jobtype> wizards

- **<Jobtype> Wizards** contain screens, tools, and buttons that help the user step through the life cycle of the job

- **Job Wizard Helper**
  - Is a Java class that provides functionality for job wizards, such as
    - Movement between steps
    - Visibility based on state of job
    - Helper methods for adding messages
  - Has a Gosu enhancement that can be edited to provide additional functionality

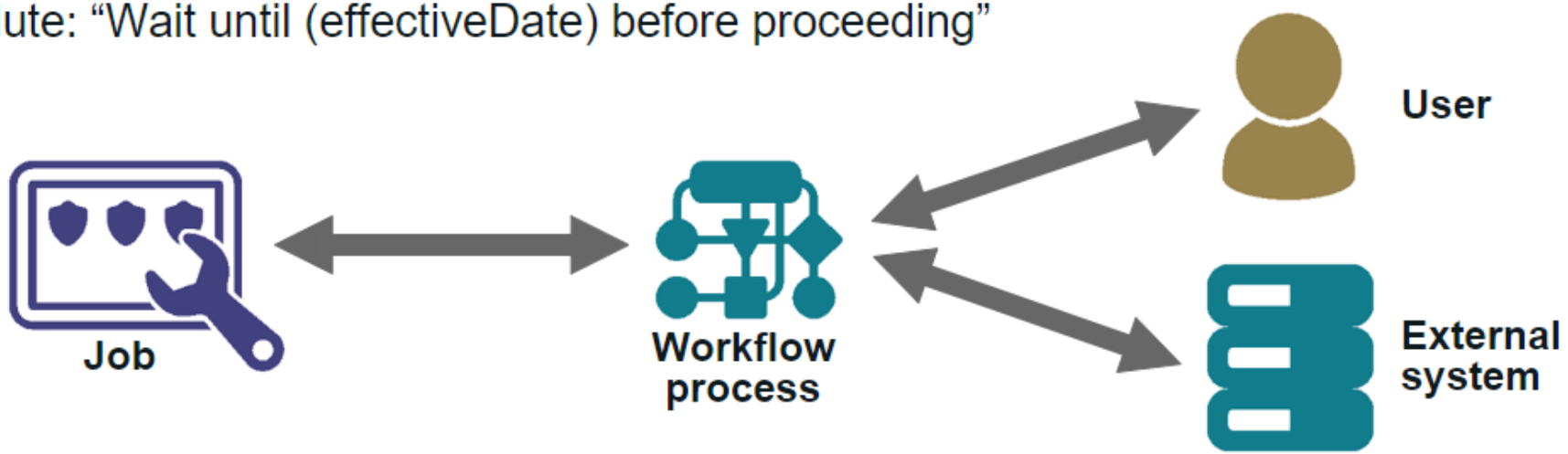


- Each job type has a **Job Wizard Button Set** containing relevant buttons
  - Buttons call Gosu code to perform the work of the job
  - Buttons appear in the job wizard's screens only if they are currently relevant

Source : Guidewire Educational Platform

Workflows

- Integration with users and external systems
  - “Wait for user to respond before proceeding”
  - “Wait for billing system to respond before proceeding”
- Time-delayed actions
  - Relative: “Wait three weeks before proceeding”
  - Absolute: “Wait until (effectiveDate) before proceeding”



Source : Guidewire Educational Platform

Job lifecycle

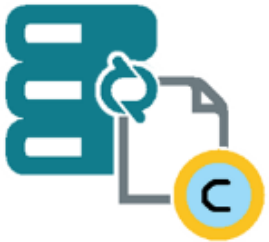
Source : Guidewire Educational Platform

Starting a job

Jobs can be started in three ways

- By users through the Actions menu

- By external systems through APIs



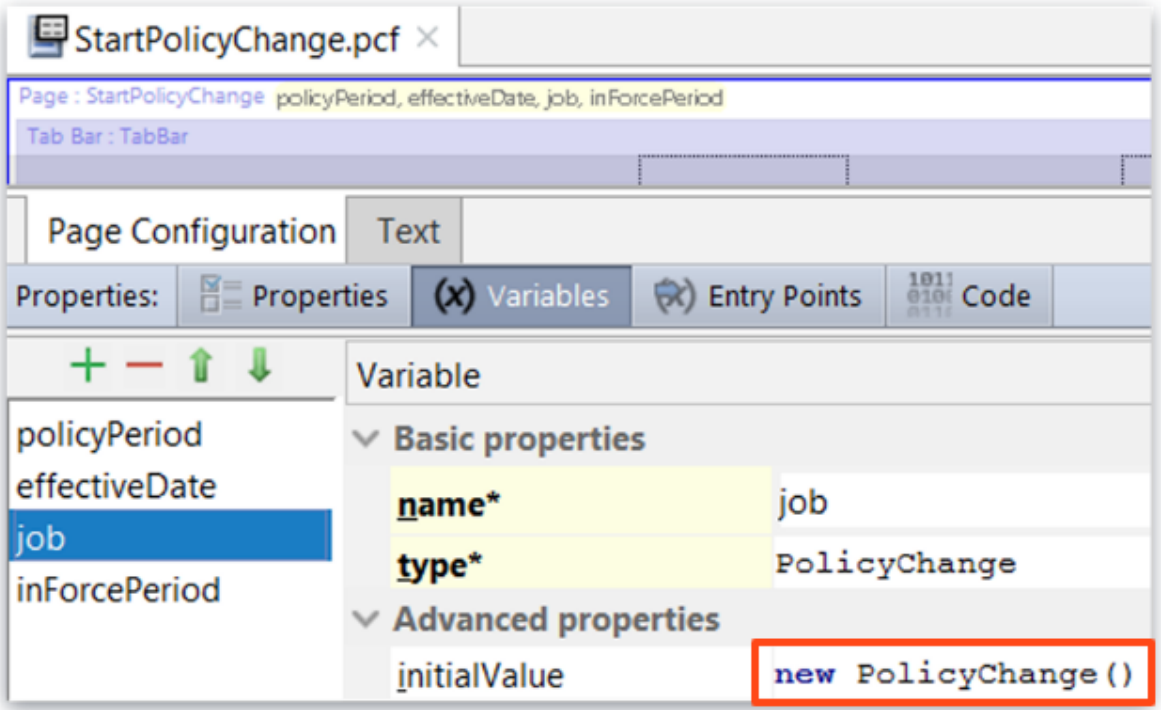
- Through PolicyCenter batch processes

Source : Guidewire Educational Platform

Instantiating the <Jobtype> object

The <Jobtype> object needs to be instantiated

- Typically on the screen that calls the job wizard and is passed as a parameter



Source : Guidewire Educational Platform

Checking job conditions

- Preconditions are checked before a job function is performed
  - Button visibility in the job wizards is based on *canXXX* methods in job process classes
    - For example, in JobWizardToolBarButtonSet.Submission.pcf
    - The Edit Policy Transaction button's visible property calls JobProcess.*canEdit*().Okay()
- **Can** methods return a JobConditions object
- **JobConditions** object Checks for error conditions
  - Contains a string builder called "_messages" which begins empty and is built by error condition checks
  - If "_messages" remains empty, derived property "Okay" returns true, otherwise, false

Source : Guidewire Educational Platform

Executing business logic

- Job wizard buttons call methods
  - They can call methods on any object available in the current context
  - Such as <Jobtype>Process, <Jobtype>Wizard, JobWizardHelper, PolicyPeriod
- Job process methods call other methods
- Job process methods call workflows
- Workflows call job process methods

Source : Guidewire Educational Platform

End of a job

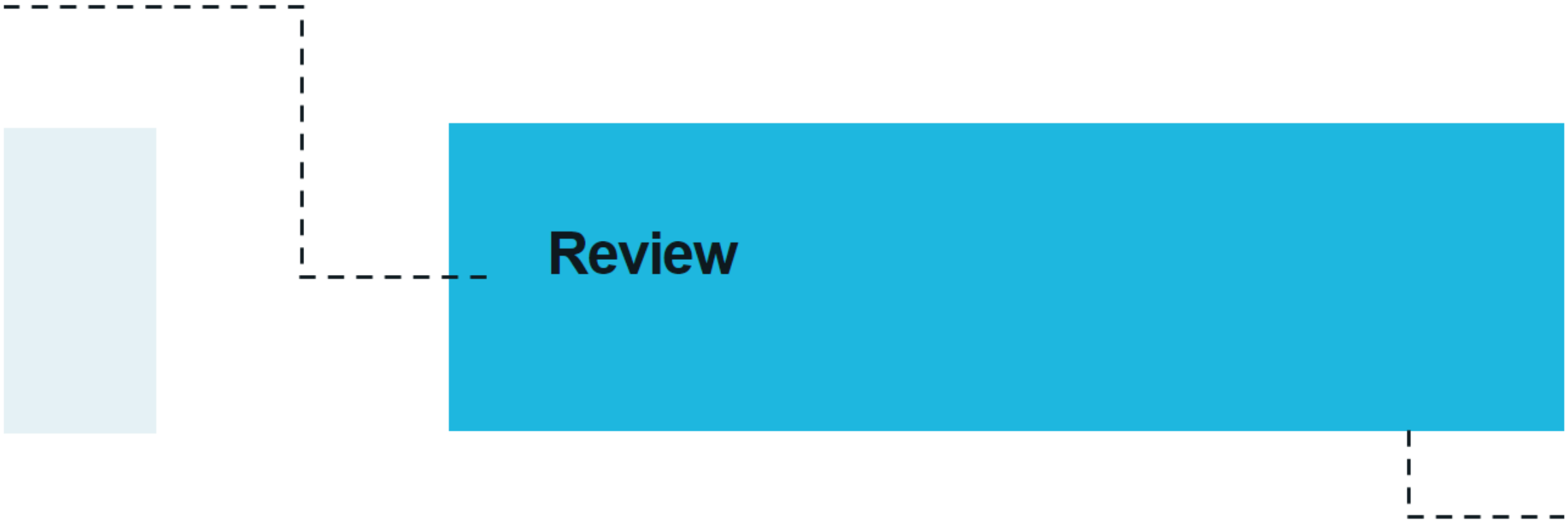
- Job ends when the branch is
  - Promoted (bound)
  - Withdrawn
  - Discarded
- Job object remains in the database, where it can be referenced
  - It can no longer be edited
- Job process object is discarded

Source : Guidewire Educational Platform

Configuring job processes

1. Extend the <Jobtype> related classes and PCF files, for example
   - JobEnhancement.gs, SubmissionEnhancement.gs, SubmissionProcess.gs
   - SubmissionWizard.pcf and related screens
2. Configure the business logic
   a) Edit existing functions to capture your business process
   b) Create additional functions as needed

Source : Guidewire Educational Platform



Source : Guidewire Educational Platform

Lesson objectives review

You are now able to:

- Describe how the special job components function during the lifecycle of a job
- Configure a job process

Source : Guidewire Educational Platform

Capgemini