

PEGA PLATFORM



System Architect Essentials

7.3.1

Student Guide

© 2018
Pegasystems Inc., Cambridge, MA
All rights reserved.

Trademarks

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.
One Rogers Street
Cambridge, MA 02142-1209
USA
Phone: 617-374-9600
Fax: (617) 374-9620
www.pega.com

File Name: File name

Date: June 2018

COURSE INTRODUCTION	1
Before you begin	2
System Architect Essentials overview	3
INTRODUCTION TO PEGA PLATFORM	5
Pega Platform	6
Introduction to Pega Platform	6
The magic of Pega	7
Pega Platform overview	8
The roles of team members	9
Capture objectives directly in the application	9
Configuring a Pega Platform application	10
Introduction to configuring a Pega Platform application	10
Best practices and application guardrails	11
Introduction to best practices and guardrails	11
Purpose of best practices	11
Pega's best practices for project success	12
Guardrails for application design	16
DESIGNING A CASE LIFE CYCLE	17
Designing a case life cycle	18
Introduction to designing a case life cycle	18
Case life cycle design	18
Stages	20
Stage transitions	22
Controlling stage transitions	24
Processes	26
Adding optional actions	28
Introduction to adding optional actions	28
Optional actions	28
Adding optional actions to a case type	30
Guiding users through a case life cycle	32
Introduction to guiding users through a case life cycle	32
Updating the case status	32
Adding instructions	34
Defining user views	35
Introduction to defining user views	35
User view planning	35
Guidelines for designing user views	37
Data elements in Pega applications	40
Configuring user views	44
Validating case data	49

Introduction to Validating case data	49
Methods of data validation	50
How to validate user data with controls	52
How to validate case data with validate rules	58
Validating a flow action with a validate rule	60
How to use edit validate rules	65
Managing case-processing dependencies	67
Introduction to managing case-processing dependencies	67
Case relationships	68
How to enforce a dependency between case types	70
Adding a child case to a case	72
MODELING CASE DATA	74
The building blocks of a Pega application	75
Introduction to the building blocks of a Pega application	75
Rules and rule types	76
Rules and rulesets	77
Classes and class hierarchy	79
How to create a rule	81
How to update a rule	82
How to reuse rules through inheritance	84
Data elements in Pega applications	87
Introduction to Data Elements in Pega Applications	87
How to manage properties	88
How to reference a property	93
Defining properties	94
Reviewing application data	97
Introduction to reviewing application data	97
Data storage in memory	97
pyWorkPage	98
How to view clipboard data	100
Setting property values automatically	102
Introduction to Setting Property Values Automatically	102
Data transforms	103
How to set values with data transforms	105
How to set default property values	109
Data transforms and superclassing	112
Setting property values declaratively	116
Introduction to setting property values declaratively	116
Declarative and procedural processing	117
Declare expressions	120
How to set a property value with a declare expression	123
Forward and backward chaining in declarative networks	125
Configuring a work party	128
Introduction to configuring a work party	128

Work parties	129
How to add a work party to a case	131
Configuring a work party for a case type	135
System tasks	136
How to add system tasks to process flows	138
Exchanging data between cases	142
Introduction to exchanging data between cases	142
Data propagation	143
Propagating data when creating a case	145
Updating data for an existing case	149
Calculating case values from child case data	151
Caching data with data pages	153
Introduction to caching data with a data page	153
Data pages	154
How to configure a data page	156
Configuring a data page	161
Managing reference data	165
Introduction to managing reference data	165
How to use reference data	166
Defining reference data for a data type	169
Using the external database mapping wizard	171
AUTOMATING BUSINESS POLICIES	173
Configuring a service level agreement	174
Introduction to configuring a service level agreement	175
Service level rules in Pega Platform	176
Adding a service level to a case	178
Assignment urgency	180
Configuring a service level agreement rule	183
Configuring and sending correspondence	187
Introduction to configuring and sending correspondence	188
Automating case communications	189
Sending an email from a case	191
How to configure correspondence	192
Routing assignments	196
Introduction to Routing Assignments	197
Assignment routing	198
Worklists and work queues	200
Configuring assignment routing	202
Delegating business rules	210
Introduction to delegating business rules	211
Business rule delegation	212
Best practices for business rule delegation	213
How to delegate rules to business users	215
Controlling the flow of a case life cycle	218

Introduction to controlling the flow of a case life cycle	219
Conditional paths in a case life cycle	220
How to model a decision point in a process	222
How to configuring processing for a case	224
Circumstancing rules	227
Introduction to circumstancing rules	227
Situational processing	228
How to circumstance rules	230
Circumstancing a rule	233
DESIGNING A USER INTERFACE	236
Configuring a user form	237
Introduction to configuring a user form	237
Section rules	237
How to configure a section	240
How to configure responsive UI behavior	244
Creating dynamic content in user views	248
Introduction to creating dynamic content in user views	248
Dynamic UI behavior	249
How to configure field attributes for dynamic display	252
Action sets	256
How to configure an action set for a field	258
DESIGNING BUSINESS REPORTS	261
Creating business reports	262
Introduction to creating reports	262
The role of reports	263
Business and process reports	267
The Report Browser	270
How to create a report	273
How to organize report results	278
Creating a report	281
Organizing report results	284
Optimizing report data	289
Introduction to optimizing report data	289
Data Storage in Pega applications	290
Property optimization	291
Optimizing properties for reporting	293
TESTING AND DEBUGGING APPLICATIONS	294
Unit testing application rules	295
Introduction to unit testing application rules	295
Unit testing	296
How to unit test a rule	298
How to record a unit test for automated testing	301

Debugging application errors **303**
Introduction to debugging Pega applications 303
The Tracer 304
How to investigate application errors with the Tracer 305

COURSE SUMMARY **307**

Course summary **308**
System Architect Essentials summary 308

COURSE INTRODUCTION

Before you begin

System Architect Essentials overview

Learn the core principles of application development on Pega Platform to prepare yourself for your first Pega development project. Business users and delivery team members use these principles to plan and deliver business applications faster and more accurately for maximum business value.

Objectives

After completing this course, you should be able to:

- Explain the benefits of using the Pega model-driven application design and development approach
- Model the life cycle of a case that mirrors the way business people think about how work is completed
- Identify the high-level responsibilities associated with Pega Platform for both Pega business architects and system architects
- Describe Pega's Direct Capture of Objectives™ approach to increasing the speed and accuracy of application delivery
- Explain the purpose and benefits of best practices and guardrails
- Validate case data to ensure that user entries match required patterns
- Configure a Wait shape to enforce a case processing dependency
- Configure user views and data elements during case life cycle creation
- Use the Clipboard tool to review case data in memory
- Set property values automatically using data transforms and declare expressions
- Configure and populate a work party with case data
- Create data classes and properties for use in a Pega application
- Automate decision-making to improve process efficiency
- Design responsive user forms for use on any platform or browser
- Design reports to deliver key insights to business users
- Incorporate and manage reference data to allow applications to adapt to changing business conditions
- Test your application design to analyze rule behavior and identify configuration errors

Intended audience

This course is for system architects who are responsible for developing business applications.

Prerequisites

To succeed in this course, students should have completed:

Some experience developing software applications is helpful, but not required.

INTRODUCTION TO PEGA PLATFORM

Pega Platform

Introduction to Pega Platform

Pega Platform is a visually driven application development environment that enables you to deliver applications faster than traditional approaches. People in business and IT roles work together, using visual models to capture business requirements at any time.

After this lesson, you should be able to:

- Explain the benefits of using a model-driven application design and development approach
- Identify the key navigational elements of the Pega Express portal
- Describe the roles of a Business Architect and a System Architect
- Describe Pega's Direct Capture of Objectives™ approach to increasing the speed and accuracy of application delivery
- Identify the high-level responsibilities associated with Pega Platform for both Pega business architects and system architects

The magic of Pega

In the following video, you learn about the magic of Pega: how it empowers your business; how you can leverage Pega smart technology to do the work for you; how Pega uses Software That Writes Your Software™; how you run your application where it makes sense for your organization; and how Pega's application offerings jump start your digital transformation journey. All through the magic of Pega.

Before you learn the technology, take a few minutes to experience the magic of Pega. The Pega mission is to change the way the world builds software to create unprecedented business outcomes by connecting, engaging, and empowering people. How do you empower employees to create software? The answer is to get rid of code. Instead of coding apps, business users connect with IT to configure applications using visual tools. This model-driven approach is faster than code and more accurate. This approach engages everyone — business people and IT — using software to transform your business. Pega empowers business people to create and evolve the systems they use every day.

How do you use Pega to create this magic? You use Pega's differentiated capabilities and products.

- **Empower the business.** Pega magic happens when business people use tools that allow them to create and evolve the applications they need to successfully run their businesses.
- **Systems do the work.** Pega magic happens when systems do the work, not people. The systems should be *smart* and leverage Pega technology such as decisioning, robotics, and AI to do the work for you.
- **Let the software write the software.** Pega magic happens through software that writes your software. Business users focus on building value into application using model driven development, not writing and managing Java code.
- **Run your application where it makes sense for your organization.** Magic happens when Pega applications run anywhere — Cloud, on premise, and in topologies such as Pivotal Cloud Foundry and containers such as Docker. With Pega, you are not tied to a topology or deployment model. As a result, Pega allows your systems to be future-proof.
- **Jump start your transformation with Pega's application offerings.** Pega magic happens when you gain advantage in your digital transformation journey by starting with Pega's customer engagement and industry applications. Pega's prebuilt applications allow you to deliver value to your customers faster than building the application from scratch. However, if you need the flexibility to build a custom application you can use Pega Platform to build whatever type of application you want.

You can start your digital transformation from the center and work your way out using end-to-end automation. You use visual models to automate the user process. Extend your model with Pega's next-best-action decisioning to dynamically meet each customer's needs. When you start with the outcomes that you want to deliver and work out from there, you can radically transform the experience for your customers. You get value, and you have built for reuse across channels and into the future.

Here is one example of using the magic of Pega. To counter deteriorating market share and the highest churn rate in the wireless industry, one mobile communications company needed to overhaul its approach to customer care. In just 90 days, the company transformed its contact center operations across 1,000 service agents, ensuring the right retention offers for each customer. The company saw a 40% increase in Net Promoter Score, an 800% increase in customer upgrades to higher value plans, and a 50% reduction in post-paid churn, to the lowest level in company history. All through the magic of Pega!

Are you ready to learn how to use the magic of Pega?

Pega Platform overview

The Pega Platform overview introduces you to some of the benefits and uses of the Pega Platform. In the following video, you will learn how Pega Platform helps you to automate your complex business processes while accommodating differences in products, regions, and channels. You will also see that Pega Platform future-proofs your application by continually updating with the latest technologies.

Pega Platform transforms the way organizations do business. You use Pega to quickly and easily model your complex business processes. Pega Platform includes built-in tools to ensure that you follow best practices while building your application. By following best practices, you reduce development costs, work more efficiently, and keep pace with today's changing business environment.

Pega uses a set of business-friendly metaphors, an intermediate visual language that enables the computer to really understand how you want things to work. Pega Platform is a model-driven application development tool you use to define your business systems. Rather than traditional programming, Pega's approach to marketing and customer engagement is in terms of specific business cases. Those cases may involve orders, supply chain, or your specific pain-points. Pega's Next-Best-Action predicts customer needs and provides the right offer.

Pega thinks in terms of specific business cases. Your case represents a business transaction you want to complete or resolve. Cases follow stages and steps in a process from beginning to end. Cases can accomplish a variety of tasks across many businesses and sectors. Examples of cases include legal cases, home mortgages, auto insurance claims, and software bug tracking. Your case has at least one process that people work on to resolve the case. And your case has a status that changes as the case completes each stage of the process.

Using Pega Platform you automate your business process by building a case. For example, imagine processing an expense for reimbursement. The work moves from an employee incurring the expense, to a manager reviewing the expense, and then to a higher-level manager approving the expense, and finally to the finance department for payment. Using Pega Platform you create a case that models the business process. The case automates each stage and the steps involved in moving the case to completion.

Here are some examples of Pega Platform automation:

- Communications – including email and text notifications,
- Business policies – including decision logic and service level agreements,
- Routing assignments – including routing work to case participants, and
- Reporting – including running and sending scheduled reports.

Once you configure a case that automates your business process, Pega Platform enables you to reuse policies and procedures throughout your organization. Pega's Situational Layer Cake managed by the Pega brain makes reusing common policies and procedures easy while allowing for differences between products, regions, channels, and customer segments.

The Pega brain enables you to think differently about the connections and think about customer journeys. It's what makes an organization agile, fluid, smart, and responsive. Pega Platform's model-driven development tools enable you to future-proof your enterprise applications. Pega does this by continually updating Pega Platform with the latest technologies that generate new code from the same business-friendly metaphors you create to automate your business processes. Pega Platform enables you to Build for Change®.

The roles of team members

The following video covers the roles of team members during the design, build, and production phases of your Pega Platform application. During the design and build phase, you see how case designers, Business and System Architects, work with subject matter experts and stakeholders to create the application. During the production phase you see how case participants, case workers and case managers, interact to resolve cases.

During the design and build phase of a Pega Platform application, a team of case designers collaborate on the project. Case designers are business architects and system architects, who collaborate with subject matter experts and stakeholders. Subject Matter Experts work with business architects and system architects to design the metaphors that represent your business processes. Stakeholders help to review, refine, and approve the solution. When your Pega Platform application moves to production, case participants, consisting of case workers and managers, use the application to resolve your business cases.

Business architects are members of the application development team who define business rules, service-level agreements, and processes. They are advocates for business users. Business architects work with subject-matter experts and key project stakeholders to understand business needs and define use cases and features.

System architects are application developers who contributes object-oriented design and technical implementation skills to an application development project. System architects configure application assets such as user interface forms and automated decisions.

Here is an example of a candidate interview and hiring process that illustrates how case workers and managers interact with the case. In production, case workers schedule phone screenings and interviews, and prepare and extend offers. Case managers evaluate and assess the candidates and approve offers. Pega Platform automates the business process and ensures case participants work together to resolve cases.

Capture objectives directly in the application

Business stakeholders can find it difficult to communicate business goals to IT stakeholders. The stakeholder groups may lack a common language or view of business goals. If business stakeholders are unsure of their business aims, IT stakeholders find it difficult to get the details they need. Business and IT stakeholders must share a common understanding of the business requirements.

In Pega Platform, business and IT stakeholders use a shared visual model to capture business requirements directly in the application. Pega calls this practice Direct Capture of Objectives, or DCO.

DCO increases both the speed and accuracy of application delivery.

Configuring a Pega Platform application

Introduction to configuring a Pega Platform application

The Pega Platform provides a unified platform with everything you need to build or modify an enterprise application. To improve application quality and maintainability, the Pega Platform includes built-in tools that ensure that you follow best practices for application development.

After this lesson, you should be able to:

- Identify the key Pega Express features you use to create and build your application
- Create a case type
- Create a field to store case data
- Configure a user view to display case data

Best practices and application guardrails

Introduction to best practices and guardrails

In this lesson, you will learn best practices for designing and building applications.

Some best practices help you deliver business application development projects successfully. Other best practices help you design business applications with fewer defects. Following best practices increases your chances of overall project success.

Objectives

At the end of this lesson, you should be able to:

- Explain the purpose and benefits of best practices.
- Identify Pega best practices
- Identify the most important best practices when building a Pega application
- Explain how Pega guardrails indicate flaws in an application design
- Describe the information provided in guardrail warnings

Purpose of best practices

Best practices are well-defined methods or techniques that lead to desired results. Every best practice has at least one goal. A best practice is proven to make progress towards achieving its goals. If an organization follows best practices, it can predict a desired result with minimal problems or complications.



Best practices are used in everyday life. For example, when preparing a meal, a best practice is to always use a potholder when touching a hot pot. The goal is to not burn yourself, and using a potholder is a proven way to prevent you from burning yourself.

As you design and build your application, establish and follow best practices to increase your chances of project success. Look for existing best practices used by your organization and by Pega. Select existing

best practices or create new best practices to meet your goals. Applying best practices increases the likelihood that your application is delivered on time and meets all of its design requirements.

Tips for establishing best practices

The standards for establishing best practices can vary, depending upon the needs of the organization and who is making the choices. You can use several criteria to help choose best practices that will deliver measurable and predictable performance improvements for your application development projects.

Is the best practice appropriate for your organizational goals?

To be effective, best practices must address the specific goals of your organization. For example, if an organizational goal is to ensure the protection of sensitive customer information, a best practice to outsource developers may not be appropriate — the developers may also be working for the organization's competitors.

Does the best practice fit with the structure of your organization?

If a best practice places authority in a single person or part of the organization, it does not provide value if your project teams are supposed to be able to make their own decisions.

Do you have the necessary resources to use the best practice?

Understanding what a particular best practice requires in regard to resources — whether money, personnel, or skills — is essential. Make sure that your organization can provide those resources before committing to a specific best practice.

Is the best practice cost-effective?

If a best practice works well for other organizations but requires an unacceptable amount of money or time to reproduce in your organization, it will be hard to justify the use of that best practice.

Pega's best practices for project success

With experience from thousands of Pega project implementations, Pega has defined best practices that are key to delivering successful Pega projects. Using some of the best practices identified below can help increase your chances of project success.

Leverage DCO to improve product quality

Directly Capture Objectives (DCO) enables a project team to directly enter business requirements for an application into Pega.

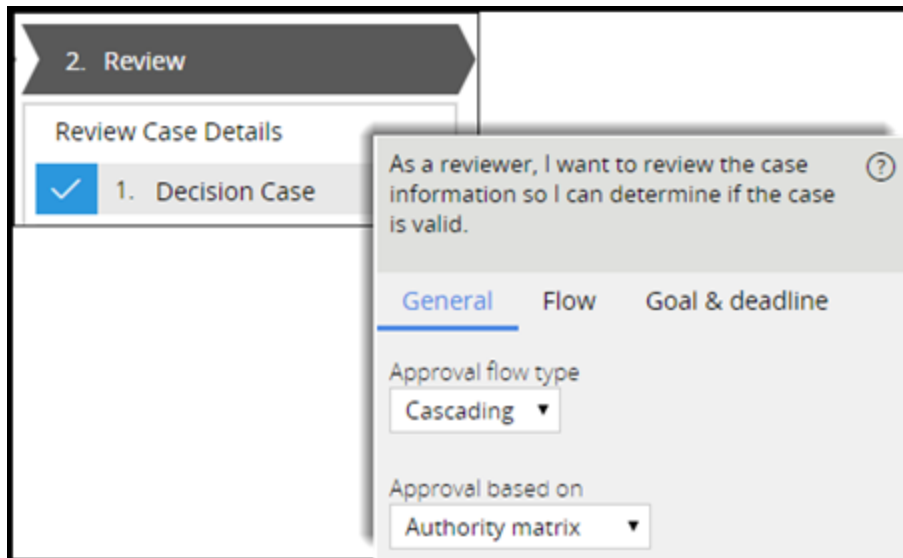
DCO helps eliminate translation errors, saves the team time and effort, facilitates direct engagement of business and IT resources around visible working models, and enables project participants to optimally review work progress.

Pega recommends that all projects leverage DCO as a core part of the delivery process.

Use standard Pega capabilities

Pega 7 has many features and capabilities built into the product. Use Pega capabilities, which have been tested and proven reliable.

For example, assume part of a case's life cycle requires increasing levels of review. Rather than building a custom review process, use an approval process available in Pega 7. You can design the life cycle of a case to support your requirements in a fraction of the time needed to build custom processes.



Iterate and test as you build

Use the most agile, iterative delivery model that your organization can adopt. First, separate large applications into smaller, more manageable components. For example, instead of building the complete application and then testing the completed application all at once, build and test individual processes incrementally. Then, demonstrate completed features to interested parties who can provide feedback.



Begin testing early in the project life cycle to drive higher levels of product quality. Test each new feature or capability to make sure it works. Then, test the system for processing issues that may affect performance. Also, check to see if the new features work together without error. Finally, have analysts test the application to make sure the application meets the requirements and business objectives.

Communicate project progress at all levels

Regular communication among all project participants helps teams focus on the right issues in a timely manner. Pega recommends the following:

- Daily standup meetings to set priorities for the day, ensure alignment, and eliminate any blockers.
- Weekly project updates to track issues and update the status report. Flag conditions that need immediate attention to keep projects moving on schedule.
- Biweekly or monthly meetings to review the entire project and determine whether the application conforms to the original design objectives. Promptly incorporate feedback into project design, and revise project plan if necessary.

Ensure project team members are certified

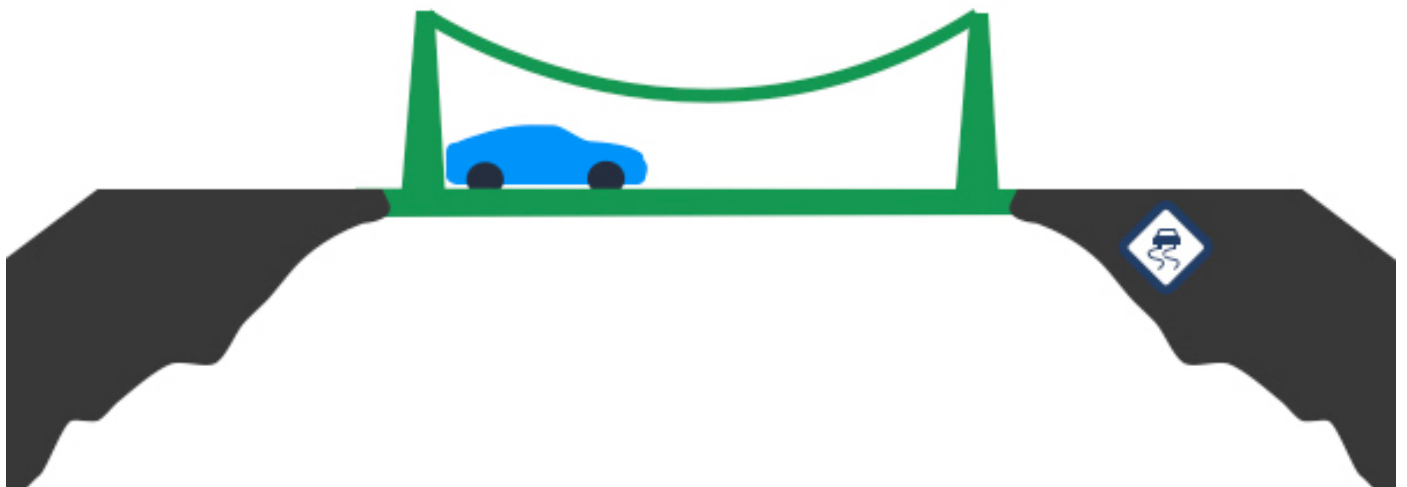
Project success depends on a complete and capable team. As a guideline, Pega recommends that all team members hold the appropriate certifications for their roles.



Pega recommends that business architect and project management resources pass the Certified Business Architect exam. All developers should, at a minimum, pass the Certified System Architect exam.

Follow Pega guardrails

Pega guardrails help ensure that you use best practices for configuring Pega applications. Pega's powerful capabilities offer many possible approaches to create a specific design requirement. Not all of those approaches are the best solution. Pega provides standard guardrail capabilities that enable development team members to track compliance with Pega best practices. Compliance with the guardrails results in applications that are easier to maintain and upgrade, and have significantly fewer defects than non-compliant applications.



Collaborate with everyone invested in the success of the project

Collaborate with individuals interested in making your project succeed. Bring business users, business analysts, and system architects together so they can share their unique skills and viewpoints. For example, business users and business analysts who have in-depth knowledge of the business process can help capture accurate requirements as well as design and optimize business processes. System architects can help provide guidance on the best way to implement a business requirement.

KNOWLEDGE CHECK



What are the possible consequences of not following best practices and guardrails when designing and building an application on the Pega 7 platform?

When not following best practices, you may spend more time re-creating existing functionality, or debugging a component that is not well designed. There is also the risk that you will implement a feature that does not work correctly.

Guardrails for application design

Following best practices when designing and building an application is important. To help ensure your project's success, Pega guardrails help you and your team track whether an application conforms to Pega's best practices.

Following best practices can help you deliver applications that require less maintenance, have fewer defects, and can be easily upgraded compared to applications that deviate from best practices.

Watch the following video to get a better idea about how guardrails can help you achieve project success, optimal performance, reuse, and maintainability for applications.

As you work with Pega Platform you will see messages.

These messages are important; they are known as guardrails. Guardrails help to ensure that the application you create follows the known best practices for Pega Platform development.

When creating an application it is always a good idea to follow the best practices for whatever technology we are working with. By not following best practices we add increased risk that our application development effort could go off course. The application does not work as well as it could or should. It can also make the application difficult to maintain and update. It is also possible that our application can get out of control and crash the project entirely.

Guardrails are best practices and guidance about situations that contain risky conditions or that might result in an undesirable outcome. Guardrails ensure you and your team use Pega Platform the right way and help you avoid troublesome situations.

DESIGNING A CASE LIFE CYCLE

Designing a case life cycle

Introduction to designing a case life cycle

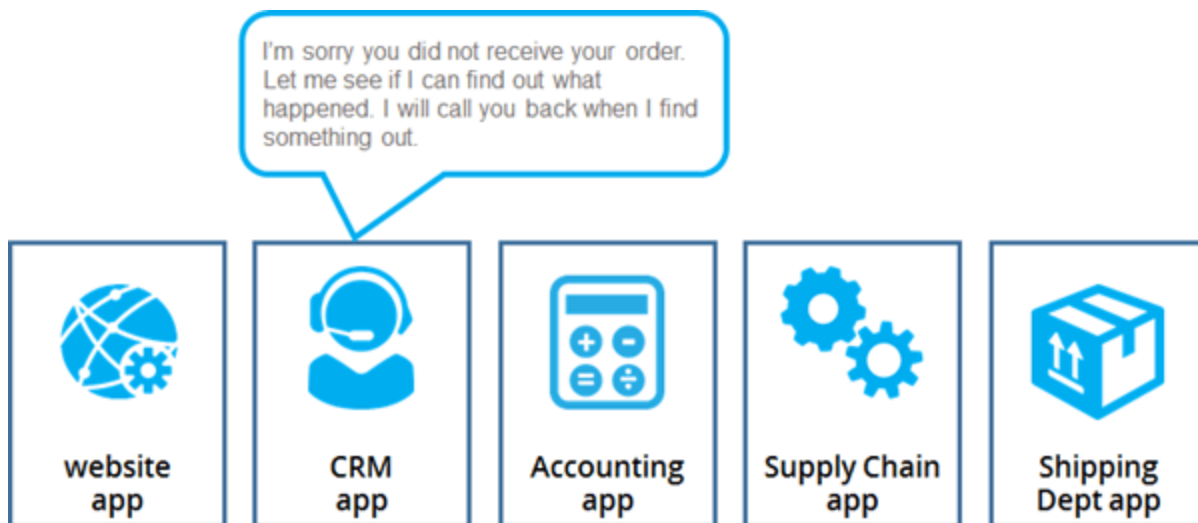
Case life cycle design is a modeling technique used to describe, in business terms, how a business application works. Case life cycle design provides a more natural, business-friendly way to develop an application, and allows business users to interact with a case in the same way they think about it.

After this lesson, you should be able to:

- Explain the relationship between a case and a business outcome
- Explain the purpose of case life cycle design
- Describe the elements of a case life cycle
- Design a case life cycle

Case life cycle design

Business applications are the foundation of every organization. Business applications automate work that must be completed to achieve specific business outcomes. For example, opening a new account, filing an accident claim, or ordering merchandise online.

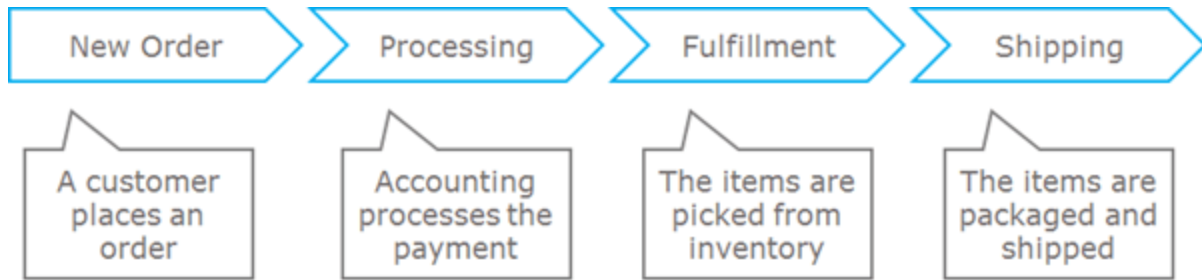


Traditional business applications are based on individual transactions and are built as standalone applications for different departmental functions.

These separate applications make it hard for business users to complete work in a coordinated manner. When working with separate applications, business users lack the visibility they need to effectively achieve business outcomes.

A business view of work

To help business users effectively achieve business outcomes, business users need a way to work in a coordinated manner. Business applications should function in the same way business users naturally think about and describe their work.



Business users often think of the work they do — investigating a fraud claim, processing a loan application or dental claim— as a case.

A **case** is work that delivers a business outcome. The outcome of a case is a meaningful deliverable provided to a customer, partner, or internal stakeholder. A deliverable can be processing a health insurance claim, onboarding a new mortgage, or designing and releasing a new product. In all these examples, the work to be done can be defined in terms of its resolution (the insurance claim is paid, the new account is opened, or the new product is released).

Business users understand, and work with the case as it moves from one person to the other, from one part of the organization to another. What business users are describing is the life cycle of a case — how they manage the case as it is opened, worked on, and resolved.

In a Pega application, you model repeatable business transactions with **case types**. A case type is an abstract model of a business transaction, while a case is a specific instance of the transaction. You can think of a case type as a template for creating and processing cases. When a new transaction starts, a new case is created based on the case type definition.

Each case type captures the life cycle of a specific type of transaction, from creation to resolution. A case type defines data structures, processes, tasks, and user interfaces required for processing the transaction.

KNOWLEDGE CHECK



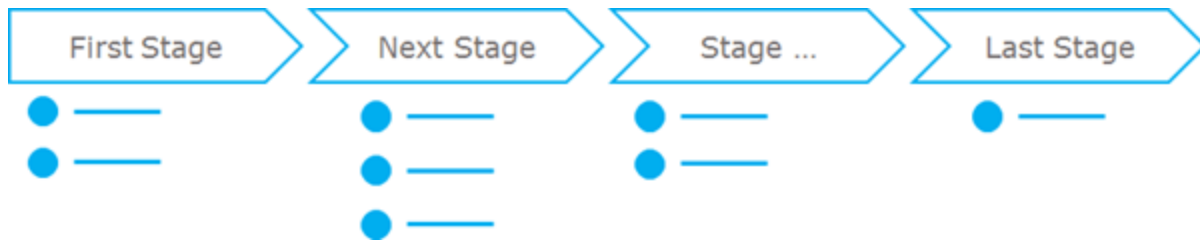
A _____ delivers a meaningful business outcome to a customer, partner, or internal stakeholder.

case

Design your application using a business view of work

A case is the central metaphor in Pega's model-driven approach to application design.

Case life cycle design is a modeling technique Pega uses to describe, in business terms, how a business application should work. Case life cycle design allows business users to see, and interact with a case in the same way they think about it.



Cases are organized into high-level milestones, known as stages. **Stages** are the first level of organizing the different tasks, or processes required to complete work associated with a case. Stages help to organize the life cycle of a case around the key events that support the primary goal of the case.

Stages are further organized into **processes** which define one or more paths the case must follow. Processes contain a series of actions, or steps that must be completed to resolve the case.

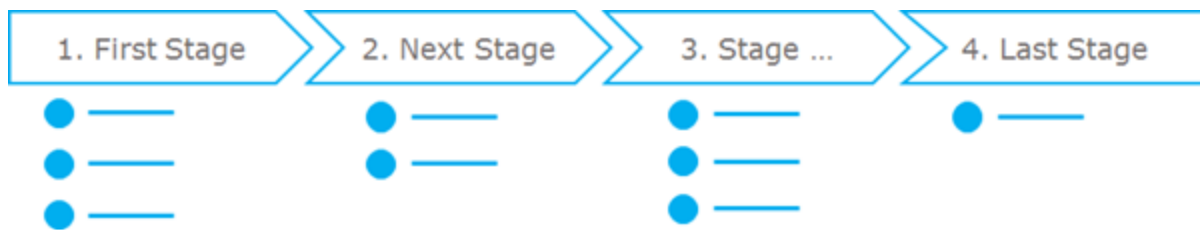
KNOWLEDGE CHECK

The modeling technique used to describe in business terms how a business application should work is called _____.

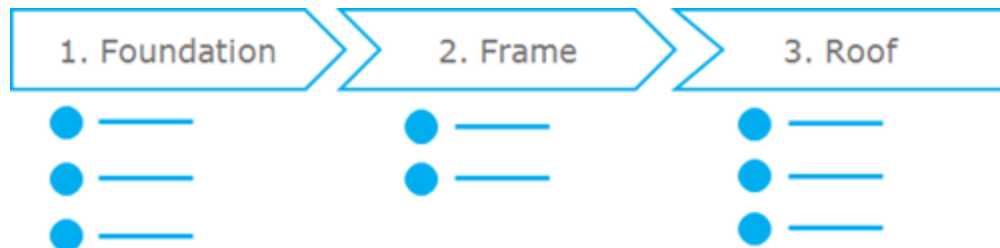
case life cycle design

Stages

A stage is the first level of organizing work in your case. It contains the workflows, or processes, that users follow before they can move a case to the next phase in the case life cycle.



By capturing business requirements in stages, you get a sense of what needs to be done first, what must happen in sequence, and what can happen in parallel during case processing. In a case life cycle, stages that lead to the expected outcome are called **primary stages**. The sequence of primary stages is often referred to as the **happy path**.

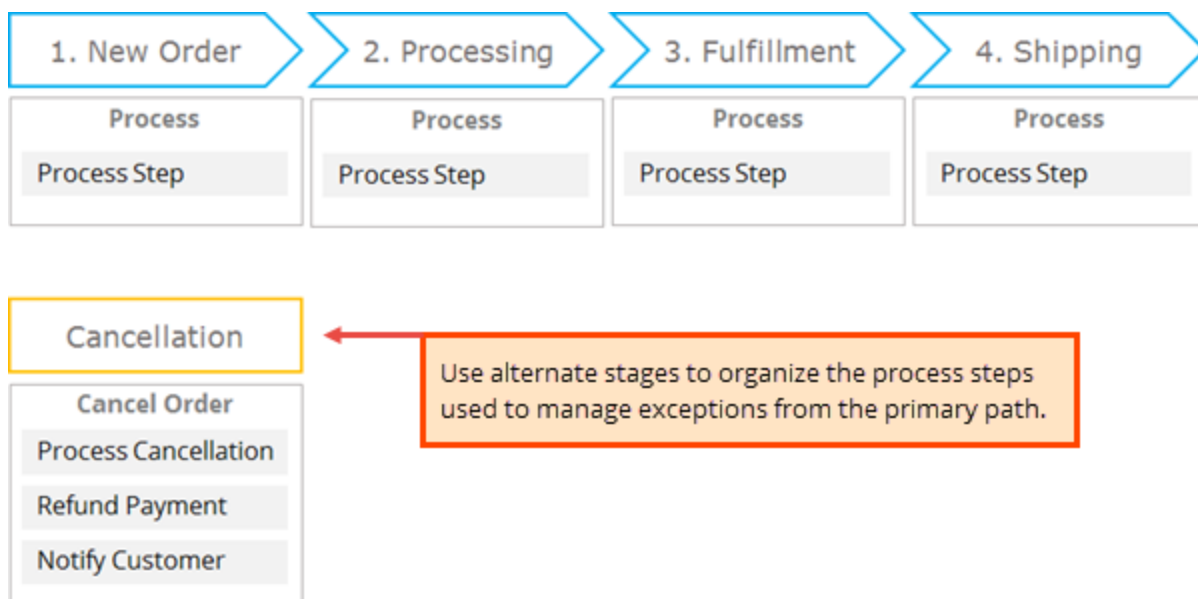


As an example, consider the construction of a new home. If you were asked to organize the tasks required to build a house into key phases of construction, you might organize the tasks in a series of three stages. The foundation of any building is always the first phase. Then the house itself — the frame — is constructed. Finally, the roof is added.

Each stage represents a distinct phase of the home construction case life cycle. In the home construction example, foundation, framing, and roofing are primary stages that lead to the completion of the house which is the business outcome.

Alternate Stages

Cases usually progress in order from one primary stage to the next. In some situations, work does not always go according to plan. When that happens, use an alternate stage to describe the actions needed to resolve the situation. **Alternate stages** are used to organize process steps that are not part of the “normal course of events” but must be available under certain circumstances.



For example, when modeling the life cycle of an online ordering application, you must consider that orders can be canceled prior to being shipped. If an order is canceled, a number of tasks must be completed before the order is considered canceled. The first task is to process the order for cancellation, then the payment must be refunded and, finally the customer must be notified that the order was canceled.

Use alternate stages to organize the process steps used to manage exceptions from the primary path.

Guidelines for defining stages

To define stages, consider the following guidelines.

Organizing stages



Stages typically represent the transfer of the case from one authority to another, or from one part of the organization to another. Stages may also represent a significant change in the status of the case.

Naming stages



Use names that are most meaningful and relevant to the business users. Use a noun, or noun phrase, to describe the context of the stage. As much as possible, try to limit the stage name to no more than two words.

Number of stages



Consider limiting the number of primary stages in any given case to seven and the number of alternate stages to five. If you find yourself needing more than 10 primary stages or more than 5 alternate stages, consider combining two or more stages, or using a separate case type.

Tip: There is no minimum number of stages for a case. Focus on maintaining a number of stages less than the maximum guidelines outlined here.

KNOWLEDGE CHECK

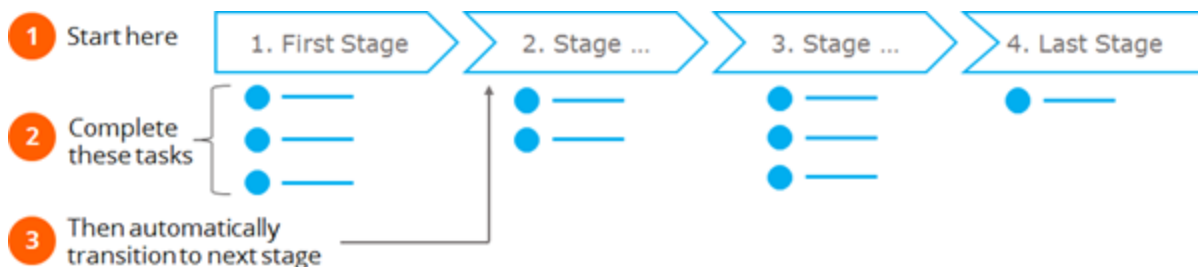


A _____ is used as a first level of organizing work in a case.

stage

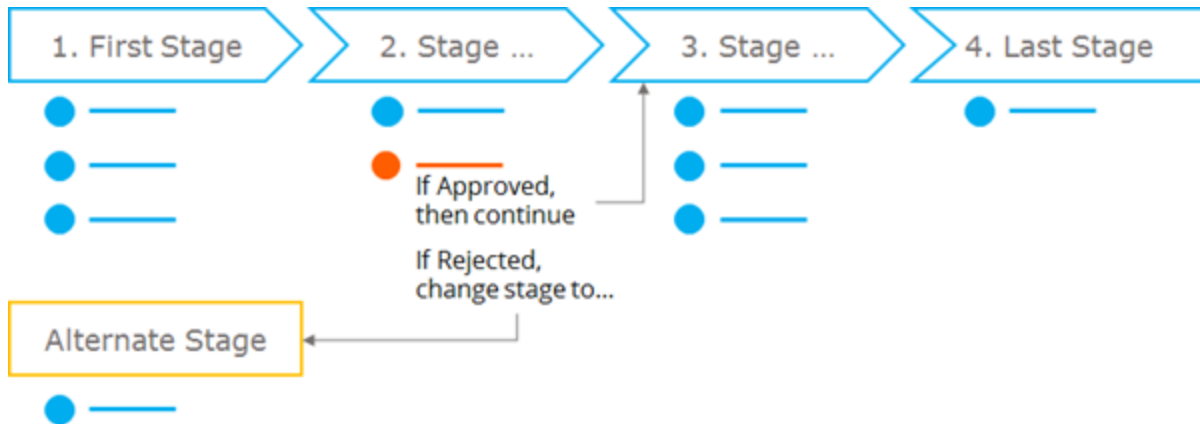
Stage transitions

Stage transitions allow you to further refine the run-time order of stages.

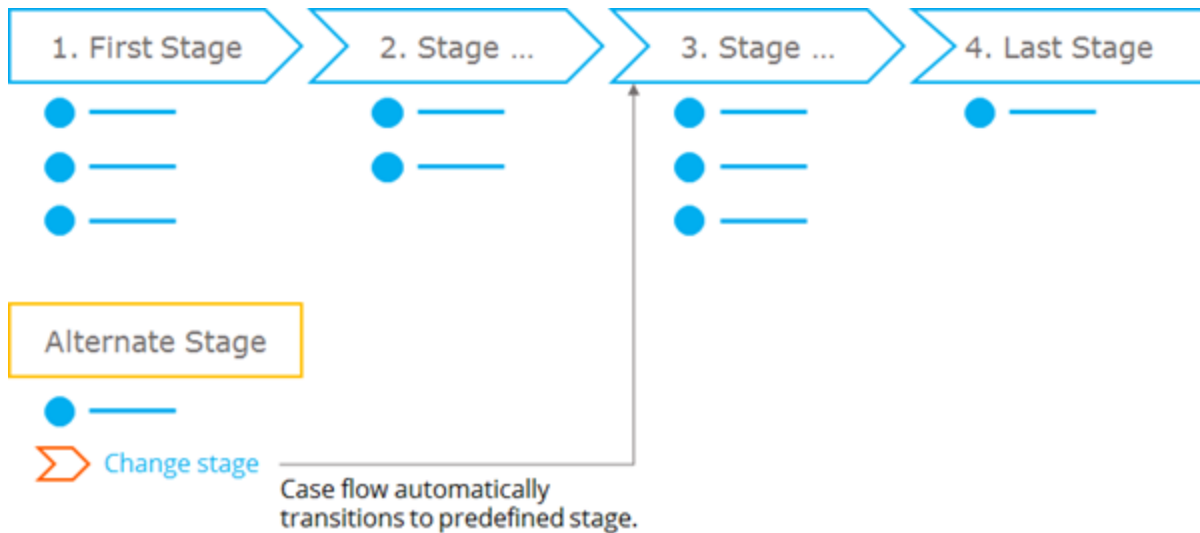


For primary stages, when all steps in a stage are completed, the default option is an automatic transition to the next primary stage.

To allow transitions to other stages before the completion of the current stage, you can add a controlled transition to a stage. Controlled transitions can be configured for any primary or alternate stage, and can occur either as an action in a step or as a specific step in a process.



You can configure a step to allow a user to select the stage to which the case transitions. This type of configuration is most useful for steps that require a Yes/No decision. For example, a case worker must review a request and can either approve or reject the request. If the request is approved, normal processing continues, and the case advances to the next step or stage in the primary path. If the request is rejected, the case advances to a predefined stage, which may or may not be in the primary path.



You can use a **Change Stage** process step to automatically transition the case flow to a specified stage. This type of configuration is most useful for automating transitions to and from alternate stages. For example, a rejected request is sent back to the originator to be updated. The process steps for updating the request are organized in an alternate stage. When the Change Stage step is encountered, the case flow automatically transitions to the stage defined in the Change Stage step.

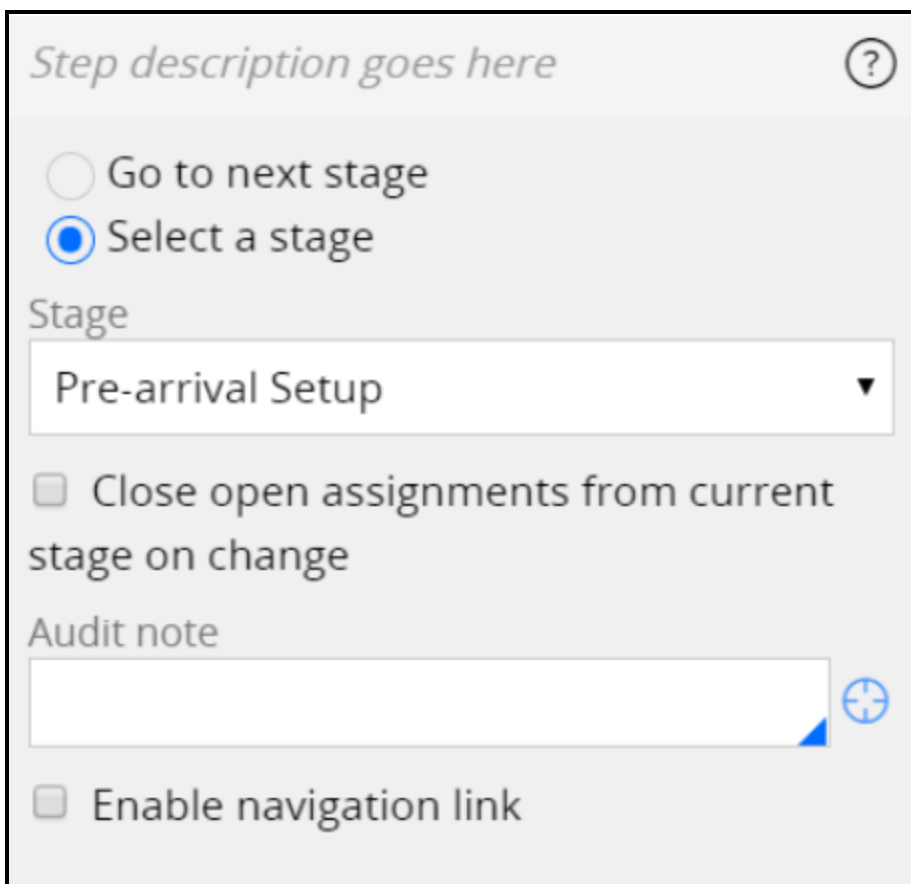
Controlling stage transitions

To allow transitions to alternate stages, or before the completion of a stage, you can add a controlled transition to a stage. Controlled transitions can be added as a process step using the Change Stage smart shape, or as an optional user action in a process step.

Use the Change Stage smart shape to control the stage to which a case transitions

Follow these steps to add a Change Stage smart shape.

1. In the process where you want to add the Change Stage smart shape, click **+STEP** to add a step.
2. In the palette that is displayed, click **More** to view choices.
3. Click **Utilities** to display a list of available smart shapes.
4. Click **Change Stage**, and then click **Select** to add the smart shape to the process.
5. In the contextual property panel, select the **Select a stage** option to specify a stage.



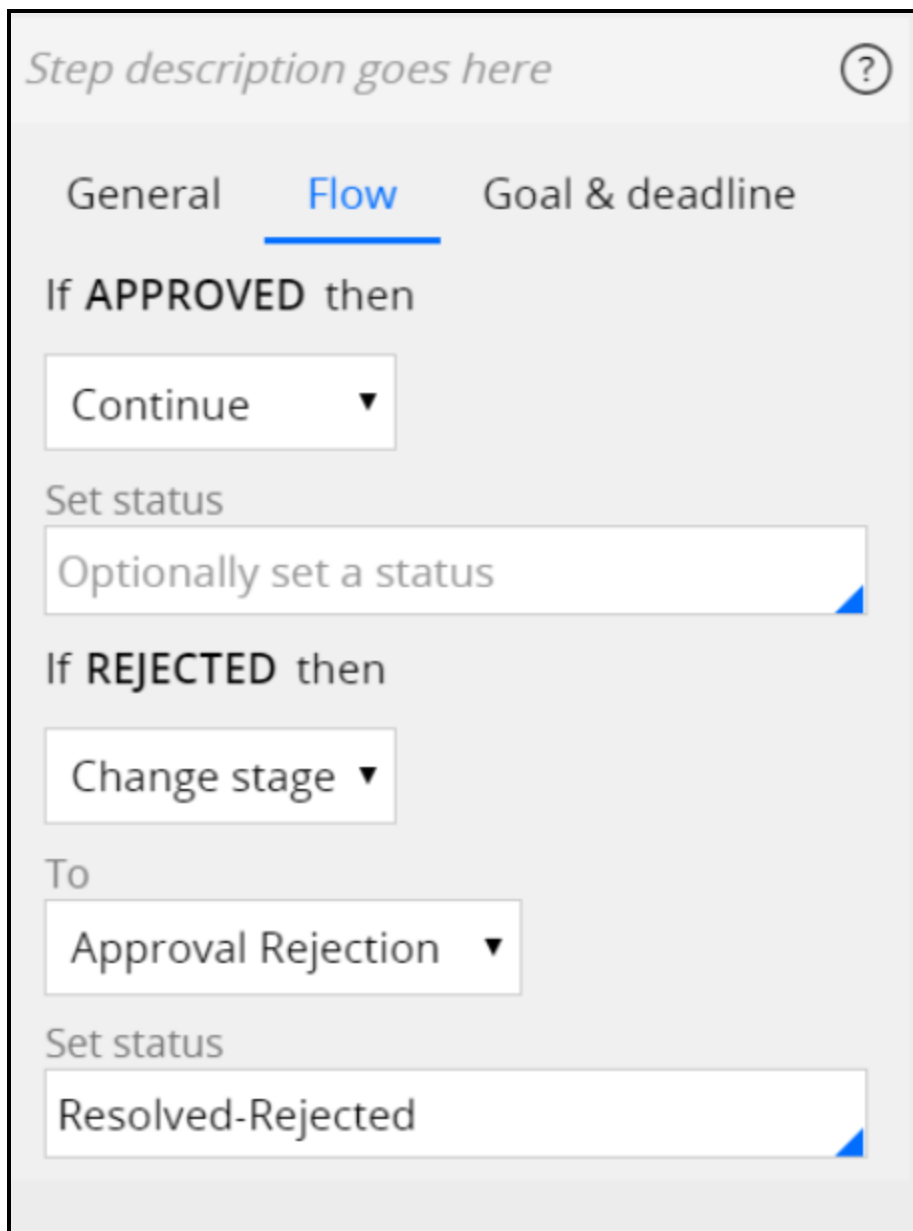
The screenshot shows a contextual property panel for the 'Change Stage' smart shape. At the top, there is a placeholder text 'Step description goes here' and a help icon (question mark in a circle). Below this, there are two radio button options: 'Go to next stage' (unselected) and 'Select a stage' (selected). Underneath, there is a 'Stage' label followed by a drop-down menu currently showing 'Pre-arrival Setup'. Below the drop-down, there is a checkbox labeled 'Close open assignments from current stage on change' which is unchecked. Further down, there is an 'Audit note' label followed by a text input field with a blue plus icon to its right. At the bottom, there is another checkbox labeled 'Enable navigation link' which is unchecked.

6. In the **Stage** drop-down list, indicate which stage the case transitions to when the Change Stage shape is executed.

Use the Approve/Reject step to control the stage to which a case transitions

Follow these steps to allow a user to select the stage to which a case transitions.

1. In the process where you want to add the Approve/Reject step, click **+STEP** to add a step.
2. In the palette that is displayed, select the **Approve/Reject** step to add an approval step.
3. On the Flow tab of the contextual property panel, set the option for **If APPROVED then** or **If REJECTED then** to **Change stage**, and then select the stage to which the case transitions when the Approve/Reject step is executed.



The screenshot shows a contextual property panel for a step. At the top, it says "Step description goes here" with a help icon. Below that are three tabs: "General", "Flow" (which is selected and underlined), and "Goal & deadline". Under the "Flow" tab, there are two main sections: "If APPROVED then" and "If REJECTED then".

If APPROVED then

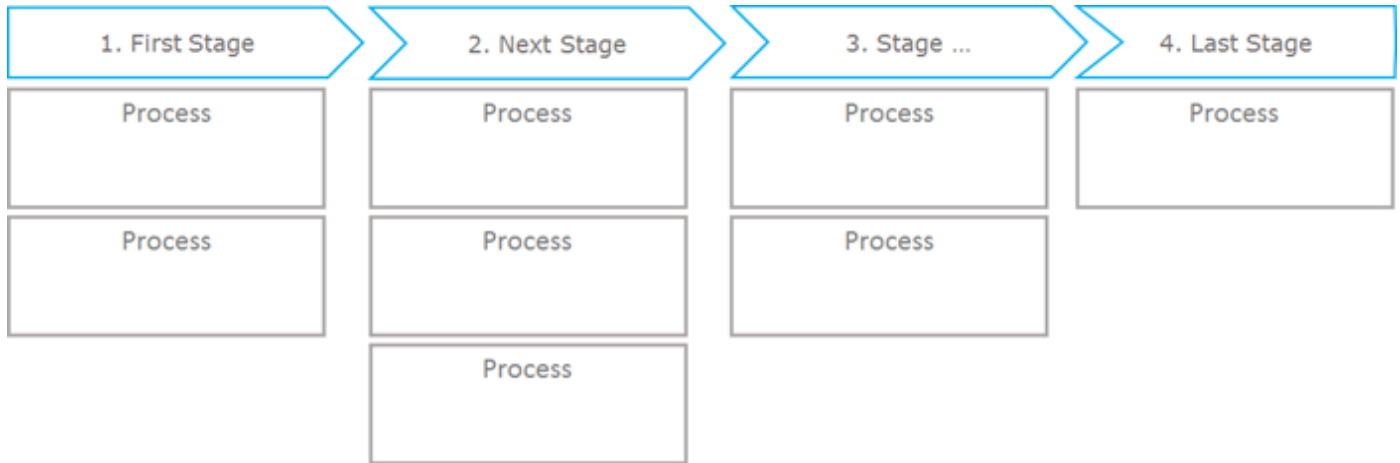
- A dropdown menu is set to "Continue".
- A "Set status" field is empty with the placeholder text "Optionally set a status".

If REJECTED then

- A dropdown menu is set to "Change stage".
- A "To" dropdown menu is set to "Approval Rejection".
- A "Set status" field is set to "Resolved-Rejected".

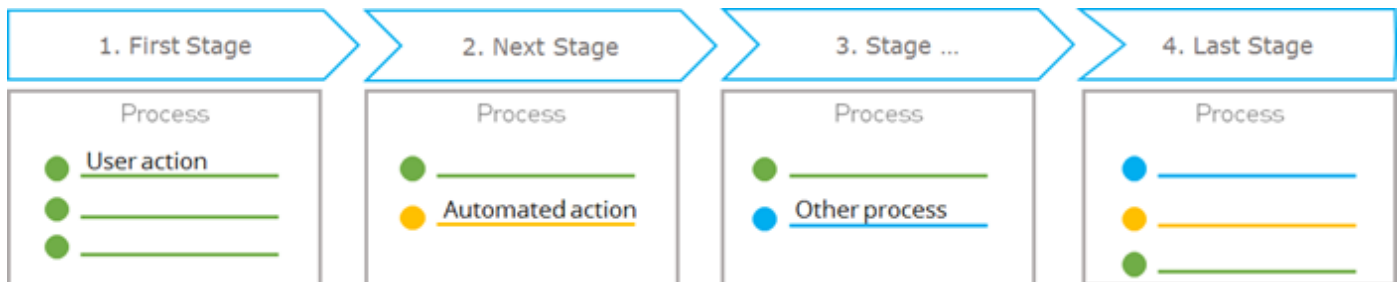
Processes

In a case life cycle, **processes** are organized within stages and define one or more paths the case must follow. You add a process to a stage in a case type to define a set of tasks that users accomplish as they work on a case.



Process steps

A process contains a series of tasks, or steps. A step can be an action that a user performs, an automatic action performed by the application, or other processes that contain a separate set of actions.

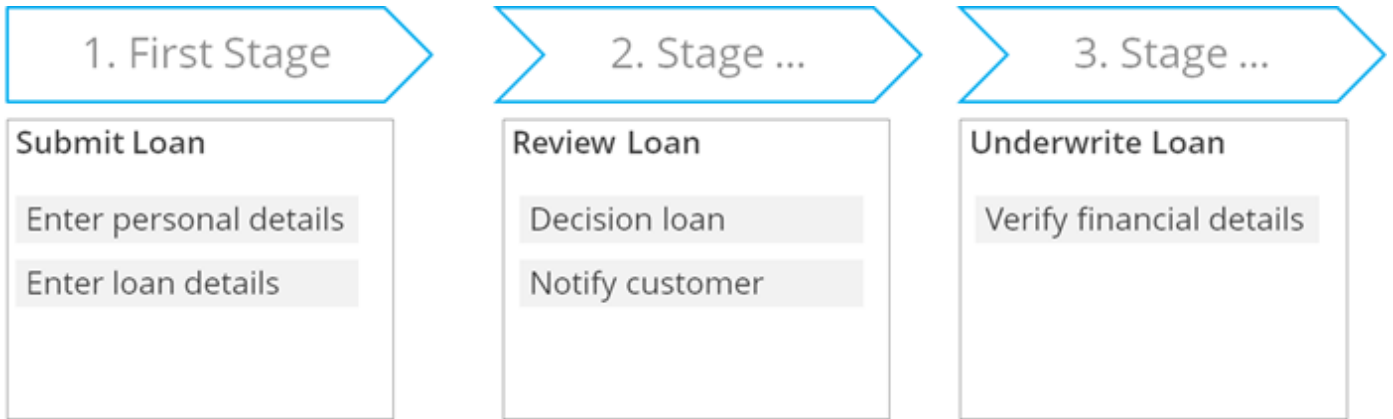


By organizing related tasks into processes, you can control how, when, and by whom work is performed in each stage of the case life cycle.

Guidelines for defining processes

Naming processes and steps

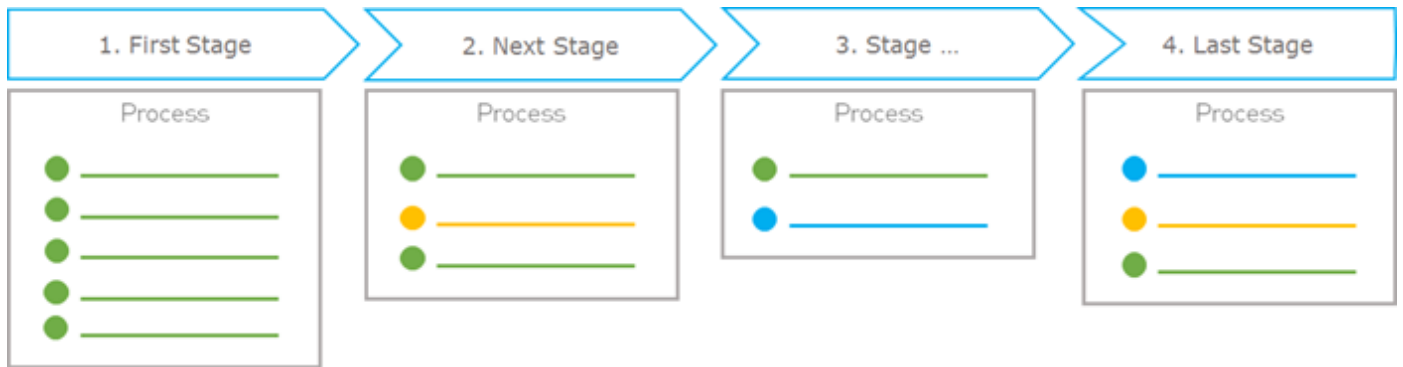
When naming processes and steps, use a “verb + noun” naming convention (ex. perform “this action” on “this object.”)



Consider every process as a distinct action taken to help resolve a case. Every process should have a goal that can be expressed as a singular outcome.

Organizing process steps

Consider limiting the number of steps in each process to five, plus or minus two.



If more than seven obvious steps are needed, consider breaking down some steps into other processes.

KNOWLEDGE CHECK



In Pega Platform, _____ are organized within stages and define one or more paths the case must follow.

processes

Adding optional actions

Introduction to adding optional actions

As a user works on a case, situations arise that may require the user to stray from the primary path to perform a task. You can define optional actions that users can perform while a case is in any stage or step of the life cycle. By allowing users to choose when additional processing is needed, you can support out-of-sequence events in a case.

Objectives

At the end of this lesson, you should be able to:

- Explain the role of optional actions in a case
- Differentiate between optional user actions and optional processes
- Add optional actions to a case

Optional actions

Optional actions supplement the tasks users can do as they work on a case. Optional actions enable users to leave the primary path of a case to complete another task or process. The key is that users make the determination to execute the optional action; the optional action is not part of the automated workflow.

For example, a customer gives you a new cell phone number while you are processing the car details of an Auto Loan. You can use the Update Contact Info optional user action to update the number. By using the optional user action, you do not need to move the Auto Loan case to an earlier step or stage.

When determining if you need to use an optional user action, consider the following questions:

- Should users be allowed to update information at any time during a case or stage?
- Do you need multiple steps to update information?
- Can information be updated in a single screen?

Optional actions can be made available for a stage or an entire case. By adding an optional action to a case, the action is available to users while the case is open. By adding an optional action in a stage, the action is available only in the stage where the action is defined.

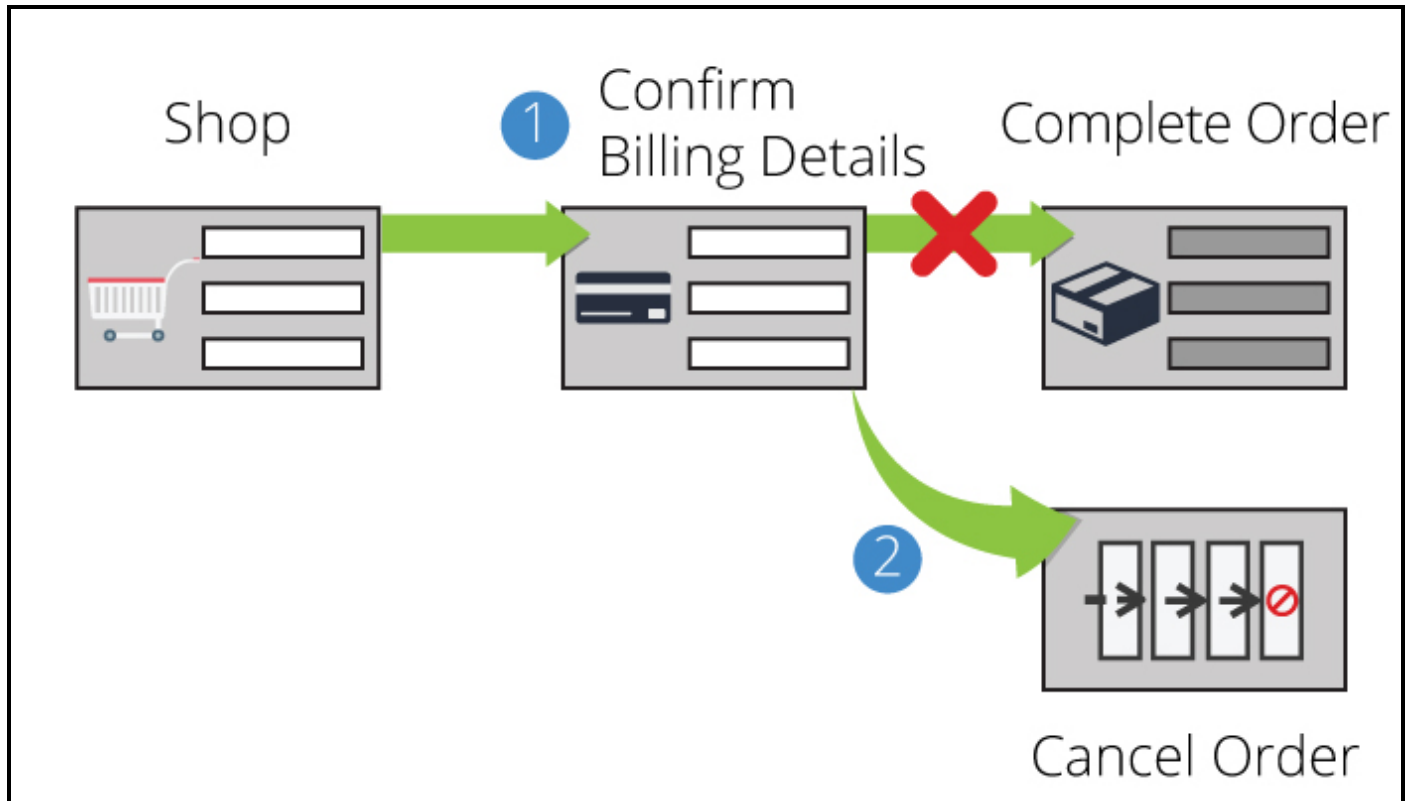
You can add two types of optional actions: optional processes and optional user actions. You use an optional user action to display a single screen to users, whereas you use an optional process to launch an entire process.

Optional process

You use an **optional process** when you need multiple steps to update information. An optional process allows users to run a new process from within the case. The only difference between an optional process and the other processes in a case is that users determine when the optional process executes.

An optional process allows users to perform a series of tasks outside of the primary path of a case. After completion of the optional process, users may or may not return to the primary path.

For example, in a commerce application, you could have a *Cancel Order* optional process that allows a user to cancel an order as long as it has not shipped.

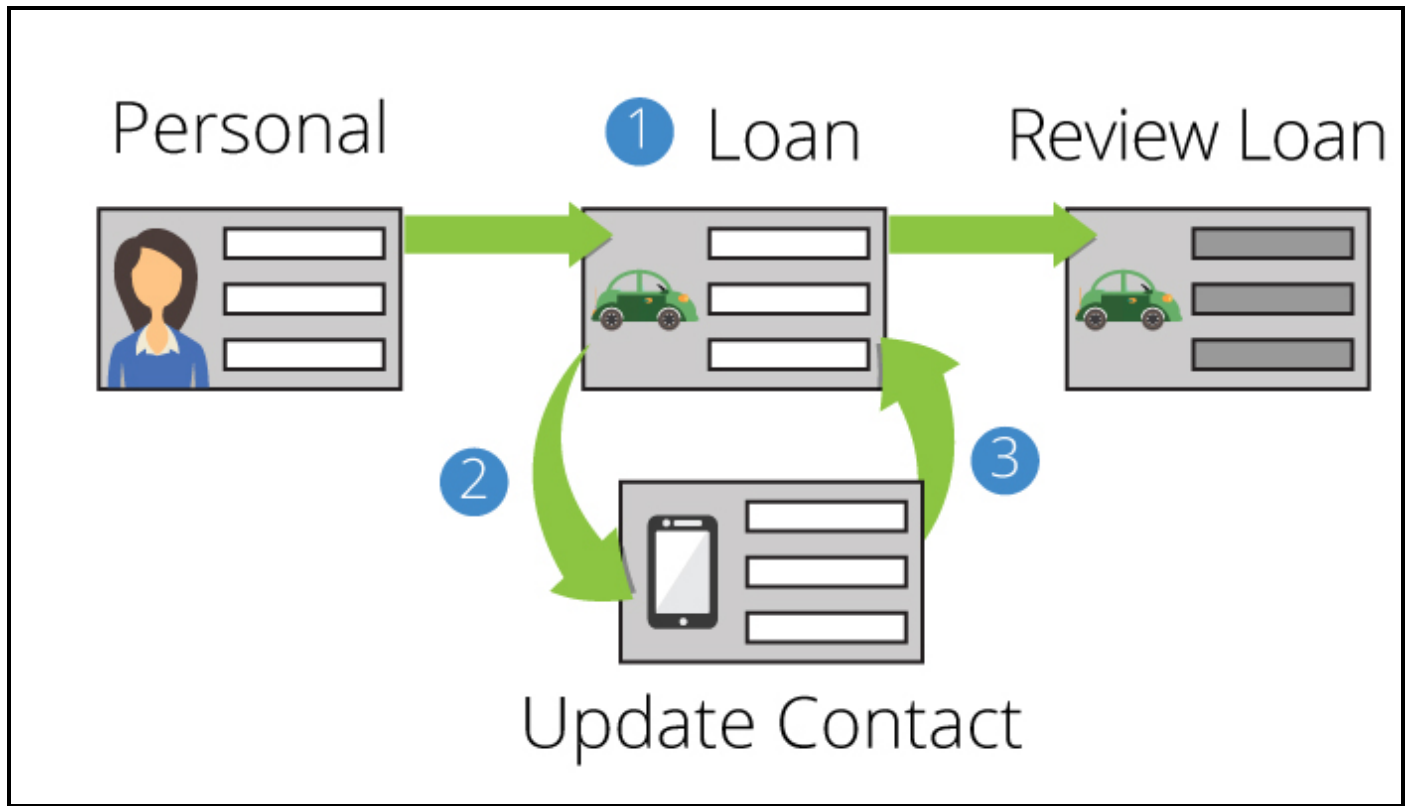


1. After shopping, a user starts the checkout process. While confirming billing details, a user might decide to cancel the order. A *Cancel Order* optional process launches.
2. The *Cancel Order* process executes. The configuration of the process determines if the case completes or it transitions back to continue the original flow.

Optional user action

You configure an **optional user action** when the case can update in a single screen. Think of an optional user action as a screen that is accessible to users. An optional user action enables users to perform a single task outside of the primary path of a case. After completion of the optional user action, the user may or may not return to the primary path of the case.

For example, in an automobile loan application, you want to give the user the ability to change customer contact information at any time in the case, but you do not want the user to lose their place in processing the case.



1. A user enters the personal information for a customer. While entering the loan information, the customer asks to update a cell phone number. The user launches the **Update Contact Information** local action.
2. The user completes the **Update Contact Information** local action.
3. After completing the local action, the user sees the loan information screen.

KNOWLEDGE CHECK



ANSWER

What are the main differences between an optional user action and an optional process?

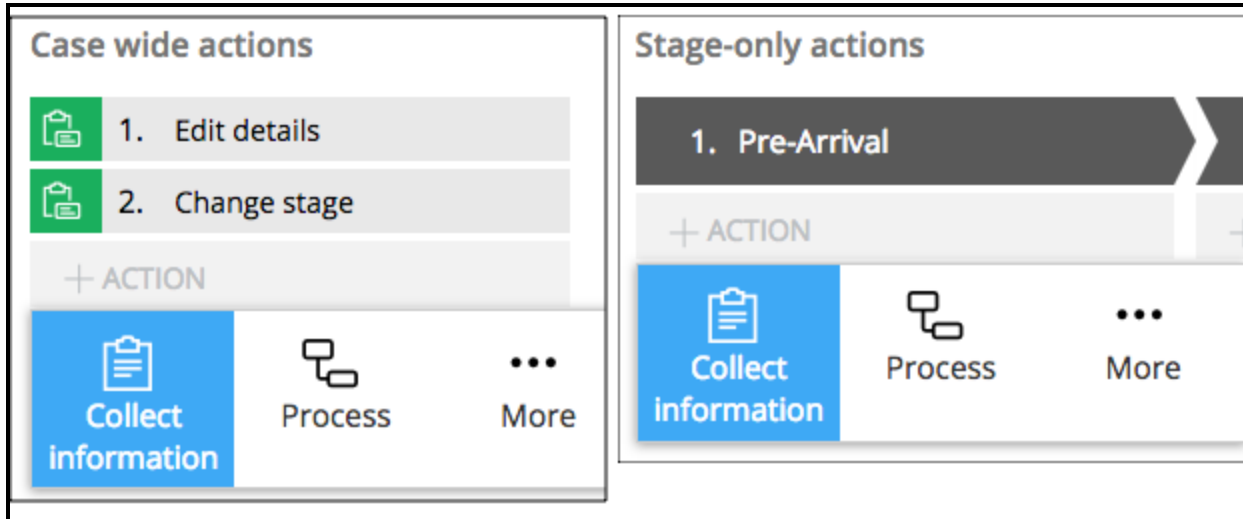
An optional user action is a single task and typically returns to the primary path of a case, whereas an optional process is a series of steps that may or may not return to the primary path of a case.

Adding optional actions to a case type

You can define optional actions that users can perform while a case is in any stage or step of the case life cycle. By allowing users to choose when additional processing is needed, you can support out-of-sequence events in a case. These optional actions can be optional processes or user actions.

Follow these steps to add an optional action to a case type:

1. In Pega Express or the Case Designer, select a case type.
2. Click the **Workflow** tab.
3. Select the **Optional actions** view to see the optional actions.
4. Click **+ ACTION** in *Case wide actions* or *Stage-only actions*, then select the type of optional action to add.



Guiding users through a case life cycle

Introduction to guiding users through a case life cycle

You can boost operational efficiency by guiding users through the most effective actions to take that help accelerate results and achieve desired business outcomes faster. Adding work statuses and instructions helps to keep users informed about the progress of a case. By using these options, you can help users complete their work more productively.

Objectives

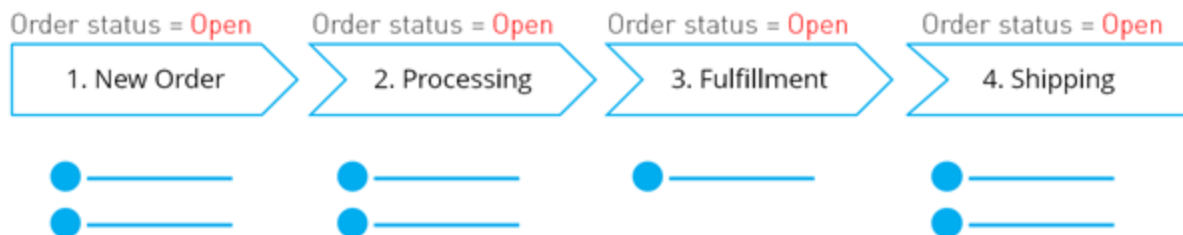
At the end of this lesson, you should be able to:

- Explain how work status adds context to a case
- Update the work status for a case
- Explain how instructions add context to an assignment for a user
- Add instructions to assignments

Updating the case status

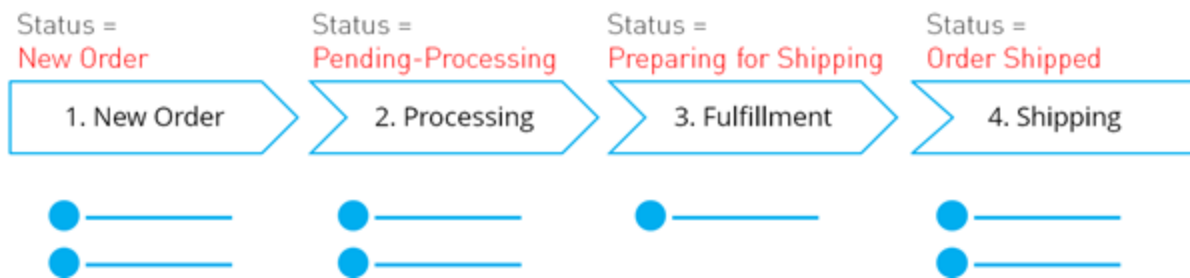
Case status

Consider what would happen if you placed an online order, and the status of that order could only be identified as *Open*.



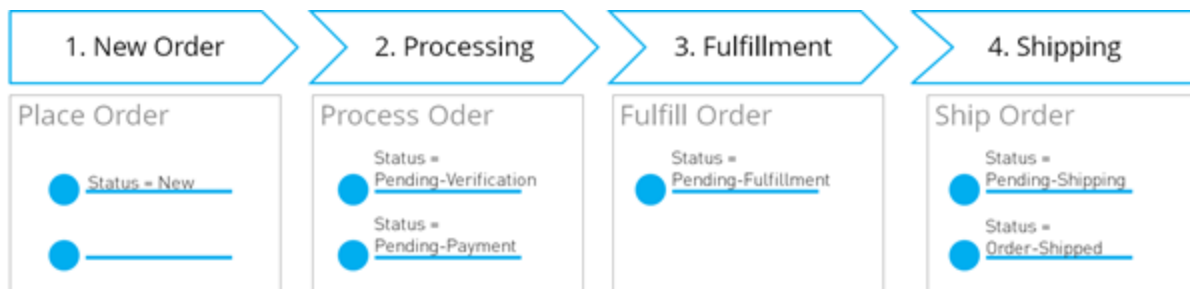
You would have difficulty determining the status of your online order.

Every case, whether it is an online order, a loan origination request, or an insurance claim, has a status. The **case status** is the primary indicator of the progress of a case towards resolution.



The case status is updated as the case moves through the case life cycle.

For example, a case status of *New* is assigned to each case when the case is created. As the case progresses through the case life cycle, the status of the case is updated at each step.



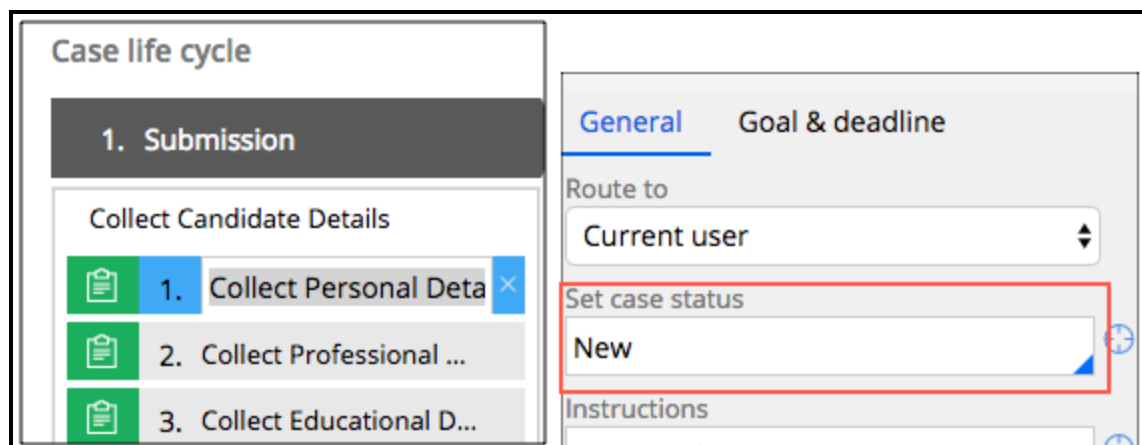
You can set the case status on each step in the case life cycle. When the case advances to a step, Pega automatically updates the status of the case to the value defined for that step.

Pega includes standard case status values, such as Open, Pending-Approval, and Resolved-Completed. You can also add custom status values.

Updating the status of a case

To update the status of a case:

1. In a case type, select a step where you want to change the status.
2. On the General tab of the contextual properties panel, enter a status in the **Set case status** field.



Adding instructions

Imagine you are a loan processing agent. Your company offers car loans, home mortgages, and personal loans. When it is time for you to work on your next case, you could see many cases with arbitrary IDs such as B-5, I-23, N-33, G-57, or O-62. Which case do you choose? These case IDs by themselves give you no help in determining what work you need to perform on those cases. Some could take five minutes, others could take one hour. Pega solves this problem by allowing an architect to write an instruction for each step in a process.

An instruction for a step identifies to a user what should be accomplished in an assignment. Users see instructions in their worklist, when they open a case, or when they are prompted for input in a user view. For example, in a loan application there is a step for the applicant to supply personal information to apply for a loan.

The screenshot shows a user interface for a step titled "Collect Personal Details". The step name is highlighted with a red box. Below the title, there is a dropdown menu for "Position applied for" with "Project Manager" selected. A "Referral" checkbox is also present. The "Candidate" section contains four input fields: "First Name", "Last Name", "Email", and "Taxpayer ID Number", each marked with an asterisk to indicate it is required. A user icon labeled "SA" is visible in the top right corner.

When configuring the case, you switch to Designer Studio and add the instruction in the contextual properties panel so it is clear what work needs to be done on the collect information step.

Defining user views

Introduction to defining user views

Pega application users perform tasks by entering information in views. In Pega, you can create views while developing case life cycles.

This lesson shows you how to configure user views and the data elements in those fields while creating case life cycles.

After this lesson, you should be able to:

- Describe the purpose of a user view
- Define data elements
- Explain how to configure user views and data elements during case life cycle creation
- Configure a user view and its data elements while creating a case life cycle

User view planning

Enterprise applications typically require some human interaction. As a case goes through a process, users perform a variety of tasks along the way. To perform tasks, users enter or review information in user forms. Pega's term for a user form is a user view.

Views are designed so that users can complete tasks. In order to complete the tasks, users enter information in fields. The system stores the values entered by users as data. The application uses this data to process a case. This data is reusable and can be included on other views so that a different set of users can perform their own specific tasks.

Views for specific tasks

Consider a process for making loans. In this example, there are two steps and two views.

The first step in the process requires customers to enter a loan application. The view contains fields for entering information such as the customer's name, the loan amount, and the loan type. After customers complete the view, the system sends the request to loan officers for review.

In the second step of the process, loan officers see a loan officer view that displays the data collected from the application. Loan officers can read the information but not update it. The loan officer view also contains fields for officers to enter information such as loan insurance eligibility and the reason for approval.



Identify the user tasks

You determine the information users need to see or collect in order to perform their specific tasks. You create a view in which users enter the specified information.

Before you create a view, ask the following questions:

- What fields do users need to see?
- How will users enter values in those fields?
- Can users modify the field values or only read the values?

Record the values users enter in a specification. The following example shows fields for a loan application form.

Collect Loan Information

Name of Field	How to Enter Field Value	User Entry
Amount	Enter dollars (\$)	Required
Loan Type	Select available options from dropdown list	Required
Requested Terms	Select available options using radio buttons	Required
Origination Date	Entered by system, display date	No

KNOWLEDGE CHECK



What should you consider before creating a user view?

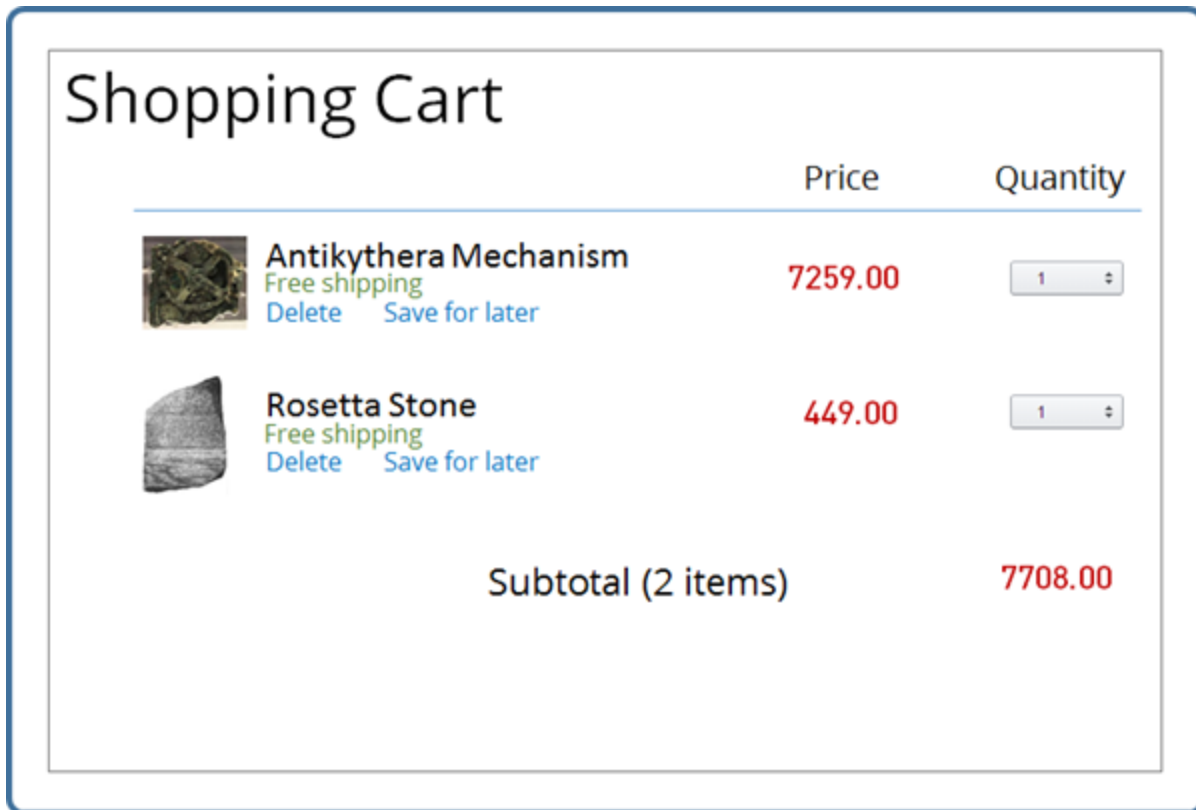
Consider what fields the users need to see, how values will be entered, and whether users can modify or only read the values.

Guidelines for designing user views

The user interface (UI) is the user's view of the application. To be effective, a UI must meet the user's needs and be easy to use. A well-designed UI provides a better understanding of what the application delivers. Consider the following guidelines when planning a user view.

Design an intent-driven UI

Provide users with information when they need it. An intent-driven UI shows users what they need to know to perform a task. The following view of a shopping cart for an online order is an example of an intent-driven UI. The shopping cart review UI displays only relevant information, such as the intended items and the number and price of each item.

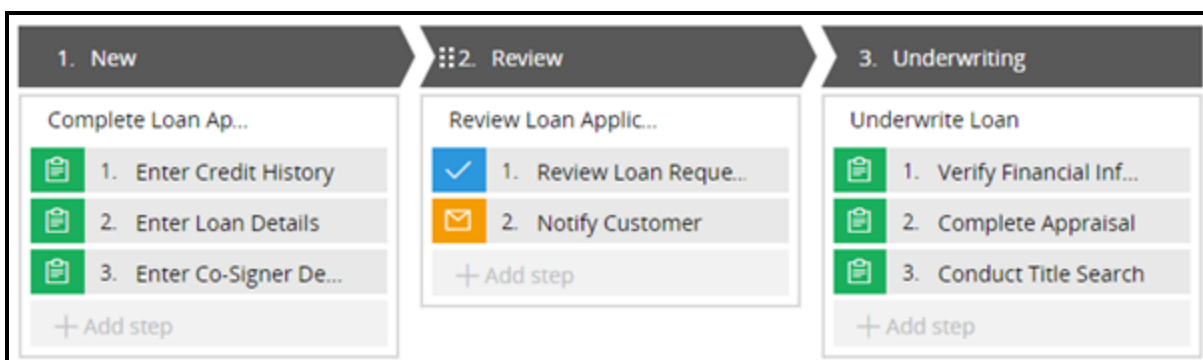


Options such as changing the billing address are not relevant in this step. Making an address change could distract the user from the shopping cart.

Users spend less time searching for relevant information when the intent is the focus of each case life cycle step.

Design a model-driven UI

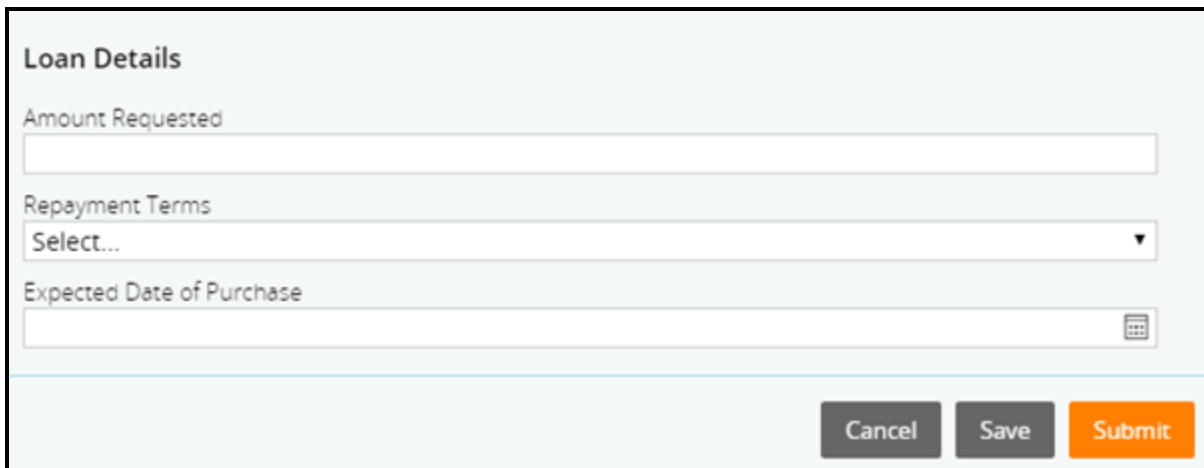
Align the user view with the case life cycle to ensure the user interface is model-driven. The following image of a case life cycle shows each step. The requirements of each step dictate the information in the view.



A model-driven approach results in faster application development, and a contextually sensitive UI. Another benefit of the model-driven approach is a UI that can quickly change when business rules change.

Keep the UI simple and obvious

The best interfaces are almost invisible to the user. The following view is an example of an almost invisible interface. It uses meaningful labels that describe the required information.



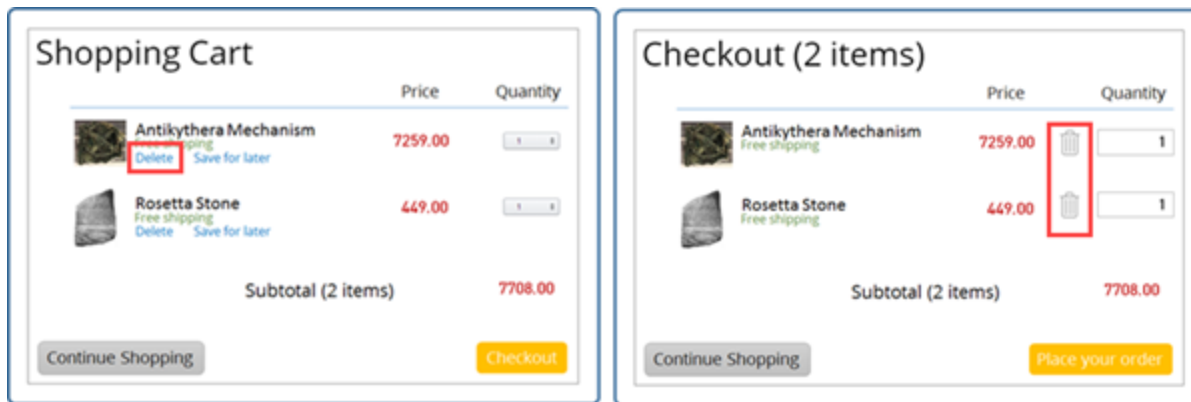
A screenshot of a 'Loan Details' form. It features three input fields: 'Amount Requested' (a text box), 'Repayment Terms' (a dropdown menu with 'Select...' as the placeholder), and 'Expected Date of Purchase' (a text box with a calendar icon). At the bottom right, there are three buttons: 'Cancel' (grey), 'Save' (grey), and 'Submit' (orange).

Minimize data elements to an optimal set that is critical to the task. Focus on what users are trying to accomplish. Be aware of where users are in the life cycle and the next best action. Users should not have to guess how to proceed while looking at a view.

Use common UI labels and elements

Use common labels and elements in the application UI. Users are more comfortable and complete tasks faster when views are consistent within an application.

The following views illustrate consistencies and inconsistencies within the application. The major elements - the items, price, quantity, and the buttons - show consistency. They are in the same location in both views. The delete function and name of button on the lower right side are different. They show inconsistency which could potentially delay the person placing the order.



Two side-by-side screenshots of a shopping application. The left view is titled 'Shopping Cart' and shows two items: 'Antikythera Mechanism' (Price: 7259.00, Quantity: 1) and 'Rosetta Stone' (Price: 449.00, Quantity: 1). Below the items is a 'Subtotal (2 items)' of 7708.00. At the bottom, there are 'Continue Shopping' and 'Checkout' buttons. The right view is titled 'Checkout (2 items)' and shows the same two items. In this view, the 'Delete' button for the 'Antikythera Mechanism' is highlighted with a red box. Below the items is a 'Subtotal (2 items)' of 7708.00. At the bottom, there are 'Continue Shopping' and 'Place your order' buttons. The 'Place your order' button is highlighted with a red box.

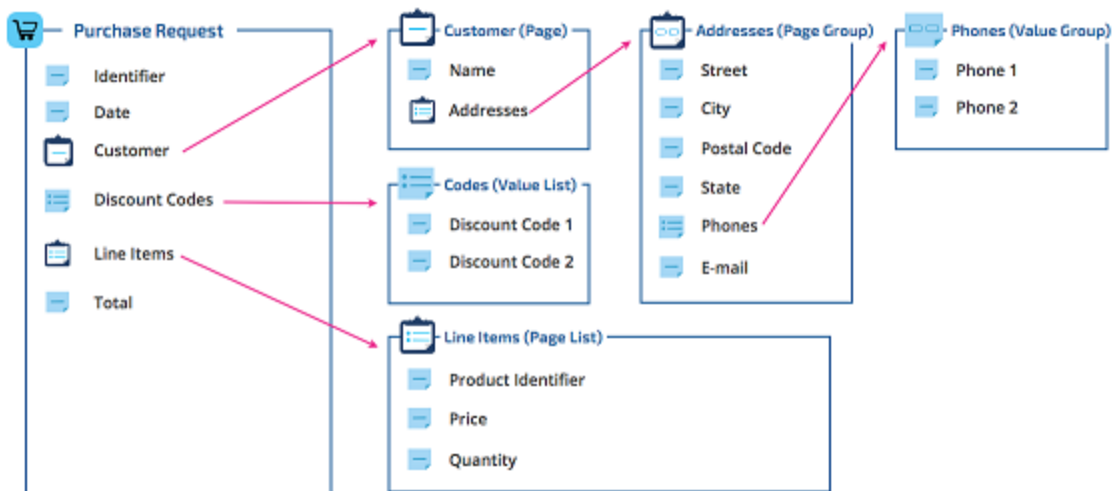
Create patterns in language, layout, and design to help users complete their tasks. Design consistency enables users to recognize similar tasks in other application views.

Data elements in Pega applications

All applications collect data to use for case processing. Collected data informs decisions on how to best process and resolve cases. For example, the data in a purchase request case can include the customer information and line items.

In Pega, you create data elements as you configure user views during case life cycle creation. The data elements or collection of related data elements in a case type comprise the case type's **data model**. The data model defines the case type data structure. A collection of related elements is called a **data type** or **data object**.

The following table shows the data model for a purchase request case.



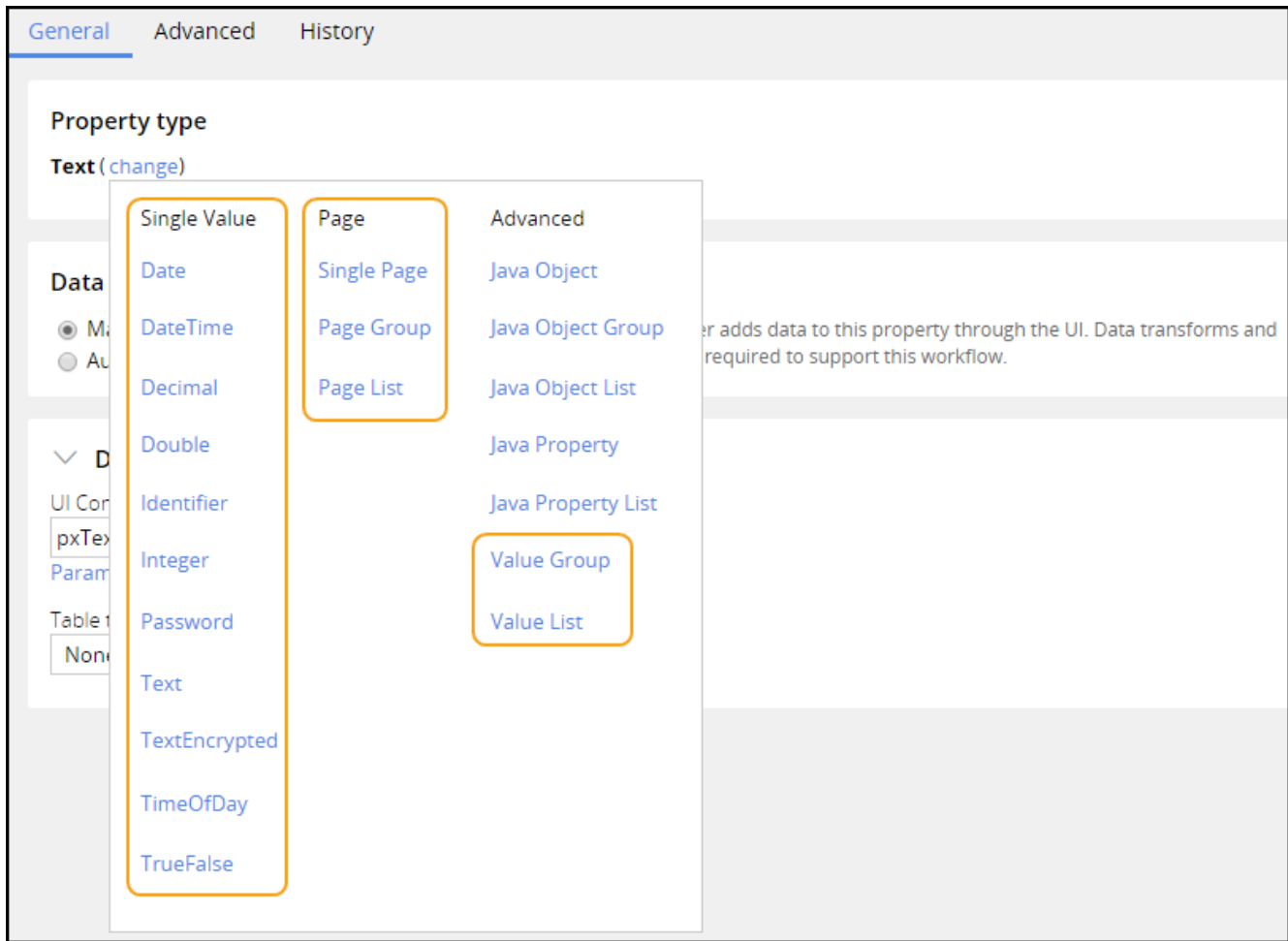
The purchase request case has a unique identifier, date, customer, list of line items, and total. The unique identifier, date, and total are single value data elements. The customer and line items are data objects. A data object or data type contains more than one related property element.

The purchase request has a one-to-one relationship with the customer and a one-to-many relationship with the line items.

The customer has a name and lists of phone numbers, addresses, and discounts. The name is a single value element. The name has a one-to-many relationship to the phone number, address, and discount code elements.

Properties

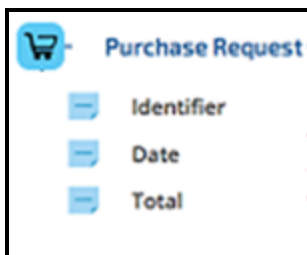
In Pega, data elements are called **properties** or **fields**. Property and field are different names for the same thing. Properties can be either single value with no intended correlation with any other value, or a collection of related values. This distinction is explained by the mode of a property. System architects typically work with two types of property modes: value modes and page modes. **Value modes** describe a single piece of information such as a total. **Page modes** describe a data object such as a customer. The following screenshot highlights the value and page mode property types.



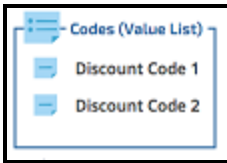
Value mode properties

Use value mode for properties with no correlation to other properties. For example, the identifier and date in the purchase request are value mode properties. There are three value mode properties available: single value, value list, and value group.

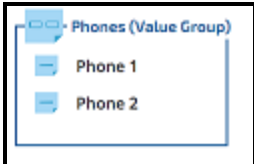
- A **single value** or scalar property stores text, numbers, dates, Boolean values, and amounts.



- A **value list** acts as a container for an ordered list of single values. The discount codes property is an example of a value list. Each code is a single piece of information, but a clear relationship exists between the codes.



- A **value group** acts as a container for an unordered list of single values. The customer's phone numbers are defined as a value group identifying the contextual meaning of each number: home, work, or mobile.



When you create a value property, you can assign it to one of 10 different property types. This identifies the type of information the property stores. By assigning a type to a property, you ensure that users provide valid information as shown in the following image.

New: Booking

Metro area
 Text

Arrival date
 Date

Departure date

Number of guests
 Integer

Title	Price per night
Cozy Room	\$100.00
Downtown Condo	\$200.00

The following table shows a list of property types and the information each type stores.

Property type	Type of data stored	Example
Text	Any text	red
Identifier	Text strings that do not contain double quotation marks (""), tabs, carriage returns, or line breaks	XYZ
Password	Encrypted graphical characters	Password
Encrypted text	Similar to the password type, but can be decrypted for display	Password

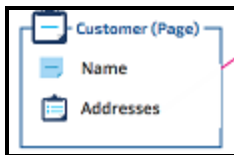
Property type	Type of data stored	Example
Date	Calendar date in the format YYYYMMDD	20131202
TimeOfDay	Local time in the format HHMMSS	052709
DateTime	UTC (Coordinated Universal Time) value normalized to Greenwich Mean Time (GMT)	20131202T052709
Integer	Positive and negative whole numbers, and zero	4
Double	Double-precision (64-bit) floating point values (available only in Designer Studio)	23.55
TrueFalse	Boolean value	True
Decimal	Numbers with a fractional component	32.5

Page mode properties

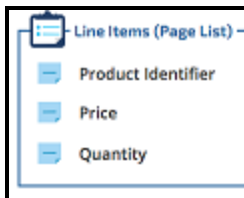
If you need to establish a contextual relationship between single value properties, you can use one of the three **page-mode properties**: pages, page lists, and page groups.

Page mode properties are organized similar to value mode properties.

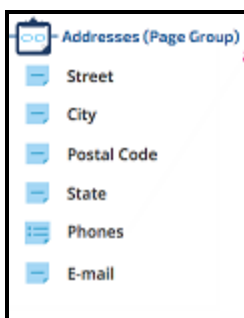
- A **page** is a single entity. The customer is an example of a page property. A page is also referred to as a field group.



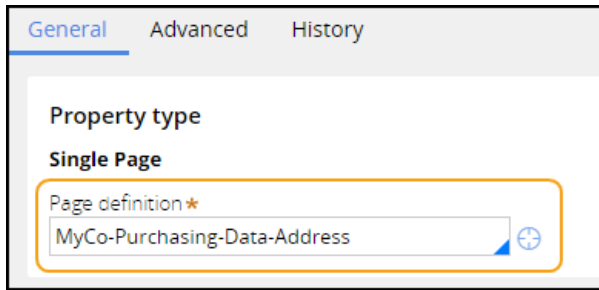
- A **page list** is a numerically ordered list. The line items that make up the purchase request is an example of a page list. A page list is also referred to as a field group list.



- A **page group** is an unordered list. The address property is an example of a page group.



Note: The page mode properties require you to specify a definition, or a data type, that defines the structure of the page property.



Configuring user views

After you add steps to the case life cycle design, you can configure user views for each of the steps.

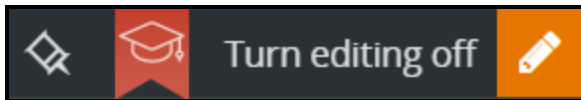
Before you configure a user view, remember to answer these questions:

- What fields do users need to see in the UI?
- How will users enter values in those fields?
- Can users modify the field values or are the fields read-only?

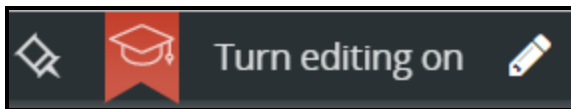
Configure a user view using Pega Express

To configure a user view using Pega Express:

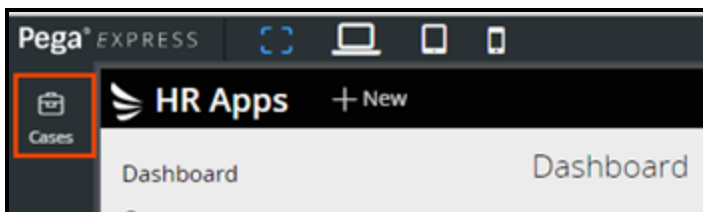
1. Confirm editing is enabled by looking at the upper right corner of the Pega Express dashboard. Look for the text **Turn editing off** as shown in the following image.



If editing is not enabled, click **Turn editing on**.



2. In the Navigation panel on the left, click **Cases** to view a list of current case types.



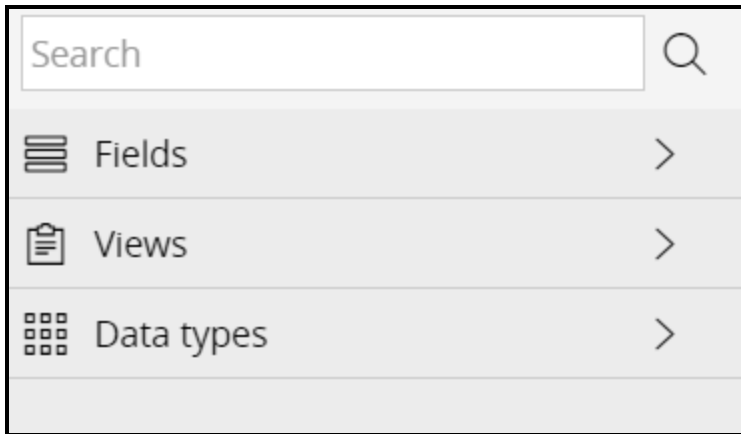
3. From the list of available case types, select the case type for which you want to configure a user view.

4. Select the step for which you want to configure a user view to display the Contextual Properties panel for that step. The Contextual Properties panel displays to the right of the case life cycle.
5. In the Contextual Properties panel, click **Configure view**. The *Views* configuration page displays.

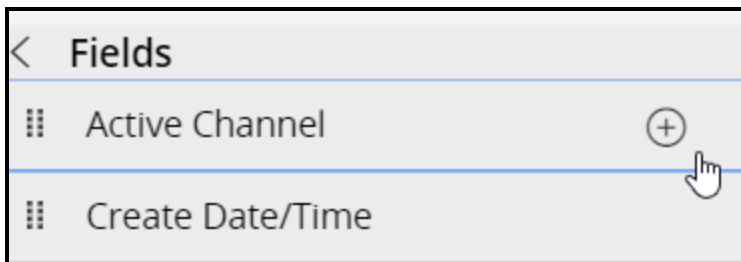
Add default fields to the user view

To view the default fields and select fields to add to the user view:

1. In the left panel, select **Fields** to view the default data elements.



2. If you require any of the default fields, select the row for the field, and then click the plus sign to the right of the field name.



3. Repeat steps 1 and 2 to add more default fields.

Add new fields to the user view

To add new fields to the user view:

1. In the Label field, type a name for the new field. The following image shows a new Loan types field.

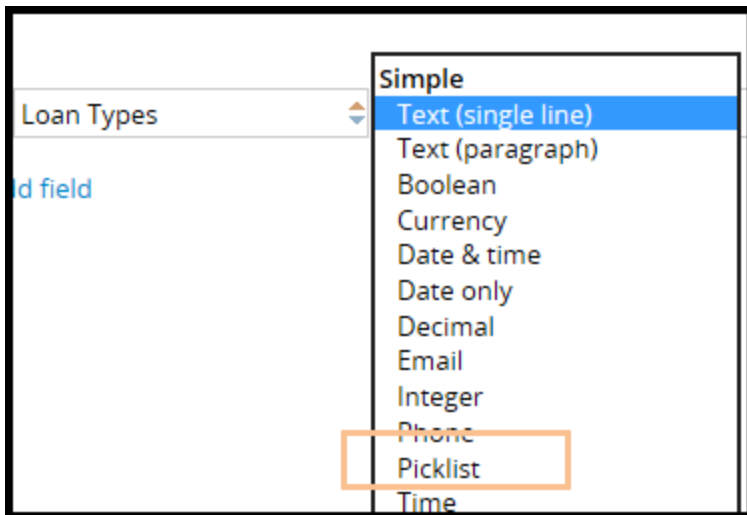
Label	Type	Options
Loan types	Text (single line) ▼	Optional ▼

Note: After you add the required data elements and save your view, the system adds new data elements to your application. You can then reuse those data elements when you create new views.

2. Use the **Tab** key to move to the second field on the row.

3. Select a data type for the data element. The data type defines how users enter a value in the UI field.

For example, if you want the user to select a type of loan from a drop-down list in the Loan types UI field, select Picklist from the drop-down in the data type field as shown in the following image.



The Picklist data type has an additional field for you to choose the type of list (drop-down list or radio buttons) and the names of the items on the list. To learn how to choose the type of list and to configure the item names, see the steps under Designing a picklist.

4. Use the **Tab** key to move to the third field, select either **Optional** or **Required** if you want to allow users to enter a value in a field. If you do not want users to enter or update the field value, select **Read-only**. In the following example, the developer is selecting **Required** to ensure the user selects a value from the Loan Types list.

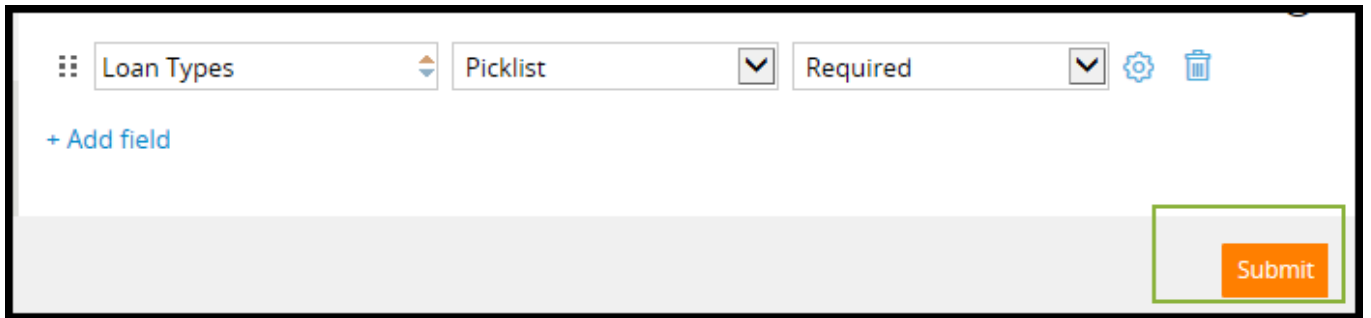


5. To add more fields, click **Add Field** beneath the bottom row.

Save and verify your work

To save your work and review the view:

1. On the bottom right corner of the View configuration page, click **Submit**. When you click **Submit**, the system saves your updates and creates the view that users see when they work on an assignment. The system also saves the data elements that you can reuse in the application.

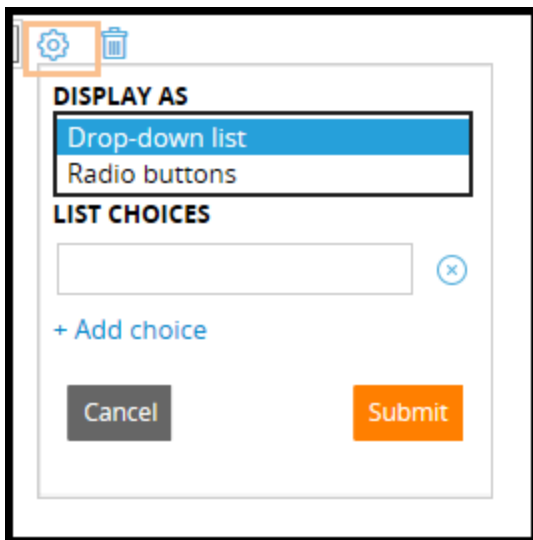


2. Click **Save** to save your changes to the case type.
3. In the upper right corner, click **Run** to run the application. The new fields in the standard Create view display.
4. Verify that the fields you selected and created appear in the user view.

Designing a picklist

If you select the picklist data type, you need to use an additional field to choose how to display the list and the names of the items you want to include on the list. To design the list, do the following:

1. From the end of the row containing the picklist data type, click the **Gear** icon.
2. In the **Display As** field, select one of the following:
 - a. **Drop-down list** if you want users to select an item from a drop-down list.
 - b. **Radio buttons** if you want users to select an item by clicking a radio button.



3. Under **List Choices**, enter the name of the first list item.
4. Click **Add choice** to add more fields for items in the list. The following example shows list choices for loan types.

The dialog box is titled "DISPLAY AS" and contains a dropdown menu set to "Drop-down list". Below this is a section titled "LIST CHOICES" with three input fields containing "Home", "Auto", and "Personal Property", each with a remove icon (x). A "+ Add choice" link is present below the fields. At the bottom are "Cancel" and "Submit" buttons.

5. Click **Submit** in the dialog box to save your list. The items you entered in the List Choices column display in the **Loan Types** drop-down list in the user view.

The user view shows a dropdown menu titled "Loan Types" with three options: "Home", "Auto", and "Personal Property". The "Home" option is currently selected and highlighted in blue.

Validating case data

Introduction to Validating case data

When you design a user form, you add all the fields and controls required by a design specification. However, users must enter data that uses a format or contains a value that the system can process correctly. Pega provides rules that validate the data and help prevent processing errors when a form is submitted.

After this lesson, you should be able to:

- Identify appropriate options for ensuring valid data entry by users
- Configure controls to require user entry in specific fields
- Explain how controls can satisfy validation requirements
- Configure and apply a validate rule to validate case data
- Apply an edit validate rule to ensure that user data matches required patterns

Methods of data validation

When you design a view, you add all the fields and controls that the specification requires. You must also consider how to ensure that the data values entered by users are valid. Valid data is required so that the system can process the information without error.

The following list describes a few important design requirements.

- The data must be the right type. For example, a user must enter a number in a total purchase amount field.
- The data must fit the business logic. For example, a date of birth field is usually in the past.
- The data must be restricted to possible values. For example, a user can only select a valid loan type by selecting the type from a list of options.

To prevent processing errors, Pega provides property types, controls, and rules that support most validation requirements.

Properties

Single value properties have property types such as date, decimal, integer, text, or true/false. Selecting the appropriate property type ensures that users enter a valid value. For example, a purchase price field that uses a decimal property type ensures that users can enter only numeric values and cannot enter text.

Controls

Controls are another way you restrict users from entering or selecting invalid values on a form. For example, when a form requires a date, using a calendar control ensures that users enter a date value. If a user needs to enter free-form text such as an address, you would use a text input control.

You can also use controls to allow users to select only valid values. For example, you can use drop-down lists or radio buttons so that users can select only the available values.

In addition to ensuring valid values, you can make required fields. This ensures that users enter a value before they can complete an assignment.

KNOWLEDGE CHECK



You have added fields for entering the name and address of a loan applicant. What validation methods would you use?

Define the name and address properties as text property types.

Validation rules

You use validation rules when you cannot predict or control the value a user enters in a form. There are two types of validation rules: **validate** and **edit validate**.

Validate rules

You use validate rules to compare a property against a condition when the user submits a form. If the user enters a value that fails to meet the condition, the form displays an error when the form is submitted. For example, assume your view contains a field for date of birth. The property type and control cannot prevent users from entering a date that is in the future. However, you can design a validate rule to display an error if the user submits a date that is in the future.

Edit validate rules

You use edit validate rules with single value, value list, and value group properties to test for patterns. For example, you can configure a zip code property to reference an edit validate rule that tests whether the entered value has five digits. In another example, an email address can reference an edit rule to test whether the entered value contains an "at" (@) symbol. If the submitted value is invalid, the field displays an error. Edit validate rules run when the user exits a field if the harness rule is configured to support client-side validation. Otherwise, edit validate rules are run when the user submits a form.

Note: The standard harnesses provided with Pega Platform are configured to support client-side validation.

KNOWLEDGE CHECK



Which validation method would be appropriate for checking that a user enters a date for scheduling a home inspection that is in the future?






A validate rule

How to validate user data with controls

Controls used on views provide the most common approach to validation. The three basic ways you can use controls for validation are control types, required fields, and editable settings.

Control types

Using the correct control type for a specific purpose helps users enter valid values. The following table shows some example use cases for the different control types.

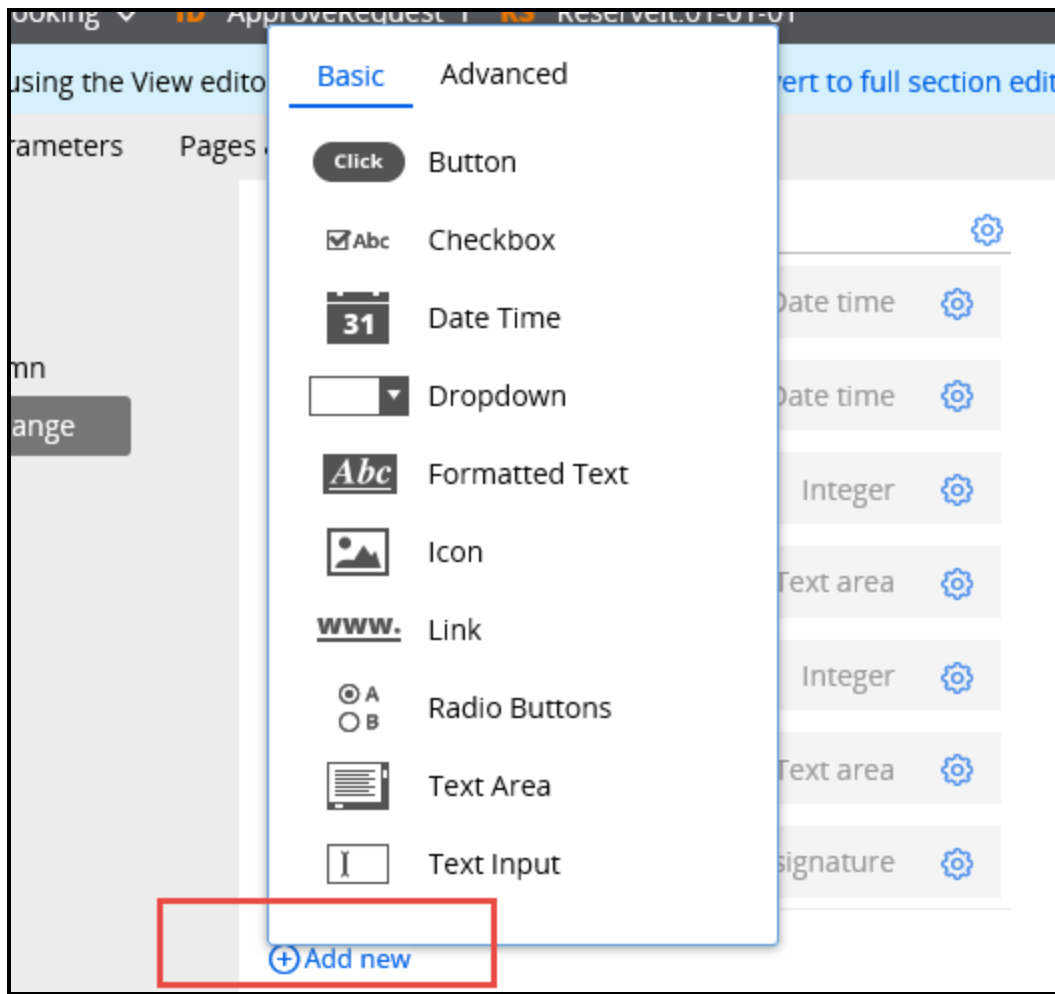
Use case	Control type	How the control helps validation
Users must enter a date that includes day, month, and year.	Calendar 	Selecting a date from a calendar icon helps ensure that the user enters a date in a valid format.
Users must select one of three possible loan types. The user must see all types on the form.	Radio buttons 	Restrict choices to a set of valid values and allows users to select only one value. You can use radio buttons when only a small number of options (for example, fewer than five) is available.
Users must select one of 10 types of office chairs from a list. The options do not need to be displayed on the form.	Dropdown 	Restricts valid values to the ones that appear in the list. A drop-down list presents the options only when the user clicks the control. This helps reduce the clutter on the form. Unlike radio buttons, you can configure the drop-down control so that users can select multiple values.
Users must select the country in which they reside from a list. The user can enter text in the control to help find the right country.	Autocomplete 	When a user enters one or more values in the control, the control filters the available options. This helps users find an option in a list if there is a large number (for example, more than 20) of possible options.
Users select an option to purchase extra travel insurance.	Checkbox 	The user can select the check box or leave it blank. This ensures that a true/false property is either true (selected) or false (unselected).

Specifying control types

Most of the fields you add in the Data Model tab in either Pega Express or the Case Designer are associated with default control types. The control type depends upon the field type you select. The following table shows some examples.

Field type	Control type
Date only Date & Time	Calendar
Boolean	Check box
Picklist	Radio buttons or a drop-down (you choose the control type when you add the property type)
Text (paragraph)	Text area

When working with sections in Designer Studio, you add fields by selecting a control type from a list as shown in the following image.



After you add the field, click the **gear icon** to open the field's Properties panel and specify the property.

Note that most properties are associated with default control types. For instance, assume you select a **Text Area** control. This control is the default for Text property types. However, you specify in the Properties panel a property that is a TrueFalse property type. The text area control you select overrides the property's default check box control. In this situation, you may want to add a **Checkbox** control to the section rather than a **Text Area** control for that property.

Note: Pega standard controls are designed to display in the application platform's native format. For example, a calendar control on an iPhone lets users select a date by tapping it. On the Windows platform, users select a date by clicking it with a mouse.

For more information about control rules, see the help topic [Standard Controls](#).

KNOWLEDGE CHECK



How would you configure a field in which a user selects one of four possible shipping methods?

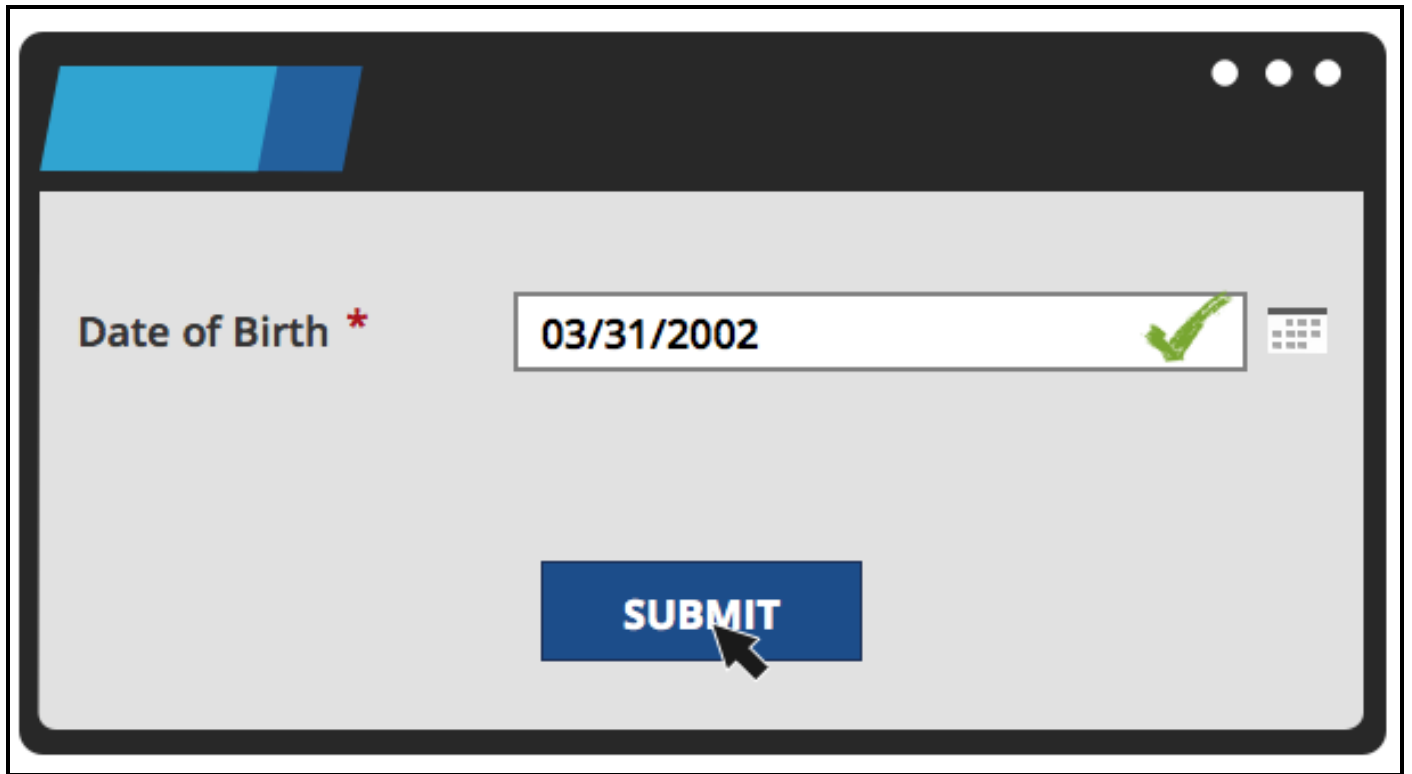
You would use a picklist field and specify either a radio button or dropdown control type.

Required fields

Configuring a control as a required field ensures that the user enters a value. If there is no value, users receive an error when they try to submit a form. For instance, assume you design a view in which users enter a date of birth to qualify for a discounted auto insurance policy. You configure the date of birth number control as a required field. If the user does not enter a date in the field, an error message appears when the user attempts to submit the form.

The screenshot shows a web form window with a dark header and three window control buttons in the top right. The main content area has a light gray background. On the left, the text "Date of Birth *" is displayed in a dark font. To the right of this text is a white rectangular input field. To the right of the input field is a small calendar icon. Below the input field, the error message "Must enter date of birth" is displayed in a bold red font. At the bottom center of the form is a blue rectangular button with the word "SUBMIT" in white capital letters. A black mouse cursor is pointing at the bottom right corner of the "SUBMIT" button.

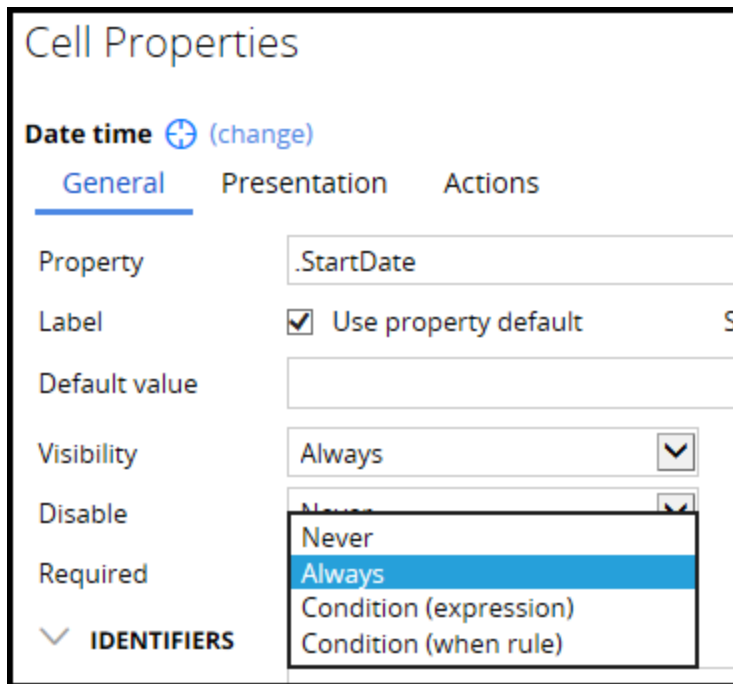
The error message does not appear if there is a date in the field.

A screenshot of a web form interface. The form has a light gray background and a dark gray header bar with three white circles on the right. The main content area contains a label 'Date of Birth *' in black text. To the right of the label is a white text input field containing the date '03/31/2002'. To the right of the input field is a green checkmark icon and a small calendar icon. Below the input field is a blue rectangular button with the word 'SUBMIT' in white capital letters. A black mouse cursor is pointing at the bottom right corner of the 'SUBMIT' button.

Configuring a required field

You can configure required fields when working with views in the Case Designer or when working with sections in Designer Studio.

- In a view, select **Required** in the **Options** drop-down for a required field.
- In a section, open the field's Properties panel. In the **Required** drop-down list, select **Always** so that the user must enter a value under any condition. If you want to make the field required under specific conditions, select either **Condition (expression)** or **Condition (when rule)**.



Note: **Never** and **Always** correspond to the **Optional** and **Required** settings respectively in Pega Express and the Case Designer. The conditional options can only be configured in a section.

KNOWLEDGE CHECK



When would you use the required fields validation approach?

You set a field as required when you want to make sure users enter a value before they can continue.

Editable settings

You can use editable settings on controls to restrict the input values to valid formats. The settings are specific to the control type. For example, you can specify the minimum and maximum of characters allowed in a text input control. You can also specify that users cannot enter dates as text — users must select a date from a calendar icon control.

Configuring editable settings

In a section, open the field's Properties panel and configure the settings on the **Presentation** tab. In the following example, the minimum number of characters is 20 and the maximum is 500. This ensures that users can only enter information that is between 20 and 500 characters.

Cell Properties

Text area [Change](#) | [Revert to property default](#)

General Presentation Actions

Edit options ▼

Read-only value ▼

Editable Settings

Specify width Auto Custom

Specify height Auto Size to Content Custom

Min/Max characters

Display character counter True False


For example, if the user enters fewer than 20 characters, the field displays the following error.

Reason for Rejecting Request

Doc not attached

⚠ The field pyTemplateTextArea should be at least 20 characters long

KNOWLEDGE CHECK

 How would you ensure that a user always enters 20 characters in a field?

You would use editable settings to set both the minimum and maximum number to 20 characters.

How to validate case data with validate rules

Property types and the associated controls — for example, a date type property that uses a date picker control — help ensure valid data entry into user forms. However, these types of validation methods cannot ensure the input provided by end users conforms to business policies.

For example, two forms may contain the same employee start date property. In a job history form, a user enters the start date of an employee who already works at the company. The user must enter a date before the current date. In a new hire form, a user enters a start date for an employee who has not started work. The user must enter a start date after the current date.

Validate Start Date is Before Current Date

Job History

April 2016						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

● Today ● Start Date

Validate Start Date is After Current Date

New Hire

April 2016						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

● Today ● Start Date

Using two validate rules, one for each business condition, you can ensure that the correct dates are entered on each form.

Validate rules are often associated with processes. The association between validate rules and processes enables you to validate the property based on specific business logic. For example, you can use a single property — in this case, a date type property — and separate the type of property from the business conditions that affect the acceptable values.

You can also use validate rules in other areas of the case life cycle. For example, you can use a validate rule on a stage. Use a validate rule to ensure users have entered the correct data or performed all the processes before the case can enter a specific stage. For example, in a mortgage application, a valid credit score must be available before the case can enter the underwriting stage.

KNOWLEDGE CHECK



Validate rules enable you to use a single property when _____.

different values are required based on business logic

Validating a flow action with a validate rule

Validating an input field with a validate rule is a two-step process. First, create and define a validate rule. Then update the process to execute the validate rule when submitting a user view on an assignment step. This is done by configuring the flow action rule Pega automatically creates when adding a Collect information step in your case life cycle. Flow actions provide additional processing capabilities such as running data transforms and calling validation rules. Flow actions can be accessed from the Process Modeler or from the Application Explorer under the Process category.

You use Pega Express to validate simple form validations that compare the value of a field to a constant value and return an error message if the condition is not met. When you use Pega Express to create simple form validations, Pega Express creates the validate rule and applies the rule to the flow action that corresponds to the user view.

You use Designer Studio to create more complex validations that require you to manually update the flow action properties panel.

Validate data in Pega Express

An example of a value comparison is if a payment amount is less than or equal to zero, then return an error message stating that the payment amount must be greater than zero.

To create a validation condition in Pega Express:

1. Select the step to which you want to apply the validation condition.
2. Click **Configure view** to edit the user view.
3. Select the **Conditions** tab to configure the business logic that defines the condition.
4. Under No validation conditions present, click **+Add condition** to make the condition entry fields available.
5. In the **Message** field, enter a message to tell users when the validation fails. In this example, enter **Amount must be greater than zero**.
6. To the right of When, from the **Field** drop-down, select the property or field to test. In this example, select **Service Cost** as the field to test.
7. From the **Comparator** drop-down, select the test to perform. In this example, select **is less or equal to**.
8. Optional: In the **Value** field, provide a comparison value. Certain tests do not require a comparison value. In this example, the selected comparison requires a value, so enter **0**.
9. Click **Submit** to complete the configuration of the validation condition.
10. Click **Save** to update the case type and prevent the user from submitting the form if the payment amount is less than or equal to zero.

The following example displays the completed validation condition.

Review Service Info

View description goes here

Fields Conditions

Validate fields based on these conditions.

1

Action

Show error (do not submit) ▼

Message

Amount must be greater than zero

When

Field

Service Cost ▼

Comparator


is less than or equal ▼

Value

0
+

+ Add condition

KNOWLEDGE CHECK



ANSWER

How is the validation creation process in Pega Express different than the process in Designer Studio?

First, Pega Express automatically applies the validation rule to the flow action that corresponds to the user view. Second, Pega Express only supports comparisons against a constant value.

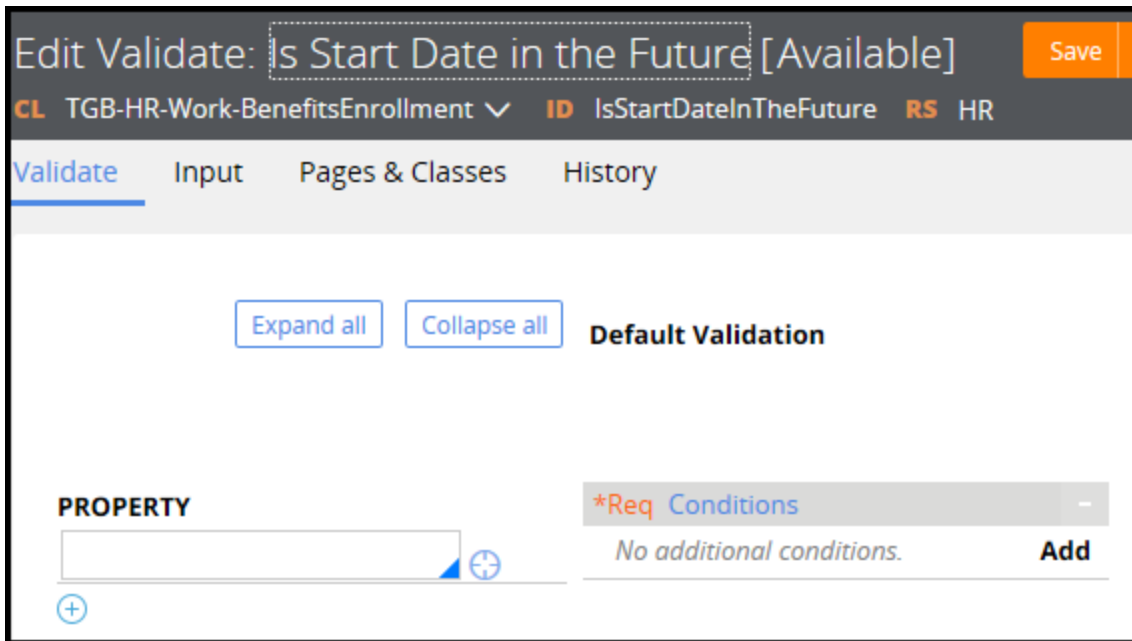
Validate data in Designer Studio

In Designer Studio, you can create validate rules that model more complex validation conditions, such as a condition that compares the results of two fields or a condition that requires a function to determine the comparison value. In addition, you can create a validate rule in Designer Studio to provide an entry condition for a stage in the case workflow.

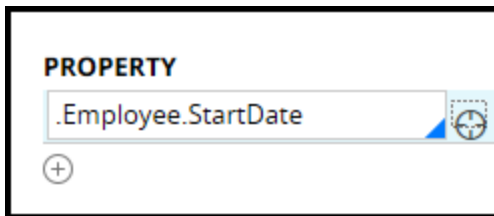
In this video, Designer Studio is used to configure, create, and apply a validate rule that ensures the date entered by the user on a form is later than the current date.

Create the validate rule

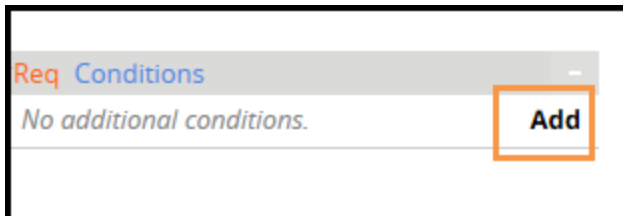
1. In the Application Explorer, right-click the case type in which you want to create the validate rule.
2. Select **Create > Process > Validate** to open a Create Validate form.
3. Enter a specific name in the **Label** field. For example, enter **Is Start Date in the Future** for a rule that validates whether the start date for a new employee is in the future.
4. In the Context area, add the *Apply to* class and ruleset. Then, select **Create and Open**. The Application Explorer displays the Validate form.



5. In the Validate form, enter the property you want to test in the **Property** field. In the following example, the value for `.Employee.StartDate` tests when a user enters a date and submits the form.



6. In the Conditions header, click **Add** to create the validation conditions.



7. In the Validation conditions dialog, enter the following information.

Field	Information
Select a function	[a datetime] is in the [past/future]
If	.Employee.StartDate is in the Past
Message	Start date must be later than the current date

The following image displays the completed dialog.

Validation conditions

Validate .Employee.StartDate Required Enable conditions

Select a function [a datetime] is in the [past/future]

⋮ If

.Employee.StartDate is in the Past

+

Message Start date must be later than cu

Continue validation

8. In the dialog, click **Submit**.

The Validation form displays your updates as shown in the following example.

Expand all Collapse all Default Validation

PROPERTY

.Employee.StartDate

*Req Conditions

IF .Employee.StartDate is in the Past

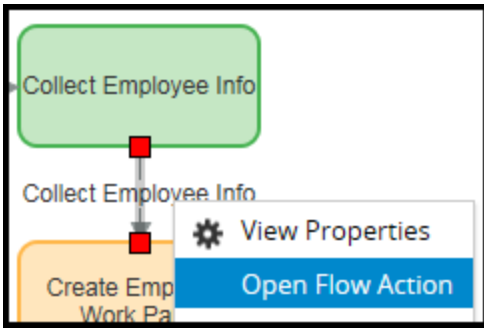
THEN display message:

Start date must be later than current date

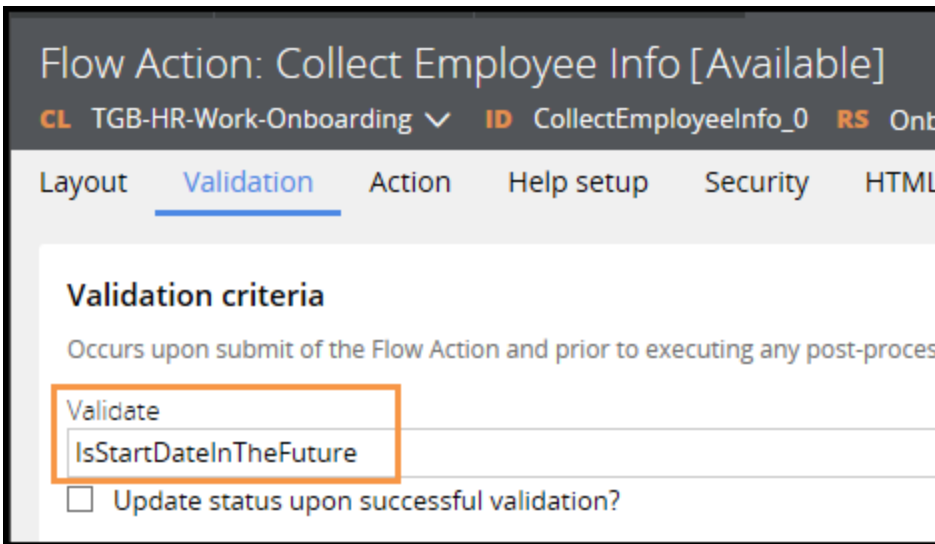
9. In the validate form, click **Save** to complete the configuration of the validate rule.

Associating the validate rule with a flow action

1. On the **Workflow** tab of the Case Designer, click the name of the process you wish to validate to open a contextual panel on the right side of the Case Designer.
2. Click **Open process** in the contextual panel on the right to open the flow rule.
3. Right-click the connector that contains the flow action you want to update and select **Open Flow Action**. The flow action form opens.

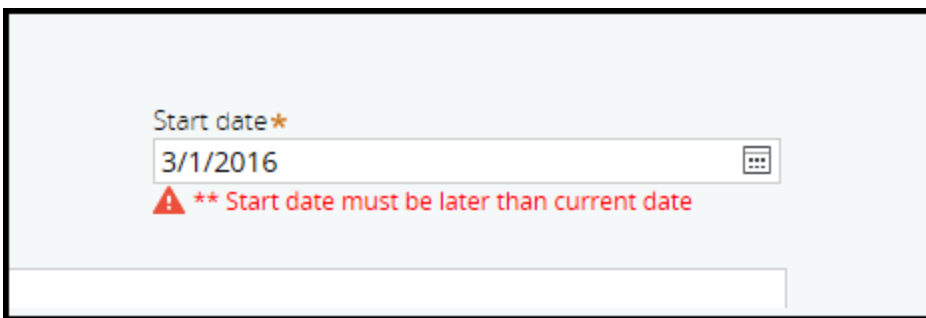


4. On the flow action form, open the **Validation** tab.
5. In the **Validate** field, select your validation rule **IsStartDateInTheFuture**.



6. Click **Save**.

When a user enters a date in the **Start date** field that is not a future date and submits the form, your error message is displayed.



KNOWLEDGE CHECK

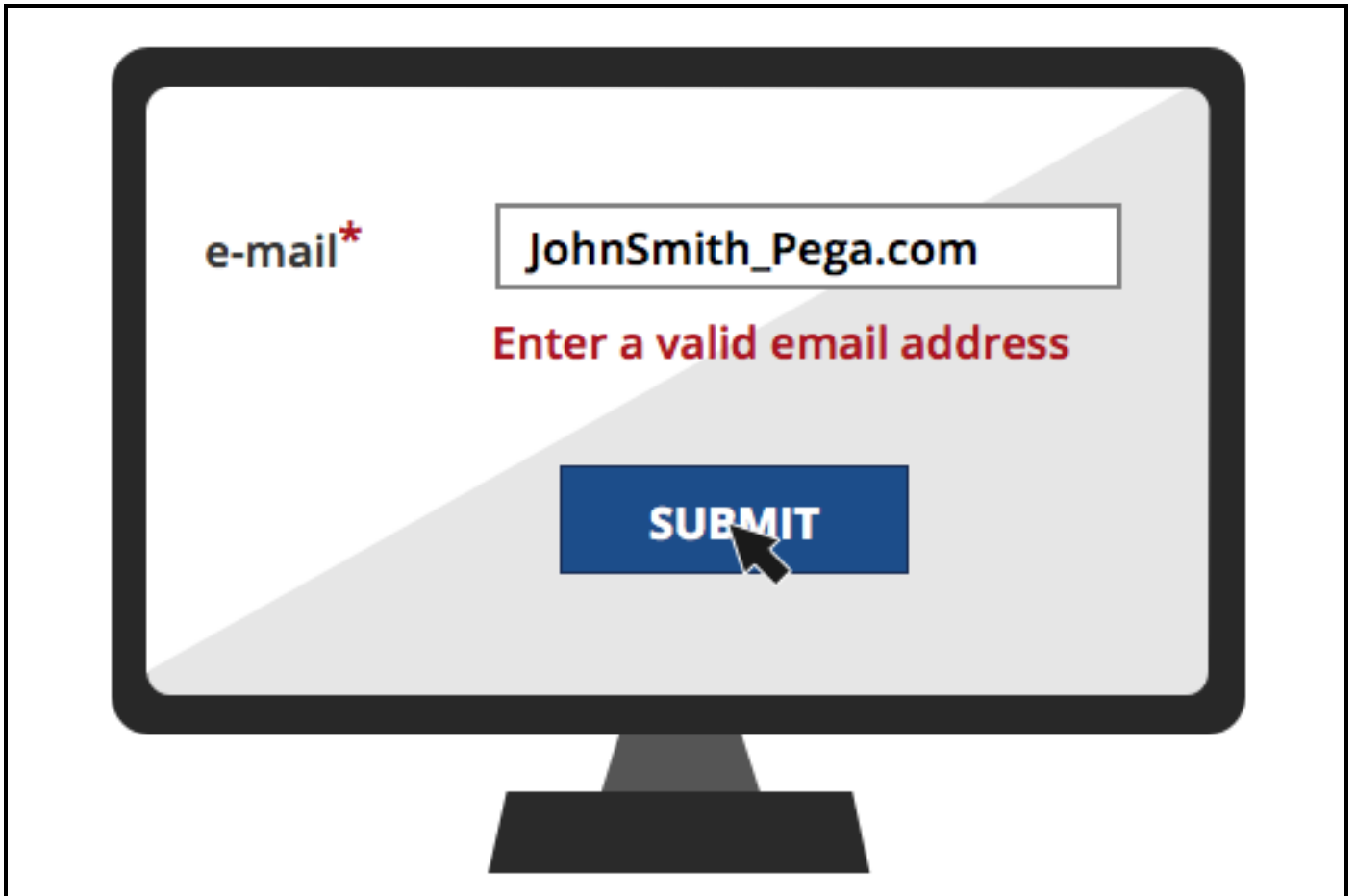


Why must a validation condition that tests whether a date is in the future or the past be configured in Designer Studio?

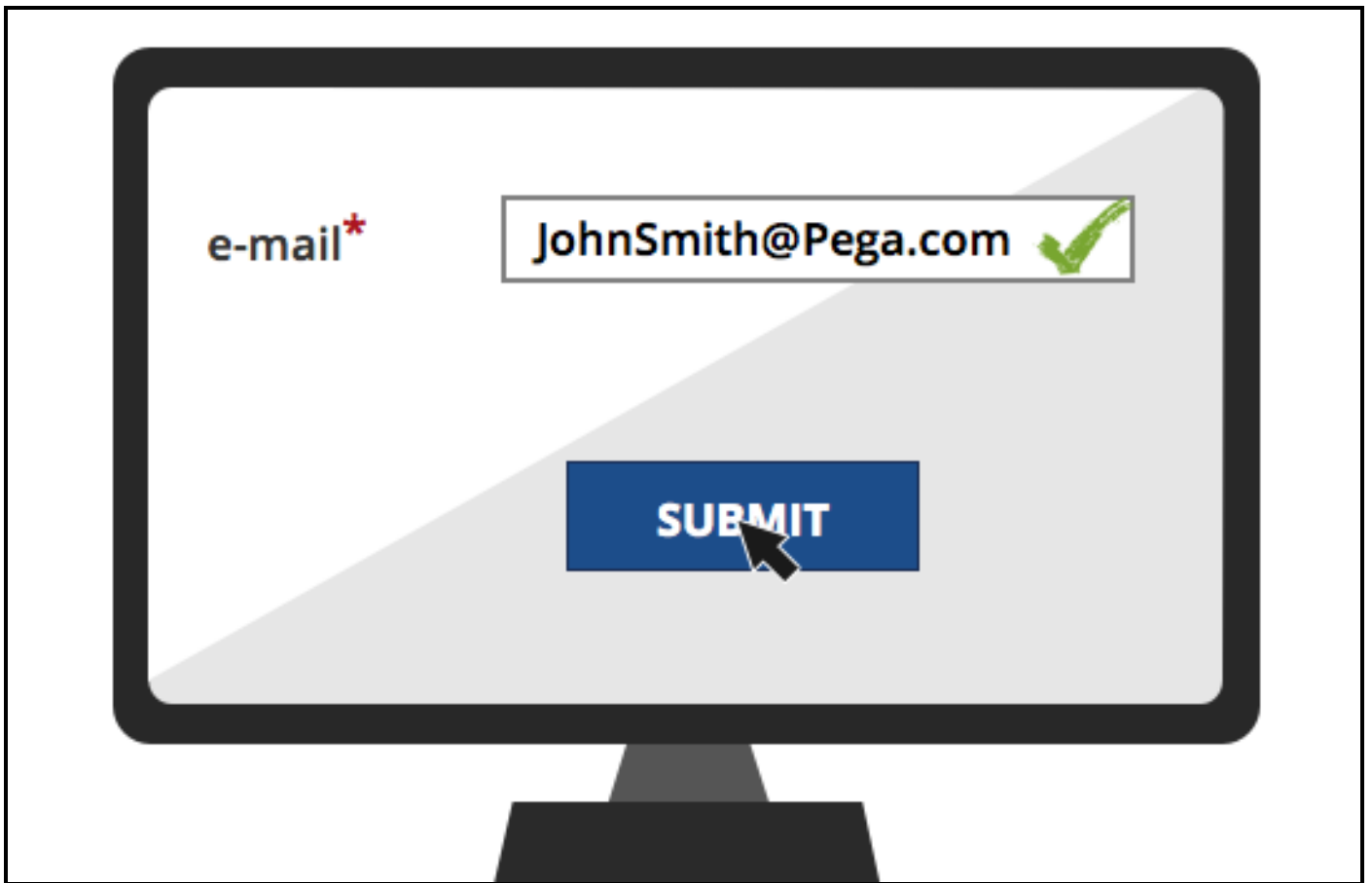
Validating that a date is in the future or the past requires the use of a function.

How to use edit validate rules

Edit validate rules validate character patterns. For example, assume you want users to enter a valid email address in a field using a text property type. However, the field cannot verify that the input value contains an "at" (@) symbol. You can use an edit validate rule to ensure that the field contains the symbol. If it does not, an error message appears when the form is submitted.



When the email address contains the symbol, the error message does not appear.



The logic in edit validate rules is written as a Java procedure. Pega provides a large set of the standard edit validate rules so you do not have to create your own rules.

You can associate a single edit validate rule with a property. You can also reference edit validate rules in a validate rule. This approach enables you to apply multiple edit rules to a single property.

- To associate an edit validate rule with a property, open the property rule. The property must be a single value, value list, or value group. Open the Advanced tab and select an edit validate rule in the **Use validate** field.
- To use an edit validate rule with a validate rule, open the validate rule. In the **Select a function** field, select the function **Validation of [Property Name] using [Edit Validate Name]**. In the **Validation of** field, enter the property you want to validate. Then in the **using** field, select an edit validate rule.

For more information about edit validate rules, see the help topic [About Edit Validate rules](#).

KNOWLEDGE CHECK



You have added a field for entering a U.S. phone number. Do you use an integer data type or an edit validate rule to validate that the phone number is in the correct format?

An edit validate rule ensures that the phone number contains the correct number of digits. The integer data type only ensures that the user enters numbers in the field.

Managing case-processing dependencies

Introduction to managing case-processing dependencies

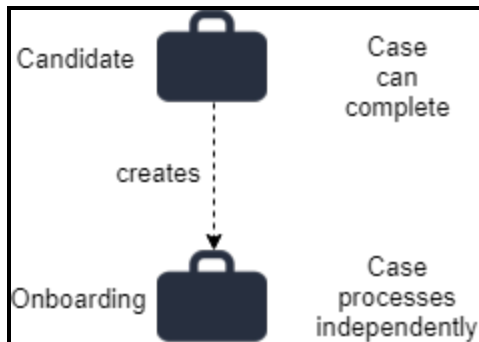
When coordinating work between parent and child cases, you may need to enforce a dependency between a parent case and one or more child cases. For example, you may need to prevent users from advancing a parent case to a new stage until all of its child cases have been resolved. Thus, the processing of the parent case is dependent upon the processing of the child cases.

After this lesson, you should be able to:

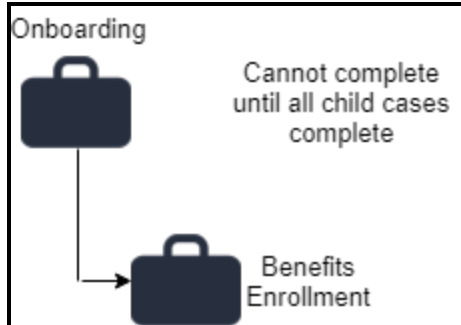
- Identify use cases that justify the use of a child case to manage case processing
- Configure a Create Case shape to create a child case during case processing
- Describe the impact of processing dependencies on the case life cycle
- Explain the role of the Wait shape in controlling the case life cycle
- Configure a Wait shape to enforce a case processing dependency

Case relationships

A business transaction can be complicated and involve multiple cases. For example, consider the new hire process. During the interview process, the human resources (HR) department opens a Candidate case for each job applicant. The applicant may be interviewed. If the interview is successful, the applicant receives a job offer. When the candidate accepts the job offer, HR considers the candidate hired and is now an employee. The Candidate case is completed and creates an Onboarding case to prepare for the new employee's start date. In this example, the Onboarding case is independent of the Candidate case.



Sometimes, the created case is closely related to the original case. In the previous example, part of the onboarding process is to enroll the new employee in a benefits plan. You create a Benefits Enrollment case because it is a separate business transaction. The outcome of the transaction is an employee's benefits plan — a business transaction that is distinct from the onboarding transaction.



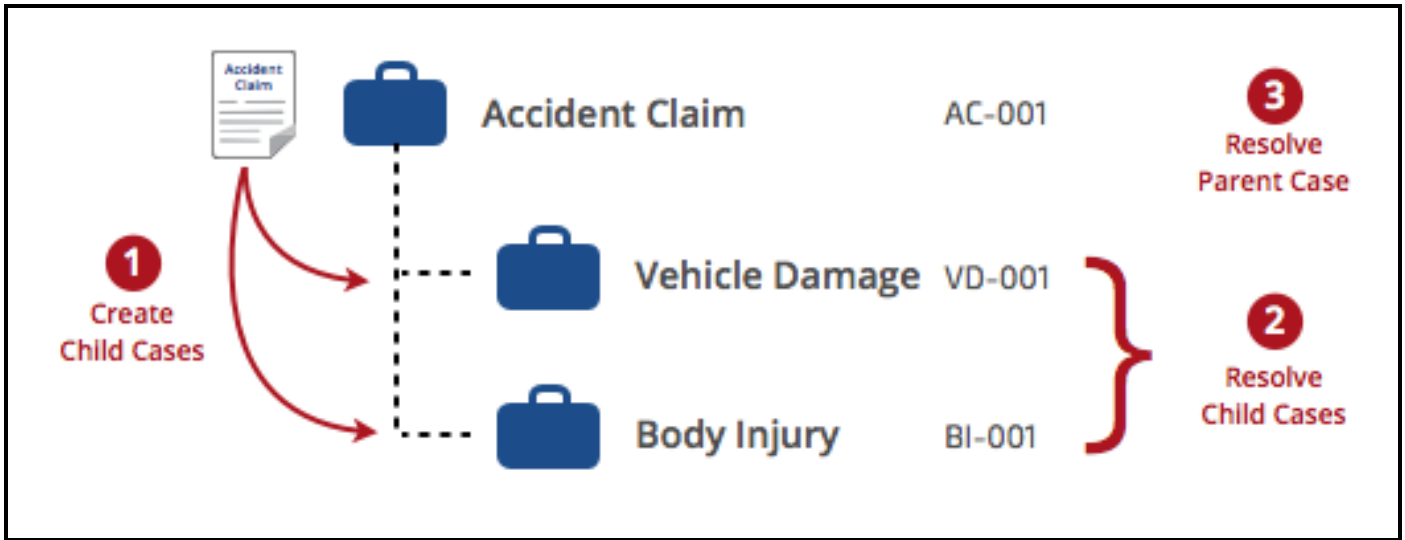
However, before an Onboarding case can be approved the Benefits Plan case must be resolved. In this example, the Onboarding case and the Benefits Enrollment case have a case-subcase relationship. The system associates the Benefits Enrollment information with the Onboarding case. This allows you to join this associated information when reporting or auditing Onboarding cases. For example, you can create a report for a set of employees showing the medical, dental, and vision plans for each employee.

In a Pega application, you can model this case-subcase relationship with a case hierarchy that contains a top-level case and child case.

- **Top-level** — A case that does not have a parent case, but can cover, or become a parent of, other cases.
- **Child**— A case that is covered by a parent case. When you configure a case as a child case, Pega maintains a relationship between the parent and child cases. Child cases represent work that must be completed to resolve the parent.

Note: Use case type records to establish the parent child relationships.

For example, an Auto Insurance application has a Accident Claim top-level case. The Accident Claim includes two child cases — Vehicle Damage and Bodily Injury. For any Accident Claim case, both of its child cases — vehicle damage and bodily injury — must be addressed before the Accident Claim can be closed. In addition, reports can associate a parent case with any or all of its child cases.



A parent case creating multiple child cases allows for work to be processed in parallel. Each child case can be handled by different parties with different expertise. When processing child cases in parallel, the parent case may need to wait until a child case is complete before the parent case can be resolved. Under the cover of an Accident Claim case, the Vehicle Damage child case can be handled between a customer service representative, an adjustor, and a repair shop. Meanwhile, the Bodily Injury child case can be handled by a medical claim specialist and certain medical providers.

Implementing a business process in a separate case type also allows you to reuse the case type as needed. For example, claims for both automobile and property insurance may involve a bodily injury claim. By implementing bodily injury claims as a separate case type, you can use the bodily injury case type with both automobile and property claims.

KNOWLEDGE CHECK

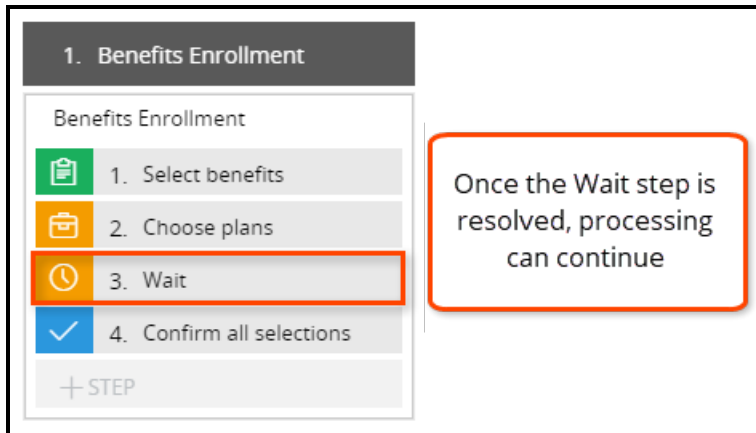


What is the difference between a top-level case and a child case?

A child case has a relationship with the parent case and a top-level case does not.

How to enforce a dependency between case types

You can enforce dependencies between parent and child cases with the Wait step. When a parent case reaches the Wait step, the case pauses until the dependency resolves. Once the dependency resolves, the case resumes processing.



The Wait step enforces the following types of dependencies:

- Pausing a case until another case (or all cases) reach a specified status
- Pausing a case until a predetermined time expires

You create a case dependency by setting the Wait step to pause for another case. The Wait step pauses until all cases or any case of a given type reach a defined status. The status could be a standard status like Resolved-Completed or a custom status defined in your application. You can also set the Wait step to **To be resolved**, where a case is resolved when the case status is set to a value that starts with the word Resolved.

Note: You are only able to define a single status for the Wait step to check. If you encounter a situation where you need to check multiple statuses, consult the [Wait Shape for mid process dependency](#) discussion on the PDN.

You can also configure the Wait step to process for a predetermined amount of time. In this situation, you either specify a specific length of time for how long the case should pause, or use a variable that contains a date when the case should resume. For example, with an accident claim, you may give someone 24 hours to upload pictures for the accident. You would want to pause the case until the wait time has elapsed.

Note: The ability to set the Wait step as a timer is only available in Designer Studio.

KNOWLEDGE CHECK



What are the two ways you can configure a Wait step to pause a case?

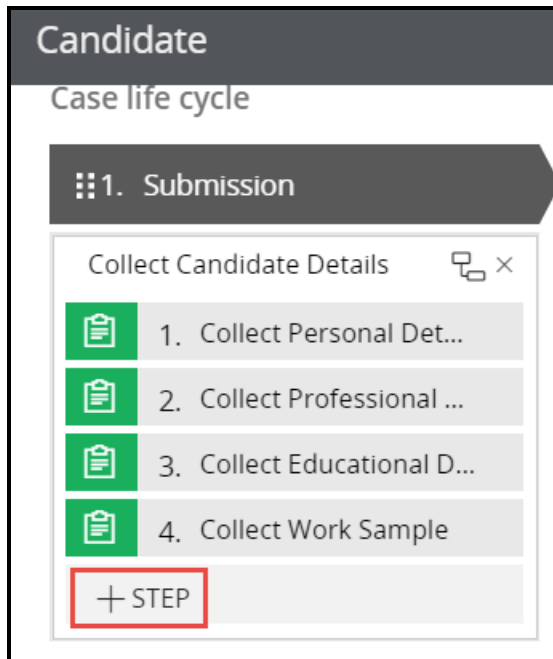
You can configure a case to wait for a set amount of time or until a subcase reaches a defined status.

Adding a child case to a case

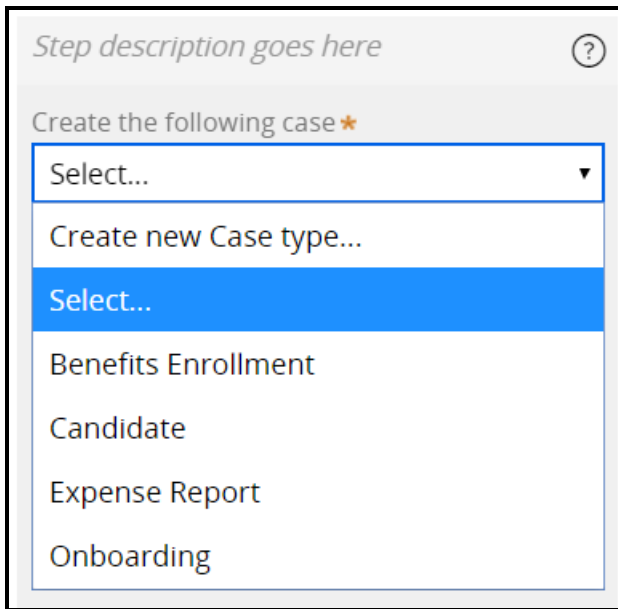
You use Pega Express to add child cases to a case. To add a child case to a case type you add a **Create Case** step to your case. The **Create Case** step creates an instance of the configured case type when the step is executed.

To add a create case shape:

1. In the Case Designer, click **+ STEP > ... More Utilities > Create Case > Select** to add a Create Case step to the Case life cycle.



2. In the Contextual Properties panel (right side of your screen), configure the step by specifying which case to create.



3. Click **Save** to update the case.

Note: Designer Studio gives you one additional option to create a case as a top level case instead of a child case.

MODELING CASE DATA

The building blocks of a Pega application

Introduction to the building blocks of a Pega application

From a business perspective, the key building block of a Pega application is a case. A case is work that delivers a business outcome, for example, creating an expense report or onboarding a new employee.

From a technical perspective, the key building block of a Pega application is a rule. A rule is an object that defines the behavior of a case, or part of an application.

When you create a case, Pega will create the rules that provide the required behavior. If you need to further define that behavior, you can directly modify or create a rule.

You can reuse existing assets (for example, cases and rules) to limit development of new assets. Pega delivers an impressive array of application assets.

When an application requires new assets, you only need to create key features and functions once. Pega's inheritance structure lets you reuse the new resources wherever needed in your application. Eliminating redundant assets simplifies maintaining and extending the application.

In this lesson, you learn how Pega manages rules and how you can reuse rules through application design and Pega's principle of inheritance.

After this lesson, you should be able to:

- Describe the relationship between an application and rules
- Differentiate between a rule and a rule type
- Explain the principles of rule inheritance and scope
- Differentiate between pattern inheritance and directed inheritance
- Update inheritance for a class

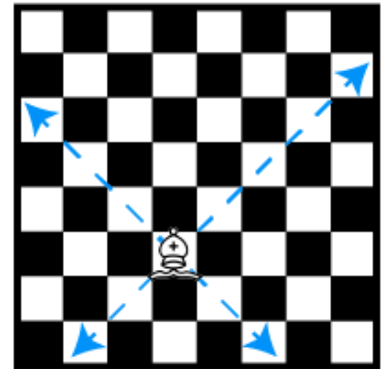
Rules and rule types

When you play a game of chess, you and your opponent agree to follow a specific set of instructions. These instructions govern game play, such as how each piece moves on the game board. These basic instructions are the rules of chess.

When you model a case type in a Pega application, you configure the application with instructions to create, process, and resolve a case. These instructions are **rules**. Rules describe the behavior of individual cases. The Pega platform uses the rules you create to generate application code.

Each rule is an instance of a **rule type**. A rule type is an abstract model of a specific case behavior. Pega provides many rule types. For example, Pega provides one type of rule to describe a process flow, and another type of rule to describe an automated email notification.

You can think of a rule type as a template for creating a rule. When you create a new rule, you first select the rule type, which determines the properties that are required to implement the associated case behavior.



KNOWLEDGE CHECK

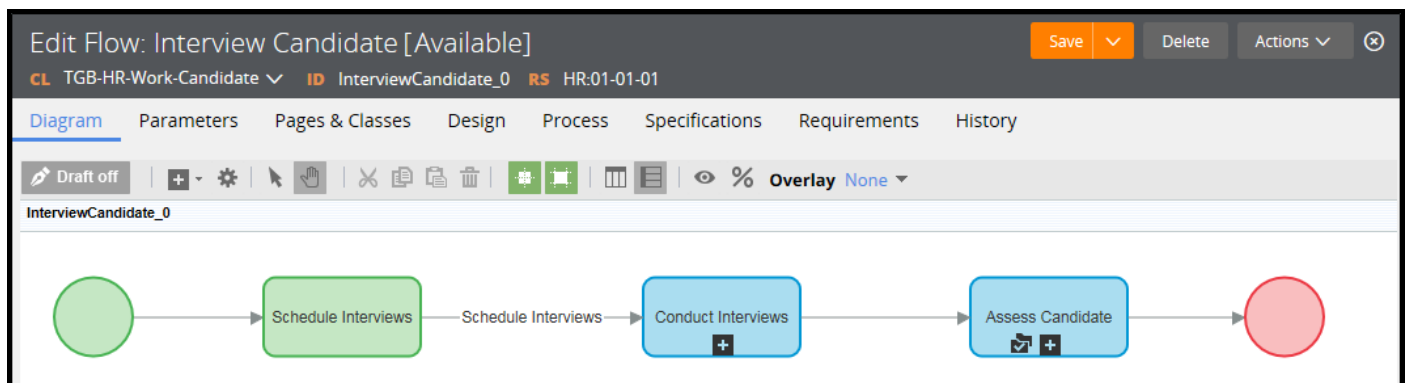


What is the purpose of a rule in a Pega application?

A rule is an instruction for describing a specific case behavior, such as a process or automated decision.

Pega provides wizards that create and modify many of the rules in an application for you. For example, when you configure a case in Pega Express, it generates rules to describe cases, processes, and UI forms. Much of the work of designing an application can be completed by using these wizards. If you need more control over how an application creates and reuses a rule, you can access a rule directly in Designer Studio.

The following screenshot shows an example of a flow rule. Pega Express and the Case Designer create a flow rule whenever you add a process to a case life cycle.



The use of individual rules makes your application modular. By describing case behavior with modular, task-focused rules, you can combine and reuse rules as needed. In this manner, rules are analogous to classes in Java or other object-oriented programming languages. For example, you create a rule to describe the content of an email to send to a customer regarding the status of a change of address. Your application sends an automated email after the customer enters the old and new address. By creating the message as a separate rule, rather than embedding the message in the case life cycle, you can update the content of the email without impacting the rest of the business process.

This modularity provides three significant benefits:

1. **Versioning** – System architects create a new version of a rule whenever case behavior needs to change. Pega maintains a history of changes to a rule, allowing system architects to review the change history and undo changes if needed. Since each rule describes a specific case behavior, the rest of the case remains unaffected. For example, a system architect updates a UI form with instructions and removes a critical field. You can review the history of the form and revert back to the version before the changes were made, without changing other rules in the application.
2. **Delegation** – System architects delegate rules to business users to allow business users to update case behavior as business conditions change. The business user updates the delegated rule, while other parts of the application remain unchanged. For example, expense reports that total USD25 or less receive automatic approval. You create a rule to test whether an expense report totals USD25 or less and delegate the rule to the Accounting department. The Accounting department can then update the rule to increase the threshold for automatic approval increases to USD50, without submitting a change request for the application.
3. **Reuse** – System architects reuse rules whenever an application needs to incorporate existing case behavior. Otherwise, you must reconfigure the behavior every time you need the behavior. For example, you create a UI form to collect policyholder information for auto insurance claims. You can then reuse this UI form for property insurance claims and marine insurance claims.

Rules and rulesets

To package rules for distribution as part of an application, you collect rules into a group called a **ruleset**. A ruleset identifies, stores, and manages the set of rules that define an application or a major portion of an application. If a rule is similar to a song, a ruleset is similar to an entire album. Just as you can copy the album to share with a friend and allow your friend to listen to your favorite song, you can share a ruleset between applications to allow several applications to use the same rules. The ability to re-use already-built rules saves development time and effort.

Ruleset versioning

System architects collect individual rules into an instance of a ruleset, called a **ruleset version**. To update the contents of the ruleset, you create a new ruleset version. Ruleset versioning allows system architects to easily update applications by providing access to an entire set of rules at once.

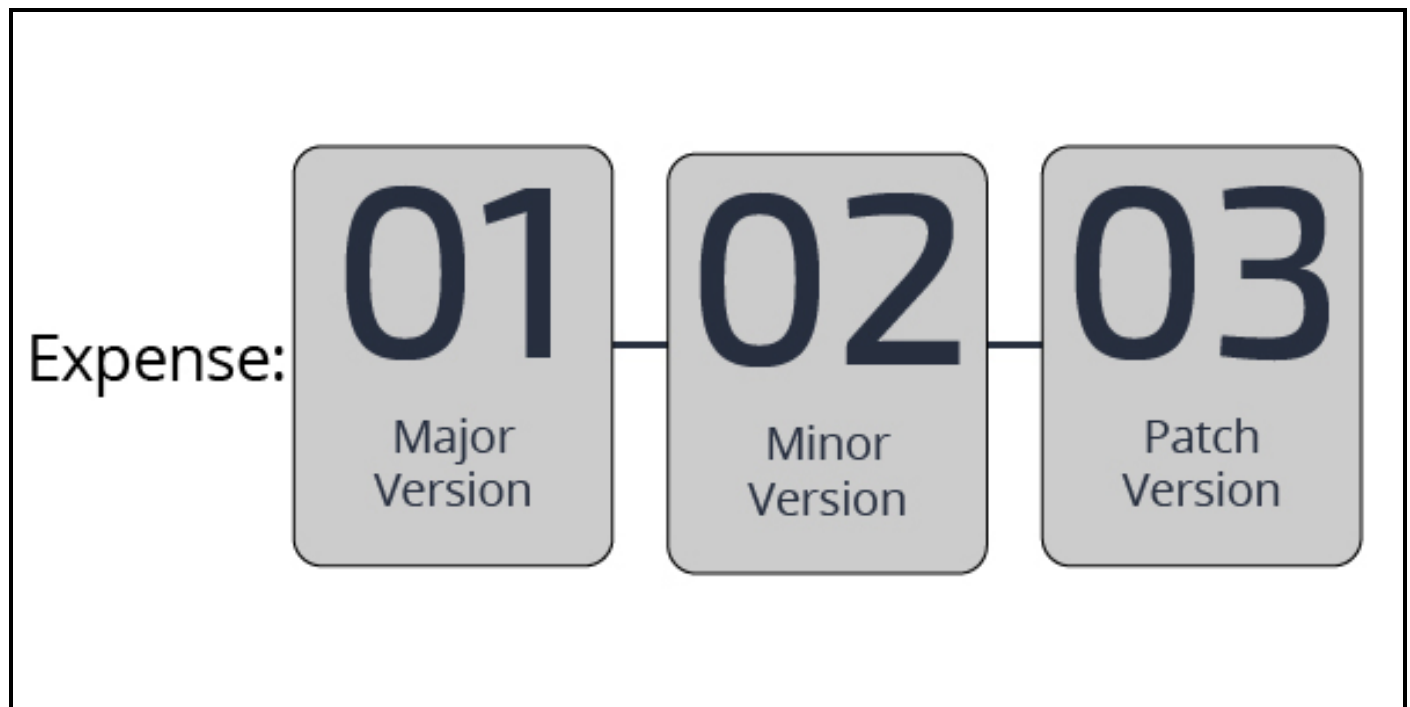
Note: If you use Pega Express to develop your application, note that Pega Express manages the creation of rules for you and identifies the ruleset and version in which to store a rule.

You identify each ruleset by its name and version number. For example, an application to process expense reports includes a ruleset named Expense. You refer to the ruleset as Expense:01-02-03, where Expense is the name of the ruleset and 01-02-03 is the version number.

The version number is divided into three segments: a **major version**, a **minor version**, and a **patch version**.

- The **major version** represents a substantial release of an application. A major version change encompasses extensive changes to application functionality. For example, the Accounting department uses an application to manage expense reports. If Accounting wants to extend the application to track employee time off for payroll accounting, you create a new major version of the ruleset.
- The **minor version** represents an interim release or enhancements to a major release. For example, you need to update an expense reporting application to make automatic audit travel reimbursements. You create a new minor version of the ruleset.
- The **patch version** consists of fixes to address bugs in an application. For example, you notice that a field in the current version of an application has an incorrect label. You create a new minor version to correct the field label.

Each segment is a two-digit number starting at 01 and increasing to 99. Ruleset version numbering starts at 01-01-01, and increments upward.



Ruleset stack

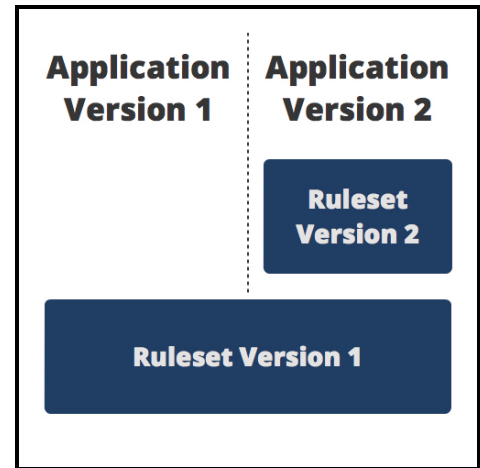
Each application consists of a sequence of rulesets, called a **ruleset stack**. The ruleset stack determines the order in which Pega looks through rulesets to find the rule being used. Each entry in the ruleset stack represents all the versions of the specified ruleset, starting with the listed version and working down to the lowest minor and patch version for the specified major version.

Each version of an application contains a unique ruleset stack. This allows an updated application to reference new ruleset versions that contain updates and new features.

Bob is a system architect working on the first version of an application to manage expense reports. Bob creates rules for the first version of the application, such as processes, UIs, and notifications. Bob collects these rules into the first version of the Expense ruleset, Expense:01-01-01.

Months later, Tanya receives an enhancement request to update a UI in the application to collect extra information from employees due to a policy change. This update enhances the rules created earlier by Bob. Tanya creates rules to model this new behavior in a second version of the ruleset, Expense:01-02-01. She then uses the Expense:01-02-01 ruleset in the updated expense reporting application.

Employees who use the first version of the application view the UI that Bob created. Only employees who use the updated application view the UI that Tanya created. The ruleset stack for the first version does not include Tanya's version. This allows users to use the first version of the application while the second version is in development.



KNOWLEDGE CHECK



A ruleset version is identified with a string of three numbers. What do these three numbers indicate?

The three numbers used to identify a ruleset version indicate the major version, minor version, and patch version of the ruleset.

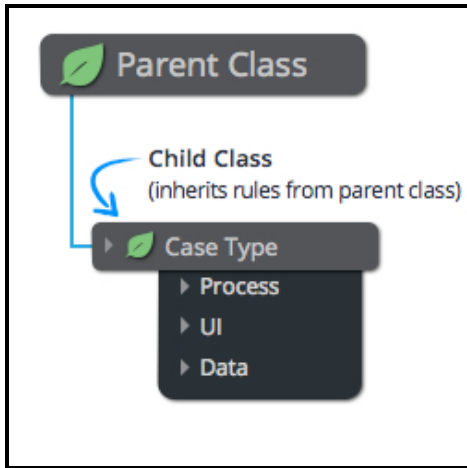
Classes and class hierarchy

One strength of the Pega platform is the reuse of rules between case types and applications. System architects often reuse rules—from single data elements to entire processes—in applications. The reuse of rules improves application quality and reduces development time. Organizations that adopt the Pega Platform have reduced development costs by 75 percent and time-to-market by 50 percent, launching new business applications up to 90 days earlier ⁽¹⁾.

Within an application, Pega groups rules according to their capacity for reuse. Each grouping is a **class**. Each application consists of three types of classes.

- **Work class** contains the rules that describe how to process a case or cases, such as processes, data elements, and user interfaces.
- **Integration class** contains the rules that describe how the application interacts with other systems, such as a customer database or a third-party web server.
- **Data class** contains the rules that describe the data objects used in the application, such as a customer or collection of order items.

Note: When you create a rule in Pega Express, Pega Express identifies the appropriate class for you. You can focus on what you want the rule to do, rather than on how to create the rule. If you need control over the class, you can use Designer Studio to create the rule. The main reason for switching to Designer Studio is to create a rule that you plan to reuse in a different application.

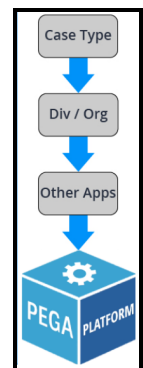


A class can also contain other classes. A class that contains another class is called a **parent class**, while a class that is contained by another class is called a **child class**. A child class can reuse, or **inherit**, any of the rules defined for its parent class.

The work class contains a child class for each case type in your application. Each child class contains all of the rules unique to a case type, such as an auto insurance claim. The data class contains a child class for each data object.

The classes that comprise an application are called a **class hierarchy**. The class hierarchy determines how system architects can reuse rules within the application. The class hierarchy consists of several groups of classes:

- Classes that describe a specific case type, such as expense reports or auto insurance claims.
- Classes that collect common rules and data elements. These classes allow the reuse of rules at the division and organization level, such as an approval process shared across the entire IT department.
- Classes from other applications, such as industry-specific Pega applications. You can use a generic application for policy administration as a base for customizing versions for different policy types.
- Base classes provided by the Pega platform. These classes contain rules that provide basic functionality for processing cases. For example, the Pega platform provides data elements that record who created a case and the time needed to complete an assignment.



Any rule available to an application through the class hierarchy is considered in scope. Rules that an application cannot access through the class hierarchy are considered out of scope.

Pega names each class to identify the position of the class within the class hierarchy. Consider the class *TGB-HR-Work*. Each level of the class hierarchy is separated by a hyphen (-). So *TGB-HR-Work* is a child of the class *TGB-HR*, which is a child of the class *TGB*.

KNOWLEDGE CHECK



What is the purpose of a class in a Pega application?

A class organizes rules within an application. The position of a class within the class hierarchy determines the reusability of the rules in that class.

¹Forrester Consulting. (2015). *The Total Economic Impact™ Of The Pega 7 Platform*. Retrieved from <https://www.pegacom/forrester-tei>

How to create a rule

When you create a rule, Pega provides you with the New Record form. The New Record form allows you to create either a rule or a data instance.

Note: When you configure application behavior in Pega Express (for example, when you create a case), Pega Express creates and manages the underlying rules for you. If you need more control over how a rule is created and reused, you can create or edit the rule in Designer Studio.

When you create a rule, the New Record form prompts you to provide four pieces of information: rule type, identifier, class, and ruleset. This information is used to identify the rule uniquely within your application.

The screenshot shows the 'Create Flow' form in Pega Designer Studio. The form is titled 'Create Flow' and has a '1' in a red circle next to the title. It contains several sections: 'Flow Record Configuration' with a '2' in a red circle next to the 'Label*' field, which has a placeholder 'Describe the purpose for this new record' and a subtext 'A short description or title for this record'. To the right of the label field is the 'Identifier' field with the value 'To be determined'. Below the label field is a link 'View additional configuration options'. The 'Context' section has a '3' in a red circle next to the 'Apply to*' field, which has a dropdown menu. To the right of the 'Apply to' field is the 'Add to ruleset*' field, which has a '4' in a red circle next to it, and contains two dropdown menus with values 'Onboarding' and '01-01-01'. There is also a 'View all' link below the 'Apply to' field.

1. The **rule type** determines the type of behavior modeled by the rule. Each rule type models a specific type of behavior, such as automated decisions, UI design, or data storage. To model a process, for example, you use a specific type of rule called a flow rule. You determine the rule type when you open the New Record form.
2. The **identifier** identifies the purpose of the rule. For example, to model the process for approving insurance claims, you use an identifier such as ClaimsApproval. This identifier allows you to differentiate the approval process from a submission process. Pega determines the identifier from your entry in the **Label** field.
3. The **class** identifies the scope of the rule. You specify the class of a rule in the **Apply to** field. The class you select determines the extent of how you can use the rule: within one case type, or across case types.

4. The **ruleset** is the container for the rule. The ruleset identifies, stores, and manages the set of rules that define an application or a major portion of an application.

The combination of rule type, name, class, and ruleset allows Pega to provide a unique identity for each rule. This combination allows an application to call the correct rule during case processing, through a process called **rule resolution**. With rule resolution, Pega determines the appropriate rule to run when an application calls a rule.

You can access the New Record form several ways. Based on your choice, Pega provides default values in some or all of the fields on the form. You can change these values before you create the rule.

How to access the New Record form	Default values provided
From the +Create menu, select the rule category, then the rule type.	Rule type, ruleset
In the Application Explorer, select the class in which you want to create the rule, then select the rule category, then select the rule type.	Rule type, apply to class, ruleset
In a field on a form, enter the name of the rule to create, then click the Target icon.	Rule type, identifier, apply to class, ruleset

After you complete the New Record form, click **Create and open** to configure the rule behavior.

How to update a rule

System architects often secure rulesets to prevent unauthorized or unintended changes to rules. When you edit the rules in a secured ruleset, you either check out the rule or perform a private edit.

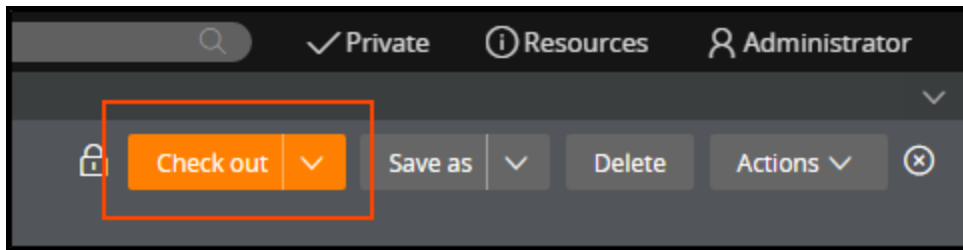
Rule check out and check in

The check-out feature is used to manage changes to rules when multiple developers work on an application. This feature allows a system architect to update a rule while preventing updates by other system architects. Rule check-out creates a copy of a rule in a ruleset that is only visible to you, called a **personal ruleset**. After you update the rule and test the changes, you check in the rule. This updates the application ruleset with a new version of the rule.

Note: When updating a rule in Pega Express, Express automatically manages the check-out/check-in process for you.

Checking out a rule

On the rule form header, click **Check out** to check out the rule.



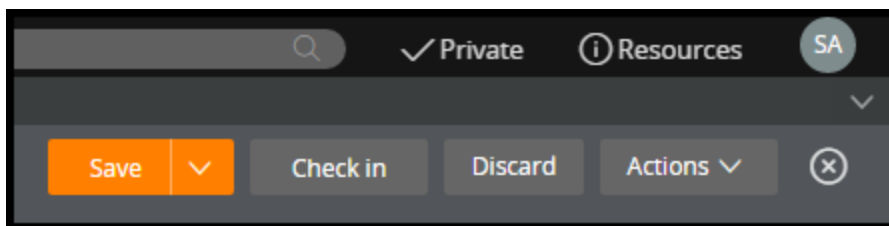
Checking out a rule creates a copy of the original rule in your personal ruleset and prevents other system architects from checking the rule out until you check in your changes.

The personal ruleset occupies the top spot in the ruleset stack. The rules in your personal ruleset override rules in the rest of the application. This allows you to test your changes to the rule without affecting other system architects.

In the rule header, click **Private** to view a list of the rules you have checked out.

Checking in a rule

When you check out a rule, the rule header updates with three new buttons: **Save**, **Check in**, and **Discard**.



When you finish editing the rule, click **Save** to save your changes to the checked out rule. This commits the updated rule to your personal ruleset. After you save the rule, you can test your changes.

After you test the rule and confirm that your configuration works as expected, click **Check in** to replace the original rule with the version in your personal ruleset. Unless approval is required, your changes immediately affect application behavior.

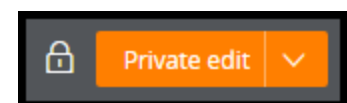
You are not required to check in your changes immediately. You can log off and return to a checked out rule later or click **Discard** to remove the rule from your personal ruleset.

Select **Private > Bulk actions** to check in several records at the same time.

Private edit

A **private edit** provides a nonexclusive check out of a rule. This allows other system architects to edit a rule at the same time. Private edits are useful for quick debugging without interrupting development by other team members. This option is not available in Pega Express.

It is best practice to lock older versions of a ruleset in order to prevent changes. For rules in a locked ruleset, a lock icon is displayed on the rule form. To update a rule in a locked ruleset version, save the rule to an unlocked ruleset version, then check out the rule if necessary.



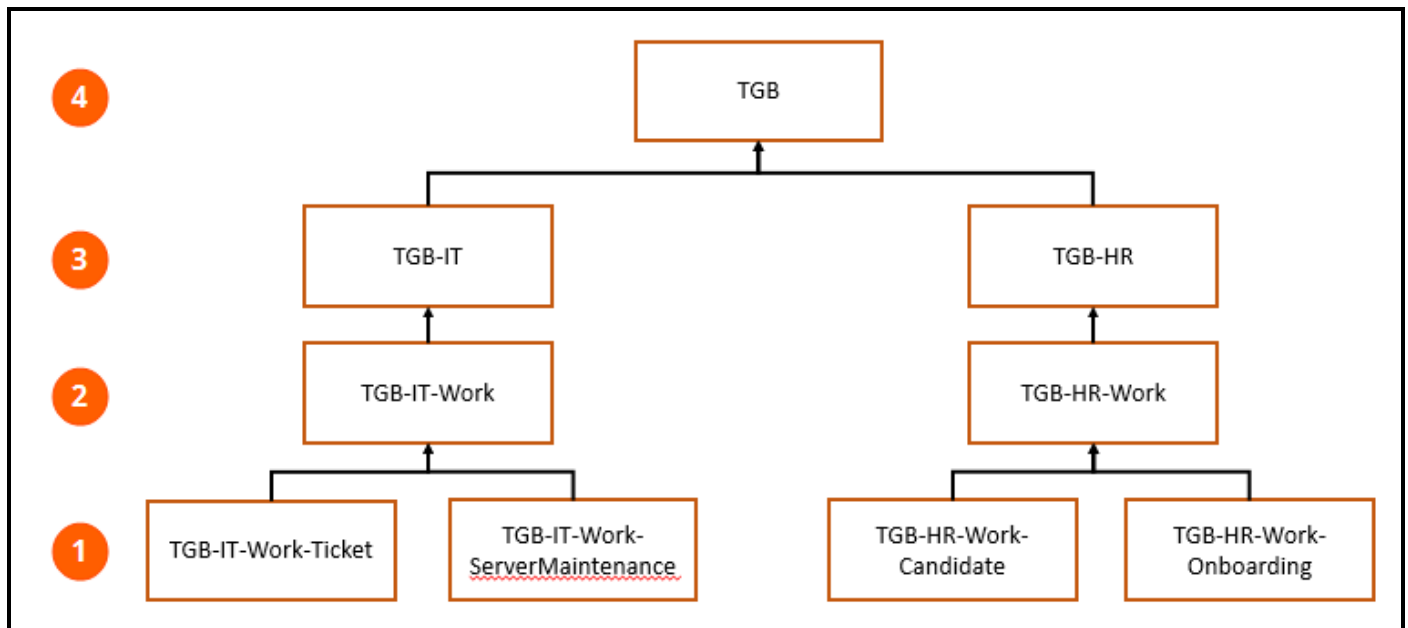
How to reuse rules through inheritance

Inheritance allows your application to reuse rules that have already been created for other cases or applications. By reusing a rule through inheritance, rather than creating an identical copy of the rule, you reduce development and testing time without sacrificing application quality.

Pega provides two methods for inheriting rules: **pattern inheritance** and **directed inheritance**.

Pattern Inheritance

Pattern inheritance describes the business relationship between classes. Pattern inheritance allows your application to share rules with other applications throughout an organization. The following image demonstrates a basic pattern inheritance hierarchy.



1. Rules for a specific type of case are stored at the lowest level of the hierarchy. Rules at this level only affect a single type of case, such as IT service tickets or onboarding requests.
2. The next level is the **class group**. The class group contains all of the case types in an application. In the previous image, TGB-IT-Work contains all of the case types for the IT department, while TGB-HR-Work contains all of the case types for the human resources (HR) department. Rules at this level affect all the case types in the class group.
3. Above the class group is the **division layer**. The division layer contains the work, data, and integration classes for the division.
4. Above the division layer is the **organization layer**. The organization layer contains all of the classes for applications across an entire business or other organization. The organization layer often contains data and integration classes that can be applied across the entire organization.

For example, an organization creates an application to manage IT requests. In this application, you use a data element to record the due date for the request. The concept of a due date is not unique to IT requests. Other business processes also use due dates, such as expense reports. If you create the data

element for the due date in the organization layer, an application used by the Accounting department to track expense reports can reuse this data element.

KNOWLEDGE CHECK



What type of relationship is described by pattern inheritance?

Pattern inheritance describes the business relationship between classes. Pattern inheritance indicates the reusability of rules throughout an organization, such as whether a rule is usable by a single case type, an entire department, or even an entire organization.

Directed inheritance

Directed inheritance describes the functional relationship between classes. Directed inheritance allows your application to reuse rules from classes in other applications and standard rules provided with the Pega platform. For example, a class that describes automobile insurance policies can inherit from a class that describes a generic insurance policy, and even the generic case type defined by the Work-Cover class provided by the Pega platform.

Organizations can get the benefits of reuse by building on existing Pega applications, which are built on the Pega Platform. For example, directed inheritance allows you to reuse the standard data element to record the case ID, provided as part of the Pega platform, in your application.

Important: Directed inheritance is the only option that allows an application class to inherit rules defined for standard Pega classes, such as the *Work-* or *Data-* class.

KNOWLEDGE CHECK



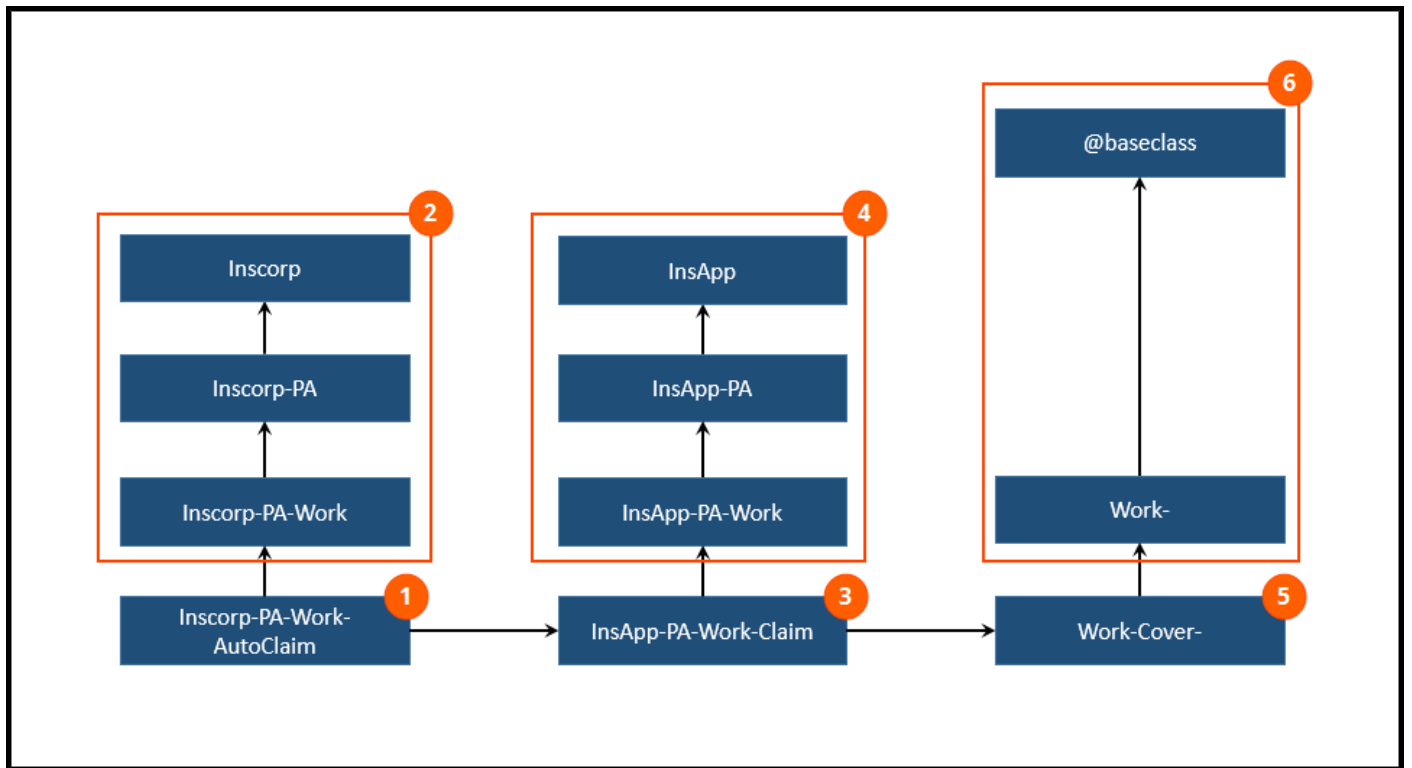
How does directed inheritance differ from pattern inheritance?

Pattern inheritance allows you to reuse rules within a single application. Directed inheritance allows you to reuse rules in other applications, including standard rules provided as part of the Pega platform.

Reusing rules through inheritance

When attempting to reuse rules through inheritance, Pega first searches through the parent classes indicated by pattern inheritance. If unsuccessful, Pega then searches the parent class indicated by directed inheritance as the basis for another pattern inheritance search. This process repeats until Pega reaches the last class in the class hierarchy, called the ultimate base class or *@baseclass*. If the rule cannot be found after searching *@baseclass*, Pega returns an error.

Consider the following example in which an auto insurance claim case references the data element that stores the case ID. This data element belongs to the ultimate base class, *@baseclass*. The application containing the auto insurance claim is built on a generic policy administration application. That generic application is built upon the Pega platform.



1. An auto claim case, described by the class *Inscorp-PA-Work-AutoClaim*, references the case ID data element.
2. The data element is not found in the class *Inscorp-PA-Work-AutoClaim*, so Pega searches through the parent classes using pattern inheritance.
3. The data element is not found though pattern inheritance, so Pega searches the parent class specified by directed inheritance, *InsApp-PA-Work-Claim*. This class belongs to the generic policy administration application.
4. The data element is not found in the class *InsApp-PA-Work-Claim*, so Pega searches its parent classes using pattern inheritance.
5. The data element is not found though pattern inheritance, so Pega searches the parent class specified by directed inheritance, *Work-Cover-*. This class belongs to the Pega platform.
6. The data element is not found in the class *Work-Cover-*, so Pega searches its parent classes using pattern inheritance, finally locating the data element in *@baseclass*.

KNOWLEDGE CHECK



From which class does *@baseclass* inherit rules?

None. In a Pega application, *@baseclass* is the ultimate base class. All other classes inherit from *@baseclass*.

Data elements in Pega applications

Introduction to Data Elements in Pega Applications

Pega applications allow users to create, process, and resolve cases. The applications collect data that is important to the case. Based on the data collected, decisions on how to best process and resolve the case are made.

For example, if you want to create a case to process a change of address for a customer, you need data. The data includes the identity of the customer and the new address.

After this lesson, you should be able to:

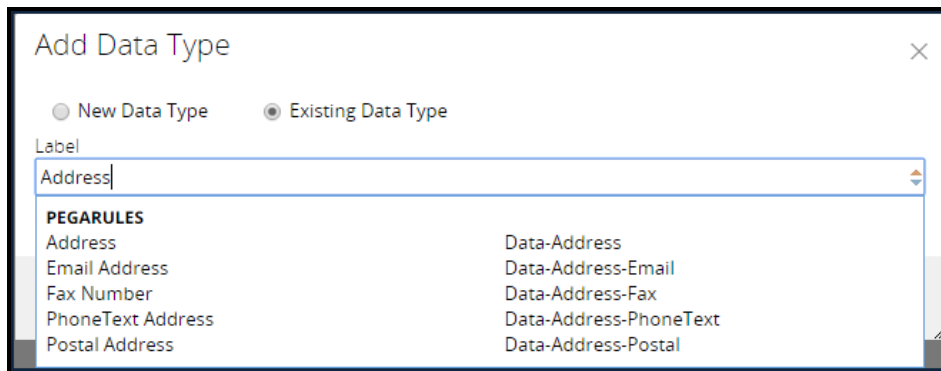
- Explain the role of the Data Explorer in managing data types
- Reference properties in Pega applications
- Define the components of a data model

How to manage properties

Pega Platform provides several tools that help manage properties. The tools provide easy-to-use interfaces that add, update, and remove classes and properties. This section looks at the Data Explorer and the Data Model tab as well as the property rule form.

The Data Explorer

Use the Data Explorer to add or remove data types. In Pega Express, the Data Explorer allows you to create a new data type and establish its position within the class hierarchy. In Designer Studio, the Data Explorer supports the reuse of existing data types, such as the data types provided as part of Pega Platform. Always check if a suitable data type is available before creating a new one. Pega comes with many standard data types you can use in your application to reduce development effort. To reuse an existing data type, click **Existing Data Type** and select the data type you want to use.



You can extend an existing data type if it only partly meets your needs. For example, you might want to create an employee data type based on the Party-Person data type. Select a new data type and specify the data type you want to extend as the parent in the advanced settings. You can use all properties defined in the parent in addition to the ones you create in your new data type.

Add Data Type

New Data Type Existing Data Type

Label*
Employee

Description*
Employee

▼ ADVANCED

Parent class
Data-Party-Person

Identifier: Data-Party-Person-Employee [Edit](#)

Development branch
[No branch]

Choose app layer
 Purchasing
 PegaRULES

Add to ruleset*
MyCo

Cancel Submit

KNOWLEDGE CHECK

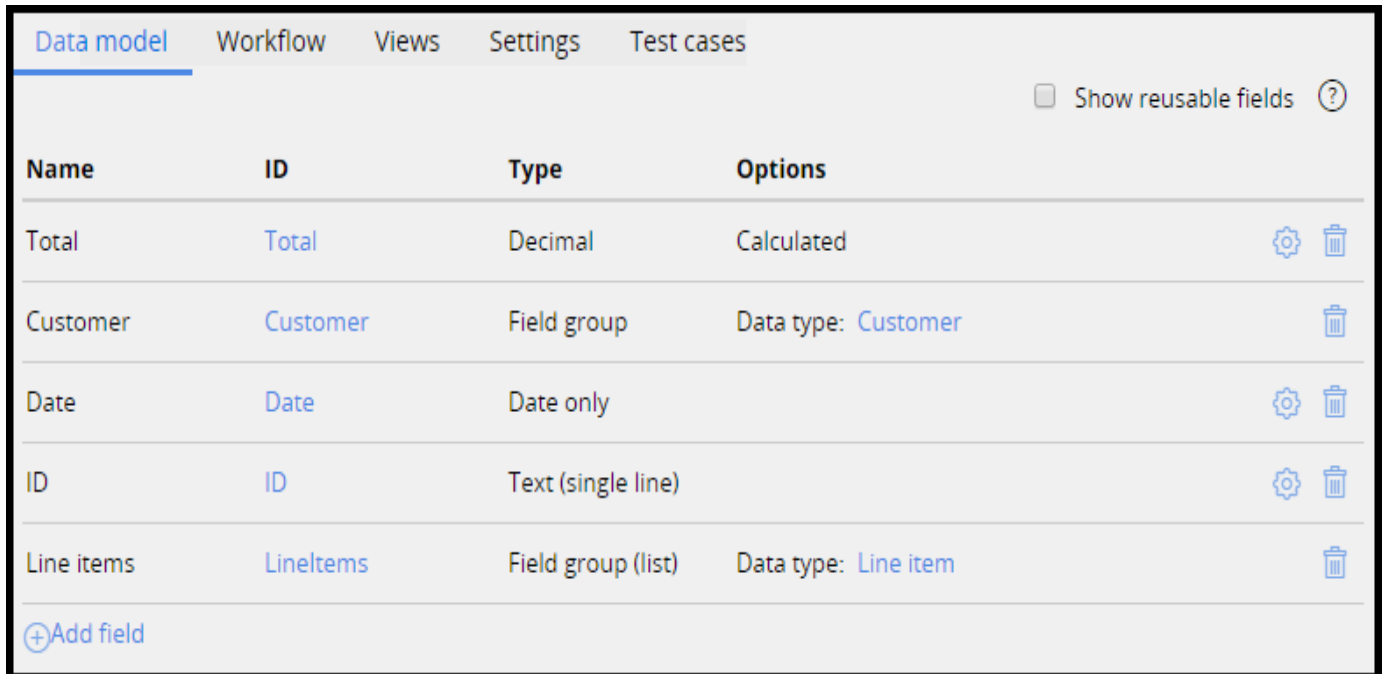


What data modeling capability is available in Designer Studio but not Pega Express?









The capability to reuse an existing data type.

The Data model tab

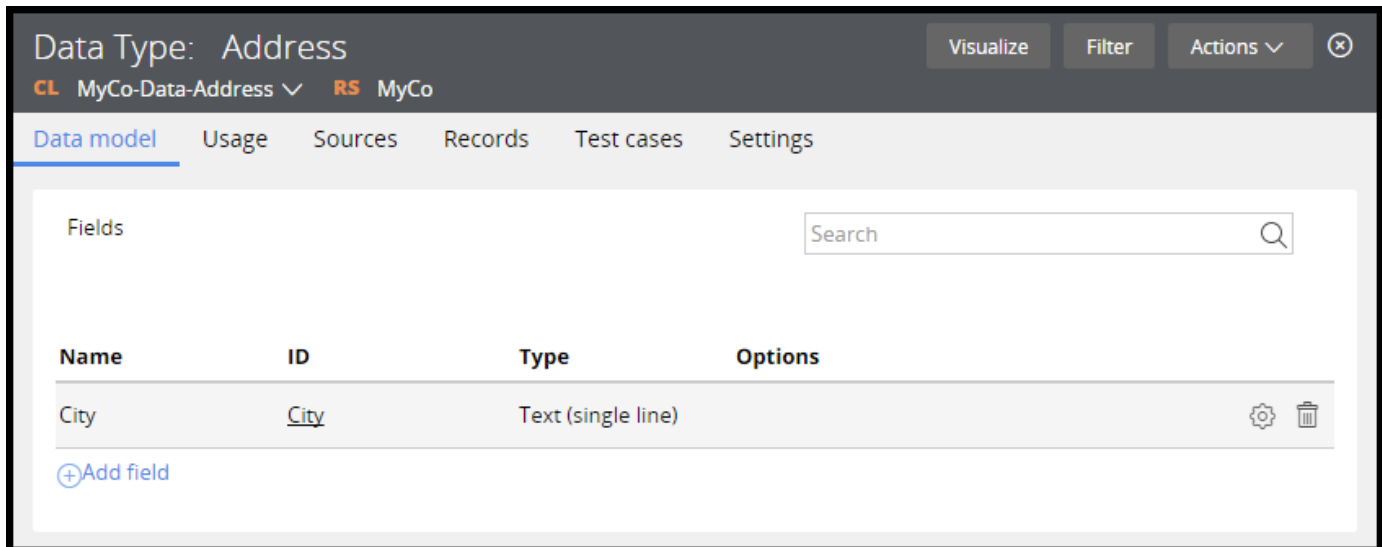
You can use the Data model tab in the Case Designer to add or remove properties from your case type. Properties are called fields in the Data model tab. In Designer Studio, you can enable **Show reusable fields** to display all fields inherited and available in the case type.





The screenshot shows the 'Data model' tab in the Case Designer. At the top, there are navigation tabs: 'Data model' (selected), 'Workflow', 'Views', 'Settings', and 'Test cases'. In the top right corner, there is a checkbox labeled 'Show reusable fields' with a help icon. Below this is a table with the following columns: 'Name', 'ID', 'Type', and 'Options'. The table contains five rows of field definitions. At the bottom left, there is a '+Add field' button.

Name	ID	Type	Options
Total	Total	Decimal	Calculated  
Customer	Customer	Field group	Data type: Customer 
Date	Date	Date only	 
ID	ID	Text (single line)	 
Line items	Lineltems	Field group (list)	Data type: Line item 

The Data model tab for a data type looks very similar.

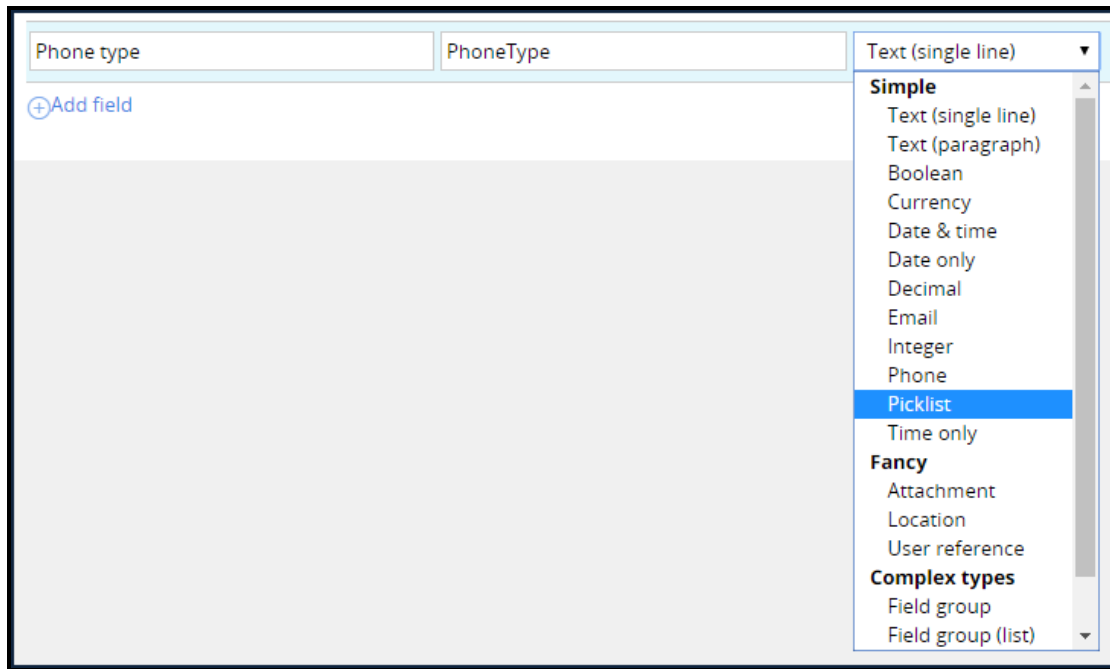


The screenshot shows the 'Data model' tab for a data type named 'Address'. The title bar reads 'Data Type: Address' and includes buttons for 'Visualize', 'Filter', 'Actions', and a close icon. Below the title bar, there are navigation tabs: 'Data model' (selected), 'Usage', 'Sources', 'Records', 'Test cases', and 'Settings'. The main area is titled 'Fields' and contains a search bar. Below the search bar is a table with columns: 'Name', 'ID', 'Type', and 'Options'. The table contains one row for the 'City' field. At the bottom left, there is a '+Add field' button.

Name	ID	Type	Options
City	City	Text (single line)	 

Selecting the field type

When creating a new field, you need to specify a type. The options in the list pair the field with a control in the user interface. The type options are divided into three categories: simple, fancy, and complex.



The simple types are similar to the property types defined on the property itself. Use a picklist if you need to display a static list of options to the user. For example, if you want to capture a phone number, you might want to specify a list of types, such as home, work, and mobile.

The fancy types provide the capability to upload an attachment, show a location on a map, or reference a user on the system.

Use the complex types to define page and page list properties. A field group is a page and a field group (list) is a page list.

The Property rule form

The property rule form contains the property definition. Because a property definition is a rule, it shares the benefits of versioning, inheritance, and access control that the Pega Platform provides to all rules.

The screenshot shows the 'Property rule form' with the following details:

- Property type:** Text (change)
- Data access:** Manual (selected). Note: At run time, the user adds data to this property through the UI. Data transforms and other rules may be required to support this workflow.
- Display and validation:**
 - UI Control: pxTextInput
 - Table type: None

The property has one of 11 types.

Note: The property type cannot be changed after the property has been saved.

Use the Data Access section to configure automatic data access and persistence settings. Use Manual if you are explicitly setting the value (for example, in a user interface). Other options depend on the property type selected and are not covered in this lesson.

The Display and validation section allows you to define how the property should appear on the screen by specifying a UI control. You also have the option to specify a table with valid values for the property.

Pega comes with a set of standard property rules. The standard properties have names that start with px, py, or pz. You cannot create new properties starting with px, py, or pz.

The table below provides a list of the prefixes for standard rules.

Prefix	Meaning
px	Identifies special properties — your application can read but not write to these properties.
py	You can use these properties in your application.
pz	Supports internal system processing — the meaning of values may change with new product releases. Your application can read but not write to these properties.

How to reference a property

You have learned about two property modes: value and page. Value mode properties store single strings of data such as text, numbers, or dates. Page mode properties act as a container for value mode properties. You refer to a property in Pega 7 by prefixing the property name with a period (or dot, ".").

- To refer to a single value property named `OrderDate`, type `.OrderDate`.
- To refer to an entry in a value group property, such as the mobile phone number, type `.Phone(Mobile)`, where `Mobile` is the group subscript.
- To refer to the first entry in a value list property, such as one of the discount codes, type `.DiscountCode(1)`, where `1` is the list index.

Page mode properties are similar.

- To refer to a page that contains customer information, type `.Customer`.
- To refer to an entry in a page group property, such as the work address, type `.Address(Work)`.
- To refer to the third page of a page list that contains purchase request line items, type `.LineItems(3)`.

To refer to a specific property on the page, use the name of the page as a prefix for the property name. By doing this, you establish an important piece of information about that property — its context. The context of a page — by itself or as part of a page list or page group — acts as a container for the properties it contains. If you want the city in the work address, specify `.Address(Work).City`.

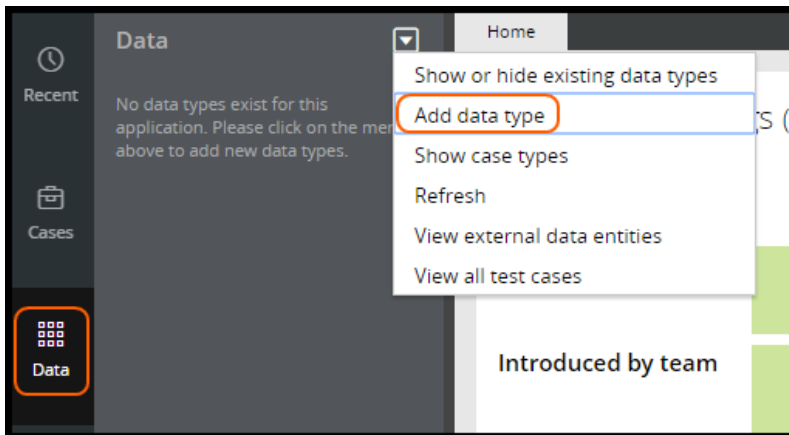
Defining properties

In Designer Studio, you use the Data Explorer to create or reuse a data type and use the Data Model tab to manage properties.

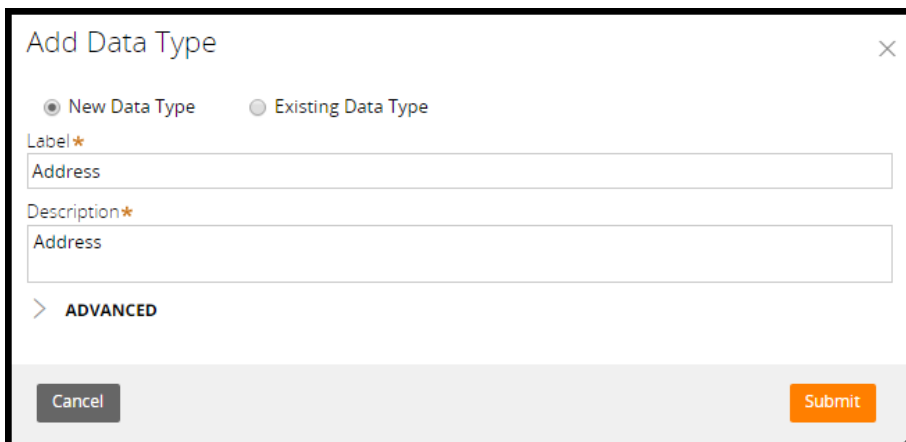
Creating a data type in Designer Studio

Follow these steps to create data types from the Data Explorer:

1. Select **Data** in the left pane to open the Data Explorer.
2. Select **Add data type**.











3. Select **New Data Type** if you want to create a new data type, or select **Existing Data Type** to include an existing one.
4. Provide a label and description.



5. Click **Submit** to create the data type.



Managing properties in a case or data types

Use the Data Model tab in the Case Designer to add or remove properties from your case type. Properties are called fields in the Data model tab.

Name	ID	Type	Options
Total	Total	Decimal	Calculated  
Customer	Customer	Field group	Data type: Customer 
Date	Date	Date only	 
ID	ID	Text (single line)	 
Line items	Lineitems	Field group (list)	Data type: Line item 

[+Add field](#)

The Data Model tab for a data type looks very similar.

Name	ID	Type	Options
City	City	Text (single line)	 


[+Add field](#)

Adding a field to a case or data type

Follow these steps to add a field:


1. Open the **Data model** tab in the Cases Explorer.
2. Click the **Add field** link.
3. Specify a name for the field (property).

4. The ID field is automatically populated by the system. You can choose to edit the ID field.
5. Select the type.
6. If you select a field group or field group list, you need to provide a data type in the options field.

Name	ID	Type	Options
Customer	Customer	Field group	Customer New 


Updating a field in the case or data type

Click the row to update the name of a field.

Name	ID	Type	Options
<input type="text" value="Customer"/>	<u>Customer</u>	Field group	Data type: <u>Customer</u> 

Remove a field from the case or data type

Click the **trash can** icon to remove a field.

Name	ID	Type	Options
Customer	<u>Customer</u>	Field group	Data type: <u>Customer</u> 

Property rules are automatically added, removed, or updated as you use the Data Model tab.

Reviewing application data

Introduction to reviewing application data

Applications generate large amounts of data. Sources of data include case data, user data, and data from outside sources. While developing applications, you may need to review the application data for debugging. Incorrect information can cause errors that lead to undesired results for cases.

In this lesson, students learn how to review application data.

After this lesson, you should be able to:

- Explain how data is stored in memory for use in Pega applications
- Describe the relationship between pyWorkPage and case data
- Explain how the Clipboard tool organizes data in memory
- Use the Clipboard tool to review case data in memory
- Use the Clipboard tool to set values for case data

Data storage in memory

Cases are collections of data. To process and resolve a case, Pega applications capture, manipulate and present data throughout a business process. While processing a case, this data remains in memory for use by one or more users.

Each **data element** in a Pega application is a pairing of two pieces of information: the name of the data element, and the value assigned to the data element. For example, when you use a data element to capture the date of birth of an person, the data element name is date of birth, and the corresponding value is a date such as July 20, 1969.

Name of the data element	Value assigned to the data element
First Name:	Neil
Date of Birth:	July 20, 1969
Occupation:	Astronaut

Each data element is stored in memory on a page. A **page** is a structure for organizing data elements in an application. Some pages are created by the system to track user or session data. Other pages are created by system architects to describe a data object, such as a hotel reservation or a customer.



During case processing, each page remains in memory in a structure known as the **clipboard**. The clipboard is the portion of memory on the server reserved by Pega for the data generated by applications. The clipboard consists of all of the pages used to track the name-value pairs that represent case and session data. The clipboard receives its name because pages can be added to or removed from memory as needed to track case or session data. So, when a value is assigned to a data element, the data element and its value are said to be on the clipboard.

As you run a process, Pega sends information to the clipboard, adding or removing pages and properties from memory. Your application uses this information to populate fields on UI forms, perform calculations, and evaluate decisions.

KNOWLEDGE CHECK



How is information, such as a customer's date of birth, stored in memory for use in a Pega application?


Information such as the customer's date of birth is associated with a data element. The data element (property and value) is stored on the clipboard in a structure called a page.

pyWorkPage

When you debug case behavior, you often need to view the case data that is in memory on the clipboard. Viewing the data on the clipboard can help you identify the cause of the issue. For example, if a data transform returns an unexpected result, you can review the contents of the clipboard to determine if one of the input properties has been set with an unexpected value.

The following image shows a list of properties and their values. A data transform that populates the *EmployeeFullName* field with the values in the *EmployeeFirstName* and *EmployeeLastName* properties

contains an error. The *EmployeeFullName* field seems to be pulling the first two letters of the *EmployeeEmail* field. Your next debugging step is to examine the data transform.



EmployeeFirstName	John
EmployeeLastName	Jones
EmployeeFullName	JJ
EmployeeEmail	jjones@tns.com

pyWorkPage stores all the data generated while creating and processing a case. `pyWorkPage` is a specific page on the clipboard. For example, `pyWorkPage` stores data such as the case creation date or the case ID. An embedded page with `pyWorkPage` stores data that describes a data type. For example, if a case uses a data type named *Customer*, then *Customer* is an embedded page within `pyWorkPage`. Pega Platform writes all the properties that describe the *Customer* data type — such as first name — to the embedded page.

When you open a child case, the clipboard also contains the page **pyWorkCover**. `pyWorkCover` contains the case data for the parent case. This enables you to copy data between the parent case and the child case.

KNOWLEDGE CHECK



What is the purpose of an embedded page within `pyWorkPage`?

An embedded page within `pyWorkPage` stores data that describes a data type.

pyWorkPage in reports

Each page on the clipboard is an instance of a specific class, including `pyWorkPage`. When you configure a report to refer to data on `pyWorkPage`, you may need to specify the class of the page. If you omit the class information, Pega cannot obtain property values from the correct page. Pega does not know if the properties are valid, and the rule that references the properties does not function correctly. To ensure that the report obtains the correct information whenever you reference `pyWorkPage`, you need to specify the class of `pyWorkPage`.

For example, consider an application to process automobile insurance quotes. To price the quote, you need to know the accident history of the driver. Each accident record is an instance of a specific data type. You create a report to return the accident history for a driver and use a filter to return only accidents for the driver requesting the quote. If the report filter uses the *UserName* property from

pyWorkPage, you must tell the report the class for pyWorkPage (for example *INSCO-Quotes-Work*). This allows Pega to reference the *UserName* property and the report filter functions as intended. Otherwise, Pega assumes that *UserName* is part of the data type, rather than the case, and the filter does not work correctly.

KNOWLEDGE CHECK

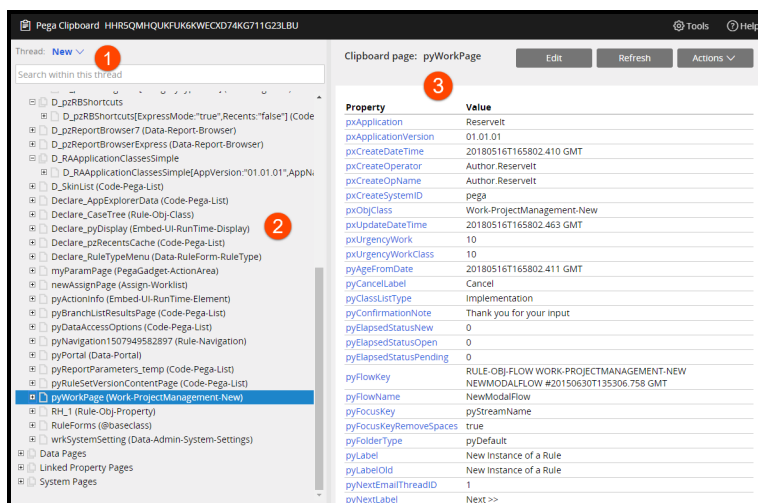


Why is it important to include class information when you reference data on pyWorkPage?

If you omit the class information, Pega cannot obtain property values from the correct page. Pega does not know if the properties are valid, and the rule that references the properties does not function correctly.

How to view clipboard data

To view data that is in memory, you use the **Clipboard tool**. The Clipboard tool organizes and presents all the pages on the clipboard. When you select a page, the Clipboard tool lists all the properties on each page and the value of those properties. To open the Clipboard tool, click the Clipboard icon on the Developer toolbar in Designer Studio.



You can use the header to select the **thread** to view. Each thread corresponds to a unique action currently managed by Pega. The clipboard contains one thread dedicated to the Designer Studio environment. Other threads are dedicated to open rule forms. Pega assigns each open case a unique thread. By assigning each case or action its own thread, Pega ensures that the data for one case or action does not affect data for another case or action.

The left pane lists each page defined on the clipboard for the selected thread. For each page, the Clipboard tool identifies the name and class of the page. If a page contains embedded pages, an expand arrow is displayed to the left of the page name. To view the embedded pages, click the expand arrow.

Pages on the clipboard are organized into four categories:

- The **User Pages** category contains pages created due to user action, either directly or indirectly. User pages contain data related to work being performed in the selected thread. While a user

processes a case, all the pages used to store data about the case are listed in the User Pages category. Likewise, when a system architect configures or tests a rule, all the pages that store data used by the rule are listed in this category. For example, the data you enter onto a form is stored on the user page pyWorkPage.

- The **Data Pages** category contains read-only data pages defined by data page rules. Data pages are persistent pages in memory, used to cache data. This data is often sourced from outside the application, such as from a third-party or a system of record. For example, your application converts currency from one type to another, such as converting US dollars to Euros. The conversion rates, which are determined by the currency markets, are cached to a data page for use by one or more users of the application.
- The **Linked Property Pages** contains read-only pages created by linked properties, which contain information from data objects referenced by a linked property. Linked properties are advanced data constructs, typically created and configured by Senior System Architects (SSAs) or Lead System Architects (LSAs).
- The **System Pages** category contains pages that describe the current user session, such as the active user and the active application. For example, while a user is logged in to Pega, Pega maintains a clipboard page containing information about the user, such as their current time zone.

The right pane lists all of the properties defined on the selected page, and their values. In the right pane, you view data in memory. You can also update property values and even add new properties to the page to represent data not captured in your application. This allows you to test application features that rely on data that has not been added to the case type, such as decisions and UI forms. For example, in an expense report case you want to branch a flow based on the project type. The application currently lacks a field in the UI to allow the user to select the project type. In this situation, you can use the clipboard to set a value for the property and verify that the flow branches properly.

When you view data with the Clipboard tool, you see a snapshot of the contents in memory. As you navigate your process, refresh pages in the Clipboard tool to ensure that the Clipboard tool always displays current property values and page contents.

KNOWLEDGE CHECK



While testing case behavior for an online shopping application, you want to confirm that the application properly generates a list of the customer's previous orders when querying the company's order management system. In which category of clipboard pages would you expect to find the page that contains this list?

This page should be located in the Data Pages category.

Setting property values automatically

Introduction to Setting Property Values Automatically

As you design and implement a case type, you may need to copy or manipulate data. For example, you may have a requirement to collect an individual's first and last name, and then combine them into a full name. In other situations, you may want to set default values for properties.

After this lesson, you should be able to:

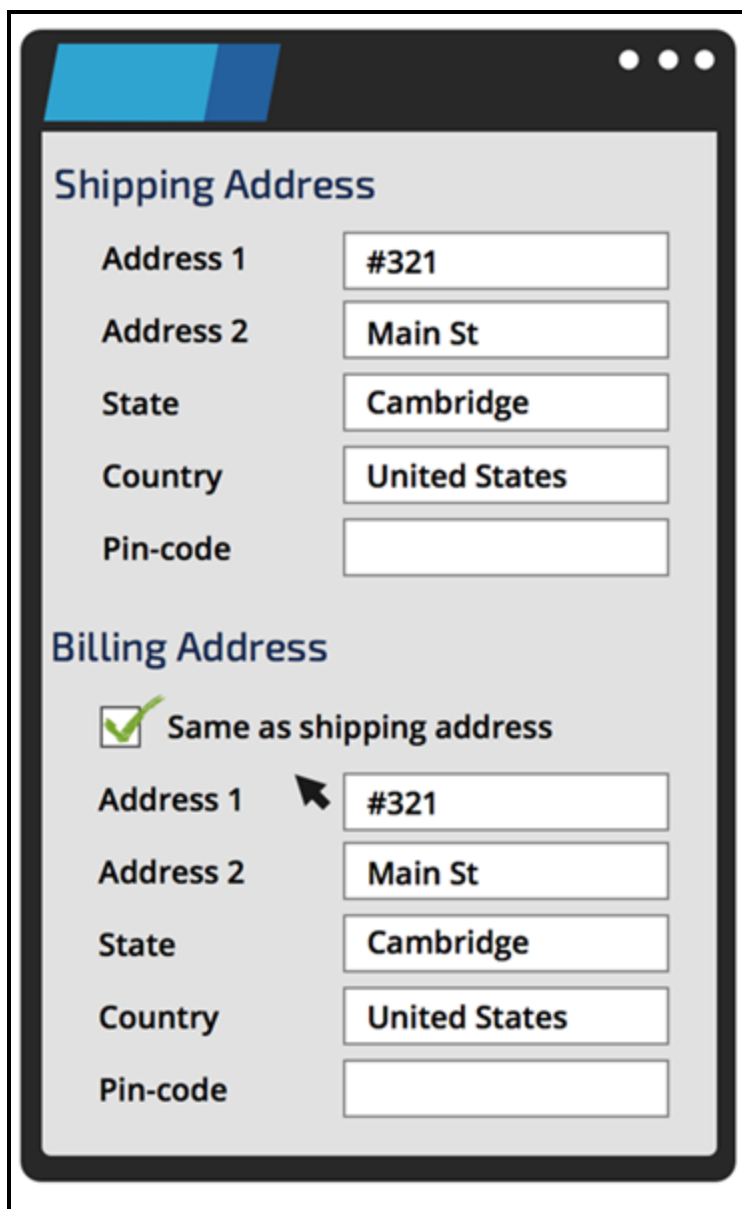
- Explain the use of data transforms in an application
- Identify situations in which to set property values automatically
- Explain how data transform superclassing works
- Set initial property values using the pyDefault and pySetFieldDefaults data transforms

Data transforms

When you create and process a case, you need data. You collect, process, act upon, and present that data back to the user. Sometimes, you need to copy data from one place to another. Other times, your data is not in the form you require, so you need to find a way to manipulate that data into an acceptable form.

In a purchasing application, for example, items are added to a cart and the checkout process begins. The customer provides a shipping address and credit card information, and is prompted to provide a billing address.

The shipping address might be the customer's home address — the billing address and shipping address are likely to be the same. Reusing rather than having to reenter the shipping address is helpful and more efficient. Similarly, you might collect an individual's first name and last name, but need to combine the two into a full name for credit card processing.



The image shows a screenshot of a web form titled "Shipping Address" and "Billing Address". The "Shipping Address" section has five input fields: "Address 1" with "#321", "Address 2" with "Main St", "State" with "Cambridge", "Country" with "United States", and "Pin-code" which is empty. The "Billing Address" section has a checked checkbox labeled "Same as shipping address" and five input fields: "Address 1" with "#321", "Address 2" with "Main St", "State" with "Cambridge", "Country" with "United States", and "Pin-code" which is empty. A mouse cursor is pointing at the "Address 1" field in the Billing Address section.

One option for copying and manipulating data is the **data transform**. The purpose of a data transform is self-explanatory: it transforms data in the application. This example uses a data transform to copy the shipping address to another page — in this case, the billing address — and to copy the first and last name properties into a single property full name.

You can use data transforms in several ways. For example, you can call a data transform from a flow action rule or from a connector. Also, you can use a special data transform rules — `pyDefault` or `pySetFieldDefaults` — to initialize property values when creating a case. Data transforms can be used to iterate over page lists or page groups, and copy entire pages at a time.

How to set values with data transforms

Use a data transform to define how to take source data values — data that is in one format and class — and turn those values into data of another target format and class. In general, data transformation involves mapping data from a source to a target as well as performing transformations on that data required to achieve the intended mapped results.

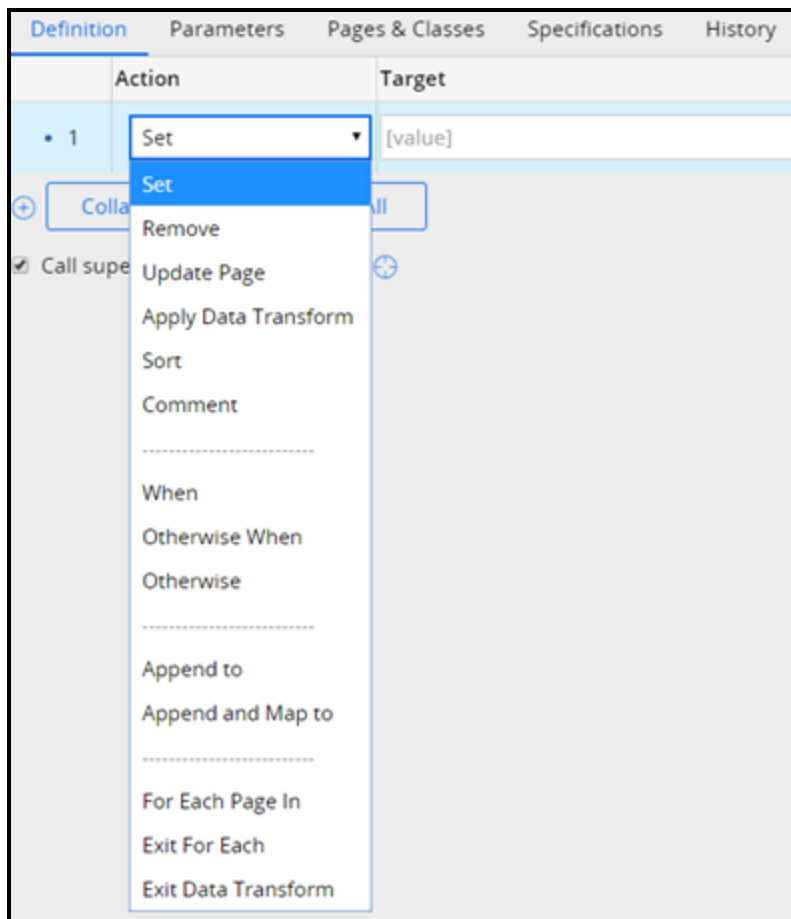
To use a data transform, you first configure it and then indicate where in the process flow the data transfer occurs using the Process Modeler.

Configuring a data transform

Configuring a data transform requires specifying an action and then entering the values for the transform.

Actions are the individual operations specified in each row on the **Definition** tab of a data transform. The system invokes the actions at run time. Most actions do some kind of data manipulation. Other actions perform conditional processing and iterate through page lists and page groups. For more information on actions, see the help topic [Data Transform form - Completing the Definition tab](#).

In the Actions column, select the appropriate action for what you want to do.



Next, enter the **Target**, **Relation**, and **Source**. Depending on the selected action, the **Target** field has a different meaning. For the **Set** and **Update Page** actions, the **Target** field identifies a property or page

reference, and the **Source** column provides an expression that results in a value or values. For the when action, a when condition needs to be specified.

KNOWLEDGE CHECK



What is an action in a data transform?

An action is the individual operation specified in each row on the **Definition** tab of a data transform.

How to reference a property

In Pega Platform, there are two property modes: value and page. Value mode properties store single strings of data such as text, numbers, or dates. Page mode properties act as a container for value mode properties. You refer to a property in Pega Platform by prefixing the property name with a period (or dot, ".").

- To refer to a single value property named OrderDate, type `.OrderDate`.
- To refer to an entry in a value group property, such as a mobile phone number, type `.Phone(Mobile)`, where Mobile is the group subscript.
- To refer to the first entry in a value list property, such as a discount code, type `.DiscountCode(1)`, where 1 is the list index.

Page mode properties are similar.

- To refer to a page that contains customer information, type `.Customer`.
- To refer to an entry in a page group property, such as work address, type `.Address(Work)`.
- To refer to the third page of a page list that contains purchase request line items, type `.LineItems(3)`.

To refer to a specific property on the page, use the name of the page as a prefix for the property name. By doing this, you establish an important piece of information about that property — its context. The context of a page, by itself or as part of a page list or page group, acts as a container for the properties it contains. For example, the shipping address on the customer page becomes `.Customer.Address(Shipping)`. Also, the type of the third asset on an asset list becomes `.Asset(3).Type`.

In the following data transform, the first step checks if the billing address is the same as the shipping address. If the two addresses are the same, the shipping address is copied to the billing address. Otherwise, the billing address is set to empty values.

Definition					Parameters	Pages & Classes	Specifications	History
	Action	Target	Relation	Source				
1	When	BillingSameAsShipping						
1.1	Set	.Customer.Address(Billing)	equal to	.Customer.Address(Shipping)				
2	Otherwise							
2.1	Update Page	.Customer.Address(Billing)						
2.1.1	Set	.AddressLine1	equal to	""				
2.1.2	Set	.AddressLine2	equal to	""				
2.1.3	Set	.City	equal to	""				
2.1.4	Set	.Country	equal to	""				
2.1.5	Set	.PostalCode	equal to	""				
2.1.6	Set	.State	equal to	""				

Call superclass data transform

KNOWLEDGE CHECK



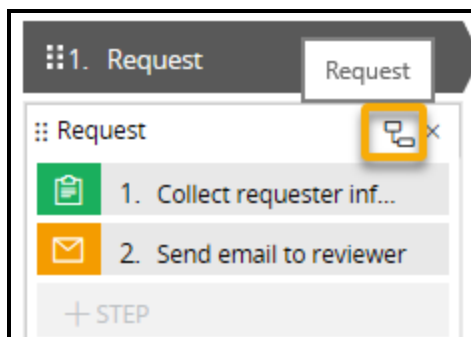
ANSWER

Briefly describe the two types of property modes.

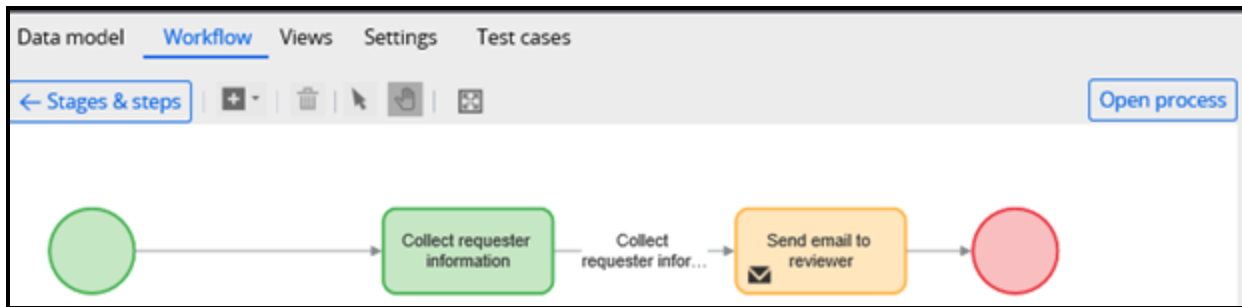
Value mode properties store single strings of data such as text, numbers, or dates. Page mode properties act as a container for value mode properties.

Adding a data transform to a process

Identify the process in which you want the data transform to occur. In Designer Studio, on the **Workflow** tab of the case type, click the appropriate process icon to view the flow rule.



To edit the flow rule in the Process Modeler, click **Open process**.



Most data transforms occur between assignments. Double-click the connector between the assignments to open the connector properties panel. In the Connector properties panel, under Set properties, select **Apply Data Transform**. The **Data transform** field is displayed, listing the available data transforms. Select the appropriate data transfer and click **Submit**, then save the process.

KNOWLEDGE CHECK



Where do most data transforms occur in a process flow?

Most data transforms occur between assignments.

How to set default property values

When a user creates a case, you may want to set default values for some properties. In an insurance claim case, you could set the date of loss to today's date. In other situations, you might want to use data from the operator record — such as the user's organization unit — to initiate property values.

In a development environment, setting default values can also be useful. When you iteratively run a process to test your changes, using a data transform to enter default values in required fields saves time.

Pega Platform provides a pair of data transforms (**pyDefault** and **pySetFieldDefaults**) that are used to automatically set default property values and [page](#) properties when you create a case.

The first time you create a view for your case type in the Case Designer, Pega Platform creates these data transforms. You can also create these data transforms manually. When a new case is created, the *pyDefault* data transform is invoked by the process. *pyDefault* then invokes the *pySetFieldDefaults* data transform.

Using pyDefault

Configure *pyDefault* to set default values when a user creates a case. For example, when a user opens a loan application case, you may want to initialize the application date to the current date and the terms to 30 years.

To set a default value, configure a row in *pyDefault* as shown here. You can enter the property in the **Target** column and the default value in the **Source** column.

Action	Target	Relation	Source
Set	<property>	equal to	< default value >

The following table shows target properties that use a number, a literal constant, and an expression source value.

Action	Target	Relation	Source
Set	.OfferedSalary	equal to	100000
Set	.EnrolledStudent	equal to	"True"
Set	.DateOfEnrollment	equal to	@DateTime CurrentDateTime

KNOWLEDGE CHECK



You are using *pyDefault* to set the manager property to Sharon Smith. What do you use as your Target and Source values?

You use the manager property as the Target and "Sharon Smith" as the Source.

Using pySetFieldDefaults

Similar to *pyDefault*, you can use the *pySetFieldDefaults* data transform to set default values for properties. In addition, you use *pySetFieldDefaults* to automatically initialize page (field group) and page list (field group list) properties. When *pyDefault* invokes *pySetFieldDefaults*, properties are immediately available to the case when it is created.

For example, your view may contain fields from a Customer page group that includes first name, last name, phone number, and email address properties. The *pySetFieldDefaults* data transform can create a blank Customer page in the [clipboard](#), which makes the page available to the view. When the case is created, *pySetFieldDefaults* sets the *.pyLabel* property to an empty string which causes the page to be created on the clipboard. If there were no page, users would not see the view.

Note: Even if the fields are not visible in the UI, the system may need access to the page for processing the case.

You do not need to update *pySetFieldDefaults* when you add a page or page list to a view in the Case Designer. Pega Platform adds these properties for you automatically. You can also use *pySetFieldDefaults* to initialize any page properties you create manually outside of the Case Designer.

Example configuration

In the following example, you have created your first view in your case type. The view contains the Employee page group. The Case Designer creates the *pySetFieldDefaults* and *pyDefault* data transforms.

pySetFieldDefaults

To initialize pages used by the view, *pySetFieldDefault* sets the *.pyLabel* fields. In this example, the **Target** value is *.Employee.pyLabel*. The default **Source** value remains empty so that the user sees a blank form that contains all the fields (such as start date, manager, and department).

Action	Target	Relation	Source
Set	.Employee.pyLabel	equal to	" "

You can update *pySetFieldDefaults* to define default property values. For example, you can set the default department page property so that Sales is displayed in the Department field when the user opens the view.

Action	Target	Relation	Source
Set	.Employee.pyLabel	equal to	" "
Set	.Employee.Department	equal to	"Sales"

pyDefault

Because the *pyDefault* data transform is always invoked when a case is created, *pyDefault* includes the **Apply Data Transform** action. This ensures that *pySetFieldDefaults* is also invoked when a case is created. This action does not use **Relation** or **Source** settings.

Action	Target	Relation	Source
Apply Data Transform	pySetFieldDefaults	<NA>	<NA>

KNOWLEDGE CHECK



ANSWER What action is defined in pyDefault to ensure that pySetFieldDefaults is invoked when a case is created?

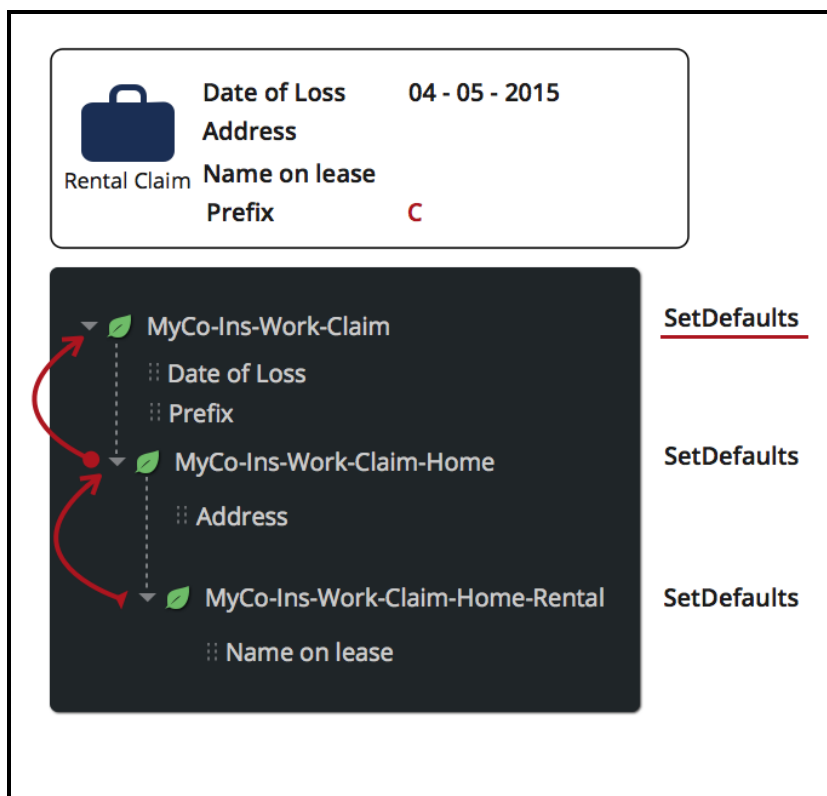
The Apply Data Transform action.

Data transforms and superclassing

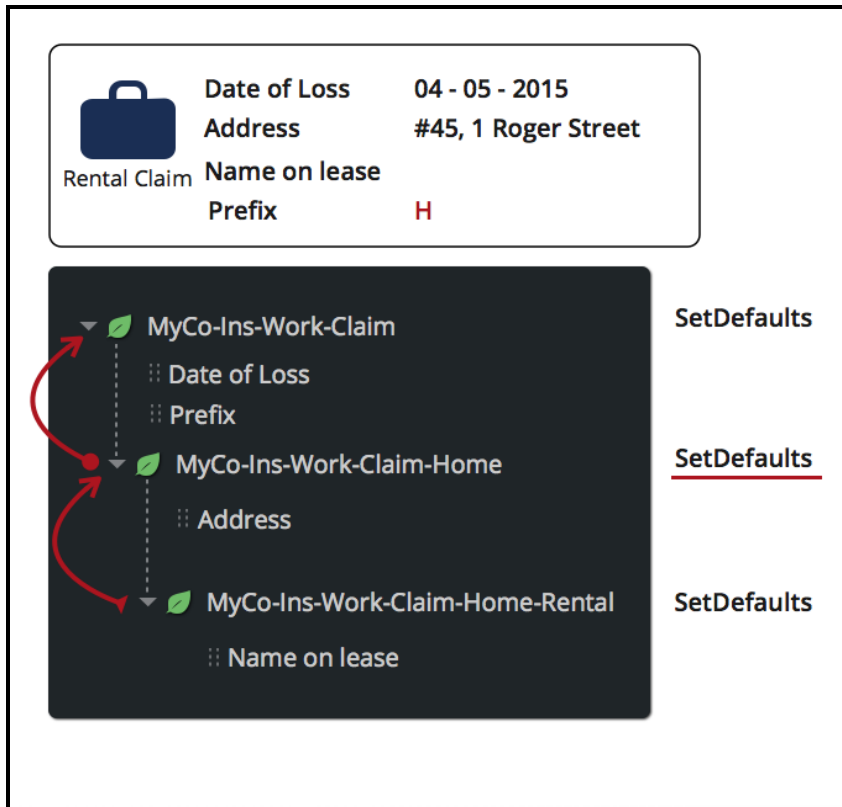
You can combine several data transforms using the superclass feature to set values at multiple levels of the class hierarchy. For example, you can have a class Claim with a subclass Home. The subclass Home in turn has a subclass Rental with data transforms at each level that sets default values.

You can set up your data transforms so that common default values are set in the claim class and specific values are set in the subclasses. Taking advantage of this feature improves the maintainability of data transforms.

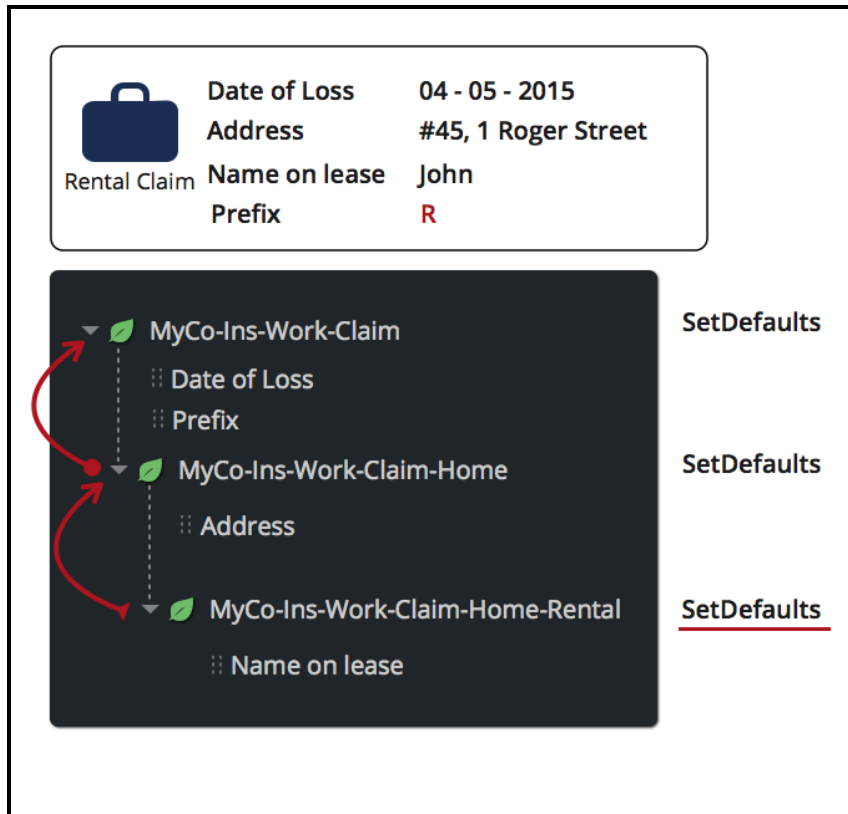
Here, the date of loss and prefix are in the Claim class, address is set in the Home class, and name on the lease in the Rental class. The system first identifies parents until it reaches the highest parent. In this case, the highest parent is the Claim class. The system locates and then invokes the data transforms with the same name in the parent class.



The system then goes down to the second highest parent and locates the data transforms with the same name to invoke. Note that the prefix is overwritten.



Finally, the data transform in the Rental class is invoked. Note that the prefix is overwritten again.



KNOWLEDGE CHECK



What data transform feature do you use when you want to set a different default value for the same property within a hierarchy of subclasses?

The superclass feature

How to configure superclassing for data transforms

Use the superclass feature by creating a data transform with the same name at each level and selecting the **Call superclass data transform** option. If properties are specified in both the super and child classes, the data transform in the subclass overwrites anything already set by the data transform in the superclass.

You must select the **Call superclass data transform** option to cause the system to invoke data transforms of the same name in parent and child classes before the data transform gets invoked itself.

	Action	Target	Relation	Source
• 1	Set	.Driver	equal to	.Policyolder

Call superclass data transform

KNOWLEDGE CHECK



When you create data transforms in a class hierarchy and want to use the superclass feature, how do you name each data transform?

You give each data transform the same name.

Overwriting standard pyDefault data transform property values

Pega Platform comes with standard `pyDefault` data transforms in the work classes from which case types inherit. The **Call superclass data transform option** is selected by default. The standard `Work-pyDefault` data transform sets property values that are used in all case types. The properties include work status, work urgency, and operator organization information. The `@baseclass pyDefault` data transform sets default values for properties used by all classes.

You can override these default settings in your case type `pyDefault` data transform. For example, the `Work-pyDefault` work urgency value is set to 10. If you want to prioritize cases in your case type, you can change the default urgency value for all new cases to 40.

Setting property values declaratively

Introduction to setting property values declaratively

When a user enters a value in a form, related values can also change. When you buy an item like a laptop online, you enter the quantity you want to purchase. The system displays the amount of your order automatically. Declarative processing allows you to easily configure your application so that the system can automatically update property values such as an order amount.

After this lesson, you should be able to:

- Describe the declarative processing model
- Describe the procedural processing model
- Calculate a property value with a declare expression
- Explain how forward chaining works to update property values
- Explain how backward chaining works to update property values

Declarative and procedural processing

When a user enters a property value in a form, related values on the same form or on other forms can change as a result. The application must make the changes automatically so that users see the most current information. For example, assume you are purchasing a laptop online. On the form, you enter 1 as the quantity. A total order amount field displays the price of one laptop. However, when you change the quantity to 2, the total order amount automatically doubles. Alternatively, you may want to see changes to a form only after the user submits the form. Declarative and procedural processing techniques help you to automate updates to your application.

Declarative processing

Declarative processing allows you to configure your application so that the system automatically updates property values such as a total order amount. Declarative processing identifies and maintains computational relationships among properties. When input values change, the declarative process automatically updates related property values. In the previous example, a declarative process maintains a relationship between the total order amount property value with the quantity and price property values. When a user orders laptops, the system multiplies the price of one laptop times the quantity of laptops to calculate the total order amount.

The primary benefit of declarative processing is that updates occur only when triggered in the application. You use declarative rules to define the **trigger event**. The system monitors the application to determine when a trigger event occurs. Using the previous example, the system is always monitoring changes to the item quantity property. When the value changes, the system triggers a computation to update the order total.

The following video describes how declarative processing works and provides an example.

KNOWLEDGE CHECK



What is the primary benefit of using declarative processing?

The primary benefit of declarative processing is that the system processes the relevant actions and sets of properties automatically. They run automatically when a value is set or changes.

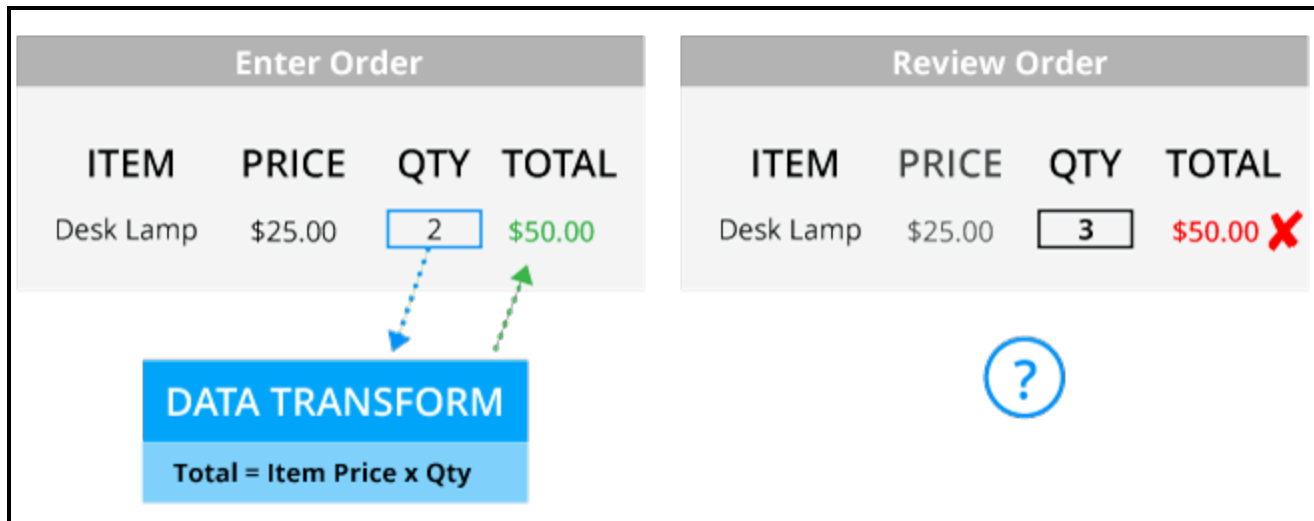
A single declarative expression can monitor trigger events no matter where that expression is used in the application. Declarative processing rules do not depend upon other rules, such as data transforms, activities, or user interface (UI) rules, to perform updates.

Procedural processing

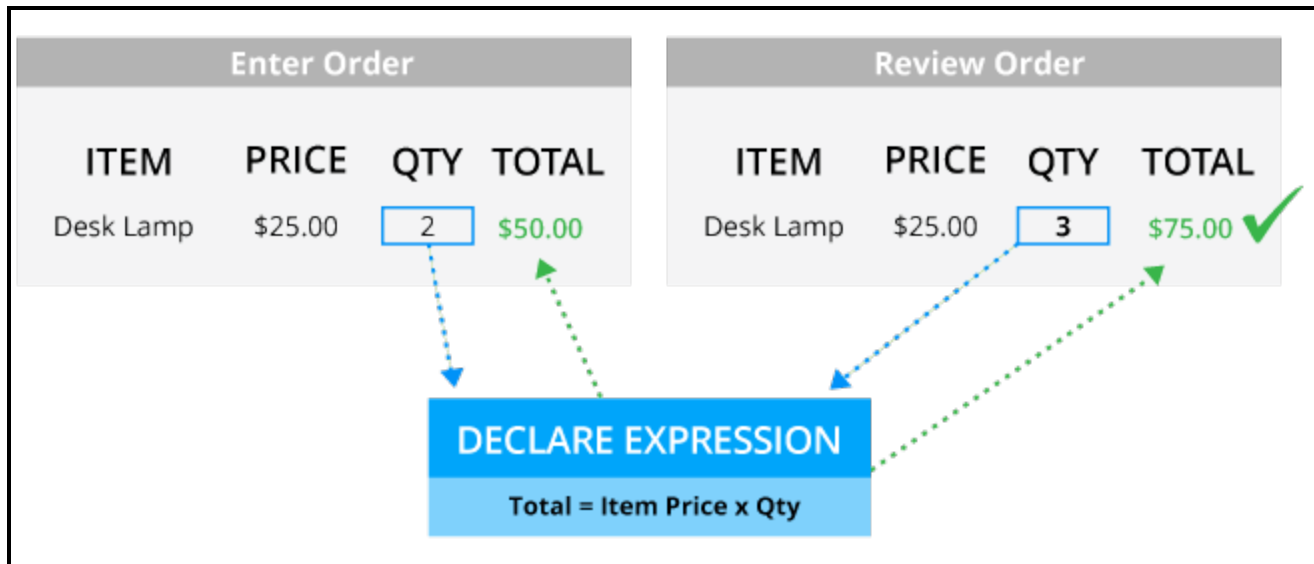
Procedural processing depends upon rules, such as data transforms, activities, or user interface (UI) rules, to instruct the application when to look for a trigger event. For instance, to trigger updates to the order total, you add a data transform to a flow action. When a user enters values, nothing changes until the user submits the form. The updates are not automatic. The submit process triggers the data transform to perform the update. In order to make the changes visible to users as they enter values, you must configure sections to use the data transform to refresh the fields.

Procedural processing maintenance

Procedural processing is more difficult to configure and maintain than declarative processing. For example, assume you have designed an Enter Order form that uses a data transform to calculate the total order amount based on the item price and order quantity. Then, you add a Review Order form to your application. This form reuses the fields for calculating the total order amount. If you do not add the same data transform to the Review Order form, the Total order amount is not updated when the user changes the order quantity from 2 to 3.



In contrast, when you use a declare expression the system monitors changes to the source property values. In the following example, when a user updates the quantity from 2 to 3 in the Review Order form, the declare expression recalculates the total.



Pega provides many types of standard declarative rules that support declarative processing. For more information, see the PDN article [Declaratives, Decisions, and Validation](#).

KNOWLEDGE CHECK



ANSWER

Which technique should you use if you want to update values when the user submits a form?

Use a procedural processing by putting a data transform on a flow action.

KNOWLEDGE CHECK



ANSWER

Which technique should you use if you want the total price to update immediately when a quantity is changed?

Use a declare expression.

Declare expressions

You most often use declare expressions to calculate and make immediate updates to property values on user forms. Declare expressions contain an **expression** and a **target property**. The expression calculates and updates the target property value. The expression uses **source property** values in its calculation. Referencing a source property used in the expression initiates the calculation. The calculation then updates the target property value.

For example, assume an office furniture purchase order form includes fields for three target properties items: chairs, desks, and lamps. Each item has fields in which users select an item and enter the quantity. A declare expression uses the two source properties, item cost, and quantity to calculate the target property — item total. The expression multiplies the item cost times the number of quantity to calculate the item total. When the user changes the quantity, the expression recalculates the item total.



KNOWLEDGE CHECK



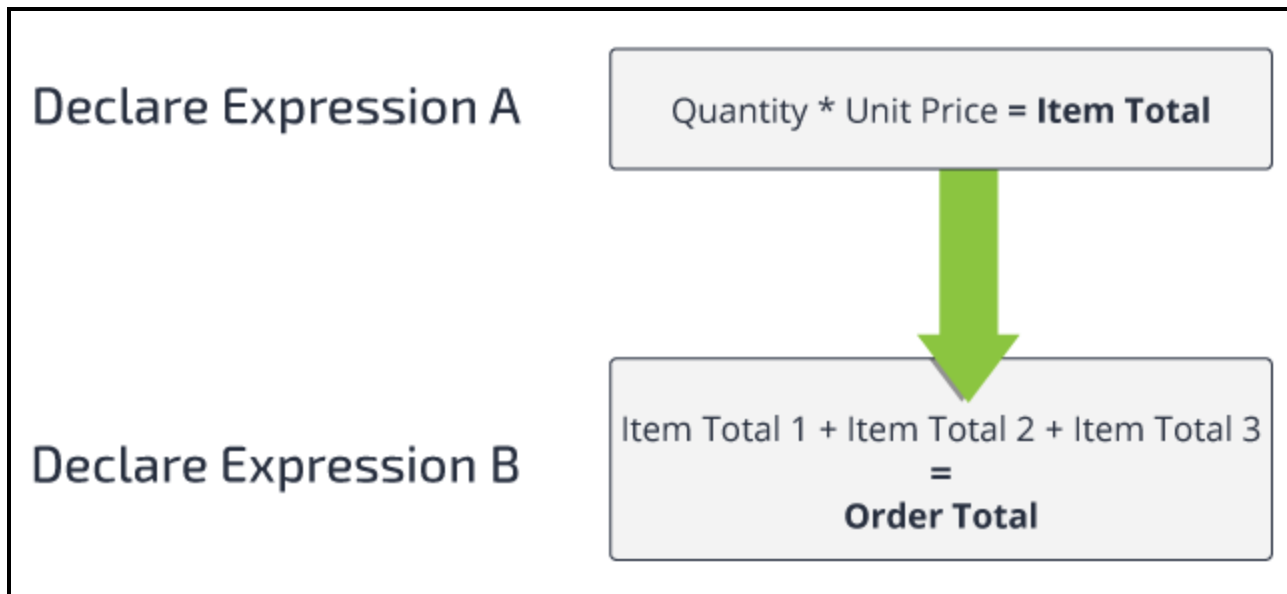
What are the three components in a declare expression?

A target property, an expression, and a source property

Declarative networks

You can use a sequence of interdependent declare expressions in a **declarative network**. A declare expression in a network can use a target property from another declare expression as a source property.

For example, assume you added an Order Total field to your purchase order form. This field uses a declare expression to calculate its target property — order total. The source property in this expression uses the item total target value.



When a user updates a Quantity value, the system updates the Order Total value.



Pega provides the Declarative Network Analysis tool to display a list of declarative networks in your application. Access this tool by selecting the **Designer Studio > Process & Rules > Business Rules** menu. For more information about using the Declarative Analysis Network tool, see the help topic [Business Rules landing page](#).

KNOWLEDGE CHECK



A declare expression in a network can use which property as a source property?

A target property from another expression

How to set a property value with a declare expression

Setting a property value using a declare expression involves three major steps. You first define the target property — the value that is updated when a declare expression calculation is performed. Then, you define the computation type. Finally, you define the expression that calculates the target property value. You can create declare expressions in either the Case Designer or in Designer Studio.

Creating a declare expression in the Case Designer

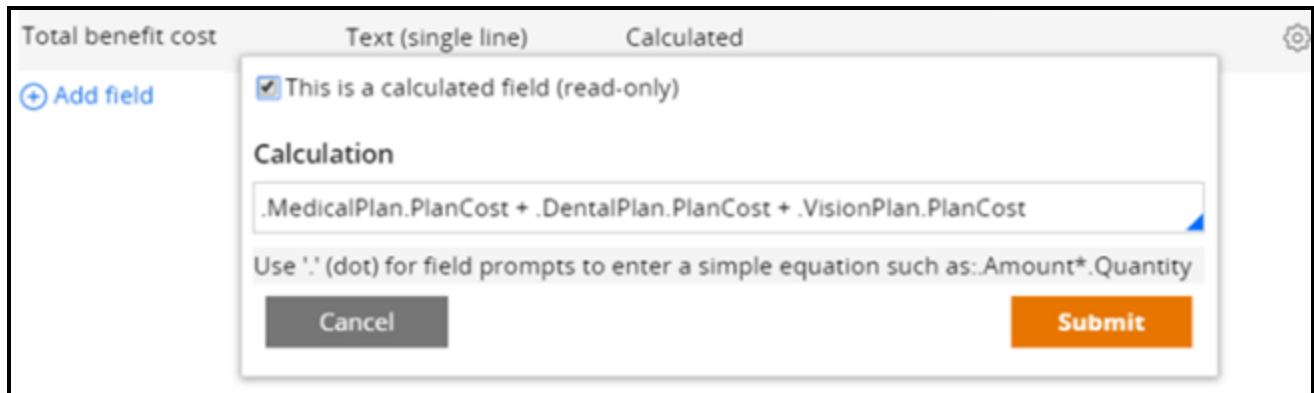
In the Case Designer on the **Data model** tab, select a value mode property you want to use as a target. Then, in the right-hand column, select the **gear icon** to define the value as a calculated field in the Calculation dialog.

Note: If you select a numeric type property such as decimal or currency, select the **Custom** function.

Enter the source or sources for the expression in the **Calculation** field. This means that the source for the declare expression is a calculation performed on one or more property values.

When you submit your calculation, Pega Platform automatically creates a declare expression using the same name as the target property.

In the following example, the declare expression is configured to calculate the total cost for an insurance benefit plan:



The screenshot shows a dialog box for configuring a calculated field. The title bar reads "Total benefit cost" and "Text (single line) Calculated". Inside the dialog, there is a checkbox labeled "This is a calculated field (read-only)" which is checked. Below this is a section titled "Calculation" with a text input field containing the expression: ".MedicalPlan.PlanCost + .DentalPlan.PlanCost + .VisionPlan.PlanCost". A hint below the input field says "Use '.' (dot) for field prompts to enter a simple equation such as: Amount*.Quantity". At the bottom of the dialog are two buttons: "Cancel" and "Submit".

Setting the TotalBenefitCost property to the sum of the PlanCost properties associated with each insurance plan property selection calculates the total cost of your employee health plan benefit.

You can add the TotalBenefitCost property as a read-only property on case views to show total employee cost as health insurance plans are selected.

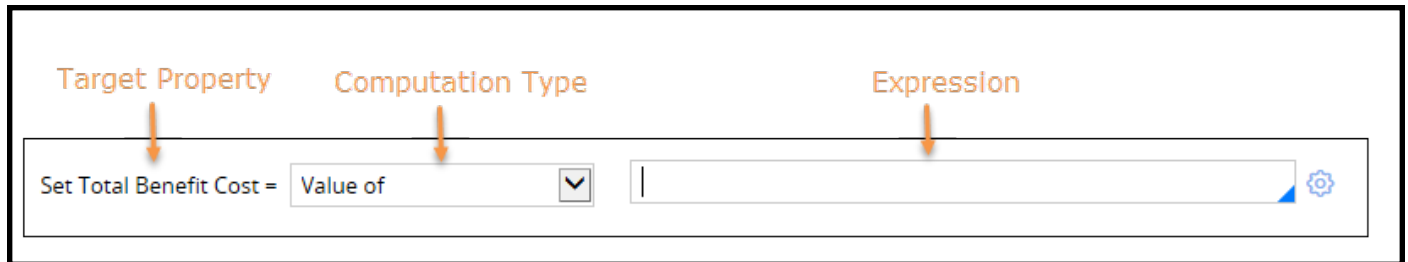
When the user selects a type of insurance plan, the total employee cost automatically updates in the view.

Note: If the target is a numeric type property, you can perform functions such as sum, average, minimum, or maximum on numeric items on a page list.

Creating a declare expression Designer Studio

You enter the target property as a key part of a declare expression rule when you create one in Designer Studio. The easiest way to create a declare expression is to select the target property in the Application Explorer. Right-click and select the **Define Expression** option.

After you have selected a declare expression, on the **Expression** tab, configure the expression in a row as shown in the following example. The row consists of three fields. The target property is entered for you. You select the Computation Type from the drop-down list. You enter the source or sources in the Expression field.



The drop-down next to the target property field allows you to select the type of computation for the expression. The default is **Value of** as shown in the previous example. This means that the source for the expression is one or more property values. You can select other options. The choices available depend upon the target property type. Options include the value of the first matching property. You can also use the result of a decision tree, decision table, or a map value to provide a value.

Note: Value of is the default computation type when you create an declare expression in the Case Designer.

You enter an expression in the form of a function and its inputs. You can also use the gear icon on the right side of the field if you want to build your expression using Pega standard functions such as CompareDates or getLocalizedValue in the Expression builder. For more information on Expression Builder, see the help topic [Building expressions with the Expression Builder](#).

The declare expression created in the Case Designer for calculating the total benefit cost in the previous example looks like the following:



Note: You can configure a declare expression to control how it performs its computations in a declarative network. This feature is discussed in the Forward and backward chaining in declarative networks topic later in this lesson.

KNOWLEDGE CHECK

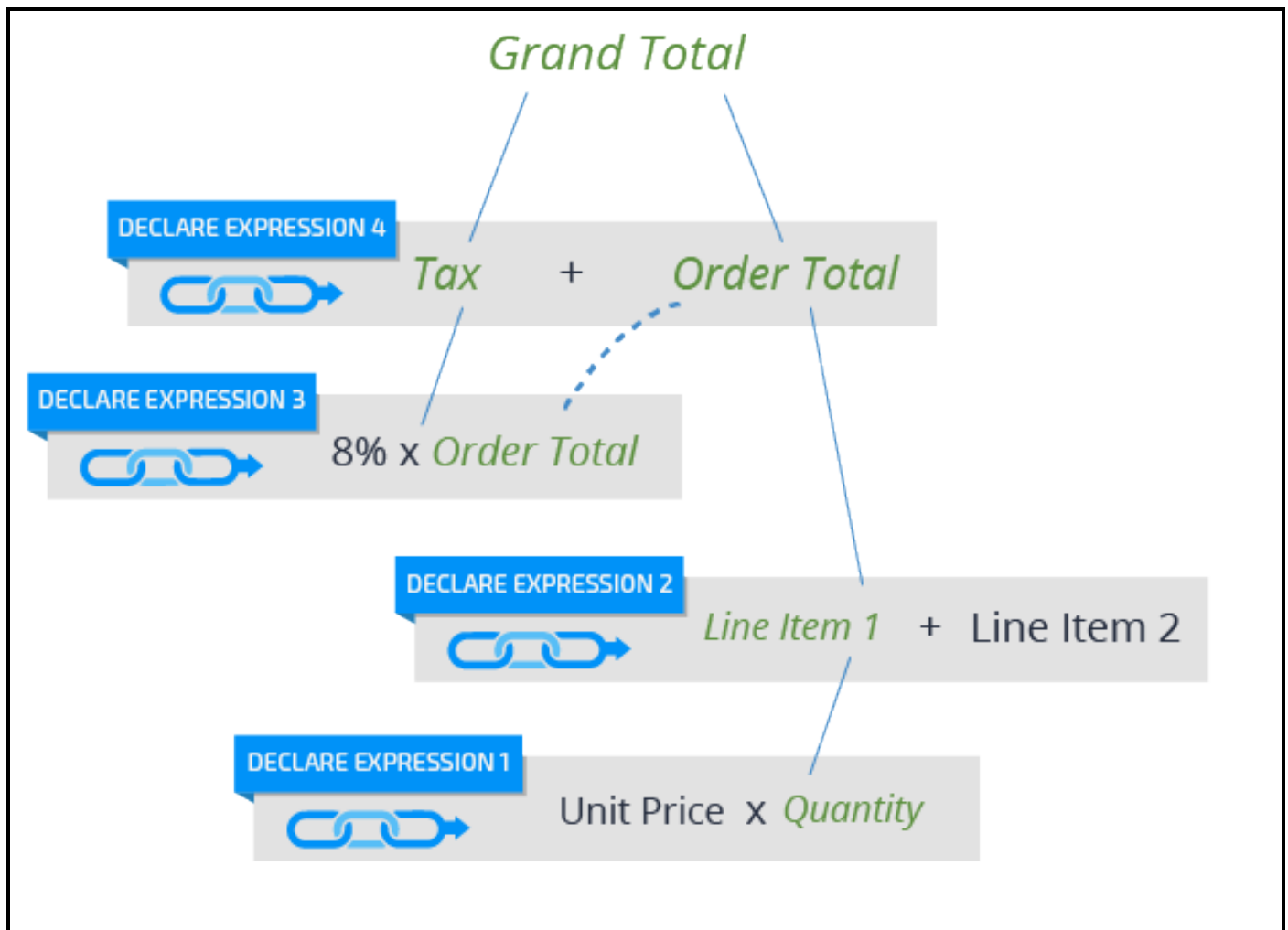


Where are the source values entered when creating declare expressions in Case Designer and Designer Studio?

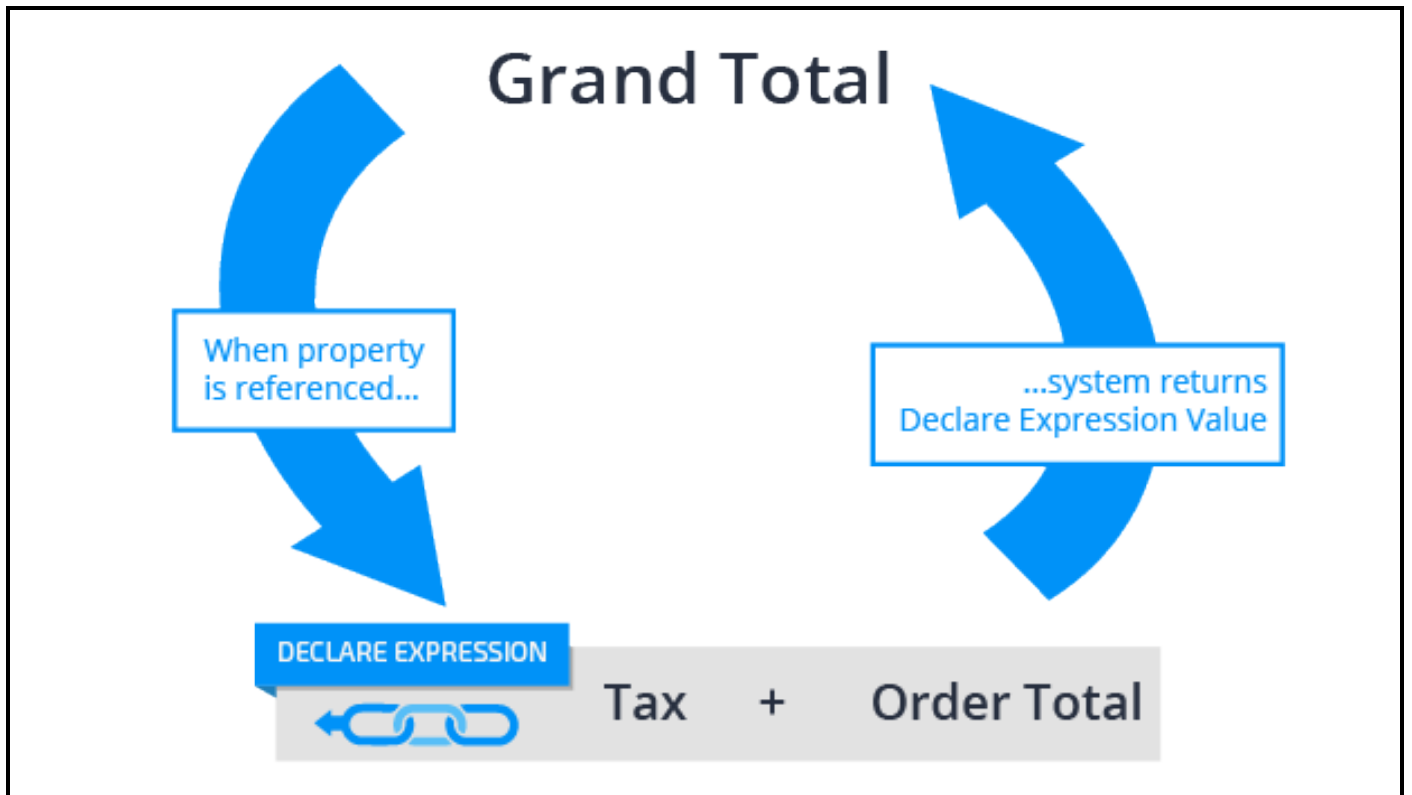
In Case Designer, the source values are entered in the Calculation field. In Designer Studio, the source values are entered in the Expression field.

Forward and backward chaining in declarative networks

Forward chaining in a declare expression updates the target property value when a source property value changes. For example, when you display a shopping cart where users add items and need the cart to reflect the total based on the changes immediately, choose forward chaining. By default, declare expressions use forward chaining. Declarative networks are commonly designed with declare expressions that are configured for forward chaining.



Backward chaining in a declare expression means that a target property value is not automatically updated when other declare expressions in a network update their target values. An expression using backward chaining only updates its target property when the application references the property by name. A form, a decision table, or a data transform can reference the property. When the property is referenced, the expression goes back in the network to reference the source property or properties the expression needs to update its target.



Specifying the chaining method

You can specify forward or backward chaining on the **Change Tracking** tab on the Declare Expression rule form. Use the **Whenever Inputs Change** option for forward chaining. There are three backward chaining options. For example, select **Whenever used** if you want the declare expression target value to be updated whenever the property is referenced in a form.

KNOWLEDGE CHECK



When does an expression using backward chaining update its target property?

When the application references the property by name

Chaining and performance

To optimize the chaining modes in a declarative network, consider where the source property is referenced and how the target property is referenced. Forward chaining can slow system performance if an expression uses many source properties that change frequently. For example, assume you are calculating the value of a home insurance quote based on more than 20 property values such as location, tax assessment, appraisal value, and land area. These values are collected in a large number of forms. When you use forward chaining, the home insurance quote declare expression recalculates the value every time users enter or change any of these 20 values. The impact to performance might affect response time when the user enters values or submits forms.

If you are only going to display the insurance quote after you collect all the values, use backward chaining for the home insurance quote expression. When you display the home insurance quote on the form, the expression performs the update only once. If you use forward chaining, the system performs the calculation even if the user does not see or need the value.

For more information about declare expressions and chaining, see the help topic [More about Declare Expression rules](#).

KNOWLEDGE CHECK



When can forward chaining have a negative impact on performance?

When an expression uses many source properties that change frequently

Configuring a work party

Introduction to configuring a work party

Pega applications allow system architects to describe the people and organizations interested in a case by their role, such as Customer or Manager. Identifying a party to the case by role allows you to describe their participation in a business process, such as receiving correspondence. For example, you can notify the Customer party of the status of their insurance claim throughout the claims adjustment process.

In this lesson, you learn how Pega describes the participants or interested parties for a case, and how you can design cases to interact with these parties throughout a business process.

After this lesson, you should be able to:

- Explain how work parties are used in an application
- List the standard work party classes available in Pega applications
- Configure a work party for a case type
- Populate a work party with case data

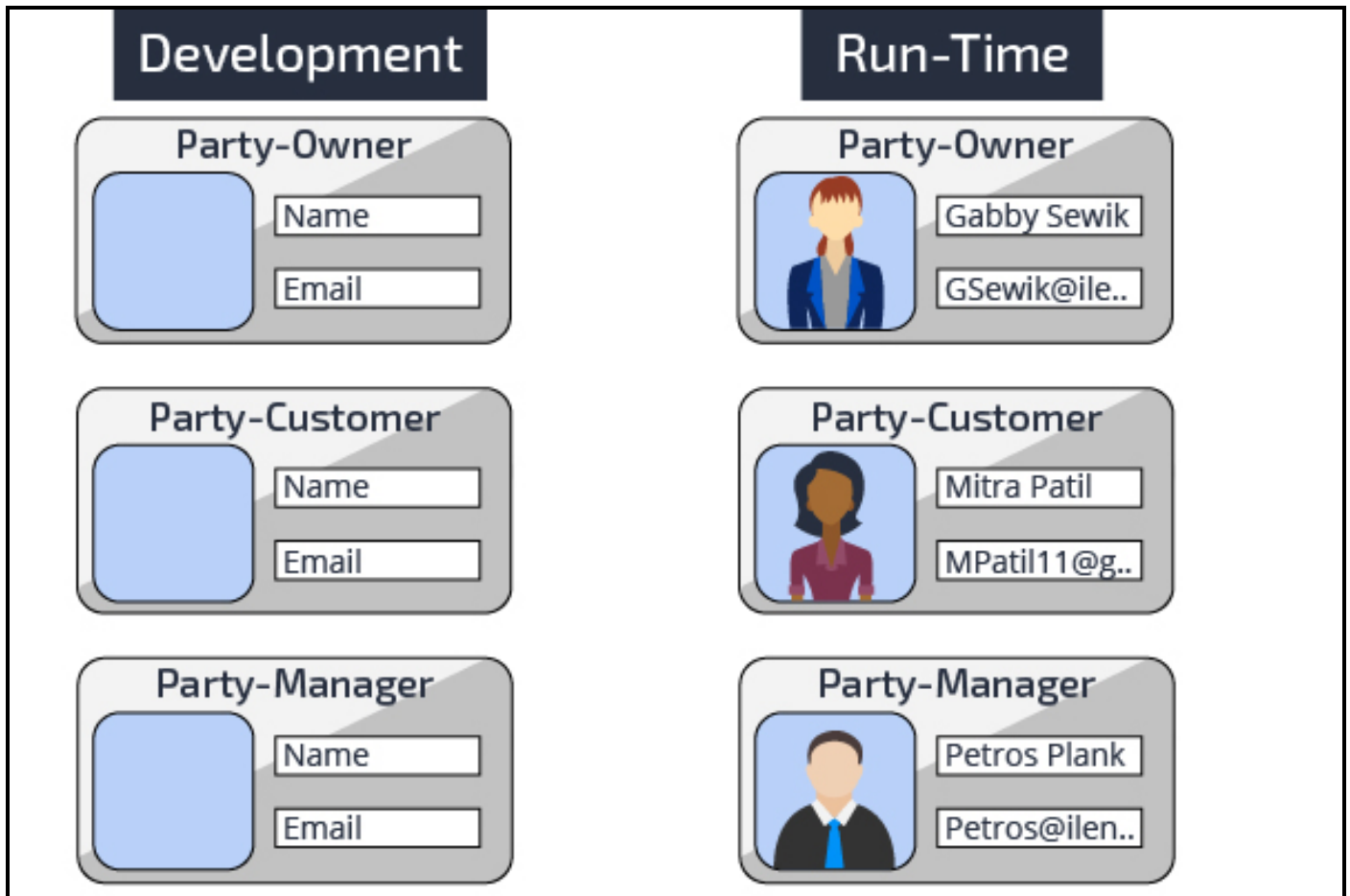
Work parties

Accomplishing work requires participants with different roles, such as managers, case workers, and customers. For example, an automobile insurance claim management process includes roles such as the customer service representative (CSR) who creates the claim, the customer, on whose behalf the claim is filed, and the claims adjustor, who reviews the claim.

In Pega, you create a **work party** to describe each role. Work parties allow you to refer to a case participant by role, without knowing any identifying information. For this reason, applications commonly use a work party as the recipient of correspondence. Also, work parties are sometimes used to assign work.

Imagine your automobile insurance claims management process requires an email be sent to every customer who submits a claim. You would need the email address for every person submitting a claim in your application. There is no practical way to get this information prior to the claim. Instead, you model correspondence by role during development and provide the identifying information (in this case, email address) for each case. So, you configure your claims management process to send an email to the Customer party to confirm receipt of the insurance claim. For each case, the application populates the work party with information about the customer submitting the claim, and sends the email to the correct customer.

In the following image, you see how the work party is defined by a system architect during development, but the information about the party is provided only at run-time when a user processes a case.



KNOWLEDGE CHECK



How is a work party used in an application?

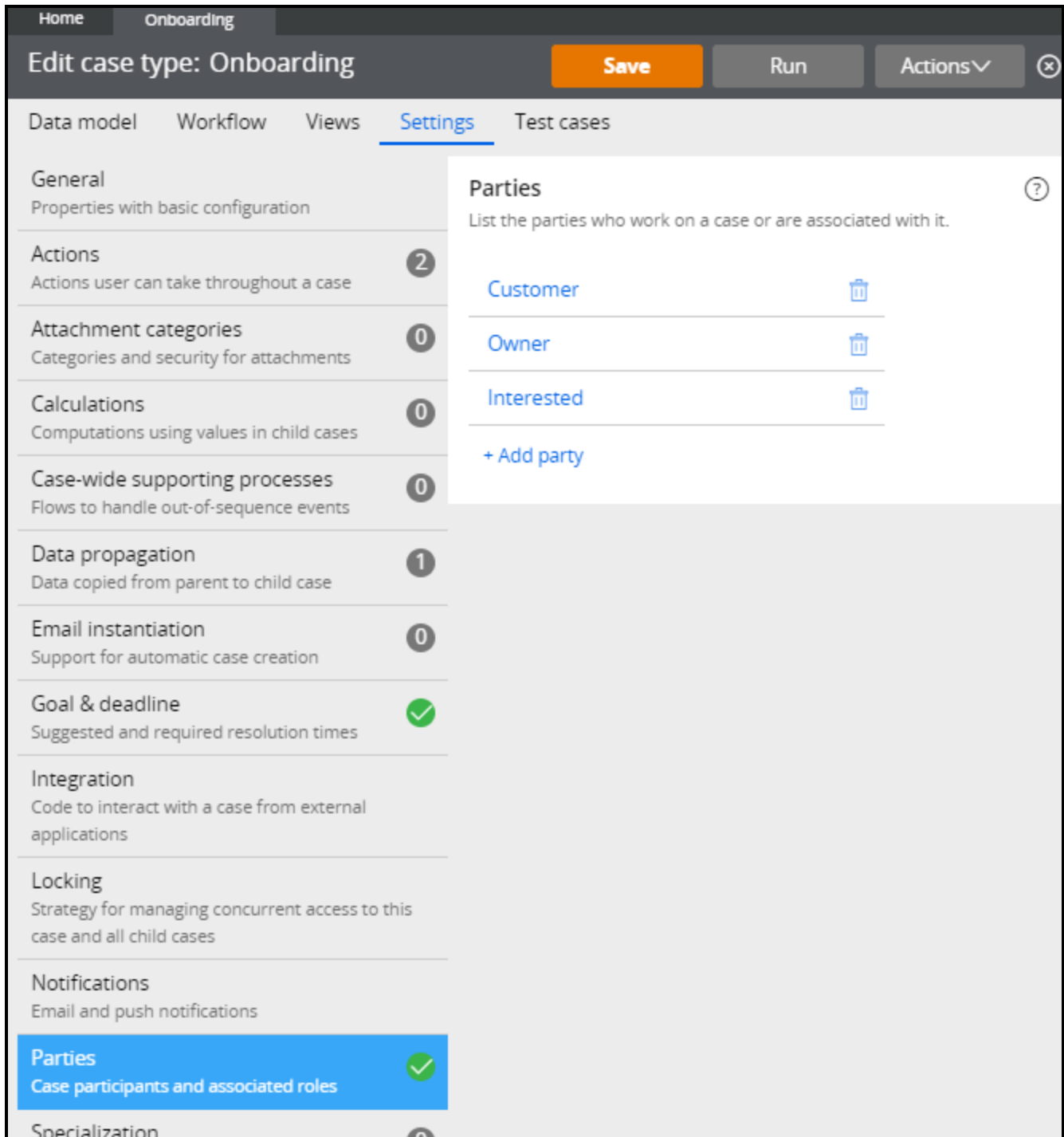
A work party represents a case participant. A work party allows you to refer to the participant by their role, and is often used to send correspondence during case processing.

How to add a work party to a case

Adding a work party to a case requires you to perform two actions: define the party and populate an instance of the party at the desired steps in your case workflow. To facilitate this, Pega creates a Work Parties rule named *pyCaseManagementDefault* for each case type. To add or modify a work party for a case type, use the **Settings** tab on the Case Designer to make your changes under the **Parties** section. Behind the scenes, Pega edits the *pyCaseManagementDefault* work parties rule created in the class of the case type.

Define the work party

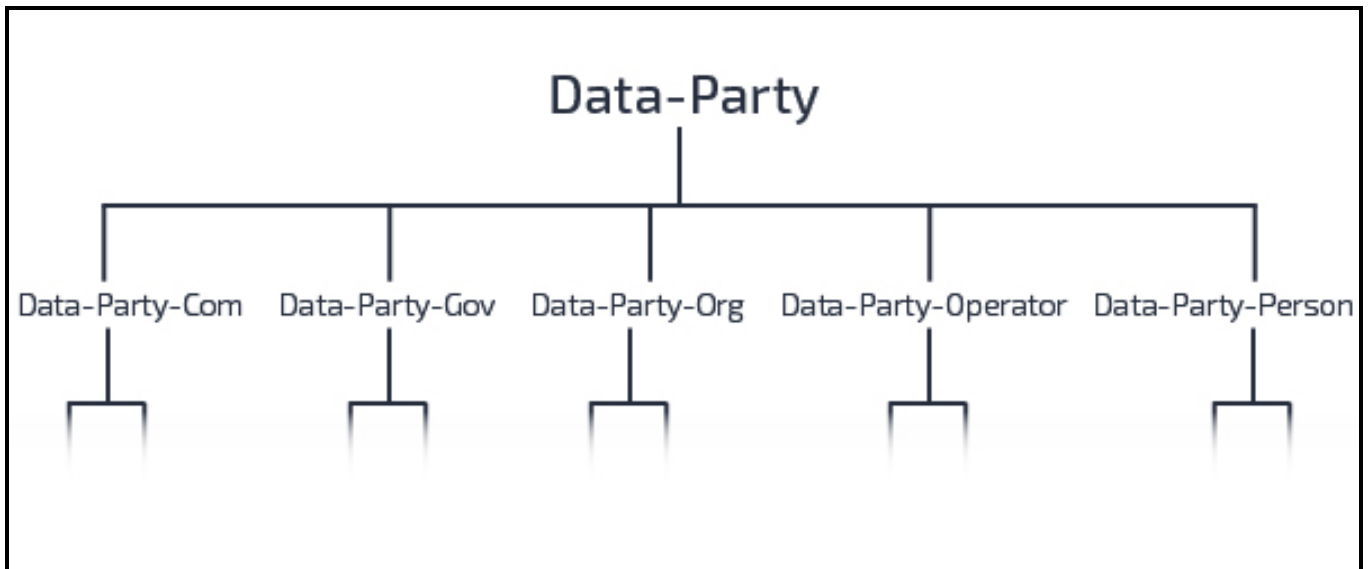
First, in Case Designer, navigate to the **Settings** tab of the case type and select **Parties**. Here you can add or modify your work parties. By default, three parties are available for a case type: Customer, Owner, and Interested. For example, a customer is the subject of a dispute resolution case, the customer service representative (CSR) is the owner of the case and responsible for resolving the dispute, while a co-signer of the credit card may be an interested party.



When you create a work party, consider the following information:

- What type of case participant do you need to model? Is the participant an individual or an organization? Does the party work on the case, or only receive status updates?

In Pega, work parties derive from the *Data-Party* class. *Data-Party* contains rules to describe a party, such as properties to store identifying information. Pega also provides five child classes that build upon the rules in *Data-Party*. These child classes represent specific types of persons and organizations. The party class for a work party describes the person or entity participating in the case and determines how the participant interacts with other participants in the business process.



Data-Party-Com models a business that has a web domain ending in .com, such as a corporation.

Data-Party-Gov models a government agency, such as the Department of Revenue.

Data-Party-Org models a non-profit organization that has a web domain ending in .org, such as a charity.

Data-Party-Operator models a case participant with a Pega login and represents a case participant, such as a case worker or case manager.

Data-Party-Person models a case participant who lacks a Pega login, such as a customer. Typically this class is used to describe a work party that receives correspondence about a case, but who does not perform any actions on the case.

- What information do you need to know about the party? For example, do you only need the name and email address, or do you also need to know their marital status?
- How will you obtain the information to populate the party details? Do you expect the user to enter this information? Can you copy the information from session information or case data?
- Is the information available when the case is created? Or does the information become available during case processing?
- Are multiple instances of the party associated with the case? For example, does the case involve one customer or several customers?

Populate the work party with case data

Next, configure your process to populate the work party with participant-specific information. Pega provides different options for populating work party information during case processing.

- If your cases already include the data needed to populate the party, use the *addWorkObjectParty* API activity. For example, a case creates several child cases to manage IT tasks prior to an employee's first day at a company. Each child case includes the HR representative assigned to the parent case as a work party. You use the *addWorkObjectParty* API activity to create an HR Partner work party using information already included in the case. The activity parameters allow you to specify the information needed to create the work party. Call the activity by adding a utility shape to the appropriate process.

- If you want case workers to provide party information when creating a case, select the **Display on creation** option for the party in the **Case Designer Parties** configuration section. When you enable this option, Pega automatically presents users with a form to enter the information needed to create the work party. For example, a CSR must enter customer information to open a new savings account for the customer. The initial form in the new account case allows CSRs to create a work party using information provided by the customer.

Configuring a work party for a case type

In Pega applications, work parties are defined using the work parties rule *pyCaseManagementDefault*. To use a work party in a case, use the Case Designer to add the work party for the case type.

Defining a work party for a case type

Create a work party to describe the role of an individual or organization in a business process.

1. In the Case Designer, select the appropriate case type, then navigate to the **Settings** tab and click **Parties**.
2. Click the **+ Add party** to add a work party.
3. In the **Role** field, enter a unique name that indicates your work party's relationship to a case, such as *Lawyer* or *Design Manager*.

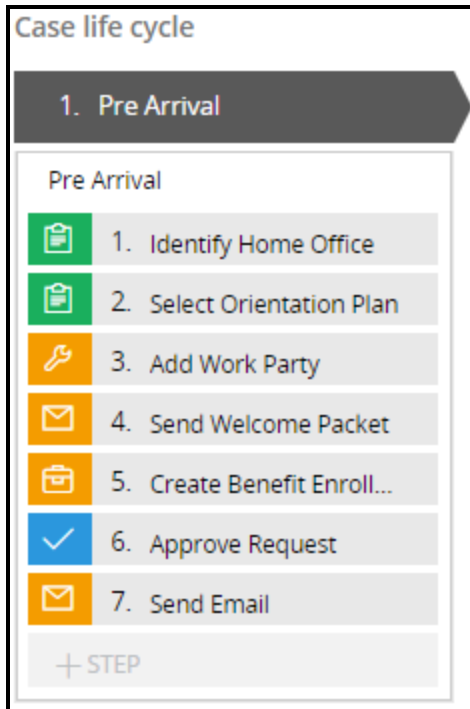
Note: The party role must be a unique value and identifies the party on the clipboard. Each work party is a page within the WorkParty page group, and the role is used as the page index. For example, a work party with the role *Customer* is identified on the clipboard as *WorkParty (Customer)*. The *Owner* party role is reserved to describe the operator who creates the case.

4. Select an option from the **Type** drop-down to identify the class used to define the party, such as *Data-Party-Company*.
5. Optional: In the **Description** field, enter a label that identifies the party role and is displayed on the case to identify the work party.
6. Optional: Press the down arrow in the **Data transform** field and select the name of a data transform that runs when users add the work party to a case. The data transform defines initial values for the work party. You can use a data transform to copy information, such as a first name or email address, to the work party. Specify a data transform if you plan to create the work party upon creating a case. The data transform must be in the party class or a parent class.
7. Optional: Select the **Display on creation** check box to prompt users to add this work party every time a case is created.
8. Optional: Select the **Required** check box to indicate that this work party must be present in every new case. Selecting this check box adds required fields to the work item entry form to collect information about the party.
9. Optional: Select the **Allow multiple** check box to allow more than one instance of your work party to participate in a case.
10. Click **OK** to apply your definitions.
11. Click **Save** in the Case Designer to save your changes.

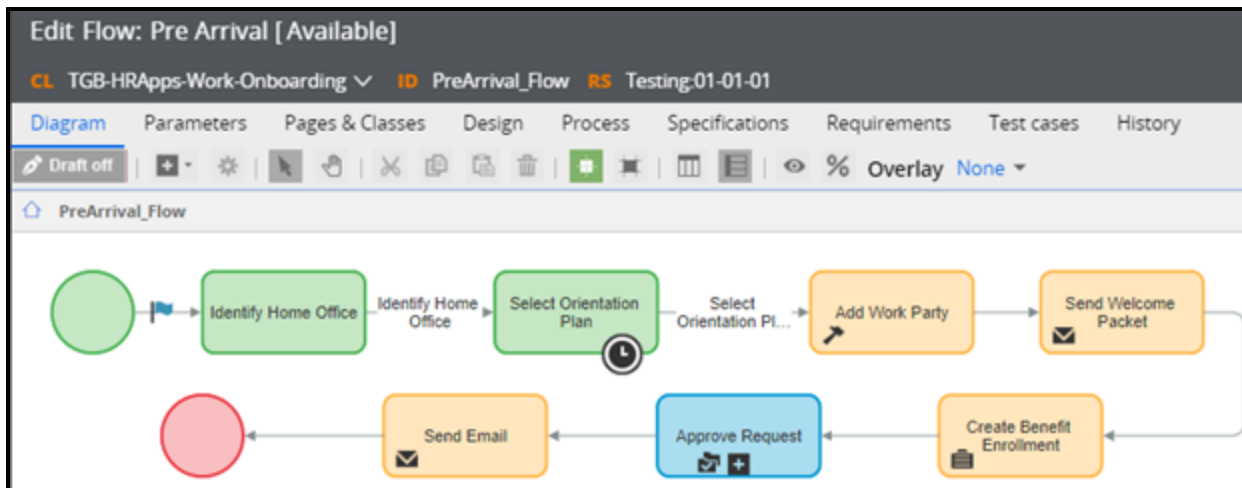
Tip: You can also define or edit a work party directly on the work party rule. To do so, go to the Application Explorer, expand the appropriate case type, and select **Process > Work Parties > pyCaseManagementDefault**.

System tasks

One goal of a Pega application is to make a business process more efficient by identifying tasks that can be automated. Tasks that are good candidates for automation do not require user input or decisions. For example, you can automate adding a work party provided the application already has access to the data necessary to define a given work party. You can add automated system tasks as an additional step in a process. In a step, an orange icon represents the system task.









The Case Designer allows you to directly add over a dozen automated tasks, such as sending an email or creating a child case. Additional system tasks beyond what is available in the Case Designer can be added using the Process Modeler, which is available on a flow rule. Whenever you add a process to a case life cycle in Case Designer, Pega automatically creates a **flow rule**. The Process Modeler provides the graphical interface to edit the flow rule, which stores the sequence of events for your business process.



The Process Modeler gives you more flexibility to add additional flow shapes that represent a step or task accomplished as part of the business process. You can sequence the shapes with connectors to accurately model a business process in your application. Flow shapes are differentiated by color, symbol, and name. For example, you use a Utility shape to add an automated system action that does not require any human intervention. The Utility shape allows you to reference an activity that defines the automated processing to perform.

Below are some common shapes Pega provides that enable you to model your process.

Shape	Name	Use
	Start	The Start shape indicates the beginning of flow processing. Each flow rule must contain a single Start shape. A single Start shape is automatically added to every flow rule.
	Assignment	The Assignment shape creates a task in a work list or work queue so a user can provide input to the case.
	Subprocess	The Subprocess shape indicates a reference to another flow rule from the current flow rule. Portions of a process can be divided into a smaller process to enable reuse in other flow rules. The standard Approve/Reject step completes via subprocess.
	Utility	The Utility shape indicates an automated system action. Pega executes automated system actions without requiring human intervention. Examples of automated system actions include changing the stage of the current case, sending an email, or creating a new case.
	End	The End shape indicates the end of flow processing. Each flow rule may include one or more End shapes to represent the potential end points of the process.
	Connector	Each flow shape connects to other flow shapes through the use of a connector. The Connector defines the sequence of flow execution. The flow execution begins with the Start shape and proceeds from one shape to the next in the order the shapes connect to each other.

KNOWLEDGE CHECK



Which standard flow shape represents an automated system action?

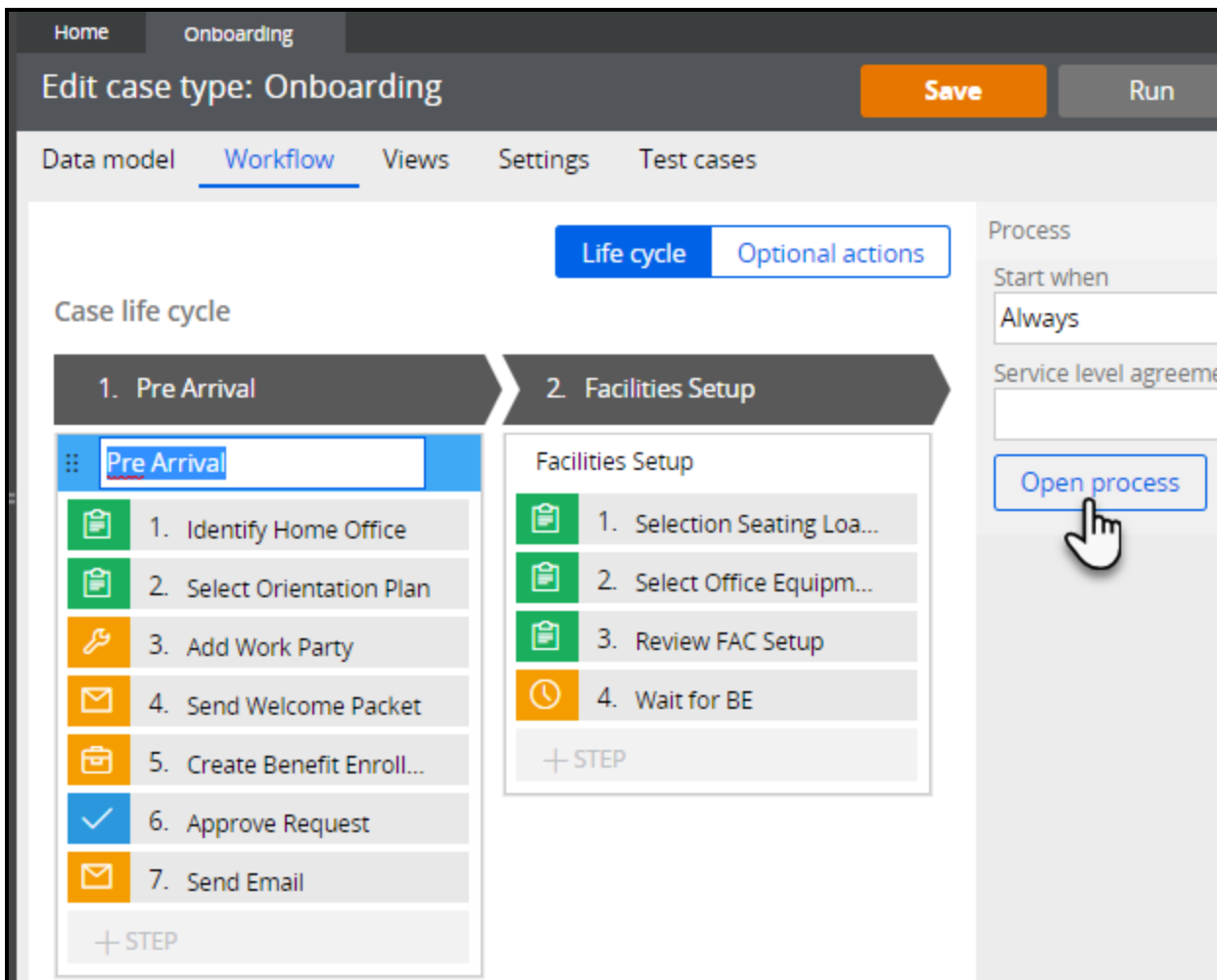
Utility

How to add system tasks to process flows

Use Pega Express or Case Designer to add standard processes used to define the case life cycle. You can then use the Process Modeler to add advanced features to the processes such as automated system tasks.

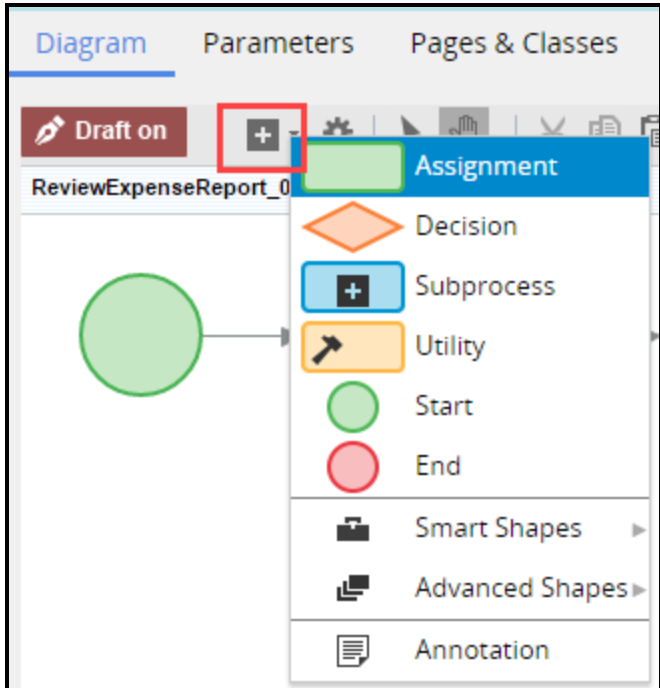
Access the Process Modeler in a flow rule

1. On the **Workflow** tab of the Case Designer, click the name of the process to open a contextual panel on the right side of the Case Designer.
2. On the right, in the contextual panel, click **Open process** to display the flow rule.



Add a shape using the Flow Shapes menu

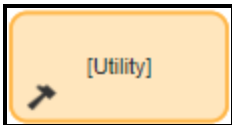
1. On the **Diagram** tab of the flow rule, click the plus icon to open the **Flow Shapes** menu.



2. Click and drag the **Utility** shape onto the diagram. A rectangle icon follows the pointer to indicate where the flow shape is added.



3. Release the mouse button. The **Utility** flow shape appends to the process with a generic name that identifies the purpose of the flow shape.



Note: To quickly add a shape to the diagram, click the shape once from the Flow Shapes menu. After you add the shape to the diagram, drag the shape to the desired position on the diagram.

4. Double click the **Utility** shape. The **Utility properties** panel is displayed.
5. Under the **Rule** text box, enter the name of the activity.


Note: Tab away from the **Rule** text box to show the list of activity parameters.

6. Enter any parameters as defined by the activity.

Utility properties

Utility:

Perform automated processing in your case.

 **Automation details**
Define automation

Rule★

Parameters

PartyRole	Customer
PartyClass	Data-Party-Person
PartyModel	Customer
PartyRepeatable	<input type="checkbox"/>
partyPage	
PartyIDUsed	
OutputPageName	pyOutput
OutputPageClass	Code-ProcessOutput

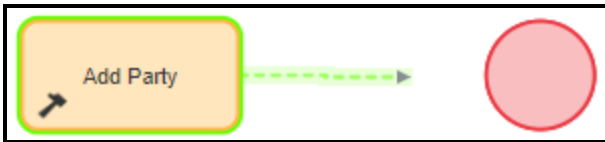
Audit note

7. Click **Submit** to save your changes.

Connect flow shapes with a connector

Connectors indicate the order of steps in a process flow.

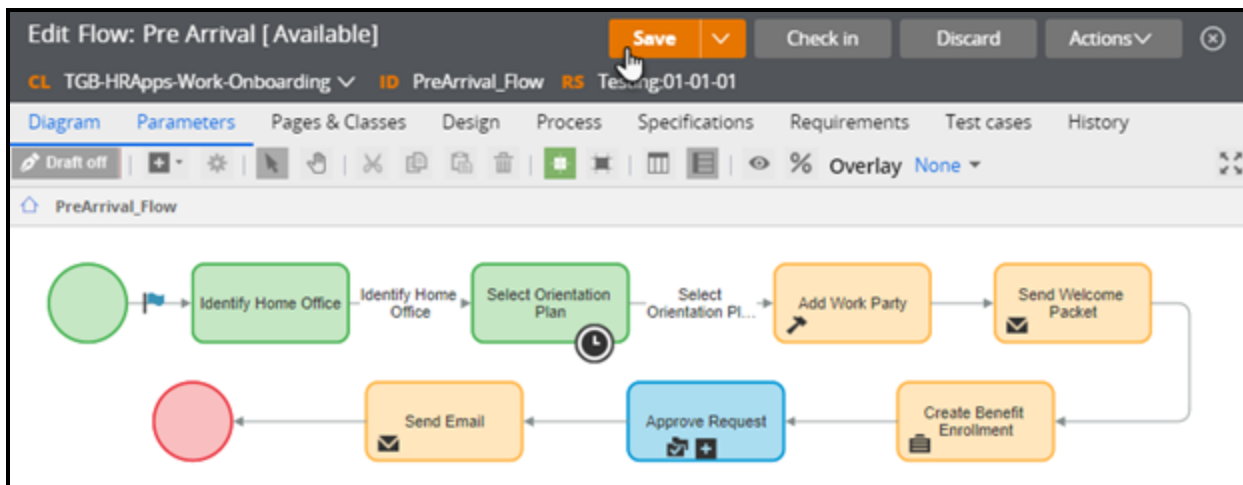
1. Position the pointer over the flow shape from which you want to connect. A set of connector points is displayed on the border of the flow shape.
2. Click the connector point from which you want to start the connector. A green square highlights the flow shape.
3. Click the mouse button and drag the pointer to draw the connector. A green dashed line indicates the path of the connector.



- Position the pointer over the shape on which you want to end the connector. A green highlight surrounds the flow shape and the connector points are displayed on the border of the flow shape.



- Release the mouse button to connect the connector to the shape.
- Save and close the flow rule to return to the Case Designer.



Exchanging data between cases

Introduction to exchanging data between cases

When creating and managing cases, applications often copy data values from one case to another. For example, information about an insured party can be copied from an insurance claim case to a medical claim child case, or cost information from a set of child cases can be summed on a parent purchase order case. By sharing information among case types, you can make data-driven decisions.

After this lesson, you should be able to:

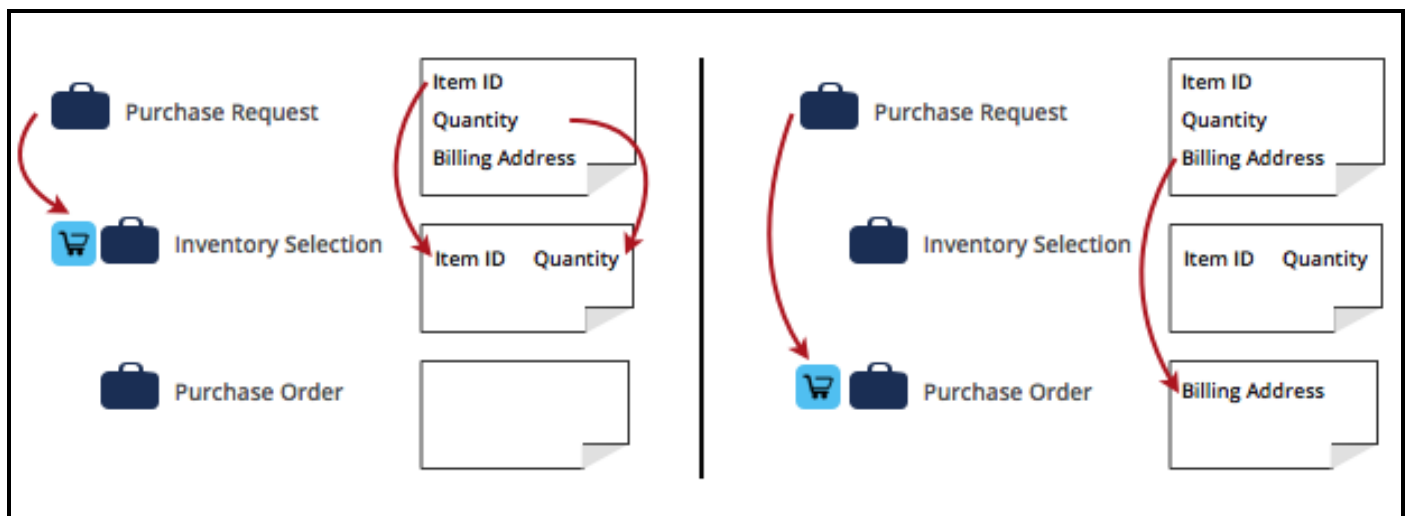
- Explain the role of data propagation in case processing
- Explain the role of case calculations in case processing
- Select the appropriate option for passing data from one case to another
- Copy data from one case to another during case processing

Data propagation

The role of data propagation

Data propagation is the mechanism of copying data within the case hierarchy. By sharing data among cases, you save time by eliminating the manual task of copying data and you provide relevant information to caseworkers.

Data propagation ensures that the appropriate information is moved into a child case. The following image illustrates an example of a purchase request case that initiates an inventory selection child case when units in stock must be confirmed. In the purchase request case, each line item in the purchase request contains a product identifier and a quantity. The inventory selection child case then uses the product identifier and quantity to verify that the units are in stock.



In Pega, data propagation happens on case creation. When you propagate a property from a parent case to a child case or to a spin-off case, and the property value later changes on the parent case, the property on the child case or spin-off case does not get updated.

For example, the product identifier is set to 0211 and the quantity is set to 3 in a purchase request. These values are propagated to the inventory selection child case on creation. If the quantity in the purchase request later changes to 4, the value is not automatically propagated to the inventory selection child case. The quantity in the inventory case will remain set to 3. You need to handle the subsequent syncing of data between the cases separately.

Data propagation options

You specify the properties to propagate from a parent to a child case in one of the following ways:

- In Pega Express, within a case life cycle, you can define properties directly one by one when creating a child case.
- In Designer Studio, in the Case Designer, you can define properties directly one by one or specify a data transform. (You use a data transform, if you need some conditional logic to determine what to

propagate.). For example, you might use a data transform to loop through a list and only propagate items that were selected.

- In Designer Studio, in the Case Designer, you can create child cases and spin-off cases using the Create Case step. The Create Case step allows you to copy property values using a data transform.

Other options for copying data between cases

Data propagation specifically refers to copying data from one case to a second case when the second case is created. Pega Platform provides additional options for updating data between existing cases during case processing.

Updating case data

If a property value later changes on the parent case, the property on the child case does not get updated. You can update a child case, or all the child cases of a parent case, after the case has been created. This option is available only in the Case Designer in Designer Studio.

Summing calculations in child cases

Case calculations allow you to sum values from child cases back to the parent case. For example, if a purchase request generates multiple purchase orders, a case calculation on the parent purchase request can sum the order total for each of the purchase order child cases. Case calculations are available only in the Case Designer in Designer Studio.

Propagating data when creating a case

Data propagation to a child case can be configured in Pega Express, with additional options available in Designer Studio, using the Case Designer.

Data propagation to a spin-off case is configured in the Designer Studio, using the Case Designer.

Define data propagation in Pega Express

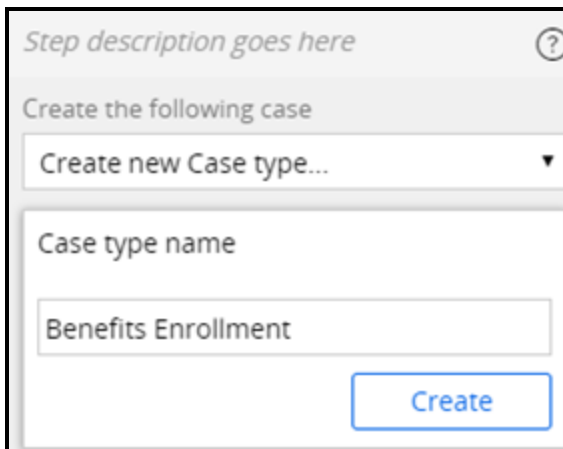
In Pega Express, data propagation is configured on the Create Case utility, which lets you create a case in the context of your current case.

Follow these steps to configure data propagation for a case:

1. In the navigation panel, click **Cases**, then click the name of the case type. The case life cycle is displayed.
2. In the case life cycle, from the **Workflow** tab, hover over a process in a stage, and then click **+ Step > More > Utilities > Create Case > Select**.

The Create Case properties are displayed in the panel on the right.

3. In the properties panel, in the **Create the following case** field, select **Create new Case type**. A pop-up displays the Case type name field.



4. In the **Case type name** field, enter a unique name for the case type and click **Create**. A child case type is added to your hierarchy of case types.
5. In the properties panel, select **Transfer information to new case**. The Transfer Information dialog is displayed.

Important: When you select **Transfer information to new case**, the selected case type converts to a child case.

Transfer information

Map fields

Transfer the following information to 'Health Benefits' case

<input checked="" type="checkbox"/> 'From' field	'To' field
<input checked="" type="checkbox"/> Employee	Employee
<input checked="" type="checkbox"/> Location	Location
<input checked="" type="checkbox"/> Office	Office

View

Add mapped fields to 'Onboarding Info' view (in 'Health Benefits' case type)

OK

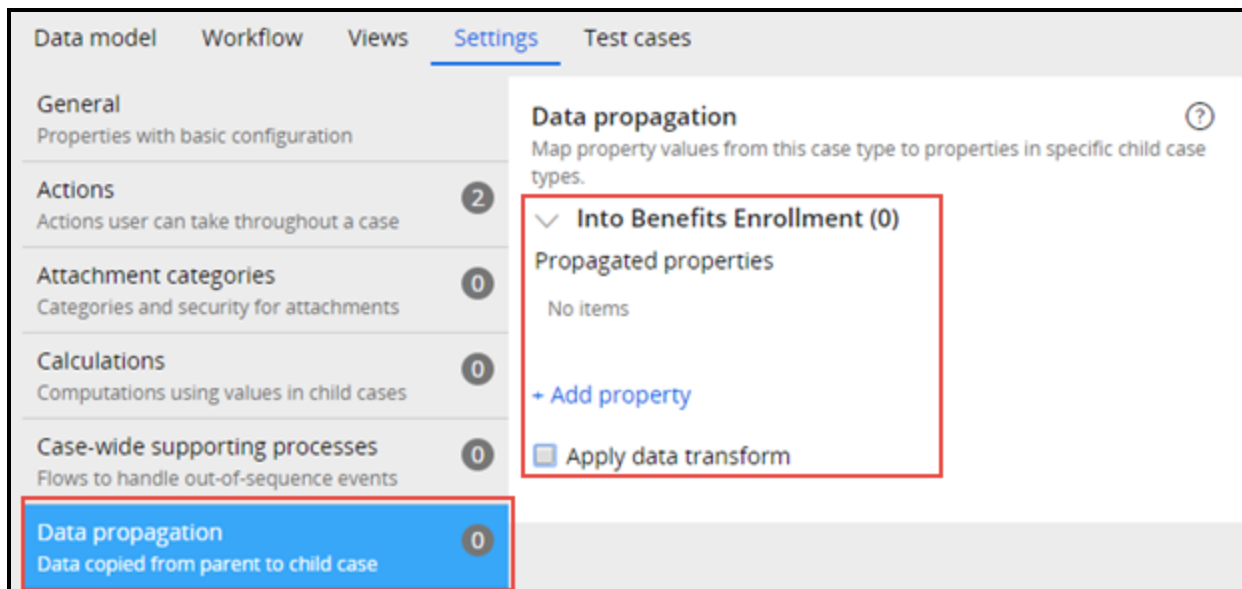
6. In the Transfer Information dialog, define how data in the current case is propagated to the new case.
 - a. Select the check box next to a **From field** to copy its value from the current case to the new case.
 - b. Optional: To change which field stores the value in the new case, enter a field name in the **To field**.
 - c. After selecting the fields you want to transfer, click **OK**.
7. Click **Save**. The new case is created with the fields propagated from the current case.

Define data propagation in the Case Designer

In Designer Studio, data propagation is configured on the parent case type using the Case Designer. The Case Designer lets you perform complex operations that are not available in Pega Express. For example, you configure a data transform rule to concatenate strings or use conditional logic to set property values.

Follow these steps to configure data propagation on the parent case:

1. In Designer Studio, select the parent case in the Case Explorer.
2. Click the **Settings** tab, then click **Data propagation**.



3. Click the child case (for example, Into Benefits Enrollment) to which you want to propagate properties.
4. In the panel on the right, below **Propagated** properties, click the **Add Property** link to select the properties you want to propagate into the purchase order.
5. If you want to specify a data transform with the data propagation settings, in the panel on the right, below **Propagated properties**, select **Apply data transform**

Copy data to a spin-off case

Data can also be copied to a case when creating spin-off cases, by applying a data transform to copy data values from the original case to the spin-off case. For example, a purchase request case might spin off a supplier case if a new supplier is provided in the purchase request.

You can use the Create Cases step utility to create child cases and spin-off cases. In this scenario, a Benefits Enrollment case is spun off. Follow these steps to configure data propagation for the Create Case utility:

1. Select the parent case in the Case Explorer.
2. In the case life cycle, from the **Workflow** tab, hover over a process in a stage, and then click **+ Step> More > Utilities > Create Case**.
3. Click **Select** . The Create Case properties are displayed in the right panel.

Step description goes here ?

Create Case
 Create child case(s)

Create the following case

Benefits Enrollment

Starting process

Benefits Enrollment

Data transform

Property to store ID of case

Audit note

Enable navigation link

4. In the Data transform field, specify a data transform for the case.
You can specify a data transform when creating a spin-off, child, or multiple child cases.

Updating data for an existing case

When you create a new case, you can configure data propagation to populate the new case with default properties. When you copy a property from a parent case to a child case or spin-off case, and the property value later changes on the parent case, the property on the child case does not get updated.

For example, consider a purchase request that has a child case for each item on the request. The parent purchase request is changed to reflect a discount for all items. To update the price for each item, you want to update the discount data on each child case.

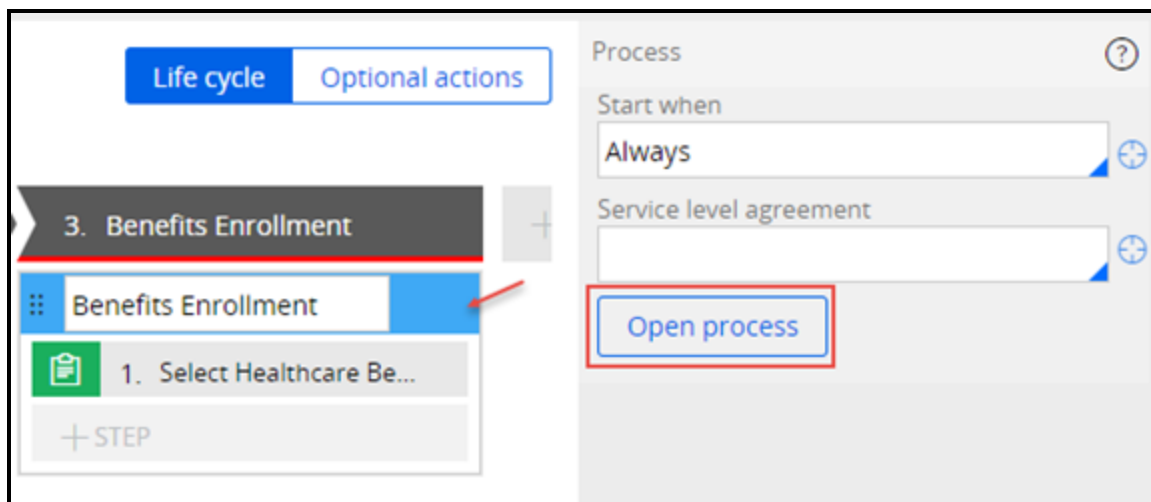
You can update a child case, or all the child cases of a parent case, by using the Update a Case smart shape in the case designer.

By maintaining case information automatically, you can save time and resolve cases more quickly.

Updating an existing case can be configured in Designer Studio, in the Case Designer. This feature is not available in Pega Express.

Follow these steps to update data for an existing child or spin-off case:

1. In Designer Studio, select the parent case in the Case Explorer.
2. In the **Workflow** tab, identify the process where you want to update the case data, select the header, then click **Open Process**.



The flow diagram is displayed.

3. Click **Check out** to create a local instance of the flow diagram.
4. Click + > **Smart Shapes** > **Update a case**. The Update a case shape is added to the flow diagram.

5. Double-click the shape to display the **Update a case** properties.

Update a case

Utility:

Perform automated processing in your case.

Automation details
Define automation

A single case
 All child cases (and descendants)

With ID*

Data transform*

Audit note

Enable navigation link

Cancel Submit

6. Indicate how many cases to update.

To update a specific case:

- a. Click **A single case**.
- b. In the **With ID**, enter the property reference or unique identifier for an open case.

To update the descendants of the current case, click **All child cases (and descendants)**.

7. In the Data transform field, click the **Down Arrow** key and select a data transform that sets property values in the case.

The smart shape requires a data transform that sets the data values to copy and the case(s) to update. A data transform is a rule that manipulates application data, such as by copying a value from one property to another, or by updating an existing value.

8. Click **Submit**. The cases you specified will be updated.
9. Integrate the Update a Case shape in the flow diagram by moving it and connecting to other shapes.

At run time, the data transform is applied to the selected cases. If an update to any child case fails, all changes made by the data transform are rolled back.

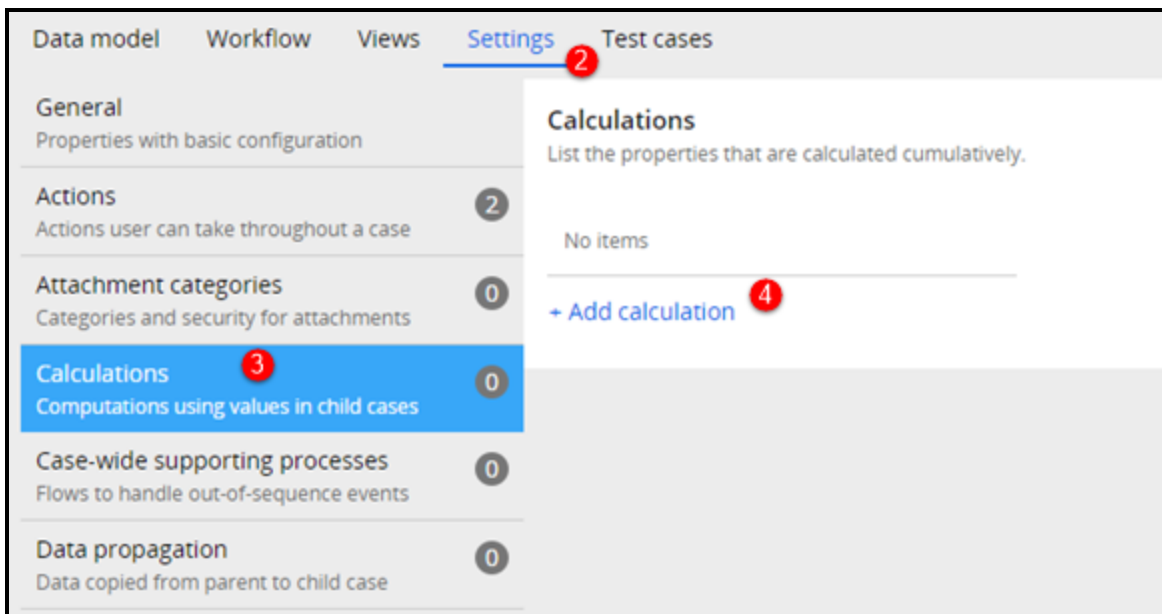
Calculating case values from child case data

Case calculations let you sum values from child cases back to the parent case. For example, consider a purchase request that generates purchase orders. The parent case is the purchase request. The child cases confirm the cost for each item on the purchase request. You can configure a case calculation on the parent purchase request to sum the total cost for all items in the order.

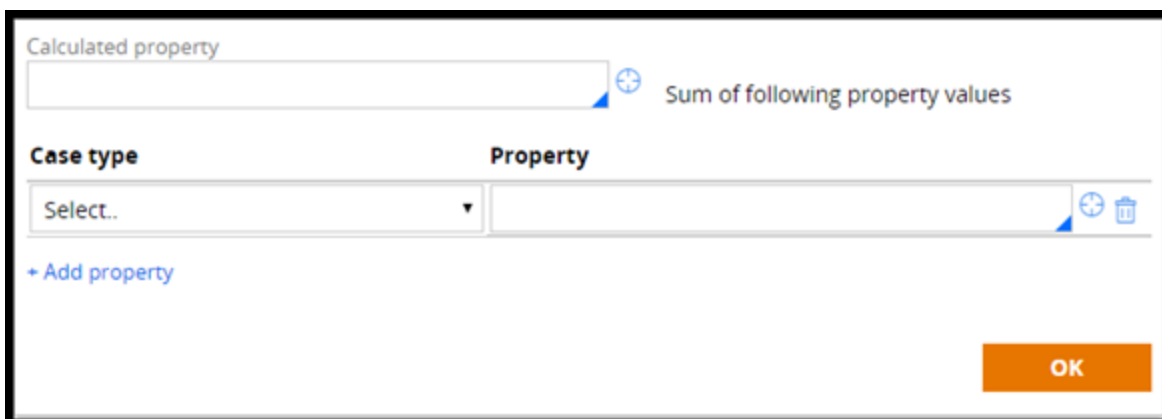
Note: Calculations are configured in Designer Studio, using the Case Designer. This feature is not available in Pega Express.

Follow these steps to configure calculations on the parent case:

1. In Designer Studio, select the parent case in the Case Explorer.
2. From the Edit case type screen, click the **Settings** tab.



3. Click **Calculations**. The Calculations properties are displayed in the right panel.
4. From the right panel, click the **Add Calculation** link to specify the properties for which you want to calculate a total for the purchase order.



5. From the right panel, press the Down Arrow key in **Calculated property** and select the property in the parent case that will store the calculated sum. This property should be a single-value property that is on a top-level page and stores an integer, decimal, or double.
6. From the right panel, click the **Case type** menu and select the child case. Then press the Down Arrow key in **Property** and select the property whose value will be added to the sum. This property should be a single-value property that is on a top-level page and is the same type as the calculated property.
7. Optionally, you can add more child case values. Click **+ Add property** to display another row where you can enter the case type and property (as described in step 6).

Caching data with data pages

Introduction to caching data with a data page

A data page loads data into memory and stores it on the clipboard, making the data accessible to an application. By storing the information in a data page, you improve application performance since the data remains on the clipboard for subsequent access. In addition to improving performance, data pages also increase maintainability of an application by abstracting the data source.

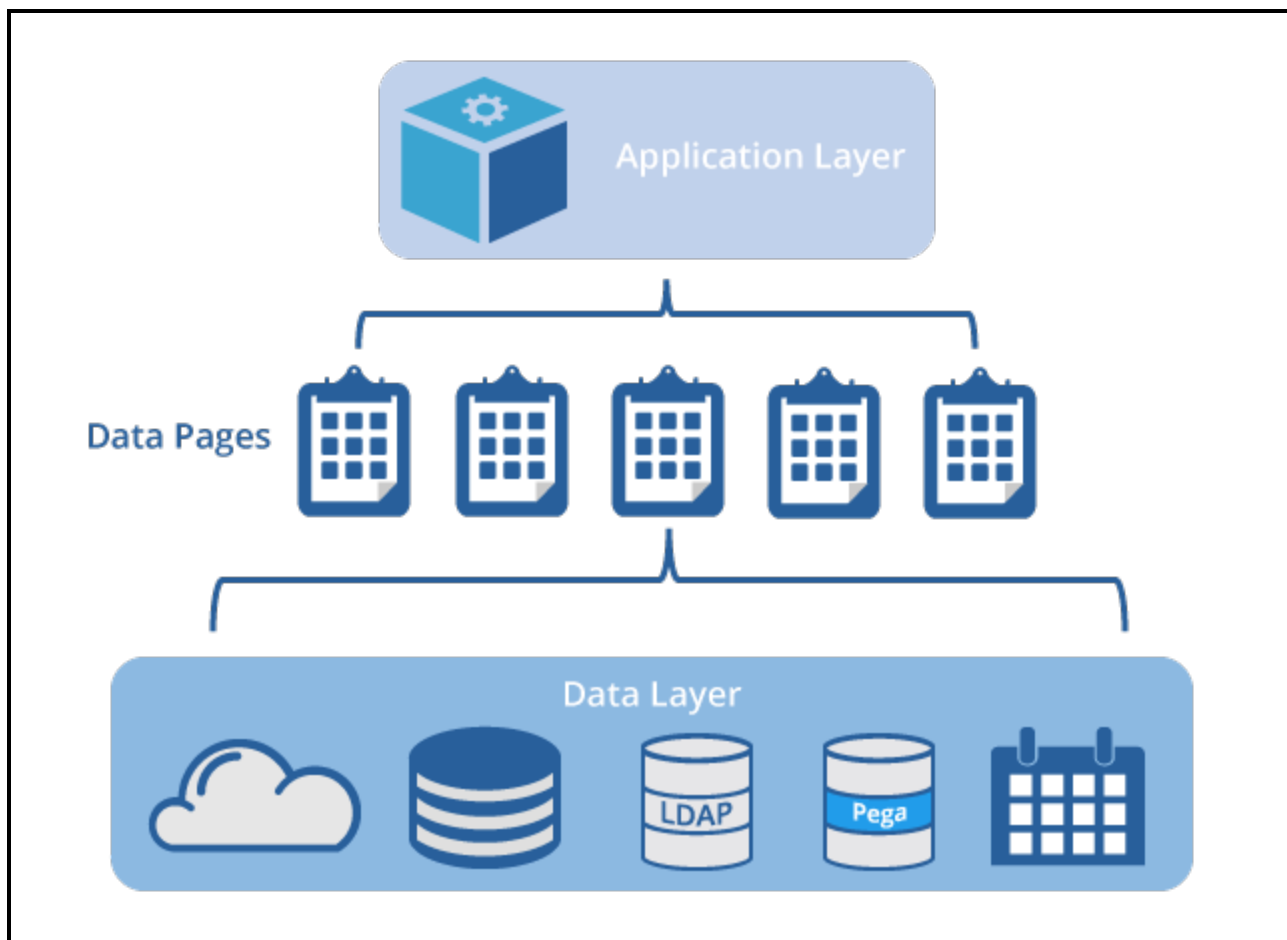
After this lesson, you should be able to:

- Identify the role of data pages in managing data
- Explain how data pages abstract data from the source
- Explain the life cycle of a data page
- Explain the affects of the scope setting of a data page
- List the sourcing options for a data page
- List the options for refreshing a data page
- Configure a data page

Data pages

When you create and process a case, you need data. Some data is collected from the user as part of the case process, while other data is retrieved from the application or from external systems. For example, if you want to see the claim history for a customer in a claims application, you retrieve application data. If you need to display customer data held in a system of record, you retrieve data from an external system.

You use a **data page** to retrieve data for your application, regardless of the source. Data pages cache data on demand to a clipboard page and have a **scope**. The **scope** defines who can access the data page. You can make the data retrieved accessible for all applications, or limited to the logged in user or specific case only.



Every data page defines a **refresh strategy**. The refresh strategy defines when the data page is considered stale and needs to be reloaded. Data pages are created and updated on demand. Even if a data page is considered stale, the page is never reloaded until it is referenced. The available refresh strategy options depend on the scope of the page.

Data pages can use a variety of **sources** to load data. A data page provides an abstraction between the application and data layers. This means that you can use the data page in your application without knowing the data source. Your application configuration does not need to change if the source does. For example, you may configure a data page to look up customer data from a database table. If the

interface to the customer data changes to a REST web service, you only need to change the data page source and not the application code.

KNOWLEDGE CHECK



ANSWER

When do you use a data page?

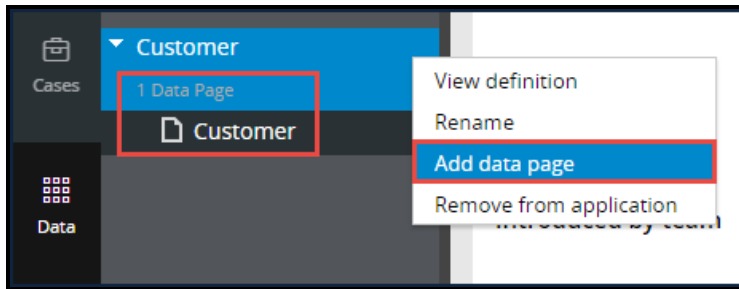
You use a data page when you need to retrieve data for your application.

How to configure a data page

Before you create a data page, remember to answer the following questions:

- What is the structure of the data page?
- What is the scope of the data page?
- How is the data page sourced?
- When does the data become stale?

You create a data page for a data type from the Data Explorer. The Data Explorer also shows data pages already defined for the data type.



A **data page rule** defines the structure, scope, source, and refresh strategy of the cached data.

Structure

First, you need to specify the structure of your data page. The structure is either page or list. Choose page if you want to load a single record (such as a single customer). Choose list if you want to load multiple records (such as a list of insurance claims filed by a customer). The Object type defines the class of the page or pages in the list.

A data page is typically read-only, but you can set the Edit mode to editable if you need to update the data page after it has been loaded. Updates to the data page are only local and not propagated back to the source.

Scope

Next, you need to decide on the scope. You have three options for the scope: thread, requestor, and node.

- The **thread** level scope is useful when the data page is context-sensitive to a particular case. For example, if a customer service representative (CSR) is working on several cases for different customers, the data page must be defined as thread scope because the customer data should be limited to a specific case. Setting the scope to thread ensures that the CSR sees only the data relevant to an individual customer's case. If the data can be shared across cases, then a broader scope, such as requestor or node, should be chosen.
- The **requestor** scope lets you share data pages for a given user session and is often used when the data page contains data associated with the logged-in operator. The work list or local weather information are both examples of data associated with the logged-in operator.

- The **node** option makes a single data page instance accessible by all users of the application and other applications running on a given node. On a multinode system, each Java Virtual Machine instance has one copy of the node level data page. Node level pages reduce the memory footprint by storing only a single copy of the data available to all users. Node level pages are an excellent option for storing common reference data such as currency exchange rates.

Data sources

Then, you need to configure the data sources. For list structures, the sourcing options include, connector, data transform, report definition, or a load activity. For page structures, the lookup data source replaces report definition as an option.

Source	Description
Connector	Use a connector to obtain data from an external data source as specified by the connector type.
Data transform	Use the data transform option to populate a data page using a data transform.
Report definition	Use a report definition to return a list of data objects mapped in the application.
Look-up	Use the look-up to return a specific data object mapped in the application.
Load activity	The activity can be used for special situations where none of the other options are suitable.

The **Request Data Transform** and **Response Data Transform** allow you to map the outgoing and incoming data to the application data structure.

Definition Load Management Parameters Pages & Classes Test Cases Usage History

Data page definition

Structure
 Page

Object type *
 MyCo-Data-Customer

Edit mode
 Read Only

Scope
 Thread

Data sources

System name

1 Source * Type Name * Request Data Transform Response Data Transform *

Connector SOAP GetCustomerData GetCustomerDataRequest Parameters GetCustomerDataResponse Parameters

MyCo-Int-Customer- Parameters Parameters

Create Data Source

Add New Source

Refresh strategy

You configure the **Refresh strategy** on the **Load Management** tab. The available reload options depend on the scope of the page. For requestor or thread pages, you can reload the data for each interaction or by evaluating a when rule. The data page can also be marked for refresh based on an elapsed time interval calculated from the last load time. If you combine the **Do not reload when** and **Reload if older than** options, the data page refreshes as soon as either condition is met.

Definition **Load Management** Parameters Pages & Classes

Page management

Clear Data Page

Refresh strategy

Reload once per interaction

Do not reload when

+

Reload if older than

Days : Hours : Minutes : Seconds

: : :

The refresh strategy for a node level page reloads per interaction and does not reload when options are not available. Notice the addition of the Load Authorization section prompting you for an Access Group.

Definition **Load Management** Parameters Pages & Classes

Page management

Clear Data Page

Load authorization

Access Group*

+

Refresh strategy

Reload if older than

Days : Hours : Minutes : Seconds

: : :

Node level data pages are not executed in the context of a logged-in operator because the data pages are available for all applications on the node. Instead, an access group is specified to provide the requestor context used by the system when loading the node level data page.

Data pages load to memory on demand. A data page remains, on the clipboard, in memory to serve requests without reloading the data. If a data page is configured to reload if older than one hour, the

page is marked for reload after one hour. It is not reloaded until the next request for the page occurs following the configured time.

KNOWLEDGE CHECK



When you configure a data page, you define the structure as either a page or a list. What other components do you need to define?

A data page rule defines the structure, scope, data source, and refresh strategy of the cached data.

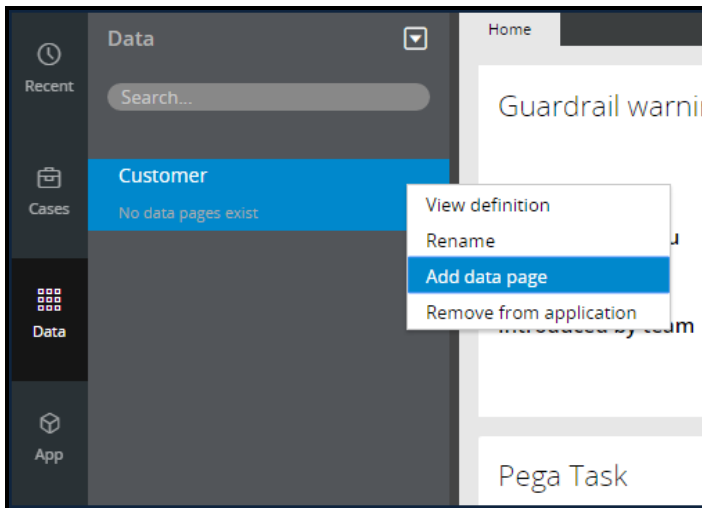
Configuring a data page

You create and configure a data page by creating the data page, configuring the data page definition, and configuring the data source. The configuration varies depending on the requirements for the application. In this example, you configure a data page that holds customer data for a claim case handled by a customer service representative. The customer data is retrieved from a SOAP connector.

Create data page

Follow these steps to create a new data page for the data type:

1. In the Data Explorer, locate the data type for which you want to create a data page.
2. Right-click the data type to select **Add data page**. In this example, you add a data page for Customer data.



3. Enter a Label that is descriptive of the data. The system populates the Identifier based on the Label. You can leave the default Identifier or change this value.

4. Select a **Context**. The system uses default values in the **Apply to** and **Add to ruleset** fields.
5. Click **Create and open**. The Edit Data Page screen, where you configure the data page definition, is displayed.

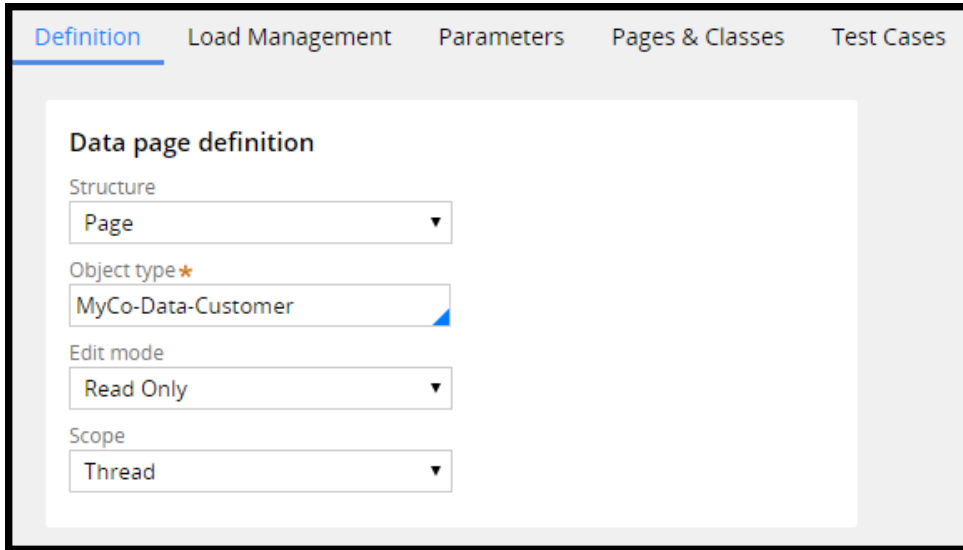
Configure data page definition

The data page holds the data for a specific customer record.

Follow these steps to configure the data page definition:

1. In the **Structure** drop-down, select **Page** because, in this example, you want to retrieve data for a single customer.
2. In the **Object Type** field, enter the class of the page or pages in the list.
3. Leave the **Edit mode** set to Read Only because the data page will not be manipulated.

4. Set the **Scope** to Thread because the customer data is specific to a case.



Configure data source

The data in this example is retrieved from a SOAP service.

Follow these steps to configure the data source for the data page:

1. Provide a **System name** for your data source.
2. In the **Source** drop-down, select **Connector**.
3. In the **Type** drop-down, select **SOAP**.
4. Specify the **Name** of the connector — GetCustomer, in this case.
5. Optionally, specify a **Request Data Transform** that maps application data to the request.
6. Specify a **Response Data Transform** to map the data returned by the connector to the application data structure.

Data sources

Simulate data source

System name
Customer DB

Source *
Connector

1 Type SOAP Name * GetCustomer

Request Data Transform
GetCustomer_Request
Parameters

Response Data Transform *
GetCustomer_Response
Parameters

Endpoint URL

Create Data Source

Add New Source

7. Click **Create data source**. The data page is created in the Customer data type. When you refresh the Data Explorer, the Customer data type will show the added data page.

Managing reference data

Introduction to managing reference data

Applications often require access to reference data. Reference data is used in the case processing, but is not directly part of the application. Reference data is often used in the user interface to provide options for the user. For example, a drop-down list for a store can contain a list of store branches for the user to select from.

Reference data is sometimes retrieved from external systems using connectors. However, in some cases, the data needs to be stored and distributed as part of the application.

After this lesson, you should be able to:

- Explain the benefits of incorporating reference data into an application
- Explain how local data storage manages reference data in an application
- Incorporate reference data in an application with local data storage
- Configure database access using the External Database Class Mapping Wizard

How to use reference data

Every application collects data. Sometimes the values for an input field are limited to a set of values. For example, a customer can typically select standard, express, or next-day shipping options. However, for a specific product the shipping option might be limited to standard. Reference data defines permissible values for data fields. Limiting the input values to valid options reduces errors and allows for automation. Reference data gains in value when it is widely reused and referenced.

Reference data should be distinguished from master data. Master data represents key business entities, such as customers. Master data contains all the necessary detail — for example, an identifier, name, address, and date of account creation for a customer. Reference data consists of a list of permissible options with relevant metadata.

A change to the reference data values may need an associated change in the business process to support the change. A change in master data is always as part of existing business processes. For example, adding a new customer is part of the standard business process. Adding a new customer level — for example, platinum — results in a modification to the business processes to manage platinum customers.

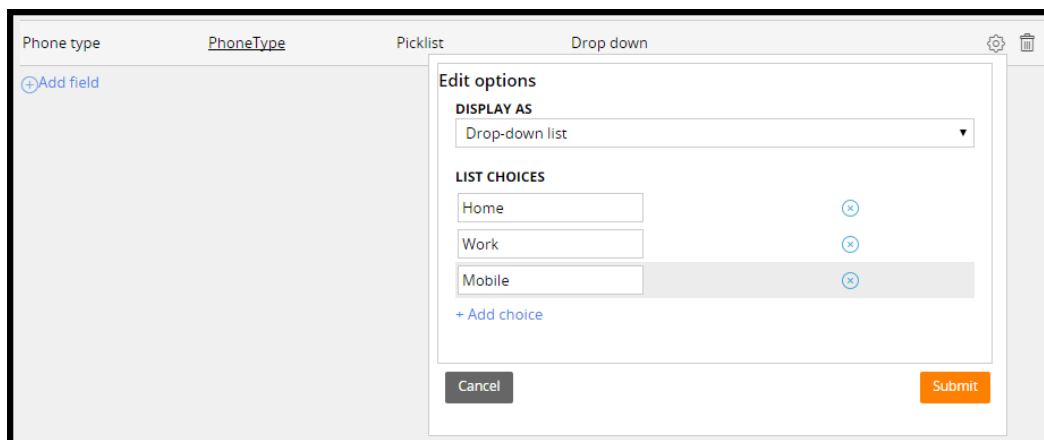
There are several ways you can use reference data in a Pega application:

- Configure a **static list** for a limited set of values
- Configure **local data storage** to store data records for a data type, without having to create or maintain database tables
- Connect to an **external database** to retrieve data

Configure a static list

A static list is a simple list of values for a property used in your application. For example, the values for the shipping property could be standard, express, or next-day shipping. Use a static list to add options to a user interface property when you have a limited set of values and those values do not change frequently.

For example, to capture a phone number, you can specify a list of types such as home, work, and mobile. To do this, you add a static list to the Phone type property.



The screenshot shows the configuration interface for a 'Phone type' property. The 'Edit options' dialog is open, showing the following settings:

- DISPLAY AS:** Drop-down list
- LIST CHOICES:** Home, Work, Mobile

Buttons for '+ Add choice', 'Cancel', and 'Submit' are visible at the bottom of the dialog.

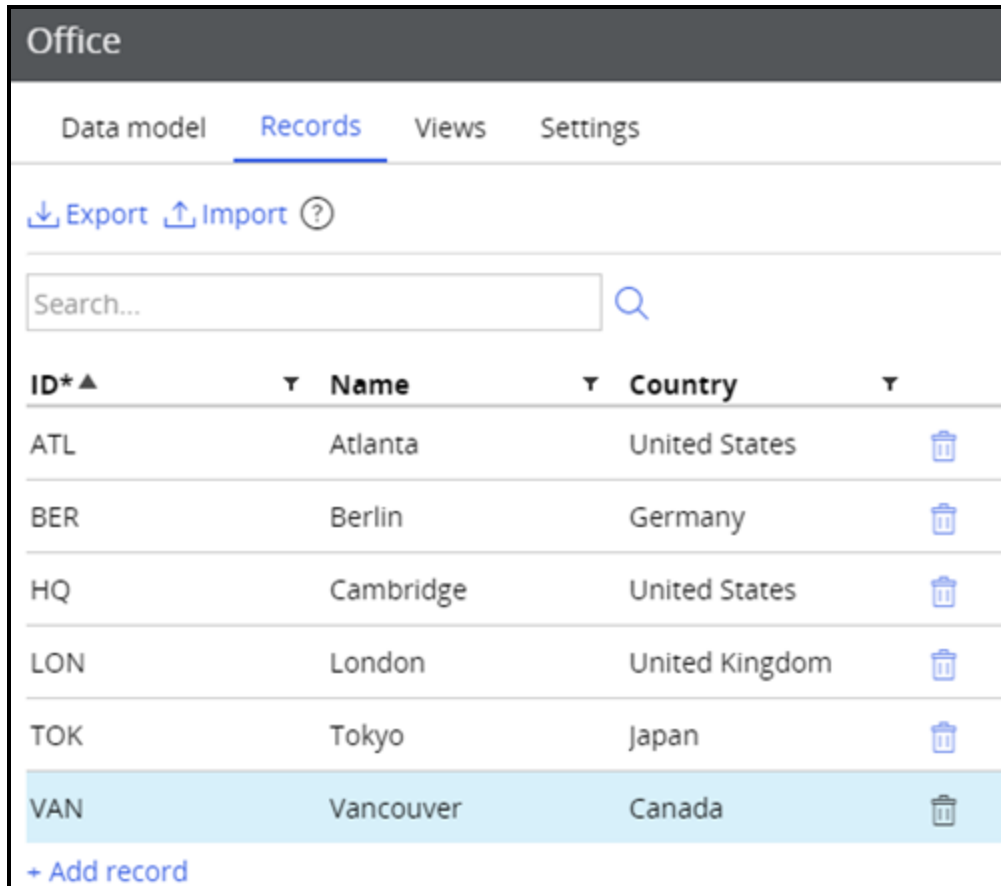
For more information, see the help topic [Configuring a picklist](#).

Configure local data storage

A local data storage lets you store data records for a data type, without having to create or maintain database tables. Reference data stored in local data storage can be packaged and distributed as part of an application.

You can create local data storage for any data type in Pega Express or Designer Studio. In the selected data type, you add the properties and specify a property to serve as the unique key for the record. After you create the local data source, you can add records to the data source.

For example, in Pega Express, in the Office data type, in the Records tab, you can add a record for each office location.



The screenshot displays the 'Office' data type interface in Pega Express, specifically the 'Records' tab. The interface includes a search bar, 'Export' and 'Import' buttons, and a table of records. The table has columns for ID, Name, and Country. The 'VAN' record is highlighted in blue. Below the table is a '+ Add record' button.

ID*▲	Name	Country	
ATL	Atlanta	United States	🗑️
BER	Berlin	Germany	🗑️
HQ	Cambridge	United States	🗑️
LON	London	United Kingdom	🗑️
TOK	Tokyo	Japan	🗑️
VAN	Vancouver	Canada	🗑️

+ Add record

Connect to an external database

You can retrieve data from an external database storage using connectors. For example, a purchase application needs access to currency exchange rates. Exchange rate information is stored in a table in an external database.

You have two options when integrating with an external database in Pega Platform:

- Database Table Class Mapping tool
- SQL connector

The Database Table Class Mapping tool provides a wizard to generate all the artifacts needed to interact with reference data in an external database. These artifacts include a data class, a database table instance, and a link between those two artifacts.

The data mapping creates a pass-through from your application to the table in the external database. When enabled, you can access data in the external database as if the data were within your application. Using the Database Table Class Mapping tools is preferred when integrating with external databases.

The SQL connector requires you to write SQL queries to interact with the data in the external database. Use a SQL connector when you need to perform advanced queries (such as advanced joins), or when you need to use vendor-specific SQL syntax. Setting up a SQL connector is a Senior System Architect or a Lead System Architect task. In this course, you use the Database Table Class Mapping tool to set up your integration.

KNOWLEDGE CHECK



You want to create a list of products and their attributes to distribute with an application. Which method do you use to create the reference data?

Configure a local data storage and add your data records to the selected data type.

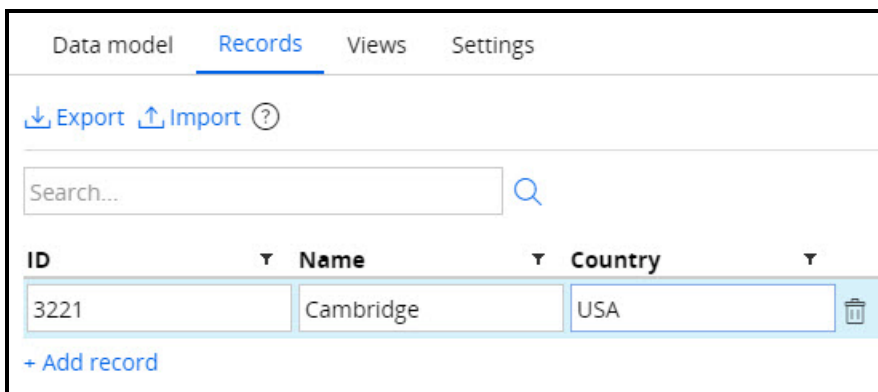
Defining reference data for a data type

When you create a new data type in Pega Express, you are given the option to use Pega local storage to store data.

Manage records in the local data source

Follow these steps to add a record to local data storage:

1. In either Pega Express or Designer Studio, click **Data** to access the Data Explorer and select the data type you want to add records to.
2. Select the **Records** tab to add data to the data type.
3. Click **+ Add record** to add a new row of data.



4. Enter the data for the properties. The data persists when you leave the field.
5. Click the **Trash Can** icon at the end of a row to remove a record.

Access the data in local storage

Follow these steps to access the data in local data storage:

1. Create a property in a case that uses your data type that uses local storage.



2. Use the property in a view to display the value. The property uses a data page to access the data

from local storage.

Label	Type	Options	
Office Location	Field group ▼	Auto ▼	Office Locations ▼
	View	Create new view... ▼	Open

[+ Add field](#)

Using the external database mapping wizard

Sometimes your application requires access to reference data in an external database. For example, a purchase application needs access to currency exchange rates. Exchange rate information is stored in a table in an external database.

The Database Table Class Mapping wizard generates all the artifacts needed to interact with reference data in an external database. These artifacts include a data class, a database table instance, and a link between those two artifacts.

The data mapping creates a pass-through from your application to the table in the external database. When enabled, you can access data in the external database as if the data were within your application. Using the Database Table Class Mapping wizard is preferred when integrating with external databases.

Take a look at how you can use the Database Table Class Mapping wizard. The wizard creates a data class with the data mapping and a database table instance that references the external table.

Before you begin

Pega Platform requires a database record for each database that your application connects to. When the database instance has been created and a suitable JDBC library has been installed on the server, you can connect to and interact with that database.

Run the tool

The Database Class Mappings landing page (**Designer Studio > Data Model > Classes & Properties > Database Class Mappings**) shows you all of the existing database table mappings.

1. Click **New External Database Table Class Mapping** to launch the Database Table Class Mapping wizard.
2. Choose the database, schema, and the name of the table you want to create the mapping for.
3. Enter a ruleset name and version.
4. Enter the name of the data class you want to create. You cannot use an existing class because the Database Table Class Mapping wizard generates a new class.
5. Select the columns you want to access and specify the name of the properties you want to map to in the data class.
6. Click **Submit**.

Database Table Class Mapping ✕

Step 1: Specify database table

Database Name* +

Schema Name

Table Name*

Step 2: Specify ruleset class

Ruleset Name*

Ruleset Version*

Class Name* +

Step 3: Map database table columns to properties in the ruleset class

Key	Column Name	Data Type	Property Name	Property Type	Map All/None <input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	to_currency	varchar	<input type="text" value="ToCurrency"/>	Text	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	from_currency	varchar	<input type="text" value="FromCurrency"/>	Text	<input checked="" type="checkbox"/>
<input type="checkbox"/>	conversion_rate	float8	<input type="text" value="ConversionRate"/>	Double	<input checked="" type="checkbox"/>

The Database Table Class Mapping wizard creates a class and properties. The External Mapping tab on the class record contains the mapping details.

You can use this class with a data page to make the currency exchange rates available to your application. In the data page, you can use the lookup feature to fetch an instance or use a report definition to fetch a list of instances. You can also create additional data pages as needed to filter the data. Alternatively, you can use the Obj-methods to open, save, and remove rows from the table.

AUTOMATING BUSINESS POLICIES

Configuring a service level agreement

Introduction to configuring a service level agreement

End users complete assignments and resolve cases to achieve performance milestones such as goals and deadlines. These milestones are service level agreements (SLAs).

In this lesson, you will learn how to configure service level agreements for goals, deadlines, notifications, and escalation actions.

After this lesson, you should be able to:

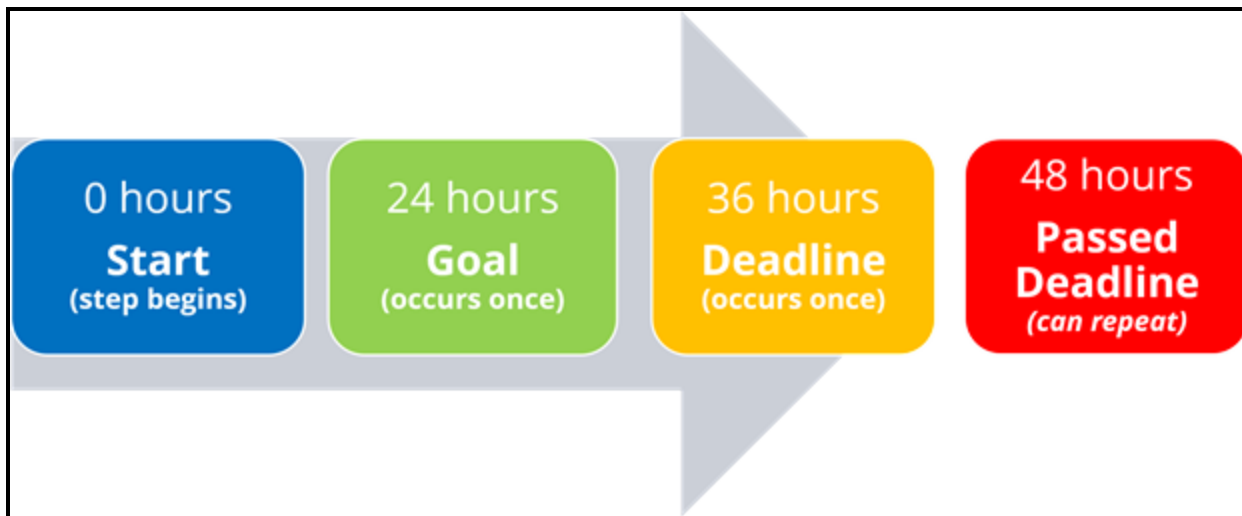
- Explain the purpose of service level agreements in Pega applications
- Identify where in a case life cycle a service level agreements can be applied
- Describe each interval of a service level agreement
- Explain how a service level agreement can trigger an escalation action
- Explain how service level agreements increase assignment urgency
- Configure a service level agreement for an assignment

Service level rules in Pega Platform

Organizations often establish service level agreements to enforce on-time performance. These obligations range from informal promises of response times to negotiated contracts. A **service level agreement (SLA)** establishes a deadline for work completion. When you establish a goal and deadline in Pega Express or the Case Designer in Designer Studio, Pega creates a **service level agreement rule** for you.

An SLA contains three time interval milestones.

- A **goal** milestone defines how long the assignment should take. It is typically measured from when the step or case begins.
- A **deadline** milestone defines the amount of time the step or case may take before it is late. The deadline is also measured from when the step or case begins.
- A **passed deadline** milestone defines when to take further action because the step or case is past the deadline. The passed deadline interval measures the time that has passed since the deadline for a still-open assignment.



For example, a company requires that employees submit a record of hours worked within two days of the end of the work week. This record of hours worked, called a time sheet, allows the Payroll department to credit the employee for hours worked during the week. Until an employee submits their time sheet, the Payroll department cannot pay the employee. The company reminds employees to submit their time sheet, even after the deadline has passed in order to pay the employees.

Unlike the goal and deadline intervals, you can configure the passed deadline interval to repeat a fixed number of times, or repeat indefinitely until the user completes the assignment. You can continue to increase the assignment urgency and remind a user of a late assignment. Goal and deadlines intervals do not repeat.

KNOWLEDGE CHECK



How does the passed deadline interval differ from the goal and deadline intervals?

Goal and deadline intervals do not repeat. You can configure the passed deadline interval to repeat a fixed number of times, or repeat indefinitely until users complete the assignment.

You define an **urgency** between 10 and 100 for each milestone. The higher the value, the higher the urgency. Typically, the urgency increases as an assignment advances to the next milestone.

The following video describes how SLAs work in Pega.

Service level agreements for processes, stages, and case types often require only goal and deadline intervals. You configure the goal and deadline intervals in Pega Express. Pega Platform automatically creates the service level agreement rule based upon the goals and deadline intervals you configure.

Service level agreements for assignments are often more complex. A service level agreement for an assignment may dictate actions to perform after a deadline passes. For example, a company establishes a deadline to respond to a customer inquiry in 48 hours. The company notifies a customer service manager every 24 hours until a representative responds to the customer for any inquiry open after 48 hours.

A service level agreement for an assignment may also affect the start of the goal and deadline intervals. For example, a stock brokerage establishes a deadline of two hours to price assets in customer accounts. Associates cannot begin pricing assets until after the stock market closes for the day. In this case, the start time for the service level agreement occurs after the stock market closes for the day. If the stock market closes at 4:30 P.M., the deadline is 6:30 P.M., even if the case reaches the assignment at 1 P.M..

You configure complex performance obligations using service level agreement rules in Designer Studio.

KNOWLEDGE CHECK



ANSWER

What capabilities do you gain by configuring a service level agreement rule in Designer Studio?

You can add behavior for late assignments, and determine when an assignment is ready for users to perform.

Adding a service level to a case

You can add service levels to entire cases and to steps within a case using Pega Express and Designer Studio. The procedure for adding service levels to cases and steps is the same in Pega Express and Designer Studio. The difference is you can access the service level rule in Designer Studio and change the default timer start.

Adding a service level to a case

To create a service level for an entire case:

1. In a case type, click **Settings**.
2. Click **Goal & deadline** to configure a service level.
3. Select **Consider goal and deadline** to display the Goal and Deadline entry fields.

Note: The default timer start is the start of the current case. You can change the timer start setting in Designer Studio by selecting the appropriate option under **Start timer from the start of**.

4. Under Goal, in the **Days** field, enter the goal in number of days.
5. Optionally, under Goal, in the **HH:MM:SS** field, enter the time in HH:MM:SS format, if you require a specific goal resolution time,
6. Under Goal, enter the amount in the **Increase urgency by** field to increase urgency.
7. Under Deadline, in the **Days** field, enter the deadline in number of days.
8. Optional. Under Deadline, in the **HH:MM:SS** field, enter the time in HH:MM:SS format, if you require a specific deadline resolution time.
9. Under Deadline, in the **Increase urgency by** field, enter the amount by which to increase urgency.
10. Click **Save** to save your work.

Adding a service level to a step

To add a service level to a step:

1. In a case type, select the step requiring a service level.
2. In the contextual properties pane, click **Goal & deadline** to configure a service level.
3. Under Goal & deadline, select **Consider goal and deadline** to display the Goal and Deadline entry fields.

Note: To access the Goal and Deadline entry fields in Designer Studio, under Service level agreement, select **Custom**.

4. Under Goal, in the **Days** field, enter the goal in number of days.
5. Optionally, under Goal, in the **HH:MM:SS** field, enter the time in HH:MM:SS format, if you require a specific goal resolution time,

6. Under Goal, in the **Increase urgency by** field, enter the amount by which you want to increase urgency.
7. Under Perform actions, select the action for the case to perform, such as notifying the assignee or manager.
8. Optional. Click the **plus sign** to the left of Add escalation action and select the additional action, if an additional action is desired.
9. Under Deadline, in the **Days** field, enter the deadline in number of days.
10. Optionally, under Deadline, in the **HH:MM:SS** field, enter the time in HH:MM:SS format, if you require a specific deadline resolution time.
11. Under Deadline, in the **Increase urgency by** field, enter the amount by which to increase the urgency.
12. Under Perform actions, select the action for the case to perform.
13. Optionally click the **plus sign** to the left of Add escalation action and select the additional action, if an additional action is desired.
14. Click **Save** to save your work.

Assignment urgency

In Pega applications, assignment urgency indicates assignment priority.

Pega applications use the **GetNextWork** selection algorithm to identify the next user assignments. GetNextWork takes into account assignment urgency (priority) when selecting the next assignment for a user. Between two or more assignments that users can perform, GetNextWork favors the assignment with the highest (greatest) urgency.

For example, a bank representative working in the fraud unit is ready for another assignment. Pega assigns the representative the fraud task with the highest priority.

Pega caps urgency adjustments at 100. At an urgency value of 100, Pega ignores further urgency adjustments and continues other actions. For example, If the deadline passes, Pega notifies the assigned user. If the assignment urgency is 100 at deadline, Pega ignores the urgency increment and still sends the notification.

You can configure a value to enable a user to increase the urgency of an assignment by running a local action. For example, a bank representative runs a local action to increase the assignment urgency when a customer reports a stolen credit card while on vacation.

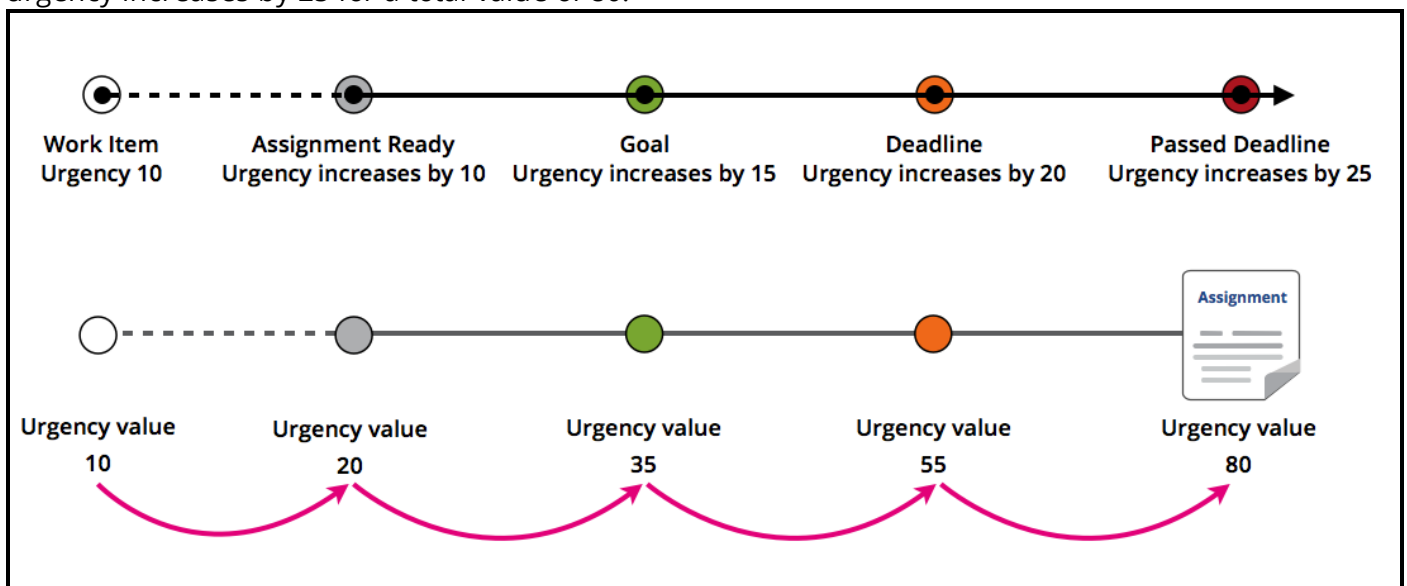
Assignment urgency factors

In Pega, the assignment urgency is recorded using the property *.pxUrgencyAssign*. Pega calculates *.pxUrgencyAssign*, as a sum of three input properties: *.pxUrgencyWork*, *.pxUrgencyAssignSLA*, and *.pyUrgencyAssignAdjust*.

Property name	Definition	Example
.pxUrgencyWork	Default urgency for the case type with a default value of 10	You change the value of <i>.pxUrgencyWork</i> to indicate that assignments for a specific type of case are more important than other cases. For example, if transaction dispute cases are a higher priority than other types of cases, you set the value of <i>.pxUrgencyWork</i> to 20. Assignments for transaction dispute cases then default to a greater urgency than assignments for other types of cases.
.pxUrgencyAssignSLA	Urgency calculated from the service level rule. The value is the sum of the initial urgency and the urgency increments for the goal, deadline, and passed deadline intervals. As an	For an Account review assignment, a service level agreement may establish an initial urgency of 10 and further increments of 15 for the goal, 20 for the deadline, and 25 for the passed deadline. When the user receives the Account review assignment, <i>.pxUrgencyAssignSLA</i> is

Property name	Definition	Example
	assignment ages in a workbasket or worklist, Pega increases the value of <i>.pxUrgencyAssignSLA</i> according to the configuration of the service level agreement.	set to 10, which increases <i>.pxUrgencyAssign</i> to 20. When each interval is exceeded, Pega increases <i>.pxUrgencyAssignSLA</i> as directed by the service level agreement, which further increases <i>.pxUrgencyAssign</i> .
.pyUrgencyAssignAdjust	Manual adjustment for the assignment urgency. This value enables a user to increase the urgency of an assignment by running a local action.	A customer service representative (CSR) runs a local action to increase the urgency of an assignment if a customer reports that their credit card was stolen while on vacation. The CSR runs the local action, which increases the value of <i>.pyUrgencyAssignAdjust</i> to 50. This increases the overall assignment urgency, <i>.pxUrgencyAssign</i> , by 50 to increase the likelihood that the assignment is completed before other assignments.

The following image illustrates how the urgency value increases as an assignment progresses toward the goal and deadline. The top row shows the urgency values at each stage — the creation of the work item — when the assignment is ready, the assignment goal date, the assignment deadline date and the passed deadline date. The bottom row shows the urgency values at each stage. At work item creation, the urgency value is 10. When the assignment is ready, the urgency value increases by 10 for a total of 20. At the goal date, the urgency increases by 15 for an urgency value of 35. At deadline, the urgency increases by 20 for a new urgency value of 55. Finally, if the assignment state is passed deadline, the urgency increases by 25 for a total value of 80.



KNOWLEDGE CHECK



What are the three factors that influence assignment urgency value?

The three factors influencing assignment urgency value are the default urgency for the case type, the urgency calculated from the service level rule, and manual adjustments (if enabled).

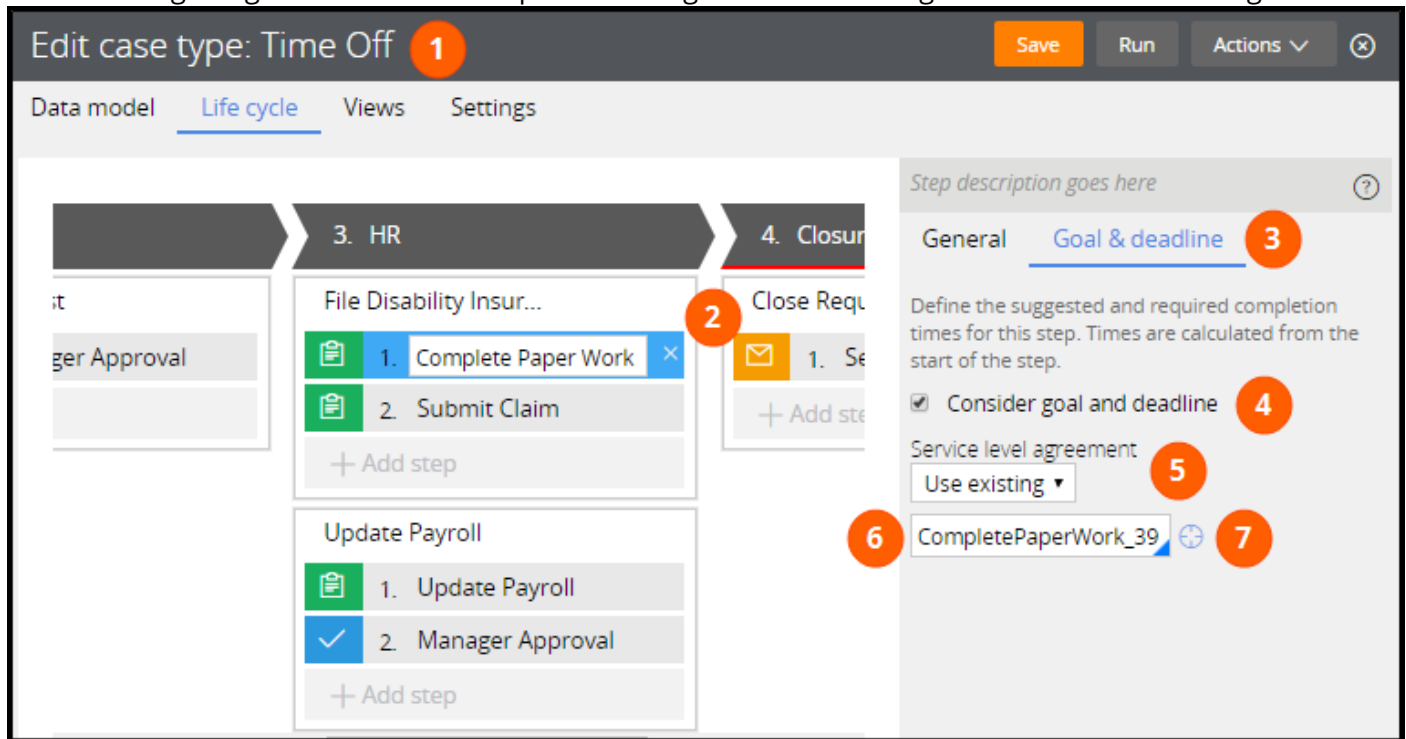
Configuring a service level agreement rule

You add a service level rule to an assignment when you require a passed deadline interval.

Add a service level agreement to an assignment

Create a custom service level agreement for assignment.

The following image illustrates the steps for adding a service level agreement rule to an assignment.



Follow these steps to add a service level agreement rule to an assignment:

1. Open the case type in Designer Studio.
2. Select the assignment to which to apply the service level.
3. In the properties panel for the step, click **Goal & deadline**.
4. Select the **Consider goal and deadline** check box.
5. From the Service level agreement list, select **Use Existing**.

Note: Selecting **Custom** from the Service level agreement list displays the **Goal** and **Deadline** entry fields. Pega uses the values in the Goal and Deadline fields to create a service level agreement rule. If you need to add a passed deadline interval, select **Use Existing**. You can create your custom rule from the options that display for **Use Existing**.

6. In the empty field under the Service level agreement list, enter the name of the service level agreement rule to apply to the assignment.
7. To the right of the field containing the name of the service level agreement rule, click the **crosshair** icon.

- If the rule is new, the New Record form opens. Click **Create and open** to create the service level agreement rule.

Configure the starting behavior for the service level

Specify an initial urgency for the service level agreement and any delay before tracking performance against the goal and deadline intervals.

The following image illustrates the steps for configuring the starting behavior for a service level agreement rule.

The screenshot shows a form titled "Start of service level" with three numbered steps:

- Initial Urgency**: A text input field containing the value "0".
- Assignment Ready**: A dropdown menu with "Immediately" selected.
- Calculate service levels**: A dropdown menu with "Interval from when assignment is ready" selected.

Below this section is a section titled "Service level definitions" with a dropdown menu for "Calculate service levels" also set to "Interval from when assignment is ready".

Follow these steps to configure the starting behavior for the service level agreement rule:

- On the service level agreement rule form, in the **Initial Urgency** field, enter an initial urgency for the service level. The assignment urgency increments by the entered value when the assignment is ready for the user to perform.
- From the **Assignment Ready** list, select when the assignment is considered available for a user to perform.

Option	Description	Usage
Immediately	Sends the assignment to a worklist or workbasket as soon as the case reaches the assignment. This is the default option.	Users can perform the assignment immediately.
Dynamically defined on a property	Delays sending the assignment to a worklist or workbasket until the specified delay interval elapses. Use the Get Date Time From field to specify a property that represents the optimal start time.	Delay the assignment until a specified time.
Timed delay	Delays sending the assignment to a worklist or workbasket until the specified delay interval elapses. Use the Days, Hours, and Minutes fields to enter the duration of the delay.	Delay the assignment for a specified amount of time.

- From the **Calculate service levels** list, select whether to track the service level intervals against a fixed interval or against the value of a property. Use a property reference to adjust the intervals for each case. Use fixed intervals to ensure that the intervals are the same length of time for each case.

Configure the goal and deadline intervals

Add a goal and deadline to measure whether the assignment is performed according to schedule.

The following image illustrates the steps for configuring the behavior for goal and deadline intervals.

The screenshot shows the configuration interface for goal and deadline intervals. It is divided into two main sections: 'Goal' and 'Actions/When'.
In the 'Goal' section, there are four input fields for 'Days', 'Hrs', 'Mins', and 'Secs' with values 2, 0, 0, and 0 respectively. A red circle with the number 1 is placed over these fields. Below these fields is a text label: 'Time interval starts when the associated assignment (or work item) is created'. There is a checkbox labeled 'Only calculate using business days' with a red circle and the number 2 next to it. To the right of the 'Goal' section is an 'Amount to increase urgency' field with the value 10 and a red circle with the number 3 next to it.
The 'Actions/When' section is on the right. It has a header with 'Actions' and 'When'. Below the header is a 'Select Action...' button with a red circle and the number 4 next to it. Below this is a 'Perform Action' dropdown menu with 'Select action ...' and a 'When' field with a dropdown menu and a blue gear icon.

Follow these steps to configure the behavior for the goal and deadline intervals:

- Enter a time for the interval. Depending on the selection in the **Calculate service levels** list, either specify the interval using the four fields labeled **Days**, **Hrs**, **Mins**, and **Secs**, or reference a property that represents the length of the interval.
- Optional: click the **Only calculate using business days** check box to only measure elapsed time for the interval in business days. Enabling this option prevents Pega from counting non-work days, such as weekends, against the interval limit. Pega determines non-work days from information in the user's operator ID record.
- In the **Amount to increase urgency** field, enter an urgency adjustment for the interval. The assignment urgency increases by the specified amount until reaching 100.
- Optional: click **Select Action...** to add an escalation action for the interval.

From the **Perform Action** list, select an escalation action to perform when the interval ends. If necessary, use the **When** field to use a when condition or when rule to determine whether to perform the escalation action.

Configure the passed deadline interval

Add a passed deadline interval to describe behavior for assignments that are considered late.

The following image illustrates the steps for configuring the behavior for passed deadline interval.

The screenshot shows the 'Passed deadline' configuration interface. It includes the following elements:

- Limit passed deadline events to:** A text input field containing the number '3', marked with a red circle '1'.
- Days, Hrs, Mins, Secs:** Four spinner controls with values 0, 12, 0, and 0, marked with a red circle '2'.
- Time interval starts when the deadline is reached:** A text label.
- Only calculate using business days:** An unchecked checkbox, marked with a red circle '3'.
- Amount to increase urgency:** A text input field containing the number '10', marked with a red circle '4'.
- Actions and When:** A table with two columns: 'Actions' and 'When'. The first row has a 'Select Action...' button (marked with a red circle '5') and an 'Edit' button. Below this, there is a 'Perform Action' dropdown menu and a 'When' field.

Follow these steps to configure the behavior for the passed deadline interval:

1. In the **Limit passed deadline events to** field, enter the number of passed deadline events to apply to the assignment. To apply the passed deadline behavior indefinitely until the assignment is completed, leave the field empty.
2. Enter a time for the interval. Depending on the selection in the **Calculate service levels** list, either specify the interval using the four fields labeled **Days**, **Hrs**, **Mins**, and **Secs**, or reference a property that represents the length of the interval.
3. Optional: click the **Only calculate using business days** check box to only measure elapsed time for the interval in business days. Enabling this option prevents Pega from counting non-work days, such as weekends, against the interval limit.
4. In the **Amount to increase urgency** field, enter an urgency adjustment for the interval. The assignment urgency increases by this value each time a passed deadline cycle completes, until the assignment urgency reaches 100. Once the assignment urgency reaches 100, further urgency adjustments are ignored, though escalation actions are processed. The Passed Deadline interval repeats if configured to do so.
5. Optional: click **Select Action...** to add an escalation action for the interval.
From the **Perform Action** list, select an escalation action to perform when the interval ends. If necessary use the **When** field to use a when condition or when rule to determine whether to perform the escalation action.
6. Click **Save** to save your configuration.

Configuring and sending correspondence

Introduction to configuring and sending correspondence

During a business process, organizations often need to communicate with parties associated with a case. Pega allows you to automate and create timely and clear communication with participants in a case. This communication ranges from simple notifications of assigned tasks to complex communications that contain case-specific data and calls-to-action. Adding correspondence to a business process keeps customers and case workers engaged throughout the case life cycle.

In Pega, you configure emails, letters, fax, and text messages with correspondence rules. You can configure your application so that the system can send correspondence automatically or enable users to send correspondence manually. Correspondence rules allow you to add case data to provide richer, more relevant communication.

After this lesson, you should be able to:

- Explain how correspondence improves a process
- Describe the process for creating correspondence
- Add correspondence to a case type
- Send correspondence while processing a case
- Incorporate case content into correspondence

Automating case communications

Common reasons for communicating with users

Organizations depend on timely communication to establish a shared understanding of transactions or assignments.

For example, consider a requirement for an auto claims application in which customers must be notified when their claims are successfully filed, or anytime the status of the claims changes.

Another common notification requirement is keeping case workers up-to-date. For example, you must notify case workers when they have a new claim to process. Also, you may want to notify them on the progress of the previous claim.

Finally, you may have a requirement to communicate with someone who is indirectly involved in the case, such as an external agency.

To achieve effective communication, answer three simple questions. First, how will the communication be sent? Second, who is the user that receives the communication? Third, when does the communication need to be sent?

Identifying how to communicate with users

To generate correspondence, you need to know "how" you want to communicate with the recipient. In other words, what's the right channel a user should receive a correspondence.

Pega provides four correspondence types to communicate with users: email, text message, fax and regular mail. Each correspondence type provides unique functionality but share the same basic template.

Pega provides a rich text editor to create formatted correspondence. You create one or more correspondence templates for each type of correspondence.

Identifying users to communicate with

When sending a correspondence first determine 'who do I need to communicate with?' You can send a correspondence to a specific address, but what if that address is no longer valid? You would have to update the application anytime that address changed. To avoid this Pega uses a set of roles to use with correspondences.

Pega defines the following roles for a correspondence:

Role	Description
Owner	The person who created the case.
Customer	The person on whose behalf the case is transacted. This person may not process the case, but may want — or need — notification of any changes.
Interested	A person who tracks the progress of a case but does not process the case.

The recipient is identified according to the channel selected, such as an email address for email correspondence. You can also represent the roles of case participants as parties, such as customer, owner or interested, and send correspondence to a specific role. Pega then identifies the recipient's address from the party information.

You are not limited to specifying a single role for a correspondence. For example, you may want to send a correspondence to the customer and all interested parties. To do this Pega uses something called a party. A party identifies the recipient of the communication and may contain one or more of these roles.

Identifying when to communicate with users

The last question you need to answer is “when” do you communicate. Pega simplifies sending a correspondence by allowing you to simply add a step to your case. Then you just configure who to send it to and the content of the message.

Sending an email from a case

A common use case for sending a correspondence during a case is sending a confirmation email after the user has completed a series of steps. You probably experienced this many times while completing a purchase online.

You accomplish this in Pega by adding a Send Email step to your case and then configuring the step.

Adding a Send Email step

To add a Send Email step:

1. Click **Add Step**.
2. Click **More > Utilities**.
3. Select **Send Email**.
4. Click **Select**.

Configure the Send Email step

Email configuration has three parts: **Subject**, **Message** and **Send to**. To configure the Send Email Step:

1. Enter a subject for the email.
2. Complete the body of the message by using the rich text editor to create a message or by specifying a Correspondence template.
3. From Designer Studio, under **Send to**, select **Field**.
4. Enter the recipient for the email using the options provided.

Note: Additional options are available in Designer Studio.

5. Click **Save**.

How to configure correspondence

Create correspondence rules to define, in HTML, templates for the content of outgoing correspondence. Each correspondence rule contains text for one type of correspondence such as email, letter, SMS phone text, or fax. JavaServer Pages (JSP) tags or directives allow correspondence to incorporate property values and calculations.

Informally, correspondence rules are sometimes called templates, as they define form letters for property values.

For simple notifications, you might add text directly in the rule. For richer content, you can enter dynamic fields that reference properties, or rules such as sections, paragraphs, or correspondence fragments. During flow processing, the system uses the source content in the correspondence rule to generate a customized message for the recipient.

Identify the correspondence type

When you create a correspondence rule, you specify the correspondence type as a key identifier. A correspondence type rule indicates whether a piece of correspondence is a printed letter, fax, email, or SMS phone text. Each type is associated with a different Data- subclass, such as Data-Corr-Email, that holds the content of correspondence items.

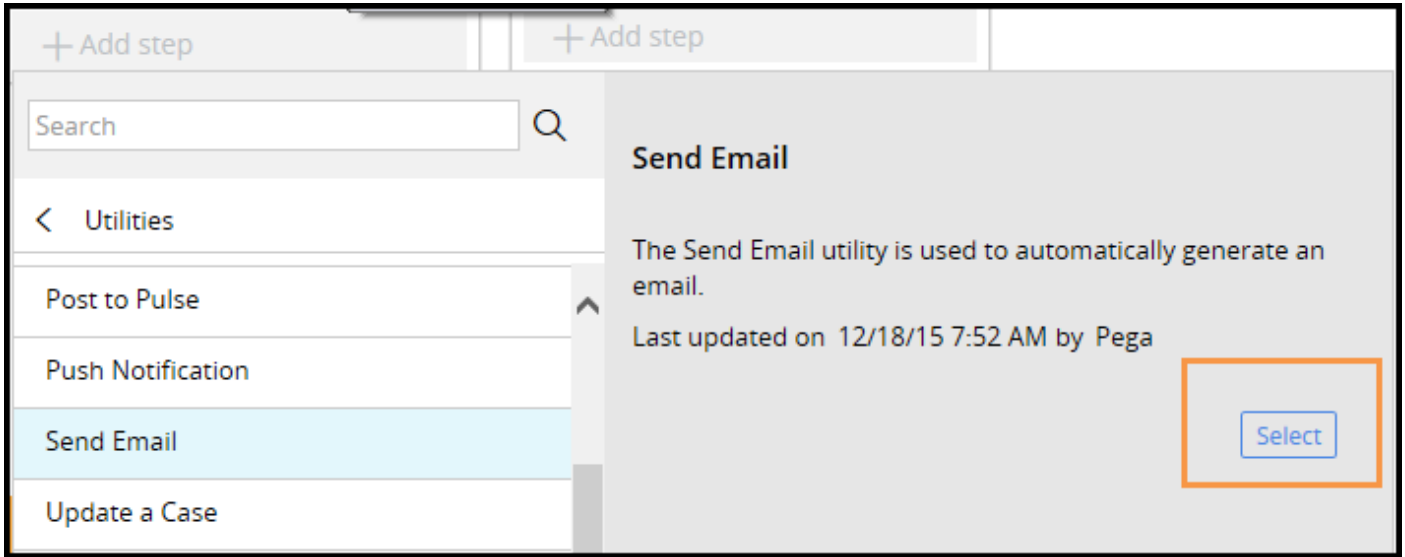
Be careful when selecting the correspondence type. Your specification should state who is receiving the correspondence and how the correspondence is sent. Before sending correspondence, the system references work party contact information to make sure the recipient can receive the correspondence. For example, the system cannot send email to a customer party that does not have an email address.

Important: Email correspondence can be configured from the case life cycle in either Pega Express or Designer Studio. For other types of correspondence, you must create a correspondence rule in Designer Studio and send the correspondence using a correspondence activity.

Send email using the Send Email step

You can add the Send Email step to your case life cycle flow diagram to automatically send emails as a case advances during the business process. This step is useful if you want to notify or send information to users after an action has been performed on a case. For example, the system can notify a manager that, as the result of an automated decision, a purchase request does not need approval.

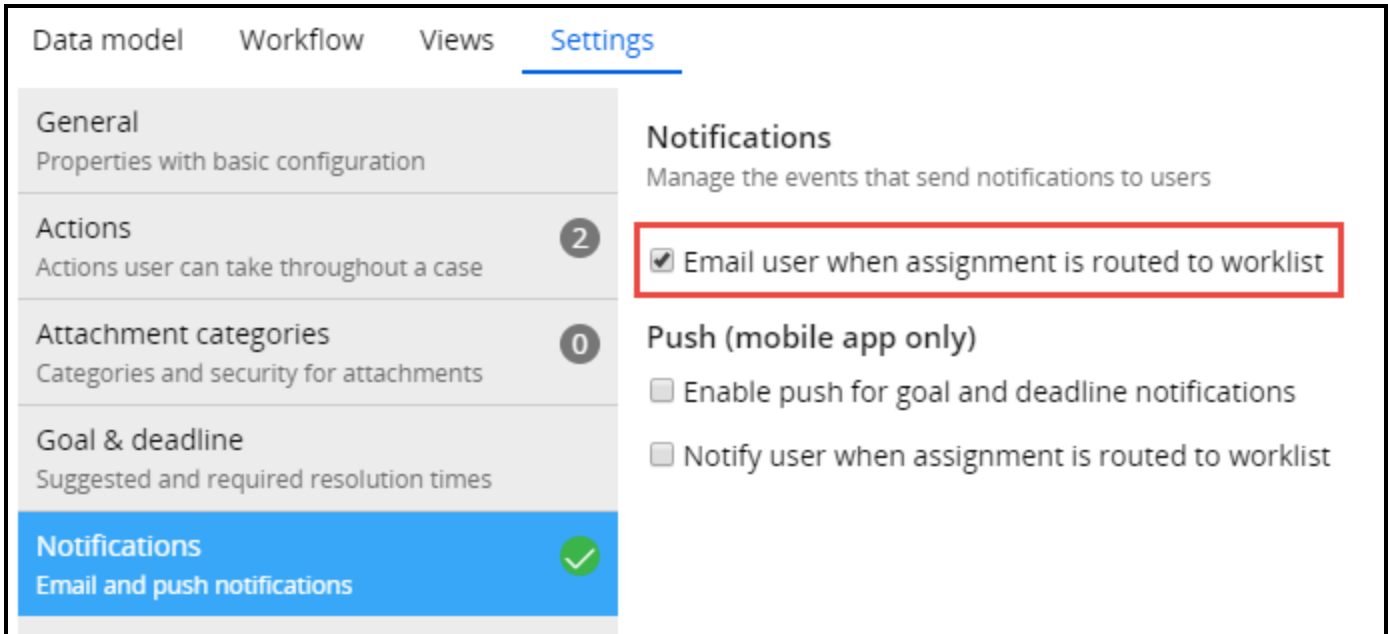
To add a Send Email step to the case life cycle, select **Utilities > Send Email**, and then click **Select**.



Send assignment notification emails

You can configure correspondence to notify a user of an assignment. When a case reaches an assignment, Pega Platform sends the assigned user an email informing the user of the pending assignment.

To enable assignment notifications for a case type, click the **Settings** tab and select the **Notifications** tab, then enable the **Email user when assignment is routed to worklist** option, using either Pega Express or Designer Studio.

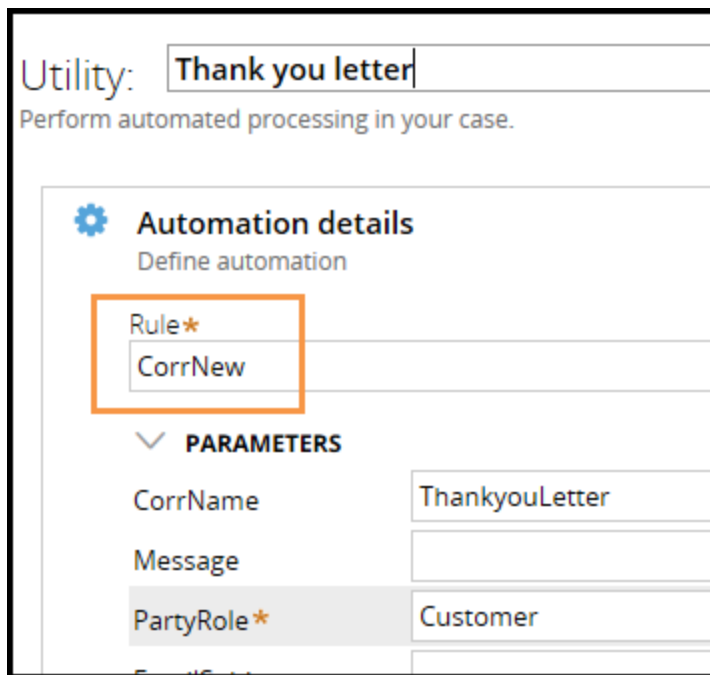


Note: In Designer Studio, the **Notifications** tab presents additional options to change the subject or content of the assignment notification.

Send other types of correspondence

Similar to the Send Email step, you can use a Utility shape in a process flow to automatically send correspondence. A Utility shape configured with the *CorrNew* activity offers greater flexibility than the Send Email step. You can use the activity to send all types of correspondence including mail, fax, and text messages.

When you add a Utility shape to your process, open the properties panel and select **CorrNew** in the **Rule** field. Select your correspondence rule in the **CorrName** field as shown in the following example. When a case reaches the utility, the system sends the Thank you letter correspondence.



Utility: **Thank you letter**

Perform automated processing in your case.

Automation details
Define automation

Rule*
CorrNew

PARAMETERS

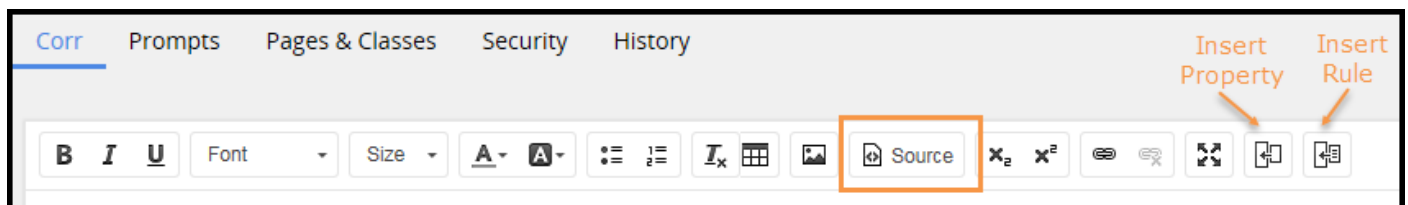
CorrName	ThankyouLetter
Message	
PartyRole*	Customer

Add the content

Add the message for the correspondence using the rich-text editor. You access the editor either from the contextual properties panel of the Send Email step, or on the **Corr** tab of the correspondence rule form. To create a correspondence rule in Designer Studio, open the **Create** menu and select **Process > Correspondence**.

The editor includes a toolbar with controls for setting text styles, spell checking, list formatting, alignment, and for inserting images and graphic elements.

The toolbar also contains controls for viewing the source HTML in the text area, and for inserting properties and rules that contain correspondence content. You can use a combination of text and referenced content sources to create your correspondence.



- Click **Source** to view or update the source as HTML codes. In source mode, you can add HTML elements and JSP tags directly. For example, you can add the when JSP tag to conditionalize a portion of the HTML code.

Note: When the correspondence is an email, the outgoing email includes HTML formatting, even if no HTML elements appear within the source. When the correspondence is phone text, the message does not contain HTML formatting.

- Click the **Insert Property** icon to include properties in your application such as pyID, LastName, and Department. For example, you may want to inform the recipient that the case is currently under review by the auditing department. You can reference the property .pyID to insert the case ID into the correspondence, rather than providing a generic message.
- Click the **Insert Rule** icon to include content in other rules such as paragraphs, sections, and correspondence fragments. You can also include other correspondence rules.

Note: The **Insert Rule** icon is available only on the correspondence rule form.

Paragraphs present formatted text that can include colors, fonts, styles, and images. Paragraphs allow you to reuse content that is used elsewhere. For example, you use a paragraph rule to present instructions on a form. You can then use that paragraph rule in correspondence to describe the action the recipient is expected to perform. Referencing shared content ensures that your correspondence includes the most current version.

Sections allow you to reproduce part or all of a form in the correspondence. Use a section to achieve greater control over the positioning of content in the correspondence.

Correspondence fragments are useful for reusing boilerplate content, such as a mandatory disclosure or links to an organization's social media channels.

The editor uses angle << >> brackets to mark properties and rules. During flow processing, these elements are replaced with the values. The following example shows how the inserted properties *.Office* and *.Employee.Manager* are replaced with case data when the correspondence is sent.

in our <<.Office>> office no later than 9AM. You will be greeted by your manager, <<.Employee.Manager>>
at TGB.

When the correspondence is sent, the system replaces the properties with their values.

in our Atlanta office no later than 9AM. You will be greeted by your manager, Sam Rivers.

Routing assignments

Introduction to Routing Assignments

An efficient process design routes assignments to users who can best perform the work. Sometimes, the correct user is an individual who has a specific role. At other times, anyone in a specific group of users can perform the assignment. Design your process so that it routes assignments to the best qualified users. This approach helps ensure that work is done correctly and completed on time.

After this lesson, you should be able to:

- Describe the role of routing in a case
- Explain the role of worklists in routing
- Explain the role of work queues in routing
- Explain the role of a router in routing assignments
- Route an assignment to the appropriate user

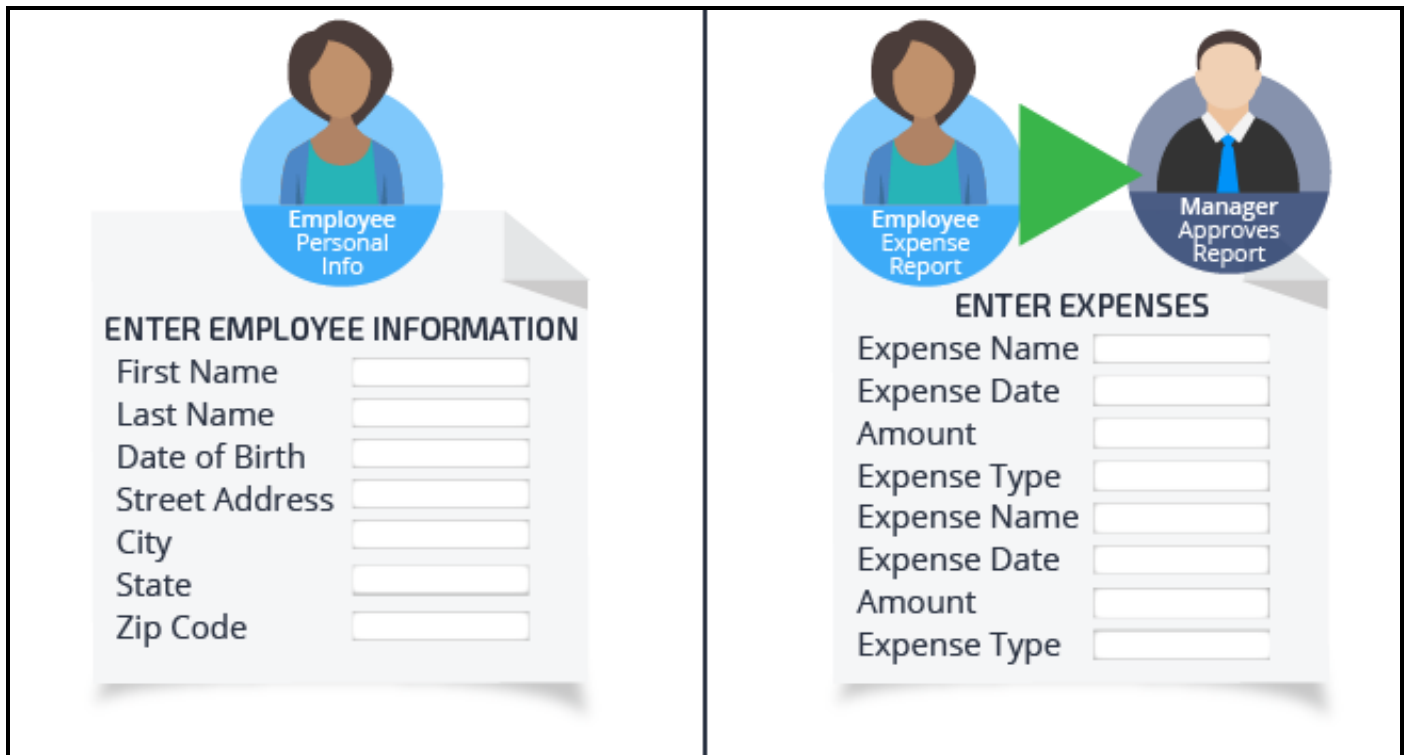
Assignment routing

When processing a case, it is common for more than one person to complete work on the case. For example, when creating an expense report, an employee creates the report, a manager approves it, and payroll sends the money — three sets of people work on the same case.

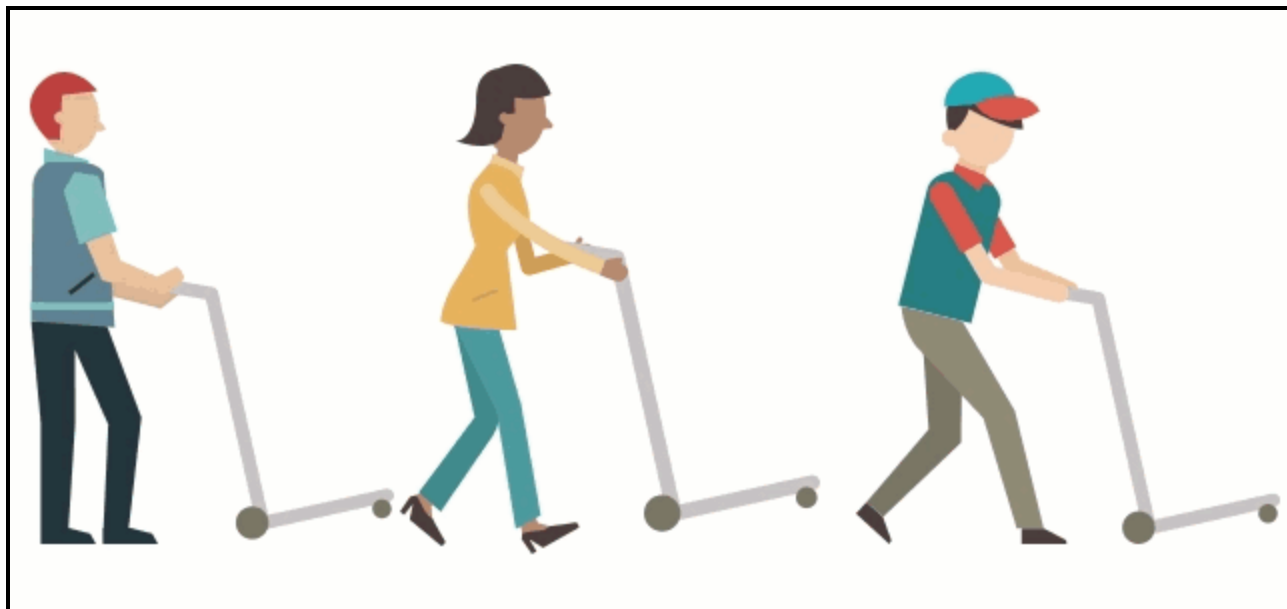


As part of modeling a process you define where the work should go. Assignment steps define the work to do. The question you need to ask yourself when designing your assignment is: who should do the work?

You route an assignment to a single user if the current user should perform the task or you know which user will do the work. For example, you would route to the current user if you have several data collection screens since the current user would likely perform them all. You would route to a specific user in the situation where you have an expense report approval process. The user who starts the expense report can't approve their own expenses. Instead you route the approval task to the manager of the employee to approve it.



You route an assignment to a group of users if a set of users could complete the task and it does not matter which user completes the task. For example, after a user completes an auto insurance claim, it does not matter which claim processor reviews the claim. The task can be routed to a work queue where claim processors go to get claims.



Worklists and work queues

Users complete assignments as a case moves toward resolution. When you configure the router setting in an assignment, you specify either a specific user or a work queue accessible to a group of users.

Worklists

A **worklist** is a list of all open assignments, ordered by urgency, for a specific user. For example, an assignment that requires a human resources manager to approve employee time off requests routes to the human resources manager's worklist.

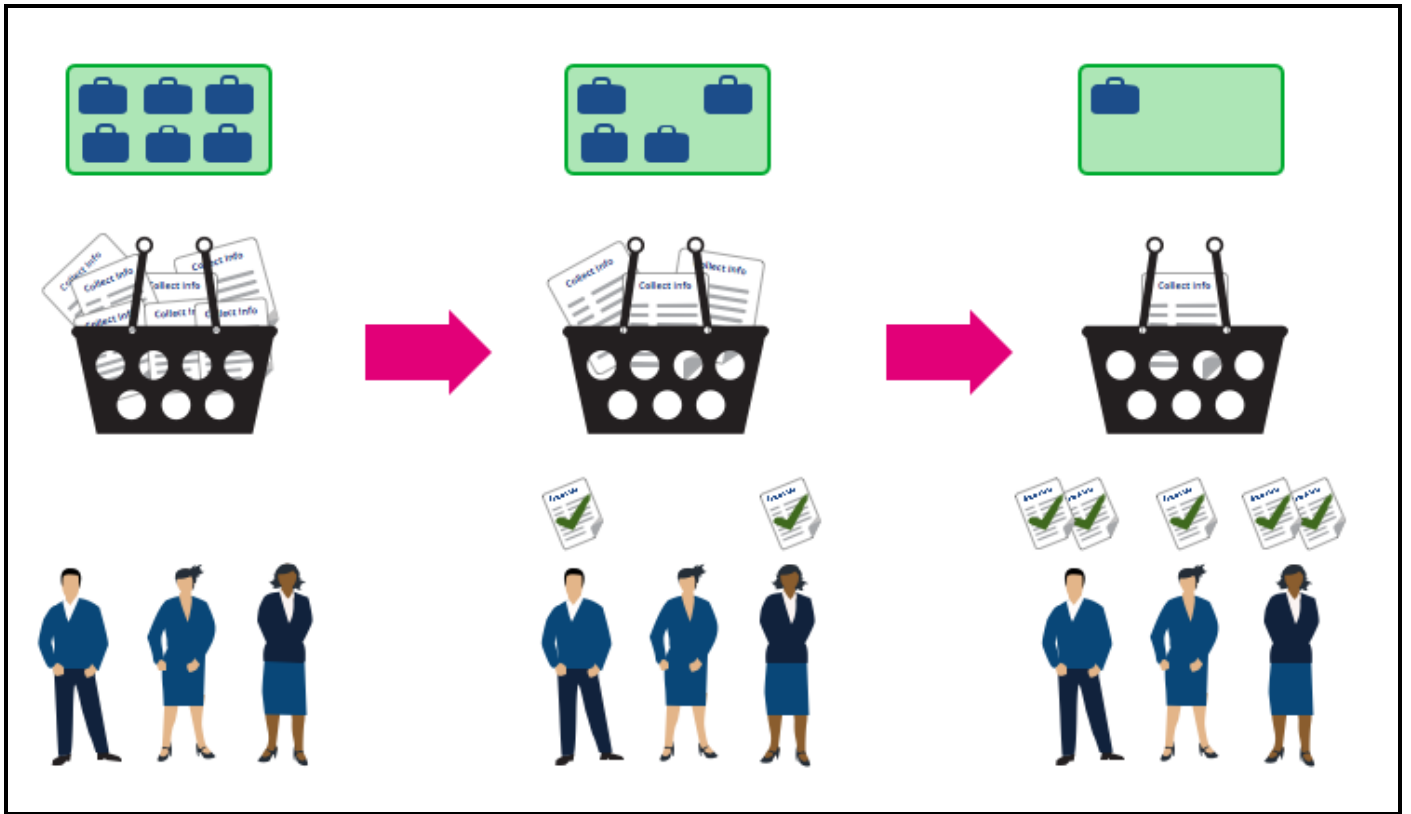
Note: Managers can access and assign work to the worklists of users who report to the managers.

Work queues

A **work queue** is a list of all open assignments, ordered by urgency, for a group of users.

Assignments stay in the work queue until a user associated with the work queue selects an assignment, or a manager sends an assignment in the work queue to a specific user. For example, any user who belongs to the benefits work queue can update an employee's medical insurance policy. The work queue for the employee benefits team receives employee benefit update requests. A user on the benefits team selects an assignment from the work queue and begins working on the assignment.

The following image illustrates routing for assignment to update customer contact information. The baskets represent work queues. The first column shows all of the open customer contact update assignments in the work queue. The second column shows that two team members each took an assignment from the work queue. The team has four remaining assignments in the work queue. The last column shows one remaining open assignment in the work queue.



KNOWLEDGE CHECK



ANSWER

A user accesses assignments from a _____. Users in a work group access assignments from a _____.

worklist, work queue

Configuring assignment routing

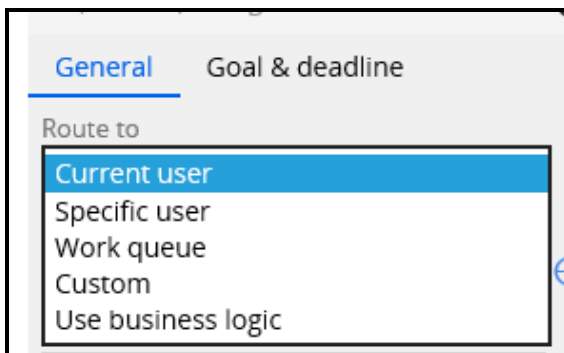
In Pega Express or Designer Studio, you configure routing in the case life cycle. When you configure routing, you are accessing Pega **router** rules. Routers contain logic that determines the appropriate user or work queue based on parameters that you define. Pega provides many standard routers to suit your specific requirements.

You can assign a task to a user's worklist or to a work queue when you add an assignment step and an Approve/Reject step. You can also leverage automated routing capabilities that enable you to route tasks based on criteria that you specify.

Configuring routing in an assignment step

To specify routing on a case life cycle assignment step:

1. On the case life cycle, select an assignment step.
2. On the **General** tab of the step property panel, select a **Route to** option from the list to indicate how to assign the task at run time.



The following table describes the routing options available in the **Route to** list.

Option	How a task is assigned	How you configure the option
Current user	Assigns the task to the worklist of the user who last updated the case.	No additional information is required.
Specific user	Assigns the task to the worklist of another user in your application.	Below the Route to field, select the user name from the list.
Work queue	Assigns the task to a work queue. The task can be processed by any user who has access to the work queue.	Below the Route to field, select the work queue from the list.
Custom	Assigns a task using the advanced routing capabilities available in Pega Platform. For example, you can route tasks to users based on their role in the organization or based on their work skills.	For instructions, see the Configuring custom routing section in this lesson. Available only in Designer Studio
Use business logic	Assigns a task to a user or work queue based on the outcome of a decision condition that you define.	For instructions, see the Configuring business logic routing section in this lesson.

3. Click **Save** when you have completed your configuration.

Configuring business logic routing

1. In the **Route to** list, select the **Use business logic** option.
2. Next to the **Use business logic** field, select the **gear icon**. This opens the **Business logic** configuration dialog as shown in the following image.

Business logic

Route work based on these conditions

1 Action Value*

Route to operator

When Field Comparator Value

Please select a field.

+ Add condition

otherwise Action Value*

Route to operator

3. In the **Action** field, select either **Route to operator** or **Route to work queue**.
4. In the **Value** field, select a user name or work queue from the list.
5. In the **When** area, in the **Field**, **Comparator**, and **Value** fields, enter values that define the decision condition you want the system to evaluate. If the condition evaluates to true, the system assigns the task to the user or work queue you specified in the **Action** and **Value** fields.

6. In the **otherwise** area, in the **Action** and **Value** fields, enter the user or work queue you want to assign the task to if the **When** condition is not true.

The following image shows business logic that routes a task to the Manager if the proposed salary is greater than USD 100,000. If the proposed salary is less than that amount, the task is assigned to the Recruiter.

Business logic

Route work based on these conditions

	Action	Value*
1	Route to operator	Manager.HRAppls

When	Field	Comparator	Value
	Proposed salary	is greater than	100000

+ Add condition

	Action	Value*
otherwise	Route to operator	Recruiter.HRAppls

Configuring custom routing

The Custom routing option in Designer Studio gives you access to the full suite of Pega routers rules. For example, in an insurance application, you can use a router to send claims for damaged art work to claims adjusters who are trained in that specialized area.

When you select the **Custom** option, do the following:

1. In the **Assignment type** field, select either **Work queue** or **Worklist**.
2. In the **Router** field, select the appropriate router. This displays the router's parameters in the **Parameters** area. The available parameters depend upon the router you select.

In the following example, users in the **Conduct Phone Screen** step must be able to speak French. The step is configured with a **ToSkilledGroup** router. For this router, you specify a workgroup that includes users who are defined as having French as a skill.

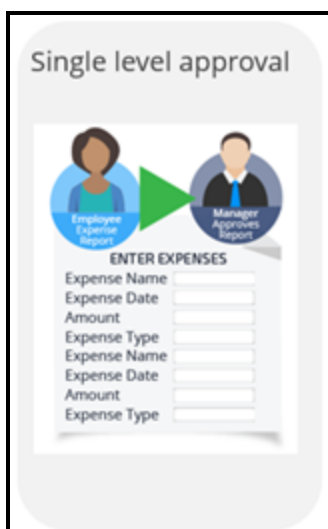
The screenshot shows a configuration interface for a step. On the left, a process flow is visible with a step titled "2. Screening" containing two sub-steps: "1. Create Candidate Par..." and "1. Conduct Phone Scree...". The "1. Conduct Phone Scree..." step is highlighted with a red box. On the right, the configuration panel for the selected step is shown. It has two tabs: "General" (selected) and "Goal & deadline". Under "General", the "Route to" dropdown is set to "Custom", "Assignment type" is "Work queue", and "Router" is "ToSkilledGroup". Below these, the "Parameters" section is expanded, showing a "workgroup*" dropdown set to "FRrecruiterTGB". At the bottom, a table lists skills for the router:

Skill	Level	Required
French	1	✓

An "Update Skill" link is located below the table.

Single level approval routing

You can configure single level approval routing in Pega Express and Designer Studio. You can assign an approval task to a specific user, work queue, or the reporting manager of the user working on the case.



To specify approval process routing:

1. In the case life cycle, select an **Approve/Reject** step.
2. On the **General** tab of the contextual step property panel, in the **Route to** area, select either **Specific user** or **Work queue**.

- If you select **Specific user**, select one of the following options.

Option	How the task is assigned	How you configure the option
User name	Assigns the task to the worklist of another user in your application.	Below the Route to field, select the user name from the list.
User Reference Field	Assigns the task based on the value the user enters in a specially configured field in a user view. These fields identify users anywhere within the organization. For example, assume Jane Doe enters her name in the Recruiter field. If you select the Recruiter field in the User Reference Field drop-down, the case is routed to Jane Doe.	In the drop-down list that is displayed beneath the User Reference Field option, select the value.
Reporting manager	Assigns the task to the reporting manager that is defined in the current user's operator ID.	No additional information required.

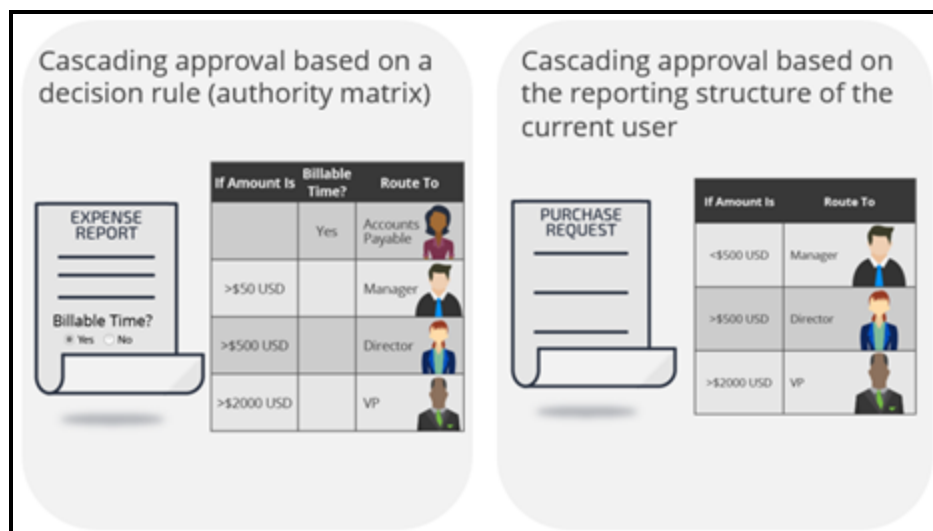
- If you select **Work queue**, below the **Route to** field, select the work queue from the list.

3. Click **Save** to save the changes to the case life cycle.

Cascading approval routing

You can assign cascading approval tasks in Designer Studio based on either:

- The current user's reporting structure.
- The outcome of decision rule (matrix routing).

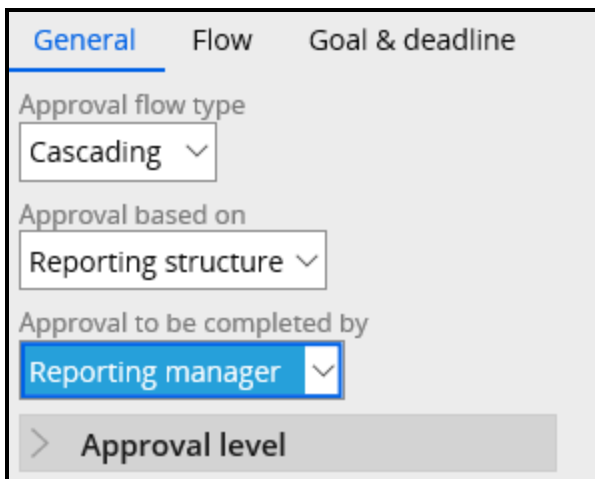


User's reporting structure — cascading approval

When you use a cascading approval process, assignments route to managers based on the current user's reporting structure, or the results of an authority decision table.

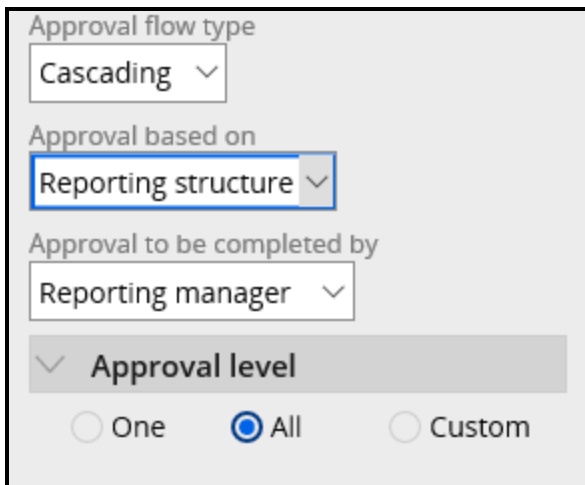
To route assignments based on a reporting structure in a cascading approval process:

1. On an **Approval Smart Shape**, click to display the Approval properties panel to specify the routing behavior.
2. Under **General**, in the **Approval flow type** field, select **Cascading**.
3. In the **Approval based on** field, select **Reporting structure**.
4. In the **Approval to be completed by** field, select the first person to review the case.



The screenshot shows the 'General' tab of the Approval properties panel. It contains three dropdown menus: 'Approval flow type' set to 'Cascading', 'Approval based on' set to 'Reporting structure', and 'Approval to be completed by' set to 'Reporting manager'. Below these is a collapsed section for 'Approval level'.

5. Expand the **Approval Level** section and click an option to indicate how many approvals are required for each case.



The screenshot shows the 'Approval level' section expanded. It contains three radio buttons: 'One', 'All' (which is selected), and 'Custom'.

The following table lists the options and their descriptions.

Approval level option	Description
One	The manager specified in the Approval to be completed by field must approve the case.
All	The entire reporting structure, starting with the manager specified in the Approval to be completed by field, must approve the case.
Custom	A variable number of reviewers must approve the case, based on a list of when conditions.

- Optionally, if you select **Custom**, click **Update custom levels** to display the current custom levels.
- Click **+Add custom approval** to display the condition entry window.

- Enter values in the **When** and **Levels of approval** fields. For example, you can require approval by two managers in the reporting structure when a purchase order exceeds USD 25,000.
- Click **Submit** to record the new custom level.
- Click **Save** to save the approval configuration.

Authority matrix cascading approval

When you route assignments based on an authority matrix, you designate a decision table. The outcome of the decision table determines where to route the assignment.

To route assignments based on an authority matrix within a cascading process:

- In the **Approval type flow** field, select **Cascading**.
- In the **Approval based on** field, select **Authority matrix**.
- In the **Decision table for matrix** field, select an appropriate decision table.

4. Optionally, if a decision table for the matrix is unavailable, populate the **Page list property** field with the name of a page list that contains a list of reviewers. You can also populate the **Approver property field** with a single-value property that provides a unique identity for the reviewers in the page list.
5. Click **Save** to save your work.

Delegating business rules

Introduction to delegating business rules

Business applications exist to meet the needs of the business. However, business needs can change frequently — and often unexpectedly. These changes manifest themselves as changes to business policy, either as updates to internal procedures, or as evolving industry guidelines or government regulations. Delegating business rules enables you to put a controlled set of business policies under the control of the business. This enables your business applications to keep pace with changing business conditions.

After this lesson, you should be able to:

- Explain the benefits of delegating business rules
- Explain the purpose of delegating business rules
- Identify the types of business rules best suited for delegation to business users
- Identify the necessary details for delegating business rules
- Delegate a business rule to one or more business users

Business rule delegation

Business policies change — sometimes suddenly — in response to internal and external factors. Pega Platform enables you to design and implement applications that respond to change with agility and efficiency.

By recording business policies in rules rather than in code — a model-driven approach — an application can provide a degree of modularity and transparency that can simplify maintenance.

An effective Build for Change® strategy requires application designers to go beyond maintenance efficiencies. You can delegate responsibility for updating selected parts of each application to business users. Delegating rule changes to business users helps promote an agile response to continuously changing business conditions.

Rule delegation can also help reduce the workload for architects of minor, low risk maintenance items and provide a degree of empowerment to those closest to the day-to-day operations.

This shared responsibility goes a long way towards the success of your applications.

Rule delegation applies to selected rules. For example, an expense report application that requires additional approvals for amounts over USD 1,000. If the requirement changes to USD 500, someone must update the appropriate decision logic. Adopting an appropriate delegation strategy improves the business's ability to adapt an application to changing business conditions.

KNOWLEDGE CHECK



ANSWER

What is the purpose of rule delegation?

Rule delegation empowers business users to respond to continuously changing business conditions.

Best practices for business rule delegation

Establishing guidelines and best practices for rule delegation is critical to a successful rule delegation strategy.

Consider the following guidelines when adopting a rule delegation strategy:

- Establish a unique access group for the business users to whom you will delegate rules.
- Delegate rules for process items that change repeatedly.
- Delegate rules that are easy for the business user to change.

Establish a unique access group

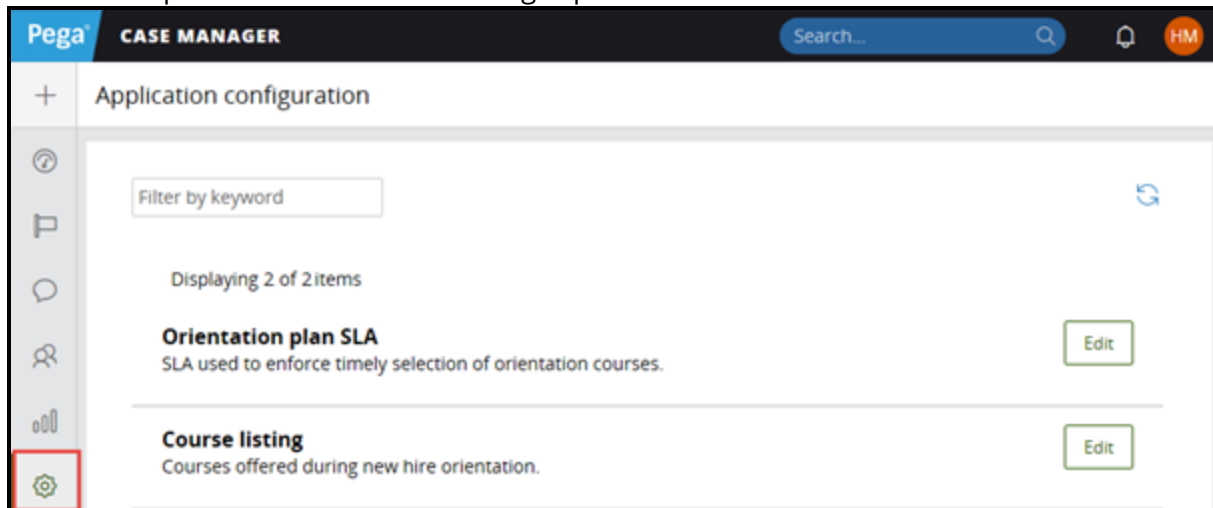
Consider the business users with the ability to update the rules. Do they belong to a unique access group, or do you have to create a new access group? For example, if all users in the HRManager access group should be able to update the courses available to new hires, then you can use the HRManager access group for the courses list rule delegation. If the business wants to restrict course list maintenance to the senior human resources managers, then you may need to configure an access group for senior human resources managers.

Rule selection

You determine which rules to delegate based on business needs and ease of business user access. Selecting a specific rule type determines how to simplify the way a business user changes them. For example, service level rules provide a set of choices for selection such as sending an email and transferring an assignment to another user. Decision table rules describe criteria for whether an action occurs. For example, a loan application processing decision table can indicate the levels of underwriter review based on certain factors such as income level and credit score.

The Pega Platform rule types you can delegate to business users so that they can access the delegated rules from their case manager portal are paragraph, decision table, data types that have data records, correspondence types and service level agreement (SLAs).

The following image shows an example of how HR managers access a delegated course listing and orientation plan SLA in the Case Manager portal.



Note: You can delegate other rule types using the **Add to favorites** action, but managing rules delegated the **Add to favorites** action can be cumbersome for business users due to the complexity of the user experience. For more information about using Add to favorites, see [Adding a rule as a favorite](#).

KNOWLEDGE CHECK

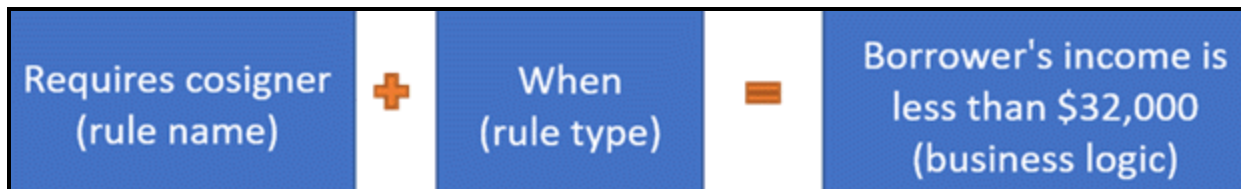


What are the criteria for determining rules to delegate?

You determine which rules to delegate based upon business needs and ease of business user access.

Rule names

Choose rule names that are meaningful in the business context. A good test of a rule name is to build a sentence using the rule name, the rule type, and the business logic. If the business users understand the sentence, then the rule name is meaningful. For example, "Requires cosigner" describes a rule that requires a loan cosigner with a "when" rule type stating that if a borrower's income is less than USD32,000, the loan requires a cosigner.



KNOWLEDGE CHECK



What is a good test for a name for a delegated rule?

A good test of a rule name is to build a sentence using the rule name, the rule type, and the business logic.

Additional considerations

A Pega Business Architect should be aware of the following additional considerations in order to participate in discussions.

- Consider the environment business users use to manage rules. Managing rules in the production environment is risky if the rules are poorly structured. A compromise is to use an authoring environment for rule management. Business users can update and test rules in an authoring environment without affecting the production environment. You can use the migration wizard to promote the rules into production.
- Consider providing appropriate training and documentation for future new business users and current business users.

How to delegate rules to business users

Rule delegation enables business users to change simple application logic without knowing all of the related technical details and without involving IT. For example, you can delegate correspondence or service level agreement rules to a business line manager. The business line manager views the delegated rules and makes changes in the Case Manager portal.

Several parties collaborate to perform the tasks necessary to delegate rules. The following table lists the tasks in a logical order to streamline the process.

Task	Role performing task
Identify the business users who manage the delegated rules.	Business architect (BA)
Create an access group for business users responsible for managing the delegated rules.	Senior system architect (SSA) or lead system architect (LSA)
Organize the business users who manage the delegated rules in the access group.	System administrator (during development, an SSA may perform this task with test operators to test the configuration)
Identify the rules to delegate.	BA
Organize the rules to delegate in a production ruleset. This ruleset must remain unlocked in the production environment as changes cannot be made to rules in a locked ruleset.	SSA
Delegate the rules for availability in a user portal.	SSA

Note: Task assignments may vary, depending on the business requirements, project team size, and project location.

Create an access group for users who manage delegated rules

Create an *Access Group* to organize the business users responsible for managing the delegated rules. A separate access group for delegated rules simplifies the management of the delegated rules. The access group includes the rights to view and modify the delegated rule while preventing other users from making unauthorized changes.

Note: In some organizations, creating an access group may be the responsibility of a system administrator or others who understand the application structure and security implementation.

Identify the rules to delegate

In Pega Platform, there are no restrictions on which rules to delegate. However, to be successful, your rule delegation strategy must consider which rule types are best to delegate. In general, BAs work with the business users to determine what components of the business logic (which rule types) they want to maintain.

The best candidates for delegation are the rules most affected by frequently changing business conditions such as correspondence, paragraphs, decision tables, service level agreements, or when rules.

Start by delegating a small, focused set of decision rules and correspondence templates. As the business users become more comfortable with delegation, you can expand the number and types of delegated rules.

Organize the rules to delegate in an unlocked, production ruleset

To update any rule on a production system, an unlocked ruleset must contain the rule. An **unlocked ruleset** is a ruleset in which rules can be modified and saved.

An unlocked ruleset is a potential security risk, so it should consign solely to delegated rules. This enables a system administrator to prevent unauthorized access to critical process flows or case design elements. Changes to these elements can have drastic effects on case processing.

Add the ruleset to the application rule as a production ruleset. A **production ruleset** is a ruleset containing rules that can be modified after the application deploys.

Using a production ruleset to organize and contain delegated rules insulates the rest of the application from frequent changes. Because the original version of the rule still exists in a locked ruleset, a system administrator can revert to the original copy if something happens to the delegated version of the rule.

Important: The ruleset containing the delegated rules must remain unlocked in production so business users can make changes to the delegated rules. Changes cannot be made to a rule in a locked ruleset.

KNOWLEDGE CHECK



Why is organizing delegated rules in an unlocked ruleset necessary?

You cannot edit rules in a locked ruleset.

Delegate the rule

Create a new decision table, paragraph, correspondence, service level, or data type rule, or open an existing one.

Important: Remember to add the rule you want to delegate to the specified ruleset created for delegated rules.

Delegate the rule to the business user access group managing the delegated rule. For example, a human resources (HR) department may send an email to a new employee prior to the employee's first day of work. You can delegate the correspondence rule containing the email content to allow HR to update the email with a revised new hire orientation procedure.

Delegate Correspondence

What will the end user be able to edit?
Select one:

Manage content

Delegate to access group *

Managers for the HRApps application

Title *

New customer welcome email

Detailed description

This will be displayed to the end-user to help them understand what they are changing

Provide a meaningful title and detailed information about how the delegated rule affects the application. For example, if delegating a decision table, include a detailed description such as: The logic in this decision table determines whether a travel request requires additional manager approval. Changing the logic in this decision table may affect all subsequent travel requests submitted in this application.

Note: To delegate a rule, an operator ID must be assigned to an available role that has the *pxCanDelegateRules* privilege. By default, this privilege extends to users assigned to the Administrator or SysAdm4 access role. If you need to delegate a rule and cannot, contact your Pega Platform system administrator to confirm that your operator ID has this privilege.

KNOWLEDGE CHECK



Why is a meaningful description for a delegated rule important?

Business user delegates need to understand the possible impact of their changes.

Controlling the flow of a case life cycle

Introduction to controlling the flow of a case life cycle

Pega Platform provides options to control the flow of a case life cycle using business logic. Using business logic, you can skip stages or processes, run processes in parallel, or select multiple paths based on information in the case. You can also add multiple outcomes to an assignment, allowing users to select the appropriate path during case processing.

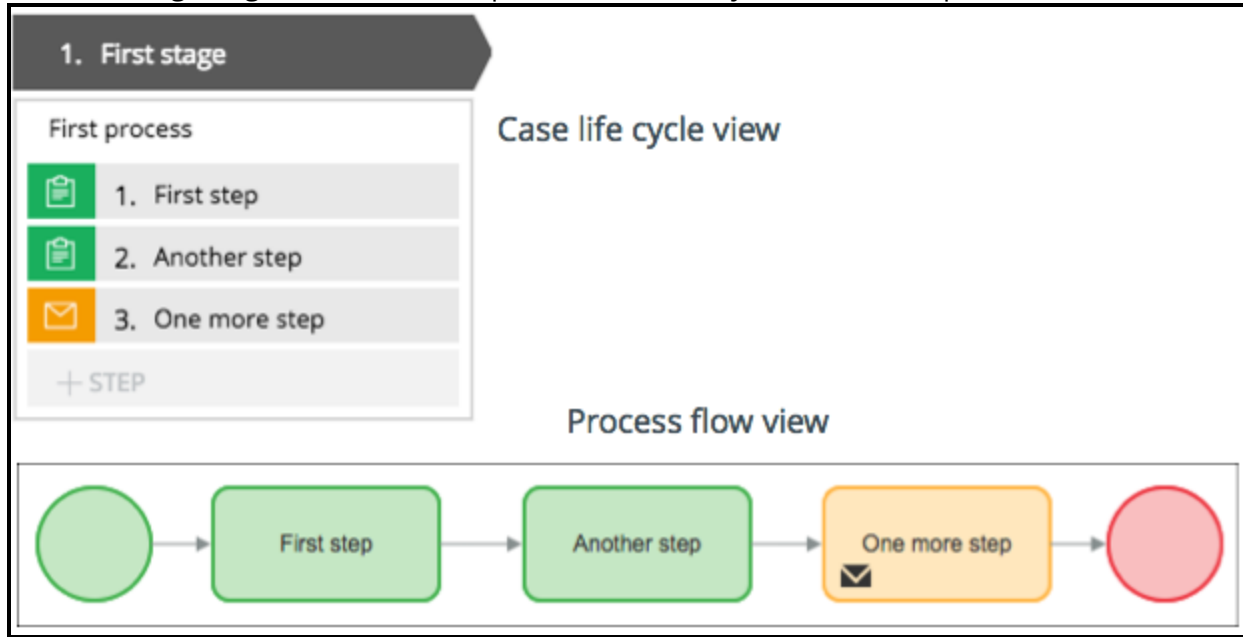
After this lesson, you should be able to:

- Identify how decision points affect case processing
- Define multiple processing paths in a case life cycle using a decision shape
- Define multiple processing paths in a case life cycle using an assignment shape
- Explain how to skip a process in the case life cycle
- Model parallel processes in a case life cycle

Conditional paths in a case life cycle

The case life cycle workflow defines the order in which tasks must be completed to move a case closer to resolution. In Pega Platform, these tasks are represented as steps, encapsulated in one or more processes. Processes in a case life cycle are implemented as process flow rules.

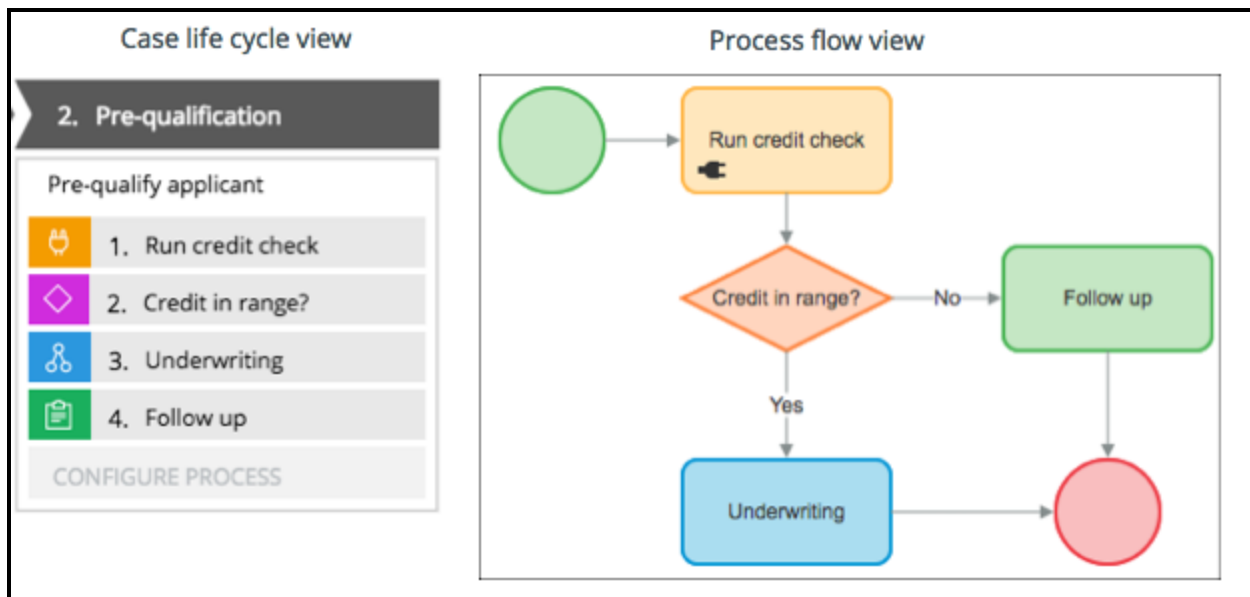
The following image shows an example of a case life cycle view and a process flow view.



Process flow rules built using Pega Express or Case Designer represent a single, linear path of action from the beginning of the flow to the end. The steps in a process execute in the order in which they are defined in the process. If the case life cycle workflow is more complex and requires branching or skipping of a step, you can add decision shapes to define conditional paths the case may take.

A conditional path is a branch in the life cycle of a case that is contingent upon run-time values. By defining the business logic decisions that cause a case to follow different paths, you can improve the flexibility of your application.

For example, you can evaluate an applicant's credit score in a mortgage loan case to determine whether the case begins the underwriting process or requires further investigation by a loan officer. The following image shows the case life cycle view and process flow view for evaluating an applicant's credit score in a mortgage loan case.



Because conditional paths are steps in a process, you can quickly transition from the sequential view of your case life cycle to a comprehensive view of shapes and connectors when you need to define a decision. You can then use the view of your choice to add steps to each conditional path.

KNOWLEDGE CHECK



What is a conditional path in a Pega process flow rule?

A conditional path is a branch in the life cycle of a case that is contingent upon run-time values. By defining the business logic decisions that cause a case to follow different paths, you can improve the flexibility of your application.

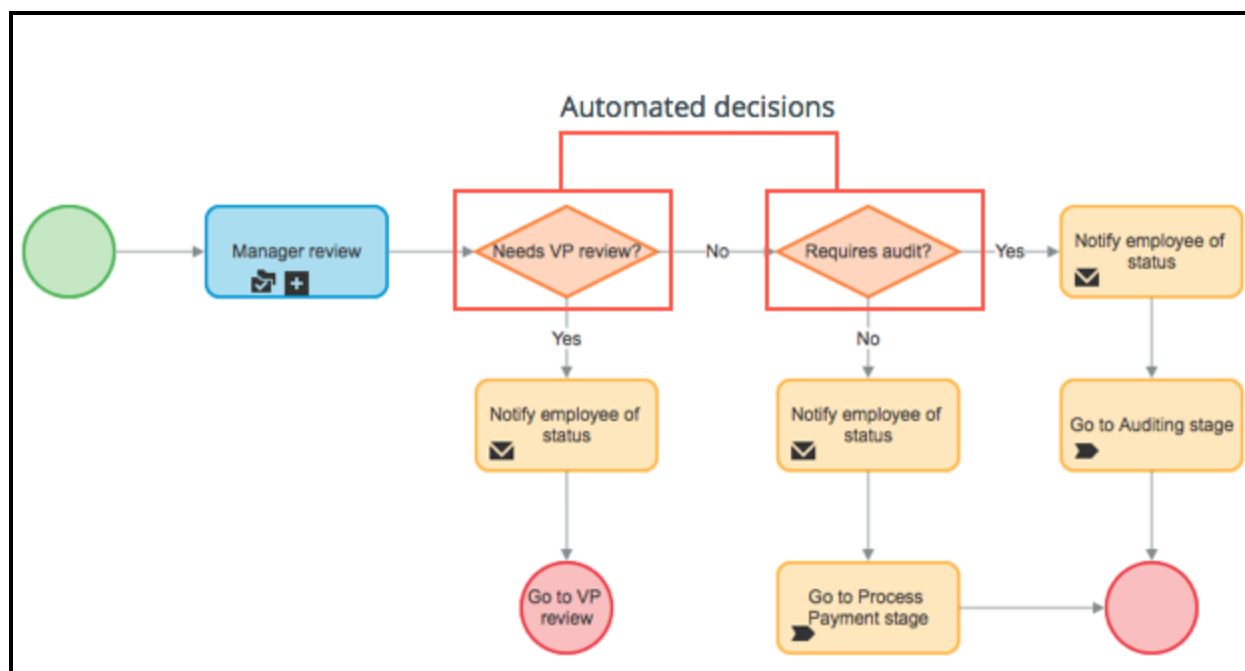
How to model a decision point in a process

Pega Platform enables you to configure processes that follow steps executed in sequence. Some processes are more complex. They need a decision to advance the case in one of several directions, or to more than one possible next step. You can model these decision points as automated decisions in a process flow rule.

Modeling automated decisions

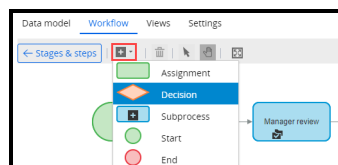
The application performs automated decisions. You define automated decisions by a set of one or more conditions to evaluate. The conditions to evaluate are also referred to as business logic. The result of the condition evaluation is a return value. An appropriate return value identifies the branch that the case follows to determine the next processing step.

In Pega Platform, you model automated decisions with a decision shape using one of the Pega supported decision rule types to represent the logic. The following image highlights two automated decisions within a process flow. Each decision shape has two possible return values, yes or no. Each yes or no return value moves the process to a different step.

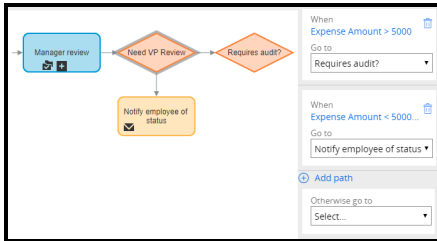


Use automated decisions when configuring the application to evaluate business logic conditions by using data elements defined in the case data model.

Add decision shapes in the Process Modeler by clicking the **Plus** icon in the diagram tab, then selecting the decision flow shape.



Configure the decision shape from its context panel. In Pega Express, define the when conditions that direct the flow from the context panel. For each when condition, from the **Go to** drop-down, choose the shape that achieves the desired flow branching.

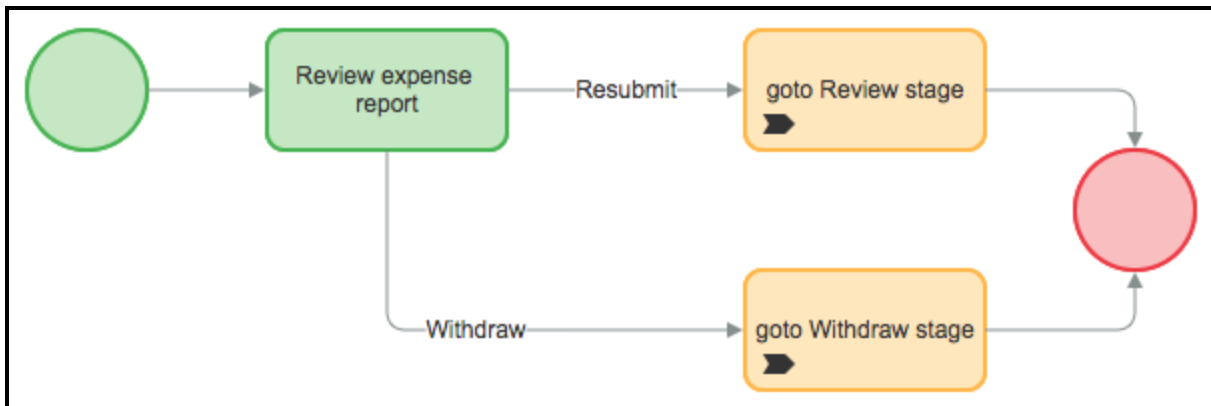


Modeling manual decisions

Users perform manual decisions. An assignment has multiple outgoing connectors that define the decisions. A user can complete the assignment by selecting any of the actions pointed to by the outgoing connectors.

Case processing follows the outgoing connector to determine the next process step. Manual decisions do not have a specific shape and are not reflected in the life cycle.

The following image shows a process flow in which a user can decide whether to withdraw or resubmit an expense report.



KNOWLEDGE CHECK



For which type of decision is the following statement true? The case life cycle does not differentiate between the branches that result from a decision.

The statement is true for both the automated and manual decisions.

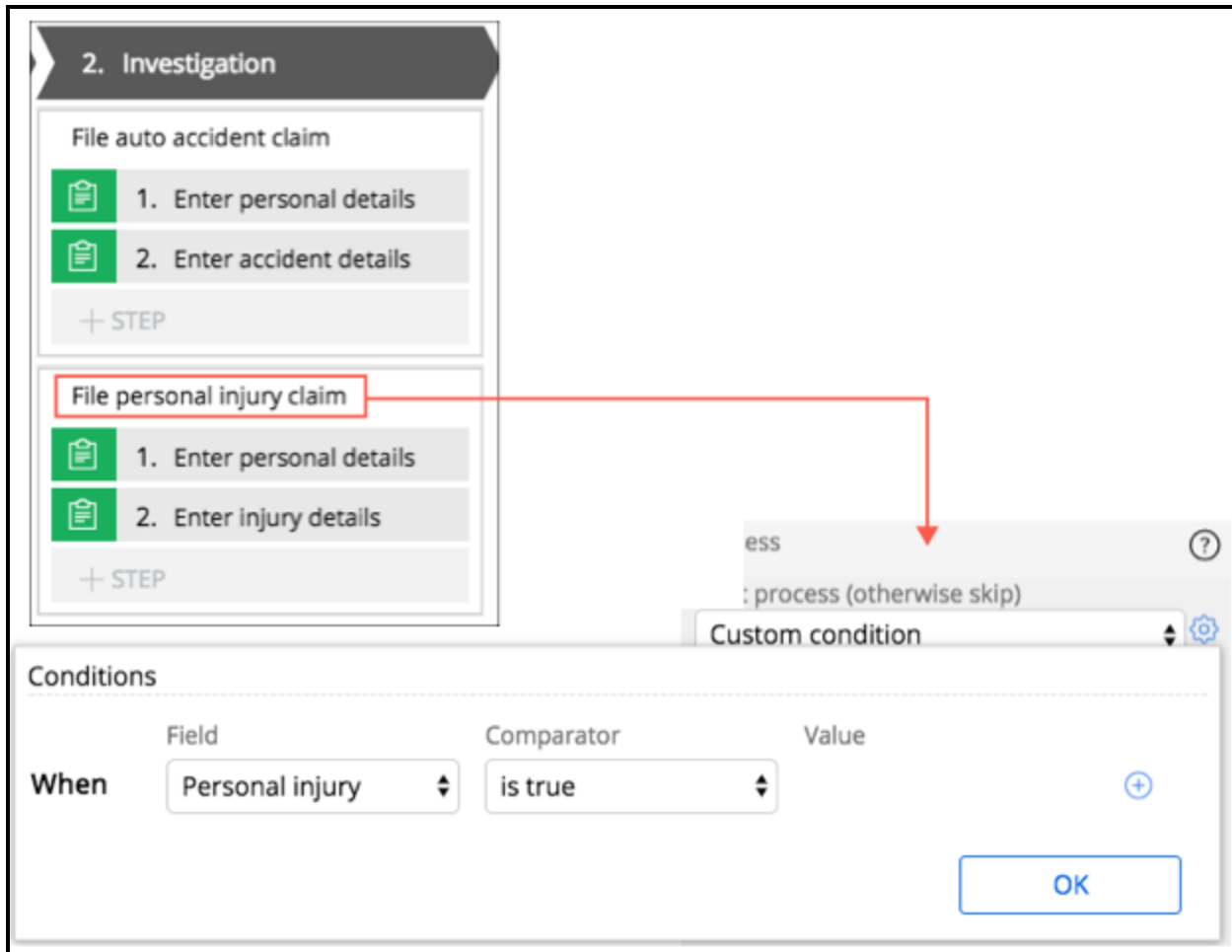
How to configuring processing for a case

You can configure processes in a case to run only when certain conditions are true, or to run concurrently with other processes in the stage.

Configuring a process to run when certain conditions are true

You can define conditions that control whether a process runs. If any of the conditions return a true result, the process runs. Otherwise, the process does not run.

For example, a process to open a medical claim resulting from an automobile accident is only needed if a party was injured in the accident. If the car was hit while parked and unattended, the medical claim process can be skipped.



Start process conditions are configured by selecting the process and entering or selecting a condition in the start when field of the properties panel.

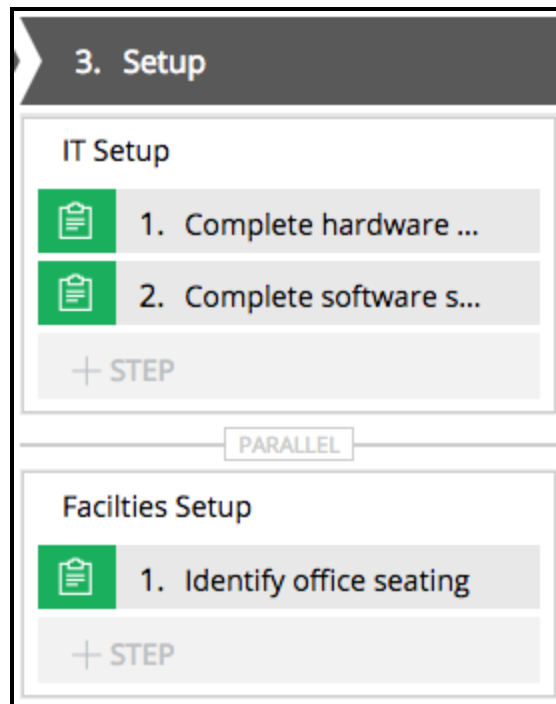
You can add more than one condition. Two or more conditions can be logically joined using the logical operators AND or OR.

Note: By default, all processes are configured with the "Always" condition, which always returns a true result. This default condition ensures that the process is always used in case processing.

Configuring a process to run concurrently with other processes

You can configure a stage to run multiple processes at the same time. Two or more processes that run at the same time are called parallel processes. Parallel processes allow a case to advance down multiple paths within a stage at the same time.

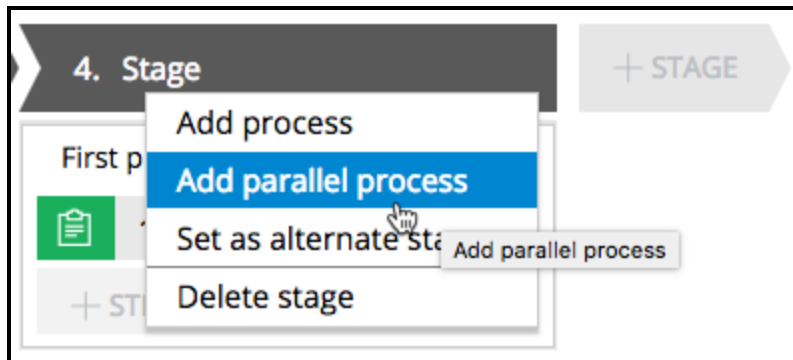
For example, a new hire onboarding case may contain one process for the IT department to provide the new employee with a configured laptop, and a second process for the Facilities department to assign the new employee an office — or other seating arrangements — and telephone number.



If the IT Setup and Facilities Setup processes can run concurrently, they can be configured as parallel processes. When an Onboarding case is executed, the IT Setup and the Facilities Setup processes run concurrently. The active assignments in either process can be performed at the same time.

Note: When a user opens the case to perform an assignment in one process, the case may be locked, preventing another user from performing an assignment in the other process.

Parallel processes can only be configured in Case Designer by adding the first process to the stage, then selecting **Add parallel process** from the stage menu.



KNOWLEDGE CHECK



What is the prime consideration for process order when configuring parallel processes?

Parallel processes do not have a configurable order. The processes you select for parallel processes must be adaptable to any order.

Circumstancing rules

Introduction to circumstancing rules

Applications often need to customize behavior to match the needs of a specific situation or circumstance. For example, a call center may need to enforce one set of performance objectives for clients with elite status, and a different set of performance objectives for clients without elite status.

In this lesson, you learn how to specialize case behavior through the use of circumstanced rules.

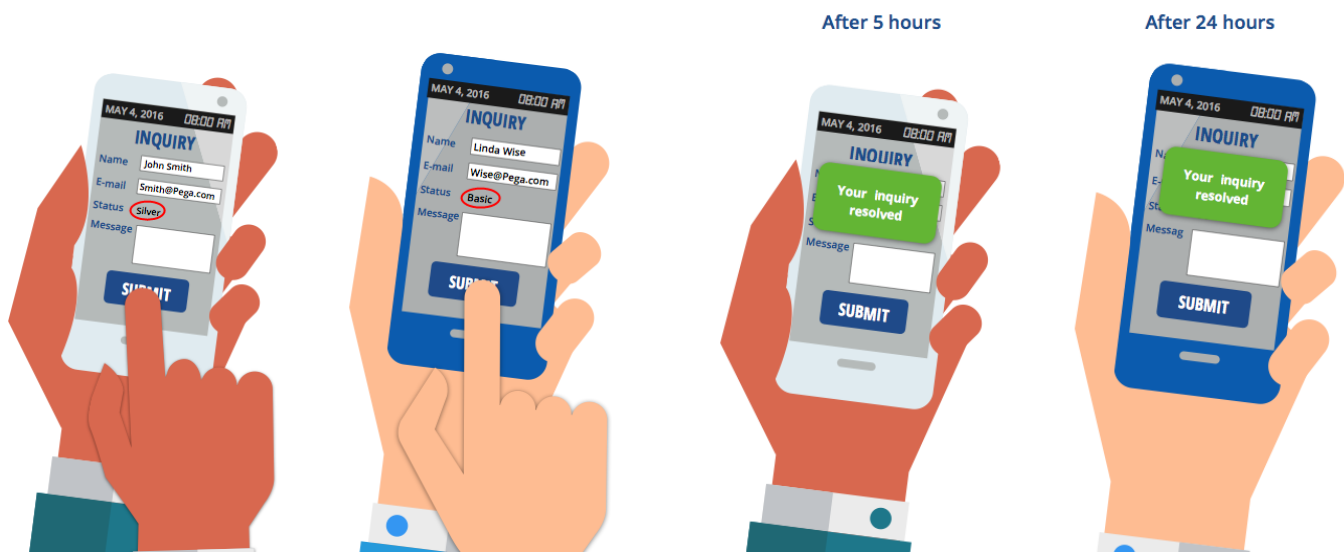
After this lesson, you should be able to:

- Explain how rule circumstancing supports rule specialization
- Differentiate between base rules and circumstanced rules
- Identify the types of circumstancing configurations supported by Pega Platform
- Circumstance a rule

Situational processing

Business processes must account for exceptions to typical case behavior. Exceptions make a business process more complex. This complexity makes processes difficult to maintain and update as business conditions change.

For example, a company promises to respond to customer complaints within one business day. For customers with silver status, the company promises a response in five hours. Reduced response times for customers with elite status are exceptions to normal business processing.



Simple exceptions like these can be difficult or impossible to model with a single rule. For example, a service level only defines one set of service expectations, and an assignment only applies one service level. To apply three different response intervals, you might design a process with three assignments, and apply the correct service level to each assignment. If the process changes, you need to update three assignments instead of one.

Complex exceptions that depend on combinations of factors become difficult to maintain and update. Consider a bank that offers different promotions that reduce or waive fees for customers who meet specific conditions.

- A new customer receives 100 commission-free trades for the first three months after opening an investment account.
- A customer receives a rebate on commissions as long as the daily balance in their investment account exceeds certain thresholds—but the rebate amount, balance threshold, and number of rebate tiers vary by account type and country.
- A customer who refers a friend to the bank receives 10 commission free trades per month for six months.

A rule that models these commission discounts according to account type, account balance, and country can become complex. This complexity may lead to configuration errors and dissatisfied customers.

In Pega applications, you model complex exceptions through **circumstancing**. With circumstancing, you create a variant of a rule—such as a decision or a service level—tailored to a specific circumstance. When an application uses a circumstanced rule, the system determines which rule variant best satisfies the need. Circumstancing allows you to customize the behavior of your application to address each exception condition you identify using a collection of targeted rules rather than one complex, difficult-to-maintain rule.

KNOWLEDGE CHECK



ANSWER How does circumstancing solve the problem of configuring exception behavior in an application?

Circumstancing allows you to describe exception behavior with a set of targeted rules rather than one complex rule. Each targeted rule configures behavior to address a specific exception.

How to circumstance rules

Circumstancing establishes a baseline for expected case behavior and adds variants to address exceptions to the behavior. The goal of circumstancing is to create a variant for each anticipated situation. Pega selects the appropriate variant, or circumstance, to use based on the details of the case.

When you circumstance a rule, you create a set of focused rules to address exceptions to case processing, rather than one all-encompassing rule. Since each rule focuses on a specific exception, application maintenance and updates are easier and can delegate to business users. Reusing the rules you create at the application or enterprise level is also easier.

How to circumstance a rule

To circumstance a rule, you start by creating a base rule to define the expected behavior. Pega uses the base rule unless a circumstanced version is more appropriate.

Consider a company with different response time obligations for elite and non-elite customers. The response time for non-elite customers is the expected behavior. In this situation, the response-time goal is 24 hours. So you create a base rule—in this case, a service level—to enforce the response time goal for non-elite customers.

Then, you identify any exceptions to the expected behavior. For each exception, you circumstance the base rule to address the difference from the expected behavior. For example, elite customers with silver status have a response time goal of 12 hours. You circumstance the base rule to enforce the response time goal for customers with silver status. You can then create another circumstance to address the goal time for customers with gold status, who have a service level response goal of six hours.

The diagram illustrates how a base rule is circumstanced. On the left, a 'WORKLIST' screen shows a table with three rows of 'Inquiry' tasks for 'Basic' status customers with due dates of 14, 18, and 22. On the right, a similar 'WORKLIST' screen shows six rows of 'Inquiry' tasks for 'Gold', 'Silver', and 'Basic' status customers with due dates of 6, 12, 14, 18, and 22. Above the right screen are two 'Assignment' cards: one for 'Gold' status with a due date of 6, and one for 'Silver' status with a due date of 12.

WORKLIST		
SUBJECT	CUSTOMER STATUS	DUE
Inquiry	Basic	14
Inquiry	Basic	18
Inquiry	Basic	22

WORKLIST		
SUBJECT	CUSTOMER STATUS	DUE
Inquiry	Gold	6
Inquiry	Silver	12
Inquiry	Basic	14
Inquiry	Basic	18
Inquiry	Basic	22

Assignment
Subject: Inquiry
Status : Gold
Due : 6

Assignment
Subject: Inquiry
Status : Silver
Due : 12

Types of circumstancing conditions

You can circumstance a rule according to the value of one or more conditions. You define a condition based on one variable, multiple variables, or the processing date, then apply the condition to a variant of the rule. When using the rule, the application evaluates the conditions defined on all the circumstanced variants. If one of the circumstancing conditions is satisfied, the application uses the corresponding rule variant. Otherwise, the application uses the base rule.

Pega supports the following types of circumstance conditions.

- **Single value** — the rule variant is effective whenever the value of a single property satisfies the circumstancing condition. You specify the property to evaluate a comparison value when circumstancing a rule. If the value of the property matches the specified value for a case, the application applies the circumstanced variant of the rule, rather than the base rule.
- **Multiple value** — the rule variant is effective whenever a combination of property values satisfies the circumstancing condition. Multiple value circumstances are based on a circumstance template and circumstance definition. The **circumstance template** defines the properties on which to circumstance a rule. The **circumstance definition** defines the combination of conditions in which a property uses a variant of a rule. You apply the circumstance template and circumstance definition to the rule variant. If the case matches a combination in the circumstance definition, the application uses the circumstanced variant of the rule, rather than the base rule.
- **Date property** — the rule variant is effective whenever the value of a date property satisfies the circumstancing condition. This condition can be either a single date or a range of dates. If the value of the property is later than the specifies date or falls within the range of dates, the application uses the circumstanced variant of the rule, rather than the base rule.
- **As-of date** — the rule variant is effective after a certain date, or during a range of dates. After the specified date or during the specified range, the application applies the circumstanced variant of the rule, rather than the base rule.

KNOWLEDGE CHECK



What type of circumstancing will a real estate firm use to display a disclosure statement that varies by both the status and location of the listing?

The real estate firm will use multiple value circumstancing to display the appropriate disclosure for each listing based on the circumstance template and definition.

Examples of circumstancing conditions

The following video provides examples of each type of circumstancing condition.

In Pega, you circumstance a variant of a rule using one of four types of circumstance conditions.

You use a single value circumstance to create a rule variant that is effective whenever the value of a single property satisfies the circumstancing condition. For example, many companies offer a quicker response to complaints or inquiries for customers with premium or elite status. In this situation, you circumstance a service level rule based upon the value of the property that records customer status. If the value is silver for a case, the application uses the circumstanced variant for customers with silver status, rather than the base rule.

You use a multiple value circumstance to create a rule variant that is effective whenever a combination of property values satisfies the circumstancing condition. For example, a bank waives account fees for customers who maintain an account balance of EUR10000. But for customers in Germany, the account balance must exceed EUR25000. In this situation, you use a circumstance template to define the properties on which to circumstance the rule. You then use a circumstance definition to specify the minimum balance and country for the exception — EUR25000 for customers in Germany.

You circumstance on a date property to create a rule variant that is effective whenever the value of a specified date property satisfies the circumstancing condition. For example, companies often update a business process in response to business conditions, but must process existing cases under the previous version of the process. A process goes into effect on January 1 to reflect a more rigorous review of loan applications. But cases created before this date are to be completed using the old process. In this situation, you circumstance the updated process based upon the value of the `pxCreateDate` property—the date a case was created. If the value of `pxCreateDate` is a date after January 1, the application uses the updated process instead of the original process.

You circumstance a rule using an as-of date to create a rule that is effective after a certain date, or during a range of dates. For example, during December, a retailer offers free shipping on any order over USD50. During the other months of the year, only orders over USD100 receive free shipping. In this situation, you apply a date range to the rule that calculates the shipping charge. If a customer places an order in December, then the application applies the USD50 threshold for free shipping. If the customer places an order in any other month, the application applies the USD100 threshold instead.

Each variant of a rule applies one type of circumstancing condition, though you can combine conditions to circumstance a rule.

Circumstancing a rule

To circumstance a rule, you first create a base rule and then create specialized versions of the rule. Each version is tailored to a specific exception in case behavior.

Follow these steps to circumstance a rule:

1. Open the base rule.
2. On the base rule, open the pull-down menu on the Save button and select **Specialize by circumstance**. The New Record form opens, with two circumstancing options: **Template** and **Property and Date**.

Service Level Agreement Record Configuration

Label* Identifier A short description or title for this record

CIRCUMSTANCE BY Template Property and Date
The conditions defined in the circumstance definition must evaluate to true for this record to be chosen at run-time.

Template Definition

3. On the New Record form, identify the type of circumstance. To circumstance on one variable, select **Property and Date**. To circumstance on more than one variable, select **Template**.

4. Specify the condition under which the rule is used. The following example shows a service level circumstance to run whenever the value of *.CustomerStatus* is "silver." The value must be entered within quotation marks.

Service Level Agreement Record Configuration

Label★ Identifier DisputeResolution not editable
A short description or title for this record

CIRCUMSTANCE BY Template Property and Date

Choose a property and/or a date property. All entered conditions must evaluate to true for this record to be chosen at run-time.

Property <input type="text" value=".CustomerStatus"/>	+	Value <input silver\""="" type="text" value="\"/>
--	---	--

Date property <input type="text"/>	Start Date <input style="width: 90%;" type="text"/>	End Date <input style="width: 90%;" type="text"/>
---------------------------------------	--	--

To circumstance by date, use the following table to configure the circumstancing condition to meet various business requirements.

Business requirement	Specify date property	Specify start date	Specify end date
Rule to be effective only if the value of the specified date property occurs within a date range	Yes	Yes	Yes
Rule to be effective only if the value of the specified date property occurs after a certain date	Yes	Yes	No
Rule to be effective only within a date range	No	Yes	Yes
Rule to be effective only after a certain date	No	Yes	No

To circumstance by more than one property, specify the circumstance definition and circumstance template rules that define the combination of conditions. For each circumstanced rule, you must provide a unique circumstance definition.

Decision Table Record Configuration

Label*

A short description or title for this record

Identifier

CommissionCalculation not editable

CIRCUMSTANCE BY Template Property and Date

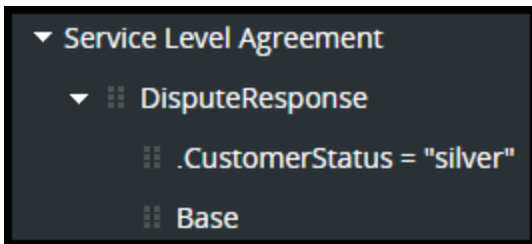
The conditions defined in the circumstance definition must evaluate to true for this record to be chosen at run-time.

Template

Definition

5. Click **Create and open** to open the rule form.
6. Customize rule behavior for the specified circumstance.

To view the circumstancing condition for a rule, locate the rule in the Application Explorer. Pega indicates a circumstanced rule with a collapse arrow. Clicking the arrow expands the rule entry to display the supported circumstances. In the following example, the *DisputeReponse* service level includes a circumstance used when the value of *.CustomerStatus* is silver.



You can also review the circumstancing condition for a rule by clicking the **Circumstanced** link in the rule header.

Edit Service Level Agreement: Dispute Response [Available, Circumstanced]

CIRCUMSTANCE BY PROPERTY

Property	.CustomerStatus	=	"silver"
----------	-----------------	---	----------

DESIGNING A USER INTERFACE

Configuring a user form

Introduction to configuring a user form

Good user interface (UI) design considers the impact of device size and how users interact with their devices. Consider an application with a screen resolution fixed to accommodate a tablet device. Users with tablets can easily use the application. Users accessing the application with laptops might find the screen resolution too small to read. Users with mobile phones might not be able to use the application. Effective UI accounts for all of these interactions.

In this lesson, you learn how to design responsive UI forms and model a user interface form.

After this lesson, you should be able to:

- Explain the role of a section in UI design
- Design responsive UI forms
- Create and configure a section record to model a user interface form

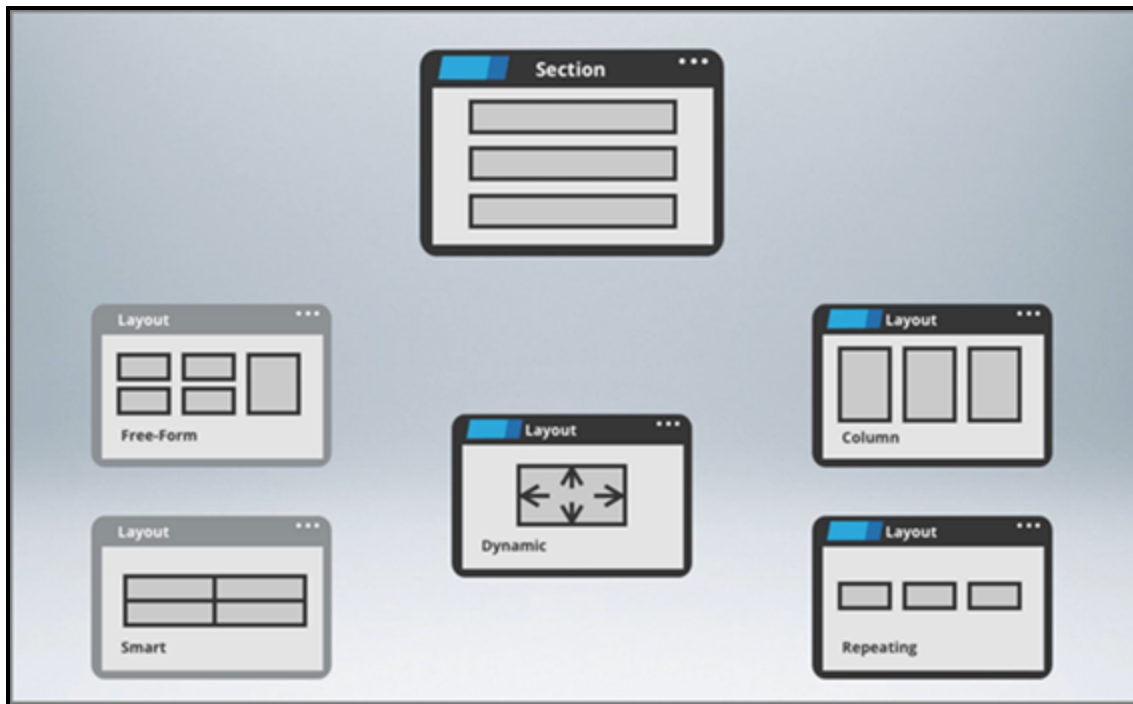
Section rules

Users interact with an application and perform tasks through user forms. A user form can be a data entry form in which users provide information for filing an insurance claim, a display of the legal terms and conditions users must accept before opening a bank account, or a list of bank account transactions during the past month.

In Pega, you build user forms with **sections**. Sections group information and functionality by context and purpose. Inside a section, you organize UI elements with **layouts**. Layouts contain rows and columns, defining a set of **cells**. A cell can be empty or contain various fields and controls.

When you create a view in the Case Designer, you are using sections to build the view. By default, the fields in the section are organized into a layout that contains a single column. When you add an embedded field such as a field group (list), the list properties are organized into their own section.

The following picture shows several types of layouts. Different layouts arrange UI elements in different fashions. A column layout arranges items in a set number of columns.



KNOWLEDGE CHECK



ANSWER

What is the relationship between a section and a user view?

A section is the rule used in Pega to configure the contents of a user view.

Sections and dynamic layouts

Sections use dynamic layouts that arrange items in a flexible form, automatically adjusting to screen size. Section rules generally use two types of layouts: dynamic layouts and repeating layouts.

- A **dynamic layout** is a general purpose layout used to organize a single set of fields. Dynamic layouts support responsive behavior to support use on multiple devices (computer, tablet, phone). As the screen size changes, the layout shifts its contents to wrap content on-screen.
- A **repeating layout** is used to organize lists, tables, and other repeating structures. Repeating layouts reference a page list or page group, either by referencing a clipboard property or a data page. As the screen size changes, elements display or are hidden to maximize the value of the displayed content.

KNOWLEDGE CHECK



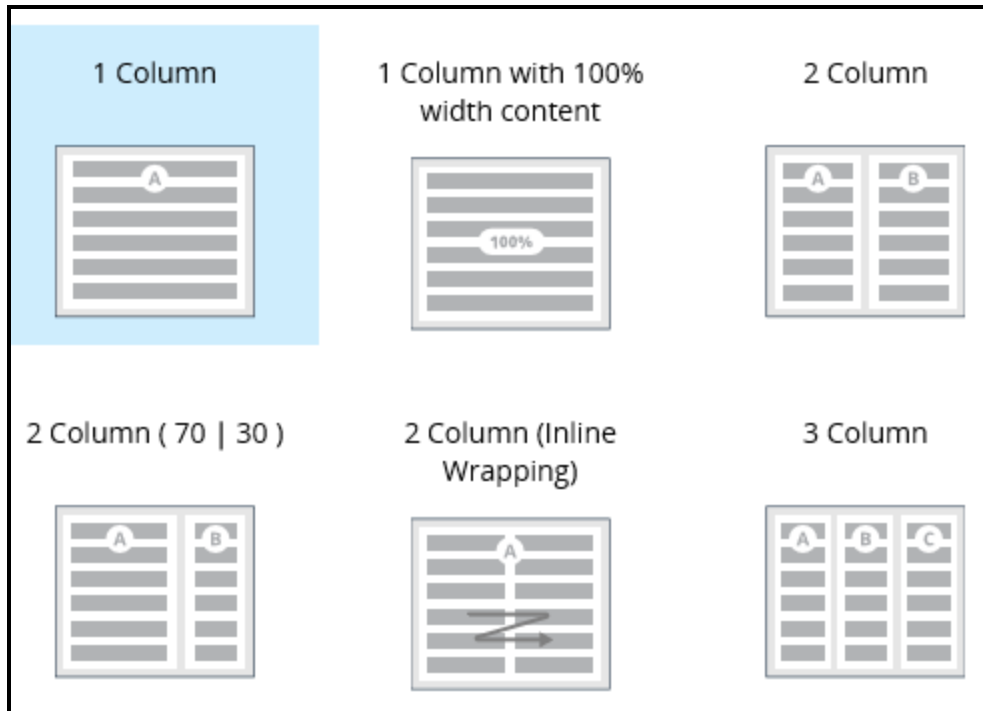
ANSWER

Describe the purpose of dynamic layouts.

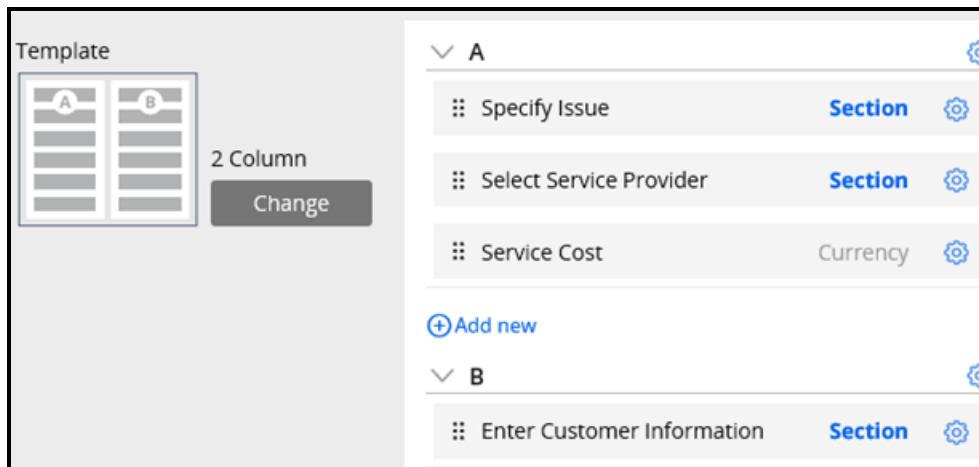
Dynamic layouts arrange items in a flexible form that automatically adjusts to screen size.

Design templates

You use **design templates** to configure section layouts. The templates provide a set of commonly used UI designs. Using templates makes it easy to add the UI elements to a section without having to manually configure the layouts. Dynamic layouts are often used in templates. The following image contains some examples of design templates.



Structurally, a section consists of one or more layouts and embedded sections. These sections enable you to logically group elements and reuse them in other layouts. The following image shows an example of a two-column template that contains embedded sections and a field.



KNOWLEDGE CHECK



What is one advantage of design templates?

Design templates allow you to apply a consistent structure to a form. You can change the structure by changing the template.

How to configure a section

The section is the fundamental UI rule in Pega applications. Section rules are created automatically when you create user views in the case life cycle. Sections are organized into layouts, which contain the fields that are presented to the user in the view.

When creating a user interface, a section can be configured through a user view using Pega. However, certain configuration needs can only be satisfied by accessing and modifying the section rule directly. For example, adding more than one layout can only be done on a section rule. And if the section is not configured from the case life cycle, you need to create the section manually.

In Designer Studio, you create sections from the Create menu, or by right-clicking the case type in the Application Explorer and selecting the User Interface category and the Section rule type. When you create a section, specify the following information:

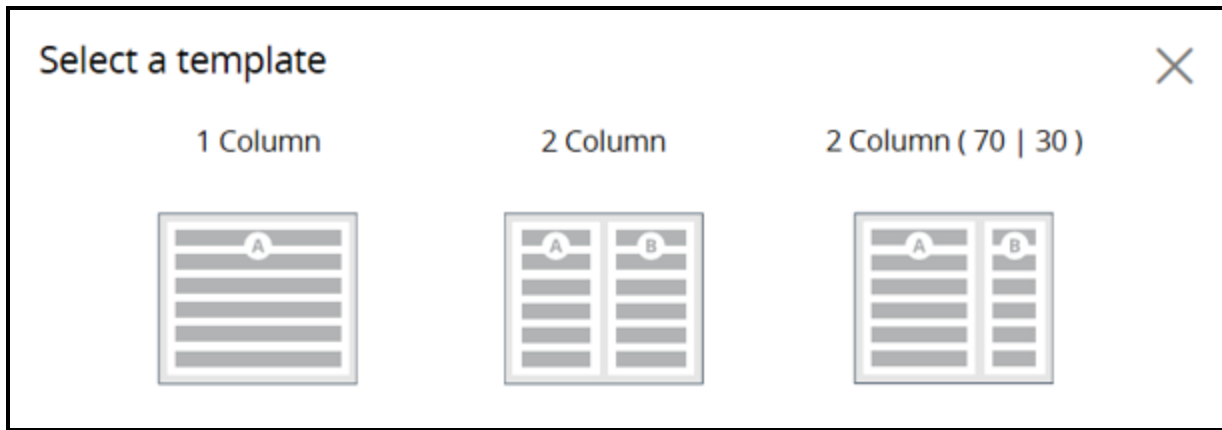
- **Label** – Used to identify the purpose of the section. The label is used to generate an identifier for the record.
- **Class** – The class to which the record is applied. The class determines the availability and re-usability of the section.
- **Ruleset and version** – The ruleset and version that contains the record. Pega defaults to the highest unlocked application ruleset when you create a record.

You can also select to configure the section using a design template. Use design templates to ensure that the application UI is organized consistently and to simplify the organization of the section and its maintenance. You can position the section contents by selecting a template and then arranging the contents within the columns defined by the template.

A screenshot of the 'Section Record Configuration' dialog box in Pega Designer Studio. The dialog is divided into several sections. At the top, there are fields for 'Label' (with a red asterisk) and 'Identifier'. Below these are 'Additional creation options', including a checked checkbox for 'Create Section using a Design Template'. The 'Context' section at the bottom has radio buttons for 'HR Apps', 'UI KIT', and 'PegaRULES'. There are also dropdown menus for 'Apply to', 'Add to ruleset', and a date field set to '01-01-60'.

As shown in the following example, you can apply a design template to organize fields into one column, two columns of equal width, or two columns split 70/30 (one column is 70 percent of the available width and the remaining column is 30 percent).

Note: Additional design templates are available but are not shown here.



In Pega Express, you can change the design template when running a case by clicking **Configure this view** in the upper right corner of the form. In Designer Studio, you can change the design template on the **Design** tab of the section rule.

KNOWLEDGE CHECK



What are the three ways a section can be configured?

You can configure a section through a user view using Pega Express. Certain section configurations can only be accomplished by accessing and modifying the section rule directly using Designer Studio. You can also configure the section using a design template.

For more information on design templates, see the support article [Designing sections by using design templates](#) on the PDN.

Note: After you convert the section, you do not have access to the design templates in the rule form.

Layouts

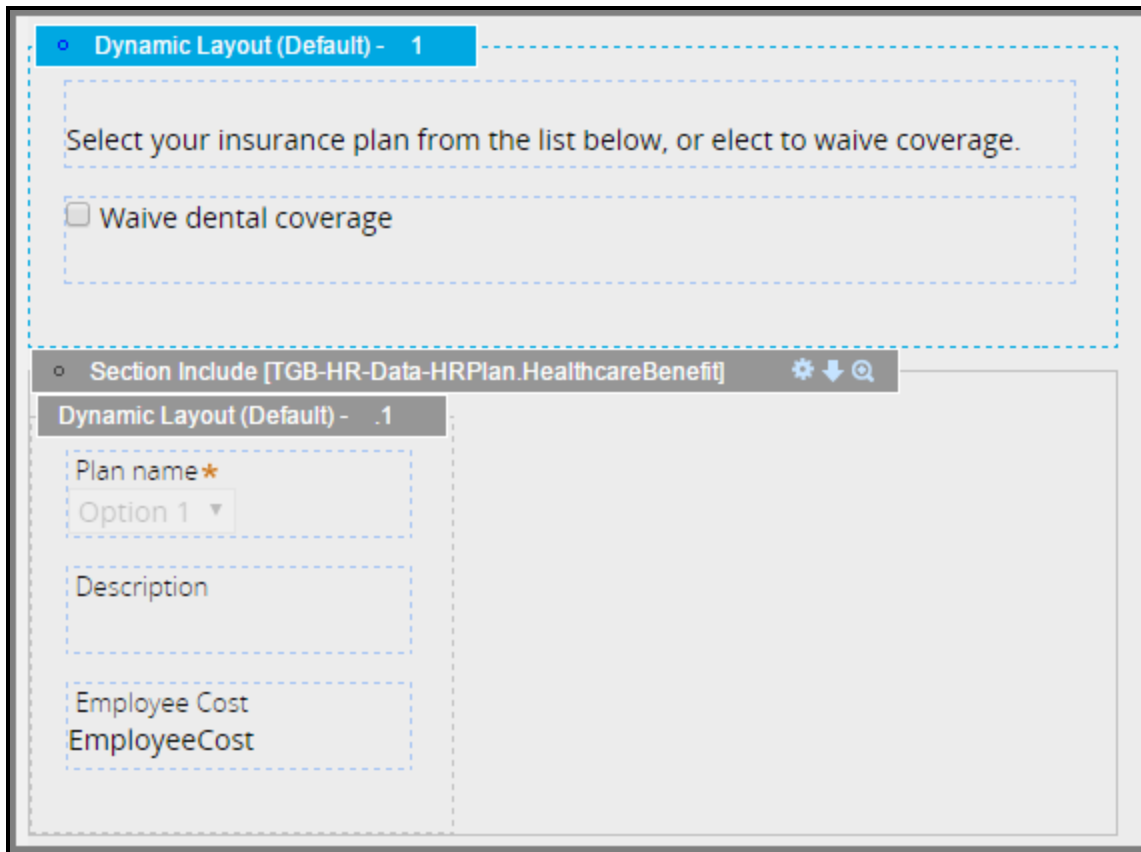
A layout is a container that governs the positioning of fields. Each design template consists of a single layout that arranges fields in a specific manner. If a design template does not provide the desired arrangement of fields, you can configure a section rule by adding layouts manually. Section rules generally use two types of layouts: dynamic layouts and repeating layouts.

Dynamic layouts

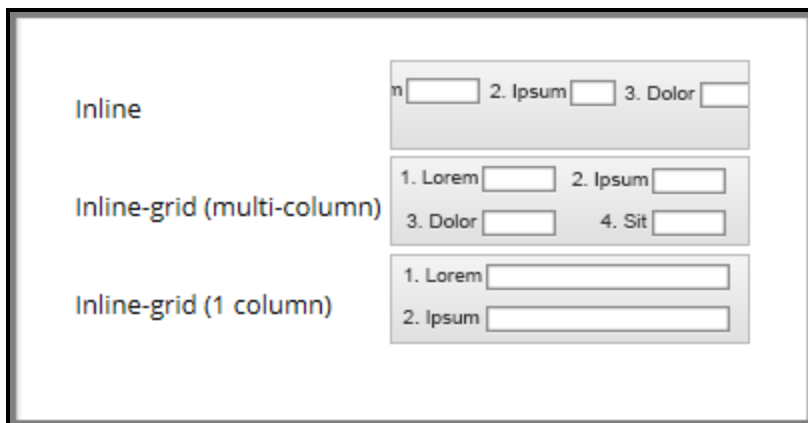
Dynamic layouts arrange items in a flexible form, automatically adjusting to screen size. When the layout width exceeds the width of the screen, a dynamic layout responds by wrapping controls to the next line. Wrapping eliminates the need for horizontal scrolling by users viewing the form on smaller screens, such as mobile devices. Many Pega design templates are built with dynamic layouts.

Note: You can see the dynamic layouts by converting the section to a full section editor.

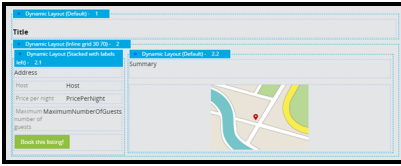
For example, in a section displaying healthcare coverage for employees, below the Dynamic Layout, a section defined in another class is included. This included section contains a dynamic layout that displays the coverage plan information.



In a dynamic layout, a single item is represented as tabular data. Item arrangement can be one of two types: inline or inline-grid. The inline arrangement displays items in a row, like words in a sentence. The inline-grid arrangement displays items in a multicolumn grid. The inline-grid with one column is equivalent to a stacked format.



Several formats of dynamic layout styles are available, including Default, Stacked, Inline, Inline-grid double, and Inline-grid triple. You can modify and create additional formats in the skin rule for the application; however, changing the format in the skin automatically affects all sections using that format. The following example demonstrates the use of multiple layouts in a single section to position content for display.



KNOWLEDGE CHECK



Describe the purpose of dynamic layouts.

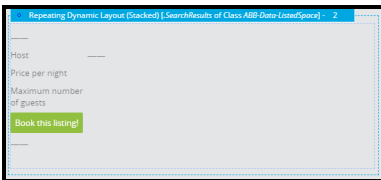
Dynamic layouts arrange items in a flexible form that automatically adjusts to screen size.

Repeating layouts

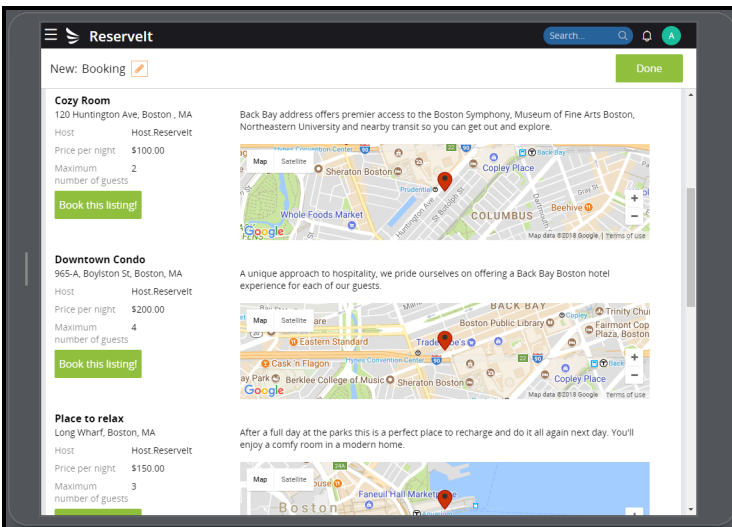
When you want to display a collection of data that belongs to a page list or a page group, you use a repeating layout.

In a repeating layout, several configurations are available including Grid layout, Tree layout, Tree Grid layout, Repeating dynamic layout, Column Repeat layout, and Tabbed Repeat layout. Data represented in a linear table is a repeating layout.

However, a repeating dynamic layout is useful when you want to present content in a nonlinear, more aesthetic format.

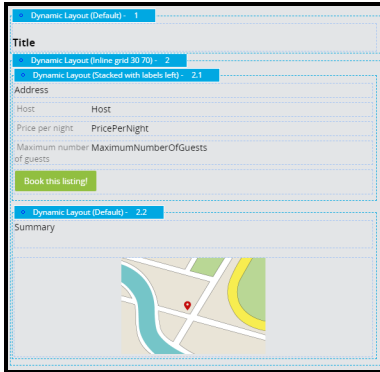


Repeating dynamic layouts help you create an interface for displaying data regardless of screen size.



Controls

Once you create the section, add controls and map the controls to specific data elements. For example, to book a listing, you can add a button. To set a True/False value, you can add a check box. To enter a number, you can add a field. To set a date or time, you you can add a calendar. For each control, you can configure the presentation and add an action set as appropriate.



How to configure responsive UI behavior

People expect easy, user-friendly access to applications on all their devices, such as laptops and mobile phones. To meet this expectation, Pega applications have system default **responsive breakpoints** that define changes in behavior on different devices. For example, when the width of a mobile phone screen crosses a breakpoint, the layout responds to provide a more useful display.

Responsive breakpoint configuration occurs in the application **skin rule**. A skin rule defines presentation formatting instructions for one or more UI forms. Skin rule formatting separates the content data from the presentation data. The default Pega skin rule provides breakpoints for managing transitions between monitor, tablet, and mobile phone displays.

A **Responsive UI** enables a layout to automatically adjust to rendering devices. Elements can move around, resize, or completely disappear based on screen resolution and size.

KNOWLEDGE CHECK



What are responsive breakpoints in Pega applications?

Responsive breakpoints are default settings built into the application skin rule. They manage the content transition of an application between monitor, tablet, and mobile phone displays.

Responsive dynamic layout

For a dynamic layout or repeating dynamic layout, use responsive breakpoints to change the layout format. For example, a dynamic layout using the inline grid triple format organizes fields into three columns. As the screen width decreases, the layout is displayed as two columns using the inline grid double format. When the screen width gets even smaller, the layout is displayed as a stacked layout, where each field is stacked on top of the next field in a single column. The following image illustrates how application elements in a dynamic layout may appear on various devices.



For more information about dynamic layouts, see the PDN article [Using dynamic layouts to create responsive user interfaces](#).

Responsive table configuration

Responsive tables — also known as grids — require additional configuration. Relying on breakpoints alone for tables can result in unimportant information having prominence over more important information.

For example, consider an auto parts ordering application. The application uses an inventory list for ordering replacement parts. The inventory list contains columns for the item name, description, part number, unit price, quantity, and line total. All columns are visible on a laptop. Without responsive configuration, the list looks much different on a mobile device. The only columns displaying are the item name, description, and part number. Users have no way to order the parts they need.

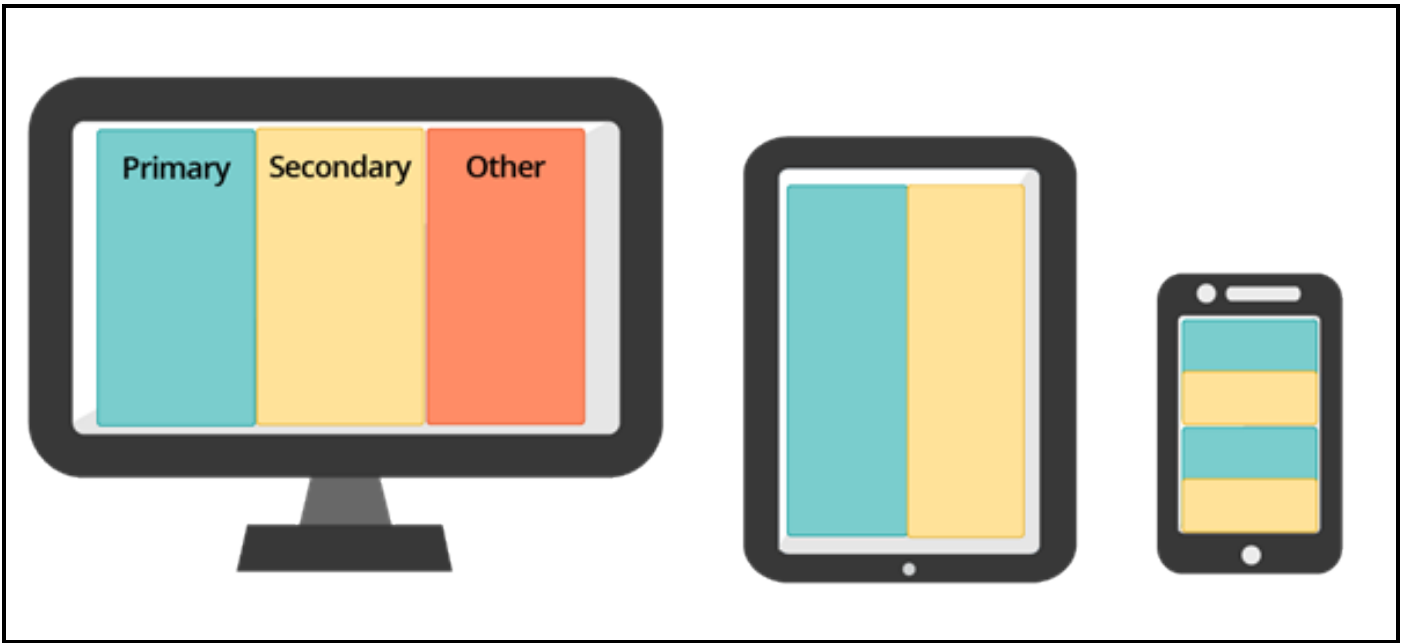
For a table, you use responsive behavior to hide less important columns to make room for the columns with vital information. Before you configure responsive behavior for a table, determine the importance settings for each column.

For example, consider how the auto parts inventory list displays with responsive configuration. The less important columns — name and description — are hidden when viewed on a tablet or mobile device. Users on a mobile device or tablet only see the part number, unit price, quantity, and line total columns. Users can submit orders on all devices.

You control this behavior by setting the importance of each column in the table. In Pega, importance settings are primary, secondary, and other.

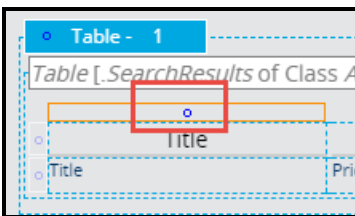
- Primary – The unique identifier for the row. Each table must have one primary column.
- Secondary – Important information with a significant effect on usability. Omitting information in a secondary column impacts the ability of the user to complete a task.
- Other – Information with minimal impact on usability.

By default, the application skin defines two responsive breakpoints for tables. When the display width is lower than the first responsive breakpoint, columns with importance set to **Other** are omitted from the table. When the display width is lower than the second responsive breakpoint, the **Primary** column displays in the heading for the list item and the remaining columns display as a list. The following image illustrates how a responsive table configuration may appear on various devices.



Note: Pega requires that the first (left-most) column be set to primary importance. The remaining columns can be set to secondary or other importance in any order.

To configure the importance of a column, first click the circle above the column header.



Then, to the right of the column, select the **Gear** icon to open the Column Properties panel. Finally, from the **Importance** drop-down, select the importance setting.

Column Properties

Width

Inline style

Enable sorting

Importance

- Primary
- Secondary
- Other

KNOWLEDGE CHECK



Why do tables need responsive configuration in addition to responsive breakpoints?

Relying on breakpoints alone for tables can result in unimportant columns having prominence over more important columns.

Creating dynamic content in user views

Introduction to creating dynamic content in user views

A simple and focused user interface (UI) is clear and intuitive for users. Users know exactly what to do right away. Building a dynamic UI is a key component of interactive simplicity. For example, you can control the display of information in user views. You can hide unneeded fields and reveal them when required. Dynamic behavior in UI enhances usability and encourages interactivity.

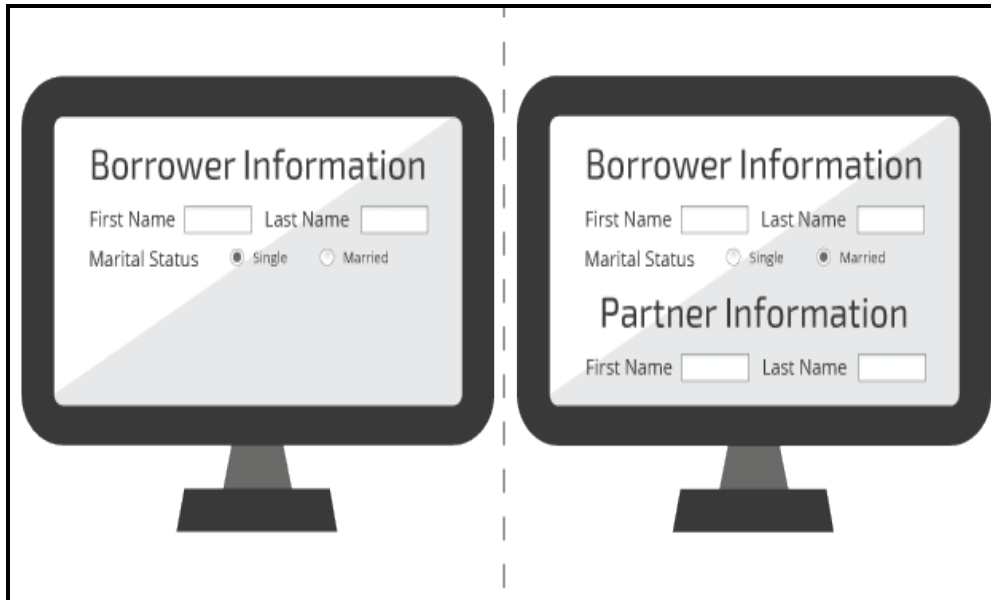
After this lesson, you should be able to:

- Explain the impact of dynamic UI behavior on user experience
- Identify configuration options for dynamic behavior of fields
- Describe how actions sets define UI behavior
- Configure action sets to perform actions in response to user-driven events

Dynamic UI behavior

In a dynamic user interface (UI), the UI content changes based on users interactions with the content. These changes reduce the UI to the fields essential for greatest efficiency.

For example, while submitting a loan request, a bank's customers must state marital status. If applicants select Married, the application displays user entry fields for the spouse or partner name. Single applicants skip to the next step of the loan process, saving time and effort. The following image illustrates two UIs. The UI on the left shows the Marital status in the default state of Single. The UI on the right shows the Marital status set to Married, with additional fields to enter partner information.



Using a dynamic UI has many benefits that lead to a more compelling and modern user experience. Dynamic UI benefits include the following:

- Real-time response to end-user behavior
- Robust functionality available for most user interactions
- Reduced visual clutter on the screen
- Fewer full page refreshes, resulting in improved UI responsiveness

Dynamic behavior enables application developers to hide portions of the UI until those portions are needed, and then expose them in response to some trigger, such as a property value falling within an allowed range, or when the user clicks a button.

Pega Platform automatically creates a section rule for each user view. You configure dynamic behavior in section rules. You can also create and edit sections manually to meet UI requirements. For example, configuring a pop-up window to display more information when users click a button.

Note: Dynamic UI design differs from responsive UI design. Dynamic UI relies on configuring what elements appear on the form. Responsive UI controls how form elements align and shift to match the display size.

KNOWLEDGE CHECK



How do application developers benefit from dynamic behavior?

Dynamic behavior enables application developers to hide portions of the UI until those portions are needed, and then expose them in response to some trigger, such as a property value falling within an allowed range, or when users click a button.

Event-action model

When designing a dynamic UI, you use an **event-action model**. Think of event and action as a cause-and-effect pair. For example, an online shopping cart that requires users' shipping and billing addresses. Users can select a box to indicate that their shipping and billing addresses are the same. The section for the billing address disappears when users select the box. In this case, selecting the check box is the event, and hiding the billing address section is the action.

Two types of events exist: **property-based events** and **user events**.

- Property-based events occur either when a data value changes or when a value meets specific criteria.
- A user event occurs when an end user takes some action on the page, such as selecting an option or clicking on a link.

The following image provides examples of events and actions.

Event = something happens	Action = something changes
Property-based event Order Total: <input type="text" value="£501.98"/>	Display message "Purchase orders limited to £500"
User action event Marital Status: <input type="radio"/> Single <input checked="" type="radio"/> Married	Display partner information section

Categorizing events into two types simplifies the event-action concept. In practice, these two event types often overlap. For example, when a user clicks a button (a user event), the action is to set a property to a value. That action then triggers a property change event.

KNOWLEDGE CHECK



How would you describe the event-action model?

Think of event and action as a cause-and-effect pair. Users perform events that trigger UI changes. The changes are the action.

How to configure field attributes for dynamic display

Controlling the fields displayed on screen removes less important and irrelevant elements. Pega Platform provides the ability to control how fields display. You can configure field attributes for dynamic display using **Visibility**, **Disable**, and **Required** settings. Setting options for all three attributes are similar. The following table defines the settings and their functions.

Setting	Definition	Attributes
Always	The UI element always displays	Visibility, Disable, Required
Condition (expression)	Uses a Boolean expression to determine visibility; visible when the expression returns true	Visibility, Disable, Required
Condition (when rule)	Uses a when rule to determine visibility; visible if the when rule returns true	Visibility, Disable, Required
If not blank	Visible if the value of that field is not blank	Visibility for cell properties
If not zero	Visible if the value of that field is zero	Visibility for cell properties
Never	The UI element is never displayed	Disable, Required

You use **visible when conditions** to hide or display data fields based on a value entered by users. In the following example, users select their marital status from a list in a form. Marital status options are single, married, and divorced. Depending on the value entered in the **Marital Status** field, additional fields may populate the form.

If the user selects **Single**, no additional fields are displayed.

Name	Marital Status
<input type="text" value="Linda Brown"/>	<input style="border-bottom: none; border-right: none; border-top: none; border-left: none;" type="text" value="Single"/> ▼

If the user selects **Married**, the **Date of Marriage** and **Name of Spouse** fields are displayed.

Name	Marital Status	Date Of Marriage	Name Of Spouse
Sarah Jones	Married ▼	4/30/2010	Michael Jones

If the user selects Divorced, the **Date of Divorce** field is displayed.

Name	Marital Status	Date Of Divorce
Sue Smith	Divorced ▼	3/25/2013

KNOWLEDGE CHECK



When do you use visible when conditions?

You use visible when conditions to hide or display data fields based on a value entered by users.

Additional Visibility and Disable options

There are two additional options available when you select a visible when condition.

Visibility	If not blank ▼
	<input type="checkbox"/> Reserve space when hidden <input type="checkbox"/> Run visibility condition on client

The **Reserve space when hidden** option keeps the space surrounding the control open. This prevents the UI elements on the screen from repositioning when the visible content is displayed.

The **Run visibility condition on client** option is displayed when you use the If not blank, If not zero, or Condition (expression) visibility options. When you select the **Run visibility condition on client** option, the clipboard page includes all the possible data it can display. The application uses the data on the page to refresh the section based on the visibility condition. If you do not select this option, the client must communicate with the server to refresh the section. If the hidden content is not likely to change during case processing, select **Run visibility condition on client**. This reduces the number of server trips and avoids page refreshes.

For Disable conditions, the option **Run disabled content on client** provides the same benefit as **Run visibility condition on client** for the Disable conditions.

KNOWLEDGE CHECK



What is the advantage of selecting **Run visibility condition on client** and **Run disabled content condition on client**?

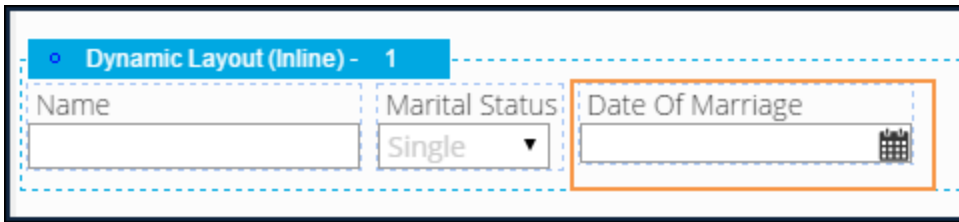
Configuring Visibility and Disable options to run on the client reduces the number of server trips and avoids page refreshes.

Configuring Visible and Disable conditions on a UI element

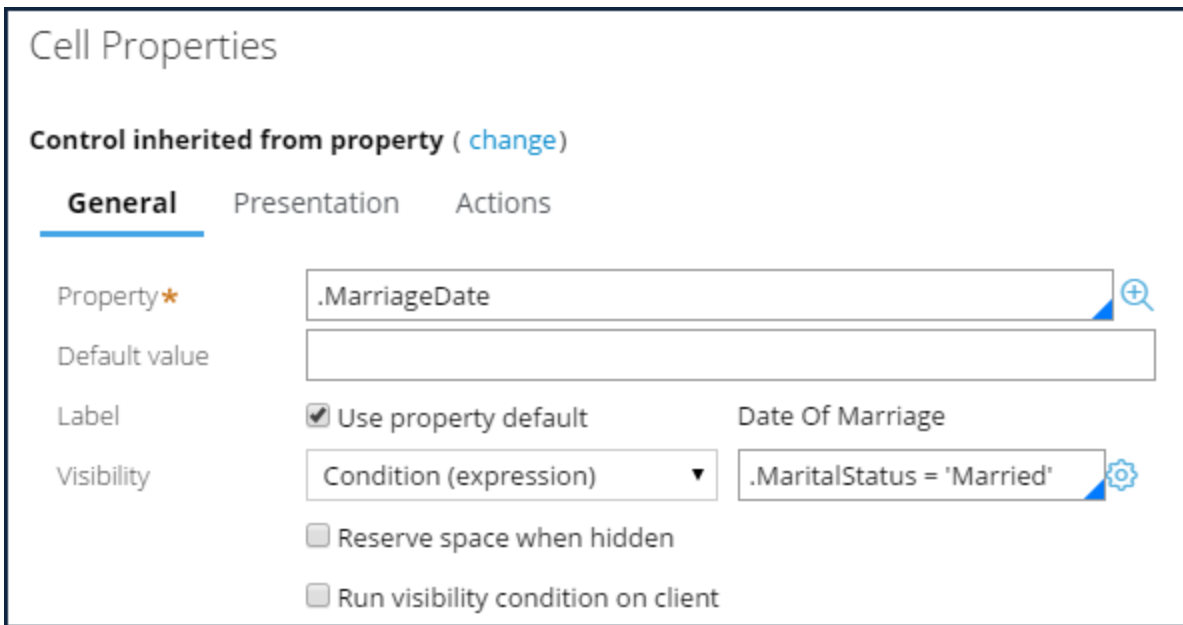
Before you begin configuring dynamic attributes for UI elements, identify the UI element target that you want to dynamically show, hide, or disable. Then, decide at which level (section, layout, or field) to apply the condition.

Configuring a Visible When condition on a UI element

In the following example, you want to display the **Date Of Marriage** field only when users select **Married** as their marital status. You configure the cell containing the field.



Open the configuration panel for the cell containing the marriage date property. Click the **Visibility** drop-down and select **Condition (expression)** to control the visibility of the marriage date property. Configure the visibility expression for the marriage date property so that the expression value is driven by the marital status value. Select **Run visibility condition on client** and then submit the configuration and save the change.



Configuring a Disable condition on a UI element

In the following example, you want to disable a field based on user selections. If users select **Email** as their **Preferred Contact**, then the **Mobile contact number** field becomes disabled. If users select **Text** as their **Preferred Contact**, then the **Email** field becomes disabled.

Perferred Contact

Email

Text

Email

Mobile contact number

This configuration uses the Condition (expression) option. The configured expression for each field is a value on the *perferredcontact* property. The following image shows how to configure the Condition expression to disable the **Mobile contact number** field when users select Email.

Disable Condition (expression) ▼ .PreferredContact='Email' ⚙️

Run disabled condition on client

The following image shows how to configure the Condition expression to disable the **Email** field when users select Text.

Disable Condition (expression) ▼ .PreferredContact = 'Text' ⚙️

Run disabled condition on client

KNOWLEDGE CHECK



What is your first consideration when configuring field attributes for dynamic display?

First identify the UI element target that you want to dynamically show and hide, enable and disable, or make required before deciding at which level (section, layout, or field) to apply the visible when condition.

Action sets

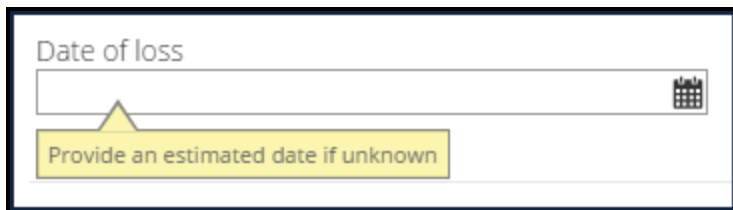
In Pega Platform, you use **action sets** to configure dynamic UI changes based on the event-action model.

An action set consists of an event, an action, and (optionally) conditions.

- **Event** – A trigger performed by users, such as clicking a button, hovering a pointer over a field, or entering a value in a field.
- **Action** – A response performed by the system as a result of the user event. For example, when users click a button, the application creates a case.
- **Conditions** – Restrictions such as when rules, which can be applied to an event and action combination. For example, you can configure conditions so that hovering over a field displays a smart tip message only if the field contains a property value.

Each action set requires at least one event and one action. For example, in an insurance claim application, you want a text pop-up to display when users hover their pointer over the **Date of loss** field explaining the purpose of that field.

The following image shows a smart tip pop-up on a **Date of loss** field.



You can also create multiple action sets for a single control or layout. Continuing the date of loss example, depending on how long ago the loss occurred, you also want to change the expected response time and a few date-related property values. When users change the value of the date field, the change event triggers a data transform that sets the date-related property values.

You can define action sets for a single control or an entire layout. In most cases, you define action sets on controls. You configure action sets on the **Actions** tab of the control's Cell Properties form. For layouts, configure action sets on the **Actions** tab of the Layout Properties form.

Note: One or more events can trigger an action and an event can trigger more than one set of actions. When configuring action sets, avoid configuring conflicting behavior for an event.

The following table shows a few examples of action sets.

Event	Action
Click a control such as a button, link, or icon	Opens a new window
Double-click a row in the grid	Opens the row in edit mode
Right-click the entire grid	Shows a menu
Press the Escape key on the	Closes the assignment and returns to the home page

Event	Action
keyboard	
Select a value from the state drop-down	Updates the list of counties
Select a check box	Unmasks the password
Enter a value in the quantity field	Calculates the total

KNOWLEDGE CHECK

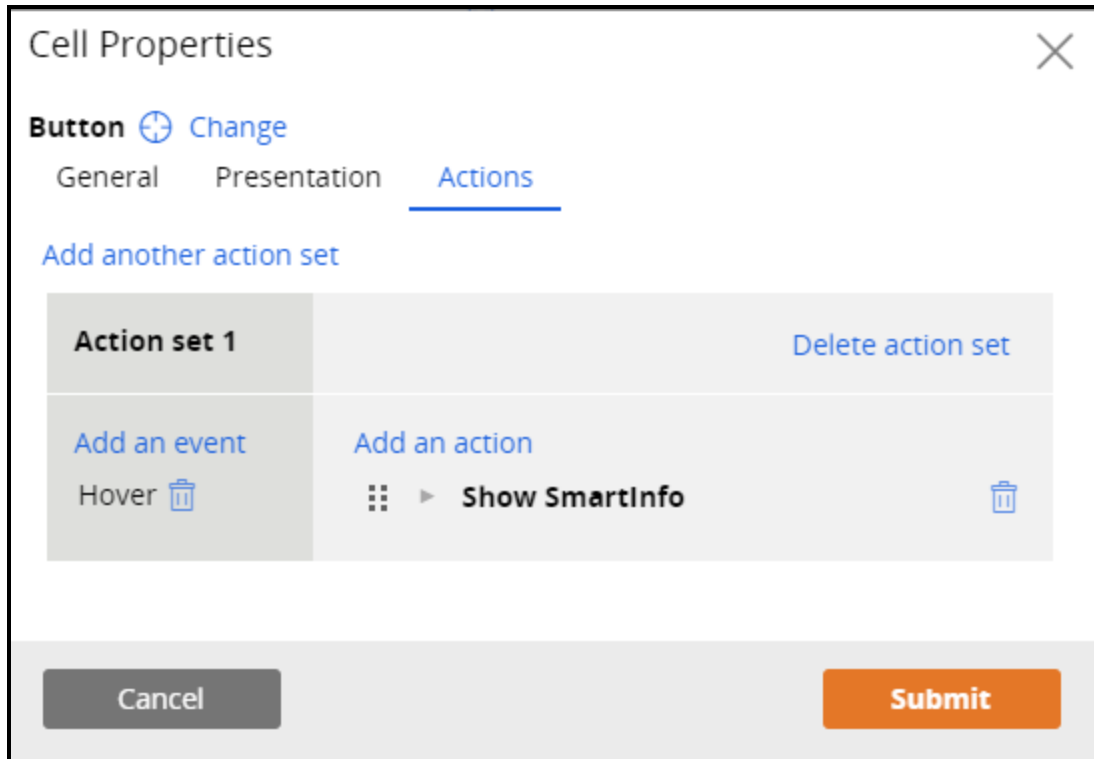


What is the purpose of an action set?

You use an action set to configure dynamic UI changes such as displaying a menu when a user clicks a button.

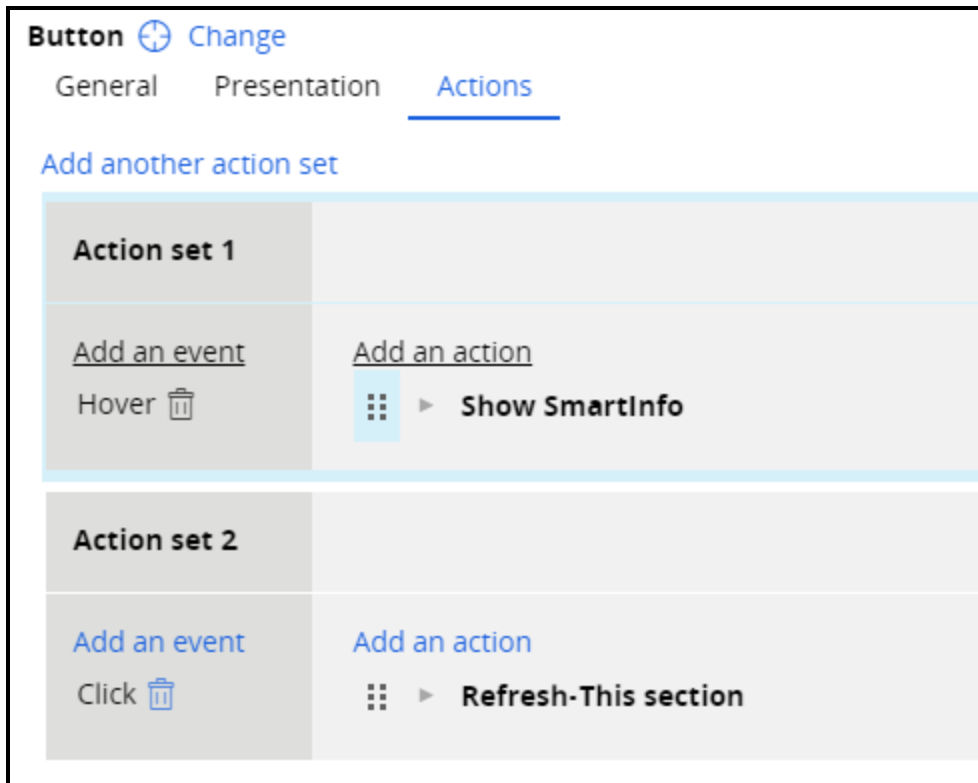
How to configure an action set for a field

You can configure an action set for common UI controls, such as fields, buttons, and drop boxes. If a control has an Actions tab, then you can configure an action set for the control. In the following image, a button control includes an action set. When users hover their pointers over the button, a SmartInfo pop-up is displayed.



Before you begin, identify the UI control and which event and action to configure. Most events involve a mouse, cursor, or keyboard. When selecting events, consider how users will interact with the application. For example, users interacting through a mobile device cannot trigger a hover event.

A control can have more than one action set. Continuing the button control example, you can add an action that refreshes the form when users click the button, as illustrated in the following image.



When selecting an action, consider the desired result. You can configure an action to show users relevant information or enter data in a field. You can also configure an action to start a new process.

KNOWLEDGE CHECK



What are the main considerations when preparing to configure an action set?

The UI control, and which event and action to configure to the control.

Action set configuration

Identify and locate or create the section rule that corresponds to the user view you want to configure.

To create an action set, open the **Cell properties** window, select the **Actions** tab, and click **Create an action set**.

Click **Add an event** to add one or more events to the action set, such as hovering, clicking, or pressing a specific key.

Click **Add an action** to pair one or more actions to the event or events selected. For each action, you configure the necessary parameters. You can also apply a when condition to determine when to perform the action. For example, only certain service items in a list may need more information from users. You can use a when condition to display a more information pop-up for the types of service requiring additional information.

You use the **Applicability** drop-down list to configure whether the action set processes if the control is displayed in read-only mode, editable mode, or both.

KNOWLEDGE CHECK



How do you configure an action set to occur when the control is in read-only mode?

You use the Applicability drop-down list to configure whether the action set is processed if the control is displayed in read-only mode, editable mode, or both.

For more information on action sets, see the Help topic [Action sets](#).

DESIGNING BUSINESS REPORTS

Creating business reports

Introduction to creating reports

Well-organized and comprehensive reports help users gain insight into the effectiveness of an application. In this lesson, you will learn how to use the Pega reporting features to assess application performance against requirements.

After this lesson, you should be able to:

- Describe the role of reports in Pega applications
- Explain how columns and filters affect the results of reports
- Differentiate between business data and process data in reports
- Describe the types of reports for Pega applications
- Describe the symbolic options for report filtering
- Create a report to query and present data

The role of reports

Pega reporting capabilities allow you to create reports that provide real-time information. Real-time information is important for two reasons. First, business analysts and work managers use reports to assess the performance of the application. And second, report information allows users to review or select items from a list or table while working in an assignment.

When designing reports, knowing what information the user needs and how it will be used is important. Business analysts can provide you with requirements that you use to create the required reports. For example, business analysts may want a report that shows the number and dates of resolved cases so managers can check process performance.

In another example, business analysts may need a report that contains a list of customers and their purchase orders. So, when a customer calls a customer service representative (CSR) to file a complaint, the CSR can view the customer's previous orders. To get this information, you create a report that populates a grid on the user form to display the purchase history for that customer.

Once you have design requirements, you use a **report definition** to build the report.

Report definitions

Report definitions retrieve records from a database. You use the report definition to specify the data from each record that you want to include in the report. The report definition retrieves the data from a database and returns the results in a table of columns and rows. The rows represent records retrieved from the database. The columns contain the data values in each record that you want users to see. In the following example, a report definition returns the case ID, employee name, employee hire date, and office location for onboarding cases.

Case ID	Employee	Hire Date	Location
O-101	James Martin	2/21/16	Atlanta
O-104	Anne Walker	2/4/16	Boston
O-746	Julia Phagan	1/12/16	Atlanta
O-983	William Kirk	9/8/16	Atlanta
O-171	Leonard Kelley	9/5/16	Boston
O-623	Kate Picardo	2/25/16	Atlanta
O-421	Robert Wang	2/1/16	Boston

Report columns

Report columns define the report's contents. Each column corresponds to a single data element. The value in the column can be a value property such as a case ID, last modified date, or work status. You can format the data value in various ways, such as text, currency, or date. For example, you can format currency properties to include a currency symbol.

You cannot display a page in a column. For instance, if you have an employee data object, you can return specific properties on the page, but not an entire page.

The following example demonstrates how you would design report definition to support a report request.

For example, a human resources application processes cases for onboarding newly hired employees. The organization has three office locations. The facilities manager wants a report to monitor new hires. The manager wants to see information about new employees so that office space can be prepared for them. You design a report in the onboarding class that has four columns. Each column includes a property — case ID, new hire name, date of hire, office location, and salary. When the manager generates the report, the report populates a list of rows with values for each column.

Defining columns in report definitions is critical to designing reports users need.

Functions

You can use **functions** in columns to make reports more useful. Functions allow you to calculate results derived from data in the database. For example, every new hire is evaluated 30, 60, and 90 days from the start date. A manager wants to see the number of days remaining until each evaluation. The function calculates the difference between the new hire date and the evaluation date.

Pega provides many standard functions you can use without having to create or customize functions. The available functions appear in a drop-down list when you open the function option in the report definition.

Report filters

By default, report queries return all the records that contain data from all the columns. You may want to only show records that are relevant to your design requirement. For example, your onboarding application collects information on all new hires. The facilities manager at each location needs a report that shows when the new hire needs a work space. You use report filters to show only the records your users need. In this case, the user needs new-hire start dates in the coming month.

A **report filter** compares a data value in the record against a defined condition. If the comparison result is true, the report includes the record. If the comparison fails the filter tests, the record is not included.

Assume you work for a company with two locations, Atlanta and Boston. You want to create a report of onboarding cases only for the Atlanta office location. You create a filter in the report that tests whether the office location for each onboarding case is Atlanta. When you run the report, the filter returns only cases for the Atlanta office. If the office is in Boston, that office is excluded.

Case ID	Employee	Hire Date	Location
O-101	James Martin	2/21/16	Atlanta
O-104	Anne Walker	2/4/16	Boston
O-746	Julia Phagan	1/12/16	Atlanta
O-983	William Kirk	9/8/16	Atlanta
O-171	Leonard Kelley	9/5/16	Boston
O-623	Kate Picardo	2/25/16	Atlanta
O-421	Robert Wang	2/1/16	Boston



Case ID	Employee	Hire Date	Location
O-983	William Kirk	9/8/16	Atlanta
O-101	James Martin	2/21/16	Atlanta
O-746	Julia Phagan	1/12/16	Atlanta
O-623	Kate Picardo	2/25/16	Atlanta

In the previous example, you use a filter to determine an office location. To create the filter, you define a filter condition in the report definition. A filtering condition is a logical expression that determines whether a record is included in the report.

The comparison can be an explicit value or the value of a property. For instance, if you want to create a report that returns open orders for a customer, you can use the `.CustomerName` property as the comparison in the filter condition. The returned records show the open orders for each customer. In the previous example, the filter condition uses a comparison that states "office location equals Atlanta." Therefore, only records that contain the data value Atlanta are included.

You can also use more complex conditions such as testing values that are greater than a specified threshold, like a date. When you use date or date time column data in your filter, you can select time periods using **symbolic dates**. Symbolic dates let you select time periods or dates without having to build functions. For example, you may want to filter all cases created in the previous month. You can select the Previous Month symbolic option rather than write a function to define the period.

Sometimes you may need to create a more complex filter to capture multiple filtering conditions. You can use multiple filters by adding AND/OR conditions. For example, assume you want to filter out cases with a status of Pending and for the manager Anne Walker. You create two filter conditions. One filter states the cases equal status of Pending. The other filter states that the manager equals Anne Walker. Use an AND condition so a record must pass both filters in order to be included in the report.

Business and process reports

To reduce costs, business applications often target performance gains in time and efficiency. Poor performance, such as long times to complete tasks, may show a poor application design. You need ways of understanding how complex processes are functioning. For example, you will learn where bottlenecks are, where there are opportunities to improve response time, and what emerging trends need attention.

A business report that provides relevant information can show what is happening, what has happened over a period of time, and how what is happening matches or differs from your plan.

Business reports and process reports

There are two types of metrics associated with report data: business metrics and process metrics.

Business metrics represent the data you define for an application. For example, business metrics are the number of orders for a certain item, or how many cancellations there are of a certain type of order.

Pega defines and tracks **process metrics**. For example, process metrics include how long it takes to complete an assignment, how often a path is followed in a flow, or the number of Service Level Agreement (SLA) violations.

Business reports

An organization can design business reports that describe and measure work. Organizations can use these business metrics to make informed decisions about improving its business performance. The application collects data and stores it in a database. The system retrieves the information when users generate a report.

The following table gives examples of business report information and how the information can be used in business decisions.

What is being measured?	What is the data?	What is the business decision?
Average profit margin for all automobile sales last year	The average margin is below the target percentage.	The sales department decides to train its sales staff on promoting cars and options that have the highest margins.
Number of auto loans made in a month as compared to personal loans for the same period	The number of personal loans is significantly lower than the number of auto loans.	The goal is to have the numbers approximately equal. The marketing department increases marketing resources for personal loans.

What is being measured?	What is the data?	What is the business decision?
Number of office desks shipped each week for the past month and how many are now in inventory	The number of orders shows an upward trend.	As a result, inventory levels are unacceptably low. The purchasing department decides to restock more desks on a weekly basis.

Process reports

In Pega applications, process reports track statistics on how work is performed. Unlike business metrics, the application automatically defines and generates process data. Process metrics operate on Pega Platform parameters. Process reports focus on process metrics that are unique to your case. Having this information enables business analysts and business managers to discover issues that may affect processing performance.

The following table gives examples of process report information and how the information can be used in process design decisions.

What is being measured?	What is the data?	What is the process design decision?
Open loan application cases exceeding the standard three-day service level deadline	Most of the open cases are for loan amounts greater than 300,000 USD.	Loan amounts that exceed this amount must go through an additional review step, which accounts for the delay. The department manager decides to increase the service level deadline for loans exceeding 300,000 USD from 3 to 4 days.
Average duration of assignments by type and action	This report identifies which user actions take the longest to complete.	Spend time on improving the efficiency of those assignments taking the most amount of time to complete.

KNOWLEDGE CHECK



A sales manager runs weekly reports on how long it takes to prepare a car for customer delivery. Which type of report metrics apply to this report?

Business metrics, because how long it takes to prepare a car for customer delivery is not a metric based on a Pega Platform parameter.

The Report Browser

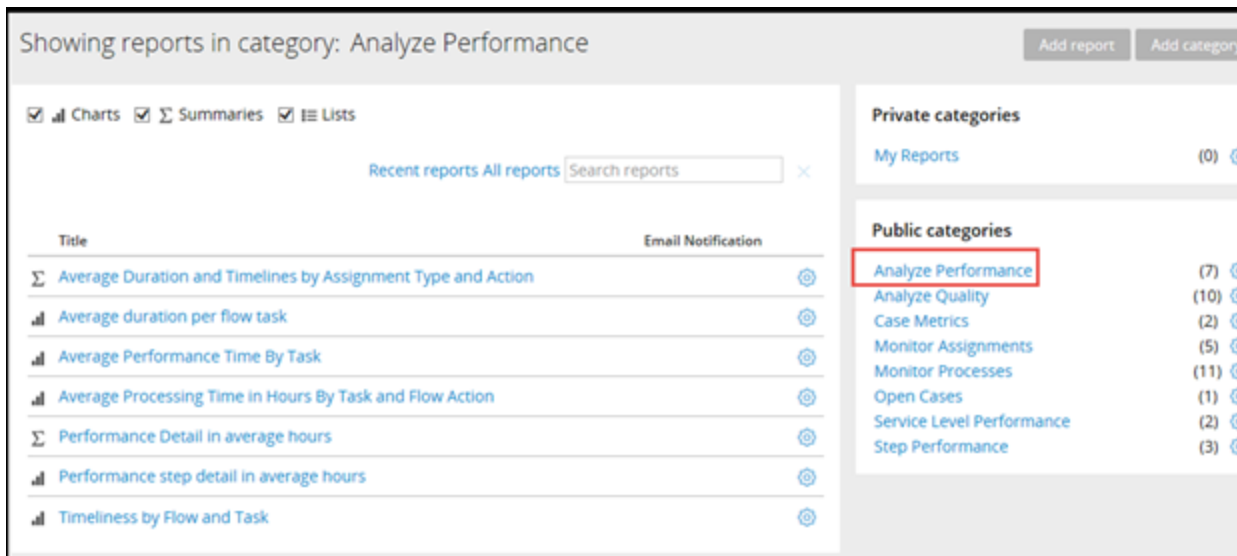
Work managers use the **Report Browser** to search for, organize, schedule, and run reports.

You use the Report Browser to review the library of available reports. Reports are grouped into public and private categories.

- The public category group include standard process reports provided by Pega. Public category reports are available to all managers in an application.
- The private category group includes reports that are created and saved for individual work managers. Managers can share their reports with other managers by putting them into the public categories group.

The Report Browser organizes the report categories into Private categories lists and Public categories lists. The lists appear on the right side of the Report Browser.

When you select a category from a category list, the available reports within the report category appear in a list on the left side of the Report Browser. The following screenshot shows the list of standard reports when you select the Analyze Performance category in the Public categories list.



Standard Pega reports

Standard Pega process reports are grouped into eight categories.

Report category	Information the reports provide
Analyze Performance	Resolved cases in an application at the level of each individual step, or actions, within a business process. The reports analyze the completed work to determine whether business processes are efficient and effective. For example, there is a report that tracks processing time in hours by task and action.
Analyze Quality	Resolved cases in an application. Similar to reports that analyze

Report category	Information the reports provide
	performance, quality reports analyze completed work to determine if business processes are efficient and effective. For example, there is a quality report that measures the average elapsed time per status.
Case Metrics	The number of cases created each day for the last seven days and the time per stage for resolved cases.
Monitor Assignments	Assignments for open (or unresolved) cases in an application. The reports tracks the work based on the user to whom the case is assigned. For example, there is a report that measures time lines by task.
Monitor Processes	Assignments for open (or unresolved) cases. The reports focus on the work and not individual users. For example, there is a report that measures throughput in the past week by work type.
Open Cases	Case-level SLA status for open cases. This report focuses on the timeliness of a case from the time a case is created to the time the case is resolved.
Service Level Performance	Assignment SLA status grouped by assignment or by operator.
Step Performance	Assignment-level SLA status grouped by assignment, and an historical view of the time it took an operator to complete a step.




Viewing reports in lists, summary layouts, or charts





When you select a report in a category, the report opens in the Report Viewer. The data can be displayed in a list, in a summary format, or in a chart.

- List reports display a list of cases whose data is organized into columns and rows.
- Summary reports aggregate case data into categories by use of a summarizing function, such as counting the number of results in a particular column or averaging the values in the column.
- Add charts to summary reports to help end users visualize report data, and allow users to “drill” down into the report data for greater insight.

As shown in the following screenshot, check boxes at the top of the report list let you filter the list based on how the reports are presented.

Showing reports in category: Monitor Processes

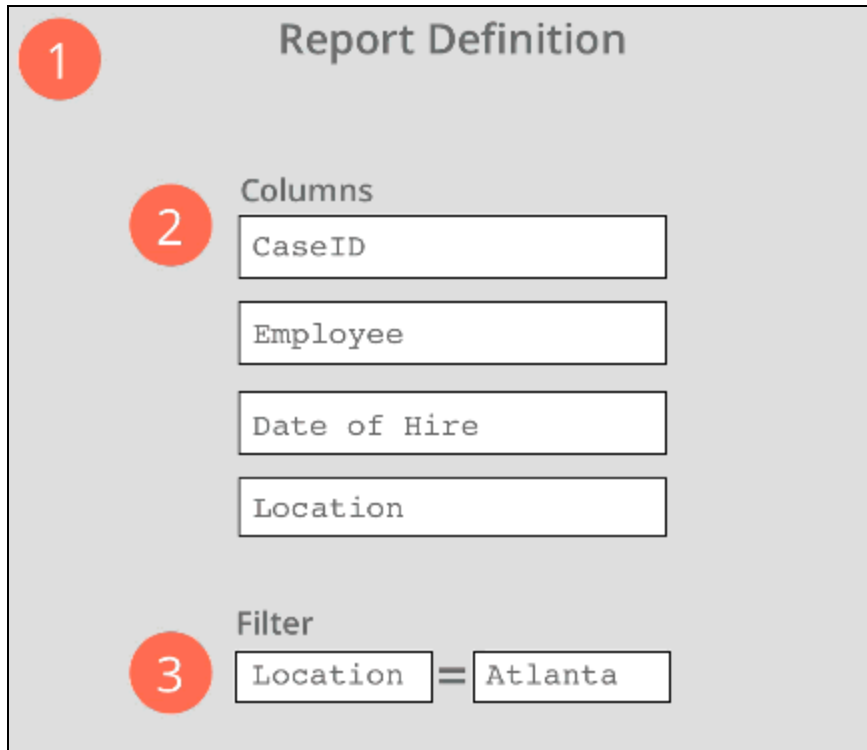
 Charts  Summaries  Lists [Recent reports](#) [All reports](#)

Title
 Effort by organization unit
 List of processes entered by operator
 Process average elapsed time per status
 Progress in days by organization

How to create a report

Creating a report for your application is simple:

First, create the report definition rule. Second, add columns in the report definition form. Third, add a filter to limit the records you want to display. The resulting report may also be made available to users in the Report Browser.



The screenshot shows a 'Report Definition' form with three numbered steps:

- 1** Report Definition
- 2** Columns: CaseID, Employee, Date of Hire, Location
- 3** Filter: Location = Atlanta

The following steps explain the process for creating a report using the Case Manager or Designer Studio.

Create a report definition rule

Create the report definition rule in the same class that contains the records on which you want to report. Each class in a Pega Platform application maps to a database table that contains the records for instances of the class. If you create the report in the wrong class, you do not get the records. Each case type in an application belongs to the case layer of the application, which maps to a common database table. Reports created in the case layer return data for any of the case types in the application.

In Pega Express, you create reports from the Case Manager portal. To create a report definition in the Case Manager portal, open the **Reports** tab and click **Add report**. Use the Case type drop-down list to select the class of the report, then select the report type: **List**, **Summary**, or **Chart**. To create the report, click **Submit**.

The 'Create New Report' dialog box contains two dropdown menus. The first, 'Case type', is set to 'HR Apps case layer'. The second, 'Report type', is set to 'List'. At the bottom, there are 'Cancel' and 'Submit' buttons.

In Designer Studio, right-click the class in Application Explorer, then select **Create > Reports > Report Definition**. Using the Create Report form, specify the class and name of the report. Give the report a name that clearly describes the purpose of the report. For example, use **Monthly New Employee Space Allocation**, not **Facilities**. This report name enables users to find and identify the report when they search in the Report Browser.

To create the report, click **Create and open**. The system displays the Edit Report Definition form as shown in the following example. The **Query** tab contains sections for specifying columns, creating filters, and creating summary reports.

The 'Edit Report Definition' form for 'Monthly New Employee Space Allocation' is shown. It has a top navigation bar with tabs: Query (selected), Chart, Report Viewer, Data Access, Parameters, Pages & Classes, and History. Below the tabs are two main sections: 'Edit columns' and 'Edit filters'.

Edit columns section: A table with columns: Column source, Column name, Summarize, Sort type, and Sort order. One row is visible with 'Column 1' in the name field, '<blank>' in Summarize and Sort type, and empty fields for Sort order. An 'Add column' button is below.

Edit filters section: A text input field contains 'A'. Below it is a table with columns: Condition, Caption, Column source, Relationship, and Value. One row is visible with 'A' in the Condition field, '<blank>' in Relationship, and a 'Select values' button in the Value field. An 'Add filter' button is below.

Note: When creating a report in Pega Express, you specify the name when you save the report, rather than when you create the report.

Add columns

Columns define the contents in your report. Each column corresponds to a single data element.

In the Case Manager portal, search for an item in the **Data Explorer** on the left and drag it to the appropriate column heading in the report at the lower right. The report data updates immediately. For example, search for **Current stage** and drag it to a location in the column heading.

Note: The data elements you use in your report can significantly impact the performance of your application. Therefore, Data Explorer only shows you optimized data elements. Designer Studio provides access to more data elements.

Current stage	Entered on
	12/5/17
	12/4/17
	12/5/17
	12/5/17

Hover your mouse cursor to the right of any column heading, A down arrow shows on the far right of that column. Select the arrow to open a menu that displays editing options. . For example, you can choose to **Sort** a column, **Delete** a column, a **Summarize this column**, and other actions.

Heading...	
Format...	
Add filter...	
Width...	
Edit...	
Delete	
Sort	Lowest to highest
Summarize this column...	Highest to lowest

In Designer Studio, in the **Edit columns** section of the report definition, click **+ Add column** to add a row to identify each column you want to include in the report. In each row, select the data element used as the **Column source** and enter a descriptive **Column name** to display when you run the report. The report retrieves values from the database for each of these data elements. You may also use functions if you want to calculate values not found in the database. Order the columns in the order in which you want them to appear in the report.

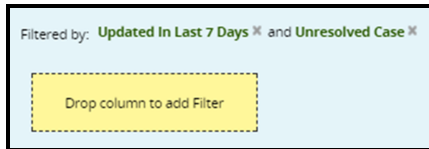
Edit columns	
Column source	Column name
.pyID	Case ID
.Employee	Employee
.DateHire	Date of Hire
.location	Location

When you generate the report, the results display values for each of the column data elements you defined.

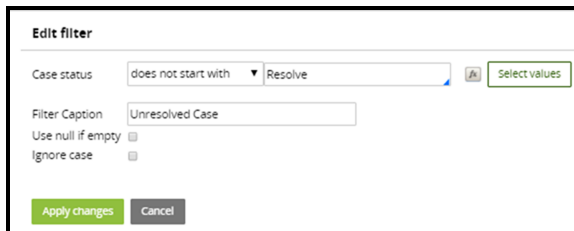
Add a filter

You may want to filter the results so the report definition retrieves a subset of the information. For example, you would use a filter if you want to display only unresolved cases instead of all cases. You create filter conditions in the **Edit filters** section.

In the Case Manager portal, filter the results of your report data by selecting an item in the **Data Explorer** on the left and dragging it to the yellow box on the right.



To further refine the filter, click on the filter listed in **Filtered by**, then create a filter expression by adding a relationship and a value. You may also add a function. Click **Apply changes**. In this example, **Case status does not start with Resolve** displays in Filtered by as **Unresolved Case**.



In Designer Studio, in the **Condition** field, enter a capital letter to identify the filter. By default, the first filter condition field is A. If you add filters, give each filter a unique Condition identifier. When you use more than one filter, you can specify AND/OR conditions in the filter conditions field.

Name the filter in the **Caption** field. This name appears in the report header so users know that the results are filtered.

In the **Column source** field, select a property reference for the condition.

Identify the data element you want to compare against a condition. In the **Relationship** field, specify a relationship such as **equals** or **greater than**. In the **Value** field, specify the value you want to use in the comparison test.

The following example shows a filter for showing only cases resolved by the user.



If you use a DateTime property in the Column source field, you can use the symbolic date feature to select a value. The system calculates specific time periods and identifies them by name. You can also use the feature to select specific dates from a calendar. To use the calendar feature, click **Select values**.

Condition	Caption	Column source	Relationship	Value
A	Resolved By Me	.pxCreateDateTime	Is equal	Previous Year

Make the report available in the Report Browser

By default, your report is available in the Report Browser when created in the Case Manager portal.

In Designer Studio, use a setting in the **Report Viewer** tab if you want to make the report available in the Report Browser. In the following example, the setting allows users to generate business reports they can use to monitor application performance.

In the User actions section on the **Report Viewer** tab, select **Display in report browser**. In the drop-down list next to the setting, select the category that will contain a shortcut to your report. Categories group reports according to the type of information the reports contain. Categories enable users to find reports in a user portal Report Browser.

User actions

Display in report browser Monitor Processes

How to organize report results

The way you format report results helps users find and analyze specific information. You can group and summarize the information for effective business presentations. Additionally, you can display reports as lists, charts, or graphs to create impact.

Summarizing results

Summarizing reports is useful when users must analyze a large amount of data. While list reports contain the detailed information a user needs, summary reports allow users to quickly identify key statistics.

When you generate a report, Pega returns the results as a list of records. You can convert list reports to summary reports, which summarize one or more columns to calculate counts, totals, or averages. For example, a user may ask for a count of cases handled by the manager who created them. A list report could show the same data but would not provide the summary counts that are useful to the user.

In the Case Manager portal, a manager can click **Edit report**, right-click a column heading and select **Summary function**, and then select how to summarize the results.

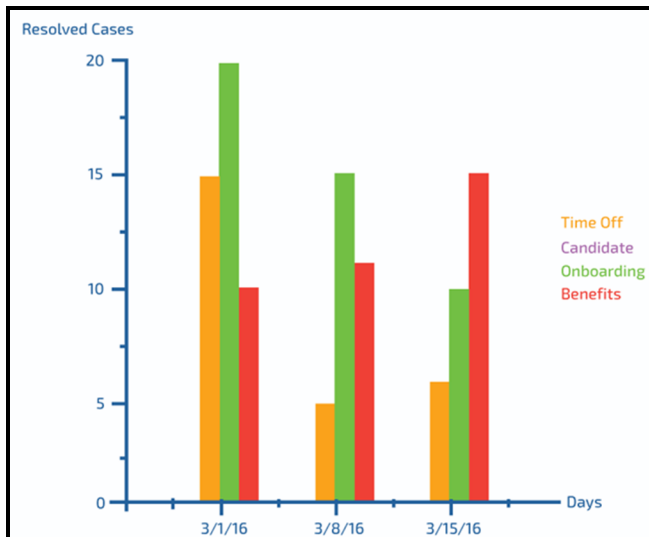
Manager	Case ID	Office
Harry	O-101	Atlanta
Fred	O-83	Boston
Fred	O-99	Boston
Harry	O-64	Atlanta
Harry	O-171	Atlanta
Harry	O-623	Boston

Manager	Case ID	Office
Harry	4	Atlanta
Fred	2	Boston

Visualizing summary results

Visual presentation of summary data can be effective for business analysts who want to quickly review and analyze the data. You can display summary report data in charts or graphs. Using charts and graphs helps users identify trends or statistics quicker than sorting through lists of data. For example, you can create a line graph that shows the number of cases resolved for each case type over a number of weeks.

In the Case Manager portal, a manager can click **Edit report**, then click **Edit chart**, and select the **All chart types** to see and select other types of charts.



Sorting values in the columns

Users who want to see a sequential ordering of data can sort values in ascending or descending order. Text characters are sorted alphabetically, and numbers are sorted numerically. You can control which column is sorted first by specifying the sort order.

For example, the facilities manager wants to sort the new hires by start date so that the manager can prepare the office space. Assume the report orders the columns as Employee, Start Date, and Location. First you select the start date as the column you want to sort first. Then you specify the column as the first one in the sort order. When you generate the report, the employee with the most current start date appears at the top of the list. The other employees are listed in descending order according to their start dates.

In the Case Manager portal, a manager can click **Edit report**, then right-click on the column heading, and select **Sort > Lowest to highest** or **Highest to Lowest**.

Employee	Start Date	Location
Kate Picardo	3/3/16	Atlanta
James Martin	2/9/16	Atlanta
Anne Walker	4/12/16	Boston
Robert Wang	2/9/16	Boston
Julia Phagan	3/8/16	Atlanta
William Kirk	2/8/16	Atlanta
Kelly Smith	4/5/16	Boston

↓

Employee	Start Date	Location
Anne Walker	4/12/16	Boston
Kelly Smith	4/5/16	Boston
Julia Phagan	3/8/16	Atlanta
Kate Picardo	3/3/16	Atlanta
James Martin	2/9/16	Atlanta
Robert Wang	2/9/16	Boston
William Kirk	2/8/16	Atlanta

Grouping results

Grouping results helps users easily analyze trends or statistics found on large reports. You can group report results so that records are grouped in a column you specify. When you group results, the grouped column values appear once for each group. The rows that contain the column value are listed under the group by value. For example, you want to show onboarding cases for each office location. You specify office location as the one you want to group. When you run the report, each location name appears once in the left column. The cases are grouped in rows next to the location name.

Note: Grouping can only be performed in Designer Studio, not in the Case Manager portal.

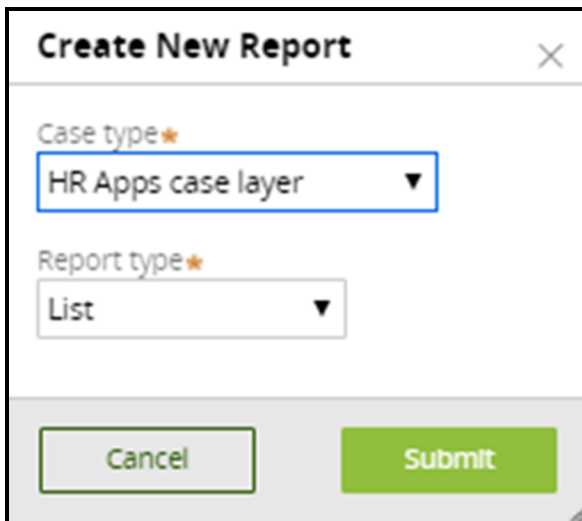
Start Date	Employee
Location: Boston	
4/12/16	Anne Walker
4/5/16	Kelly Smith
2/9/16	Robert Wang
Location: Atlanta	
3/3/16	Kate Picardo
2/9/16	James Martin
3/8/16	Julia Phagan
2/8/16	William Kirk

Creating a report

When you create a report, you design the report by creating the report definition rule and adding columns. Optionally, you can add filters and make the report available in the Report Browser.

Create a report definition rule

1. In the Case Manager portal, click **Reports** to access the Report Browser.
2. In the Report Browser, click the **Add report** button to create a new report.
3. In the Create New Report pane, select the **Case type** on which you want to report. The **HR Apps case type** includes all case types.

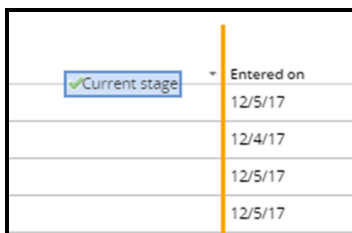


The screenshot shows a dialog box titled "Create New Report" with a close button (X) in the top right corner. It contains two dropdown menus. The first is labeled "Case type" with a red asterisk and has "HR Apps case layer" selected. The second is labeled "Report type" with a red asterisk and has "List" selected. At the bottom, there are two buttons: "Cancel" (light green) and "Submit" (dark green).

4. Select the report type to create. If you are not sure what type of report to create, select **List**. You can convert a list report to a summary report or a chart during configuration.
5. Click **Submit** to access the Report Editor to add columns and filters to your report.

Add columns

1. In the Data Explorer search field, search for the data element to add to the report.
2. Drag-and-drop the data element into the appropriate location in the report heading to add the column to the report.



The screenshot shows a table with a column menu open over the "Current stage" column heading. The table has two columns: "Current stage" and "Entered on". The "Entered on" column contains four rows of dates: 12/5/17, 12/4/17, 12/5/17, and 12/5/17.

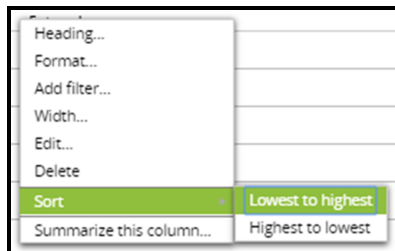
Current stage	Entered on
	12/5/17
	12/4/17
	12/5/17
	12/5/17

3. To open the column menu, hover your mouse cursor over the column heading and a down-arrow

displays to the left of the next column on the right. Alternatively, left-click on the column heading.

Current stage ▾	Entered on
Pre-arrival	11/30/17
Pre-arrival	12/1/17

4. Open the column menu to see what parameters you can change. For example, click **Sort** to sort the column and put the contents of the column in ascending or descending alphabetical order.



Add a report filter

Note: Adding a report filter is an optional step.

1. In the Data Explorer search field, search for a data element.
2. Drag-and-drop the data element into the yellow Filter box to filter your report based a value, such as when a case is created. The Edit filter pane displays.

A screenshot of the 'Edit filter' dialog box. It has a title bar 'Edit filter'. Inside, there is a text field 'Create Date/Time' with a dropdown menu set to 'is equal' and a text field containing 'Last 90 Days'. To the right of the text field is a 'Select values' button. Below this are three checkboxes: 'Filter Caption' (empty), 'Use null if empty' (unchecked), and 'Ignore case' (unchecked). At the bottom are two buttons: 'Apply changes' (green) and 'Cancel' (grey).

3. As shown in the image in step 2, select a relationship such as is equal to build the filter expression.
4. Also as shown, enter an appropriate value to filter out the cases that do not meet these requirements, such as the last 90 days.
5. Click **Apply changes** to save your filter. Other filters may already be included in Filtered by. The report data updates immediately.

Filtered by: **Unresolved Case** ✕ and **Create Date/Time = Last 90 days** ✕

6. Click the **Done editing** button and the **Save report as** pane displays.
7. Enter a name for the report in the **Title** field.
8. The default category is **My Reports** so that your reports are available in the Private category of the Report browser. You may also save the report the public category. Click **Submit** to save your report and make it available in a private or public category.

9. Close the **Report Editor**.
10. In the Report Browser, click the appropriate category to locate your report.
11. Verify that your report is in the list of available reports.

Organizing report results

You built your basic report by adding columns and filtering the results. You can organize the results so that users can easily find and analyze the information in the report.

You can organize report results in three ways:

- Summarize the values in one or more columns. When you summarize report results, you can also graphically display the summary results.
- Group the results under a column that you specify.
- Sort values in columns.

Summarize the results

1. To create a summary report, in the Edit Columns section, identify the data element column you want to summarize.
2. In the Summarize column, select how you want the to summarize the column. If the column contains text values, you can summarize by count or sum. If the column contains numeric values, you can summarize them by count, minimum, maximum, average, or sum. In the following example, you are summarizing the Case ID by count.

Column source	Column name	Summarize
.Employee.Manager	Manager	<blank>
.pyID	Case ID	Count
.Office	Office	<blank>

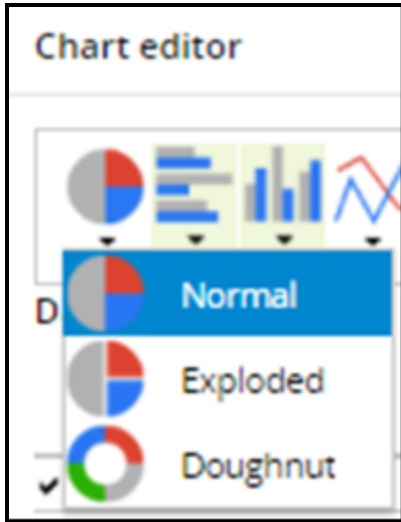
3. In the form header, click **Actions > Run** to test the report.

Manager	Office	Case ID
▼ ALL		15
▼ Frank Foley		3
	Atlanta	1
	Boston	2
▼ Jim Johnson		3
	Atlanta	3
▼ Robin Redford		2
	Cambridge	2
▼ Tom Davis		1

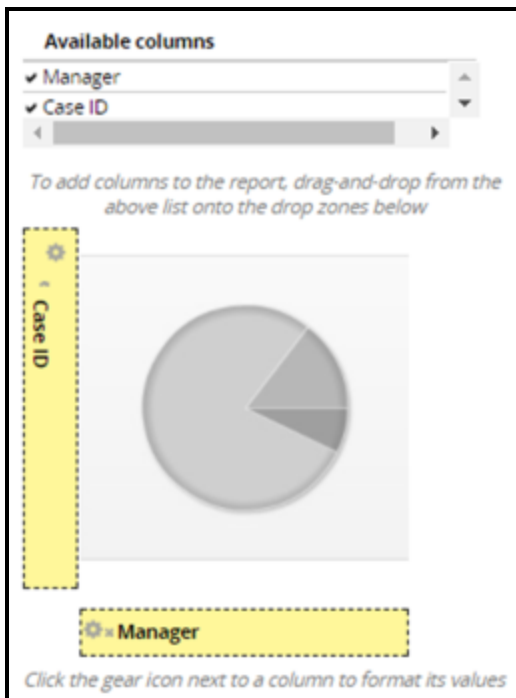
Note: The results show the count of cases for each of the managers.

Graphically display summary results

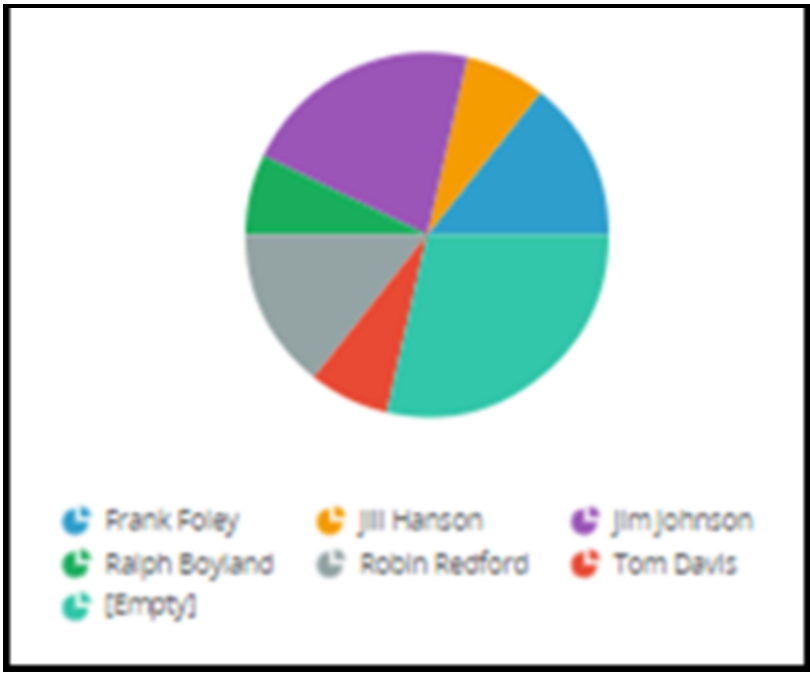
1. To display a summary report in a chart or graph, open the **Chart** tab.
2. In the Chart editor section, select **Include Chart**.
3. In the Chart editor, select the type of chart you want. In the following example, you choose a pie chart.



4. From the Available columns list:
 - a. Select the column you summarized and drag it to the vertical drop-zone.
 - b. Select the column you want to measure and drag it to the horizontal drop-zone.
 - c. In the following example, you design a pie chart that groups in each section cases by manager.



- In the form header, click **Actions > Run** to test the report.
The chart looks like the following when you run the report.



Grouping the results

Note: You cannot group summary report results. Remove the summarizing function on any report columns before grouping results

- To group report results, in the Query tab, identify the column you want to group the results under and enter 1 in the Sort order column.

Column name	Summarize	Sort type	Sort order
Manager	<blank>	Lowest to Highest	1
Case ID	<blank>	Lowest to Highest	2
Office	<blank>	Lowest to Highest	3

- Open the **Report Viewer** tab.
- In the Grouping section, select **Group results**.

Grouping

Group results and display values in sort columns 1 - 1 as group headings

Group results when exporting to Excel

Display groups using custom section

4. In the form header, click **Actions > Run** to test the report. The following image shows the results.

Office ↑ ²	Case ID ↑ ¹	
Manager: Frank Foley		3
Atlanta	O-19	
Boston	O-23	
Boston	O-6	
Manager: Jim Johnson		3
Atlanta	O-17	
Atlanta	O-20	
Atlanta	O-21	
Manager: Robin Redford		2
Cambridge	O-18	
Cambridge	O-22	
Manager: Tom Davis		1

Sort values in a column

1. To sort value in a column, open the **Query** tab.
2. In the Sort order column, enter 1 in the first column you want sorted. Enter 2 in the next column you want sorted, and so on. After column 1 is sorted, then column 2 is sorted.
3. In the Sort type column, select the order to sort the values. The following example sorts the manager name alphabetically starting with the letter A first. Then, the Case ID values are sorted, starting with the highest value.

Column name	Summarize	Sort type	Sort order
Manager	<blank>	▼ Lowest to Highest ▼	1
Case ID	<blank>	▼ Highest to Lowest ▼	2
Office	<blank>	▼ Lowest to Highest ▼	3

4. In the form header, click **Actions > Run** to test the report. The following example shows the results. In the rows under the first column, Manager, the first letter in the first name begins with "F" and ends with "T" — lowest to highest. The rows under the second column, Case ID, starts at the highest

number for each manager per Office location.

Manager ↑ ¹	Case ID ↓ ²	Office ↑ ³
Frank Foley	O-23	Boston
Frank Foley	O-19	Atlanta
Jim Johnson	O-21	Atlanta
Jim Johnson	O-20	Atlanta
Jim Johnson	O-17	Atlanta
Robin Redford	O-22	Cambridge
Robin Redford	O-18	Cambridge
Tom Davis	O-9	

Optimizing report data

Introduction to optimizing report data

Pega Platform applications allow system architects to improve report performance through a process called optimization. Optimizing properties allows Pega to extract report data without the need to open each case.

In this lesson, you learn how Pega Platform stores case data and how data storage affects report performance. You also learn how to optimize case data to improve report performance for users.

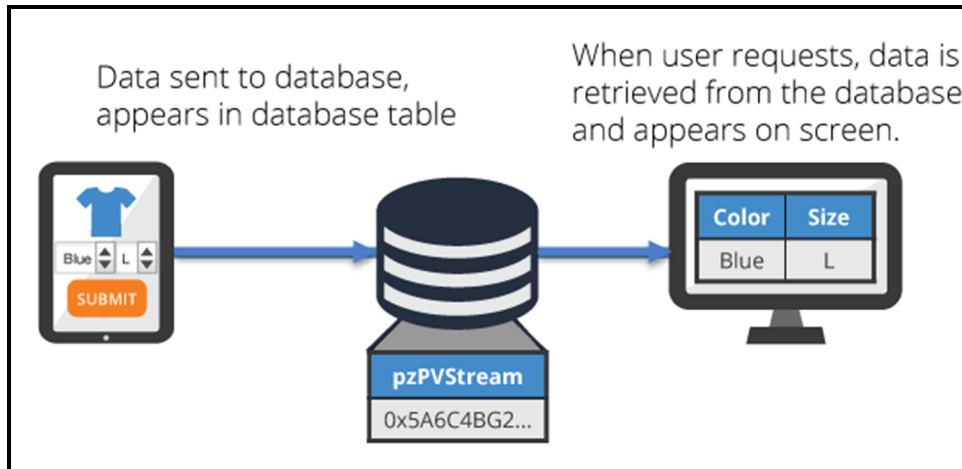
After this lesson, you should be able to:

- Explain the impact of property optimization on report performance
- Describe how Pega stores case data
- Explain how property optimization affects properties
- Optimize case data for reports

Data Storage in Pega applications

Pega Platform applications store each case as a unique record in a relational database. Within each record, Pega stores case data in a **binary large object** (BLOB) field. Each time an end user completes an action by clicking OK or Submit, Pega writes the case data to the BLOB field.

Within the database, each record is a row in a database table. Each column in the table displays the contents of a field from the case record, including the BLOB field. When an end user opens a case, Pega locates the record in the correct table, then reads the contents of the BLOB column to extract the case data.



A BLOB field offers three advantages for storing case data:

- Unlimited storage size — BLOB fields are not constrained by size, so a BLOB field can contain any amount of information.
- Flexibility — Pega writes all case data to the BLOB, so changes to the data model of a case are contained within the BLOB. Using the BLOB field avoids the need to update the database structure, or schema, as the data model changes.
- High performance — Since the BLOB field stores all case data, Pega reads and writes the entire case at one time. This optimizes application performance for end users as they create and process cases.

Using the BLOB penalizes performance for reporting. When an end user runs a report, an application must decompress the BLOB to extract the required data. For a large number of cases, this process significantly increases the time needed to run the report.

KNOWLEDGE CHECK



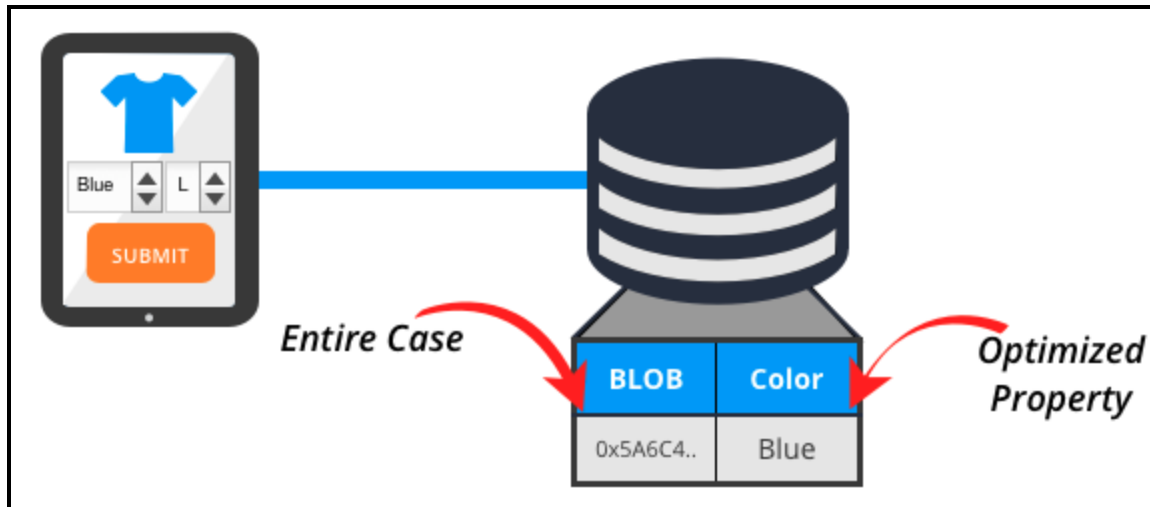
Why is case data stored in a BLOB column?

The BLOB field provides greater flexibility and performance for case data. The BLOB field is capable of storing an unlimited amount of data. The BLOB field also allows the data model of a case to change without impacting database storage. And the BLOB field allows an application to read or write the entire case at once.

Property optimization

End users rarely need to retrieve an entire case when running a report. Rather, end users only need the data elements required by the report. Extracting data from a BLOB impacts performance compared to reading property values from a database table. This impact is most pronounced when extracting data for report filters and sorting or grouping the content of a column.

To improve report performance, Pega offers a hybrid data storage model. This hybrid model allows applications to store data both in dedicated indexed columns and in a BLOB field. To store property values in indexed columns, you must **optimize** the property for reporting.



When you optimize a property, Pega creates a column for the property in a database table. Because the value of the property is then visible in the table, or exposed, optimizing a property for reporting is also called "exposing" the property.

When a case uses an optimized property, Pega writes data to both the property field and the BLOB field. When a report uses optimized data, Pega reads from the property column, rather than the BLOB. By not decompressing the BLOB field to read case data, optimization reduces the time and memory needed to run the report.

By default, Pega optimizes properties that store process data such as:

- The creation date of a case
- The status of a case
- The case ID

Properties that store process data begin with the letters px, py, or pz.

Properties created by system architects to store business data are not optimized by default. Reports that use an unoptimized property display a warning that states the potential impact on performance. Performance warnings due to an unoptimized property are resolved by running the Property Optimization tool.

KNOWLEDGE CHECK



Why are properties exposed, or optimized, for reporting?

Exposing properties allows Pega to read the property value without decompressing the BLOB to extract the property value.

Optimizing properties for reporting

Pega provides the Property Optimization tool to optimize properties for reporting.

When you use the Property Optimization tool on a property, Pega exposes the property as a database column, and populates the new column by extracting values from the BLOB column. For an embedded property in a page group or page list, the tool automatically creates a new database table for the property, and the appropriate Index- class and Declare Index rule to update the new table.

To optimize a property:

1. Using the Application Explorer, expand the Data Model node.
2. Right-click a property name and click **Optimize for reporting**.
3. Select the tables in which you want to create a dedicated database column for this property. You must select at least one table.
4. If the property is embedded, specify the ruleset and version that is to contain the new Index- class, the properties in that class, and the Declare Index rule.
5. Select whether to optimize the property now or later. If you select later, click the calendar icon to select a date within seven days of the current date to optimize the property.
6. Click **Next**.
7. Review the tables to which the property column will be added and the classes that will be mapped to each table.
8. Click **Next**.
9. Click **Finish**.

Note: Background processing may take minutes or longer, depending on volume. Computations in your applications involving the property value may fail or produce incorrect results until all background processing is complete.

You can view the status of your background job by clicking **Designer Studio > System > Database > Column Population Jobs**.

You can view the classes for which a property has been optimized on the **Advanced** tab of the Property record.

TESTING AND DEBUGGING APPLICATIONS

Unit testing application rules

Introduction to unit testing application rules

Unit testing supports the continuous delivery of applications by enabling quality testing of individual rules. In this lesson, you learn how to unit test individual rules, and record tests to run as part of automated tests.

After this lesson, you should be able to:

- Describe the role of unit testing in application development
- Unit test a rule with the Run Rule window
- Record a unit test to run at a later time

Unit testing

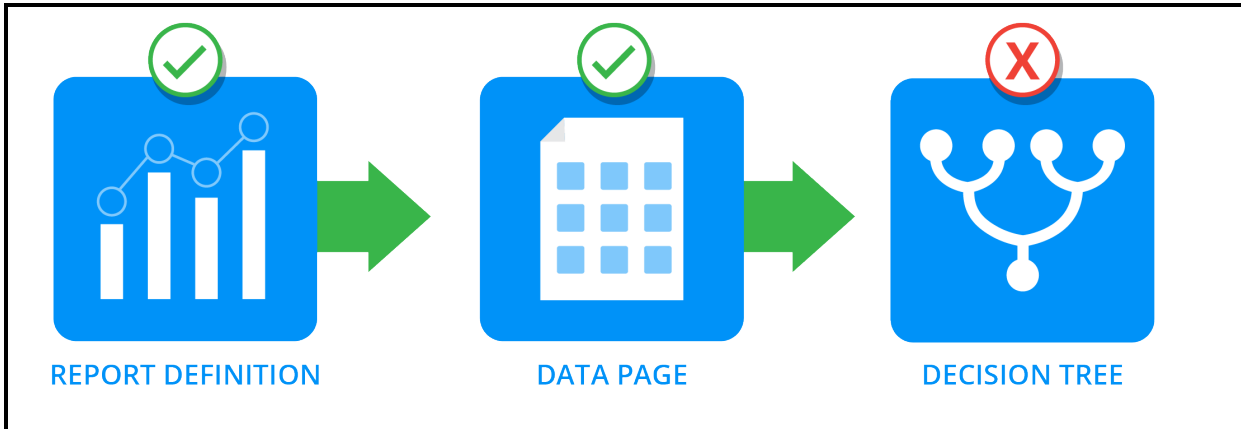
An incorrect rule configuration in an application can cause delays in case processing. When an error occurs, end users may need to reassign work, or a case may require repair by an administrator. For example, consider a case that should route to the Fulfillment department. If the case routes instead to the Accounting department, an accountant must reroute the case to Fulfillment. The accountant wastes time rerouting the assignment, while the Fulfillment department is idle. The result is a delay to the customer during case reassignment.



To avoid configuration errors such as incorrectly routed assignments, developers test their applications. The most basic form of application testing is **unit testing** individual rules. The purpose of unit testing is to verify that each element of the application — for example, a decision table or a report definition — works as expected. Unit testing reduces the risk of a configuration error in one rule propagating to other rules in the application, causing significant delays to case processing.

Use unit testing to reduce configuration errors. For example, consider a decision tree that evaluates a property. As illustrated in the following image, the application reads the property from a data page

sourced from a report definition. By unit testing the individual rules as you configure them, you know that each rule works as expected. If the decision tree returns an incorrect result, but the data page contains the correct data, you can isolate the error to the decision tree.



KNOWLEDGE CHECK



The purpose of unit testing is to _____.

identify configuration errors in a rule before the errors are compounded in an application when the result of the rule depends on another rule.

Unit testing is a key enabler of a DevOps culture for application development. DevOps relies on the automation of release management and packaging tasks to support a continuous development and continuous integration model of application development. Unit testing on a recurring basis can identify issues quickly after introduction. Unit testing improves application quality because developers can fix any issue before releasing an application.

How to unit test a rule

You can unit test many of the rules used to configure application behavior.

Perform the following tasks to successfully unit test a rule:

- Open the Run Rule window from the rule form.
- Initialize the rule with test data.
- View the result returned by the rule.

Open the Run Rule window from the rule form

To unit test a rule, open the rule form and select **Actions > Run**. For some rule types, such as binary file rules, Pega does not provide an option for unit testing. If the rule cannot be unit tested, the **Run** option does not appear in the **Actions** menu.

Run Rule window appearance

The appearance of the Run Rule window varies by the type of rule tested. For example, data page rules and when rules use a dialog that mimics the appearance of the clipboard. For details on unit testing a specific type of rule, see the Help topic [Unit testing of rules](#).

If the rule operates on data in memory, Pega displays the Run Rule window. For example, declare expressions, decision rules, and UI rules operate on data in memory, so Pega displays the Run Rule window when you select **Actions > Run**.

Report definitions run against the contents of a database table, rather than memory. As a result, Pega skips the Run Rule window and returns the report data when you select **Actions > Run**.

Initialize the rule with test data

In the Run Rule window, use the Test Page section to identify a page of data used to unit test the rule. This page is the **test page**. The Run Rule window creates the test page on the clipboard. The name of the test page is the same as the *Apply to* class of the rule, with the prefix *temp_* applied. This isolates the test data from other pages in memory.

When you unit test a rule, determine how you want to source data for the test page. The Run Rule window provides several options to add data to the test page, depending on the type of rule tested.

Typically, you can create a page of test data using a data transform. By default, Pega applies the *pyDefault* data transform to populate the test page with data. You can also select another data transform to apply. For example, to unit test a decision table, you can create a data transform to provide values for the properties evaluated by the table, rather than manually enter values when you run the rule.

You can also copy a page of the appropriate class already on the clipboard. For example, to test a decision table used by a case type, you can create a case and copy data from *pyWorkPage*.

KNOWLEDGE CHECK



The purpose of a test page is to _____.

store data to use when unit testing a rule

View the result returned by the rule

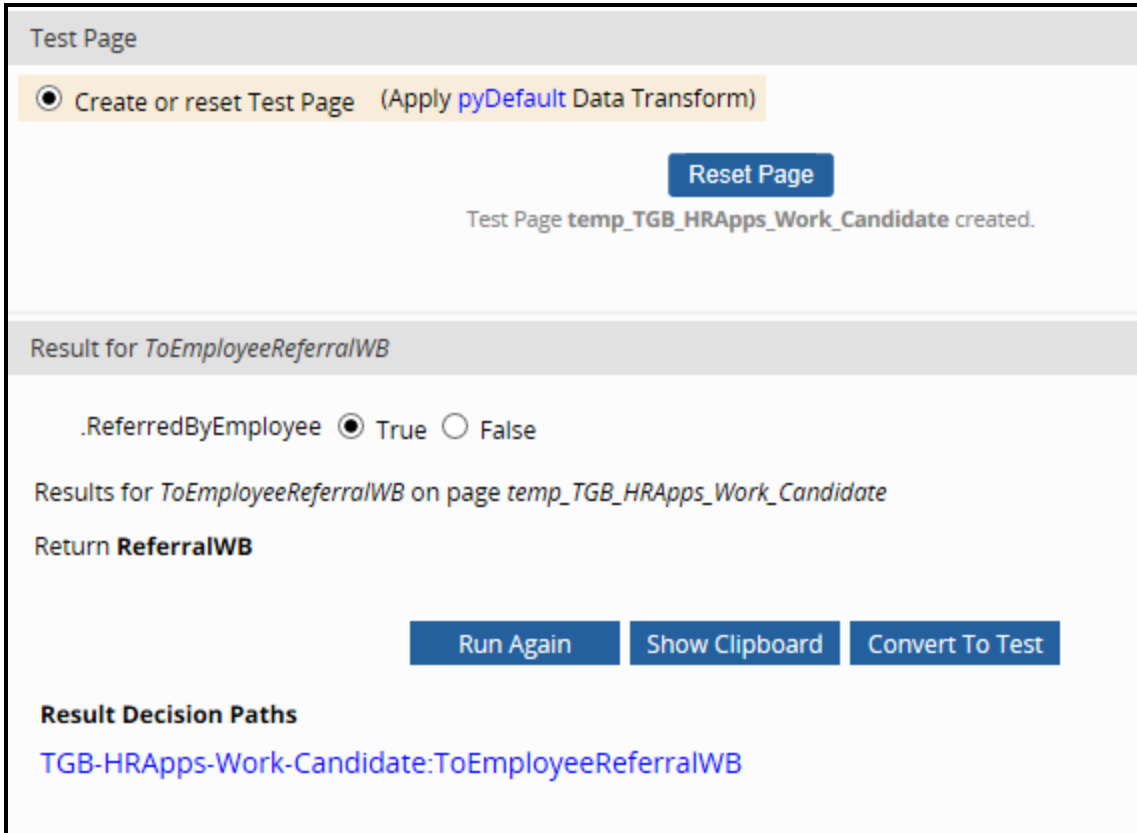
Once you select a method to populate the test page, click **Reset Page**. This populates the page with data and runs the rule to return a result.

Tip: Click **Reset Page** to clear the result of a previous test.

If the rule accepts inputs, you can enter values to test. The values you enter in this section of the window override the values on the test page. For example, consider the *ToEmployeeReferralWB* decision table shown in the following image. The decision table determines the work queue to which to route an assignment. If the value of the property *Referred by employee* is true, then the application routes the assignment to the *ReferralWB* work queue.

	Conditions		Actions
	<input type="radio"/> Referred by employee		Return
<input type="radio"/> if	true	→	ReferralWB
otherwise		→	RecruitingW

For a decision table or decision tree, assign values to each input property and click **Run Again**. The window displays the result of the decision table or tree. The following image shows the result of the test when *ReferredByEmployee* is set to true.



You can also review the result of the unit test by viewing the output of the test with the Clipboard tool. The Run Rule window generates several pages on the clipboard. These pages provide information about the rule test.

- **RuleToRun** is the clipboard representation of the rule you tested. If more than one version of the rule is present in the application, view this page to identify the tested rule version.
- A **temp_** page is created or copied when you test a rule. The names of these pages begin with the string temp_. View this page to examine the data used to initialize the rule for the unit test.

KNOWLEDGE CHECK



What are two ways to view the output of a unit test?

Run the unit test or view the output of the test with the Clipboard tool.

How to record a unit test for automated testing

After you successfully unit test a rule, you can create a **test case** based on the result of the test. A test case identifies one or more testable conditions used to determine whether a rule returns an expected result. Creating a reusable test case supports the continuous delivery model, providing a means to test rules on a recurring basis to identify impacts of new or modified rules.

KNOWLEDGE CHECK



What is the role of a test case in application development?

A test case identifies one or more testable conditions that determine whether a rule returns an expected result.

You can run a saved unit test from the **Test Cases** tab of a rule. You can also run the unit test automatically using the PegaUnit testing facility. However, configuring PegaUnit to run automated tests is not addressed in this lesson.

Create the test case

To create a test case, click **Convert to test** in the Run Rule window after completing a unit test. In the **Expected results** area of the test case, define the expected results that indicate a successful unit test. Each expected result consists of an **assertion** that describes one or more conditions to test.

Test cases support several types of assertions designed to test various aspects of rule execution. The assertions available for a test case vary according to the type of rule tested.

For a full explanation of the supported assertion types and their usage, see the Help topic [Assertions](#).

Some examples of assertions and their uses are provided in the following table:

Assertion type	Usage	Example
Property	Tests the value of the specified property. Requires the page on which the property is defined, a comparison operation, and a comparison value	<i>pxUrgencyWork</i> is equal to 10
Decision result	Tests the value returned by a decision rule. Requires values for each input property needed by the decision rule to return the expected result	When <i>Referred by employee</i> is false return <i>RecruitingWB</i>
Expected run time	Tests whether a rule executes within an allowed amount of time. Requires a comparison operation and an allowed time in seconds	Expected run time is less than or equal to three seconds
Page	Tests for the presence of a page in memory. Requires the name of the page and a comparison operation	Page <i>D_CoursesList</i> has no errors

KNOWLEDGE CHECK



What is the relationship between an assertion and a test case?

A test case describes one or more assertions that describe expected behavior for a rule.

Save the test case

When you complete the set of expected results, click **Save** to complete the configuration of the test case.

Saving the test requires access to a ruleset that is configured to store test cases. If the ruleset you select is not configured to store test cases, Pega Platform returns an error. Before you record a unit test, work with your system administrator to identify a suitable ruleset for storing test cases.

Save test cases to a dedicated testing ruleset for maintenance and packaging. For ease of configuration, use an application built on the development application, and include a ruleset designed specifically for test cases. When the development application is released to production, you can migrate the application without including the test cases.

Run the test case

You can access a saved test case from the rule by using the **Test cases** tab. The Test cases tab lists all of the test cases recorded for the rule and the status of each test case as of its last execution.

Tip: In Designer Studio, the Automated Testing landing page lists all the test cases defined for an application and the status of each test case as of its last execution. From the landing page, you can create test suites consisting of one or more test cases. To access the landing page, click the **Designer Studio** menu and select **Application > Automated Testing**.

Re-run a test case by selecting the test and clicking **Run selected**. If a test case fails, click **Failed** to open the result and identify each assertion that returned an unexpected result. If a test case returns expected results, a green **Passed** status button is displayed. If a test case passes, no test results are displayed when you click the button.

Tip: When unit testing a chain of dependent rules, begin with the rule that has no dependency on other rules.

KNOWLEDGE CHECK



What does the Automated Testing landing page show?

It lists all the test cases defined for an application and the status of each test case as of its last execution.

Debugging application errors

Introduction to debugging Pega applications

Software applications are rarely written free of errors. Finding problems early on and giving system architects the tools to find problems is instrumental to a successful development cycle.

Some errors in an application are easy to diagnose. Other errors can prove difficult to identify and resolve. To help you identify and resolve errors in your applications, Pega provides the Tracer tool to allow you to review application execution and identify the root cause of errors.

After this lesson, you should be able to:

- Identify the role of the Tracer in debugging applications
- Use the Tracer to investigate application errors

The Tracer

When an error occurs in an application, you need to identify the cause of the error so you can correct the application behavior. For example, a declare expression may return an unexpected value. If you forget to add one of the input properties to a UI form, users cannot provide the missing value and the declare expression returns an incorrect result. If you forget to make an entry required for an input property, users can submit a form without providing a needed value.

Or, perhaps you use a data page to populate a drop-down list. If the contents of the drop-down list are incorrect, you need to determine whether the control or the connection to the data source has an improper configuration.

Identifying the root cause of an error is critical to correcting application behavior. You view the events that led to the error and determine which behavior to address to fix the issue.

To view events such as those that occur when a case processes, you use the **Tracer**. In Pega, the Tracer allows you to capture and view the events that occur during case processing. Unlike the Clipboard tool, which presents the current value of properties in memory, the Tracer presents a complete log of the events that occur during case processing. This allows you to identify the cause of execution errors, such as Java exceptions or incorrect property values.

To help identify errors in case processing, the Tracer identifies processing steps that lead to an error. In the Tracer log, most steps return a status of Good, indicating that the step completed successfully. If a step returns a status of Fail, an error occurred and the step completed unsuccessfully. An error in an application may only indicate the last step in a sequence of failed steps, but not the cause of the fail sequence. Reviewing the sequence of events in the Tracer helps to identify the root cause that leads to the error seen by users.

How to investigate application errors with the Tracer

To investigate an issue with the Tracer, you configure the Tracer to monitor application execution. As your application executes, Pega logs all the processing events that result from application execution. You then view the events logged by the Tracer to analyze processing errors and identify their cause.

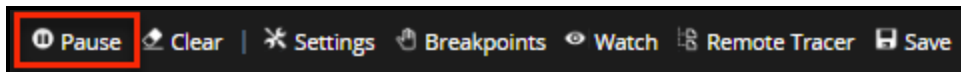
To open the Tracer, on the Developer toolbar, click **Tracer**. The Tracer logs all the actions and events that occur in a requestor session in Designer Studio. Each event is logged in order of occurrence and is identified by thread, event type, and status.

LINE	THREAD	INT	RULE#	STEP METHOD	STEP PAGE	STEP	STATUS	EVENT TYPE	ELAPSED	NAME	RULESET
- Events for Requestor : H16E37F94FCEE190580F2425D55818578											
6164	TABTHREAD4	11	179		pyLanding			Activity End	0.0210	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6163	TABTHREAD4	11	179	Page-Remove	pyLanding	13	GOOD	Step End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6162	TABTHREAD4	11	179	Page-Remove	pyLanding	13		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6161	TABTHREAD4	11	179	Property-Set	pyLanding	12	GOOD	Step End	0.0010	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6160	TABTHREAD4	11	179	local.skinName != ""	pyLanding		True	When End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6159	TABTHREAD4	11	179	local.skinName != ""	pyLanding			When Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6158	TABTHREAD4	11	179	Property-Set	pyLanding	12		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6157	TABTHREAD4	11	179	java	pyPortal,pyPortalskin	11	GOOD	Step End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6156	TABTHREAD4	11	179	java	pyPortal,pyPortalskin	11		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6155	TABTHREAD4	11	179	Property-Set	pyPortal,pyPortalskin	10		Skip Step / ...	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6154	TABTHREAD4	11	179	(param.pzSkinName != ...	pyPortal,pyPortalskin		False	When End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6153	TABTHREAD4	11	179	(param.pzSkinName != ...	pyPortal,pyPortalskin			When Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6152	TABTHREAD4	11	179	Property-Set	pyPortal,pyPortalskin	10		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6151	TABTHREAD4	11	179	java	pyLanding	9	GOOD	Step End	0.0040	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6150	TABTHREAD4	11	179	When Rule Evaluation	pyLanding		True	When End	0.0000	@baseclass pyMayOptim...	Pega-UIEngine 07-10-17
6149	TABTHREAD4	11	179	When Rule Evaluation	pyLanding			When Begin		@baseclass pyMayOptim...	Pega-UIEngine 07-10-17
6148	TABTHREAD4	11	179	When Rule Evaluation	pyLanding		True	When End	0.0010	@baseclass pyMayOptim...	Pega-UIEngine 07-10-17
6147	TABTHREAD4	11	179	When Rule Evaluation	pyLanding			When Begin		@baseclass pyMayOptim...	Pega-UIEngine 07-10-17
6146	TABTHREAD4	11	179	java	pyLanding	9		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6145	TABTHREAD4	11	179	Property-Set	pyLanding	5	GOOD	Step End	0.0040	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6144	TABTHREAD4	11	179	local.skinName != ""	pyLanding		True	When End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6143	TABTHREAD4	11	179	local.skinName != ""	pyLanding			When Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6142	TABTHREAD4	11	179	param.pzUsePreference...	pyLanding		True	When End	0.0000	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6141	TABTHREAD4	11	179	param.pzUsePreference...	pyLanding			When Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6140	TABTHREAD4	11	179	When Rule Evaluation	pyLanding		True	When End	0.0000	Rule-Obj-Class pyUseDe...	Pega-Desktop 07-10-19
6139	TABTHREAD4	11	179	When Rule Evaluation	pyLanding			When Begin		Rule-Obj-Class pyUseDe...	Pega-Desktop 07-10-19
6138	TABTHREAD4	11	179	Property-Set	pyLanding	5		Step Begin		@baseclass GetWorkSty...	Pega-UIEngine 07-10-19
6137	TABTHREAD4	11	179	Property-Set	pyLanding	4		Skip Step / ...	0.0010	@baseclass GetWorkSty...	Pega-UIEngine 07-10-19

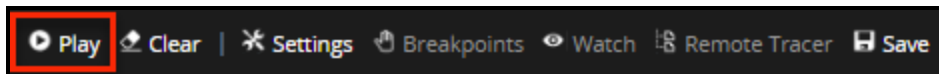
Click a line in the Tracer to view details about the event. The step's details are displayed in a new window. From this window, you can view the contents in memory at the time the event occurred. When you finish reviewing the event properties, close the window to return to the Tracer.

Properties on Page TraceEvent [6150]	
Header	
Sequence	6149
Interaction	11
Timestamp	May 16, 116 - 12:37:49 14:19:19 (20160516T163749.035 GMT)
Elapsed Time	0.0000 (s)
Event Type	When End
Event Name	When End
Event Key	RULE-OBJ-WHEN @BASECLASS PYMAYOPTIMIZESKIN #20150213T181214.144 GMT
Thread Name	TABTHREAD4
Requestor ID	H16E37F94FCEE1905B0F2425D55818578
Correlation ID	H16E37F94FCEE1905B0F2425D55818578
Work Pool	TGB-HR-Work
Last Step	@BASECLASS GETWORKSTYLE #20150818T155516.374 GMT Step: 9 Circum: 0
Input	Activity=@baseclass.doUIAction
Ruleset Name	Pega-UIEngine
Ruleset Version	07-10-17
Standard	
Activity Number	179
Parameter Page Name	=unnamed=
Primary Page Class	Rule-Obj-Class
Primary Page Name	pyLanding
Step Method	When Rule Evaluation
When Status	True

The events that occur as you use Designer Studio are logged and displayed in the Tracer. To suspend the logging of events in the Tracer, click **Pause** on the toolbar that runs along the top of the Tracer window.



While the Tracer is paused, **Pause** is replaced with **Play**. To resume the logging of events in the Tracer, click **Play** on the toolbar.



When you use the Tracer, you may want to focus on specific parts of your application. The Tracer provides several options to focus event logging, which are available on the toolbar.

Button	Function
Settings	Select the rule types, rulesets, and events to trace. Also, identify when a step results in a Java exception or a status of Fail or Warn.
Breakpoints	Identify when the application reaches a specific step in an activity.
Watch	Monitor a variable to detect when its value changes.

COURSE SUMMARY

Course summary

System Architect Essentials summary

Now that you have completed this course, you should be able to:

- Explain the benefits of using the Pega model-driven application design and development approach
- Model the life cycle of a case that mirrors the way business people think about how work is completed
- Identify the high-level responsibilities associated with Pega Platform for both Pega business architects and system architects
- Describe Pega's Direct Capture of Objectives™ approach to increasing the speed and accuracy of application delivery
- Explain the purpose and benefits of best practices and guardrails
- Validate case data to ensure that user entries match required patterns
- Configure a Wait shape to enforce a case processing dependency
- Configure user views and data elements during case life cycle creation
- Use the Clipboard tool to review case data in memory
- Set property values automatically using data transforms and declare expressions
- Configure and populate a work party with case data
- Create data classes and properties for use in a Pega application
- Automate decision-making to improve process efficiency
- Design responsive user forms for use on any platform or browser
- Design reports to deliver key insights to business users
- Incorporate and manage reference data to allow applications to adapt to changing business conditions
- Test your application design to analyze rule behavior and identify configuration errors

Next steps

Completion of System Architect Essentials helps prepare students for the Certified System Architect exam. To help you study for the exam, enroll in the CSA Practice Exam course in Pega Academy.

[Register for the exam.](#)