Pega Platform '24.2

3 September 2025

©2025 Pegasystems Inc

CONTENTS

Application development 12
Introduction to application development 17
App Studio overview 20
Dev Studio overview 27
Admin Studio overview 29
Prediction Studio overview 31
Changing your studio 34
Exploring application-editing mode 35
Building your first application 35
Creating a new application from a Blueprint 40
Exporting an existing application as a Blueprint 46
Importing a Blueprint into an existing application 48
Implementing a Blueprint-generated application 53
Using development aids 56
Extending the Knowledge tab 60
Creating user stories from checklists (application guides) 62
Creating applications 64
Inviting collaborators to your application 71
Configuring advanced settings for new applications 73
Switching between applications 78
Adding the My Applications gadget 78
Switching between applications in a Constellation Portal 79
Switching between applications in Dev Studio 81
Exploring the Application Definition 81
Adding built-on applications 83
Best practices for using multiple built-on applications 85
Application stack hierarchy for multiple built-on applications 88
Configuring advanced application settings 95
Managing application Cases and data 97

©2025 Pegasystems Inc

Configuring applications for reuse. 99
Setting your application password. 101
Security tab of the Application Definition. 102
Specifying the content of application documentation. 107
Basic requirements for deploying public-facing applications. 113
Restricting access to operator information in Pega 8.5. 117
Restricting access to operator information in public-facing applications in Pega Platform 8.4 and earlier. 118
Best practices for compliance and updatability. 121
Best practices for Pega Infinity application development. 130
Customizing the App Studio navigation pane. 155
Understanding Personas and access management. 165
Operators. 168
Creating an operator ID. 172
Defining user contact information and application access. 173
Defining work routing settings for an operator. 177
Defining operator availability. 181
Defining security information for an operator. 183
Deleting operators. 187
Creating a Work Queue. 187
Creating work groups. 193
Updating the organizational structure by using the organizational chart. 194
Learning about Access Groups. 199
Viewing Access Groups and operators. 202
Creating an Access Group. 205
Assigning work pools to an Access Group. 207
Work Pools and Access Groups. 208
Granting Portal access to an Access Group. 209
Configuring tools access. 210
Viewing tools authorizations in all Access Groups. 211
Viewing tools authorization for a single Access Group. 212
Editing tools authorization for a single Access Group. 213

PEGA ©2025 Pegasystems Inc

Designing applications for reuse and extension. 214
Managing reusable assets in the Reuse Library. 217
Relevant records for Rule reuse. 224
Managing relevant records. 234
Marking a record as relevant. 236
Adding a relevant record to a specified class in your application. 238
Marking relevant records as active or inactive. 239
Components. 240
Installing components. 241
Enabling components. 242
Disabling components. 244
Creating a component. 244
Documenting a component. 245
Packaging a component. 246
Shared Ruleset. 246
Creating a Configuration Set. 247
Best practices for Configuration Sets. 250
Creating Configurations. 252
Authoring decision trees in Configurations in App Studio. 260
Adding Administration landing page to an application. 263
Technical considerations for Configuration Sets. 266
Referencing properties. 268
Naming conventions for properties. 270
Configuring page, page group, and page list properties. 271
Application modularization, encapsulation, and naming for better reuse and maintenance. 274
Building logic and calculating values in your application. 278
Creating an activity. 281
Configuring steps in an activity. 285
Activity form - Completing the Steps tab - Entering Method parameters. . 289
Activity form - Completing the Steps tab - Entering preconditions. 291
Defining step transitions in an activity. 293

©2025 Pegasystems Inc

Testing method results using a transition. 297
Repeating steps in an activity. 298
Calling an Automation from an activity. 306
Best practices for writing activities for background jobs. 308
Security tab on the Activity form. 313
Defining the local variables for an activity. 320
Activity form - how to create activities for Processes. 321
Intelligent routing. 330
Unit testing an activity. 332
Clipboard contents created from activity unit tests. 332
Testing an activity in context. 333
Constraints Rules. 333
Creating constraints Rules. 334
Constraints form - Completing the Constraints tab. 336
Constraints form - Completing the Pages & Classes tab. 341
More about Constraints Rules. 343
Data Transforms. 346
Creating a Data Transform. 349
Configuring a Data Transform. 351
Data Transform actions for Clipboard. 356
Data Transform actions for JSON. 358
Data Transform form - Append and Map to action. 359
Data Transform form - Append to action. 362
Data Transform form - Apply Data Transform action. 364
Data Transform form - Auto-map action. 365
Data Transform form - For Each Page In and Exit For Each actions. 367
Data Transform form - Remove action. 369
Data Transform form - Set action. 370
Data Transform form - Sort action. 373
Data Transform form - Update Page action. 374
Data Transform form - When, Otherwise, and Otherwise When actions. . 378

©2025 Pegasystems Inc

Configuring the JSON Data Transforms settings column on the Data Transform form. 380
Data Transform form - Completing the Pages & Classes tab. 382
Adding Data Transform to a Process. 383
Standard Data Transforms. 385
More about Data Transforms. 388
Unit testing a Data Transform. 390
Creating a Data Transform in App Studio. 390
Decision tables. 393
Creating decision tables. 396
Specifying pages and classes of a decision table Rule. 403
Configuring additional options for a decision table. 404
Calculating values with decision tables in App Studio. 409
More about decision tables. 413
Unit testing a decision table. 416
Debugging decision tables with the Tracer. 417
Decision trees. 418
Creating decision trees. 422
Specifying pages and classes of a decision tree Rule. 431
Controlling output and actions for decision trees. 432
More about decision trees. 437
Configuring property evaluation in a decision tree. 438
Unit testing a decision tree. 441
Debugging decision trees with the Tracer. 442
Declare Expression Rules. 443
Creating Declare Expression Rules. 446
Defining Expressions in Declare Expression Rules. 447
Specifying pages and classes for a declare expression. 452
More about Declare Expression Rules. 453
Declare Expression form - Completing the Change Tracking tab. 459
Supported and unsupported configurations in simplified declare expressions. 466

©2025 Pegasystems Inc

Viewing the declarative network. 468
Testing Rules in the Declarative Network Viewer. 469
Declare OnChange Rules. 470
Creating Declare OnChange Rules. 471
Declare OnChange form - Completing the OnChange Properties tab. . . . 473
Declare OnChange form - Completing the Pages & Classes tab. 477
More about Declare OnChange Rules. 479
How to implement business exception processing with policy overrides. 481
Declare Trigger Rules. 485
Creating Declare Trigger Rules. 487
Declare Trigger form - Completing the Pages and Classes tab. 490
More about Declare Trigger Rules. 492
Map values. 494
Map values. 495
Completing the Matrix tab. 496
Completing the Configuration tab. 500
Completing the Pages & Classes tab. 503
More about map values. 503
Unit testing a map value. 506
Configuring rows and columns in a map value. 507
Validations for ensuring data integrity. 510
When Condition Rules. 515
Creating a When Rule. 519
Defining conditions for a When Rule. 521
Editing a When Rule in the legacy mode. 523
Configuring advanced options of a When Rule. 525
Specifying pages and classes of a When Rule. 526
More about When Condition Rules. 527
Performing unit testing on a When Rule. 528
Debugging When Rules with the Tracer tool. 530
Defining Conditions in the Condition builder. 531
Expression Builder. 535

PEGA ©2025 Pegasystems Inc

Building Expressions with the Expression Builder. 537
Expressions. 545
Examples of Expressions. 547
Constants in Expressions. 549
Functions in Expressions. 555
Displaying all Functions in the Expression Builder. 566
Defining URL patterns for Cases. 567
Developing applications in Branches. 573
Implementing branched application development. 575
Creating a development application. 576
Adding development Branches to your application. 578
Final Rules and utility Functions in branched development. 580
Developing Branches with libraries and Functions. 580
Adding system Branches. 581
Creating Branch Rulesets. 581
Branch operations. 583
Setting Branch development preferences. 583
Viewing Branch quality and Branch contents. 584
Understanding Branch quality metrics. 585
Reordering Branches. 588
Locking a Branch. 589
Packaging a Branch. 590
Removing a Branch from an application. 590
Deleting a Branch from the system. 591
Branch reviews. 592
Creating a Branch review. 594
Modifying a Branch review. 594
Deleting a Branch review. 595
Closing a Branch review. 595
Reopening a Branch review. 596
Reviewing Branches. 596
Merging Branches into target Rulesets. 597

PEGA ©2025 Pegasystems Inc

Conflicts and warnings in the Merge Branches wizard. 600
Conflicts in the Merge Branches wizard. 600
Resolving conflicts in the Merge Branches wizard. 602
Addressing warnings in the Merge Branches wizard. 603
Branch merge process customization. 610
Branches and unlocked Rulesets. 615
Branches and Branch Rulesets. 618
Developing applications in Workspaces. 621
Delivering application documentation. 630
Documenting your application. 634
Creating project documents for stakeholders. 635
Exporting a Data Model into a document. 635
Contents of application overview and Data Model documents. 636
Creating project documents for stakeholders in Dev Studio. 637
Changing the style of a project document. 639
Changing the structure of a project document. 639
Creating a guide for application developers and administrators. 640
Describing features to end users. 648
Creating content for a guided tour. 649
Associating content with a feature in a guided tour. 650
Integrating a guided tour with your application. 651
Troubleshooting tools and techniques. 653
Improving your compliance score. 658
Analyzing a compliance score. 660
Resolving an application warning. 661
Justifying an application warning. 662
Reviewing user interface components. 663
Exporting compliance data. 664
Scheduling a compliance report. 665
Monitoring project compliance. 667
Compliance score logic. 667
Unit testing individual Rules. 668

PEGA ©2025 Pegasystems Inc

Troubleshooting newly created Rules. 670
Application debugging by using the Tracer tool. 671
Tracer best practices. 673
Tracer disk usage FAQ. 675
Configuring trace conditions. 679
Configuring Tracer settings. 680
Tracer events to trace. 682
Tracer event types to trace. 684
Rulesets to trace. 687
Reducing Tracer event output. 687
Setting the number of lines to display in Tracer. 688
Adjusting the buffer size of the Tracer header. 689
Tracer event queue overflows. 689
Sending Tracer output to a file. 690
Setting breakpoints in the Tracer tool. 691
Setting watch variables in the Tracer tool. 692
Selecting a session to trace. 693
Tracing and capturing events. 694
Tracing services. 695
Cluster-wide tracing of service Rules. 697
Viewing Tracer results in the Tracer window. 698
Offline debugging by using Pega-TracerViewer. 704
Clipboard tool. 708
Using the Clipboard tool. 711
Setting property values with the Clipboard tool. 719
Adding a user page to the Clipboard. 721
Measuring Clipboard size. 723
Indirect pages. 723
Viewing Clipboard Pages created by unit testing a Rule. 724
Troubleshooting long-lived Clipboard Pages. 725
Analyzing application performance with PAL. 726
My Alerts display. 729

©2025 Pegasystems Inc

Parsing the alert log. 731
Suppressing sensitive data in alerts. 732
Summarizing the alert log. 733
Alerts. 733
Alert format. 734
Viewing and resolving errors. 737
Viewing in-progress and completed wizards. 737
Creating connector simulators. 738
The DateTime parse tester tool. 739
About the bulk Revalidate and Save tool. 741
Validation tool. 743
Using the Revalidate and Save tool. 745
Application Structure landing page. 747
Ruleset Stack tab. 748
Ruleset Prerequisites tab. 752
Referencing Applications tab. 753
Access Groups & Users tab. 753
Other Applications tab. 755
Pega GenAI Autopilot in application development. 759

©2025 Pegasystems Inc

Low-code application development

Turn creating applications into a low-code and user-friendly experience by using the tools that Pega Platform™ offers. With Pega Platform, you can deliver flexible applications that help your customers reach business goals in dynamically changing scenarios, in an efficient way.

When you develop your application by using the digital solutions that Pega Platform provides, you can approach your project delivery holistically and efficiently at the same time. Implement the following techniques to achieve your results faster:

Planning flexible Microjourneys

When you design your application, focus on a goal that your customers want to achieve. In dynamically changing circumstances, predicting a fixed path through a business process might be difficult and lead to inefficient results. Pega Platform refers to a path that leads to a successful resolution as a Microjourney, and supports creating flexible templates for Microjourneys that can adjust to events that take place on the way. In App Studio, you can clearly visualize the main elements of your Microjourney: the people that are involved in the process, the channels that provide communication with your application, and the data that is required to meet the goal. Consequently, you can think about flexible sets of actions that might be useful during the processing of your Microjourney. Visualization of the essential parts of your business processes can include also non-technical stakeholders in the planning process.

For more information, see Creating a Microjourney for customer success.

The following figure shows a visualization of a recruitment process that depicts main stages, people involved in the process and channels that they use to provide and work with information, and data that the process requires. You can conveniently build such process templates in App Studio.

©2025 Pegasystems Inc

Recruitment process

Adopting feature-driven development

Develop capabilities in the context of a feature to maintain functional requirements and project status directly in your application. You associate features with elements of your application that you want to develop, for example, a language pack. For enhanced project management and effort tracking purposes, you can use features to estimate project costs and schedules before you start actual development.

For more information, see Adopting feature-driven development.

The following figure shows an overview of features in an application. The `Reporting and monitoring` feature includes two subfeatures: `Business reporting` and `Analytics monitoring`. The overview also shows progress bar for the development of each feature along with associated user stories, bugs, and feedback items.

©2025 Pegasystems Inc

Feature overview

Reusing build-on capabilities

Reuse the functionality that you inherit from built-on applications, so that you can focus your development efforts on unique features and capabilities. For example, you can create an application for banking operations, and then reuse parts of it in applications for reviewing mortgage requests and loan requests. For those three applications, the basic process of collecting and validating information that customers provide might be the same. As a result, you save time but also deliver flexible applications that you can edit independently from each other if your business requirements change, for example, when the process for reviewing loan requests needs additional elements.

For more information, see Adding built-on applications.

In the following graph, the Mortgage requests application is built on Process claims, Banking operations, and default PegaRULES applications. As a result, you can reuse elements of all built on applications in the Mortgage requests application, saving money and development time.

©2025 Pegasystems Inc

Built on application

Engaging and communicating with stakeholders and team members

Communication is a crucial part of developing applications. Starting with application planning, App Studio offers tools to visualize business processes so that stakeholders from outside IT can be involved in designing a Microjourney – a path towards customer success. Apart from that, Pega Platform supports the uninterrupted exchange of information for development team members, as well as for users who process cases in your application. Enhance collaboration with shareable documents, spaces that gather professionally connected users, and messages that users can post in a Pulse gadget.

For more information, see Creating a Microjourney for customer success and Collaborating on Cases.

The following figure shows a sample Pulse thread between developers created in App Studio. When you post Pulse messages, you can use various formatting, attach documents, and insert hyperlinks for enhanced exchange of information.

©2025 Pegasystems Inc

Messages in Pulse

- **Introduction to application development**
- **Building your first application**
- **Understanding Personas and access management**
- **Designing applications for reuse and extension**
- **Building logic and calculating values in your application**
- **Developing applications in Branches**
- **Delivering application documentation**
- **Troubleshooting tools and techniques**

PEGA ©2025 Pegasystems Inc

- **Pega GenAI Autopilot in application development**

- Case Management
- Traditional UI

Introduction to application development

Start building your applications in a user-friendly and seamless way by implementing low-code digital solutions. Save time when you create your applications by using templates, develop various aspects of your application by engaging collaborators with different skills and roles, and increase work efficiency by configuring your application for reuse. To meet your unique needs, use the Pega Platform authoring Portals.

Application development considerations

Before you start developing your application, consider the following factors:

Your business goal

In many situations, predicting an exact course of action for application development leads to fixed and inefficient solutions. With Pega Platform, you can avoid that and instead create agile applications that adjust to dynamically changing needs. center your application development around a goal that you want to achieve to ensure that work can reach a successful resolution under any circumstances.

People in your business process

Determine what types of users need to access your application, and then define the content that is appropriate for every user type. For example, a manager might need access more features than a regular worker. Additionally, consider what communication Channels, such as email or chat bot, your users might need, and what devices they use to process work.

Data in your business process

Every business Case requires data to reach a resolution. Think about the information that you need to collect from users, as well as the ways of gathering data. To save time, define how you can organize and reuse data in your

©2025 Pegasystems Inc

application. For example, instead of storing separate data objects for every piece of personal information about a user, such as a name, surname, and phone number, you can gather all this information in a personal details data object. Then, you can update or reference one data object instead of multiple different instances.

Reusable elements

You can save considerable amounts of time and resources during application development by reusing various assets. For example, if you already have an application to review loan requests, you can reuse any relevant elements to develop an application for reviewing mortgage requests, or performing other banking operations.

Authoring Portals

Pega Platform offers multiple tools that you can use based on your experience, role, and level of expertise. You can use the following authoring Portals to develop your applications:

App Studio

Select this Portal if you are a developer, front-end developer, business analyst, or data engineer. App Studio holds a great variety of low-code, time-saving solutions for you to implement during your application development. With a real-time UI design, any person that is involved in a business process can contribute to application planning and development.

For more information, see App Studio overview and Creating a Microjourney for customer success.

Dev Studio

Select this Portal if you are an experienced application developer, a security administrator, or an account administrator. Dev Studio offers tools for advanced application configuration, such as versioning, Rules management, or branched development.

PEGA ©2025 Pegasystems Inc

For more information, see Dev Studio overview.

Admin Studio

Select this Portal if you are a system administrator. In Admin Studio, you can monitor and manage your system resources, such as nodes and Background Processes.

For more information, see Admin Studio overview.

Prediction Studio

Select this Portal if you are a data scientist. In Prediction Studio, you can develop, monitor, and adjust models for analyzing customer interactions and communications to predict their future behavior.

For more information, see Prediction Studio overview.

The following figure summarizes factors and authoring Portals to consider before you start creating your applications:

Pega Platform ecosystem

- **App Studio overview**
- **Dev Studio overview**

©2025 Pegasystems Inc

- **Admin Studio overview**
- **Prediction Studio overview**
- **Changing your studio**
- **Exploring application-editing mode**

Related concepts

- Case Management
- Traditional UI
- Data Modeling
- Understanding Personas and access management
- Creating a Microjourney for customer success
- Designing applications for reuse and extension
- Troubleshooting tools and techniques

App Studio overview

Turn your application development into a no-code, user-friendly experience by working in App Studio. In this authoring environment, you can configure the main elements of your applications - templates for your business processes, Personas that are involved in the processes, interaction Channels, and data.

App Studio purpose

App Studio provides you with tools to develop your applications in a way that is convenient and understandable for business representatives and citizen developers with little or no knowledge of any programming language. By using App Studio, business members of your organization can be actively engaged in all stages of the application development process.

In App Studio, you can prepare the following application elements:

PEGA ©2025 Pegasystems Inc

- reusable templates for your business processes that Pega Platform refers to as Case Types, and that users complete at run time to reach a successful outcome.

  For more information, see Adding Case Types to organize work.

- Personas that you create to visualize groups of users that interact with your application, such as customers, workers, or managers. You can define access settings for each group so that users view and process only relevant information.

  For more information, see Adding Personas to organize users.

- data objects to ensure that the users have information required to resolve Cases.

  For more information, see Adding data objects to organize data.

The approach of building your business processes on Case Types, Personas, and data objects is a part of the Pega Express delivery approach, the method of delivering projects in a no-code and user-oriented way. In the Pega Express delivery approach, you use App Studio to model your business processes by creating Case Types, Personas, and data objects. A single business process that results in an expected outcome is a Microjourney®.
You can use App Studio to create usable and scalable applications, whose elements include:

- Reusable templates for your business processes that Pega Platform™ refers to as Case Types, and that users complete at run time to reach a successful outcome. For enhanced design of your business processes, you can supplement your Case Types with Personas that you create to visualize groups of users that interact with your application, such as customers, workers, or managers, as well as data objects to ensure that the users have the information that is required for them to resolve Cases.

  For more information, see Adding Case Types to organize work.

PEGA ©2025 Pegasystems Inc

- A user-friendly and intuitive UI. Additionally, if you build your application on Constellation, you can deliver software even faster by taking advantage of numerous built-in features, such as default accessibility standards.

  For more information, see Constellation UI.

- Data objects and integrations that you implement to ensure that your application can source data both from internal and external systems of record.

  For more information, see Data management and integration.

App Studio roles and collaboration

App Studio targets business analysts, citizen application developers, front-end developers, and data engineers. For improved communication between developers and stakeholders, App Studio supports real-time UI design, so that anyone involved in a business process can contribute to application planning and development. This approach has been found to help achieve the best effects when providing IT solutions for business.

Thanks to a low-code interface, App Studio enhances collaboration between IT and business roles, providing the following benefits:

- A more holistic and aligned application development process
- A language shared by both business and technical representatives
- Access to a wide range of people who can understand, interpret, and author processes and applications
- Business roles that can clearly communicate more complex requirements that need addressing by developers with a deeper knowledge of Pega Platform

Reuse of assets in App Studio

For faster and more efficient application development, App Studio supports reusing assets in the following ways:

PEGA ©2025 Pegasystems Inc

Reusing assets across multiple applications

Thanks to the Reuse Library that App Studio includes, you can conveniently manage and edit Case Types, data objects, and business logic Rules that come from other applications in your application stack. As a result, you avoid duplicating assets that are common to multiple applications in your enterprise, such as a data object that stores the personal information of a customer.

The following figure shows the **Data objects** tab in the Reuse Library:

Case Types in the Reuse Library

For more information, see Managing reusable assets in the Reuse Library.

Reusing assets between Studios

While responding to complex business scenarios, you can complement your application development in App Studio with more advanced solutions designed in Dev Studio. By marking an asset as a relevant record in Dev Studio, you make it available for reuse in App Studio, for example, for citizen developers and business analysts. As a result, you accelerate your application development by engaging more people and providing accurate tools for their roles. Pega Platform promotes the approach of creating the building blocks of your application in Dev Studio, and then assembling them in App Studio, taking advantage of the modular reuse.

For more information, see Relevant records for Rule reuse and Reusable elements in App Studio – modular approach in enterprise reuse.

©2025 Pegasystems Inc

App Studio tools

In App Studio, you can find multiple tools for developing and maintaining applications. On the **Overview** page, you can find basic information about your application, for example, the Case Types and Personas that your application includes. By exploring work areas, you can quickly view and create the main parts of your application, such as Case Types, Personas, Channels, and UI elements. Each work area offers no-code solutions that you can use to model your business processes in the most efficient way. The following video shows what you can find in each work area:

To see the results of your work at run time, you can preview an application display on a device, and launch the Portals associated with your application. While in a preview, you can turn on the design mode to conveniently make any necessary adjustments by interacting directly with the UI elements. Apart from building applications, in App Studio you can work with tools to collaborate with other developers and track your development work, for example, by posting messages and toggling Agile Workbench. To ensure that you maintain the good health and quality of your application, explore the options that you can use to monitor and maintain your application performance. These tools include Clipboard, Tracer, and Accessibility Inspector.

In App Studio, you can also take advantage of the latest AI solutions that Pega Platform offers, such as Pega GenAI Autopilot™, which supports you in your application development process by providing instructions and related information, or Pega GenAI Coach™ which automates and enhances the experience for users at run time. The following video shows how Autopilot helps you create a Case Type:

©2025 Pegasystems Inc

Creating a Case Type from Pega Autopilot

For more information about Pega GenAI tools, see Pega GenAI Autopilot in application development and Enabling Pega GenAI Coach in a Case Type or Data Object.

App Studio and Dev Studio

App Studio and Dev Studio are authoring environments that are included as part of Pega Platform, which you use to create applications. Using both of the Studios in specific scenarios helps you maximize the efficiency of your application development process. The following table describes use cases typically suited to each Studio:

App Studio	Dev Studio
Application design	Complex business Rules

©2025 Pegasystems Inc

App Studio	Dev Studio
Case Types design	Creating reusable functionalities and making them available in App Studio
Data Modeling	Complex configurations of integrations
Managing UI and UX of Cases	Complex configurations of data pages
Basic business Rules	
Channel management	
Localization	
Managing reusable assets in the Reuse Library	
Exploring data and managing Insights	
Managing Configuration Sets	

As a best practice, perform as much of your work as possible in App Studio. Switch to Dev Studio to manage complex configurations and prepare advanced solutions, that you can later mark for reuse in App Studio.

App Studio access

To access App Studio, you associate the *pxExpress* Portal with your Access Group. For more information, see Granting Portal access to an Access Group.

From App Studio you can switch to another Studio at any time and change the tools and features that are available in your work environment. For more information, see Changing your studio.

For relevant learning materials, see the App Studio module on Pega Academy.

What to do next: Start planning and implementing your Microjourney. See Creating a Microjourney for customer success.

PEGA ©2025 Pegasystems Inc

Related concepts

- Creating a Microjourney for customer success
- The Microjourney in the Pega Express delivery approach FAQ
- Dev Studio overview
- Admin Studio overview
- Prediction Studio overview
- Agile Workbench

Related information

- Workspaces

Dev Studio overview

Configure advanced technical options for your applications in a low-code and goal-oriented way by exploring the tools available in Dev Studio. You can use Dev Studio to save time by managing the reusable resources in your application in a more detailed way, and to configure advanced settings for applications that focus on complex or exceptional business Cases.

Dev Studio purpose and roles

Dev Studio contains tools that you can use to develop advanced aspects of your application in a convenient, low-code way. In many scenarios, Dev Studio expands the possibilities that App Studio offers, so that you can create applications for unique, complex, and less common business processes. Dev Studio combines access to all of the capabilities that you might need when you create an application, such as Case Management, security, reporting, mobile, and UI, so that you can provide end-to-end digital solutions for your projects.

The focus in Dev Studio is on advanced functionality, such as system settings and Rules, with access to Rule forms and Rules management. In Dev Studio, you can also manage security, versioning, and source control of your application. Target Dev Studio users

PEGA ©2025 Pegasystems Inc

include experienced application developers, security administrators, and account administrators.

App Studio and Dev Studio

App Studio and Dev Studio are authoring environments that are included as part of Pega Platform, which you use to create applications. Using both of the Studios in specific scenarios helps you maximize the efficiency of your application development process. The following table describes use cases typically suited to each Studio:

App Studio	Dev Studio
Application design	Complex business Rules
Case Types design	Creating reusable functionalities and making them available in App Studio
Data Modeling	Complex configurations of integrations
Managing UI and UX of Cases	Complex configurations of data pages
Basic business Rules	
Channel management	
Localization	
Managing reusable assets in the Reuse Library	
Exploring data and managing Insights	
Managing Configuration Sets	

As a best practice, perform as much of your work as possible in App Studio. Switch to Dev Studio to manage complex configurations and prepare advanced solutions, that you can later mark for reuse in App Studio.

Dev Studio tools

Dev Studio provides you with tools to first configure and than maintain your application. On the **Home** tab, you can quickly view recent guardrail warnings to check

©2025 Pegasystems Inc

the compliance as well as the security status of your applications. By exploring the various work areas, you can access and manage the resources that your application includes. For example, you can view, edit, and create the reusable templates of your business processes that Pega Platform refers to as Case Types. The Data types work area helps you manage data for your application. By creating data types and supplementing them with Data Pages, you ensure that users of your application have the information they require to resolve business processes. In the App and Records areas, you can find the elements that are the technical building blocks of your application – the Rules, classes, and Branches. You can also quickly preview the results of your work at run time by launching a Portal that you associate with your application. As a result, you can create applications that precisely meet your unique business needs.

Apart from building applications, Dev Studio provides you with tools for collaborating with other developers and tracking your development work, for example, by posting Pulse messages and opening Agile Workbench. To ensure that you maintain the good health and quality of your application, you can explore different options for monitoring and maintaining your application performance with tools such as the Clipboard, Tracer, and Accessibility Inspector.

Dev Studio access

To access Dev Studio, ensure that you associate the Developer Portal with your Access Group. For more information, see Granting Portal access to an Access Group.

From Dev Studio you can switch to another studio at any time and change the tools and features that are available in your work environment. For more information, see Changing your studio.

Admin Studio overview

Monitor and manage resources in your system more efficiently by using the tools that Admin Studio provides. You can use Admin Studio to access run-time information and configuration options for the resources that you create in Dev Studio.

©2025 Pegasystems Inc

Admin Studio purpose and roles

Admin Studio focuses on system administration and provides tools for monitoring and debugging resources in your system. Target users of Admin Studio are system administrators.

Admin Studio tools

By exploring the work areas that Admin Studio includes, you can maintain the top performance of your application. In the **Overview** work area, you can view data about your system and analyze information about all of the resources available in your environment. For example, you can check how many nodes your application includes in total, and how many of those nodes are currently running. The **Resources** work area provides access to landing pages that focus on specific resources in your system. Depending on the resource, apart from viewing a list of resource instances available in your system, you can also investigate and fix issues, start and stop resources, or even remove resources from the system. In Admin Studio, you can also configure the settings for a mobile build server and manage the cache for users of your application's mobile Channel, as well as preview system details, available nodes, other applications, and Java classes.

Admin Studio access

To access Admin Studio, ensure that you associate the *pxAdminStudio* Portal with your Access Group. For more information, see Granting Portal access to an Access Group.

From Admin Studio you can switch to another studio at any time and change the tools and features that are available in your work environment. For more information, see Changing your studio.

For learning materials about Admin Studio, see an Admin Studio topic on Pega Academy.

Related concepts

©2025 Pegasystems Inc

- Managing system resources

Prediction Studio overview

Prediction Studio is an authoring environment in which you can control the life cycle of AI and machine learning models (such as model building, monitoring, and update). From Prediction Studio, you can also manage additional resources, such as predictor fields, Data Sets, taxonomies, and sentiment lexicons.

To access Prediction Studio, you must specify *pxPredictionStudio* as one of the portals associated with your Access Group. From Prediction Studio, you can switch to another studio any time and change the tools and features that are available in your work environment. For more information, see Changing your studio.

Overview of Prediction Studio

The following components are available in Prediction Studio:

©2025 Pegasystems Inc

Prediction Studio header

The header at the top of Prediction Studio enables you to view the name of the current application, perform a search across the Rules in that studio, view the online help, and toggle Agile Workbench.

Landing page header

The page header at the top displays the name of the current landing page, for example, **Predictions** and enables you to perform a number of common actions such as viewing model reports, clearing deleted models, and so on. This toolbar also allows you to add models or additional resources.

Navigation pane

The navigation pane provides quick access to the following landing pages:

Predictions

On the Predictions landing page, you can create Predictions by answering several questions about what you want to predict. You can access, manage, and run existing Predictions within your application. You can also perform Machine Learning Operations (MLOps) to replace models and scorecards in Predictions with different models, and approve the model updates for deployment to your production environment.

For more information about Predictions, see Creating and managing Predictions.

Models

On the Models landing page, you can access, sort, and manage Adaptive and Predictive Models, as well as text analytics models within your application. By default, the models are displayed as tiles. Each model tile contains an icon for quick identification of model type, the model name, and the indication of whether the model is completed or being built. For more information, see Developing and managing models.

PEGA ©2025 Pegasystems Inc

Data

On the Data landing page, you can access the data dictionary that provides an overview of all predictor fields. You can create and manage Data Sets or interaction history summaries. In addition, you can access resources such as taxonomies or sentiment lexicons that provide features for building machine learning models. For more information, see Managing data.

Reports

The Reports landing page contains relevant AI-related reports, for example, the ADM predictor importance report. Administrators can create custom reports. For more information, see Viewing reports.

Settings

The Settings landing page contains the global settings for model development, model transparency policy, machine learning services, monitoring and notifications. For more information, see:

- Setting up your environment
- Defining global settings for text analytics

Notifications

The Notifications icon indicates the number of new messages. By clicking the icon, you can display the Notifications detail overlay with summaries of the most recent notifications. The detail overlay includes a link to the Notifications landing page that contains the complete list of notifications about the performance of your Predictions, Adaptive Models, and Predictive Models. For more information, see Viewing Prediction Studio notifications.

Prediction Studio transitions

You can quickly transition between Prediction Studio and business portals such as Pega Customer Decision Hub™. When in Prediction Studio, you can click the **Back** widget to return to a business portal with no context change.

©2025 Pegasystems Inc

Prediction Studio permissions

System architects and data scientists configure models and Predictions in lower environments, for example, in the Development environment. Deployment managers then migrate these changes to the Production environment.

To prevent users from changing the configuration in a live system, Prediction Studio is read-only in the Production environment. For special uses cases, an administrator can grant the read-and-write permissions to selected users. To do this, the administrator adds the *pyEditPredictionStudio* privilege with production level 5 to the *@baseclass* of an Access Role in the user's Access Group.

For more information, see Security of Access Roles and Specifying privileges for an Access of Role to Object rule.

- Developing and managing models

Changing your studio

For the complete and multidimensional development of your application, switch from one studio to another to change the tools and features that are available in your work environment. For example, you can create resources such as job schedulers in Dev Studio, and then manage and monitor those resources in Admin Studio.

1. In the header of your studio, click the **Switch Studio** menu.
   The label for this menu is the name of your current studio, for example, App Studio.
2. Select the studio in which you want to work.
   You have access only to the studios that are relevant to your role. For more information, see Granting Portal access to an Access Group.
   You have access only to the studios that are relevant to your role. For more information, contact your security administrator.
   You have access only to the studios that are relevant to your role. For more information, review your Access Groups in Dev Studio.

©2025 Pegasystems Inc

Prediction Studio is a role-based studio for data scientists. You have access only to the studios that are relevant to your role. For more information, contact your system administrator.

- Adding Personas to organize users

Exploring application-editing mode

Explore application-editing mode to learn about the tools and information that support application development.

Application-editing mode is enabled by default.

1. If you have turned off application-editing mode, turn it on.
   a. In the footer of App Studio, click the **Open runtime toolbar** icon.
   b. Click **Turn editing on**.
2. In the header of App Studio, click the **Resources** icon.
3. Click **Take a tour**.
4. Follow the prompts on the screen to discover the different tools that you can use to develop your application.

> **What to do next:**
>
> Leave application-editing mode on while you set up your environment.

- Building your first application

Building your first application

Create low-code applications in a user-friendly and intuitive way by working with Pega Platform™. By using the various out-of-the-box tools, you can define customer journeys, develop applications, track your progress, and communicate with team members and stakeholders.

©2025 Pegasystems Inc

Application creation process

After the conceptual work of defining who will use the application and how, and sketching the resulting minimum lovable product (MLP), the next step is to actually build the application. That means creating the application in Pega Platform, preparing the user stories that will guide the development, customizing the application structure and components, and ensuring that the application meets security requirements before it is ready for deployment. Pega Platform provides you with the tools to structure the entire process.

Application templates

You create an application by using a template. You can use either an out-of-the-box template, or your own template. Often the best, most time-saving option is to reuse one of your existing applications as a template. This greatly simplifies the design process, and helps ensure matching structure and requirements, as needed.

The following screenshot shows a sample selection of templates to choose from:

Choosing a template for a new application

©2025 Pegasystems Inc

For example:

When creating applications for two departments that follow similar processes, such as mortgages and personal loans, you can use one application as a template for another. Reusing existing applications as templates for new ones saves time and effort because the new application can immediately have access to the same assets as the original one.

Developer collaboration

You can add collaborators to your application, both end users (run-time users) and developers (design-time users). Their privileges in the system are determined by their role, meaning, the Persona that you assign to a given collaborator. As a best practice, define the Personas before this stage. For more information, see Creating Personas.

You can work on several applications at the same time, and switch between them both in App Studio and Dev Studio.

For example:

For an application to process loan requests, you can create the following Personas:

Run-time Personas	Design-time Personas
Applicant	Developer
Underwriter	Administrator
Customer Service Representative	
Manager	

The following video shows managing Personas in App Studio:

©2025 Pegasystems Inc

Developer Assistant

You can organize the work using the AI-powered developer aids that are available in the App Studio Developer Assistant pane. The Developer Assistant pane provides access to documentation, automatic suggestions based on analysis of your application's structure, integration with Agile Workbench, as well as detailed checklists for various areas of development, including security. The following figure shows the four different tabs in the Developer Assistant pane, labelled **Guidance**, **Knowledge**, **Work**, and **Checklists**:

The four tabs of the Developer Assistant pane

For more information, see Using development aids.

Continuous development

To fine-tune and enrich your application, use the options available in the Application Definition. You can access the Application Definition in Dev Studio. Here you can configure the look and feel of the application by customizing the application skin, the

©2025 Pegasystems Inc

URL, and the password. You can also manage advanced settings for the application structure and behavior, for example:

- Reuse elements and assets from other applications by adding built-on applications.
- Add more components, either ones you made yourself or ready-made components available on Pega Marketplace.
- Manage Branch development, for example separate sandboxes for developers to avoid conflicts.
- Organize Rules into Rulesets.
- Add application guides (checklists) for your developers to follow.
- Allow users to modify Rules in a production environment, define how the application renders the UI, or manage system resources and background processing.

Security for public-facing applications

Public-facing applications need special scrutiny in terms of security. To help you assess your application, Pega provides a list of minimum security requirements for deployment. These requirements concern the following areas:

- Best practices for security
- Authentication
- Restricting users' access to information about other users
- Protecting sensitive data
- Avoiding showing logos and branding that would expose server information

> ⚠ **IMPORTANT:** Fulfilling these requirements significantly improves the security of your application. Do not deploy a public-facing application until you meet all Pega security requirements.

To help improve the security of the application, use the Application Security Checklist. For more information, see Assessing your application using the Security Checklist.

PEGA ©2025 Pegasystems Inc

Start building your application by performing the following tasks:

- **Creating a new application from a Blueprint**
- **Exporting an existing application as a Blueprint**
- **Importing a Blueprint into an existing application**
- **Implementing a Blueprint-generated application**
- **Using development aids**
- **Creating user stories from checklists (application guides)**
- **Creating applications**
- **Inviting collaborators to your application**
- **Configuring advanced settings for new applications**
- **Switching between applications**
- **Switching between applications in Dev Studio**
- **Exploring the Application Definition**
- **Basic requirements for deploying public-facing applications**
- **Best practices for compliance and updatability**
- **Best practices for Pega Infinity application development**
- **Customizing the App Studio navigation pane**

- Automating work by creating Case Types
- Traditional UI
- Security Checklist
- Low-Code App Builder

PEGA ©2025 Pegasystems Inc

Creating a new application from a Blueprint

Develop the foundations of your new application from a Pega GenAI™-powered definition that you created with Pega GenAI Blueprint™. By importing an existing Blueprint, you significantly accelerate the application development process.

The following video shows the process of designing an application with Blueprint and then importing the design with the New Application wizard:

Before you begin:

- Create a Blueprint of your application with Pega GenAI Blueprint and download it to your device. For more information, see https://pega.com/blueprint.
- If you plan to use an existing application as a template, configure the built-on application for reuse. For more information, see Configuring applications for reuse.
- If your Blueprint contains Unicode characters, you must enable the *EnableUnicodeRuleNames* dynamic system setting. For more information, see Enabling Unicode character support for object names.

You begin building an application from a Blueprint by uploading a BLUEPRINT file in the New Application wizard, and then you make the necessary adjustments and select the assets to use.

1. In the header of App Studio, click the name of the application, and then click **New Application**.
2. Click **Build from a Blueprint**, as shown in the following figure:

©2025 Pegasystems Inc

Building a new application from a Blueprint

3. In **Step 1 - Select a Blueprint**, select the source of your Blueprint:
   - To upload a new Blueprint, click **Select a Blueprint to upload**, and then browse for the BLUEPRINT file on your device.
   - To use a Blueprint that is already in your system, click **Select an uploaded Blueprint**, and then select the Blueprint that you want to use.

> **Result:**
>
> The basic structure of your new application is ready to build, including Case Types, data objects, and Personas.

4. **Optional:** To change the application name, in the **Application name** field, enter a meaningful name for your application.
5. **Optional:** To use an application from which to reuse assets, in the **Built-on application** list, select the application.

> ⚠ **IMPORTANT:** Ensure that all the Rulesets in the built-on application are locked. Otherwise, saving Rules in App Studio results in errors because Branch preferences are active by default. For more information, see Configuring Ruleset version settings and Setting Branch development preferences.

©2025 Pegasystems Inc

The following figure shows the settings for an application that can reuse assets from an existing application:

Configuring a Blueprint

6. Click Next.
7. In **Step 2 - Case Types**, in the Build action list, select how to create each Case Type from your Blueprint:

Choices	Actions
Create a new Case Type as specified in the Blueprint	Select Build from Blueprint.
Create a new Case Type that inherits assets from an existing Case Type in the built-on application **NOTE: Available only if you selected a different built-on application in step 4.**	a. Select Inherit from existing. b. In the Inherits from list, select the source Case Type.

©2025 Pegasystems Inc

Choices	Actions
Omit the data object from your new application	Select Do not build.

8. **Optional:** To copy Case Types from the built-on application into your new application, click Clone an existing Case Type, select the Case Types that you want to copy, and then click Submit.

> **NOTE:** This option is available only if you selected a different built-on application in step 4.

9. Click Next.
10. In **Step 3 - Data Objects**, in the Build action list, select how to create each data object from your Blueprint:

Choices	Actions
Create a new data object as specified in the Blueprint	Select Build from Blueprint.
Create a new data object that inherits assets from a data object in the built-on application **NOTE: Available only if you selected a different built-on application in step 4.**	a. Select Inherit from existing. b. In the Inherits from list, select the source data object.
Create a data object that is linked to the built-on application and reuses assets from a data object in that	a. Select Reuse from existing. b. In the Reuse from list, select the data object whose assets you want to reuse.

©2025 Pegasystems Inc

Choices	Actions
application but does not have its own class in your new application **NOTE: Available only if you selected a different built-on application in step 4**	
Omit the data object from your new application	Select Do not build.

11. **Optional:** To copy data objects from the built-on application into your new application, click Add an existing data object, select the data objects that you want to copy, and then click Submit.

> **NOTE:** This option is available only if you selected a different built-on application in step 4.

12. Click Next.
13. In **Step 4 - Personas**, for each Persona from your Blueprint, in the Build action list, select how to create the Persona in your new application:

Choices	Actions
Create a new Persona as specified in the Blueprint	Select Build from Blueprint.
Omit the Persona from your new application	Select Do not build.

14. Click Next.

©2025 Pegasystems Inc

15. In **Step 5 - Configurations**, review the basic settings and structure of your new application, and then ensure that they are correct for your intended business use case:
    a. In the **Application settings** section, ensure that the application ID, version, and base language are correct.
    b. In the **Organizational settings** section, ensure that the Organization, Division, and unit names are correct.
    c. In the **Class layers** section, ensure that the Class layers that your use case requires are included in the application.
    For more information, see Understanding class layers.
    d. In the **Case classes** section, ensure that the list of Case classes is correct.
    For more information, see Managing application Cases and data.
    e. In the **Data classes** section, ensure that the list of data classes is correct.
    For more information, see Managing application Cases and data.
16. Click Submit.

Result:

The system creates a new application from the Blueprint that you uploaded. All the assets of the new application, except for Access Roles, are created in a Branch. The newly created application is displayed.

> **NOTE:** By default, all base Rulesets in the new application are locked. If you want to merge any changes to the Rulesets, you must use the following password: pega.

Related tasks

- Importing a Blueprint into an existing application

©2025 Pegasystems Inc

Exporting an existing application as a Blueprint

Create a shareable template of your existing application by exporting the application as a Pega GenAI Blueprint™. For example, an organization has a customer service application that significantly reduces response times and improves customer satisfaction. Users can export the existing application for reuse in sales and marketing applications to save development time.

1. In the navigation pane of App Studio, click **Overview**.
2. In the Pega GenAI Blueprint section, click **Export as Blueprint**, as shown in the following figure:

Exporting an application to a Blueprint

3. In the dialog box that is displayed, select a location to save the BLUEPRINT file.

Result:

The system creates a Blueprint of the application.

©2025 Pegasystems Inc

What to do next:

You can share the Blueprint with other developers, or use it as a template for building similar applications yourself. For more information, see Creating a new application from a Blueprint.

Importing a Blueprint into an existing application

Extend your application by importing a Pega GenAI Blueprint™. As a result, you can quickly begin expanding the scope of business use cases that your existing application covers.

Before you begin:

- Create a Blueprint of your application with Pega GenAI Blueprint and download it to your device. For more information, see https://pega.com/blueprint.
- If you plan to use an existing application as a template, configure the built-on application for reuse. For more information, see Configuring applications for reuse.
- If your Blueprint contains Unicode characters, you must enable the *EnableUnicodeRuleNames* dynamic system setting. For more information, see Enabling Unicode character support for object names.

1. In the navigation pane of App Studio, click **Overview**.
2. In the **Pega GenAI Blueprint** section, click the **Import a Blueprint** link, as shown in the following figure:

©2025 Pegasystems Inc

Import a Blueprint *into your application*

3. In **Step 1 - Select a Blueprint**, select the source of your Blueprint:
   - To upload a new Blueprint, click **Select a Blueprint to upload**, and then browse for the BLUEPRINT file on your device.
   - To use a Blueprint that has already been uploaded, click **Select an uploaded Blueprint**, and then select the Blueprint that you want to use.
4. Click **Next**.
5. In **Step 2 - Case Types**, in the **Build action** list, select how to create each Case Type from your Blueprint:

Choices	Actions
Create the Case Type as defined in the Blueprint	Select **Build from Blueprint**.
Inherit assets from an existing Case Type	a. Select **Inherit from existing**. b. In the **Inherits from** list, select the source Case Type in your existing application.

©2025 Pegasystems Inc

Choices	Actions
	Result: The Case Type inherits assets from a Case Type in your existing application **NOTE:** This option is available only if your current application contains Case Types.
Clone the Lifecycle of an existing Case Type	a. Select **Clone from existing**. b. In the **Clone from** list, select the source Case Type in your existing application. **Result:** The new Case Type reuses the Lifecycle of the Case Type that you selected. **NOTE:** This option is available only if your current application contains Case Types.
Omit the data object from your new application	Select **Do not build**.

6. Click **Next**.

©2025 Pegasystems Inc

7. In **Step 3 - Data Objects**, in the Build action list, select how to create each data object from your Blueprint:

Choices	Actions
Create the data object as specified in the Blueprint	Select Build from Blueprint.
Inherit assets from your existing application	a. Select Inherit from existing. b. In the Inherits from list, select the source data object from your existing application. **Result:** The data object inherits assets from the data object that you selected. **NOTE:** This option is available only if your current application contains data objects.
Link an existing data object	a. Select Reuse from existing. b. In the Reuse from list, select the data object in your existing application whose assets you want to reuse.

©2025 Pegasystems Inc

Choices	Actions
	Result: The data object reuses assets from the data object that you selected. **NOTE:** This option is available only if your current application contains data objects.
Omit the data object	Select Do not build.

8. Click Next.
9. In **Step 4 - Personas**, for each Persona from your Blueprint, in the Build action list, select how to create the Persona in your new application:

Choices	Actions
Create a new Persona as specified in the Blueprint	Select Build from Blueprint.
Omit the Persona from your new application	Select Do not build.

10. Click Next.
11. In **Step 5 - Configurations**, review the basic settings and structure of your new application, and then ensure that they are correct for your intended business use case:
    a. In the **Case classes** section, ensure that the list of Case classes is correct. For more information, see Managing application Cases and data.
    b. In the **Data classes** section, ensure that the list of data classes is correct. For more information, see Managing application Cases and data.

©2025 Pegasystems Inc

12. Click **Submit**.
13. Follow steps 3 to 16 in Creating a new application from a Blueprint.

Result:

The system creates the assets from the Blueprint that you uploaded according to your configuration. All the assets, except for Access Roles, are created in a Branch.

Related tasks

- Exporting an existing application as a Blueprint
- Creating a new application from a Blueprint

Implementing a Blueprint-generated application

Develop a Pega GenAI Blueprint™ into a fully functioning application by following a detailed list of action items.

After you import a Blueprint into Pega Platform™, the system creates user stories that represent the tasks that you must complete to bring the application to production in Agile Workbench. This list of detailed tasks helps you to estimate the effort required to complete the application, aiding in project planning and resource allocation. As a result, you can save development time by prioritizing tasks based on their importance, which helps to ensure an efficient development process.

Before you begin:

Create a Blueprint and import it into Pega Platform. For more information, see the following articles:

- Creating a Blueprint
- Creating a new application from a Blueprint

©2025 Pegasystems Inc

- Extending an existing application with a Blueprint

You continue the development of your application by completing user stories in Agile Workbench. For more information, see Agile Workbench. These user stories are associated with features and subfeatures that the system creates to reflect the structure of the application.

1. On the Overview landing page of App Studio, in the **Application profile** section, click **Manage**.
2. Review the list of features and subfeatures:
   a. Click the **Inventory** tab, and then click **Features**.
      Features represent Case Types and data types in your application, and subfeatures represent Stages in these Case Types.
   b. Ensure that the features are aligned with the correct releases.
      For more information, see Managing application inventory and Reviewing application features.
3. On the **Feature map** tab, click the **Switch to tree view** icon.
   The tab contains a list of features, their subfeatures, and a list of work items, as shown in the following figure:

©2025 Pegasystems Inc

Features and user stories

4. In the **Work items** section, review the list of user stories.
5. Refine the stories to ensure that they reflect the work that is necessary to complete the configuration of the features and subfeatures in your application.

> **For example:**
>
> Increase the complexity of a story that requires configuration of Rules that you are unfamiliar with to **High**, and increase its priority to **Must have** to signify its importance for the usability of the application.

For more information, see Feature-driven development tracking.

6. **Optional:** To configure integration of Agile Workbench with your work management system, for example, Agile Studio, see Integrating Agile Workbench with Pega Agile Studio and Integrating Agile Workbench with Jira in Pega Platform 8.3.x and later.

©2025 Pegasystems Inc

Related concepts

- Agile Workbench

Related tasks

- Using development aids
- Reviewing application features
- Managing application inventory

Related information

- Pega GenAI Blueprint journey

Using development aids

Develop robust applications in App Studio more efficiently by using the Developer Assistant pane. The pane helps you quickly and conveniently access the tasks that you need to complete before the application can go to production, such as completing the Application Security Checklist.

The information is grouped into four tabs:

- The **Knowledge** tab provides access to documentation.
- The **Guidance** tab contains suggestions to improve the application, organized by importance.
- The **Work** tab provides access to Agile Workbench, with feedback and the development status of the application features.
- The **Checklists** tab contains checklists, also called application guides, including the Application Security Checklist.

1. In the header of App Studio, click the **Toggle developer assistant** icon.
2. Review the guidelines and best practices for the successful design and implementation of your application.

©2025 Pegasystems Inc

a. Navigate to the element of your application for which you want to see guardrail compliance suggestions and warnings.

> **For example:**
>
> In a Case Lifecycle, click a Step in a Process.

b. To learn more about creating and configuring Case Types from Pega Documentation, on the **Knowledge** tab, review the content that is relevant to your most recent action, such as an article about a Process in the Case Lifecycle.

> **NOTE:** You can view more articles that are related to the current element by clicking the links on the **Knowledge** tab.

> **NOTE:** You can extend **Knowledge** tab by adding custom articles that you find useful for the Case. For mor information, see Extending the Knowledge tab.

c. On the **Guidance** tab of the Developer Assistant pane, review the list of suggestions.
You can filter the suggestions for improving your application according to severity.

> **NOTE:** At the bottom of each item, you can find the name and type of the Rule to which the guidance applies.

If you have access to Dev Studio, you can click an item to open the Rule to which the suggestion applies in Dev Studio, as shown in the following figure:

©2025 Pegasystems Inc

Guidance of different severity for a Case Type

For more information, see Improving your compliance score.

For relevant training material, see Guardrails.

3. Access Agile Workbench.
   a. Click the **Work** tab.
   b. Manage feedback and the development status of your application.
      For more information, see Agile Workbench.
4. Access and follow application guides.
   a. On the **Checklists** tab, click the application guide that you want to access.

      For example:

      Click **Application** Security Checklist.

   b. Follow the instructions for a task, and then select the **Check** checkbox to mark the task as complete, as in the following figure:

©2025 Pegasystems Inc

A completed and an uncompleted task in the Application security guide

c. **Optional:** To assign a task from a guide to the appropriate developer, create a user story from the task.
   For more information, see Creating user stories from checklists (application guides).

d. **Optional:** To keep the **Last updated by** section current for tasks that must be completed periodically, clear the **Check** field.

©2025 Pegasystems Inc

- **Extending the Knowledge tab**

Related concepts

- Agile Workbench

Related tasks

- Creating a guide for application developers and administrators
- Creating user stories from checklists (application guides)

Extending the Knowledge tab

Provide access to a wider range of documentation that is relevant to your Case by adding custom articles to the **Knowledge** tab in the Developer Assistant pane.

The **Knowledge** tab provides guidance in application development by including links to content that is relevant to your Case, for example Pega articles that cover Data Transforms or the Expression Builder. Extending the **Knowledge** tab with additional articles makes the guidance more personalized and suitable for your Case.

Creating and customizing a decision table based on the existing *pzFetchKnowledgeArticles* decision table that forms the **Knowledge** tab

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Decision Table**.
3. In the list of decision table instances, open *pzDetermineKnowledgeArticle*.
4. Click **Save as**.

> **Result:**
>
> A new tab with decision table settings opens.

©2025 Pegasystems Inc

5. In the **Decision Table Record Configuration** section, in the Label field, enter a meaningful name.
6. Ensure that there is no prefix in the **Identifier** field, so that the Identifier is the same as the Label.
7. In the **Context** section, pick a development Branch from the drop-down menu, and your application from the list.
8. Leave the other values as default. See the following example for reference:

Custom decision table settings

9. Click **Create and open**.
10. In the **pxHelpURI** field, paste the URL of the document that you want to add to the **Knowledge** tab.

	Conditions	Actions	
	Original Shape Name	pyTitle	pxHelpURI
if	Change goal and deadlin	"Setting service-level agreements"	"https://docs.pega.com/search?q=change%20goal%20and%20deadline&f%5B0%5D=product_area
else if	Change stage	"Changing a stage with open assignment	"https://docs.pega.com/search?q=change%20stage&f%5B0%5D=product_area%3APrd78"
else if	Create	"Creating cases"	"https://docs.pega.com/search?q=create%20case&f%5B0%5D=product_area%3APrd78"
else if	Edit details	"Editing application details in App Studio	"https://docs.pega.com/search?q=Edit%20details&f%5B0%5D=product_area%3APrd78"
else if	Create tasks	"Creating a task in pulse"	"https://docs.pega.com/search?q=Create%20tasks&f%5B0%5D=product_area%3APrd78"
else if	Reopen	"Reopening a resolved case"	"https://docs.pega.com/search?q=reopen&f%5B0%5D=product_area%3APrd78"
else if	Resolve case	"Resolving triage cases"	"https://docs.pega.com/search?q=Resolve%20case%20&f%5B0%5D=product_area%3APrd78"
else if	Transfer assignment	"Transferring an assignment"	"https://docs.pega.com/search?q=Transfer%20assignment&f%5B0%5D=product_area%3APrd78"
else if	Collect information	"Collecting information from a user"	"https://docs.pega.com/search?q=collect%20information&f%5B0%5D=product_area%3APrd78"
otherwise		"App Studio overview"	"https://docs.pega.com/search?q=App%20studio&f%5B0%5D=product_area%3APrd78"

Customizable decision table for extending the Knowledge tab

Invoking the custom decision table by using the *pyAddKnowledgeArticles* Data Transform

11. In the navigation pane of Dev Studio, click **Records**.
12. Expand the **Data Model** category, and then click **Data Transform**.

©2025 Pegasystems Inc

13. In the list of Data Transform instances, open *pyAddKnowledgeArticles*.
14. Click **Add a row** and then select the **Set** action.
15. Set the relation as shown below:
    a. In the **Expression** field, enter `Param.InvokeDecisiontable`.
    b. In the **Source** field, enter `@(Pega-RULES:DecisionTable).ObtainValue(tools,Primary,AddCustomArticle)`.

pyAddKnowledgeArticles Function parameters

16. Click **Save**.

Related tasks

- Using development aids

Creating user stories from checklists (application guides)

Assign a task from a checklist (also known as an application guide) as a user story to the appropriate resource to complete the task. For example, assign the Lock application and Rulesets task from the Application security checklist to an experienced application developer.

For more information about application guides, see Creating a guide for application developers and administrators.

1. In the header of App Studio, click the **Toggle developer assistant** icon.
2. On the **Checklists** tab, click the name of the application guide whose tasks you want to assign.

©2025 Pegasystems Inc

For example:

Click Application security checklist, as shown in the following figure:

Opening Application security checklist

3. Next to a task, click More > Create story.
4. In the Owner field, assign the user story to the developer who can complete the work.
5. In the Due field, select the expected due date of the item.
6. In the Complexity drop-down list, select the complexity level of the task.
7. In the **Priority** section, select the urgency of the item.
8. **Optional:** To include additional elements in the user story, such as a screen capture or a media file, in the **Attachments** section, click Add attachment.
9. In the **Acceptance criteria** section, click Add new criteria, and then specify the criteria for the approval of the task.

©2025 Pegasystems Inc

10. Click Save.
11. **Optional:** To create user stories for more tasks in the application guide, repeat steps 3 through 10.

Related tasks

- Creating a guide for application developers and administrators
- Using development aids

Creating applications

Start developing your projects conveniently and intuitively by building your application on an application template. When you create an application on a template, you save time because you reuse key elements of an existing application, such as Case Types or data objects.

You can choose a default application template that Pega Platform™ provides or reuse one of your existing applications as a template. For example, you can create an application to review mortgage requests, and then supply the application with necessary Case Types, data objects, and other elements. Then, you can build an application to review loan requests by using the application to review mortgage requests as a template. As a result, all assets from the application to review mortgage requests are automatically available in the new application to review loan requests.

Before you begin:

- If you want to build an application in App Studio, ensure that you have an Access Group based on *SysAdm4* role or an *pxCreateNewApplication* privilege, and that the production level is equal to or lower than 4. For more information, see Learning about Access Groups, Creating a privilege, and Specifying the production level.
- If you plan to use a custom application template, configure the template for reuse. For more information, see Configuring applications for reuse.

©2025 Pegasystems Inc

> **NOTE:** If you build your application on a custom template, fields to complete and the creation process might vary.
>
> - If you plan to collaborate with users without an operator ID, configure the *Default* email account. For more information, see Configuring outbound email.

For relevant training materials, see the Creating Pega Platform applications module on Pega Academy.

1. Navigate to the New Application wizard:
   - To build an application in App Studio, in the header of App Studio, click the name of your application, and then click **New Application**, as shown in the following figure:

Creating an application in App Studio

   - To build an application in Dev Studio, in the header of Dev Studio, click the name of your application, and then click **New Application**.
2. Choose a type of application to create:
   - To build an application from a Blueprint that you created in Pega GenAI Blueprint™, click **Build from a Blueprint**, and then continue by completing the procedure in Creating a new application from a Blueprint.

©2025 Pegasystems Inc

- To build an application that uses Constellation UI authoring and run-time experience, and is also client-side rendered, click **Build from scratch**, and then select Constellation.

  Constellation is an innovative way of designing applications that currently supports a limited range of Case Management functionalities. For more information, see Constellation UI.

- To build your application on a custom application template, click **Build from an existing application**, click **Continue** under the selected template, and then click **Build with** *application template name*.

  To search for an application template, at the bottom of the page, click **Search all types**.

  For more information, see the implementation guide for your application.

The following figure shows building an application from scratch:

Selecting the way to build an application

The following figure shows the available selection of UI architecture:

©2025 Pegasystems Inc

Constellation UI architecture

3. **Optional:** If you build your application on a custom template, and you want to reuse resources from the application, such as Case Types or data objects, import the resources to your new application:
   a. In the **Select Case Types** section, select the checkbox next to each relevant Case Type.
   b. Click **Continue**.
   c. In the **Select data objects** section, select the checkbox next to each relevant data object.

©2025 Pegasystems Inc

   d. Click **Continue**.
4. In the **Name your application** field, enter a unique name for your application.
5. **Optional:** To configure a custom class structure, organization, or select the base language for your application, click **Advanced configuration**.
   For more information about the settings that you can change, see Configuring advanced settings for new applications.
   The following figure shows sample advanced configuration:

©2025 Pegasystems Inc

Advanced application configuration

6. Click **Create application**.
7. **Optional:** To develop your application more quickly, in the **Optionally, add users** section, create a team:

©2025 Pegasystems Inc

a. In the text field, press the Down arrow key, and then select a user name or an email address.
   If you enter an email address that is new in the system, the system creates a new user.
b. Control which type of access the user has to your application by selecting a role in the list.
c. Click **Add**.

> ⚠ **CAUTION:** If an email account error occurs when you add a user, take note of the user ID and generated password, because the user may not receive these credentials in an email.

The following figure presents a dialog box to invite new users to your application:

Inviting users to an application

You can invite run-time and design-time users to your application.

8. Click **Go to app**.

- Constellation UI
- Configuring advanced application settings
- Styling your application with design systems
- Configuring applications for reuse
- App Studio overview

©2025 Pegasystems Inc

Inviting collaborators to your application

Enhance your application and begin processing your business Cases by inviting collaborators with different skills and roles.

When you invite collaborators, you define the type of access that you grant to the users of your application. Your application illustrates different user and access types by Personas. You can select from Personas that reflect run-time users, and developer roles that correspond with design-time users. For example, you can invite members of a development team and associate them with developer roles, so that the team can start working on your application. You can also invite run-time users to process work in your application and, for example, associate them with a customer service representative (CSR) Persona.

Before you begin:

- Populate the list of available Personas. For more information, see Creating Personas.

  Administrator and author Personas are available by default.

- Define the types of access that different users of your application can have. For more information, see Granting Personas access to Channels and pages.

1. In the navigation pane of App Studio, click **Users**, and then click **User management**.
2. On the **People** tab, in the working area, click **Invite people to your application**, as shown:

©2025 Pegasystems Inc

Inviting collaborators to an application

3. In the **Add users to application** window, in the autocomplete field, press the Down arrow key, and then select a user name or email address.

> **Result:**
>
> If you enter an email address that does not exist as an operator in the system, the system creates a new user.

4. In the list of Personas, define the type of access for the user of your application, by selecting a role:
   - To invite run-time users, select an option from the **Personas** section of the list.
   - To invite design-time users, select an option from the **Developer roles** section of the list.

> **For example:**
>
> Associate the user with a run-time CSR Persona, as shown in the following figure:

©2025 Pegasystems Inc

Selecting a Persona for a user

5. Click **Add**.
6. Send an email with an invitation to the user by clicking **Send invitation**.
7. To invite more users, repeat steps 3 through 6.
8. Close the **Add users to application** dialog box.

What to do next:

Analyze what data your Case requires to reach a resolution by defining data objects. See Adding data objects to organize data.

- Creating a Microjourney for customer success
- Adding Personas to organize users
- Creating Personas
- Creating applications

©2025 Pegasystems Inc

Configuring advanced settings for new applications

To fully benefit from the possibilities that the Pega Situational Layer Cake™ provides, configure advanced settings for your application and create organization layers. To save time, reuse common Processes within your organization, and to adjust the application to your departmental business needs, allow for greater customization of certain elements.

You can create division and unit layers in your application to customize different layers. For example, if your organization includes a Human Resources division and an Engineering division, configure your application to reuse common Processes between the divisions, and change elements that are different for each division. For instance, reuse a Process for calculating expenses, and specify different maximum amounts for each division.

> **Before you begin:**
>
> Create your application. For more information, see Creating applications.

1. In the **Application structure** section, select a structure for your application:
   - To create a generic application that you can reuse or extend later, select Framework.
   - To create a specialized application for a specific business audience, select Implementation.
2. In the Application ID field, enter an identifier for the application, for example, `HiringApp`.
   An identifier starts with a letter, and contains only letters, numbers, and hyphens. If you set an application name, the system generates an identifier based on the name.
3. In the Version field, enter a number that identifies the version of this application record, for example, `01.01.01`.

©2025 Pegasystems Inc

4. **Optional:** To create localization Rules in the language in which you build the application, in the **Base language** field, select the default language for your application.

   > **For example:**
   >
   > If you build your application in European French, select **French FR (France)**.

   > **NOTE:** This option is available only for applications that you build on Constellation.

5. In the **Organization settings** section, in the **Organization name** field, specify an organization that owns the application:
   - To reuse an existing organization, press the Down arrow key, and then select an organization.
   - To create an organization, enter a unique and descriptive name, for example, `HROrg`.

   The default value is an organization of the user that creates the application.
6. In the **Division name** field, specify a division that owns the application:
   - To reuse an existing division, press the Down arrow key, and then select a division.
   - To create a new division, enter a unique name.
7. In the **Unit** field, specify a unit in the division:
   - To reuse an existing unit, press the Down arrow key, and then select a unit.
   - To create a new unit, enter a descriptive name.
8. In the **Class layers** section, configure the class structure of your application:
   a. **Optional:** To create a new division layer, select the **Generate division layer** checkbox, and then, in the **Division** field, specify a name of the division from the new layer.
   b. **Optional:** To create a new class layer, select the **Generate class layer** checkbox, and then, in the **Class** field, specify the class for the new layer.

©2025 Pegasystems Inc

c. If you create a new organization, in the **Organization** field, enter the organization name.
   This option is read-only when you choose an organization that already exists.

9. In the **Application** field, enter a value that defines the application class layer for the application stack.
   If you leave this field blank, the application uses the value that you provide in the **Application ID** field.
10. In the **Class group name** field, enter a name that an application uses to create Work Queues for users.
    The default value is `Work`.
11. Click **Save**.

For example:

The following figure shows sample advanced configuration for an application:

©2025 Pegasystems Inc

Advanced application settings

- Creating Rules
- Organizing Rules into Rulesets
- Organizing Rules into classes

©2025 Pegasystems Inc

- Rule resolution

Switching between applications

When you work on multiple applications, you can switch between them by using the **My Applications** gadget. By default, the **My Applications** gadget is available in App Studio.

> **Before you begin:**
>
> To switch between applications in Portals other than App Studio, add the gadget.
>
> For more information, see Adding the My Applications gadget.

1. In the header of App Studio, click **My Applications**.
2. From the list, select an application that you want to open.
3. **Optional:** If you have more than one Access Role in the application, select the role that you want to use, and then click **Go**.

   You can switch between applications without reentering login credentials if the applications share a common authentication service. However, if you are using two Pega Platform applications without a common authentication service or where the user is not authenticated with a common authentication service, and you try to switch applications, a dialog will launch with the message: `Switch application with different authentication service`. When you click the Login button, a new tab will open for you to reauthenticate using the default authentication service for that application.

   For more information, see Mapping authentication services in App Studio.

- **Adding the My Applications gadget**
- **Switching between applications in a Constellation Portal**
- Changing your studio

©2025 Pegasystems Inc

Adding the My Applications gadget

To conveniently navigate between your applications, add the *My Applications* gadget to your existing Portal or application by embedding a header and a menu. By using the *My Applications* gadget, you can see an overview of your applications with short descriptions, customize the icons of your applications, and search the applications by name. By default, the *My Applications* gadget is available in App Studio.

> **Before you begin:**
>
> Ensure that your application uses UI Kit 14 or any of its later versions.

1. In Dev Studio, open an application in which you want to add the **My Applications** gadget.
   For more information, see Switching between applications in Dev Studio.
2. In the navigation pane, click **Records** > **User Interface** > **Section**, and then open the **pyPortalHeader** section.
3. Right-click the embedded **pyAppLauncher** section, and then click **Copy**.
4. Open the section where you want to add the **My Applications** gadget.
5. Right-click the section, and then click **Paste**.
6. Open the **pyPortalHeader** section.
7. Repeat steps 3 through 5 for the **My Applications** icon.
8. Click **Save**.

Switching between applications in a Constellation Portal

Navigate between different Constellation and traditional applications in your environment by using the application switcher in the Portal main header.

> **NOTE:** To retrieve the application URL, the user's privilege must allow opening all Rule applications defined in the Access Groups. This requires the `rule-application` class to be readable, even when outside the current Ruleset

©2025 Pegasystems Inc

list. The standard `WorkMgr4` Access Role grants the user read-only access to the Ruleset list, which prevents the application switch from working properly. To address this issue, extend the out-of-the-box Access Role with relevant permissions or use the `User4` role.

1. In the header of your application, in the upper-left corner of the screen, click Switch application.

Application switcher

2. In the list, select an application that you want to open.

NOTE:

- The list includes both the Constellation and traditional applications that are available to the user.
- The application switcher does not switch between Portals, because best practice for Constellation is to use one Portal for each application.
- The switcher uses the application URL alias.
- You can customize the list of applications that are displayed in the application switcher by editing the *pyMyApps* activity. For example,

©2025 Pegasystems Inc

you can remove applications from the list or add links to external applications.

Switching between applications in Dev Studio

When you work on multiple applications, you can switch between them by using the Application menu.

1. In the header of Dev Studio, click the name of the application, and then click Switch Application > [*Application name*].
2. In the alphabetized list, select an application that you want to use as your current application.

NOTE: The list of available applications is based on the Access Groups that are defined on your **Operator ID** form. If more than one Access Group points to the same application, the Switch Application menu displays the application label followed by the Access Group in parenthesis.

You can switch between applications without reentering login credentials if the applications share a common authentication service. However, if you are using two Pega Platform applications without a common authentication service or where the user is not authenticated with a common authentication service, and you try to switch applications, a dialog will launch with the message: `Switch application with different authentication service`. When you click the Login button, a new tab will open for you to reauthenticate using the default authentication service for that application.

Exploring the Application Definition

Analyze and edit basic information about your application by exploring the Application Definition. In the Application Definition, you can edit what built-on applications your

©2025 Pegasystems Inc

software uses to source elements, manage branched development, and add components to extend the functionality in your application.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. On the **Definition** tab, explore and edit the Application Definition to meet your business needs:
   - To reuse elements and assets from other applications, such as Rules or features, explore the **Built on applications** section.

     For more information, see Adding built-on applications.

   - To expand the functionality of your application by adding components, explore the **Enabled components** section.

     Components represent reusable features, such as a tool to capture multimedia within an application. You can create your own component or you can use ready-made components that are available on Pega Marketplace.

     For more information, see Installing components.

   - To define the look and feel of your application by applying a skin, explore the **Presentation** section.

     For more information, see Specifying a skin for your application.

   - To define a URL to launch your application, explore the **Application URL alias** section.

     For more information, see Adding an application URL alias.

   - To define advanced configurations for your application, explore the **Advanced** section.

     For more information, see Configuring advanced application settings.

PEGA ©2025 Pegasystems Inc

- To provide separate sandboxes for application developers to avoid conflicts and overrides, explore the **Development Branches** section.

  When you create Branches, developers can build and modify Rules without interfering with the work of another team that works on the same application.

  For more information, see Branches and Branch Rulesets.

- To enrich your application with a specific element, such as assets that are associated with a specific Case Type or Access Group, explore the **Application Rulesets** section.

  For more information, see Organizing Rules into Rulesets.

- **Adding built-on applications**
- **Configuring advanced application settings**
- **Managing application Cases and data**
- **Configuring applications for reuse**
- **Setting your application password**
- **Security tab of the Application Definition**
- **Specifying the content of application documentation**
- Designing applications for reuse and extension
- Basic requirements for deploying public-facing applications

Adding built-on applications

To save time and reduce development costs you can reuse elements between your applications, such as Rules and features, by adding built-on applications. When you

PEGA ©2025 Pegasystems Inc

build your application stack, you ensure that you meet business requirements in an efficient way.

For example, when you create an application to review mortgage requests, you can add a built-on application that workers use to review loan requests, and then reuse the elements that are relevant to the mortgage Cases.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. In the **Built on applications** section, click Add application.
3. In the Name field, enter the name of the application that you want to add.
4. In the Version field, enter the version of the application that you want to add. As a best practice, add the latest version of your application.
5. To add more built-on applications, repeat steps 2 through 4.
6. Click Save.

> **Result:**
>
> When you work on your current application, you can reuse resources from your built-on applications.

- **Best practices for using multiple built-on applications**
- **Application stack hierarchy for multiple built-on applications**

Related concepts

- Designing applications for reuse and extension
- Organizing Rules into Rulesets
- Organizing Rules into classes
- Rule resolution

Related tasks

©2025 Pegasystems Inc

- Adopting feature-driven development

Best practices for using multiple built-on applications

To increase reuse between your Pega Platform™ applications and speed up the development process, enhance your applications by using multiple built-on applications. Built-on applications are structured as a hierarchical application tree.

By using multiple built-on applications, you break up common and stand-alone components into their own applications, and provide the ability to point an application to another application Rule, helping to automatically update any required Rulesets and versions across applications when applications undergo changes.

For example, the uPlus organization creates an Application Layer to contain the definition of common assets and integrations in corporate systems. Separating each integration into its own built-on application can simplify the revisioning of each integration by removing dependencies. As a result, applications use features independently and when developers want to update a feature, they can update only a specific built-on application, without affecting the others.

Best practices for using multiple built-on applications

Consider the following tips when developing an application that uses multiple built-on applications:

- Limit development to Rulesets in the top-most application and keep built-on applications locked.
- To develop in the lower layers of the hierarchical application tree, switch to a built-on Application Layer.
- Avoid having the same Ruleset in multiple applications. Instead, refactor the Ruleset to its own application or a common application.
- Avoid using different versions of an application in the hierarchical application tree.
- Avoid Branch IDs that span several applications.
- Avoid making any application structure changes without utilizing the Validation Tool to validate Rules.

PEGA ©2025 Pegasystems Inc

- Avoid using Ruleset Validation mode Rulesets. Ruleset prerequisites can be challenging to maintain, especially when you need to refactor the application stack. Always default to using Application Validation mode Rulesets, which create the Ruleset stack depending on the application in which you define the Rulesets.

Built-on applications and the Pega Process Fabric Hub

The Pega Process Fabric Hub is an application that gathers Assignments from multiple distributed remote applications and provides a unified user interface to manage work that comes from multiple other sources. The Pega Process Fabric Hub and built-on applications are different solutions that fit different scenarios. When you need to compare scenarios that support multiple built-on applications or the Pega Process Fabric Hub, consider the following factors:

Built-on applications	The Pega Process Fabric Hub
You can deploy all built-on applications as a single stack.	Different stakeholders can manage different remote applications that publish work to the Pega Process Fabric Hub, and you can connect applications that you host on different systems.
The Lifecycle of built-on applications is synchronized; for example, you deploy all applications simultaneously to a new environment.	Each remote application has a separate Lifecycle that you manage without interfering with other applications.
Maintenance and update of all built-on applications are simultaneous and smooth without contradicting the behavior of different applications.	Changes that you deploy in one remote application do not affect other applications that publish work to the Pega Process Fabric Hub.
Built-on applications provide the capacity for customizations of selected applications without negative interference to other applications.	The Pega Process Fabric Hub offers out-of-the-box functionalities without a need for any customizations, which helps you begin processing work faster.

PEGA ©2025 Pegasystems Inc

Built-on applications	The Pega Process Fabric Hub
All applications in the built-on applications stack are Pega Platform applications.	You can connect both Pega and non-Pega applications to the Pega Process Fabric Hub.
By implementing built-on applications, you can reuse different features between applications.	The Pega Process Fabric Hub focuses on processing Assignments that different applications publish. Functionalities such as Service-Level Agreements or notifications are specific to each remote application and are independent of the Pega Process Fabric Hub.

For more information on managing work from multiple applications in a unified interface, see Pega Process Fabric Hub .

Creating a unified view of work items in built-on applications and in the Pega Process Fabric Hub

You can use both built-on applications and the Pega Process Fabric Hub to create a unified view that summarizes Assignments from multiple applications. The solution that you select depends on your scenario. Consider the following factors:

- Use built-on applications when you host your applications on one server without any need for external connections, and all your applications are based on the same version of Pega Platform.
- Use the Pega Process Fabric Hub when you gather work from applications that you host on different servers or when you use applications that are based on different versions of Pega Platform. You can also create connections between the Pega Process Fabric Hub and non-Pega applications.

Limits for applications in a built-on applications stack

The number of applications in a built-on applications stack depends on your scenario. If updating or modifying any of the applications in the stack might result in severe issues and extensive effort to maintain and adjust other applications, consider limiting your

PEGA ©2025 Pegasystems Inc

application stack. When you add an application as a built-on application, ensure that the benefits of reuse outweigh any risks connected with maintainability.

Related concepts

- Application stack hierarchy for multiple built-on applications

Related tasks

- Adding built-on applications
- Configuring Ruleset version settings

Related information

- Validation tool

Application stack hierarchy for multiple built-on applications

When you use multiple built-on applications, Pega Platform™ automatically converts the hierarchical application tree of multiple built-on applications into a single linear application stack. As a result, you can shorten development time and increase consistency in your application by reusing Rulesets and frameworks.
Pega Platform creates the linear application stack using the following Rules:

- The system always locates PegaRULES at the end of the linear application stack list.
- Pega Platform processes each built-on application by using a depth-first algorithm that searches as far as possible along an application tree branch before moving on to another branch.

  An application tree branch consists of a built-on application and its built-on applications.

©2025 Pegasystems Inc

- For duplicate applications with the same application version, Pega Platform retains only the application that has the lowest position in the tree in the application stack list, and ignores the other application.
- For duplicate applications with different application versions, as well as for PegaRULES, Pega Platform uses the numerically highest available application version.

The following image represents the automatic conversion of a hierarchical application tree into a single linear application stack at run time:

Automatic conversion into a single linear application stack

Ruleset behavior at design time

Each application in the hierarchical application tree consists of various Rulesets that Pega Platform automatically assembles into a linear Ruleset stack. This Ruleset stack defines which Rulesets a specific Rule can use for context purposes at design time.

Ruleset stack assembly generally follows the same guidelines as for creating the linear application stack at run time. However, Ruleset stack assembly for a specific Rule is based on the specific application that the Rule belongs to, which is also called the Rule's owning application.

©2025 Pegasystems Inc

After Pega Platform determines a Rule's owning application, Pega Platform automatically creates the linear Ruleset stack for that specific Rule, starting with the Rule's owning application and working through the hierarchical application tree branch to PegaRULES.

> **NOTE:** The behavior described above applies only when you save a Rule in an Application Validation mode Ruleset. When saving a Rule in a Ruleset Validation mode Ruleset, you must specify prerequisites in the REQUIRED RULESETS AND VERSIONS section on the Versions tab of the Ruleset.

When you use Application Validation mode, at design time, the saved Rule cannot reference certain Rulesets that are available at run time. This restriction allows for the appropriate isolation of applications, and makes applications easier to reuse.

Example of the transition from hierarchical to linear Ruleset stack assembly

In the following example, the hierarchical application tree contains only Rulesets that use the Application Validation mode:

MortPlus 01.01 (*Ruleset1*)	
Underwriting 01.01 (*Ruleset2*)	Pricing 01.01 (*Ruleset4*)
uPlusFin 01.02 (*Ruleset3*)	uPlusFin 01.01 (*Ruleset5*)
PegaRULES	PegaRULES

If you save *RuleABC* in *Ruleset2*, the Underwriting application becomes the Rule's owning application. Pega Platform then assembles the Ruleset stack that *RuleABC* can use at design time according to the following pattern:

Ruleset2 > *Ruleset3* > *PegaRULES*

Pega Platform starts building the Ruleset stack at the Underwriting application, and then processes the stack through PegaRULES in that specific application tree branch.

©2025 Pegasystems Inc

Because the Pricing application is located on a separate application tree branch, at design time, Ruleset stack processing cannot reference *Ruleset4* and *Ruleset5* from the other tree branch.

Ruleset behavior at run time

At run time, Pega Platform creates the linear application stack for the entire tree of built-on applications by following the preceding guidelines. This application stack is the basis for application functionality in a live environment.

Use case examples

The following use cases describe how Pega Platform processes the hierarchical application tree into a single linear application stack at run time.

NOTE: These examples are for demonstration purposes only and might not represent the application structure of current application offerings.

For example:

In this example, the TopApp application is built on the BaseApp1, BaseApp2, and BaseApp3 applications. All applications are built on the same version of PegaRULES.

TopApp [*Ruleset1*, *Ruleset2*]		
BaseApp1	BaseApp2	BaseApp3
PegaRULES		

The resulting application stack has the following structure:

TopApp > BaseApp1 > BaseApp2 > BaseApp3 > PegaRULES

©2025 Pegasystems Inc

The system processes each built-on application by using a depth-first algorithm.

For example:

In this example, the MyCustomerApp application is built on the BillingApp and SupportApp applications. Each of these applications is built on a different version of PegaRULES.

MyCustomerApp (*Ruleset1*)	
BillingApp (*Ruleset2*)	SupportApp (*Ruleset4*)
BillingPlansApp (*Ruleset3*)	PegaRULES 07.05
PegaRULES 07.01	

The resulting application stack has the following structure:

MyCustomerApp > BillingApp > > BillingPlansApp > SupportApp > PegaRULES:07.05

Pega Platform uses version of PegaRULES with the highest available number in the application stack list.

For example:

In this example, the ProductionApp application is built on the FinalAppTemplate application, which is built on the CustomerApp and EmployeeApp applications. Each of these applications has additional built-on applications, including the duplicated MyBusinessApp application, which has two different versions in use. All applications are built on the same version of PegaRULES.

ProductionApp 01.01 (*Ruleset1*)
FinalAppTemplate 01.01 (*Ruleset2*)

©2025 Pegasystems Inc

CustomerApp 07.01 (*Ruleset3*)	EmployeeApp 01.01 (*Ruleset6*)
ServicesApp 01.01 (*Ruleset4*)	MyBusinessApp 01.01 (*Ruleset7*)
MyBusinessApp 01.02 (*Ruleset5*)	PegaRULES 07.01
PegaRULES 07.01	

The resulting application stack has the following structure:

ProductionApp > **FinalAppTemplate** > **CustomerApp** > **ServicesApp** > **EmployeeApp** > **MyBusinessApp:01-02** > **PegaRULES**

Even though the system retains the application with the lowest position in the tree list, the application stack list uses the numerically highest available version for the duplicated application.

For example:

In this example, the MyCustomApp application is built on several other applications, including two applications that both contain the same MyHealthApp application. All applications are built on the same version of PegaRULES.

MyCustomApp			
ComplexApp	BasicApp1	BasicApp2	
MyPlanner	ResultsApp	SurveyApp	PegaRULES
MyReports	MyHealthApp	PegaRULES	
MyHealthApp	PegaRULES		
PegaRULES			

©2025 Pegasystems Inc

MyCustomApp has four built-on applications: ComplexApp, BasicApp1, BasicApp2, and PegaRULES. The resulting application stack has the following structure:

MyCustomApp > ComplexApp > MyPlanner > MyReports > BasicApp1 > ResultsApp > MyHealthApp > BasicApp2 > SurveyApp > PegaRULES

Duplicate applications with the same version use the lowest-positioned application in the tree list.

For example:

In this example, the MyCallCenter application has two branches: Branch_MyCallCenterRS and Branch_Policies. All applications are built on the same version of PegaRULES.

MyCallCenter • Branch_MyCallCenterRS • Branch_Processes	
MyStaffingApp	Processes
Policies	PegaRULES
PegaRULES	

The resulting application stack has the following structure:

MyCallCenter > MyStaffingApp > Policies > Processes > PegaRULES

Related concepts

- Best practices for using multiple built-on applications

©2025 Pegasystems Inc

Related tasks

- Adding built-on applications

Configuring advanced application settings

Incorporate advanced options into your application to ensure that the software that you develop meets your unique business requirements. For example, you can define how your application renders the UI, or enable your system to automatically manage Background Processes.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. On the **Definition** tab, expand the **Advanced** section.
3. Modify the advanced application settings to meet your needs:
   - To enable users to modify Rules in a production environment, click **Add production Ruleset**, and then in the **Production Rulesets (customization)** field, enter the Ruleset that includes the Rules that you want to make available for users to edit.

     Ensure that you unlock at least one Ruleset version in the production Ruleset.

     For more information, see Production Rulesets.

   - To configure your system to automatically manage the resolution of job scheduler and queue processor Rules in an application without any manual configurations, select the **Include in background processing** checkbox.

     For more information, see Managing system resources.

   - To define how your application renders the UI at run time, in the **UI Runtime** section, define whether you want your application to use standard server-rendered UI or Constellation.

©2025 Pegasystems Inc

For more information, see Constellation UI.

- To maintain properties on both the *pxRequestor* and *pxThread* pages, clear the **Place properties on thread page only (5-4 or later)** checkbox.

  By default, your application maintains the properties on the *pxThread* page only. Change this setting only for backward compatibility with applications built on Pega Platform version 5.3 or earlier. In any other scenario, keep the default setting.

- To define a screen that opens after a user logs out of the application, in the **Log off redirection** list, select the screen that you want to use.
- To indicate that the application is a representation of another, existing application, in the **Base application** field, enter the name of the existing application. As a result, you indicate that both applications have the same set of Rules and other elements.

  For example, you can create a development application to perform work that is completely separate and has no effect on your existing application, to ensure that it remains unaffected by any unexpected changes. Then, in the development application, in the **Base application** field, you can enter the name of your existing application.

  > **NOTE:** The **Base application** field has no impact at processing and only holds metadata.

4. Click **Save**.

- Basic requirements for deploying public-facing applications
- Clipboard tool
- Managing job schedulers
- Managing queue processors

©2025 Pegasystems Inc

- Organizing Rules into Rulesets
- Traditional UI

Managing application Cases and data

Explore the Application Definition to gain a holistic view of data in your application and Cases that are available for creation at run time. As a result, you can globally manage resources in your application from one place.For example, you can select which Cases users can create at run time and disable the Cases that are not required for your business scenarios. Consequently, at run time, users have a more relevant choice of Cases to create. To speed up application development, you can also provide a more relevant Data Explorer by adding necessary data types and removing data types that are not required.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. Click the Cases & data tab, and then manage any of the relevant settings for your application:

Choices	Actions
Enable creation of selected Cases at run time	In the **Case Types** section, select the checkbox in the row of the Case Type that you want to enable for run-time creation.
Add a new Case from the Ruleset	a. In the **Case Types** section, click Add. b. In the Name field, select the Case that you want to add to your application. c. In the Work ID prefix of the new Case Type, enter a Case ID value. d. In the Implementation class field, enter the Class in which the Case is stored.

©2025 Pegasystems Inc

Choices	Actions
Change the prefix of run-time Case instances of a Case Type	In the **Case Types** section, in the Work ID prefix field of the selected Case Type, enter a new value. By default, the prefix is the first letter of the name of a Case Type. If you have multiple Case Types that start with the same letter, you can change prefixes so that users can conveniently distinguish Cases at run time. The following figure shows an example of how changing the prefix can help you differentiate between Cases at run time: *Changing Case prefix*
Change the Scope of resources that are available for a Case Type	In the **Case Types** section, in the Implementation class field, enter a new class to store the Case Type. For more information, see Understanding class layers.

©2025 Pegasystems Inc

Choices	Actions
Manage data types in your Data Explorer	a. In the **Data** section, in the text field, enter the class of the data type that you want to add to your Data Explorer, and then click **Add**. b. **Optional:** To remove a data type from the Data Explorer, click **Delete** in a row of the data type.
Change how your application stores information by associating new classes with main application elements	a. In the **Associated classes** section, in the **UI class** field, provide the default class for UI Rules. b. In the **Associated classes** section, in the **Integration class** field, provide the default class for integration-related Rules. c. In the **Associated classes** section, in the **Data class** field, provide the default class for data-related Rules.

3. Click **Save**.

- Creating and managing Cases
- Data management and integration
- Data objects overview
- Updating your Data Model
- Configuring Case Type settings
- Changing the Case ID format

Configuring applications for reuse

To save time and immediately start application development, make your application available as a template. When you save your application as a template, other users can

©2025 Pegasystems Inc

quickly build a new application from that sample. The newly created application automatically inherits template logic, data, and Personas.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. On the Application wizard tab, select the Show this application in the New Application wizard checkbox.
3. In the **New Application wizard settings** section, in the Short description field, provide the name and purpose of your application:
   - To reuse existing information, press the Down arrow key, and then select a field value that stores the information that you need.
   - To use a new name, enter a unique and descriptive label, for example, `Hiring process`.
4. In the Application description field, provide a description so that users can decide whether to reuse your application:
   - To reuse existing information, press the Down arrow key, and then select a field value that stores the information that you need.
   - To use a new description, enter a unique and descriptive text, for example, `Use this application to process job applications`.
5. In the New application access list, select the Access Group that users receive when they reuse your application.
   You can only choose from development Access Groups, which are Access Groups that contain the developer or *pxExpress* Portals.
6. **Optional:** To distinguish your application when users select reusable templates, replace the default logo and preview images:
   a. In the **Brand** section, click Upload Image.
   b. In the explorer dialog box, navigate to the file for your company or organization logo, and then click Upload Image.
   c. In the **Desktop snapshot** section, click Upload Image.
   d. Navigate to a screen capture that is at least 370 pixels by 233 pixels, and then click Upload Image.
   e. In the **Mobile snapshot** section, click Upload Image.

©2025 Pegasystems Inc

   f. Navigate to a screen capture that is at least 110 pixels by 194 pixels, and then click **Upload Image**.
7. Click **Save**.

> **Result:**
>
> When users run the new application wizard to create an application, they can choose your application as a template and reuse its logic, data, and Personas.

- Creating applications
- Adding built-on applications

Setting your application password

Improve your application security by setting an application password, which limits the number of users who can make changes to your application. For extra security, you can periodically update this password based on your security framework and security plan.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. Click the **Security** tab, and then in the **Application security** section, select the **Require password to update application** checkbox.
3. Click **Update password**.
4. In the **Update password** dialog box, enter the new password in the **Enter password** and **Confirm password** fields.

   > **NOTE:** To successfully update your application password, you must enter identical values in both fields.

5. Click **Submit**.
6. Click **Save**.

©2025 Pegasystems Inc

> **Result:**
>
> Users need to provide the password before they can save their changes to the application.

- Basic requirements for deploying public-facing applications

Security tab of the Application Definition

Use this tab to define security settings in your application, map authentication services, and enable content, mashups, and digital messaging security.

Application Definition

To view the Application Definition, in the header of Dev Studio, click your application name, and then click **Definition**.

Security Tab

The Application Definition contains seven tabs, including the Security tab. To view the **Security** tab, in the header of Dev Studio, click your application name, and then click **Definition** > **Security**.

The **Security** tab contains the following sections: **Application security**, **Authentication**, **Content security**, **Mashup security** and **Digital Messaging security**.

Application security

Application security has one setting: the **Require password to update application** check box, which should be selected when creating an application. The **Require password to update application** check box should remain selected.

Select the **Require password to update application** check box if you want to change or update the password that users must enter when updating the application, and then click **Update password** to set the password.

©2025 Pegasystems Inc

For more information, see Setting your application password.

The following figure shows the **Application security** section:

The application security section of the Security tab of the Application definition

Authentication

Authentication services are new to Pega Platform. Users with the *pzAdvancedSecurityUser* privilege can map authentication services to an application. By default, the *PegaRULES:SecurityAdministrator* includes the *pzAdvancedSecurityUser* privilege.

By mapping an authentication service, a security administrator can define the authentication service mechanism that users use to log in to an application. Administrators can also create a default login mechanism and interactive screen the user selects an authentication service to login.

The Authentication section changes depending on whether you have already mapped authentication services.

The following figure shows the Authentication section for a system that does not have any mapped authentication services:

©2025 Pegasystems Inc

The authentication services section of the Security tab of the Application definition

The following figure shows the Authentication section for a system that already has mapped authentication services:

A populated authentication services section of the Security tab of the Application definition

For more information, see Mapping authentication services in Dev Studio.

©2025 Pegasystems Inc

You can also map authentication services in App Studio, without using the Application Definition. For more information, see Mapping authentication services in App Studio.

Content security

The content security policy (CSP) is a set of directives that inform the user's browser of locations from which an application is allowed to load resources, such as fonts, images, and style sheets.

Use the appropriate UI Portal field to select the content security policy, or the **Policy name** field for traditional UI applications only. You can configure the content security policy directly from this page by clicking the **Configure** icon after you select the policy.

Use the **Mode** setting to specify what the browser does when a policy is violated:

Mode	System behavior
Reject and Report	Enforce the policy and report the violation. **NOTE:** The content source must be in the correct list.
Report Only	Report the violation, but do not enforce the policy.

The following figure shows the **Content security** section for a traditional UI application:

©2025 Pegasystems Inc

The content security section of the Security tab of the Application definition

The following figure shows the **Content security policy** section for a Constellation UI application:

The content security policy section of the Security tab of the Application definition

For more information, see:

- Content security policies
- Creating a content security policy

Mashup security

Mashup security is used to define the external URLs that are allowed to access Pega Platform so that the host page can communicate with the mashup gadget, if you use the mashup feature to embed Pega Platform content in an external application. This feature functions, in part, as an allow list.

©2025 Pegasystems Inc

The following figure shows the **Mashup security** section:

The mashup security section of the Security tab of the Application definition

For more information, see Securing your application for mashup communication.

Digital Messaging security

This security feature is used to set up messaging platforms for Pega Intelligent Virtual Assistant (IVA) for Digital Messaging. To start using Digital Messaging, you must configure and define the security settings for the channel and the system. Examples of messaging platforms that are useable by the IVA include Apple Business Chat, Facebook Messenger, MMS/SMS (Twilio), and WhatsApp Messenger.

The following figure shows the **Digital Messaging security** section:

The Digital messaging security section of the Security tab of the Application definition

For more information, see Configuring Digital Messaging channel security.

©2025 Pegasystems Inc

Specifying the content of application documentation

Provide guidance and methodology information to developers and other stakeholders by defining which information the application documentation includes. As a result, you can provide relevant details about your software and the methodology that your team implements.For example, you can define the business goals that you want your application to meet and provide additional relevant materials by adding attachments to your application overview.

> **NOTE:** Defining the content of documentation in Dev Studio is suited to advanced developers or developers that use legacy document types. To implement low-code solutions, prepare application documentation in App Studio. For more information, see Creating project documents for stakeholders.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. Click the Documentation tab.
3. In the **Application guides** section, add guides that you want developers to follow during creating applications.
   For more information, see Creating a guide for application developers and administrators and Adding an application guide to a Portal.

   > **For example:**
   >
   > You can add a security checklist to ensure that developers provide secure applications. At design time, developers can view and tick tasks that they complete while they develop an application, as in the following figure:

©2025 Pegasystems Inc

©2025 Pegasystems Inc

Security checklist

4. In the **Supporting specification types** section, describe the functionalities that you want to implement:
   a. In the **Name** field, enter a label for the specification.
   b. In the **Short description** field, provide brief information about the functionality.
   c. **Optional:** To add more specifications, click **Add specification**, and then repeat steps 4.a through 4.b.

> **NOTE:** To take advantage of the latest Pega Platform tools, use features instead of specifications. For more information, see Adopting feature-driven development and Feature-driven development tracking.

5. In the **Business objectives** section, in the **Description** field, briefly describe a goal that you want to achieve in your application.

> **For example:**
>
> Enter `Review loan requests from VIP customers.`

> **TIP:** You can also provide business objectives in a low-code way in App Studio. For more information, see Editing application details in App Studio.

6. In the **Releases** section, click **Add item**, and then in the field, provide a new release version for your application.
7. **Optional:** To change the value of the release version that Pega Platform provides by default, click the field for the release, and then provide a value.

©2025 Pegasystems Inc

For example:

The following figure shows custom release versions for an application:

Application releases

8. **Optional:** To provide a more detailed timeline of your application development, in the **Iterations** section, click **Add item**, and then provide an iteration for your application.
9. In the **Attachments** section, provide reference materials about your application development and planned design, by clicking **Add/Edit attachments**, and then attaching content:
   - To attach a file, in the **Attach files** section, drag a file or click **Select file(s)** and navigate to the file that you want to add, and then click **Submit**.
   - To attach a URL, in the **Enter a link** section, provide a URL and a short description of the URL content, and then click **Submit**.

©2025 Pegasystems Inc

Result:

The attached content is available in the **Overview** section of your application.

10. In the **Organization** section, specify the organizational owner for your application:
    a. Click Update.
    b. In the **Update Organizational Unit** dialog box, select the organization, division, and unit that own your application.
    c. Click Submit.

For example:

The following figure shows an application that belongs to an HR department in an Administration division for a UPlusTelco company:

Application organizational ownership

The organizational structure impacts the routing and reporting in your application of, for example, notifications to managers.

11. In the **DCO settings** section, in the Project Methodology list, select a methodology that you want to implement during application development, for example, select Scrum.
12. Click Save.

©2025 Pegasystems Inc

- Estimating application development
- Managing application inventory

Basic requirements for deploying public-facing applications

To ensure the security and branding of your public-facing applications, review and follow the minimum requirements for deployment through Pega Platform™.

> **NOTE:** Do not deploy any public-facing applications that you develop in Pega Platform without following these basic requirements. If you cannot complete an item on this list, immediately notify the application owner of the security risks and of the omitted tasks.

The Security Checklist: Best practices for securely deploying your application

Security is critical in all applications, especially in applications that are used by external operators, such as customers and prospective customers, who are not your employees. Inadequate security might cause the deployment of your application to fail.

To secure your application, complete the Security Checklist to:

- Follow best practices for securely deploying applications.
- Ensure the confidentiality, integrity, and availability of your application in production.
- Identify when each task should be performed: at or near the beginning of development, on an ongoing basis, or just before deployment.
- Avoid expensive rework late in your development process.

Unless otherwise noted, these recommendations apply to all deployment environments, including Pega Cloud services.

PEGA ©2025 Pegasystems Inc

The most important security requirement for all applications is to maintain guardrail-compliance because Pega Platform security features cannot always be successfully enforced in custom code. You can secure applications by configuring only the built-in features in Pega Platform, without relying on custom code that is built by developers who are not security experts.

For more information, see Security Checklist.

Pega Platform shows the overall completion of the Security Checklist on the Dev Studio home page, and provides tools to track the status of each task.

For more information, see Assessing your application using the Security Checklist.

Public users authentication

Unauthorized individuals should not have access to view or modify the application or its data. Always verify the identity of application users, and apply one of the following types of Authentication Services, based on the user type:

Anonymous user

A user is initially anonymous (identity unknown). The users may register themselves or have their identity verified partway through a session.

For example, in online shopping, unauthenticated users can browse and add items to a shopping cart, and then either create an account or enter their credentials to check out.

To configure authentication for applications that require anonymous users, you define the following parameters:

- An *Authentication Service* Rule of the **Anonymous** Type
- An Access Group for all anonymous users
- A limit for the Access Group to provide access to only data and application functions that are not in any way confidential (such as the product catalog and shopping cart)

PEGA ©2025 Pegasystems Inc

After a user becomes a known user, either by registering or entering credentials, you can use the Re-Authentication gadget to dynamically change their roles and privileges, so that the user can continue their session without logging in again.

For more information, see Configuring an application to use an anonymous authentication service.

Authenticated user

User credentials must be verified during login through an external data store or database, such as a customer master file.

User credentials are verified at the start of a session, but those credentials are stored outside of Pega Platform.

To configure authentication for applications with authenticated users, you define the following parameters:

- An *Authentication Service* Rule of the **Basic Credentials** type
- The **Verify credentials using an external identity store** option
- A Data Page for accessing an external data store

Single sign-on

Single sign-on (SSO) is a property of identity and authentication that enables users to securely authenticate with a set of login credentials, such as a name and password. SSO is used and the user is authenticated through their credentials on a social media site, such as Google.

For more information, see:

- Creating an authentication service
- Creating a Google authentication service
- Authentication

©2025 Pegasystems Inc

Restrict access to data in *Data-Admin-Operator-ID* class

In public-facing applications where end users do not need access to information about other operators, we recommend that you restrict all access to data in the *Data-Admin-Operator-ID* class to only the end user's data through an Access Control Policy. You can do this by enabling the out-of-the-box Rules *pyDefault* and *pyRestrictToSelf* in the *Data-Admin-Operator-ID* class.

If this Rule does not exist in your Pega Platform release, you can create it in any release from 7.3 forward. For more information, see Restricting access to operator information in public-facing applications in Pega Platform 8.4 and earlier.

For more information, see:

- Attribute-based access control
- Creating an Access Control Policy condition

Sensitive client data protection

Never deploy any application that creates or stores personally identifiable information (PII), such as social security numbers, account numbers, health data, date of birth, addresses, phone numbers, or any other sensitive data that could be used to identify an individual, without protecting that data from unauthorized access.

To encrypt sensitive property values, apply these guidelines:

- Set up a master key in an external Key Management System.
- Define a Keystore Rule instance that references the master key.
- In Dev Studio, on the **Data Encryption** landing page, reference the Keystore Rule that accesses the master key.
- Define PropertyEncrypt Access Control Policies, and then list the properties to encrypt.
- Define PropertyRead Access Control Policies to conditionally mask or obfuscate these property values for operators who should not be able to view them.

PEGA ©2025 Pegasystems Inc

For more information, see:

- Encrypting system data by using a custom key management service
- Creating a keystore for application data encryption
- Creating an access control policy

Branding and exposed information during user logins

Set up a descriptive and memorable, or vanity, URL or custom domain name that does not unnecessarily expose server information to users.

For more information about configuring the domain name on Pega Cloud, see Requesting a custom domain name for applications hosted in Pega Cloud .

- **Restricting access to operator information in Pega 8.5**
- **Restricting access to operator information in public-facing applications in Pega Platform 8.4 and earlier**

Restricting access to operator information in Pega 8.5

Restrict all access to data in the *Data-Admin-OperatorID* class to only the end user's data by using an Access Control Policy. You can restrict access to personally identifiable information (PII) for security purposes, such as protection against unauthorized exposure of PII data. Restricting access to only the end users' data increases the security and peace of mind of users who must communicate with clients and customers through public-facing Channels.

To enable the *pyDefault* and *pyRestrictToSelf* Rules in the *Data-Admin-OperatorID* class, follow the steps below:

1. In the header of Dev Studio, click **Create** > **Security** > **Access Control Policy Condition**.
2. In the header of Dev Studio, click Records
3. Open the *pyRestrictToSelf* Access Control Policy condition Rule.

©2025 Pegasystems Inc

   a. In the **Policy Condition** field, select **pyRestrictToSelf**.
   b. In the **Applies To** field, select **Data-Admin-OperatorID**.
4. Click **Save As**, then select the application Ruleset for which you want to enforce this restriction, and click **Create and open**.
5. On the Rule form, click **Availability**, and then select **Available** from the list.
6. In the header of Dev Studio, click **Records** > **Security** > **Access Policy Control**.
7. Open the *pyDefault* Access Control Policy Rule.
   a. In the **Policy name** field, select **pyDefault**.
   b. In the **Action** field, select **Read**.
   c. In the **Applies to** field, enter `Data-Admin-OperatorID`.
8. Click **Save As**, then select the application Ruleset for which you want to enforce this restriction, and click **Create and open**.
9. On the Rule form, click **Availability**, and then select **Available** from the list.

Related concepts

- Configuring dynamic system settings
- Access Control Policy rule

Related tasks

- Restricting access to operator information in public-facing applications in Pega Platform 8.4 and earlier

Restricting access to operator information in public-facing applications in Pega Platform 8.4 and earlier

Restrict all access to data in the *Data-Admin-OperatorID* class to only the end user's data by using an Access Control Policy. Enable this access to personally identifiable information (PII) for security purposes, such as protection against unauthorized exposure of PII data. Restricting access to only end users' data increases the security

©2025 Pegasystems Inc

and peace of mind of users who must communicate with clients and customers through public-facing Channels.

Before you begin:

enable this to restrict access to PII data for security purposes. It provides for hardening their application against unauthorized exposure of PII data.

If you are using a version of Pega Platform earlier than 8.2, attribute-based access control (ABAC) is disabled by default. To enable this feature, you need to create a dynamic system setting with the following attributes:

1. In the header of Dev Studio, click **Create** > **SysAdmin** > **Dynamic System Settings**.
2. In the **Short Description** field, enter `Enable Attribute BasedSecurity`.
3. In the **Owning Ruleset** field, enter `Pega-RulesEngine`.
4. In the **Setting purpose** field, enter `EnableAttributeBasedSecurity`.
5. Click **Create and Open**.
6. On the **Settings** tab, in the **Value** field, enter `True`.

If your installation of Pega Platform does not contain the *pyRestricttoSelf* Rule, from Pega Platform 7.3 and later you can create it yourself.

1. In the header of Dev Studio, click **Create** > **Security** > **Access Control Policy Condition**.
2. Create an Access Control Policy Condition Rule by clicking **Create** > **Security** > **Access** > **Control Policy Condition**, and enter the following information:
   a. In the **Identifier** field, enter an identifying name.
   b. From the **Ruleset** list, select the application Ruleset in which you want to enforce this restriction.

©2025 Pegasystems Inc

   c. In the **Apply To:** field, enter `Data-Admin-Operator-ID`.
3. On the **Pages & Classes** tab, enter the following information:
   a. In the **Page Name** field, enter `OperatorID`.
   b. In the **Class** field, enter `Data-Admin-Operator-ID`.
4. Click *Definition* and then enter the following Conditions:
   a. In the **Conditional logic** section, enter a name for the Condition.
   b. In the **Policy Conditions** section, in the **Condition** field, enter the same name that you provided in the **Conditional logic** field.
   c. In the **Column source** column, select **.pyUserIdentifer**.
   d. In the *Relationship* column, select **Is equal**.
   e. In the *Value* column, select **OperatorID.pyUserIdentifer**.
5. Click **Save**.
6. In the header of Dev Studio, click **Records** > **Security** > **Access Control Policy**.
7. Create an Access Control Policy Rule with the following details:
   a. In the **Identifier** field, enter a name for the Rule.
   b. In the **Action** field, select **Read**.
   c. In the **Ruleset** field, enter any Rulesets in the application for which you want to enforce this restriction.
   d. In the **Applies To** field, enter `Data-Admin-Operator-ID`.
8. Click *Definition*, and then enter the name of the Access Control Policy Condition Rule that you create in Step 4 to the **Permit access if** field.
9. Save the Rule form.

Related concepts

- Configuring dynamic system settings
- Access Control Policy rule

Related tasks

- Restricting access to operator information in Pega 8.5

©2025 Pegasystems Inc

Best practices for compliance and updatability

Follow best practices for App Studio compliance and updatability of Pega Platform™ applications that you build in App Studio to improve the performance.

The guidelines help you ensure that the assets and Rules that you create in Dev Studio are available in App Studio. You also ensure that applications that you build in Dev Studio continue to function as expected with subsequent Pega Platform updates.

Comply with best practices for each area of application development:

General best practices

- Apply the Situational Layer Cake™ approach to maximize reuse of assets across your enterprise.

  For more information, see Modules and the Situational Layer Cake on Pega Academy.

- Limit the use of pattern inheritance. Instead, apply direct inheritance for greater ease of maintenance and reuse of assets in App Studio.

  For more information, see Understanding class hierarchy and inheritance.

- Avoid using activities. Instead, use Declare Expression Rules, Data Transforms, or other Rule types that are easier to define and maintain.

  For more information, see Declare Expression Rules.

- Assign one Portal to a Persona. Ensure that the Portal includes all landing pages, widgets, and so on that meet the Persona's business requirements.

  For more information, see Providing persona-based access to a Portal.

PEGA ©2025 Pegasystems Inc

Case Management

- Add every Case Type to the list on the **Cases & data** tab of the application record, with the appropriate configuration to support its creation.

  For more information, see Managing application Cases and data.

- Ensure that every Case Type has Stages, even if the Stages are only `Create`, `Open`, and `Resolved` that are analogous to the create, read, update, and delete (CRUD) operations.

  For more information, see Stages in a Case Lifecycle.

- Ensure that every Case Type has a `Create` Stage and at least one `Resolution` Stage.

  For more information, see Stages in a Case Lifecycle.

- Avoid using temporary Cases. Instead, use the `Create` Stage.

  For more information, see The Create Stage.

- Define entry conditions for Stages and gather related details for each Stage before the Case enters the Stage.

  For more information, see Defining the entrance criteria for a Stage.

- Use the option to update the Case Status during the Case Lifecycle.

  Case Statuses help communicate Case progress to the users. For more information, see Changing Case Statuses.

- Use the standard prefixes `New-`, `Open-`, `Pending-`, and `Resolved-` for all statuses.

  For more information, see Case Status values.

PEGA ©2025 Pegasystems Inc

- Clearly define the complete Microjourney® Lifecycle to the correct level of granularity.

  If the Lifecycle becomes large or complex, consider wrapping logical items into a Subprocess or Subcase and instantiating from the top-level Lifecycle. For more information, see Creating a Microjourney for customer success.

- In applications that you build with Theme Cosmos, consider relying on Subprocesses while processing a list of Embedded Data objects.

  This approach facilitates running common logic against each list item. For more information, see Embedding data in a Case.

- List all participants that are directly involved in the Case as parties.

  For more information, see Defining Case participants.

- Include Personas, Channels, as well as system and data methodology where required.

  For more information, see Understanding Personas and access management.

- Avoid hardcoding Actions directly into the UI; instead, use standard Case processing or Optional Case-wide and Stage-wide Actions in the Case Lifecycle.

  For more information, see Configuring and working with optional Actions in Cases.

- Avoid using tickets; instead, use features such as the Wait shape.

  For more information, see Pausing and resuming Processes in Cases.

- Use standard routing features.

  For more information, see Assigning tasks to users.

- Identify the list of Case Types whose instances the system associates with one another at run time.

PEGA ©2025 Pegasystems Inc

A Case and its association with other Cases provide a more comprehensive understanding of how various Cases are related. For more information, see Understanding Case hierarchy.

- Create child Cases when your work can be broken down into multiple pieces that have a relationship or dependency with one another. Use the following criteria to identify when to use a child Case:
  - There is a distinct Case Lifecycle from the top-level Case Type.
  - This work can be run independently or by many top-level Case Types.
  - You need to report on the work independently.
  - This work will progress in parallel to other child Cases or the top-level Case.

  For example, a Case Type that represents opening a new bank account requires that the applicant has a credit check. Opening a new bank account is visualized in a Case Type, but the credit check has its own Case Life Cycle that is critical to the overall Microjourney® of opening an account. The Credit Check should be a child Case because a credit check can run independently from other Processes in opening a bank account. If you separate the credit check into its own child Case, it can be reused by other top-level Case Types, such as applying for a loan. For more information, see Creating a child Case Type in Case Designer and Understanding Case hierarchy.

- Use standard Views and design templates, and ensure that all Assignment forms are configurable with the Configure form.

  For more information, see Designing the UI in Case Types.

- Add correct validation to forms that require it, and add Lifecycle validation on the **Data Model** tab of a Case Type.

  Form validations ensure data integrity regardless of the Channel through which the submission of data occurs. For more information, see Restricting user input in a field and Configuring Case-wide validation for correct data entry in fields.

PEGA ©2025 Pegasystems Inc

- Use attachment fields when you expect users to attach and view a file or document as part of the tasks that are assigned to the users.

  For more information, see Attaching content to a Case.

- Define custom attachment categories and avoid relying only on the out-of-the-box categories.

  Explicit categorization helps you group related files and use the reference while sending out attachments by email. For more information, see Categorizing Case attachments.

- Consider defining relevant Service-Level Agreements (SLAs) for every task to ensure that users get the appropriate amount of time to work on a task.

  You can also define SLAs for a Process, Stage, or Case. For more information, see Completing work on time.

- Consider relying on Urgency to convey the importance of a task.

  Higher values indicate higher importance. You can manage Urgency through a Local Action or an SLA. For more information, see Urgency.

- Use only the fields that enable tracking and auditing changes (field-level audit).

  For more information, see Enabling field-level auditing.

- When you need multiple users to access a Case simultaneously, consider setting the locking strategy to allow multiple users.

  For more information, see Managing concurrent access to a Case.

- Provide a few default tags that users can use while working on a Case.

  Relying on the tag infrastructure provides an easier way for associating multiple Cases. For more information, see Tagging in Cases.

PEGA ©2025 Pegasystems Inc

- Configure additional feed sources in addition to standard Pulse notifications so that users have a comprehensive overview of information related to a Case.

  For more information, see Adding feed sources to activity feeds and Configuring Pulse feed sources for Constellation.

- Use the Send Notification shape and Notification framework to send Notifications, especially when targeting large groups.

  For more information, see Sending event notifications from Cases.

- Extend standard Pulse Notifications to send updates to specific users who are related to the Case.

  By default, Pega Platform sends Notifications to followers, @ mentions, and users who participate in a thread. For more information, see Configuring Pulse email notifications.

- Mark important properties, Actions, Rules, and other assets as relevant records.

  As a result, you improve the authoring experience in App Studio by curating records that you need to include in your Case Types and data types. For more information, see Relevant records for Rule reuse.

For more information, see Case Management.

Data management

- Ensure that all queries use Data Pages and no top-level pages or direct report definition calls.

  For more information, see Using data pages.

- Ensure that all data types in an application are listed in the application record.

  For more information, see Managing application Cases and data.

PEGA ©2025 Pegasystems Inc

- Ensure that every data type has a lookup Data Page with the key as a parameter and a standard list page that receives a standard or full list of the data objects.

  For more information, see Creating a data type in Dev Studio.

- For greater reuse of assets and facilitated data management, create data relationships.

  For more information, see Data relationships.

- Use autopopulated properties for easier access to data from multiple hierarchically related Data Pages.

  For more information, see Data page reference by using autopopulated properties.

- Avoid using named Clipboard Pages. Instead, use Data Pages.

  For more information, see Differences between data pages and clipboard pages.

- For facilitated reuse of assets in App Studio and enhanced low-code development, mark properties as relevant records.

  For more information, see Relevant records for Rule reuse.

- For applications that you build with the Constellation architecture, apply the following best practices:
  - When you use Constellation DX API, avoid using savable or editable Data Pages to ensure that your application can access data correctly.

    For more information, see Data page limitations in REST calls.

  - Ensure that your Data Pages are configured correctly to support Constellation.

    For more information, see Data Pages in Constellation.

PEGA ©2025 Pegasystems Inc

For more information, see Data management and integration.

User experience

- For applications that you build with Theme Cosmos, use the prescribed experience, which includes the standard, three-pane layout of a Case. Modifying the standard layout breaks updates to future versions of the Theme Cosmos Ruleset and hampers migrations to upcoming Pega Platform architectures, such as Constellation. For more information, see Constellation and prescribed design and Out-of-the-box UI tools.
  - Section Rules
    To avoid problems with updates, do not point the following Section Rules to different templates:

Rules	Default templates
pyCaseMainInner	Main Case Page UI template
CaseHeader	Case header UI template
CaseActionHeader	Case action header UI template
CaseDetails	Case details UI template

    These Section Rules intentionally point to templates in the Theme Cosmos Ruleset. By design, developers can use App Studio or Dev Studio to override these Section Rules to easily map Case data to regions defined in the template. The functionality in Theme Cosmos requires this standard layout; pointing these Section Rules to another design template might break your current application or result in difficult updates to future versions of the Theme Cosmos Ruleset.

  - Harness Rules
    Avoid modifying the CSS and JavaScript that is attached to the predefined screen layouts in the following harness Rules:
    - *UserPortal*
    - *Review*

©2025 Pegasystems Inc

- Build Sections with out-of-the-box design templates (for example, 1 column; 2 column; List Item) whenever possible for improved UI consistency and faster development.

  For more information, see Creating Sections.

- Ensure that the business logic and actions that you use are in the correct Case management models and not embedded into a custom UI.
- Use out-of-the-box formats in skins.

  For more information, see Skins.

- Use mixins to maximize consistency and reuse and to minimize code and Rule maintenance.

  For more information, see Mixins.

- Use out-of-the-box CSS helper classes and avoid new skin formats or custom CSS to make small styling adjustments.

  For more information, see CSS helper Classes.

- Avoid using custom controls whenever possible.
- If you work with traditional UI architecture, follow best practices for design templates to maximize the reuse, consistency, and responsiveness of your UI.

  For more information, see Best practices for design templates.

- If you work with traditional UI architecture, ensure the consistency of your UI by following layouts best practices.

  For more information, see Best practices for layouts.

- Ensure that the UI that you build meets accessibility requirements.

  For more information, see Configuring an accessible UI.

PEGA ©2025 Pegasystems Inc

For more information, see User experience.

DevOps

- Use production Rulesets only for delegated Rules, not for the creation of new Rules.

  Avoid using production Rulesets in development and staging environments. For more information, see Production Rulesets.

- Use the Branch development feature while modifying existing applications with locked Rulesets instead of simple Branch creation in applications.

  For more information, see Branch operations.

For more information, see DevOps.

Security

- Adhere to the Security Checklist.

For more information, see Security.

Best practices for Pega Infinity application development

This document provides application development best practices for Pega Platform™ developers to create efficient, sustainable, and guardrail-compliant applications. We collated and consolidated the following list of best practices from the development experiences shared by lead system architects, consultants, and development teams.

The following Pega Infinity development best practices are sorted by impact category and then by the severity of consequences if developers do not follow these guidelines.

> **NOTE:** The following best practices do not apply to Constellation applications.

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Access Roles	Security	Implement custom security roles to maintain security requirements. Do not grant the *PegaRULES:SysAdm4* security role to end-user Access Groups. Roles should default to having the least privileges to end-user Access Groups.	High
Activity	Functionality	When you use the *History-Add* method, use field values to support localizations.	Medium
Activity	Functionality	Use `@equals` Functions instead of `'=='` for string comparisons.	Medium
Activity	Functionality	Do not use the *Obj-Save* method to save Rules. Instead, use `call pxUpdateRecord/pxCreateRecord` or `Call Save`.	Medium
Activity	Functionality	In unavoidable scenarios, if you use the *Page-Change-Class* method, use the dynamic class referencing (DCR) pattern rather than a hard-coded class.	Low
Activity	Maintanability	Do not use the *Page-New* method to create a page that you plan to use as a step page for the *Obj-Open*, *Obj-Browse*, *RDB-List*, or *RDB-Open* methods. These methods create the specified page themselves.	Medium
Activity	Maintainability	Do not use hard-coded reusable values in properties such as URL or classes (such as the activity class) in activities. Instead, use Data Pages, decision tables/map values.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Activity	Maintainability	The *Page-New* method with an empty class initializes the page with the class that is defined on the **Pages and Classes** tab. Consider using dynamic class referencing for easy maintenance and extensibility.	Medium
Activity	Maintainability	Use Automation Rules instead of Activity Rules when you create an API.	Medium
Activity	Maintainability	If scope is defined only within an activity, give precedence to a local variable over a property. If local variables are defined but are not used in the activity, consider removing them.	Low
Activity	Maintainability	Provide comments to describe the context of significant steps to improve the readability and maintainability of an activity.	Low
Activity	Maintainability	Comment out activity step lines to prevent them from running at run time during testing. Commented lines should usually be removed before merging your Rules. However, you might want to keep commented lines if they are helpful in debugging or if they help someone understand the activity.	Low
Activity	Maintainability	Start activity names with a verb or noun.	L
Activity	Maintainability	Activities should not have more than 20 steps.	Low
Activity	Maintainability	If an Activity Rule is required, set the activity type on the **Security** tab to remove warnings. For example, set **Utility (default)**, **Validate**, or **Load Data Page**.	Low

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Activity	Performance	Use Data Pages backup by RD or lookup rather than using the *Obj-Browse* method. Do not use the *Obj-Browse* method for lists of objects. If you use the *Obj-Browse* method, fetch only the column that you need and not the entire BLOB. If some BLOB data is used frequently, consider exposing that column.	High
Activity	Performance	The MaxRecords parameter in the *RDB-List* method is not automatically specified. If you leave the *MaxRecords* parameter blank, it defaults to 10,000. Consider specifying a maximum limit for improved query performance.	Medium
Activity	Performance	If you use a parameter with the *Obj-Browse* method (to/from/like), define a pre-condition on the activity step to verify that all required parameters exist before the *Obj-Browse* method is invoked.	Medium
Activity	Performance	Avoid using the `Infoforced` logging level in the *Log-Message* method. Remove unnecessary log messages before check-in. Change the logging level to prevent cluttering of log files and avoid potential performance impact.	Low
Activity	Security	The **Allow direct invocation from the client or a service** or **Allow invocation from browser** checkbox is cleared by default. If you select the **Allow direct invocation from the client or a service**" checkbox, you must add a privilege to the activity.	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Agent	Performance	Use queue processors or job schedulers instead of agents for better performance, resiliency, fault tolerance, and flexibility.	Medium
All Rules	Functionality	Do not assume that a Clipboard Page is available. Instead, perform the required verification checks before running a Data Transform or activity.	Medium
All Rules	Functionality	Add Pega units for all the supported Rule Types. Run the units during the development in Branches and during the merge process. Look for 100% Unit Test passed before merging the Branches.	Medium
All Rules	Maintainability	Reduce or eliminate the use of duplicate loops in Rules such as activities and Data Transforms, which can have a significant performance impact. Whenever possible, do as much processing as possible in one pass to avoid looping over the same data multiple times. When looping over a data structure, be aware that the larger the list, the more times processing occurs. For example, you can use a report definition Class Join feature to gather data in one database call rather than making a separate database call for every result returned. If services are called within a loop, use Data Pages to cache results to improve overall performance.	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
All Rules	Maintainability	Consider deprecating Rules to post warnings on usage. Provide documentation on the deprecated Rules.	High
All Rules	Maintainability	Avoid creating custom-coded Rules. If no better option is available, then ensure custom-coded Rules, such as activities with Java steps, non-autogenerated UI Rules, and Functions, are final.	High
All Rules	Maintainability	When you create a Rule such as an activity or Data Transform, use parameters to clearly define the inputs and outputs so that system architects understand how the Rule is used. If the Rule needs a specific value to process, rather than passing a complete page reference, pass a scalar parameter.	Medium
All Rules	Maintainability	When you create a new Rule that is similar to an existing Rule, minimize code duplication. If you change only values but do not change the code in a Rule, refactor the Rule to use parameters instead of creating multiple versions of the same Rule. If the same code is used multiple times, create well-documented, reusable assets that can be called from different places rather than copying the code. Determine the reusability of the Rule, as this determines where to place the Rule in the class and Ruleset hierarchy.	Medium
All Rules	Maintainability	Carefully consider the Applies To class of your Rules. Put Rules in a higher parent class if you anticipate reusing them across all children or if you want to avoid duplicate Rules. If a Rule is not reusable, put it	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		in the class where it is used. Do not flood abstract classes with Rules that can be used only in particular concrete classes. If a Rule is needed in multiple concrete classes, do not copy the Rule to the sibling. Instead, promote the Rule to the child class and withdraw the original Rule to avoid duplicating code.	
All Rules	Maintainability	Ensure that the name of a workbasket clearly identifies the purpose of the workbasket.	Medium
All Rules	Maintainability	Add test Rules to a separate test application that is built on the target application.	Medium
All Rules	Maintainability	Remove all unused Rules before you move to production.	Medium
All Rules	Maintainability	Document your Rules when the purpose or use of a Rule is not clear to improve maintainability. Document what the Rule does in the description and how the Rule is used in **Usage**. Ensure that the Rule label indicates the purpose of the Rule in human-readable format.	Low
All Rules	Maintainability	When defining parameters, select the correct data type and provide a meaningful description and purpose.	Low
All Rules	Maintainability	Provide appropriate comments when checking in a Rule after modification. Include a work item, such as a change request or error number, at the beginning	Low

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		of the description of the change. For example: `US-00001 – Removed pyWorkPage Page Delete Step from Activity`.	
All Rules	Performance	Use Alert logs, PAL, and Log Usage to monitor performance during development, UAT, and performance testing. If you observe performance issues, collaborate with performance tuning experts to resolve those issues. Assess performance load and verify the capacity of your production system.	High
All Rules	Performance	Implement Debug Log (oLog) in activities to debug and add warnings or errors to log files if an exception occurs. When writing log information externally, verify that logged information provides sufficient detail to assist in debugging an issue. By default, the application logs class name, activity name, step, date and time, requestor, and error type. Additional log items might include Case ID, Case Status, and operator. Make sure that you do not log sensitive information. Specify the correct log level settings so that production logs and PDC are not cluttered. Remove or comment out System.out.print after development.	Medium
All Rules	Performance	Avoid using large text values in top-level pages. Elasticsearch indexes all top-level properties when it is enabled on a class. Consider keeping large text values in embed page properties.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
All Rules	Performance	Use a wrapper activity to consolidate actions and reduce the number of requests to the server.	Medium
All Rules	Update	Do not block the availability Rules in the framework layer.	High
All Rules	Update	Replace deprecated Rule references with recommended, available Rules. Do not use deprecated activity methods.	High
All Rules	Update	Do not reference internal Pega Rules in implementations.	High
All Rules	Update	The use of locatable pages was deprecated in Pega Platform 7.3.1. Locatable pages are a legacy feature that is not supported by the Internal Language (IL) assembly. Using locatable pages would require falling back to legacy assemblers that do not utilize the IL assembly optimizations.	High
All Rules	Update	Use extensions wherever available, instead of overriding Rules. Use Pega Rules as-is and append new functionality in your application Rules.	High
All Rules	Update	Pega Infinity has been enhanced with more guardrails and validations and with better coding standards. Some legacy features are no longer supported and must be modified to Pega Infinity standards. Update references such as `$ANY` or `$NONE` with appropriate class references.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Circumstance-Definition	Functionality	List more likely outcomes in rows at the top of the table.	Medium
Circumstance-Definition	Maintainability	When circumstancing Rules, use circumstance templates rather than circumstancing on properties to allow implementation layers to customize the templates.	Medium
Class	Maintainability	Consider enabling **Propagate schema changes to child classes** for work classes to avoid explicit optimization at the customer end.	Medium
Class	Performance	Ensure that classes are not mapped to more than one purpose database table, such as *pr_data* or *pr_other*. Maintenance of exposed properties for more than one type of data prevents using the correct indexes.	High
Class	Performance	Consider not storing the history of classes in environments in which changes are frequent and for which you have no business reason to track updates.	Medium
Clipboard viewer	Performance	Identify and remove unnecessary Clipboard Pages at the end of each process. Redundant pages use JVM memory.	High
Data Pages	Performance	Running the load mechanism on every interaction for a Data Page can impact performance. Select this option only when the business use case requires the freshest information.	Medium
Data Pages	Security	Select the correct user-level Access Group for all node-level Data Pages. Create an Access Group used	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		specifically for batch processing (Data Page loading and agent activity processing).	
Data Pages	Update	Do not remove parameters for a Data Page. Never remove parameters from shipped Data Pages in the framework layer. The implementation layer might use your Data Pages with Key,Value; removing a parameter might make referenced Rules invalid, causing exceptions in flows.	High
Data Pages	Update	Avoid adding required parameters to old Rules. Avoid adding new required parameters to shipped Data Pages in the framework layer. The implemention layer might use these Data Pages, and adding new parameters might make their referenced Rules invalid, causing exceptions in flows.	High
Data Transform	Maintainability	Ensure that Data Transforms do not use the *pxExecuteAnActivity* Function to call an activity. Evaluate whether the called activity can be transformed to a Data Transform.	High
Data Transform	Maintainability	Consider modularizing a Data Transform so that it does not require more than 20 steps. This improves reusability and unit-level quality.	Medium
Data Transform	Maintainability	Inline Java is always overhead in terms of maintenance. Consequently, consider using a Data Page or Data Transform. If unavoidable, RUF is recommended.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Data Transform	Maintainability	Do not use more than five levels of nested steps in Data Transforms.	Medium
Data Transform	Maintainability	Consider adding comments to Data Transforms for easier readability, maintenance, and potential reuse.	Low
Data Transform	Maintainability	Under most circumstances, do not leave disabled steps in Data Transforms. Review and remove disabled steps if they are not required.	Low
Data Transform	Maintainability	To improve readability, avoid using `(<APPEND>)` `(<CURRENT>)` Instead, consider using **Append and map to**.	Low
Data Transform	Performance	Enable `call super-class Data Transform` checks only if you have a Data Transform with the same name in the class inheritance tree. Doing so avoids the overhead of looking up to parent classes.	Low
Data types	Update	Pega Infinity has updated data management and modeling features that provide better design and run-time performance. Update these components to new features. There are no auto-update utilities for most of the features in this area. Update old data tables to the new local storage data types to comply with Pega Infinity standards.	Medium
Database	Maintainability	Do not use custom store procedures to accomplish a task.	High
Decision tables	Functionality	Specify the most likely outcome rows before you specify less likely outcome rows.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Decision tables	Performance	To avoid slow performance, limit your decision tables to no more than 300-500 rows.	High
Decision tree	Functionality	Do not redirect a Rule if the property in the **Circumstance Value** field is referenced in a field on the **Decision** tab. Redirecting a Rule to itself is not possible. Avoid creating a circular set of redirection as it might cause run-time exceptions. For example, do not redirect a Decision tree A to Decision tree B if Decision tree B is already redirected to Decision tree A.	High
Decision tree	Performance	To avoid slow performance, limit your decision trees to no more than 300-500 rows.	High
Declare expressions	Maintainability	Adopt a naming convention for properties designed to be computed in Declare Expression Rules. Create Declare Expression Rules early, using stub or dummy Expressions. By following naming conventions, developers can easily identify declare expression properties and can avoid setting values procedurally. The use of reference properties in target or page context of an Expression can result in unpredictable behavior. Consider using non-referenced properties instead.	Low
Declare trigger	Functionality	If there are multiple triggers on the same class (user-defined or inherited), then the order in which the triggers fire is not guaranteed. Implement trigger activities in such a way that they are mutually	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		exclusive and independent of each other in terms of functionality.	
Declare-Index	Functionality	Do not create declarative indexes for top-level Single Value properties.	Medium
Declare-On Change	Maintainability	Do not use a **Complex Java Expression** under **When condition** in the **Conditions** field of the On Change **Properties** tab.	Medium
Exception handling	Functionality	Handle database calls and other integration calls by incorporating preconditions and transitions that are related to errors and exceptions. Use `Jump To Later Step` and `Exit Activity` effectively in activities to handle exceptions. Consider raising the correct PDC alerts when errors occur. Using the *Obj-open*, *Obj-refresh-and-lock*, *Obj-open-by-handle*, and *Obj-Delete-by-handle* methods requires the handling of lock acquisition failures. Use the *Obj-save* and *Commit* methods to handle potential DB transaction errors. Use *Connect-* methods to handle the potential for connector failure. Not handling an error might result in ungraceful errors.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Exception handling	Functionality	Implement the Error flow in all critical use cases.	
Flow	Functionality	Always provide the status of an Assignment and the status of work in the parameters of an Assignment task.	Medium
Flow	Functionality	Do not hardcode the workbasket or router user name in the routing unless it is important. Use a parameter page instead.	Medium
Flow	Functionality	Provide meaningful and readable audit information in a flow. For example, enter a correct action description in the Assignment task. Remember that whatever action comes in an Assignment is described in the work item history as `Assignment To` and the added text.	Medium
Flow	Functionality	As a best practice, use one of the following prefixes to support standard reports and maintain continuity in between your custom status and standard status values: • New- • Open- • Pending- • Resolved-	Low
Flow	Maintainability	To create the implementation layer Case Type, select the **Other** option and use DCR to specify the class name rather than selecting the Case Type in the `Create Case` smart shape.	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Flow	Maintainability	Ensure that a Business Rule flow is neat and readable, as it can be exposed to business users. Improve readability by adding annotation.	Medium
Flow	Maintainability	Ensure that utilities serve a business purpose. Instead of adding one utility that performs an entire step, separate the utility into separate smaller utilities, and use these smaller utilities as individual identity in the flows. This makes the flows descriptive and readable.	Medium
Flow	Maintainability	Ensure that the number of smart shapes (excluding connectors) is not higher than 15 per work flow.	Medium
Flow	Maintainability	Flow implementation must be model-driven. For example, identifying and using decision Rules in a flow rather than an activity makes the logic of the flow clearer.	Medium
Flow actions	Maintainability	Give flow actions names that describe the functions of the flow actions. For example: `AttachAFile`, `Notify External`.	Medium
Flow actions	Maintainability	Use a Validate Rule to perform data validations, and invoke the same Validate Rule from the flow action. Validating data with an activity in post-processing using the *Obj-Validate* or *Property-Set-Messages* methods does not support extensibility.	Medium
Map values	Functionality	Do not redirect a Rule if the property in the **Circumstance Value** field is referenced in a field on the **Decision** tab. Redirecting a Rule to itself is not possible. Avoid creating a circular set of redirection as	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		it might cause exceptions at run time. For example, do not redirect a Decision tree A to Decision tree B if Decision tree B is already redirected to Decision tree A.	
Map values	Performance	To avoid slow performance, limit your Decision Map Rules to no more than 300-500 rows.	High
Navigation	Maintainability	Do not configure the Control action by using a hard-coded class. Consider using dynamic class referencing (DCR) for easy maintenance and extensibility.	Medium
Portal	Update	Legacy fixed or custom Portals are no longer recommended in Pega Infinity, because they do not support cross-browser or responsive design. Debugging tools such as Live UI might not work for these legacy Portals. Replace fixed or custom Portals that lack responsiveness with Pega Infinity composite Portals.	Low
Property	Functionality	Use **Date Time** rather than the **Date** or **TimeOfDay** types for properties that are part of persistent objects in your application. Avoid using **Date** property types in persistent objects even if your application is used in only a single time zone, unless a time zone value is also stored in another property of the object.	
Property	Maintainability	Do not duplicate properties if you can avoid it. If multiple classes use a property, define the property once where it is accessible to all classes that need it.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		If one of two classes that need to use the property do not share a parent (for instance, one is Embed- and the other is not), you can copy the property. Ensure that you use the exact same name.	
Property	Maintainability	Avoid copying data. During processing, data should exist in only one place. In Pega Infinity, that means in a single property. If user input has a cascading effect (setting a value in one place creates defaults for a number of other values), then you can copy data if the effect is one-directional. In other words, you can copy data only when you can define it using something like a declare expression. Verify that you do not create circular dependencies when you copy data.	Medium
Property	Maintainability	Select property types from the **Type** drop-down list rather than specifying them in text.	Medium
Property	Maintainability	Configure property names that are based on nouns or noun phrases and describe the property.	Low
Report-Definition	Functionality	Consider using the SmartInfo capability to enrich the value of reports.	Low
Report	Performance	Fetch only the required columns and not all columns from the database.	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Report	Performance	Avoid unnecessary Join filters while fetching database records.	High
Report	Performance	Verify that the report definition filters load only necessary data to the Clipboard. Configuring filters correctly is better than using row visibility conditions to hide rows in a grid.	High
Report	Security	Add the correct privileges and restrictions to reports to control which user groups can run a report.	High
Report	Update	Do not remove properties from report definitions if they are declared as available, especially in the framework layer. Never remove properties from report definitions that are shipped, even if those properties are not used. If the implementation layer attempts to reuse a report definition for which properties have been removed, it affects the referencing Rules.	High
Rule-Obj-Activity	Performance	Always use *Obj-* methods (rather than *RDB-* methods) to ensure that the BLOB and columns stay in sync. Use *RDB-* methods only for complex query processing and connecting to an external database.	Medium
Rule-Utility-Function	Functionality	Use the `String.Equals ()` or `String.equalsIgnoreCase ()` Java method instead of the `==` or `!=` operators, to compare the value of a property to a literal string or to the value of another property.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
Ruleset	Update	Only application validation (AV) is recommended for Rulesets. For more information, see Ruleset validation modes.	Medium
Ruleset	Update	If you want to use an AV-mode Ruleset in more than one application, first create an additional Application Layer and then use the additional layer as a built-on application for all applications that use the AV mode Ruleset.	High
Section	Functionality	Use Refresh When and Refresh Other Section. Use Refresh When whenever possible, especially for read-only references that need to remain synchronized with data on the server, as it declares dependencies (`This section needs to be updated when .x changes`). When this is not possible, for example, if you have editable references that are updated based on other input or based on untrackable inputs such as button clicks, use a Refresh Other Section targeted refresh.	Medium
Section	Functionality	**Avoid Refresh section.** Refreshes interrupt the user experience. The user must wait, the screen seems to freeze until the section refreshes, and the user might lose focus and manually return to the original screen. Manage client-side interactions by using the change tracker whenever possible. For example, instead of	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		refreshing a section with a pre-processing Data Transform, call a Data Transform or use **set value** and set your visibility conditionals to evaluate on the client. **Refresh a section in the following circumstances:** • Property values are updated on the server and the new values must be reflected in the UI. • If you must submit an action that causes a change to more than one property and that occurs only on the client, such as deleting a row. • If you must remove parts of the UI must be removed from the DOM because of other input. **Do not refresh a section in the following circumstances:** • If you need to submit user input. Instead, use post value. • If you need to call a Data Transform or an activity after a user action. • If you need to recalculate visibility, the enabled/disabled state, or the read-only state. Select the **Evaluate on client** checkbox next to the **Expression** field. **Make a judgment call regarding refreshing a section in the following circumstances:**	

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		• If you have a complex Expression to set visibility, enabled/disabled, or read-only state that requires a When condition. Consider using a declare expression to provide a value for a property and conditionalize on changes to that declare expression. • If you must handle style changes based on value changes. • If you must validate input and potentially show messages related to various properties. Note that validation provided by the controls, combined with a Validate Rule on the flow action, should cover most situations.	
Section	Functionality	Use set value to set flags. Consider the following use case: you have a property *.pyIsThingVisible*. Put the property in a hidden input field in the UI and use `set value 'pyIsThingVisible = true` (or `false`). This is a more efficient method than using `Run Data Transform pzSetIsThingVisible` (and creating and maintaining the Data Transform).	Medium
Section	Functionality	Rely on the harness behavior to drive the read-only settings. A `Review` harness turns automatically when all the sections are in read-only mode. To maximize reuse, limit the number of sections that are using the read-only settings on the layout.	Low
Section	Maintanability	Do not hard-code the create work action class input parameters. Consider using dynamic class	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
		referencing (DCR) to improve maintainability and extensibility.	
Section	Maintanability	When deciding whether to use a section or to nest a dynamic layout, minimize the number of sections that you create to improve performance and maintenance. Use a section in the following circumstances: • You plan to use the section in multiple places. • The contents of the section must be refreshed from the server more often than the content around it. • The content of the section uses a different step page than the UI around it. In other circumstances, nest a dynamic layout.	Medium
UI Rules	Functionality	Do not use horizontal scrolling. Minimize vertical scrolling.	High
UI Rules	Functionality	Do not use fixed widths. For table-style layouts such as grids and tree grids, always set the width to 100%. This allows the display to adapt to the device that the viewer uses.	Medium
UI Rules	Functionality	Improve the user experience by removing unneeded clicks, especially for tasks that are simple or frequently repeated.	Medium
UI Rules	Functionality	To support localization, use field values for text that is displayed to users.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
UI Rules	Functionality	To ensure that client validation works, enable client validation on a harness.	Medium
UI Rules	Functionality	Select **Localize** on Section Rules.	Medium
UI Rules	Functionality	Avoid using icons for controls. Use link text or a button for clarity and ease of localization.	Low
UI Rules	Functionality	Bundle your transactions in minimum calls. For example, `call activity` and `refresh section` can be bundled.	Low
UI Rules	Functionality	Configure input validation on both the client and the server.	Low
UI Rules	Maintainability	Whenever possible, set the label value for a control from the short description for its property rather than from inline labels, which increase maintenance costs.	Medium
UI Rules	Maintainability	Use When Rules rather than Expressions in section/layout/control visibility conditions if the same condition is used in multiple places for easy maintenance and extensibility.	Medium
UI Rules	Performance	Consider using pagination of records for memory and performance optimization based on your specific use case requirements. Enable pagination when fetching records from the database.	High
UI Rules	Performance	Records that are fetched in memory should be just-in-time rather than persisted throughout the user session. Use the correct Data Page scoping.	Medium

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
UI Rules	Performance	Enable defer load for tabs/section for faster screen loads. Load data just in time for when the end user needs it.	Medium
UI Rules	Performance	Use JPEG and PNG images only if the image size is greater than 100x100 px. For smaller images, consider using the Icon font so that you can avoid time-consuming requests to the server.	Medium
UI Rules	Update	Non-autogenerated, custom, or hand-coded Rules are not likely to be cross-browser-compliant or responsive. For better performance, security, ease of maintenance, and support for cross-browser consistency and responsiveness, move to autogenerated UI components. Replace custom JavaScript, HTML, JSP, and other code with standard features that are available in Pega Infinity. Do not create non-autogenerated records, such as sections, HTML Rules, HTML fragments, and text files. Generate all screens for JSP (including *Rule-HTML-Property*, *Rule-HTML-Fragment*, and *Rule-Obj-HTML*.	High
UI Rules	Update	Pega Infinity has controls that support new features, such as offline capability, responsiveness, and cross-browser support. Deprecated or custom controls do not have these capabilities, so you should update to controls that are shipped with Pega Infinity. Replace custom or deprecated controls with the latest Pega Infinity controls. If no updated control is available, make custom controls cross-browser and responsive.	High

©2025 Pegasystems Inc

Rule Type	Impact category	Best practice	Severity
UI Rules	Update	Use the skin Rule for all styling needs in the application. This reduces maintenance cost and provides flexibility to style consistently, regardless of browser or device. Make applications cross-browser and cross-device-compliant. Inline styles and custom CSS must be re-engineered using Pega Infinity skin, mixins, and style formats.	High
Workbasket	Security	Add the correct roles to workbasket definitions to control who can process Assignments in the workbasket.	Medium

Customizing the App Studio navigation pane

Ensure an optimal experience for application development and management by editing the list of navigation pane menu options that App Studio displays. You can both add and remove elements from the navigation pane. For example, you can add settings that developers frequently use that are not included by default.

For greater flexibility and a more convenient experience, you can define visibility conditions for a customized display of App Studio, such as assigning the display to an Access Group. Consider a scenario in which business analysts focus on the data in your application. To tailor App Studio to their role, you can hide the navigation pane options connected to security, UI, and user management.

You extend the navigation pane by editing Rules in *Pega-Ext*, which is the super class that defines some foundational information, such as properties, field values, and Data Transforms. The *Pega-Ext-ExpExplorer* class stores Rules that define the App Studio navigation pane.

©2025 Pegasystems Inc

Adding custom landing pages to App Studio navigation pane

Provide an authoring experience tailored to different users of App Studio in your organization by adding new tabs in the navigation pane, and new submenus in the sliders that expand after you click the tab.

For example, you can add custom options in the **Settings** slider that are required by the application development teams in your enterprise. As a result, you speed up the development process and facilitate work in your organization.

1. In the navigation pane of Dev Studio, click App.
2. On the Classes tab, in the search field, enter `Pega-Ext-ExpExplorer`.

   > **Result:**
   >
   > Each subclass of the `Pega-Ext-ExpExplorer` class represents a UI element in App Studio, such as tabs in the navigation pane or the **Overview** section.

3. Right-click the class under which you want to add a new element in the navigation pane, and then click Create > SysAdmin > Class.

   > **For example:**
   >
   > To add a new submenu in the Settings slider, click Settings > Create > SysAdmin > Class.

4. In the **Class Record Configuration** section, in the **Label** field, enter a descriptive name for the new tab.
5. In the Class Name field, enter a hyphen followed by the name of the new class.

   > **For example:**
   >
   > The name of the new `Display` class in the `Settings` class is `Pega-Ext-ExpExplorer-Settings-Display`, as shown in the following figure:

©2025 Pegasystems Inc

Naming a new class

6. In the **Context** section, select your current application, and then click **Create and open**.
7. On the **General** tab, in the **Settings** section, enter the version of your application in which you create the class.
   The default class type is `Abstract` and the class inheritance is set as direct pointing to the parent class. Maintain the default values, as shown:

©2025 Pegasystems Inc

Creating the Display Class

8. On the **History** tab, provide a description and some usage details.
9. Click **Save** to create the new Class.

Result:

After you refresh the **App** explorer, it displays the new class, as shown in the following figure:

©2025 Pegasystems Inc

Newly created class

10. In the **App** explorer, right-click the newly created class, and then click **Create** > **Data Model** > **Field Value**.
11. In the **Field Value Record Configuration** section, in the **Label** field, enter `ExtensionLabel`, and in the **Field Name** field, enter `pyCaption`, and then click **Create and open**.
12. On the **Localized label** tab, in the **To** field, enter the name that you want to display on the slider or in the navigation pane, and then click **Save**.

For example:

Enter `Additional display settings`.

©2025 Pegasystems Inc

Result:

App Studio includes the new element, as shown in the following figure:

Customized App Studio slider

©2025 Pegasystems Inc

13. **Optional:** To change the display order of the tabs in the navigation pane or on the slider, override the *pyDefault* Data Transform from the parent class:
    a. In the **App** explorer, right-click the parent class of the newly created class, and then click **View** > **Data Model** > **Data Transform**.
    b. In the list of Data Transform instances, click the **pyDefault** Data Transform.
    c. In the Rule form header, click **Save as** > **Specialize by class or Ruleset**
    d. In the **Specialize Data Transform** Rule form, in the **Context** section, in the **Apply to** field, enter the name of the newly created class.

    > **For example:**
    >
    > Enter `Pega-Ext-ExpExplorer-Settings-Display`.

    e. Click **Create and open**.
    f. On the **Definition** tab, in the **Source** column for the `.pyRank` property, enter the number representing the rank of the new element.

    > **For example:**
    >
    > To place the new element at the top of the list in the slider, enter `1`, as shown in the following figure:
    >
    >
    >
    >
    > *pyDefault Data Transform*

    g. Click **Save**.

> **Result:**
>
> The slider displays the **Additional display settings** as the first item on the list, as shown in the following figure:

©2025 Pegasystems Inc

Customized order of items

14. **Optional:** To customize the header of the new landing page, create a section:
    a. In the App explorer, right-click the newly created class, and then click **Create > User Interface > Section**.

©2025 Pegasystems Inc

b. In the **Section Record Configuration** section, in the Label field, enter `pyExpressLPHeaderTitle`, and then click Create and open.
c. Configure the section as required.
For more information, see Using full Section editor.

Otherwise, the header displays the extension label.

15. Show custom content on your new landing page by creating a section:
    a. In the App explorer, right-click the newly created class, and then click Create > User Interface > Section.
    b. In the **Section Record Configuration** section, in the Label field, enter `pyExpressLPContentTemplate`, and then click Create and open.
    c. Configure the section as required.
    For more information, see Using full Section editor.
16. **Optional:** To initialize additional values that could not be initialized during the default Data Transform, override the *pyInitializeExtension* Activity:
    a. In the navigation pane of Dev Studio, click Records.
    b. Expand the Technical category, and then click Activity.
    c. In the list of Activity instances, click the pyInitializeExtension Activity.
    d. In the Rule form header, click Save as.
    e. On the **Save as Activity** form, in the **Context** section, enter the class and Ruleset of the newly created App Studio extension, and then click Create and open.

    > **For example:**
    >
    > In the Apply to field, enter `Pega-Ext-ExpExplorer-Settings-Display`.

    Ensure that you leave the `pyInitializeExtension` value in the Label field.
    f. Configure the Activity to meed your business needs, and then click Save.
    For more information, see Configuring steps in an activity.

©2025 Pegasystems Inc

17. **Optional:** To perform post-processing after the new landing page is loaded, override the *pyPostProcessExtensions* Activity:
    a. In the navigation pane of Dev Studio, click **Records**.
    b. Expand the **Technical** category, and then click **Activity**.
    c. In the list of Activity instances, click the **pyPostProcessExtensions** Activity.
    d. In the Rule form header, click **Save as**.
    e. On the **Save as Activity** form, in the **Context** section, provide the Class and Ruleset of the newly created App Studio extension, and then click **Create and open**.

    > **For example:**
    >
    > In the **Apply to** field, enter `Pega-Ext-ExpExplorer-Settings-Display`.

    Ensure that you leave the `pyPostProcessExtensions` value in the **Label** field.
    f. Configure the Activity to meed your business needs, and then click **Save**. For more information, see Configuring steps in an activity.

Conditionally displaying custom App Studio landing pages

Adjust App Studio to better suit users with different roles in your enterprise by conditionally displaying custom landing pages.

For example, for business analysts, you can hide the **Case Types**, **Channels**, and **Settings** tabs from their landing pages.

You conditionalize the visibility of landing pages by editing the *pyShouldExtensionBeLoaded* When Rule. Some tabs include this Rule by default. For other tabs, and custom tabs, you must create the Rule. The default value of this Rule is `true`.

1. Navigate to the *pyShouldExtensionBeLoaded* When Rule:

©2025 Pegasystems Inc

Choices	Actions
Open an existing Rule	a. In the navigation pane of Dev Studio, click **App**. b. In the search field, enter `Pega-Ext-ExpExplorer`. c. Right-click the class with the custom landing page, and then click **View** > **Decision** > **When**. d. In the list of When Rule instances, open the **pyShouldExtensionBeLoaded** Rule.
Create a new Rule	a. In the navigation pane of Dev Studio, click **App**. b. In the search field, enter `Pega-Ext-ExpExplorer`. c. Right-click the class with the custom landing page, and then click **Create** > **Decision** > **When**. d. In the **When Record Configuration** section, in the **Label** field, enter `pyShouldExtensionBeLoaded`, and then click **Create and open**.

2. Define conditions for display of the landing page by using the Condition builder. For more information, see Defining Conditions in the Condition builder.
3. Click **Save**.

©2025 Pegasystems Inc

Understanding Personas and access management

For the best user experience, first determine user needs by analyzing the roles and Personas needed in your project. When you know who uses your application, and how they use it, you can shape the interactions in such a way that the application provides only the most relevant content to every user.

Personas, operators, and access management

Pega Platform™ manages users, their access privileges, and work routing through Personas and Access Groups. A Persona is a type of user for your application that corresponds to a work role. An operator is a human user or an automated process. An Access Group is a group of privileges in the system. You define Personas, operators, and Access Groups during development.

Persona

Represents a type of user for your application, defined by business role and the resulting responsibilities in a business process. A user's Persona determines the kinds of tasks that this user normally performs in the system, and the data needed for those tasks. You can create Personas not only for human users, but also for automated, unattended processes.

Access Group

Represents a group of privileges in the system. Every time you create a Persona, an Access Group is created in the background. However, not all Access Groups correspond to a Persona. Access Groups can include not only human users, but also automated processes such as robotic systems.

Operator

Defines a unique identifier, password, preferences, and personal information for a user. You can create operators not only for human users, but also for automated, unattended processes, such as robots.

PEGA ©2025 Pegasystems Inc

For example:

In a hiring application, the Personas would be the applicant, an HR representative processing the application, and the hiring manager. The applicant needs to have access to the submission Portal, while the HR representative and the hiring manager both have access to the company Portal. Only the hiring manager can approve or reject applications. These roles correspond to different permissions and default Channels in the application.
The following figure shows the hiring process with Personas and channels that they can access:

©2025 Pegasystems Inc

Personas in the hiring process

Access Group settings define, among other things, the user roles and privileges, the default Portal that is accessible to users, the work pools, and the types of work items that operators can perform. Operators can belong to multiple Access Groups and switch between them, but only one Access Group can be active for a given operator at a given time.

To enable users to start using your application, define operators and Access Groups. By creating operators and Access Groups, you specify which elements of your application users can access. As a result, you can ensure that every user can access only the features that are relevant to their role. By exposing users only to data that they need, you increase efficiency and improve workload management in your application.

For relevant training materials, see the Organization records and Creating and managing teams of users modules on Pega Academy.

- **Operators**
- **Learning about Access Groups**

Related concepts

- Creating a Microjourney for customer success
- Adding Personas to organize users
- Designing and tracking Microjourney development

Operators

An operator defines a unique identifier, password, preferences, and personal information for a user. Create operators so that people and processes can access your application.

©2025 Pegasystems Inc

For example, in an application for reviewing loan requests, you can create operators for customer service representatives (CSRs), reporting managers, and an administrator to manage the application from the technical side.

By creating operators you provide access for users and processes to your application so that processing work can happen, and so that business processes can reach successful resolution. You can create operators not only for users, but also for automated, unattended processes. Unattended operators are robotic automation virtual machines (VMs). The system generates unattended operators for each registered VM in a robotic process automation (RPA) solution. Creating unattended operators increases the automation of your business processes, lowers costs, and speeds up work resolution.

Creating operators gives you the following benefits:

Increased security

Every operator can have a unique password with which to log in to your application. As a result, you protect your data and operations from inappropriate or random users.

Improved routing of work

Every operator has an individual queue of Assignments to complete that Pega Platform refers to as a Worklist. Additionally, you can give an operator access to a Work Queue that groups Assignments that are available for members of a specific team. For every operator, you can specify the level of relevant skills, for example, you can indicate that an operator is a highly advanced Java developer, or speaks French fluently. Grouping Assignments and defining skillsets improves the routing of work and ensures that an application assigns tasks to the most appropriate users. For every operator, you can also define periods of unavailability and a substitute worker or Work Queue to ensure continuous Case processing.

Relevant access type

You can define which data and operations an operator can access and perform in your application. For example, you can ensure that only a manager can approve or reject new job candidates, or that a CSR can access all information that a Case

PEGA ©2025 Pegasystems Inc

requires to reach resolution. Consequently, the users of your application can perform only relevant actions and process their work faster.

The following figure shows a Rule form for a sample operator ID. The operator can use all Access Roles available in the application, for example, to recreate and debug issues that users might encounter,

Sample operator Rule Form

Clusters and operator IDs

After you save a new, or update an existing operator ID instance, the change might not be reflected on another node in a cluster until the Pega-RULES agent on that node performs the next system pulse — typically after no more than 60 seconds. Unlike instances of most other *Data-* classes, the system saves operator ID instances to the Rule cache. As a result, until the next time that the Rule cache is synchronized, one node might access a stale copy from its Rules cache.

©2025 Pegasystems Inc

Bulk operator load

To save time, you can create operator ID instances by importing a comma-separated values (CSV) file, such as a Microsoft Excel spreadsheet.

Operator ID property

After a requestor logs in, the operator ID identifier is available on the *pxRequestor* page as the *pxUserIdentifier* property.

Operator IDs and external identity providers

If you implement authentication by using an external identity provider (IdP), the login process accesses IdP for authentication and ignores the password in this operator ID instance. However, the system still requires an operator ID data instance for each user.

Operator passwords

The system saves passwords for operator IDs as hashed values in the PegaRULES database, by first salting the clear text password value and then hashing it using the default bcrypt algorithm. The system uses two property types when changing the password: Password type for the **New Password** field, and Text type for the **Confirm Password** field. The *Data-Admin-Operator-ID.pyPwdCurrent* property stores the hashed password after the password passes validation.

- **Creating an operator ID**
- **Defining user contact information and application access**
- **Defining work routing settings for an operator**
- **Defining operator availability**
- **Defining security information for an operator**
- **Deleting operators**

PEGA ©2025 Pegasystems Inc

- **Creating a Work Queue**
- **Creating work groups**
- **Updating the organizational structure by using the organizational chart**
- Bcrypt hashing algorithm for Password property types
- Managing Requestor Type data instances
- Fields for operator contact information and application access
- Defining security information for an operator

Creating an operator ID

Provide users with access to your application by creating operator IDs. When you create an operator ID, you can specify the background and general information about the user, define application access, and configure password credentials for the user.For example, in a banking application, create operator IDs for customer service representatives (CSRs) who review loan requests, and the manager to whom the CSRs report.

> **Before you begin:**
>
> To create an operator ID, you must have the *pxCanManageUsers* privilege, which is part of the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Create** > **Organization** > **Operator ID**.
2. In the **Short description** field, enter the full name of the new operator.
   The value that you enter populates the **Full name** field in the operator contact information on the **Work** tab. The maximum length of the description is 64 characters.
3. In the **Operator ID** field, enter a unique identifier.
   When you enter the identifier, consider the following guidelines:
   - In addition to letters and digits, the identifier can include the following characters: period (.), single quote ('), tilde (~), underscore (_), exclamation

©2025 Pegasystems Inc

point (!), ampersand (&), octothorpe (#) and no more than one at (@) character. Avoid using the forward slash (/) or backslash (\) characters in the identifier.
- An operator ID can consist of a maximum of 128 characters.
- Special processing applies to any user identifier that begins with the word `External`. Use such identifiers only when defining external operators that use Directed Web Access (DWA) for one-time processing of Assignments.
- Avoid using `system` as an operator ID name. This term is reserved and refers to agents.

4. Click **Create and open**.

The following video shows creating an operator ID:

Result:

The operator Rule form is displayed on the page and you can continue to add information about the user. For example, you can configure access settings or provide organizational information.

What to do next:

- Define an Access Group and contact information for the operator. For more information, see Defining user contact information and application access.
- Define organizational information for the operator. For more information, see Defining work routing settings for an operator.
- Define a password for the operator. For more information, see Defining security information for an operator.

- Adding an operator by using the organizational chart
- Operators

©2025 Pegasystems Inc

Defining user contact information and application access

Define contact information and application access for a user so that all operators can communicate and access the required applications. The Access Groups that you specify affect which applications a user can access and what their role is. For example, you can configure access settings so that a user can access a Loan requests application both as a manager and as a customer service representative (CSR).

Before you begin:

- Create a user as described in Creating an operator ID or Adding an operator by using the organizational chart.
- To view an application's access settings, you must have the *pxCanManageUsers* privilege, which is part of the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Organization** > **Operators**.
2. On the **Operators** tab, select the user that you want to edit.

   TIP: To find your user faster, in the **Search Text** field, enter the user name, and then click **Search**.

3. **Optional:** To more quickly distinguish between operators, on the **Profile** tab, in the **Contact Information** section, provide a user image:
   a. Click **Choose File**.
   b. In the files window, select a graphic file that you want to use, and then click **Open**.
   c. Click **Upload Image**.

©2025 Pegasystems Inc

> **Result:**
>
> The image is visible after you save the user record.

4. Provide basic information about the user:
   a. **Optional:** To inform users how to address the user, in the **Title** field, enter a relevant personal title.

   > **For example:**
   >
   > Enter `Mr`, `Mrs`, or `Dr`.

   b. **Optional:** To provide personal information about the user, complete the **First Name** and **Last name** fields.
   c. In the **Full name** field, enter a value that you want to use to represent the user across your system.
   By default, the system autopopulates the **Full name** field with the value that you provide in the **Short description** field when you create a user ID. The user's full name is visible, for example, in the list of operators.
   d. **Optional:** To inform users about the job title of the user, complete the **Position / job title** field.

   > **For example:**
   >
   > Enter `Software engineer`.

   e. **Optional:** To provide the phone contact information of the user, complete the **Phone** field by entering the phone number in the internationally standardized E.164 format. E.164 numbers can have a maximum of fifteen digits and use the following format: [+] [*country code*] [*subscriber number including area code*].

   > **For example:**
   >
   > For a user in the United States, enter `+16175551212`.

©2025 Pegasystems Inc

The system uses the phone number to send Short Message Service (SMS) Correspondence to the user.

f. **Optional:** To provide the email contact information of the user, complete the **Email** field.
The system uses the email address to send email Correspondence to the user.

5. In the **Application Access** section, list the Access Groups that the user can access at run time:
   a. In the **Access Group** field, enter an Access Group that you want to associate with the user.

      > **TIP:** To see the roles in an Access Group, click the **Expand row** icon.

   b. **Optional:** To provide additional roles and access to additional applications, click **Add item**, and then in the **Access Group** field, enter another Access Group.

      > **For example:**
      >
      > You can provide multiple roles for a user within one application, such as a manager role and a CSR role.

   c. Indicate the default Access Group by selecting a relevant radio button.

      > **Result:**
      >
      > When a user logs in, the system directs the user to the application and role that the default Access Group defines. For example, if you define `LoanRequests:Managers` as a default Access Group, the user logs into the Loan requests application with a manager role.

6. In the **Localization** section, list the default locale and time zone of the user.

©2025 Pegasystems Inc

For example:

The following figure shows the sample operator ID Rule form. The operator can view content associated with all Access Group available in the application to recreate issues that other users might encounter:

Operator ID Rule form

What to do next:

Before you can save the user record, you need to define the organizational unit of the user in your organization. For more information, see Defining work routing settings for an operator.

- Fields for operator contact information and application access
- Learning about Access Groups
- Operators

©2025 Pegasystems Inc

Defining work routing settings for an operator

For continuous and efficient workload management and routing, define skills, teams (work groups), and Work Queues for operators. As a result, an operator can receive work based on skill set and team membership. For reporting purposes and advanced routing use cases, configure the reporting structure. For example, when you define the reporting manager of an operator, the manager might receive notifications about the operator's progress on a Case.

> **Before you begin:**
>
> Create an operator so that a user can access your application. For more information, see Creating an operator ID or Adding an operator by using the organizational chart.

Every operator has a unique Worklist that accumulates tasks that the specific operator can retrieve. Additionally, an operator can access a Work Queue that collects tasks for a team to which the operator belongs. Any team member can pick up tasks from a Work Queue. Another name for a team is a work group.

For an automated selection of tasks to process, operators can use the Get Next Work logic. Get Next Work automatically prompts users with a top-priority Assignment, such as the one that currently has the highest Urgency. For more information, see Automated and intelligent work assignment with Get Next Work.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Organization** > **Operators**.
2. On the **Operators** tab, select an operator that you want to edit.

> **TIP:** To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.

©2025 Pegasystems Inc

3. On the **Work** tab, in the **Organizational unit** section, define the affiliation of the operator within the organization by clicking **Update**, and then providing a relevant organization, division, and unit.

   > **For example:**
   >
   > For an organization, enter the name of your company, for example, `UPlusBanking`, and then provide a division and unit, such as `BankingOperations` for a division and `Mortgages` for a unit.

4. In the **Team** field, enter a work group to which the user belongs.
   A work group defines routing of tasks and a reporting manager of the user.

   > **For example:**
   >
   > If you create Team A to review mortgage requests and Team B to investigate issues with mortgage operations, the two teams can use separate areas of your application so that the work is logically separated.

5. **Optional:** To assign a user to multiple work groups, click **Add a work group**, in the text box enter a work group, and then select the radio button next to the work group that you want to set as the default.
6. **Optional:** To enhance the reporting options of your application, in the **Reports to** field, enter the reporting manager for the user.
7. **Optional:** To enable routing of tasks based on a skillset, in the **Skill** field, enter the ability of the user, and then in the **Rating** field, enter the level of the ability.
   You can apply a rating from 1 to 10, where 10 implies the highest level of the skill.

   > **For example:**
   >
   > To indicate that a user is a fluent speaker of French, in the **Skill** field, enter `French`, and then in the **Rating** field, enter `10`.

©2025 Pegasystems Inc

8. **Optional:** To allow the user to process Assignments from a Work Queue that corresponds with the user's work group, in the **Work Queue** section, click **Add item**, and then provide a Work Queue and an Urgency Threshold.
   During Get Next Work processing, your application ignores Assignments with Urgency lower than the Urgency threshold. For more information about Get Next Work, see Automated and intelligent work assignment with Get Next Work and Get Next Work customization.
9. **Optional:** To assign the user to multiple Work Queues, click **Add item**, and then repeat step 8.
10. **Optional:** To trigger the Get Next Work algorithm to retrieve Assignments from the Work Queues of the user first, select the **Get from Work Queues first** checkbox.
    Otherwise, Get Next Work picks up the top Assignment on the user Worklist, and accesses Work Queues only if this user's Worklist is empty.

    > **For example:**
    >
    > If a user is a member of the Team A to review mortgage requests, the Next Get Work algorithm suggests Assignments that the application routes to the Team A Work Queue first and then moves to the Assignments from the user's individual Worklist.

11. **Optional:** To trigger the Get Next Work algorithm to retrieve Assignments from all Work Queues that correspond with the team that you provided in step 4, select the **Use all Work Queue Assignments in user's team** checkbox.
12. **Optional:** To trigger the Next Assignment item to consolidate Assignments from all of the Work Queues of the user, sort by Assignment Urgency, and return the most urgent Assignment in any Work Queue, select the **Merge Work Queues** checkbox.
    If you select the **Use all Work Queue Assignments in user's team** checkbox in step 11, the system selects the **Merge Work Queues** checkbox by default and you cannot clear this checkbox.
13. Click **Save**.

PEGA ©2025 Pegasystems Inc

For example:

The following figure shows the **Work** tab of a sample operator. The tab includes information about organizational structure, routing work, skills, and a planned absence of the user assigned to the operator ID:

Defining routing settings

- Fields for operator teams, Work Queues, and schedules
- Operators
- Automated and intelligent work assignment with Get Next Work
- Intelligent routing

Defining operator availability

Ensure that the workflow in your organization is uninterrupted by defining operator availability and substituting operators that can receive work when the operator is unavailable. As a result, you ensure that the processing of your business Cases continues even when some of the users are unavailable to perform work.

©2025 Pegasystems Inc

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Organization** > **Operators**.
2. On the **Operators** tab, select an operator that you want to edit.

> **TIP:** To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.

3. **Optional:** To allow applications to route additional Assignments to the Worklist of the user that the operator ID identifies, on the **Work** tab, in the **Availability** section, select the **Operator is available to receive work** checkbox.
   If you leave the checkbox cleared, the operator can log in, enter work items, or complete Assignments that are already in the Worklist. The operator can receive work from other operators and from a manager.
4. **Optional:** To define the working days and national holidays for the operator, in the **Business Calendar** field, select the relevant calendar.

> **For example:**
>
> Enter `USDefault`.

5. If the operator might be unavailable in the future, in the **Scheduled absences** section, in the **Unavailable form** and **To** from, define a time period during which the operator is unavailable to perform work.

> **For example:**
>
> If the operator has a scheduled medical leave between January 1 and January 15, select the relevant dates.

6. **Optional:** To define more periods of unavailability for an operator, click **Add item**, and then repeat step 5.

©2025 Pegasystems Inc

7. **Optional:** To define how an application routes work when the operator is unavailable, in the **When unavailable** section, in the **Substitute operator type** list, select who or what Work Queue receives work:
   - To route work to another operator, select **Operator**.
   - To route work to a list of Assignments for a team, select **Work queue**.
8. **Optional:** To determine a substitute for the unavailable operator at run time, in the **Decision tree to find substitute** field, enter a decision tree that evaluates conditions and finds another assignee for the Assignment.
   Based on your configuration in step 7, select a decision tree that returns either an operator ID or a Work Queue.
9. **Optional:** To identify the operator ID or Work Queue name of a substitute for new Assignments that an application routes to the unavailable operator, enter a relevant value in the **Default to assignee** field:
   - To route work to a specific user, enter the operator ID of the user.
   - To route work to a team's list of Assignments, enter the Work Queue.

   The system uses the value that you provide if you do not specify the decision tree to determine a substitute operator or the decision tree does not return a result.
10. Click **Save**.

Related tasks

- Creating an operator ID
- Creating work groups

Defining security information for an operator

Ensure that your organization complies with security policies by defining security information for the operators in your system. You can manage operator authentication, passwords, and license types, to allow Rule check out, and enable or disable the operator. As a result, operators can access your application safely, and then perform only the actions that are relevant to their roles.

©2025 Pegasystems Inc

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Organization** > **Operators**.
2. On the **Operators** tab, select an operator that you want to edit.

> **TIP:** To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.

3. On the **Security** tab, in the **Access Settings** section, secure access to your application with a password:
   a. Click **Update password**.
   b. In the **Change Operator ID Password** dialog box, in the **New password** field, enter the new password.
   c. In the **Confirm new password** field, reenter the password.
   d. Click **Submit**.

> **NOTE:** If the operator is provided with Pega Platform, enter a password that is consistent with your security policies, and then send the new password to the enabled operator.

The system converts the password to a hash value by using the salted bcrypt algorithm. The Storage Stream (BLOB) column of the pr_operators table contains the hashed value. By using the **View XML** action, you can discover only the hashed form of any operator password.

For more information about security polices, see Security policies.

4. If the operator is an unattended operator, select the **This is an unattended operator (robot)** checkbox.

©2025 Pegasystems Inc

Unattended operators are robotic automation virtual machines (VMs). The system generates unattended operators for each registered VM in a robotic process automation (RPA) solution.

5. **Optional:** To allow this operator to update Rules in Rulesets that use Rule checkout, select the **Allow Rule check out** checkbox.
6. **Optional:** To authenticate this operator only through external authentication facilities, select the **Use external authentication** checkbox.
7. **Optional:** To prompt the operator to change their password the next time the operator logs in, select the **Force password change on next login** checkbox.
8. **Optional:** To disable the operator, select the **Disable Operator** checkbox.
9. **Optional:** In the **Starting activity to execute** field, specify the first activity that the system runs after authentication for this user is complete.
   The default is `Data-Portal.ShowDesktop`.
10. In the **License type** list, indicate the operator type:
    - To indicate that the operator is a person who does business operations by using an application or customer-created interface, select **Named**.
    - To indicate that the operator is an abstract user to run agents, services, and other Background Processes, or an external user that interacts with the application through the Directed Web Access feature, select **Invocation**.

      For unattended operators, the system selects **Invocation** by default.
11. Click **Save**.

> **NOTE:** If you modified security information in your own operator record, you must log out and then log back in so that Pega Platform can apply the security change.

©2025 Pegasystems Inc

For example:

The following figure shows the **Security** tab of the sample operator. The operator can check out Rules and must update the password upon the next login.

Defining operator security settings

- Operators

©2025 Pegasystems Inc

- License compliance
- Checking out a Rule

Deleting operators

To ensure that only designated workers can access your application, you can disable access for users that are no longer active, for example, when they no longer work in your organization.

You cannot delete an operator when a data instance, such as an organization unit or Work Queue, still references the operator. To save time, instead of deleting all of the references individually, change the operator password and mark the user as unavailable.

1. Ensure that the user that you want to delete is not logged in.
2. Transfer or complete all Assignments in the Work Queue for the user.
   For more information, see Transferring Assignments.
3. Update the password to a value that is unknown to the user, so that the person can no longer log in.
   For more information, see Defining security information for an operator.
4. Ensure that the user has no Rules checked out. If any Rules are checked out, sign on and delete or check in the checked-out Rules.
   You cannot delete an operator ID if the operator has Rules checked out. Ensure that the operator deletes or checks in all Rules in their personal Ruleset. For more information, see Checking in a Rule.
5. Mark the user as unavailable to receive work, and then define the user's departure date and a substitute operator.
   For more information, see Defining operator availability.

- Operators

Creating a Work Queue

To improve workload management and task routing in your application, create a Work Queue that holds Assignments for operators and robotic queues. Because you

©2025 Pegasystems Inc

associate a Work Queue with a group of users or a robotic queue, you logically and efficiently categorize work inside your organization.For example, create a Work Queue for customer service representatives (CSRs) who manage loan requests to ensure that an application routes loan request tasks to users with relevant skill set and qualifications.

> **NOTE:** When you create an application, the system creates a default Work Queue and work group. Because of mutual dependencies, you need to provide a Work Queue when you create a work group, and when you create a work group, you need to provide a Work Queue. Use the default values, and then provide the actual values after you create your work group and Work Queue.

1. In the header of Dev Studio, click **Create** > **Organization** > **Work Queue**.
2. On the **Create Work queue** tab, in the **Short description** field, describe the purpose of the Work Queue.

   > **For example:**
   >
   > Enter `Loan requests`.

3. In the **Work Queue** field, enter a name that clearly conveys the purpose of the Work Queue and its contents to users of the application.

   > **NOTE:** For a better distinction between Work Queues and other organization elements, append the `WQ` characters to the Work Queue name.

   > **For example:**
   >
   > Enter `LoanRequestsWQ`.

4. Click **Create and open**.

©2025 Pegasystems Inc

5. On the Work Queue tab, in the **General** section, specify a Work Queue type:

Choices	Actions
Create a Work Queue that distributes Assignments to a virtual machine	a. In the Type list, select Robotic. b. In the Maximum queue length field, enter an integer that defines the maximum number of tasks that a queue can contain. c. In the Maximum automation execution time (seconds), enter the maximum amount of time in seconds that the virtual machine can take to run the Assignment before the system cancels the Assignment.
Create a Work Queue that distributes Assignments to users	In the Type list, select Standard.

6. In the **Organization** section, specify an organization layer that contains your Work Queue:
   a. In the Name field, enter the name of an organization to which the Work Queue belongs.

      For example:

      Enter `UPlus`.

   b. In the Division field, enter the Work Queue division.

      For example:

      Enter `BankingOperations`.

   c. In the Unit field, enter the Work Queue unit.

©2025 Pegasystems Inc

> **For example:**
>
> Enter `LoanRequests`.

d. In the **Work group** field, enter a work group that you want to associate with a Work Queue.
A work group consists of a Work Queue and a manager to which the Work Queue members report.

7. **Optional:** To notify operators about new Assignments, in the **Contacts** section, in the text field, enter the ID of an operator who receives a notification when an application routes work to the Work Queue.
8. **Optional:** To add more operators who receive notifications, click **Add item**, and then enter the operator ID.
9. **Optional:** To specify which users can process work in this Work Queue, in the **Roles** section, in the text field, enter the required role for a user to resolve Assignments from the Work Queue.

> **CAUTION:** If you leave this field blank, any user who can access the Work Queue can process work.

10. **Optional:** To add more roles, click **Add item**, and then enter a role.
To process work, a user needs to have at least one role from the set of roles that you specify.
11. Click **Save**.

> **For example:**
>
> The following figure shows a sample configuration for a Work Queue that queues Assignments for workers in the UPlus organization. Two operators receive a notification when a new Assignment appears in the Work Queue. Operators with

©2025 Pegasystems Inc

the `Candidate1:ReviewRequest` role can access the Work Queue and process the Assignments.

©2025 Pegasystems Inc

©2025 Pegasystems Inc

Configuring a Work Queue

- Creating an operator ID
- Configuring advanced settings for new applications
- Copying a Rule or data instance
- Creating a Rule specialized by circumstance

Creating work groups

Manage work inside your organization logically and efficiently by creating work groups. A work group connects a manager and a group of reporting users to enable relevant workflow management and improve communication. You can also add authorized managers who can view the associated Work Queues. For example, at run time, when a customer service representative misses a deadline for a task, a reporting manager receives a notification. Your application determines the manager by using a work group setting.

NOTE: When you create an application, the system creates a default Work Queue and work group. Because of mutual dependencies, you need to provide a Work Queue when you create a work group, and when you create a work group, you need to provide a Work Queue. While following the procedure below, use the default values, and then provide the actual values after you create your work group and Work Queue.

1. In the header of Dev Studio, click **Create** > **Organization** > **Work Group**.
2. On the **Create Work Group** tab, in the **Short description** field, briefly describe the purpose of the group.

   For example:

   Enter `Loan application reviews`.

©2025 Pegasystems Inc

3. In the **Work Group Name**, enter a label for the work group.
   As a best practice to provide more information in the name, append to the work group name the at sign (@), followed by your organization name.

   > **For example:**
   >
   > Enter `Loans@UPlus`.

4. Click **Create and open**.
5. On the **Work group** tab, in the **Settings** section, provide basic information about the work group:
   a. In the **Manager** field, enter the ID of a user who is a reporting manager for the group.
   b. In the **Default Work Queue** field, enter a default Work Queue to receive work coming into this work group.
   c. **Optional:** To add additional users who can view associated Work Queues, on the **Work group** tab, in the **Authorized managers** section, select the Manager ID.
   d. **Optional:** To add another user, click **Add item**.
6. Click **Save**.

Related tasks

- Creating a Work Queue

Updating the organizational structure by using the organizational chart

Reflect the growth of your company by creating new elements in your organizational structure. As your organization develops, you can add new divisions, units, and sub-units that correspond with new elements of your enterprise. As a result, you can reuse different application elements across your organization.

©2025 Pegasystems Inc

For clear visualization of the levels in your company, use an organizational chart to add new elements. The organizational chart displays each level with expandable sub-levels, such as a unit for a division. For example, when an Engineering division gains a new team, you can create a new unit that visualizes the new team as a whole.

In an organizational structure, the organization reflects your company. A division corresponds with a part of the company that gathers teams that operate in a similar business sector, such as engineering or human resources. A unit visualizes a single team. For further granularization, you can create sub-units. For example, if you need a unit to represent multiple teams that cooperate, a sub-unit can reflect those individual teams. Consider a scenario in which the UPlusTelco organization needs to create an Engineering division. The division includes a Mobile unit with two sub-units, Prepaid and Postpaid.

You can also reuse rules and other elements of your application across organizational layers. For example, reuse a process for calculating expenses for an entire organization and then specify different maximum amounts for each division.

1. Navigate to the organizational chart:

Choices	Actions
Navigate in Pega Express	a. In the navigation pane of App Studio, click **Users** > **Organization**. b. In the **Organization** workspace, identify the element that you want to edit. **TIP:** Expand the organization so that you can navigate to divisions and units.

©2025 Pegasystems Inc

Choices	Actions
	c. Click **Actions** next to the element that you want to edit.
Navigate in Dev Studio	a. In the header of Dev Studio, click **Configure** > **Org & Security** > **Organization** > **Organizational Chart**. b. Right-click an element that you want to edit.

2. Add a new organizational element:

Choices	Actions
Add an organization	a. In the list of available options, select **Add organization**. b. In the **Add Organization** dialog box, in the **Organization** field, enter the name of your organization. Use the organization's short name, ticker symbol, or Internet domain name. **For example:** Enter `UPlusTelco`. c. **Optional:** To provide more information about your organization, in the **Description** field, enter additional text. d. Click **OK**.

©2025 Pegasystems Inc

Choices	Actions
Add a division	a. In the list of available options, select Add division to organization. b. In the **Add Division** dialog box, in the Division field, enter the name of your division. **For example:** Enter `Engineering`. c. **Optional:** To provide more information about your division, in the Description field, enter additional text. d. Click OK.
Add a unit	a. In the list of available options, select Add unit to division. b. In the **Add Unit** dialog box, in the Unit name field, enter the name of your unit. **For example:** Enter `Mobile`. c. **Optional:** To provide more information about your unit, in the Description field, enter additional text. d. Click OK.
Add a sub-unit	a. In the list of available options, select Add sub-unit to unit.

©2025 Pegasystems Inc

Choices	Actions
	b. In the **Add Unit** dialog box, in the **Unit name** field, enter the name of your sub-unit. **For example:** Enter `Prepaid`. c. **Optional:** To provide more information about your sub-unit, in the **Description** field, enter additional text. d. Click **OK**.

For example:

The following figure shows a sample organization that consists of three divisions. The Administration division has two units, HR and Office, whereas the Engineering division has one unit, Mobile. The Mobile unit includes two sub-units, Prepaid and Postpaid.

©2025 Pegasystems Inc

Organizational chart in Dev Studio

What to do next:

Associate users with elements of your organization. For more information, see Adding an operator by using the organizational chart.

Learning about Access Groups

An Access Group is a group of permissions within an application. Pega Platform uses these permissions for operators, external system access, and Background Processes. You define an Access Group for operators who have similar responsibilities. For example, most applications allow Case managers to do actions that are different from the actions of regular operators, so Case managers and regular operators belong to different Access Groups.

Access Group names have the format *application name:Access Group name*. For example, for the MyApp application, you can define the MyApp:Administrators Access Group for administrators and the MyApp:Users Access Group for regular operators.

©2025 Pegasystems Inc

Operators can belong to multiple Access Groups. You select one of the Access Groups as the default, which is used when the operator initially logs in. If an operator belongs to multiple Access Groups, the operator can switch between groups. Only one Access Group is in effect at any given time during a session.

When you create an Access Group, you define permissions and settings that are used for operators who belong to that Access Group and who use the application defined for that Access Group. These permissions and settings include the following:

- Access Roles and privileges. For more information, see Managing Access Roles.
- The Portal layout.
- The work pools that are available.
- The types of work items that operators can work on.
- The Rulesets that are displayed at the top of the Ruleset list.
- Details of Rule caching for performance.
- For developers, the initially displayed Ruleset and version for Rules that they create.

Access Groups and Ruleset lists

When an operator logs in, Pega Platform looks for an Access Group in the following order until an Access Group is found, and uses that Access Group to assemble the operator's Ruleset list:

1. The default Access Group defined on the **Profile** tab of the **Operator ID** form.
2. The default Access Group for the Org Division that is identified on the **Work** tab of the **Operator ID** form.
3. The default Access Group for the Org that is identified on the **Work** tab of the **Operator ID** form.
4. The default Access Group for the appropriate requestor type.

©2025 Pegasystems Inc

Access Groups and external systems

An Access Group determines the Ruleset list that is available to an external system that requests services. The following data instances and Rules reference Access Groups directly, or indirectly by specifying an operator:

- Listener data instances
- Service package data instances
- Agent Rules
- Agent schedule data instances

When effective

When you save an Access Group, active requestor sessions on the current node that are associated with that Access Group are immediately updated. Requestors at other nodes in a cluster are updated when the next system pulse occurs on their nodes.

Facilities provided to unauthenticated (guest) requestors

Guest users, or unauthenticated requestors, typically have access to only the Rules in the Rulesets in the *PRPC:Unauthenticated* Access Group, as referenced in the requestor type instance named *pega Browser*.

> **CAUTION:** You should review **Security** tab of each activity in the Rulesets, to verify that the **Require authentication to run** checkbox is not selected if you update the:
>
> - *pega Browser* requestor type to reference a different Access Group, or
> - *PRPC:Unauthenticated* Access Group to make additional Rulesets available to unauthenticated users.

The Clipboard for a guest requestor does not include pages for the operator ID, organization, division, or organization unit.

©2025 Pegasystems Inc

- **Viewing Access Groups and operators**
- **Creating an Access Group**
- **Assigning work pools to an Access Group**
- **Granting Portal access to an Access Group**
- **Configuring tools access**

- Managing Access Roles
- Fields for operator contact information and application access
- Managing Requestor Type data instances

Viewing Access Groups and operators

Check relevant information about users of your application by exploring Access Groups and operator IDs. For example, by viewing an Access Group you can better understand what roles and permissions users have in your application, or by checking operator IDs you can view their role in your organization.

Access Groups define sets of permissions that you can assign to a group of operators, background processes, or external systems that interact with your system. For example, managers and case workers have different sets of actions to perform in your application, so you can assign them to different access groups. In a sample scenario, only a manager can approve and reject cases, so they need a separate set of permissions from case workers.

Operators define users and automated processes that perform work in your application. By viewing operators, you can better understand the structure of your organization and check who has access to specific applications in your system.

Navigate to the Access Groups or operator IDs depending on the scope that you want to check:

PEGA ©2025 Pegasystems Inc

Choices	Actions
View all Access Groups in your instance of Pega Platform	a. In the navigation pane of Dev Studio, click **Records**. b. Expand the **Security** category, and then click **Access Group**. c. Explore the list of available Access Groups. **TIP:** To find the Access Group faster, you can search the records by name, the time of the last update, and the author of the last update. The following figure shows a sample list of Access Groups: *Access Groups* d. **Optional:** To view the list of operators in an Access Group, click the Access Group, and then click the **Operators** tab. **Result:** The list includes full names, operator IDs, job positions, and the time of the last sign-in.

Instances of Access Group — Create — Refresh

Access Group Name*	Name	Updated on	Updated by
Mortgage:Users	MORTGAGE:USERS	October 6, 2023 9:14:34 AM EDT	
Mortgage:Managers	MORTGAGE:MANAGERS	October 6, 2023 9:14:33 AM EDT	
Mortgage:Authors	MORTGAGE:AUTHORS	October 6, 2023 9:14:33 AM EDT	
Mortgage:Administrators	MORTGAGE:ADMINISTRATORS	October 6, 2023 9:14:33 AM EDT	
VetApptS:Users	VETAPPTS:USERS	October 4, 2023 8:00:08 AM EDT	
VetApptS:Managers	VETAPPTS:MANAGERS	October 4, 2023 8:00:08 AM EDT	
VetApptS:Authors	VETAPPTS:AUTHORS	October 4, 2023 8:00:08 AM EDT	
VetApptS:Administrators	VETAPPTS:ADMINISTRATORS	October 4, 2023 8:00:08 AM EDT	

©2025 Pegasystems Inc

Choices	Actions
View only the Access Groups that reference the current application	a. In the header of Dev Studio,, click Configure > Application > Structure > Access Groups and Users. b. **Optional:** To view the operators in an Access Group, click the Access Group. **Result:** The following figure shows a list of operators in a given Access Group: *List of operators*
View the operators who are assigned to a particular organization, division, or unit	a. In the header of Dev Studio,, click Configure > Org & Security > Organization > Organizational Chart. b. Expand the node that you want to explore. c. Right-click the organization level, and then click **View Operators**, as shown in the following figure:

©2025 Pegasystems Inc

Choices	Actions
	 Viewing operators
View all operators in your instance of Pega Platform	a. In the navigation pane of Dev Studio, click **Records**. b. Expand the **Organization** category, and then click **Operator ID**. c. Explore the list of available operator IDs. **TIP:** To find the operator ID faster, you can search the records by full name, the time of the last update, the author of the last update, the work group, and the Access Group.

- Operators
- Fields for operator contact information and application access

©2025 Pegasystems Inc

Creating an Access Group

An Access Group is a group of application permissions that are used by an operator, external system, or Background Process. Create an Access Group to define the actions that are allowed when such an entity uses an application.

Before you begin:

To create an Access Group, you must have the *pzCanAlterRoles* privilege, which is included in the*PegaRULES:SecurityAdministrator* role.

1. In the Dev Studio header, click Create > **Security** > **Access Group**.
2. Enter a short description for this Access Group data instance.
3. In the **Access Group Name** field, enter a name. As a best practice, use the format of an application name followed by a single colon and a description of the job or user function of the users or other requestors who belong to this Access Group. Do not use an asterisk character (*), parentheses (), or spaces in an Access Group name. Begin the name with a letter and use only letters, digits, dash, ampersand, period, and colon characters.

   For example:

   ConsumerLoans:Supervisor

4. Click **Create and open**.

   Result:

   The Access Group is created in a Ruleset for the application that you are currently logged in to. You can change the Ruleset by clicking **Edit** next to the Ruleset name.

5. In the **Application** section, in the **Name** field, press the Down arrow key and select the name of the application that this Access Group can access.

©2025 Pegasystems Inc

6. In the **Application** section, in the **Version** field, press the Down arrow key and select the version of the application that this Access Group can access.
   When a user logs in, the Ruleset names and Ruleset versions identified in the application Rule are added to the user's Ruleset list.
7. Grant Portal access to the Access Group.
8. Add roles to the Access Group.

- Learning about Access Groups

Assigning work pools to an Access Group

Work pools are the Case Types in which users in an Access Group are allowed to create Cases. You specify the work pools that are available to an Access Group.

> **Before you begin:**
>
> Complete the following task before assigning work pools to an Access Group: Creating an Access Group.

1. Create an Access Group, or open an existing instance from the navigation pane by clicking **Records** > **Security** > **Access Group** and selecting an instance.
2. On the **Advanced** tab, in the **Name** field of the **Work Pools** section, list the work pools that are accessible to this Access Group.
   a. In the **Name** field, press the Down Arrow key and select a work pool.
   b. To add more work pools to the Access Group, click **Add item** and select a work pool.
   c. To select the user's default work pool, click the button next to the work pool name.
      As a best practice, select the work pool that contains the Case Types in which users that belong to this Access Group most often create Cases. This selection determines the default work pool that appears on the top of the Application Explorer tree when you open an application.

©2025 Pegasystems Inc

- **Work Pools and Access Groups**
- Adding Personas to organize users

Work Pools and Access Groups

Work Pools are the Case Types in which users in an Access Group are allowed to create Cases. The Work Pools that you add to an Access Group affect the user experience at run time.

- Work Pools are Classes that are marked as **is a Class Group** on the **General** tab of the Class Rule form. A Work Pool is a named collection of Case Types in which users that belong to the Access Group associated with the Work Pool are permitted to create Cases.
- When you define the Access Group, in the Access Group's Work Pool list, you can enter any Class Group for which the container Class (the *Rule-Obj-Class* instance with the same name as the Class Group) belongs to a Ruleset listed in the **Production Rulesets** list for the Access Group or that is available to users through application Rules. This restriction ensures that the Rules of the application are available to users associated with the Access Group.
- At run time, the Work Pools that you add to an Access Group appear in the Application menu **Switch Work Pool** list of the users who belong to the Access Group. Work Pool names appear in the order in which you list the Work Pools for the Access Group.

> **TIP:** If this list contains many Work Pools, consider reordering them to display the Work Pool names alphabetically or in another order that is meaningful to users.

- At run time, if you leave a Work Pool list empty and the Application Rule is not used for Work Pools, the system behaves in the following manner:

©2025 Pegasystems Inc

- When users that belong to the Access Group associated with the Work Pool open an application, the Application Explorer Class autocomplete field is empty, and no Classes appear in the tree.
- Users that belong to the Access Group associated with the Work Pool cannot create Cases. The Work Pool list does not affect which Cases the user can update. The list determines only the types of Cases that the user can create. The Assignments that the user can open and update are determined by Access Roles and Ruleset lists, not by Work Pools.

- Assigning work pools to an Access Group
- Learning about Access Groups
- Working with class groups

Granting Portal access to an Access Group

Associate a Portal with an Access Group to control which studios or web Channels are available to users while they work in your application.

1. Open an Access Group by searching for it or by using the Records Explorer.

   > **TIP:** To open your current Access Group, select the **Access Group** option from the **Profile** menu.

2. Click the **Definition** tab.
3. In the **Available Portals** section, click **+ Add Portal**.
4. In the **Name** field, press the Down arrow key, and then select a Portal.

   For example, select *pxPredictionStudio* to give the data scientists on your team access to Prediction Studio.

5. **Optional:** To make this Portal the default for all operators in the Access Group, select **Default** next to the Portal name.
6. Click **Save**.

©2025 Pegasystems Inc

7. Log off and then log in to Dev Studio.

Result:

Users who belong to the Access Group can switch from their current studio or web Channel to the Portal that you provide.

- Configuring Portals
- Changing your studio
- Learning about Access Groups

Configuring tools access

The Access Manager **Tools** tab provides information about the authorizations that users have to the tools you can secure in your application. Use Access Manager to configure the actions that Access Groups can do with tools.

You can view all Access Groups in the application and see a summary of authorizations for each Access Group.

You can also customize the tools that appear on the **Tools** tab. For more information, see Adding custom tools to Access Manager.

When you expand a tool category, icons indicate whether full access, no access, or conditional access is granted to members of the Access Group.

Summarized views can alert you to areas in your business processes that need tighter restrictions. You can expand the view to edit the authorizations of a role.

- **Viewing tools authorizations in all Access Groups**
- **Viewing tools authorization for a single Access Group**
- **Editing tools authorization for a single Access Group**
- **Viewing tools authorization for a single Access Group**

PEGA ©2025 Pegasystems Inc

- **Viewing tools authorizations in all Access Groups**
- **Editing tools authorization for a single Access Group**

- Access Manager
- Application tools

Viewing tools authorizations in all Access Groups

You cannot modify authorizations when viewing all Access Groups. The view shows you only whether:

> **Before you begin:**
>
> To use the tools tab in Access Manager, you must be logged in as an operator associated with an Access Group that contains the *PegaRULES:SecurityAdministrator* role.

- All operators in an Access Group are authorized to use all the tools listed in a tool category.
- No operators in an Access Group are authorized to use any tools listed in a tool category.
- A combination of conditional access and possibly no access to this tool among operators in the Access Group.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Access Manager** > **Tools**.
2. Click **Application**.
3. Select or deselect the applications you want to modify.
4. Click **Apply**.
5. In the **Access Group** field, select **All Access Groups**.
6. The page displays the Access Groups for the application and their access status for each tool. Expand a tool category to see the authorization for each tool in the category.

©2025 Pegasystems Inc

- Access Manager
- Application tools

Viewing tools authorization for a single Access Group

Access Manager allows you to view settings for Access Groups other than your own. You can see the following information:

> **Before you begin:**
>
> To use the tools tab in Access Manager, you must be logged in as an operator associated with an Access Group that contains the *PegaRULES:SecurityAdministrator* role.

- Operators in this Access Group have full access to the tool.
- Operators in this Access Group have no access to the tool. The menu item or UI element is disabled.
- A combination of conditional access and possibly no access to this tool among operators in the Access Group.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Access Manager** > **Tools**.
2. Click **Application**.
3. Select or deselect the applications you want to process.
4. Click **Apply**.
5. Select an Access Group from the **Access Group** list.

   > **Result:**
   >
   > The page displays the tools and their access status for the Access Group.

6. View or edit the tool category.
   a. Click an individual Access Group to edit.

©2025 Pegasystems Inc

b. Expand a tool category to see the authorization for each tool in the category.

- Access Manager
- Application tools

Editing tools authorization for a single Access Group

You can change the access status of a role for all the operators in the same Access Group.

> **Before you begin:**
>
> To use the tools tab in Access Manager, you must be logged in as an operator associated with an Access Group that contains the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure** > **Org & Security** > **Access Manager** > **Tools**.
2. Click **Application**.
3. Select or deselect applications that you want to process.
4. Click **Apply**.
5. Select an Access Group from the **Access Group** list.
6. Expand a tool category to see the authorization for each tool in the category.
7. In the column for the Access Role to be authorized, click the icon.
8. Select the access type.
   - **Full Access** – Grant operators with this role full access to the item.
   - **No Access** – Deny operators with this role access to the item. At run time, the system can hide or deactivate the item.
   - **Conditional** – Grant access based on an Access When condition. Select an Access When condition under which an operator in the Access Group can access the item.
9. Click **OK**.

©2025 Pegasystems Inc

- Access Manager
- Application tools

Designing applications for reuse and extension

Speed up the application development process by designing your applications with reuse and extension in mind. When you create reusable elements, such as Rules, classes, and components, you can implement them later in other projects that need the same kinds of data or a similar functionality.

By building a library of reusable elements you make your development process more efficient. You also improve the flexibility of your projects, because you can adapt them faster to changing needs.

Strategies to facilitate reuse in your development process

Relevant records

Relevant records are Rules marked for reuse, either manually or automatically. In App Studio, Rules are automatically marked as relevant whenever you create records in the

©2025 Pegasystems Inc

context of a Case Type or data type. In Dev Studio, you can manually mark Rules as relevant.

For more information, see Managing reusable assets in the Reuse Library.

For example:

If you mark a Service-Level Agreement (SLA) Rule as relevant in Dev Studio, that SLA becomes available in the drop-down menu for custom SLAs in App Studio.

Relevant records improve the choices available to developers in Pega Platform. As a result, application development becomes easier and faster.

For example:

Here are some examples of how relevant records speed up application development:

- Properties marked as relevant records create the Data Model for a Case Type.
- Sections marked as relevant records appear as reusable sections.
- Fields marked as relevant records are automatically added to the list of fields for which you can add a condition, and to the list of inherited fields of a data type.

For more information, see Relevant records for Rule reuse.

Components

A component is a small feature that you can add to any application in Pega Platform™. From a technical standpoint, a component is a collection of Rulesets that are packaged together as a ZIP file that you upload into the system. Using components can significantly reduce the effort involved in covering required functionalities.

PEGA ©2025 Pegasystems Inc

You can make your own components, or use the ones available on Pega Marketplace.

Configuration Sets

A Configuration Set is a group of settings that determine the run-time behavior of the application. A Configuration corresponds to a property in the background of the application. At run-time, when a user changes the value of that property, this triggers a change in the application behavior. In this way, Configuration Sets create an underlying structure that allows non-developers to control application behavior at run time in a low-code and user-friendly way.

> **For example:**
>
> In a financial application, you can create a Configuration that displays currency rates by default in all Cases of all Case Types. At run time, the application administrator can disable this Configuration to hide the currency rates from view.

Properties (fields)

Properties are data items that you can reference, such as a customer name or address. To call a property, use a dot notation: *.MyProperty*. You can reference properties with a single value (called value properties) or properties that store multiple values (called page properties, such as a *.Customer* property that contains the customer's name, address, and possibly other relevant information). Thanks to the referencing system, you can store information in one place and use it later as needed.

> **NOTE:** To help your designers and developers effectively reference and manage properties, follow the proper naming conventions. For more information, see Naming conventions for properties.

- **Managing reusable assets in the Reuse Library**

©2025 Pegasystems Inc

- **Relevant records for Rule reuse**
- **Components**
- **Creating a Configuration Set**
- **Referencing properties**
- **Application modularization, encapsulation, and naming for better reuse and maintenance**

Related concepts

- Troubleshooting tools and techniques

Managing reusable assets in the Reuse Library

Filter, review, add, and edit reusable assets from multiple categories in the Reuse Library in App Studio. By understanding which reusable assets your application includes, you can reduce the time and cost of your application development.

The Reuse Library lists assets available in your built-on applications stack and your current application. Each asset is displayed as a clickable tile with information about the layer that stores the asset. After adding the asset to your current application, you can reuse it in your current application layer. Consider a scenario in which the Onboarding application is built on the Candidate and Meeting applications. The Reuse Library in the Onboarding application displays the reusable assets from both the Candidate and Meeting applications. After adding the asset, the Reuse Library displays it as belonging to the Onboarding application, and the asset can be used when you develop the Onboarding application. The following figure shows the described example:

©2025 Pegasystems Inc

Data objects in the Reuse Library

The Reuse Library includes assets divided into the following sections:

- Case Types
- Data Objects
- Business Logic:
  - Correspondence
  - Data Transform
  - Decision table
  - Decision tree
  - Process
  - User Action
  - When Rule

You can reuse assets from the built-on stack of your application or enable components that you download from Pega Marketplace for use by your application. For more information about built-on application, see Adding built-on applications.

For more information about the Reuse Library, see Reuse Library - efficient reuse of assets.

1. In the navigation pane of App Studio, click **Library**.

©2025 Pegasystems Inc

Result:

The system displays each reusable asset in the form of a clickable tile, as shown in the following figure:

Reuse Library

2. In the **Library** section, specify the category of reusable assets that you want to manage by clicking the **Case Types** tab, the **Data Objects** tab, or the **Business Logic** tab.

Result:

By default, for the **Case Types** and **Data Objects** tabs, the list displays `All applications` as the search context. For the **Business Logic** tab, the **Context** list displays the label of the user's default work pool.

3. **Optional:** To find a specific asset, filter and search through the assets:

©2025 Pegasystems Inc

Choices	Actions
Case Types or Data Objects	a. In the drop-down list, select the context for your search. You can search on specific built-on layers or on `All application` layers, which is a default option, and displays assets from your current application and your application stack. After selecting an option, the system refreshes the list of available assets. b. In the search box, enter the term that you want to search for, and then press Enter. c. **Optional:** To clear the search, click the **Clear** icon, and then return to the complete listing of assets.
Business Logic	a. In the **Rule type** list, select the type of Rule that you want to find. b. In the **Context** list, select the context for your search. You can choose between **Case Types**, **Data Objects**, and **Reusable Layer** categories.

4. Review the available assets:
   - To see a summary of the asset, on the tile that you want to review, click Discover.
     The **Discover** window displays the **Name**, **ID**, **Class**, and **Ruleset** of the asset, as well as the Rule Type and location of the asset in the application stack, as shown in the following figure:

©2025 Pegasystems Inc

Viewing details of a reusable asset

- To open the asset in Dev Studio, click **Discover**, and then click **Open in** Dev Studio.

> **NOTE:** This option is only available if you have access to Dev Studio.

5. Manage the assets:

Choices	Actions
Add Case Types or Data Objects to the Library	a. Click the **Case Types** or **Data Objects** tab. Assets that come from built-on applications are not included in the Library by default. b. If the asset tile does not display the word Added, click **More** > **Add to application**.

©2025 Pegasystems Inc

Choices	Actions
	Result: The asset tile displays Added and the asset is available for reuse in your current application. You can modify the asset as needed within your current application. **NOTE:** You cannot remove assets from the Reuse Library.
Open Case Types or Data Objects	a. Click the Case Types or Data Objects tab. b. If the asset tile displays the word Added, either click the title of the tile, or click the More > Open. **Result:** The system opens the asset in App Studio.
Edit Business Logic in App Studio	a. On the Business Logic tab, on the asset tile, click the asset name. **Result:** A landing page with the authoring workspace opens. The authoring experience depends on the Rule type that you want to edit. b. Perform the required edits based on the Rule Type of the asset.

©2025 Pegasystems Inc

Choices	Actions
	NOTE: Editing an asset requires an unlocked Ruleset. For more information, see Defining the security of a Ruleset. c. Click Save. **Result:** In the background, the system selects the most appropriate Ruleset to save your changes.
Edit Business Logic with advanced configuration	a. On the **Business Logic** tab, on the asset tile, click the asset name. b. In the workspace for the asset, click **Open condition in** Dev Studio. **Result:** The system opens the asset in Dev Studio. c. After configuring the asset in Dev Studio, return to App Studio by clicking **Back to Pega** App Studio.
Manage components	a. In the header of the Reuse Library header, in the **Components** section, click Manage. For more information about adding components to your application, see Installing components.

Related tasks

- Creating a Data Transform in App Studio

©2025 Pegasystems Inc

- Sending automatic emails from Cases
- Creating decision tables
- Creating decision trees
- Defining conditions for a When Rule
- Creating a stand-alone Process

Relevant records for Rule reuse

Relevant records are Rules that Pega Platform automatically marks for reuse in App Studio, or that application developers manually designate for reuse in Dev Studio. By using relevant records, you configure your application with developer-approved Rules, improve its quality, and reduce development time.

Relevant records creation

Typically, Pega Platform automatically manages relevant records for you. In App Studio, when you create records, such as fields, Views, Processes, and user actions in the context of a Case Type or data type, Pega Platform automatically marks these records as relevant.

You can also configure a Rule in Dev Studio, then mark the Rule as a relevant record so that the Rule is accessible to other developers from prompts in App Studio.

For example, when you configure an Assignment and select an existing Service-Level Agreement (SLA) option in App Studio, the SLA is available in the drop-down list only if another developer first creates and marks the SLA as a relevant record in Dev Studio.

PEGA ©2025 Pegasystems Inc

SLAs as relevant records in App Studio

Rule Types that you can manually mark as relevant records in Dev Studio include:

- Properties, also known as fields
- Sections in traditional UI architecture

  In Constellation, you create Views directly in App Studio, which makes them relevant by default.

- Harnesses
- Paragraphs

©2025 Pegasystems Inc

- Correspondences
- Service-level Agreements
- Strategies
- Processes, also known as Flows
- User actions, also known as Flow Actions
- Data Transforms
- Decision Tables
- Notifications
- Pulse Feed Rules
- Validate Rules
- When Rules

> **NOTE:** You can mark records that are outside classes that Pega Platform provides by default, such as `@baseclass` or `Work-` classes.

You can manually designate records as relevant in the following ways:

Mark a Rule as a relevant record in the Rule form

You can mark a selected Rule as relevant directly in the Rule form in Dev Studio.

For more information, see Marking a record as relevant.

Add a Rule as a relevant record on the Relevant records tab

You can designate a selected Rule as a relevant record on the **Relevant records** tab in Dev Studio.

For more information, see Adding a relevant record to a specified class in your application.

You can access relevant records in App Studio, in the Data Designer or the Case Designer, when you add a Step to a Process, add fields to a user View, or apply a Service-Level Agreement.

©2025 Pegasystems Inc

For example, when you configure a user View for a Step in a Case Lifecycle, Pega Platform populates the Fields and Views lists with relevant records, as shown in the figure:

Relevant records in the Fields list

©2025 Pegasystems Inc

Relevant records in the Views list

The functions of relevant records

Relevant records control design-time prompting and filtering in the following areas:

In Case Types

Relevant records for a Case Type can include references to properties, Sections, Processes, and user actions that are explicitly important to your Case.

Properties marked as relevant define the Data Model of the Case. Processes and user actions marked as relevant appear in prompts for Case Type settings to encourage reuse. Sections marked as relevant appear as reusable Sections.

In data types

Relevant records designate the most important inherited fields for a data type. Relevant records can include fields that are defined for the class of the data type and fields inherited from parent classes.

©2025 Pegasystems Inc

In Condition Builders

When you build conditions in a Condition Builder in App Studio, or on the **Condition** tab of a When Rule in Dev Studio, you see a list of fields and When Conditions that you can use in your custom condition. The list populates with relevant records. If a record is not in the list, you can add it to relevant records in your application.

For more information, see Adding a relevant record to a specified class in your application and Defining Conditions in the Condition builder.

In proposition filters

When you use properties, strategies, and When Rules as proposition filter conditions, you designate these elements as relevant records for your primary context class, which by default is the *Customer* class.

For more information, see Proposition Filter Rules.

Creating new properties on the **Properties** tab of a strategy or a proposition automatically marks the properties as relevant records. You must manually add strategies and When Rules to the list of relevant records for a specific class.

For more information, see Managing relevant records.

Relevant records in Data Designer

The Data Designer displays properties for the selected data type that the system marks as relevant records. In Dev Studio and App Studio, use the filtering options to show reusable fields, which are relevant records defined elsewhere in the selected data type's inheritance path, and to show internal system fields.

The Dev Studio Data Designer provides the **Show inherited** and **Show relevant records** filtering options.

©2025 Pegasystems Inc

The Data Designer in App Studio displays only relevant records (properties) for the selected data type, with an option to display or hide relevant fields defined in inherited classes.

New fields that you add to the data types are automatically marked as relevant records as shown in the figure.

Data Designer with relevant records

Relevant records in Case Designer

You can access relevant records from several locations in Case Designer:

From the Data Model tab

The **Data Model** tab displays properties for the selected Case Type that the system marks as relevant. Use the filtering options to display reusable fields or system fields.

©2025 Pegasystems Inc

A Data Model tab with relevant records

From the Workflow tab

On the **Workflow** tab, the View configuration window for a selected Assignment displays records for the selected Case Type that the system marks as relevant. The **Fields** list displays fields (properties) that are configured on the current Case Type and marked as relevant records. The **Views** list displays Views that are configured on the current Case Type and marked as relevant records.

The system automatically marks new fields that you add to a form as relevant for that Case Type.

©2025 Pegasystems Inc

A Workflow tab showing relevant records

On the **Workflow** tab, the **More** section of the Step menu displays flows and Flow Actions that the system marks as relevant records from the Process and User actions lists, respectively.

The system automatically adds new user actions that you add to the Case to the relevant records for that Case Type as shown in the figure.

©2025 Pegasystems Inc

The flows and Flow Actions with relevant records

From the Views tab

The **Views** tab displays the Views (Sections) that the system marks as relevant records for the current Case Type.

The system automatically marks any additional Views that you create as relevant to the current Case Type, as shown in the figure.

©2025 Pegasystems Inc

The Views tab with relevant records

- **Managing relevant records**
- **Marking a record as relevant**
- **Adding a relevant record to a specified class in your application**
- **Marking relevant records as active or inactive**

Related information

- Configuring Relevant records - a demo

Managing relevant records

Speed up your application development by curating records that you need to include in your Case Types and data types. Relevant records control design-time prompting and filtering in several areas of App Studio, and as a result, reduce time-consuming searching through unrelated records in your Cases.

©2025 Pegasystems Inc

For example, in an insurance application, you can mark a value field of an insured object as a relevant record so that it is available to all child classes, such as a class that stores records for jewelry insurance or a vehicle insurance Case Type.

1. In the header of Dev Studio, click **Configure** > **Application** > **Inventory** > **Relevant Records**.
2. In the **Class Name** field, enter the class of a Case Type or data type for which you want to display relevant records.
   By default, the system displays only active relevant records for the selected class.

   > **Result:**
   >
   > The system displays a list of active relevant records for the specified class.

3. Do any of the following actions:
   - To display inactive records for the specified class, select the **Show inactive relevant records for the class** checkbox.
   - To display records that the current class inherits in the class hierarchy, select the **Show inherited relevant records for the class** checkbox.

     > **NOTE:** The **Mark relevant at** column shows at which class level the particular record is relevant.

   - To mark a record as active or inactive for the current class, click the **Actions** icon, and then select an option that meets your need. For example, select **Mark active for current class**.

Related concepts

- App Studio overview
- Rules management
- Designing the UI in Case Types

©2025 Pegasystems Inc

Related tasks

- Extending a Data Model for a Case

Marking a record as relevant

To save resources and speed up application development, promote reuse in data types and Case Types by marking a Rule as a relevant record. Relevant records control design-time filtering options in Case Types and data types in App Studio. As a result, when you create an application, you receive a set of relevant options, instead of an extensive or incomplete selection of choices.For example, in a banking application, you can mark a property that holds an interest rate as a relevant record for a loan Case Type, so that it is easily available in App Studio when you build your loan application form.

You can mark only certain Rule Types as relevant records. For more information, see Relevant records for Rule reuse.

You can mark records that are outside classes that Pega Platform provides by default, such as `@baseclass` or `Work-` classes.

Pega Platform automatically marks any records that you create in App Studio, such as properties, as records relevant to an appropriate context.

1. In the navigation pane of Dev Studio, click **Records**.
2. Open the Rule form of a record that you want to mark as relevant:
    - To open a property, click **Data Model** > **Property**.
    - To open a Data Transform, click **Data Model** > **Data Transform**.
    - To open a strategy, click **Decision** > **Strategy**.
    - To open a decision table, click **Decision** > **Decision Table**.
    - To open a When Rule, click **Decision** > **When**.
    - To open a Correspondence Rule, click **Process** > **Correspondence**.
    - To open a flow, click **Process** > **Flow**.
    - To open a flow action, click **Process** > **Flow Action**.

©2025 Pegasystems Inc

- To open a notification Rule, click **Process** > **Notification**.
- To open a Pulse feed Rule, click **Process** > **Pulse Feed**.
- To open a Validate Rule, click **Process** > **Validate**.
- To open an SLA, click **Process** > **Service-Level Agreement**.
- To open a harness, click **User Interface** > **Harness**.
- To open a paragraph, click **User Interface** > **Paragraph**.
- To open a Section, click **User Interface** > **Section**.

3. In the list of instances, open the Rule that you want to mark as a relevant record.
4. On the header of the Rule form, click **Actions** > **Mark as relevant record**.
5. Click **Submit**.

> **Result:**
>
> Your application marks the record as relevant for the `applies to` class of the record. The record is now available for reuse and configuration in your application.

6. **Optional:** To verify that the record is now relevant for the class, on the confirmation message, click **View**, and then review the relevant records landing page.

Related concepts

- App Studio overview
- Designing the UI in Case Types

Related tasks

- Installing components
- Creating Rules
- Extending a Data Model for a Case

©2025 Pegasystems Inc

Related information

- Configuring Relevant records - a demo

Adding a relevant record to a specified class in your application

Expand the library of relevant records by adding records and classes that are most likely to be reused for a Case or data type.For efficient management of resources, promote a Rule from a built-on application so that you can reuse the Rule in another class. By supplementing your inherited Rules with relevant Rules of your choice, you can create data types and Case Types in a quicker and more efficient way.

Before you begin:

For Pega Platform version 8.5 and earlier, ensure that you meet one of the following conditions to successfully add relevant records to a specific class:

- Ensure that the classes to which you add relevant records are in Rulesets that do not start with *Pega-* unless you have developer access.
- The class to which you want to add relevant records is one of the following: *PegaSample-Data-Product*, *PegaSample-Data-ContactInformation*, *Work-Channel-Triage-Email*, *Data-Portal*, *Data-*.

1. In the header of Dev Studio, click **Configure** > **Application** > **Inventory** > **Relevant Records**.
2. On the **Relevant Records** tab, in the **Class Name** field, enter the class of a Case Type or data type in which you plan to use the Rule.
3. Click **Add Record**.
4. In the **Add Records** dialog box, select the type and name of the Rule to reuse.

NOTE: For data-type classes, you can select only properties.

©2025 Pegasystems Inc

5. Click **Submit**.

Result:

Relevant records added for a selected class

The relevant Rule is ready for reuse in the list of relevant records that is available in the application.

Related information

- How to add a relevant record-a demo

©2025 Pegasystems Inc

Marking relevant records as active or inactive

Make a record available or unavailable in App Studio by marking the record as active or inactive. By marking certain records as inactive you can narrow down the list of relevant records to suit the current implementation context best, without the need to delete any records from your application.

1. In the header of Dev Studio, click **Configure** > **Application** > **Inventory** > **Relevant Records**.
2. On the **Relevant Records** tab, select the record of interest.
3. Click the **Actions** menu, and then define the availability of the record:
   - To make a record unavailable, select **Mark as inactive**.

The Relevant Records tab with the Mark as inactive option

   - To make a record available again, select **Mark as active**.
4. Click **Submit**.

Related information

- How to mark a relevant record as inactive - a demo
- Inactive Record Removal Tool

Components

A component is a collection of Rulesets that create a small feature that can be added to any application created in the Pega Platform.

©2025 Pegasystems Inc

Creating a component creates a new Ruleset. As a best practice, do all component-related development in this Ruleset to avoid naming conflicts. After you create a component, you can package the component and import it into another system. Then, you can enable the component in other applications on the target system.

Access the Components landing page by clicking **Configure** > **Application** > **Components**.

- **Installing components**
- **Enabling components**
- **Disabling components**
- **Creating a component**

- Creating a component
- Documenting a component
- Packaging a component
- Installing components
- Enabling components
- Disabling components

Installing components

Install a component to make a reusable feature available to applications on your system.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. On the **Definition** tab, in the **Enabled components** section, click **Manage components**.
3. Select **Install new** and select the ZIP file that includes the component.
4. **Optional:** Browse for components on the Marketplace site:

©2025 Pegasystems Inc

   a. Click **Browse apps & components** to open the Marketplace site to search for other components.
   b. Browse the available components. When you find a component to add to your application, follow the instructions on the component page to download the component ZIP file to your system.
   c. Return to your application, and then select the component ZIP file.
5. Click **Open**.
6. **Optional:** Select **Enabled** to enable the new component for this application. For more information, see Enabling components.
7. Click **OK**.

> **What to do next:**
>
> For more information about the component, review the component documentation in the component ZIP file.

- Adding a relevant record to a specified class in your application

Enabling components

Enable components to add the component feature to your application.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. On the **Definition** tab, in the **Enabled components** section, click **Manage components**.
3. Select the **Enabled** checkbox for each component that you want to use in your application.

> **NOTE:** Use caution when you add unlocked components to your application, as such components might still be in development. The

©2025 Pegasystems Inc

unlocked icon indicates that the component is unlocked, as shown in the following figure:

Unlocked icon

4. **Optional:** To change the component settings, click its name in the list, then make your changes and click Save.

For example:

You can unlock a locked component by clearing the **Locked** checkbox, as shown in the following example:

Unlocking a component

5. In the **Manage components** dialog box, click OK.
6. Click Save.

- Components
- Disabling components

©2025 Pegasystems Inc

- Installing components

Disabling components

Disable components to remove access to the component feature from your application.

1. In Dev Studio, open the application in which you plan to use components.
2. Click **Manage components**.
3. Clear **Enabled** for each component you want to remove from your application.
4. Click **OK**.
5. Click **Save**.

- Components

Creating a component

Create a component to support feature reuse in another system or application. You can control which Rulesets define your component and whether your component has dependencies on other Rulesets or applications

1. In the header of Dev Studio, click **Configure** > **Application** > **Components**.
2. Click **New component**.
3. In the **Label** field, enter a name for your component Ruleset.
4. In the **Version** field, enter a number that distinguishes this version of the component from previous versions.
5. Click **Create and open**.
6. Update the Component Rulesets. Change the default Ruleset or add Rulesets to the component.
7. List required Prerequisites for this component.
8. Click **Save**.

©2025 Pegasystems Inc

What to do next:

Document your component so that users can decide whether to install and reuse it in their applications.

- **Documenting a component**
- **Packaging a component**
- **Shared Ruleset**
- Components
- Adding a relevant record to a specified class in your application

Documenting a component

Document the components that you create to help users decide whether they can reuse your component in other applications.

1. In the header of Dev Studio, click **Configure** > **Application** > **Components**.
2. Select the component that you want to document.
3. Click the Documentation tab.
4. **Optional:** Upload an icon to associate with this component.

   NOTE: The icon file must be a .png, .jpg, or a .bmp file. The icon file size must be 3 MB or smaller. The icon must be exactly 120 pixels high and 120 pixels wide.

   a. In the Overview section, click **Upload icon**.
   b. Click **Choose File**.
   c. Select the icon file, and click **Open**.
   d. Click **Submit**.
5. Enter a **Description** of the component.

©2025 Pegasystems Inc

6. In the Key features section, enter a key feature of the component to help users understand what it does. Click **Add new** to add additional features.
7. Upload a PDF file with a maximum size of 25 MB that provides information about this component.
   a. **Optional:** Click **download template** to download a documentation template. Click **download example** to see a sample installation guide.
   b. Click **Upload**.
   c. Click **Choose File**.
   d. Select the file you want to add, and click **Open**.
   e. Click **Submit**.
8. Click **Save**.

- Components
- Packaging a component
- Creating a component

Packaging a component

You can use a component in applications on other systems. To import a component into another system, package the component.

To package a component, follow these steps:

1. Click **Configure** > **Application** > **Components**.
2. Select the component you want to package.
3. On the Definition tab, select Locked.
4. Enter and confirm the password. You will need this password to unlock the component.
5. Click **Submit**.
6. Click **Package Component**. The system creates a zip file.
7. Click **Download** to download the zip file.
8. Save the component .zip file. Do not change the default name of the file.

©2025 Pegasystems Inc

Shared Ruleset

A shared Ruleset contains a small number of Rules that each operate on a common, top-level page of a single class (or of subclasses of that class).

Shared Rulesets have the following restrictions:

- A shared Ruleset cannot contain a class Rule.
- Rules must apply to classes that extend existing classes such as the *Work-* base class and the *Data-* base class.

Typically, shared Rulesets contain only a few Rules. The intent is to provide a small group of Rules that provide common Functions that can be shared among (unrelated) applications.

To facilitate reuse of the Ruleset in multiple situations, use the keyword `Top` in property references in the Rules of a shared Ruleset to refer to the top-level class on which the Rules operate. Do not identify this class explicitly.

When you log in, a Ruleset list is developed from the information in your Access Group and Rules that it references. Shared Ruleset versions appear near the bottom of the list, just above the standard *Pega-* Rulesets.

To mark a Ruleset as a shared Ruleset , set the **Ruleset Type** on the **Category** tab of the Ruleset form to `Shared`.

To include a component Ruleset in your own Ruleset list, identify the Ruleset in the **Enabled components** array on the **Definition** tab of an application Rule referenced by your Access Group.

Creating a Configuration Set

Allow users of your application to change application settings at run time by creating Configuration Sets. As a result, users can adjust an application to their business needs

PEGA ©2025 Pegasystems Inc

in a no-code way. You also provide software that is easier to maintain and update, as Configuration Sets minimize the need for Rule overrides and custom Rule changes.

A common type of Configuration is an approval threshold, such as a dollar amount that you can use to control levels of approval. For example, in a loan request journey in which all loans over $50,000 require additional, VP-level approval, $50,000 is the approval threshold.

Consider a scenario in which every bank branch has a different approval thresholds, and some bank branches even require different dynamic ways of retrieving that value. For example, branch A has a threshold of $75,000, whereas branch B requires a threshold of $25,000. Additionally, the approval threshold for branch C is calculated based on the credit score of the loan applicant. Building a solution that catches all of the possible approval threshold values is very complicated. If you allow bank branches to change the references or the definition of that threshold, significant backward compatibility and updatability issues might cause bank branches to use old, outdated versions of the application. Calculating the threshold value with Configuration Sets eliminates the complexity of this traditionally challenging problem by providing a layer of abstraction. By creating a Configuration for the approval threshold and referencing it in the loan request journey, you allow each bank branch to provide their own value for that threshold without ever having to change the underlying application behavior or references.

Configuration Sets gather together settings called Configurations. By adding a Configuration, you create a property in the background that the application references. At run time, users can change the value of the property in a no-code way to provide flexible application behavior.

To provide flexible application that can be efficiently used by different groups of users, you can also define which Configuration Sets are available for specific Personas in your application. For example, only managers can have access to the Configuration Set that defines the approval thresholds for a loan.

1. In the navigation pane of App Studio, click **Settings** > **Configurations**.

©2025 Pegasystems Inc

2. In the header of the **Configurations** section, click Manage Configuration Sets.
3. In the **Configuration sets** dialog box, click Add.
4. In the Configuration set name field, enter a descriptive name for your Configuration Set, and then click OK.

   For example:

   For a Configuration Set to collect settings about displaying and calculating account policies, enter `Account policies`.

5. **Optional:** To add more Configuration Sets, repeat steps 3 through 4.
6. Close the dialog box by clicking Close modal.

For example:

The following video shows how to create the Account policies Configuration Set:

What to do next:

- Populate your Configuration Sets with Configurations that define application run-time behavior. For more information, see Creating Configurations.
- Define which Configuration Sets are available for specific groups of users at run time. For more information, see Configuring access options for a Persona.

- **Best practices for Configuration Sets**
- **Creating Configurations**
- **Adding Administration landing page to an application**
- **Technical considerations for Configuration Sets**

PEGA ©2025 Pegasystems Inc

- Referencing properties
- Designing applications for reuse and extension
- Building logic and calculating values in your application

Best practices for Configuration Sets

Configuration Sets help you build robust, flexible applications that are easier to maintain and update. Follow best practices to ensure optimal performance of your application Configuration.

Align Configuration Sets with the configurable features of your applications and not with Case Types

Think of features from the perspective of an end user. What is the logical grouping of business features within an application? Consider the example of Pulse, which has a variety of Configurations, ranging from types of messages to display options. These Configurations apply to many different Case Types and classes. Grouping the Configurations by Case Type can be confusing to end users. Grouping these Configurations into a single set called Pulse or Collaboration provides a clean, understandable grouping.

Provide a meaningful name and description for each Configuration in a set

The description is displayed in the end user **Administration** page and is critical for ensuring that end users understand the impact of changing a given Configuration.

The following figure shows sample Configurations with their descriptions:

PEGA ©2025 Pegasystems Inc

Descriptions of Configurations

Ensure that the setting type reflects the expected values of a Configuration

Setting type is used to validate the default values and end-user values. If you have a default value that is an integer but you expect text values from your end users, text is a better choice for the type than integer because text encompasses all expected values. It is also important to remember that settings originally defined as scalar default values can be updated to be calculated. Setting type is a critical piece of keeping backward compatibility.

The following figure shows a relation between a type of a Configuration and its output on the Administration landing page where users can change Configuration values:

©2025 Pegasystems Inc

Configuration type in App Studio and in the Administration landing page

Before deleting a Configuration, ensure that you consider its Ruleset version

Configuration Sets provide transparency into Configuration changes between versions by automatically tracking changes. You can delete a Configuration only if the Ruleset version in which the Configuration was created is unlocked. To ensure backward compatibility, other ways of deleting a Configuration actually result in deprecation. For all applications that allow clients to author Rules, remember that the clients might have custom references to the Configurations in your application. For any settings that you do not want to be leveraged outside of your application, provide the appropriate security. For more information, see Defining the security of a Ruleset and Deleting a Rule.

©2025 Pegasystems Inc

Creating Configurations

Provide no-code tools for run-time administrators of your application by creating Configurations. Configurations determine run-time behavior of your application that users can flexibly adjust to meet their unique business needs.

For example, in a financial application, you can create a Configuration that displays currency rates for every Case that Case workers process. When the business objective changes, an application administrator can disable the Configuration to hide the currency rates. For more organized management of your resources, you can combine Configurations into Configuration Sets.

> **Before you begin:**
>
> Create a Configuration Set to store your Configuration. For more information, see Creating a Configuration Set.

1. In the navigation pane of App Studio, click **Settings** > **Configurations**.
2. In the header of the **Configurations** section, click **Add Configuration**.
3. In the **Create Configuration** dialog box, in the **Name** field, enter a name for your Configuration.

   > **For example:**
   >
   > Enter `Currency rate`.

4. In the **Description** field, briefly describe the purpose of the Configuration.

   > **For example:**
   >
   > Enter `Display currency rates for dollar to euro.`

5. In the **Configuration Set** list, select the Configuration Set to store the Configuration.

©2025 Pegasystems Inc

6. In the **Configuration Scope** list, define elements to which the Configuration applies:
   - To apply the Configuration to an entire application, select **Application**.

     The Configuration can apply to assets in an entire application, such as Data Model, UI elements etc.

   - To apply the Configuration to a specific Case Type, select **Case Type**, and then select the Case Type that you want to use.

     The Configuration can apply to assets related to a selected Case Type, such as a Case Type Data Model or views in the Case Type.

     > **TIP:** You can also select all Case Types.

7. In the **Type** list, select a type for the Configuration.

   > **For example:**
   >
   > For a Configuration that either displays or hides currency rates, select **Boolean**.

8. If you create a picklist, in the **Add choices** section, click **Add choice**, and then, in the text field, enter a value that users can select at run time.
9. **Optional:** To add more choices for the picklist, repeat step 8.

   > **For example:**
   >
   > The following figure shows a Configuration that includes email addresses as choices in the picklist. At run time, users can select an email address from the list.

©2025 Pegasystems Inc

Picklist in a Configuration

10. In the **Default value** list, select a default value for the Configuration:

©2025 Pegasystems Inc

Choices	Actions
Select a specific value	a. Select Constant. b. In the value field, select or enter a default value for the Configuration. **For example:** To display currency rates in Cases for a Boolean-type Configuration, select True.
Select a specific value for each production level (only in the Application Configuration Scope)	a. Select Constant per production level. b. In the value field, enter a default value for the Configuration. c. **Optional:** To use a different value for any of the production levels, in the value field for the selected production level, enter a new value. **For example:** You can define different URLs for development and testing environments than for staging and production environments.

©2025 Pegasystems Inc

Choices	Actions
	 A Configuration Setting with different values for the staging and production levels than default
Calculate a value at run time from a decision table	a. Select Calculated value from decision table. b. In the list of decision tables, select a decision table that you want to use, or create a new decision table. For more information about decision tables, see Calculating values with decision tables in App Studio. **For example:** You can create a decision table that returns a loan limit for a customer based on the customer's income and account type.

©2025 Pegasystems Inc

Choices	Actions
	When you build a decision table, the system sources the columns for the table from the Data Model of the Scope that you select for the Configuration.
Calculate a value at run time from a decision tree	a. Select **Calculated value from decision tree**. b. In the list of decision trees, select a decision tree that you want to use, or create a new decision tree. For more information about decision trees, see Authoring decision trees in Configurations in App Studio. **For example:** You can create a decision tree that returns an expense approval limit for an employee based on the position of an employee in the company and duration of employment.

11. **Optional:** To provide a unique ID for the Configuration, expand the **Advanced** section, and then, in the **ID** field, enter a new value.
12. Click **Submit**.

©2025 Pegasystems Inc

For example:

The following figure shows a Configuration that, by default, displays currency rates for Cases of all Case Types within an application. At run time, an application administrator can disable the Configuration to hide the currency rates.

Currency rates Configuration

- **Authoring decision trees in Configurations in App Studio**
- Referencing properties
- Designing applications for reuse and extension
- Building logic and calculating values in your application

©2025 Pegasystems Inc

- Defining Conditions in the Condition builder
- Configuring Case-wide validation for correct data entry in fields

Authoring decision trees in Configurations in App Studio

Respond to complex business situations by calculating values in your Configurations automatically. With decision trees, you can use existing When Rules, fields, and Configurations to define conditions in which true comparisons can lead to additional comparisons, organized and displayed as a tree structure. Consequently, you deliver an application that helps users successfully resolve business processes in various, dynamically changing scenarios.

For example, you can create a condition that checks whether the location of a job candidate is equal to a specific city. If the condition is true, your application evaluates additional conditions, such as work experience and education.

> **Before you begin:**
>
> Create a Configuration for which you want to return values, and then configure the Configuration to use a decision tree. For more information, see Creating Configurations.

When you create a decision tree, you select one or more fields, When Rules, or Configurations that you want to compare against specific values, and then you define a method of comparison. Each branch of the tree is a condition that returns a value. At run time, the application evaluates the decision tree from the top level to the bottom.

1. Navigate to the Configuration in which you want to use a decision tree.
2. In the **Default value** list, select **Calculate a value at run time from a decision tree**.
3. Click **Configure decision tree**.

> **NOTE:** If the decision tree that you create in App Studio is edited in Dev Studio, you can no longer edit that tree in App Studio.

©2025 Pegasystems Inc

4. In the **Business logic** dialog box, click the Edit icon, and then enter a name for the decision tree.

For example:

Enter `Check candidate location`, as shown in the following figure:

Authoring a decision tree

5. In the first branch, in the **Value** field, enter the value that the branch returns when all conditions evaluate to true.

For example:

If you want your Configuration to evaluate whether to disable wrapping up a customer service call and you set the Configuration type to Boolean, select **True**.

6. In the list of values to evaluate, select the name of a field, When Rule, or Configuration.
7. In the comparator list, select a comparator.
8. In the value field, enter or select a value that your application compares with the field, When condition, or Configuration.

©2025 Pegasystems Inc

For example:

Enter a condition that checks whether the number of service Cases is greater than 0, as shown in the following figure:

Example condition in a decision tree

9. **Optional:** To define more conditions that apply to this decision tree branch, click the **Add a row** icon, and then repeat steps 6 through 8.
10. If you add multiple conditions, between the rows, select the **and** or **or** operator to define how to evaluate the conditions.

Result:

If you select **and**, the condition evaluates to true when all the rows evaluate to true. If you select **or**, the condition evaluates to true if at least one of the rows evaluates to true.

11. **Optional:** To change the order in which your application evaluates the conditions, define a new order by dragging the conditions.
12. In the **otherwise** section, in the **Value** field, enter the value that the decision tree returns when all conditions evaluate to false, as shown in the following example:

©2025 Pegasystems Inc

Setting the value that a decision tree returns when all conditions are false

13. Click **Submit**.

Result:

At run time, an application evaluates the conditions in sequence and returns the values that you configure. If an application evaluates all conditions as false, it returns the value that you provide in the **otherwise** section.

> **NOTE:** In applications that you build on Theme Cosmos, at run time, you can edit your existing decision trees on the Administration landing page. For more information, see Custom pages.

Related tasks

- Creating Configurations
- Creating decision trees

©2025 Pegasystems Inc

Adding Administration landing page to an application

To ensure that application administrators can edit values in Configurations at run time, add the Administration landing page to an end user Portal of your application. As a result, you provide no-code tools that users can apply at run time to adjust the application to dynamically changing business circumstances.For example, at run time, users can hide or display various UI elements in the application, such as currency rates or financial forecasts, without logging in to any design-time Portal.

Before you begin:

Create Configurations that define how your application works at run time. For more information, see Creating Configurations.

1. In Dev Studio, mark the *pxAdministration* harness as a relevant record.
   For more information, see Marking a record as relevant.
2. Add the Administration landing page to your application:
   - If you build a Constellation application, add the landing page to your Portal.

     For more information, see Organizing the main navigation for a standard work Portal.

   - If you build a standard Pega Platform application, add the *pxAdministration* harness to your application.

     For more information, see Harnesses.

For example:

The following video shows how to add the Administration landing page to a default Work Portal in a Constellation application, as well as the Administration landing page in the application run-time preview:

©2025 Pegasystems Inc

Result:

Your application includes the Administration landing page that users can edit at run time to change the application behavior. The following image shows a Configuration Set with editable fields:

Administration landing page

On the Administration landing page, users can also get a holistic view of all Configurations in the application, check default and current values for Configurations, and override Configuration values:

All Configuration Sets in an application

©2025 Pegasystems Inc

- Referencing properties
- Designing applications for reuse and extension
- Building logic and calculating values in your application

Technical considerations for Configuration Sets

Configuration Sets provide tools for run-time users to flexibly adjust application behavior in a no-code way. When you create Configuration Sets, considering crucial technical aspects can help you deliver an application that precisely targets business objectives. Additionally, Configuration Sets minimize custom Rule overrides, so that maintaining and upgrading your application is more convenient.

Rule status

When you create a Configuration, in the background the system creates a property whose value users can adjust at run time. You can also create a decision table to calculate the value automatically. When you move your application to the production environment, ensure that the Rules that include Configuration Sets and decision tables are final. As a result, you ensure that users cannot edit the Rules, and avoid future update issues caused by custom Rule overrides. For more information, see Setting Rule status and availability.

One Configuration Set per feature

For efficient and transparent organization of resources, ensure that one Configuration Set includes Configurations that define one feature. For example, in a financial application, you can create a Configuration Set that gathers Configurations that define the application UI, and another set for Configurations related to account policies and limits. At run time, an application displays each Configuration Set on a separate tab. By collecting related Configurations in one set, you help users quickly find the Configuration that they need to edit, as in the following example:

PEGA ©2025 Pegasystems Inc

Configuration Sets at run time

Impact of Access Groups on the run-time experience

An application user's Access Group defines the actions that the user can perform on Configurations at run time in the following ways:

- Users with the *ApplicationName:Administrators* Access Group can view, edit, and delete values for Configurations.
- Users with the *ApplicationName:Managers* Access Group can view and edit values for Configurations.
- Users with the *ApplicationName:Users* Access Group can view values for Configurations.

To ensure that users can perform additional actions without modifying an Access Group, you can grant access to Configuration Sets to Personas in App Studio. For more information, see Configuring access options for a Persona.

The user Access Group also impacts the editing of decision tables at run time. Only users with the *ApplicationName:Administrators* Access Group can view and edit decision tables in the Configuration Sets. For users with other Access Roles, you must first edit

©2025 Pegasystems Inc

their Access Groups by adding the *PegaRULES:ViewConfigTable* privilege to view decision tables, and the *PegaRULES:EditConfigTable* privilege to edit decision tables.

Related concepts

- Learning about Access Groups

Related tasks

- Calculating values with decision tables in App Studio

Referencing properties

Provide the data necessary to process your Cases by referencing information in the form of properties. When you refer to a property, your application calls a specific piece of information, such as a customer phone number or an address.

You can reference value properties that are strings of data such as text, number, or dates. You can also reference page properties that contain multiple value properties. For example, a page property `.Customer` might include value properties `.Name`, `.Address`, and `.Phone`.

- To refer to a property, prefix the property name with a period.

  > **For example:**
  >
  > To reference an order date, enter `.OrderDate`.

- To refer to a Single Value property, enter the property name.

  > **For example:**
  >
  > To reference a shipping date, enter `.ShippingDate`.

- To refer to an entry in a Value Group property, enter the Value Group name, and then the property name in parentheses.

©2025 Pegasystems Inc

> **For example:**
>
> To reference a mobile phone number from a Value Group property of phone numbers, enter `.Phone(Mobile)`.

- To refer to an entry in a Value List property, enter the Value List name, and then the index number of the entry in the list.

> **For example:**
>
> To reference the first entry in a list of discount codes, enter `.DiscountCode(1)`.

Apply the same guidelines when you reference Page properties.

- To refer to a Page property, enter the Page property name.

> **For example:**
>
> To reference a page with customer contact details, enter `.ContactDetails`.

- To refer to an entry in a Page Group property, enter the Page Group name, and then the entry in parentheses.

> **For example:**
>
> To reference a work address in a page group of addresses, enter `.Address(Work)`.

- To refer to a page in a Page List property, enter the Page List name, and then the page number in parentheses.

©2025 Pegasystems Inc

> **For example:**
>
> To reference the third page in the page list that contains ordered items, enter `.OrderedItems(3)`.

- To refer to a specific property on a page, enter the page name as a prefix for the property name.

> **For example:**
>
> To reference a city in the work address, enter `.Address(Work).City`.

- **Naming conventions for properties**
- **Configuring page, page group, and page list properties**
- Data Transforms

Naming conventions for properties

Creating descriptive and logical names for your properties can help you convey essential information just by looking at the name. As a result, you speed up development of your application, promote reuse across your application, and avoid duplicating properties that already exist in your system.For example, you can create *CustomerName* and *PhoneNumber* properties that clearly describe the type of data that the properties reference.

For the names of a property, choose nouns or noun phrases that clearly describe the property. Use names that are descriptive and meaningful, so that the contents of the property are easier to understand. For example, *LoanNumber* is a good choice for a numeric account number, while *LoanID* is a better choice for an alphanumeric account number. Enter the name in a camel case convention with the first letter in uppercase and then capitalize the first letter of each additional connected word.

For clarity, follow the following guidelines:

©2025 Pegasystems Inc

- End names for lists, that are Page List properties, with `list`.
- End names for groups, that are Page Group properties, with `group`.
- Because property names are case-sensitive, use only letters and digits.
- Do not start property names with an underscore (_) or dollar sign ($).

Standard property names

Standard property names that Pega Platform includes start with the `px`, `py`, and `pz` characters. Do not start names of your properties with these prefixes. The following table describes the purpose of each of the prefixes:

Prefix	Description
px	Computed properties that users see on a form, but cannot directly enter or change values, for example a *pxCreateDateTime* property.
py	Properties that users can explicitly enter or change, for example a *pyDescription* property.
pz	Properties that are reserved for internal use, for example a *pzInsKey* property. Users cannot see, enter, or change properties with the `pz` prefix.

Related concepts

- Naming conventions for records
- Naming conventions for classes
- Rules management
- Building logic and calculating values in your application

Configuring page, page group, and page list properties

To provide complete data for your Cases, define properties that store related information in the form of a page, page group, or page list. By providing the properties of a page type, you organize data in your application in a logical way, and as a result, improve application development and speed up Case resolution.

©2025 Pegasystems Inc

When you create single page and page list properties, you also define how your application sources data for the properties. To save time, you can configure your application to refer to a Data Page or to copy data from a Data Page while providing values for properties. For greater flexibility, you can also enable users to provide values manually.

1. In the header of Dev Studio, click **Create** > **Data Model** > **Property**.
2. In the **Label** field, enter a short description for the property.
3. **Optional:** To manually set the name key part of your record to a value that is different from the default, in the **Identifier** field, click Edit, and then update the name.
   The default value of this field is `To be determined`. The system automatically populates the field with a read-only value based on the sentence that you enter in the **Label** field. The system ignores spaces and special characters.
4. In the **Context** section, select the Application Layer in which you want to store the record.
5. In the **Apply to** field, select the class to which this record applies.
6. In the **Add to Ruleset** field, select the name and version number of a Ruleset to contain the record.
7. Click **Create and open**.
8. On the **General** tab of the property form, in the **Property type** section, click **Change**, and then define a property type:
   - To create a page, click **Single Page**.

     Single pages contain embedded pages as values. For example, you can create a single page that stores information about a user that creates a Case, such as a user ID and an email address.

   - To create a page group, click **Page Group**.

     Page group properties contain unordered groups of embedded pages. For example, you can create a page group that stores information about work

©2025 Pegasystems Inc

parties, and each embedded page stores information about one work party, such as a customer or a worker.

- To create a page list, click **Page List**.

  Page list properties contain ordered lists of embedded pages. For example, you can create a page list that store multiple addresses of a customer.

9. In the **Page definition** field, enter a page class.
   The page definition stores information about what is the class of each page in this list. When you provide the page definition, the system determines what fields each page has or what Rules the system can use on pages in this list.
10. If you configure a single page or page list property, in the **Data access** section, define how the property sources its values:

Choices	Actions
Users provide data manually at run time	Select **Manual**.
Property refers to a Data Page	a. Select **Refer to a Data Page**. b. **Optional:** If you configure a PageList, to retrieve each embedded page in the PageList separately, select the **Load each page in this page list individually** checkbox. c. In the **Data Page** field, enter a Data Page that stores values to refer.
Property copies data from a Data Page	a. Select **Copy data from a Data Page**. b. **Optional:** If you configure a PageList, to retrieve each embedded page in the PageList separately, select the **Load each page in this page list individually** checkbox.

©2025 Pegasystems Inc

Choices	Actions
	c. In the **Data Page** field, enter a Data Page that stores values to copy. d. **Optional:** To copy only selected information from the Data Page, in the **Optional Data Mapping** field, enter a Data Transform that determines what information the system copies.

11. Click **Save**.

Related tasks

- Configuring a Data Transform

Application modularization, encapsulation, and naming for better reuse and maintenance

Large, monolithic applications are costly to maintain and debug because significant portions of such applications must be duplicated and maintained in other parts with similar functions. On the other hand, modularization requires careful planning and design, but drastically reduces the cost of ownership in the long term.

To avoid the disadvantages of large, monolithic applications, you can identify the discrete functions that a feature requires and encapsulate the associated logic into appropriately named, granular modules that have clearly defined interfaces for input and output. These modules, or even modules made of smaller modules, can then be stitched together to create a neat, loosely coupled application.

This basic design choice has significant advantages:

©2025 Pegasystems Inc

Reusability

Because these modules follow a headless approach with well-defined interfaces, you can use them in more than one feature in your application, or even across applications.

Maintenance

Bugs and issues can be easily isolated and fixed within a module. As a result, if the input and output of a module is maintained, the impact analysis that is required and the number of regression issues that are introduced are drastically reduced. Additionally, fixes and enhancements do not need to be duplicated in other parts of the application.

Readability

The code is easier to read and follow because each module has only one function.

Testing

Instead of just testing the feature end-to-end, developers can focus on each module individually and run targeted tests for each module. Based on the size of the module, tests can be as elaborate as functional tests or as basic and strict as Rule-based automated unit tests.

Collaboration

Instead of building a large feature, developers can focus on smaller modules independently, and then connect them to create various features in your application. Promoting collaboration and fast-paced development results in shorter and more efficient development cycles.

Independent Releases

Because of modularization, you can release an application as soon as the basic application features are ready, and add more modules with only few changes to the existing modules.

Use case examples

You can implement modularization in an application at a level as fundamental as that of the activities and Data Transforms, or as abstract as the application, Rulesets, and Case Types. Consider the following two examples:

PEGA ©2025 Pegasystems Inc

- Consider an onboarding application that must connect to an external system to verify the customer data, and then proceed with or abandon the onboarding. You can build the interface for connecting with the external system directly within the application and its Rulesets. However, if an institution that uses an insurance application has a similar requirement, you must rewrite the whole interface for the new application.

  If you modularize the interface and build it in separate Rulesets or as components, you can then include these Rulesets or components in both the applications, as well as in other applications that you build. As a result, you can integrate with the external system without rewriting the interface.

- Consider a review Case where a customer record must be fetched from the database for a given customer ID. The data must then be evaluated to determine the customer risk, and appropriate due diligence questionnaires must be applied to the Case, based on the derived risk. Even though this use Case can be simply implemented by writing everything within one activity or Data Transform, each of these actions might be used in different contexts throughout the Lifecycle of the Case. For example, the risk evaluation can be performed on the customer data after the data is amended through the user interface to determine a change in the initial and final risk.

  This use case warrants modularization into three separate routines: a Data Page for fetching the customer record, and two Data Transforms: one for determining the customer risk and another for applying due diligence questionnaires. These routines can then be stitched together into a final Data Transform to fulfill the use case.

Initial considerations

Excessive modularization can increase code size and affect the performance of your application. Before deciding whether to modularize your code, consider the following key points:

©2025 Pegasystems Inc

- The modules must be reusable. Research any present or future use cases where a module can be applied.
- Not every function in your application must be a module. For example, setting up a property by copying an input parameter does not need to be a module.
- If it is a small one-off application that does not require maintenance or extension in the future, it might be better to avoid modularization in favor of faster implementation.
- Modularization can also be a good choice when you have a large team that must collaborate and build an application, provided that the application can be functionally and technically divided into smaller modules.

Best practices for modularization and encapsulation

Modularization options in Pega Platform™ are conceptually limitless and you can achieve every single Rule within Pega Platform at various levels and in various ways. Consider the following general guidelines when modularizing your code:

- When designing the architecture of your application, determine whether the features that you plan to build for your application are reusable. If they are, create separate applications, components, or Rulesets.
- Consider breaking down each routine into smaller modules that have well-defined input and output. A routine can be a Case Type, Stage, Process, Data Page, activity, When Rule, or any other Rule Type.
- Consider interoperability between modules. Each module must be self-sufficient, and communication with other modules should use only in and out parameters.
- Do not create modules for every small thing. Creating unnecessary modules is counterproductive, increases maintenance costs, and can also lead to performance issues.
- Ensure that the name of a module represents the function of the module. As a best practice, entering a brief description of what the module does in the usage field makes the code easy to locate, which promotes reuse.
- Educate your team on modularization and encourage them to reuse code rather than rewriting it.

PEGA ©2025 Pegasystems Inc

Benefits

Implementing modularization provides a greater degree of reuse and maintainability as the modules for each function are centralized and can be easily reused, tested, updated, or even replaced to propagate the changes to all the features that use these modules. Modularization helps you crush complexity by a large margin.

Related concepts

- Designing applications for reuse and extension
- Relevant records for Rule reuse
- Rules management

Building logic and calculating values in your application

Pega Platform™ offers several ways for you to shape application logic to best reflect your business scenarios. You can use simple or complex conditional structures, Validations, and Automations to fine-tune the application so that it flexibly responds to each set of conditions at run time.

In Pega Platform, building application logic means creating Rules that determine the paths that users can take in the application and Rules that determine the allowed input values. It also means automating calculations for some values as well as the decision-making based on the outcomes of those calculations or specific events in the application.

After the deployment of the application, you can delegate some of those Rules to certain users to manage in a low-code way. For example, a sales manager can edit a Decision Rule to change the threshold for an online order that qualifies for free shipping.

PEGA ©2025 Pegasystems Inc

Rules for handling application values

Values are entered into forms by users, stored in the application, and used to calculate other values, or evaluate conditions. For the application to run smoothly, you need to ensure that all the values that users provide are in the correct format and the acceptable range (data validation). For other values, such as tax rates on online orders, it is best to automate the calculations to avoid mistakes and speed up the process. Occasionally, it might be useful to convert values from one format and class to another.

Rules that help you handle values include:

Declare expressions
: Automatically calculate a value based on other values in the application. For example, you can calculate the tax rate for an online order based on the order value.

Declare OnChange Rules
: Call an activity to calculate a new value for a property if the value of another property changes. For example, you can calculate the total value of work done by a team in real time.

Data Transforms
: Convert data from one format and class to another.

Map Values
: Convert one or two input values, such as latitude and longitude numbers, into a calculated result value, such as a city name.

Constraints Rules
: Validate a property by comparing it to some other property or value.

Rules for handling application logic

Application logic is the set of Rules that determine what happens when a user performs an action or when an event occurs in the application. Those Rules are usually a condition of some sort. In Pega Platform, you can also build more complex conditional structures with multiple values to evaluate complex scenarios without losing track of all possible outcomes.

PEGA ©2025 Pegasystems Inc

Rules that help you shape application logic include:

Decision trees
: Evaluate simple conditional scenarios that consist of a series of yes/no conditions.

Decision tables
: Evaluate more complex conditional scenarios with multiple input values and multiple possible outputs. For example, calculating car insurance based on the age of the driver and the value of the car.

Declare Trigger Rules
: Call an activity when an event occurs. For example, when a customer changes their postal code, you can configure a Rule so that the system emails the relevant customer representative for the region.

When Condition Rules
: Evaluate a Boolean logic statement that compares field values to return true or false. You can apply When conditions many times in an application. You can group When conditions by using AND and OR operators to reflect complex business scenarios.

  For example, a hiring application verifies whether a candidate speaks French by using a When Rule. If they do, the application begins an additional process.

Activities
: Activities are custom Automations, used for automating processes that you cannot automate in a standard way.

> **NOTE:** As a best practice, use activities as a last resort only when no other method that Pega Platform provides can solve the issue.

An example of an activity is when an insurance company must submit insurance claims to the Registry of Motor Vehicles. You can configure an activity to automate claim uploads so that your application submits insurance claims outside peak hours without user intervention.

©2025 Pegasystems Inc

Additional options

Pega Platform offers some additional tools to help you structure application logic:

- Use the Condition Builder to help you with building conditions. You can also save custom conditions for future reuse with the Condition Builder.
- Create arithmetic and logical Expressions using natural language in the Expression Builder.
- As part of your application logic, customize URL patterns for work items such as Cases to facilitate collaboration and sharing.

Creating an activity

Automate a system task for which a more appropriate rule type is not available by creating an activity. With activities, you define a sequential set of instructions, or steps, that the activity completes automatically. Each step calls a method or supported rule type to perform the required processing.Consider a scenario in which an insurance company must submit insurance claims to the Registry of Motor Vehicles. To minimize the impact on users, you can configure an activity to automate claim uploads so that your application submits insurance claims outside of peak hours, without user intervention.

> **Before you begin:**
>
> As a best practice, before creating an activity, investigate model-driven alternatives that can deliver the desired results. Alternatives such as Data Transforms, collections, Declarative Rules, and features of Process and Case Management can remove the need to create new activities. If possible, use the standard activities that Pega Platform includes. To view a list of all standard activities, Processes, and Flow Actions that you can use in Case Management, in the header of Dev Studio, click **Configure** > **Case Management** > **Processes** > **APIs**. Create a custom activity only if the standard activities cannot fulfill your business needs.

©2025 Pegasystems Inc

1. In the header of Dev Studio, click **Create** > **Technical** > **Activity**.
2. In the **Activity Record Configuration** section, select **Activity (legacy)**.
3. In the **Label** field, enter a name that describes the purpose of the activity and helps identify the activity.
   Start activity names with a verb that indicates the purpose of the activity. Follow the verb with a noun or noun phrase that indicates an element on which the activity operates. Capitalize the first letter of each word in the name of the activity.

   > **For example:**
   >
   > Name your activity `CreateClassPage` or `SubmitClaimsLater` to clearly convey the purpose.

4. **Optional:** To change the identifier that other rules use to reference this activity, in the **Identifier** section, click **Edit**, and then enter a name that is unique within the **Apply to** class.
   Choose a name that starts with a letter and contains only letters, numbers, and hyphens. The name must be a valid Java identifier. The length of the class name plus the length of the identifier cannot exceed 128 characters.

   > **For example:**
   >
   > The following figure shows a sample activity with a changed identifier:
   >
   >
   >
   >
   > *Activity identifier*

5. **Optional:** To automatically set the activity type and, in some cases, add prepopulated steps to the form, click **View additional configuration options**, and then select one of the available activity templates:

©2025 Pegasystems Inc

- To create a route activity that you can use in an assignment shape to route an assignment to a worklist or work queue by using custom routing criteria, select **Template for Route type activity for worklist**. For more information, see Assignment shapes in Processes.
- To create a trigger activity that you can use in a Declare Trigger rule to set the values of parameters, select **Template for Trigger type activity**. For more information, see Creating Declare Trigger Rules.
- To create a utility activity that you can use to automate the processing of a case, select **Template for Utility type activity**. For more information, see Calling an activity or Automation from a Process.

6. In the **Context** section, define the context in which to execute the rule:
   a. In the list of built-on applications, select an application layer for the activity.
   b. In the **Apply to** field, enter a class that you want to associate with the activity. At run time, the activity runs in the context of a page. The class that you specify in the **Apply to** field must be either the class of that page or in the class hierarchy of that page's class. For more information, see Understanding class hierarchy and inheritance.

      The list of available class names depends on the application context that you select.

   c. In the **Add to ruleset** list, select a ruleset and a ruleset version in which you want to store the activity.
7. **Optional:** To override the default work item that your application associates with this development change, press the Down arrow key in the **Work item to associate** field, and then select a work item.
   For more information about your default work item, see Setting your current work item.
8. Click **Create and open**.

©2025 Pegasystems Inc

For example:

The following figure shows a Create Activity form. The activity has an identifier changed by adding a number and is a template for Route type activity for worklists. The **Context** section includes information about application and a ruleset that store the activity. The activity is assigned to a work item to track development work.

Creating an activity

What to do next:

Define a sequential set of instructions, or steps, that the activity completes automatically. For more information, see Configuring steps in an activity.

- Unit testing an activity
- Creating Rules

©2025 Pegasystems Inc

- Branches and Branch Rulesets
- Creating an Automation
- Setting Rule status and availability
- Deleting a Rule
- Methods

Configuring steps in an activity

After you create an activity, define a sequential set of instructions, or “steps,” that the activity completes automatically. Each step calls a method or supported Rule Type to perform the required processing.

Consider a scenario in which a customer service department wants to make available the content of their support requests through a REST service. To fulfill this requirement, you can configure an activity to validate the service parameters, gather the appropriate data, and construct a JSON response.

Before you begin:

- Create an activity. For more information, see Creating an activity. Determine whether and how other Rules can reference this activity by selecting an appropriate activity type on the **Security** tab. For more information, see Security tab on the Activity form.

NOTE: Limit each activity to 25 or fewer steps. To promote modularity and maintainability, use 15 or fewer steps and set the activity to complete one task.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Technical** category, and then click **Activity**.
3. Open the activity that you want to configure.

©2025 Pegasystems Inc

4. **Optional:** To temporarily exclude a step from running by commenting that step out, on the **Steps** tab, in the **Label** field, enter `//`.

   > **NOTE:** An excluded loop step excludes all child steps from running.

5. **Optional:** To allow other steps to reference this step in their When or Jump criteria, on the **Steps** tab, in the **Label** field, enter a short string.

   > **For example:**
   >
   > For a step that sets an error code, enter `error`.

6. **Optional:** To repeat a step or a contiguous sequence of steps a number of times, click **Loop**, and then define a loop.
   The most common way to loop is to iterate over a Page List property, such as *pxResults*, by using the **For each embedded page** option. For more information, see Repeating steps in an activity.

   > **For example:**
   >
   > Delete each embedded page by using the *Page-Remove* method.

   For more information, see Methods.
7. **Optional:** To complete a step only if specific parameters fulfill conditions that you define, click **When**, and then define the preconditions for the step.
   For more information, see Activity form - Completing the Steps tab - Entering preconditions.
8. Click the **Method** field, press the Down arrow key, and then select a method or instruction that you want to complete.

©2025 Pegasystems Inc

> **For example:**
>
> To retrieve a list of data instances that match a specific set of parameters, enter `Obj-Browse`. To set the value of a specified property, enter `Property-Set`.

9. **Optional:** To set a context for the step that is different than the Primary Page of the activity, specify the name of the page or property that you want to use as the context.
   By default, an activity runs within the context that calls that activity. For example, an activity that a Utility shape calls during Case processing runs within the context of *pyWorkPage*, which is the page that is assigned to the Case Type. Complete the following steps only if you want to change the default context:
   a. On the **Pages & Classes** tab, add the page or property that you want the system to use at run time.
      For more information, see Defining the pages and classes of a Rule.
   b. In the **Step Page** field, enter the name of the property that you want the step to use as context.
      Depending on the scenario, you can use additional syntax. For example, if you select the **For each embedded page** loop in step 6, enter a full property reference to a Page List property, such as *pagename.pxResults*.

> **For example:**
>
> To copy a specific order from a list of previous orders in memory, set the step page to the appropriate page in the page list of results.

10. Click **Expand to see method parameters**.
11. In the **Method Parameters** section, configure how your application applies the method or instruction that you selected in step 8:
    - For methods, enter parameters specific to a particular method.

      For example, if you selected the **Obj-Browse** method, in the **PageName** field, enter the name of the page on which you want to store the results, and then

©2025 Pegasystems Inc

in the **ObjClass** field, enter the name of the class that you want the method to browse.

- For instructions, such as **Call**, **Branch**, **Collect**, or **Queue**, enter parameters specific to the Rule that is associated with the instruction.
- If you selected **Java**, in the **Java Source** field, enter Java source code that you want the activity to run.

For more information, see Activity form - Completing the Steps tab - Entering Method parameters

> **Result:**
>
> At run time, the system passes the values that you specify to the method or instruction. The system holds the parameter names for a step on a special Clipboard Page that is called the parameter page, which is visible during tracing, but not with the Clipboard tool.

12. **Optional:** To specify conditions to evaluate after the method in a step runs but before the activity moves to the next step, click the **Jump** link, and then define a transition.
    For more information, see Defining step transitions in an activity.
13. Click **Save**.
14. If your activity contains Java steps, ensure that the activity causes no security issues by running the Rule Security Analyzer tool before locking a Ruleset version.
    For more information, see Implementing security guidelines for custom HTML.
15. **Optional:** To configure another step in the activity, click **Add a step**, and then repeat steps 4 through 14.

> **What to do next:**
>
> - Define input parameters for the activity. For more information, see Defining the input parameters of a Rule.

©2025 Pegasystems Inc

TIP: When you define the input parameters, select `In` or `Out` to indicate whether the parameter is used for input to the called activity, or output from the activity. Using input parameters can make activities easier to understand, call, and test.

CAUTION: The system does not prevent you from setting the value of an input parameter. A called activity can access and change both input and output parameters on the parameter page.

- Create scalar variables that you can use to pass information between the steps of an activity. For more information, see Defining the local variables for an activity.

- Creating an activity
- Unit testing an activity
- Creating Rules
- Branches and Branch Rulesets
- Methods

Activity form - Completing the Steps tab - Entering Method parameters

The Method Parameters section allows you to pass information from an activity to a method or instruction specified in the **Method** field of a step.

Completing the Method Parameters section in a step

1. Enter a method name or instruction in the **Method** field.
2. Click the **Expand** icon to expand the step.
3. Review the fields in the Method Parameters section. These fields dynamically update based on the value you specified in the **Method** field:

©2025 Pegasystems Inc

- For method names, parameters specific to the method appear.
- For instructions, parameters specific to the Rule name associated with the instruction appear. The keywords `Call`, `Branch`, `Collect`, `Queue`, `Flow-New`, or `Rule`, followed by a Rule name are examples of instructions.
- For a `Java` instruction, a text area appears where you can enter inline, Java source code.
- The message "No parameters" appears when the specified method or instruction does not accept any parameters.

4. **NOTE:** Enter values in each field, using the SmartPrompt to help you select an option from a list where available. Icons appear next to a field when appropriate:

   - **Star** – Indicates the field is required.
   - **Magnifying glass** – Opens the Rule referenced in the field.
   - **Build an Expression** – Launches the Expression Builder.

At run time, the system passes the values specified in the Method Parameters section to the method or instruction. If applicable, a parameter page is created for the step that is separate from the parameter page of the activity. This separate parameter page is destroyed when the step completes.

In some cases, you can use the parameter page of the activity to pass information instead of explicitly setting values in the Method Parameters section. This technique is referred to as passing parameters by reference.

Passing parameters by reference

1. Ensure that the parameter values required by the method or instruction are present on the parameter page of the activity before the step executes.

   This is typically done by a previous `Property-Set` step in the activity or a Rule higher up in the call stack.

©2025 Pegasystems Inc

2. Enter a method name or instruction in the **Method** field.
3. Click the **Expand** icon to expand the step.
4. Select the **Pass current parameter page** checkbox.

At run time, the parameter page of the current activity is passed to the method or instruction. This page returns to the activity when the step completes.

Passing parameters by reference allows the called method or instruction to modify the parameter page.

- Creating an activity
- Methods

Activity form - Completing the Steps tab - Entering preconditions

Use the **When** button to define a when precondition that conditionally skips execution of a step. The system evaluates whether specified conditions are met before it executes the method or instruction referenced in the step. If false, the rest of the step is skipped.

Click the button to access the details of the precondition.

Use the add row and delete row icons to define the condition. Order is significant. You can rearrange the order by selecting the row number and dragging it up or down. Click **OK** to save your edits and close the dialog.

Field	Description
Enable conditions before this action	Select to turn on the precondition test. If this box is not selected, all the rows are hidden and are ignored at run time. The gray **When** button appears.

©2025 Pegasystems Inc

Field	Description
When	Identify a When Condition Rule that specifies this precondition. Or enter an Expression starting with "=" that returns true or false, such as `=(.pyDoorisLocked == false).` Click the **Open** icon to review or update the When Condition Rule. Click to start the Expression Builder. `You can use the keyword <CURRENT>` within the precondition and transition elements of a step that also includes a repeat element to evaluate the current element of the repeat.
If True	Select to indicate what is to happen if the When Condition evaluates to True. Complete both an If True and an If False option for each precondition row. • `Continue Whens` — Advance to the next row of the precondition array, before executing the step. • `Skip Whens` — Skip all the later rows and execute the step. • `Jump To` — Jump to a specific higher-numbered activity step that contains a label. Entering the step's Label name in the True Param field. Processing resumes at the labeled step. You cannot jump to a label for a lower-numbered step. • `Skip step` — Skip the current step. For a repeating step, this ends the iteration. • `Exit This Iteration` — End the current iteration, and advance to the next iteration. • `End Activity` — End execution of this activity. The rest of the step is not executed.
True Param	If you selected `Jump To`, enter the Label name of the step to jump to.

©2025 Pegasystems Inc

Field	Description
	If you entered Exit Activity, enter an integer: `0` for a status of `Good`, any negative integer for a status of `Fail`, and any positive integer for a status of `Warn`.
If False	Choose what is to happen if the condition evaluates to false. The choices are the same as those for If True. As a good practice, complete both an If True and If False option for each row.
False Param	If you selected `JumpTo`, enter the label of the step to jump to.

- Creating an activity

Defining step transitions in an activity

Use the blue **Jump** button to open a pop-up dialog, and identify transitions, optional fields that can end iteration, terminate the activity, or cause control to jump to a later (higher-numbered) labeled step. Use transitions to specify conditions that are evaluated after the method in the step is performed, but before the execution continues with other steps.

Often a transition condition tests whether the method in the current step was successful. See How to test method results using a transition for more on this use of transitions.

Use the add row and delete row icons to define the condition. Order is significant. You can rearrange the order by selecting the row number and dragging it up or down. Click **OK** to save your edits and close the dialog.

Field	Description
Enable conditions after this action	Select to activate the transition. If this box is not selected, all rows of the transition are hidden and are ignored at run time. The gray **Jump** button is available.

©2025 Pegasystems Inc

Field	Description
On exception, Jump to a later step label:	Optional. If you anticipate that this method might cause a Java exception, enter the label of a higher-numbered step to execute next. This can prevent users from seeing any evidence of the exception, such as the red X. This setting does not apply if an exception occurs in a precondition to the step.
When	Identify a When Condition Rule that the transition is to evaluate. Alternatively, enter an Expression that returns true or false, starting with =. Click the **Open** icon to review or update the When Condition Rule. Click to start the Expression Builder. `You can use the keyword <CURRENT>` within the precondition and transition elements of a step that also includes a repeat element to evaluate the current element of the repeat.
If True	Select which action the system is to take when the When Condition or conditions evaluate to true. Complete both an If True and If False option for each row. • `Continue Whens` — Advance to the next row of the transition. • `Skip Whens` — Skip all the other, later rows of the transition.

©2025 Pegasystems Inc

Field	Description
	• `Jump to Later Step` — Jump to a higher numbered step that contains the specified label. Indicate which step to jump to by entering the step's Label name in the True Param field. After the activity jumps to the specified step, it processes that step and later steps in the activity. You cannot use `Jump to Later Step` to transfer control to a label of a lower-numbered step. • `Skip Step` — End processing of this step, continuing at the next step. If the current step involves an iteration, this choice operates similarly to a break statement in many programming languages. • `End Activity` — End execution of this activity. This is similar to the Exit-Activity method, not the Activity-End method.
True Param	If you selected `Jump to Later Step` for the If True field, enter the label of a higher-numbered step to jump to. If the If True value is `Jump to Later Step` and the transition

©2025 Pegasystems Inc

Field	Description
	is true, but this field is empty, a Java exception occurs.
If False	Select what action the system is to take if the condition evaluates to false. The choices are the same as those for If True . Complete both an If True and If False option for each row.
False Param	If you selected `Jump to Later Step` for the If False field, enter the label of a higher-numbered step to jump to. If the If False value is `Jump to Later Step` and the transition is false, but this field is empty, a Java exception occurs.

NOTE:

You cannot use an iteration to jump from outside an iteration sequence to any child step within the iteration. You can jump to the parent step — first step (NNN.0) of an iteration sequence. From a child step in an iteration, you can jump to a later step within the iteration or to a later step outside that iteration.

When a method in an activity step affects the value of a property referenced in a Declare Expression Rule, the declarative Rule execution occurs before the system evaluates the transition. For example, if the method updates the value of a Diameter property, and a Declare Expression Rule computes a CircleArea property based upon the Diameter value, only the updated CircleArea value is available to the transition.

When a step contains an enabled when-based transition, the Tracer shows a step status of "Good" regardless of the step status that existed before the transition evaluation. This status is consistent with the processing status that

©2025 Pegasystems Inc

> will be perceived by the next activity step; it reflects that any error condition that existed is "noted" by the activity.

Testing method results using a transition

When an application invokes an activity, the activity calls the methods referenced in each step (implemented in Java) sequentially. Most methods update the *pxMethodStatus* property on the `pxThread` page with a prefixed message key status.

The prefixed message starts with one of three values: `Good:`, `Warn:`, or `Fail:`. The rest of the message key is a refenced field value Rule. You can choose to have your activity place additional information about an error or result in the property named *pxMethodStatusInfo*. For more information, see TaskStatus-Set method.

As a best practice, test this status prefix against `Good` in those activity steps that may fail. In the When dialog, reference a When Condition Rule, and branch or jump to handle the failure.

In transitions, you can use three standard When Conditions: *@baseclass.StepStatusFail*, *@baseclass.StepStatusWarn*, and *@baseclass.StepStatusGood*. Note the following information:

- The `StepStatusFail` condition returns `True` when the status is `Fail`.
- The `StepStatusWarn` condition returns `True` when the status contains `Warn`.
- The `StepStatusGood` condition returns `True` when the status is `Good`.
- The Obj-Validate method does not usually update the *pxMethodStatus* property. Reference the standard When Rule *@baseclass.hasMessages* to test whether a page contains one or more page messages.

To enter a transition:

1. Select the **Jump** button on the step row to open the condition pop-up dialog.
2. Enter the second key part of a When Condition Rule.

©2025 Pegasystems Inc

3. In the **If True** field, indicate what processing will occur when the condition is true.
4. In the **If False** field, indicate the processing to occur when the condition is false.
5. Make sure the **Enable condition after this action** checkbox is selected, and click **OK** to save your edits.

> **NOTE:**
>
> - The value in the *pxMethodStatus* property changes often. The value displayed when you review the Clipboard using the Clipboard tool may be stale or deleted. Use the Tracer to see the current value.
> - An activity step can explicitly reset the value of *pxMethodStatus* to a less severe status by the Activity-Clear-Status method.
> - Using a transition in a step alters the Tracer display for the step. Normally, a red Fail row in Tracer results indicates an unhandled exception condition. If a method returns a Fail status but the step contains a transition, the Tracer row displays the status as `Good` and has a gray background. The transition mechanism is intended to allow you to catch previously unhandled Java `RuntimeException`s, not checked exceptions.

- Repeating steps in an activity

Repeating steps in an activity

Use the **Loop** button to identify iterations, optional fields that can define an enumeration of or loop condition for this step.

Loops repeat a step or a contiguous sequence of steps a number of times, or until a condition is met. You can iterate over the elements in an aggregate property — a `Value List`, `Value Group`, `Page List`, or `Page Group`.

©2025 Pegasystems Inc

This feature provides a limited form of enumeration or looping, similar to the "repeat until" or "do while" or "repeat from X to Y" structures of programming languages.

> **NOTE:** In a child step within a multistep loop, right-click to access a context menu. You can copy, cut, or paste child steps within the same iteration.

Repeating a single step using the loop field

Click the **Loop** button to access the details of the iteration. Use the pop-up dialog to indicate whether and how to repeat the step. You can cause the step to repeat a specific number of times, or until a specific result is reached. Click **OK** to save your edits and close the dialog.

Use the Repeat selection box to apply the step to more than one page or object. Select:

- `For each top level page`
- `For each embedded page`
- `For each element in value list`
- `For each element in value group`
- `For loop`

To exit the iteration before processing all elements in the iteration, complete a transition that is true after the last desired element is processed.

When processing the `For loop` or iterating over a `Value List` or `Page List` property, the parameter `param.pyForEachCount` holds the current iteration number. You can examine (but not alter) this value in the step. For example, to iterate only over the first ten pages or values, you can exit the iteration when `pyForEachCount` is greater than 10.

> **NOTE:** If the body of the iteration requires more than one step to code, create an activity and call the activity in the iterated step. Select the **Pass current**

©2025 Pegasystems Inc

> **parameter page?** checkbox to share the parameter page of the current activity with the called activity.

For each top level page

Select the `For each top level page` option to sequence through all pages of a specified class or classes and perform a method or instruction for each top level page. Leave the Step Page field blank. Each time the step repeats, the step's page changes.

For the optional Only Loop for Certain Classes parameter, enter a class or classes. (Click the Add Row icon to add more than one class.)

When Only Loop for Certain Classes is not blank, iteration processing skips over any page with a class that does not appear on the list, or is not derived from one of these classes.

Use the add row icon to add a class. You can rearrange the order by selecting the row number and moving it up or down.

For each embedded page

Use `For each embedded page` option to apply a method or instruction to each embedded page in a `Page List` or `Page Group` property. Use the iteration choice to specify processing that is to occur for each embedded page. For example, you can:

- Copy each embedded page
- Delete each embedded page using Page-Remove
- Delete a property on each embedded page
- Add a property to each embedded page

and so on.

©2025 Pegasystems Inc

Identify the property containing the embedded pages in the Target field. The SmartPrompt list for the Target field shows only `Page List` and `Page Group` properties.

For the optional Only Loop for Certain Classes parameter, enter a class or classes. (Click the Add Row icon o add more than one class.)

- When Valid Classes is blank, all pages of the `Page List` or `Page Group` are included in the iteration.
- When Only Loop for Certain Classes is not blank, iteration processing skips over any page with a class that does not appear on the list, or is not derived from one of these classes.

For each element in a value list

Select the `For each element in value list` option to repeat the step for each element in a `Value List` property.

Each time the step repeats, the system applies a method or instruction of your choice (such as Property-Set) to the current property. Use the `<CURRENT>` index value to refer to the current property element.

When you select this iteration form, a Value List Property field appears. Identify the `Value List` property in the field.

For each element in a value group

Select the `For each element in value group` option to repeat the step for each element in a `Value Group` property.

When you select this iteration form, a Value Group Property field appears. Identify the `Value Group` property in the field.

©2025 Pegasystems Inc

Each time the step repeats, the system applies a method or instruction of your choice to the current property. Use the `<CURRENT>` index value to refer to the current property.

For loop

Select the `For loop` option to repeat the step a number of times determined by the values of integer constants or integer properties.

Enter integer constant values or integer property references for the Minimum Value, Maximum Value, and Increment fields. The Increment must be a positive integer.

The system accesses these three values once before starting the iteration. The iteration may occur zero, one, or many times, repeating until the incremented value is greater than or equal to the stop value (unless a transition causes processing to end earlier).

Understanding `<CURRENT>`

The symbolic reference `<CURRENT>` can be used only in the activity step that defines the iteration. If that step calls another activity, the symbolic reference `<CURRENT>` cannot be used in the called activity; rather, aspects of the current iteration are available on the parameter page of the called activity:

- For a repeat type of `For each element in value list` or `For each element in value group`, the symbol `<CURRENT>` refers to the value of ClipboardProperty corresponding to the iteration. For the called activity, this corresponds to the `pyPropertyValue` parameter described below.
- For a repeat type of `For each embedded list`, `<CURRENT>` refers to the `ClipboardPage` of the current iteration: for example, when used in an RUF call that takes a `ClipboardPage` parameter or on the right side of a property set when the target is a page (when you add to a page list using `.list(<append>)`).
- For other repeat types, `<CURRENT>` refers to the index number of the current iteration, and corresponds to the `pyForEachCount` parameter.

©2025 Pegasystems Inc

- `<CURRENT>` can be used within the precondition and transition elements of a step that also includes a repeat element to evaluate the current element of the repeat.
- Within a repeat block, `<CURRENT>` can appear in the Expression Builder for the **Target** field of a Property-Set method; the keyword will refer to the current element's property so that it can receive the new value.

Multistep loops

The scope of a loop can be more than one step. The loop type and conditions are set in the first step of a multistep loop.

To create a multistep loop:

1. Identify the step that is to be first in the loop. Click **Loop** to access the pop-up dialog, and complete the loop details.
2. Complete information for the first step. Identify the step page. Optionally, you can leave the method blank.
3. Select the stage, open the right-click context menu, and select *Add Child*. The current step is renumbered as NNN.0, where NNN was the previous step number. The form changes to include a new later step NNN.1. The scope of the iteration is now both NNN.0 and NNN.1. This zero-numbered step is known as the parent step.
4. Complete NNN.1, a child step. Leave the Step Page field blank and the iteration criteria blank in most cases, to have the child steps operate on the step page identified in the parent step. Optionally, you can override this step page of a child step by entering one of the following:

- a top-level page (such as the Primary Page).
- a reference to a page relative to the step page of the parent.
- a reference to a `Page List` or `Page Group` property on the step page of the parent. (In this case, complete the `For each embedded page` iteration details for the child step.)

©2025 Pegasystems Inc

1. To create an iteration of more than two steps, click within the parent step NNN.0 again to make it active, and select the *Add Child* menu option gain.

Numbered child steps are executed in sequence NNN.0, NNN.1, NNN.2 ... then NNN.0 again, until the iteration ends or a transition causes a jump to a labeled step.

You can add addition child steps at the same level by placing the mouse pointer in the Description field and pressing the Enter key.

To convert an iteration sequence to comments, enter two slash characters // in the Label field of the parent step. The parent and all child steps become comments.

> **NOTE:** You cannot jump from outside a multistep iteration to any step within the iteration (except to the starting — parent — step). You can jump from any step in an iteration (parent or child) to a higher-numbered step (that is not within any iteration).

For an example of multistep and nested iterations, see the Pega Community article *How to use multistep loops and nested loops in activities*.

Nested iterations

By using the *Add Child* menu option on a child step numbered NNN.Z, you can create a nested iteration.

For example, clicking a child step numbered 14.8 converts this to 14.8.0, and inserts a second-level child step numbered 14.8.1. Complete the iteration conditions for step 14.8.0.

Use care in entering the Step Page for the parent step of a nested iteration (14.8.0 in this example). If you refer to an aggregate property (Value Group, Value List, Page Group or Page List), select a property that reside on the `<CURRENT>` step page of the parent's parent. (That is, the structure of the iteration must follow the structure of the

©2025 Pegasystems Inc

page). Within a nested step that does not have a repeat element, `<CURRENT>` inherits the semantics of its parent step that does have a repeat element.

> **NOTE:** From within a nested iteration, you can jump (using a transition) to a labeled step that is within an outer iteration but higher numbered. You can jump from 14.8.9 to 14.10 or 17, but not from 14.8.9 to 14.3.

Restrictions

If a step is part of an iteration sequence, the method cannot be any of the following:

Connect-Java	Page-Change-Class
Obj-Open	Page-Merge-Into
Obj-Open-By-Handle	Page-New
Obj-Sort	Page-Validate
	RDB-Open

Parameters available to called activity

The Pega Platform automatically adds the following parameters to the parameter page of a called activity with a step that contains an iteration.

- `pyForEachCount` — Index of the current iteration, normally starting at 1 and incrementing by one. (If the iteration type is `For loop`, the starting value may be a value other than 1 and the increment may be a value other than 1.) This parameter is available throughout the iteration as well as in called activities.
- `pyIterationType` — Indicates the iteration type, internally represented by one of the constants `page`, `embedded`, `propertylist`, `propertygroup`, or

©2025 Pegasystems Inc

`repeat`. (This parameter values are not available to other methods in an iteration step.)

- `pyIterationTarget` — Identifies the name or reference of the iteration. For page and repeat types, it is blank. For embedded, it is the step page reference. For `propertylist` or `propertygroup`, it is the property specified in the iteration. (This parameter values are not available to other methods in an iteration step.) (This parameter values are not available to other methods in an iteration step.)
- `pyPropertyValue` — Available only for `propertylist` and `propertygroup` iterations. The text value of the current property. (This parameter values are not available to other methods in an iteration step.)
- `pyPropertyReference` — Obsolete. Set to "???" for backward compatibility.
- `pyPropertyType` — Obsolete. Set to "???" for backward compatibility.

Calling an Automation from an activity

Call an Automation from an activity without using a Java step.

> **Before you begin:**
>
> Create an activity from which you want to call an Automation. For more information, see Creating an activity.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Technical** category, and then click **Activity**.
3. Click in the row of the activity from which you want to call an Automation.
4. On the **Steps** tab, in the **Method** field of a step in the activity, select **Call-Automation**.
5. Click the arrow to the left of the **Method** field to expand the step and view the method parameters.
6. In the **Resource** field, select the type of Automation that you want to call.

©2025 Pegasystems Inc

7. In the **Automation** field, select the Automation that you want to call.
8. Complete the **Map from** and **Map to** fields for the inputs and outputs for the Automation you are using.
   This establishes the relationship between the resource inputs settings, and the resource outputs generated Automation errors. You can view a description of these fields by viewing the Automation Rule. For more information, see Viewing automations.

   You do not have to use all of the outputs. To use an output, you must provide a property reference that can hold that type of output, for example, a string property for a string output. Like inputs, outputs are also validated. Errors that occur when the Automation runs are returned in the `Automation Errors` output. If an error occurs that is not in the Automation Rule, it is converted to a run-time error before being returned.

   > **NOTE:** Page type outputs expect a property type string. A page reference is stored in that property so that you can return to that page later

9. If you specify a default value in the Automation, in the **Inputs** section, a **Use default value** option is displayed next to the corresponding **Map from** field. You can use the default value or specify a new default value:
   - To use the default value specified in the Automation, click **Use default value**. The value in the **Map from** field becomes read-only, and the **Use default value** option changes to **Specify value**.
   - To specify a new default value, in the field for which you want to change the value, press the Down arrow key to select a value.

   > **NOTE:** If a list of values is specified for a text input in the Automation Rule, only those values are available in the **Map from** field.

©2025 Pegasystems Inc

10. Click **Save**.

- Call-Automation method
- Automations

Best practices for writing activities for background jobs

Ensure that work items are processed correctly and that data is not corrupted during a node shutdown by using best practices for writing activities for background jobs.

Queue processors, job schedulers, and Rule-type agents are background jobs that use an activity to implement processing logic. Use these processes appropriately to complete activities; use job schedulers for looped activities and queue processors for activities that can be processed in less than one second. For more information, see Background processing best practices. When you have to process very large amounts of data that cause the background processes to take a long time to complete, you can avoid processing errors by designing your activities appropriately. Review the following recommended practices to ensure that your activities can respond to a node shutdown in the system infrastructure.

Create small work items

When you design an activity, break down large work items into the smallest work items you can and then process them individually using a queue processor. For more information, see Managing queue processors. When you optimize an existing activity, use a queue processor instead of Rule-type agents because queue processors provide better performance, resiliency, fault tolerance, and flexibility than Rule-type agents.

Use checkpoints

For activities that take a long time to run, such as an overnight job that handles many records, design the activity to be resumable. This ensures that activity logic can gracefully handle restarts without corrupting data or introducing inconsistencies into

PEGA ©2025 Pegasystems Inc

your application. Resumable activities use checkpointing: Pega Platform saves the state of jobs in persistent storage or as messages in a queue, as appropriate to the use case.

For example, you can create a resumable activity where Pega Platform assigns activity stage process statuses to work objects: NEW, IN-PROGRESS, and PROCESSED. Your job scheduler reads the work object records that are to be processed, loops through them one by one, marks the status of the current work object as IN-PROGRESS, continues with the business logic, and when the processing is finished, changes the status to PROCESSED. In this use case, if a node restart occurs in the middle of the job, the new node can take up the activity assignment and see where the last object was PROCESSED, so that it starts on the appropriate object in the queue, rather than starting the activity from the beginning and needlessly reprocessing work objects.

Divide and scale tasks

If you have background tasks that you can break down into multiple tasks that have different performance capabilities, consider dividing the tasks and then adding them appropriately to job schedulers or queue processors, so that you can scale the tasks independently. In this way, you have a job scheduler that runs steps in loops, while within the loop, a queue processor does some of the work. However, the two background processors still work in tandem to complete the task.

Create sub-activities

Split activities into sub-activities when possible so that the state is saved frequently. This ensures that if there is an abrupt node shutdown, the sub-activities can proceed based on the recovered state.

For more information about agents, job schedulers, and queue processors, see the Admin Studio help.

Recommended practices for long-running activities

When you plan to use a background process for activities such as overnight batch processing of records, keep these recommended practices in mind:

PEGA ©2025 Pegasystems Inc

- Design the activity so it is simple, resumable, and so that you can assign it to a job scheduler to perform the steps in a loop. This ensures that the activity can handle background processing node restarts and resume from where it left off. Design this resumable activity following these guidelines:
    - The activity operates in batch mode.
    - Select the **OnlyIfNew** checkbox of the *obj-save* that performs an insert operation instead of an update.
    - If the activity reads bulk data from a database, use `rownum`, `offset`, or another status to track processed records.
    - The activity output generates file data of a minimal size so that the processing and database call stays under 60 seconds. This ensures that your process does not create out-of-memory or performance issues.
    - For bulk data processing, use a batch database interaction so that it is resumable.
    - Be aware that the system does not check for existing items before insertion, which can lead to duplicate key exceptions. Implement appropriate logic to handle this.
- If your application is running in a Pega Cloud® 3 environment, design the activity to complete in less that 300 seconds (5 minutes). Pega Cloud 3 systems wait five minutes for background processing to complete before the shutdown occurs. An activity that finishes within this time frame ensures that the job scheduler can restart the activity gracefully, without errors. Implement the suggestions above in your activity design.
- Review the activity for multiple dependencies on other Rules, such as calling another activity or Data Transform from this activity, or making a call to an external service. Your activity design must be able to robustly handle edge cases such as exceptions or timeouts, so that it can independently ignore these scenarios and become stuck and drain processing resources.

PEGA ©2025 Pegasystems Inc

Example of optimizing a resumable job scheduler activity

The following is an example of a resumable Job Scheduler activity. It can run nightly on a single background node and processes a set of work objects from a workbasket in three steps, as shown in the following figure:

Activity that processes a set of work objects from a workbasket

The *D_WorkBasket* data page only returns work items in the inprogress@pega.com workbasket when run. As the flow processes work items, they move out of the workbasket to the next stage of processing. If the node running this Job Scheduler activity shuts down for some reason, such as a health check failure, and the processing only gets through 60 of 100 items, the system will recognize that the processing was not complete and another background node will restart the activity. However, when the other node begins to run the activity, the *D_WorkBasket* data page result will now only contain the last 40 records.

While the Job Scheduler activity can process many work objects over a long period and is responsive to unplanned restarts, the process is single-threaded. As the following table shows, the more work items it processes and the longer the average processing time (APT) a single work item takes, the time it takes to process large batches of work items becomes very long:

# Work Items	.500 Seconds APT	.250 Second APT	.125 Second APT
100	50	25	12.5
1000	500	250	125
10000	5000 (83 min)	2500 (42 min)	1250 (21 min)
100000	50000 (13 hours)	25000 (6 hours)	12500 (3 hours)

©2025 Pegasystems Inc

You can optimize this example of a resumable Job Scheduler activity to reduce the amount of time that it takes to process a large volume of work objects. You can greatly reduce the processing time for this activity and improve its fault tolerance by replacing the final, *Call ResumeFlow* step in the job scheduler that references the activity, to instead use the default Pega-provided queue processor, *Queue-For-Processing* method. Making this change will quickly reload the data page asynchronously and handle work object processing across multiple threads per background processing node, as shown in the following image:

Reload the data page using the Queue-For-Processing method to optimize a long-running activity

Related concepts

- Managing background processing resources

Related tasks

- Creating an activity

©2025 Pegasystems Inc

Related reference

- Activity form - how to create activities for Processes

Security tab on the Activity form

Use the **Security** tab to specify the activity type and optionally to restrict which users (or other requestors) can execute the activity. This optional security supplants restrictions based on Ruleset and version.

You can specify zero, one, or more than one privilege to restrict access. Order is not significant. At run time, any match between a privilege listed and those a user possesses allows users to execute this Rule.

Field	Description
Restrict access	
Allow invocation from browser	Select to allow users to start this activity directly through user input processing, for example through a Submit button or a `pyActivity=` element in an URL. Clear this if this activity is to be started only from another activity, through a Call, Branch, or other means. For example, select the box for a service activity, or if this activity is called by an AJAX event from a form. At run time, if the box is not selected and a user attempts to start this activity by user input, the activity does not run and returns a method status of `Fail:Security`.

©2025 Pegasystems Inc

Field	Description
	For most activities, leave this box cleared to promote security of your application. Unless needed by your design, allowing activities to be started from a URL or other user input — whether the requestor is authenticated or a guest — may let users bypass important checking, security, or setup.
Require authentication to run	Select to require that only authenticated requestors can start this activity. Clear to allow guest users to run this activity, if they meet other security and access criteria. Guest users — unauthenticated requestors — typically have access to Rules in the Rulesets provided in the *PRPC:Unauthenticated* Access Group, as referenced in the Requestor type instance named *pega.BROWSER*. **CAUTION:** If you update the *BROWSER* requestor type to reference a different Access Group, or update the *PegaRULES:Unauthenticated* Access Group to make additional Rulesets available to unauthenticated users, review carefully this checkbox for each activity in the Rulesets. Select this checkbox for all but those

©2025 Pegasystems Inc

Field	Description
	specific activities that guests need to run. In most cases, clear this checkbox if the activity is for an agent. Agents are not true authenticated users and by default cannot run activities that are restricted to authenticated users. However, this checkbox is ignored by agents for which the Bypass activity authentication checkbox (on the Security tab is checked; they can run activities regardless of the Authenticate? value. Identify privileges in this array to restrict which users and other requestors can execute this activity. At run time, if the user does not possess an Access Role that — through an Access of Role to Object Rule — provides access to one of the identified privileges, the execution of the activity fails.
Privilege Class	Optional. Identify the Applies To key part of a class to use at run time to locate a privilege Rule. Normally this is the same as the Applies To key part of this activity.
Privilege Name	Optional. Identify the name for a privilege in that class (or an ancestor class). The class you enter and the name must together identify a privilege (using Rule resolution including class inheritance.)

©2025 Pegasystems Inc

Field	Description
Usage	
Activity Type	Determine whether and how this activity can be referenced in other Rules. For an activity that is not to be referenced in a flow, choose one of these values: **Activity**: Select when no more specific value is applicable. Activities with this value cannot be referenced directly in flows. (Select `Activity` for this field on Parse Structured forms.) **Asynchronous**: Select for an activity that runs in a background thread asynchronously. The activity can call only these step methods: Load-DataPage, Connect-Wait, and Call-Async-Activity (step

©2025 Pegasystems Inc

Field	Description	
		instruction to allow user to load an asynchronous activity)
	LoadDeclarativePage	Select for an activity that adds values to Data Pages. Reference this activity on the Definition tab of a Data Page Rule (*Rule-Declare-Pages* Rule Type).
	Locate	The Locate activity type is not supported. Existing activities that use locatable pages display this option. New activities do not.
	OnChange	Select for an activity to be executed automatically by a Declare OnChange Rule, Such activities must not use any

©2025 Pegasystems Inc

Field	Description	
		methods that directly change the properties or the database. Declare Expression Rules do not evaluate during the execution of an OnChange.activity. OnChange activities must not perform any forward chaining themselves.
	Trigger	Select for an activity to be executed automatically by a Declare Trigger Rule. Because triggered activities run during database commits, they cannot themselves commit database transactions.

©2025 Pegasystems Inc

Field	Description
	Select `Notify`, `Rule Connect`, `Assign`, `Route`, or `Utility` if the class of the activity inherits from the Work- base class and you designed and implemented the activity to be referenced in a flow shape. See How to create activities for use in flows for more details on these choices: Choose `Notify` for a notification activity. For example, a notify activity can send an email message to someone conveying news of the Assignment. Choose `Assign` for an Assignment activity, one that creates an Assignment. Choose `Route` for a router activity, one that determines which user Worklist or Work Queue is to receive an Assignment. Choose `Utility` for a utility activity, one that automates processing without user interaction. Choose `Rule Connect` for an activity that operates without user interaction and calls a connector Rule to interface with an external system. Do not choose `Validate` or `Assembler` for this field.

- Declare OnChange Rules

©2025 Pegasystems Inc

- Declare Trigger Rules
- Activity form - how to create activities for Processes

Defining the local variables for an activity

Create scalar variables that you can use to pass information between the steps of an activity. Local variables require less memory and run faster than regular parameters.

For example, you can simplify a highly nested Expression by using local variables to break up that Expression into component parts. As a result, the Expression is easier to read and troubleshoot.

Your application stores local variables together as fields in the Java class that Pega Platform generates to implement the activity. Local variables do not appear on the parameter page.

You can reference a local variable in a Property-Set method by using the `Local` keyword, as well as in Java steps.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Technical** category, and then click **Activity**.
3. Open the activity that you want to configure.
4. Click the **Parameters** tab.
5. In the **Local variables** section, in the **Name** field, enter a unique identifier for the local variable.
   You can choose any valid Java identifier. Do not choose a name that starts with pz or any of the reserved names.
6. In the **Description** field, enter text that describes how your Rule logic processes the local variable, so that other developers can understand that Rule more quickly.
7. In the **Data type** list, select the type of data that the value for the variable has:
   - For a `Java.lang.string` object, select **String**.
   - For a Java `StringBuffer` object, select **StringBuffer**.
   - For a Java integer, select **Integer**.

©2025 Pegasystems Inc

- For a Java double, select **Double**.
- For a Java Boolean, select **Boolean**.
- For a Java char, select **Char**.
- For the `BigDecimal` class, select **BigDecimal**.
- For any Java object, select **Object**.

> **NOTE:** An application can assemble a local variable of the StringBuffer type in multiple steps, because a Property-Set method on a string buffer class appends the value to the string buffer.

> **NOTE:** To avoid Java null pointer exceptions, the system initializes `String` local variables to the empty string, and the `StringBuffer` local variable as: `public StringBuffer aBuffer = new StringBuffer();`

8. **Optional:** To define another local variable, click **Add item**, and then repeat steps 5 through 7.
9. **Optional:** To use a flow action at run time that produces a prompt form in which the user can enter values for the flow parameters that are specified in this tab, in the **Local action for parameter display** section, specify the flow action.
   For more information, see Prompting users for parameter values.

Activity form - how to create activities for Processes

Use these instructions to create activities that can be called directly in Processes.

Flow processing automatically controls locking and transaction boundaries for work items. Do not use the *Commit* method (or the *Obj-Save* method with the **WriteNow** parameter selected) in any custom activity called by a Process.

©2025 Pegasystems Inc

> **NOTE:** "Flow" is a legacy term for a Process.

Activity Type

The value of the **Activity Type** field on the **Security** tab determines which Process shapes accept this activity. To create an activity for a Process, set the **Activity Type** field to `Utility`, `Rule Connect`, `Assign`, `Notify`, or `Router` as appropriate.

First, explore the standard activities of the type you need, as examples and as starting points for your activity.

In all but rare situations, choose a class derived from the *Work-* baseclass for the **Applies To** key part of your activity. This makes your activity available to Processes in that class plus all subclasses of the class (subject to security restrictions, Rule resolution restrictions, and so on).

Guidelines for Utility activities

Utility	Follow these guidelines to create a Utility activity, one that can be referenced in a Utility shape: • Select **Utility** as the **Activity Type**. • In the activity, do not use the *Show-HTML* method or interact with a user. • The activity can save a new or updated object with the *Obj-Save* method. • Optionally, this activity can use the *TaskStatus-Set* method to return a text value in the *pxTaskStatus* property to the Process as a basis for decisions. Connectors leading from the activity can be based on this status value.

©2025 Pegasystems Inc

	Using this method can simplify Process diagrams by eliminating fork or Decision shapes that follow a Utility shape.

Guidelines for Assign activities

Assign	Follow these guidelines and constraints to create an activity of the Assign type, one that you can reference in an Assignment shape: • In the activity, do not use the *Show-HTML* method or other means to interact with human users. • Create and save an instance of a concrete class derived from the *Assign-* baseclass. Typically, a new Assignment is an instance of the *Assign-Workbasket* or *Assign-Worklist* standard class.

Guidelines for Integrator activities

Integrator	Use an **Activity Type** value of *Connect* Rule to make the activity available to an Integrator shape. Generally, use this for activities in a *Work-* class that call a Rule to start a connector interface to an external system. This activity type (and the corresponding Integrator shape) is a reminder that this shape depends on an external system.

©2025 Pegasystems Inc

	Response time, availability, and performance may be affected by outside factors. Like the Utility activity type, a Connect activity type cannot include HTML displays.

Guidelines for Notify activities

Notify	Select an **Activity Type** of the Notify type to identify activities in a *Work-* class that generate Correspondence. Correspondence generation can occur without any user interaction, or can capture user input in a simple HTML form, or may start Microsoft Word on the user desktop.

Guidelines for Route activities

Route	Follow these guidelines to create an activity with **Activity Type** of the Route type, one that can be referenced in a Router shape: • The results of the activity are a Work Queue name, an Operator ID (for the Worklist) or an agent name. • The activity does not display HTML forms or interact with a user. • The activity returns its results in an output parameter named `AssignTo`. This parameter is an output only, and

©2025 Pegasystems Inc

does not need to be declared on the Parameters tab.

Many routing activities accept these Boolean input parameters:

Parameter	Description
`CheckForAvailability`	If this parameter is true, the activity is to check Operator ID instances for availability, and follow the Rules to obtain a substitute if the operator is not available. (Standard routing activities support this parameter.)
`SwitchToWorkbasket`	Allows an Assignment to be routed to either a Work Queue or Worklist. Use this Boolean output parameter in a router activity that is referenced in Assignment shapes where the

©2025 Pegasystems Inc

Parameter	Description
	Rule field on the Assignment Parameters panel is set to `Worklist`. When `SwitchToWorkbasket` is set to false, the `AssignTo` parameter identifies an Operator ID for a Worklist; when set to true, the `AssignTo` parameter is to identify a Work Queue name.
`PassthroughAssignment`	Allows creation of an Assignment instance to be bypassed, with the Process execution continuing with a default flow action — the most likely flow

©2025 Pegasystems Inc

Parameter	Description
	action — that requires no user input. Set this Boolean output parameter to false in the normal case, or true to skip the Assignment and continue Process execution as if the user chose the highest-likelihood flow action.
`UsesSkills`	Influences the **Properties** panel to allow the entry of one or more skill names (literal values or property references) and skill levels (integers). A checkbox for each skill can

©2025 Pegasystems Inc

Parameter	Description
	mark the skill as required to perform an Assignment, or (if not selected) desirable to perform the Assignment. The activity can call standard Functions to test whether a specific operator possesses the required skills, or possesses the desired skills, at the indicated level of proficiency.

The standard *Routing* library (in the *Pega-ProCom* Ruleset) contains several Functions useful in router activities.

Available parameters

When a Process calls an `Integrator`, `Utility`, `Notify`, or `Assign` activity, or another Process, it provides twelve parameters to the activity beyond those declared in

©2025 Pegasystems Inc

the **Parameters** tab. Click the **Show System Parameters** button on the **Parameters** tab to review the names and purpose of the six parameters most often used.

Parameter	Description
`flowName`	The subscript of this Process in the work page *pxFlow* property, a `Page Group`.
`TaskName`	The shape name of the shape that this activity or Process was called from, such as `SendResolutionEmail` or `SplitForEach999`.
`pyDraftMode`	True if the Process is in draft mode.
`ReferenceInsKey`	*pzInsKey* of the work item page.
`ReferenceClass`	Class of the work item page (the work type).
`ReferenceInsName`	*pxInsName* of the work item page.
`ReferencePageName`	Clipboard page name of the work item page.
`TimeFlowStarted`	Date and time that the Process execution began.
`InterestPage`	Property reference to the embedded page of the work item that is the Primary Page of the Process. (This is null if the Process's Primary Page contains the work item.) For example, a Process executing on the `Customer` work party has an interest page of *.pyWorkParty(Customer)*.
`InterestPageClass`	Class of the interest page, for example *Data-Party*. This is an empty string if the Process's Primary Page is the work item itself.

©2025 Pegasystems Inc

Parameter	Description
`FlowHasEnded`	A Boolean used by internal Process processing to control whether one Process calling another is to wait or to continue.
`flowType`	Second key part of the Process.

- Creating an activity
- Creating a stand-alone Process
- Process shapes
- Assignment shapes in Processes
- Assigning tasks to users
- Flow Actions
- Creating a Notification Rule

Intelligent routing

Intelligent routing is the process of comparing the characteristics of a new Assignment with the characteristics of the workforce to route the Assignment to the most appropriate operator. Like a supervisor who thoughtfully distributes work to her team, intelligent routing in your application can significantly affect the productivity and throughput of a team.

An intelligent routing algorithm can examine many factors, such as backlogs, the presence or absence of operators, operator skills, the Urgency or priority of an Assignment, and a customer's location or time zone. Your system includes the following three standard activities as examples:

- *Work-.ToSkilledGroup* — Implements skill-based routing. This activity sends an Assignment to a randomly selected operator within a specific team. This operator is available between the time the Assignment starts and is due, and has the minimum skill proficiencies to complete the Assignment. Using indexes maintained in the *Index-OperatorSkills* class, this activity operates efficiently.

©2025 Pegasystems Inc

- *Work-.ToLeveledGroup* — Sends an Assignment to an operator within a specific team who has the least urgent total Worklist, based on a computed score.
- *Work-.SampleToSkilledGroupList* — Demonstrates use of the *getAvailableSkilledOperatorsList()* Function. Part of the *Routing* library, returns a list of operators who are available and meet the required skill requirements. Your routing algorithm can apply additional filtering and criteria to select which of the operators on the list receives the Assignment. (If none of the operators in the group meet the standard Function qualifications, the list contains only the team manager's Operator ID.)

Skill-based routing is an important type of intelligent routing. Skill-based routing compares the skill proficiency of operators in a team with the required and desired skills to perform an Assignment accurately and quickly. For example, an Assignment might require an operator who passed the NASD Series 7 exam, and might prefer that the operator have a speaking proficiency in Spanish.

Skill-based routing affects only which Work Queue or which operator Worklist receives an Assignment. Skill-based routing does not prevent an operator who does not meet the skill profile used in routing from accessing an Assignment in a Work Queue and performing the Assignment.

Related concepts

- Automated and intelligent work assignment with Get Next Work
- Assignment shapes in Processes
- Flow Actions

Related tasks

- Defining work routing settings for an operator
- Assigning tasks to users

Related reference

©2025 Pegasystems Inc

- Fields for operator teams, Work Queues, and schedules
- Process shapes

Unit testing an activity

You can test an activity individually before testing it in the context of the application that you are developing. Additionally, you can convert the test run to a Pega unit test Case.

You can use unit test the activity if one of the following criteria is met:

- The **Require Authentication to run** checkbox is selected on the **Security** tab of the **Activity** form.
- Your Access Role allows you to update Rules that have the *Applies To* class of this activity.

In the navigation pane of Dev Studio, click **Records** > **Technical** > **Activity**, and then select the activity that you want to open.

- Creating an activity
- Unit testing individual Rules
- Using the Clipboard tool
- Opening a unit test Case
- Application debugging by using the Tracer tool

Clipboard contents created from activity unit tests

When you test activities with the Run Rule feature, the system might create one or more Clipboard pages, including copies of the named pages that you specified in the **Run** dialog box. The system retains these pages when you close the dialog box; properties on these pages remain available for you to use when you perform subsequent testing of this or other Rules with the Run Rule feature.

If the activity returns results to a parameter marked as `Out` on the Parameter tab of the activity, you cannot see the results in the Clipboard. Out parameters reside on the

©2025 Pegasystems Inc

parameter page of the activity, which you can view only with the Tracer tool. To simplify testing, you can modify the activity temporarily to store an Out parameter value in a property that is visible on a Clipboard page. Delete the temporary step when you finish testing.

Testing an activity in context

If the activity requires extensive setup and is more appropriately tested in context rather than through unit testing, you can trigger the Tracer to start only when the activity is reached.

To test an activity in the context of other activities and Rules that produce the needed initial conditions, complete the following actions:

1. Open the activity.
2. Select **Actions** > **Trace**. The Tracer tool starts.
3. To avoid using more memory than needed while tracing, click **Settings** in the Tracer window and ensure that only the Ruleset containing the activity is selected in the **Rulesets To Trace** section.
4. Begin the other processing that eventually calls the activity in your own requestor session. Tracer output begins when the activity starts.

- Application debugging by using the Tracer tool

Constraints Rules

Create constraints Rules to define and enforce comparison relationships among property values. Constraints Rules can provide an automatic form of property validation every time the property's value is "touched", in addition to the validation provided by the property or other means.

The following tabs are available on this form:

- Constraints

©2025 Pegasystems Inc

The system evaluates constraints automatically each time a property identified in a constraints Rule is changed. This technique is known as forward chaining.

Where referenced

No other Rules explicitly reference constraints Rules. When you save a constraints Rule, it is enforced immediately and thereafter. The system automatically adds a message to any property that is present on the Clipboard and fails a constraint. This message marks the page containing the property as invalid and ordinarily prevents the page from being saved.

Use the Application Explorer to access the constraints Rules that apply to work types in your application. Use the Records Explorer to list all the constraints Rules available to you.

After you complete initial development and testing, you can delegate selected Rules to line managers or other non-developers. Consider which business changes might require Rule updates and if delegation to a user or group of users is appropriate. For more details, see Delegating a Rule or Data Type.

Category

Constraints Rules are instances of the *Rule-Declare-Constraints* class. They are part of the Decision category.

Note the s in the Rule Type. The *Rule-Declare-Constraints* Rule Type replaces the *Rule-Declare-Constraint* Rule Type, which is deprecated.

- Viewing Rule history

Creating constraints Rules

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing Rules by creating a copy in a specific Ruleset, against a different class or (in some cases) with a set of circumstance

©2025 Pegasystems Inc

definitions. You can copy data instances but they do not support specialization because they are not versioned.

Create a constraints Rule by selecting `Constraints` from the `Decision` category.

> **NOTE:** Because constraints extend the definition of a property value, use nouns in the declarative expression names for better distinction from property values.

Key parts:

A constraints Rule has two key parts:

Field	Description
Apply to	Select a class for this Rule. At run time, a Clipboard Page of this class must be a top-level page. The properties to be constrained may be in this class (or in the class of an embedded page). You cannot use a Rule-Declare-* class or any ancestor of the *Rule-Declare* - class (including *@baseclass*) here. You cannot use a class derived from the *Code-* or *Embed-* class here.
Identifier	Enter a name for this constraint, unique within the **Apply to** class. Begin the name with a letter and use only letters, numbers, the ampersand character, and hyphens. No other Rules explicitly reference this **Identifier** value. However, because of normal

©2025 Pegasystems Inc

Field	Description
	class inheritance, a constraints Rule named MaximumLimit at one level in the class structure may override (and so prevent execution of) a constraints Rule named MaximumLimit at a higher level in the class structure.

Rule resolution

When searching for instances of this Rule Type, the system uses full Rule resolution which:

- Filters candidate Rules based on a requestor's Ruleset list of Rulesets and versions
- Searches through ancestor classes in the class hierarchy for candidates when no matching Rule is found in the starting class
- Finds circumstance-qualified Rules that override base Rules
- Finds time-qualified Rules that override base Rules

- Constraints Rules
- Viewing Rule history
- More about Constraints Rules

Constraints form - Completing the Constraints tab

Record the computations that constrain the property values. Each row defines a separate constraint. Enter at least one row. Order is not significant.

When the system detects that a constraint fails, it adds a message to the page of the property identified in the left portion of the **Require That** field (if any). The presence of this message ordinarily prevents the system from saving the page containing this property value to the database.

©2025 Pegasystems Inc

You can cause a specific additional message (*Rule-Message* Rule Type) to become associated with that property or another property when the constraint fails, in the **Else** row.

Detection of this message and any later processing to correct the constraint — such as displaying a message to a user, recalculation of the property value, and so on — is an application responsibility.

Each property involved in the constraint expression must be on the Primary Page (the page that has the class corresponding to the **Applies To** key part), or on an embedded page identified in **Page Context** field on the **Pages & Classes** tab, or on a linked property page.

Using the controls

Click any constraint to enable controls.

Control	Action
open icon	Click to review a property (for a field that contains a property reference). If the property does not exist, click the **Open** icon to create a new property.
Expression selector	Click the drop-down button to display a drop-down list of all boolean Expressions.
row buttons	Select any constraint to enable. Click the Add a row icon to add a new constraint, or click the Delete this row icon to delete the selected constraint. (Equivalently, use the **Insert**, **Shift+Insert**, and **Delete** keys.)
Display Label	When reviewing this tab, you can see a presentation using either property names or the Short Description (`pyLabel`) of

©2025 Pegasystems Inc

Control	Action
	properties. Click to toggle the display. (Note that two or more properties may have the same Short Description text.)

Modifying constraints with drag and drop operations

To move a constraint to a different location, click and drag the small circle located to the left of the entry.

To copy a constraint, drag the entry while holding down the `CTRL` key.

Completing fields

Field	Description
When	Click the drop-down button to select a boolean Expression from the drop-down list. The form changes to reflect the Expression selected. Most Expressions require properties. Use SmartPrompt to select a property. You can enter a linked property reference here.
Require that	This area defines the constraint. Click the drop-down button to select an Expression from the drop-down list. The form changes to reflect the Expression selected. Most Expressions require properties. Use SmartPrompt to select a property, or enter a linked property reference.

©2025 Pegasystems Inc

Field	Description
	You can't specify single elements of a `Value List`, `Value Group`, `Page List` or `Page Group` in property references here. Use only empty parentheses containing no index or subscript, for example: `pyWorkPage.pyWorkParty().pxCity = "Chicago"`
Else add message	Optional. Type the text of a message within quotes or use the SmartPrompt to select the name of a *Rule-Message*. The list of options is limited to `Error` and blank category messages. Commonly used messages include: • *ValueRequired* — Value cannot be null (blank) • *ValueOutOfRange* — Value is out of range Some messages require parameters. To provide fill-in-the-blank text values for each parameter, follow the message name with a backslash character, the single letter t, and text for each message parameter.
to	Optional. Identify the property to be marked with the message. The message remains until this property has a new value (or until

©2025 Pegasystems Inc

Field	Description
	an activity performs a Page-Clear-Messages method). Ordinarily, choose the property that failed the constraint. Some constraints involve two or more properties, on the same or different pages. Select the one most likely to have a new value that will restore the constraint condition. For example, if the constraint is of the form BoxWidth IS LESS THAN 40 then the message can be associated with the BoxWidth property. If the constraint has the form BoxWidth IS LESS THAN BoxLength then the message can be associated with either property. In unusual cases, you can leave this blank. When you leave this blank, the message is associated with the entire page rather than a property on that page. However, this is temporary, as page messages are removed upon the next user HTTP Submit operation. You can't use symbolic page names (such as `primary` or `steppage`) here, as they are

©2025 Pegasystems Inc

Field	Description
	not meaningful except within an activity execution context.

About Constraints Rules

- Constraints Rules
- Creating constraints Rules
- Viewing Rule history
- More about Constraints Rules

Constraints form - Completing the Pages & Classes tab

Complete this tab to list the pages referenced by name in the **Constraints** tab, and to identify the context of the properties constrained.

1. About
2. New
3. Constraints
4. Pages & Classes
5. History
6. More...

For basic instructions, see How to Complete a Pages & Classes tab.

Field	Description
Page Name	Optional. Identify a page that is the source of properties referenced on the **Constraints** tab.

©2025 Pegasystems Inc

Field	Description
	Optionally, add a row with the keyword `Top` as the page name, to identify a top-level page. The Top keyword allows you to use the syntax Top.propertyref to identify properties, on other tabs of this Rule form.
Class	Optional. Select the class of that page.
Locate Page Expression	The Locate Page Expression option is not supported. Use a Data Page instead. Rules created in previous versions that use the Locate Page Expression display this field. New Rules do not display this field. Changes to a Rule that uses this option cannot be saved unless Locate Page Expression is replaced by supported functionality.
Locate Page Activity	The Locate Page Activity option is not supported. Use a Data Page instead. Rules created in previous versions that use the Locate Page Activity display this field. New Rules do not display this field. Changes to a Rule that uses this option cannot be saved unless Locate Page Activity is replaced by supported functionality.
Page Context Data	
Page Context	Optional. Leave this blank if the constrained property or properties (on the Constraints tab) all have mode `Single Value`. Otherwise, identify a `Page List` or `Page Group` property on the Clipboard. You can

©2025 Pegasystems Inc

Field	Description

	supply literal index (subscript) values or property-reference index values. Precede the property reference with a period.
	
	To indicate that all the values of the index are acceptable, omit any index between parentheses. For example, these are acceptable:
	
	`Invoices.py Orders(2).pyItems("Manuals").pyItems`
	
	`Invoices.py Orders().pyItems().pyItemNames`

Page Class	Optional. If the Page Context field is not blank, identify the class of the page or pages that are identified in the Page Context field. (You cannot use the `$ANY`, `$NONE` or `$CLASS` keywords here.)

About Constraints Rules

- Constraints Rules
- Creating constraints Rules
- Viewing Rule history
- More about Constraints Rules

More about Constraints Rules

Conflicting constraints

©2025 Pegasystems Inc

Multiple constraints that are impossible to meet are neither detected or rejected, such as A > B and B > A. Both properties are marked as not valid each time either is changed.

Primary Page

During execution of a constraints Rule, the page on which the Rule operates temporarily becomes the Primary Page. The page keyword `PRIMARY` and the results of the *tools.getPrimaryPage()* PublicAPI method reflect this change. When the constraints Rule completes, the Primary Page of the calling activity resumes as primary.

Alias Function Rules

The contents of the selection lists on the Constraints tab depend on property alias Rules and alias Function Rules.

Change tracking exceptions

The constraints are tested when a property value on the Primary Page (the page that matches the Applies To key part of the Rule) changes. The following do not cause the constraints to be evaluated:

- Changes to properties on other pages referenced in the Pages & Classes tab
- Saving a Rule form that contains a property on the page
- Retrieving a property value with the Obj-List or Obj-Browse method
- Retrieving a property with the RDB-List method, unless the ApplyDeclaratives parameter of that method is selected

Workstation display of constraint failures

When appropriate, your application can display constraint Rule violations immediately as a user changes an input value on a user form or flow action form, rather than later when the form is submitted. The user can see the constraint message immediately.

©2025 Pegasystems Inc

For example, the constraint may require that the Due Date field be at least a week after the Start Date field. As a user enters and revises work item data, the constraint is checked immediately as user focus leaves either input field.

This feature can improve user productivity and accuracy, while also reducing the number of server interactions and HTTP traffic required to complete a valid input form. Consider whether and where such interactive checking may simplify the user's task of completing complex input forms in your application.

To implement this capability:

1. For best results, use this feature on flow actions or harnesses that use the SmartFrames layout and JSP tags rather than declaratives.
2. Include the properties involved in the constraints on the same flow action form, or on the same user form (in the same or different sections). (Don't place the input in the flow action form and the target in the harness form or vice versa.)
3. Select the **Enable Expression Calculation?** box on the HTML tab of the Flow Action form or Harness form.
4. Test.

This feature is based on AJAX technology.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

Testing and debugging constraints Rules

Using the Tracer, you can watch the evaluation of a constraints Rule. Start the Tracer and select a requestor session. Click **Settings** and check the Declare Constraint box in the Event Types to Trace section. Also check the Ruleset that contains the Rule to be traced.

The statistic Tracked Property Changes on the full details page of the Performance tool shows how many property changes have occurred (for the current requestor since log-in) that are tracked for declarative Rules computations. />.

©2025 Pegasystems Inc

OnChange activities

Constraints Rules, like Declare Expression Rules, do not evaluate during the execution of an OnChange activity.

- Property aliases
- Function Alias Rules
- Application debugging by using the Tracer tool
- Constraints Rules
- Creating constraints Rules
- Viewing Rule history

Data Transforms

A Data Transform defines how to convert data that is in one format and class (the source) into data of another format and class (the target). The supported formats are Clipboard and JSON. Using a Data Transform instead of an activity to set property values speeds up development and makes application maintenance easier.

The two main types of Data Transforms are the standard Data Transform and the JSON Data Transform. When you want to convert data formats, use the standard Data Transform to manage your application data within internal Pega Platform™ applications. When you need to integrate disparate data, for example, web-based data with information provided by internal sources, use the JSON Data Transform.

For more information about configuring a Data Transform, see the following topics:

- Configuring a Data Transform
- Configuring the JSON Data Transforms settings column
- Unit testing a Data Transform

Data Transforms are useful in the following situations:

- When using Data Pages to manage application data (see the Where referenced section below).

©2025 Pegasystems Inc

- In activities (see the Access section below).

You can create a Data Transform for either a Clipboard or JSON model format. You select the type of Data Transform in the **Additional configuration options** section on the Create Data Transform form. The options are different depending on which model format you choose.

For the JSON model format, you can choose whether to automatically map all data in the JSON string, or you can map individual properties. Mapping individual properties is useful when you have a large JSON string and you only care about a couple of properties, when you want to change the JSON structure, and when the fields in the JSON have different names from the Clipboard properties to which you are mapping. In addition, you can select classes to exclude.

Where referenced

The following Data Transforms that are used for data management use Data Pages:

- Optional data mapping on the property form – On the Edit Property form, the **Optional Data Mapping** field appears when you select **Copy data from a Data Page**. Use this Data Transform to copy a subset of the data from the Data Page to the property. If you do not specify a Data Transform, the system copies all of the data from the Data Page to the property.
- Request Data Transform – When a data source for a Data Page is a connector (an integration with an external data service), the request Data Transform lets you map Pega Platform data to the fields that the connector needs to communicate with the data service. Select the Data Transform to use in the **Request Data Transform** field on the Data Page Rule form on the **Definition** tab, in the **Data sources** section.
- Response Data Transform – Response Data Transforms normalize data provided by the data sources into the common application Data Model. A response Data Transform is required when the data source class is incompatible with the Data Page class (the recommended pattern to achieve true data virtualization). Select

©2025 Pegasystems Inc

the Data Transform to use in the **Response Data Transform field** on the Data Page Rule form on the **Definition** tab, in the **Data sources** section.

Other parts of Pega Platform that reference Data Transforms include:

- Several activity methods that operate on pages can use a Data Transform, such as the Apply-DataTransform, Apply-Model (deprecated), Page-New, Page-Copy, and Page-Change-Class methods.
- Starter flows, flows that create work items, can specify a Data Transform (on the flow's **Process** tab) for a Work- class to set initial properties for the work item.
- The **Action** tab for a flow action can specify that the system apply a Data Transform before or after displaying the flow action user interface to the user.
- For flows that have been edited by using the Flow form, Data Transforms can be specified on the **Set Properties** tab of the **Connector Properties** panel.
- In various user interface elements, which include the Client Event Editor's:
  - `Refresh This Section` action
  - `Refresh When` conditions
  - On-click control Rule actions `Refresh Section`, `Show Harness`, and `Open URL In Window`

Data Transform Rules can be referenced by other Data Transforms.

Rule Types that create Clipboard Pages (such as Connector Rules, Service Rules, and other Rule Types) can use Data Transforms.

Actions

For more information about a specific Data Transform action, see the following topics:

• Append and Map To • Append To • Apply Data Transform • Auto-map	• Otherwise When • Remove • Set • Sort

©2025 Pegasystems Inc

• Exit For Each Page • For Each Page In • Otherwise	• Update Page • When

Action restrictions

The following property types have limited support in Data Transforms and have no support in JSON data format transforms:

- Page Groups
- Value Lists
- Value Groups

Access

Use the Application Explorer to access Data Transforms that apply to the work types in your application. Use the Records Explorer to list all Data Transforms available to you.

Category

Data Transform Rules are part of the Data Model category. A Data Transform is an instance of the *Rule-Obj-Model* class.

Creating a Data Transform

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing Rules by creating a copy in a specific Ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Based on your use case, you use the Create, Save As, or Specialization form to create the record. The number of fields and available options varies by record type. Start by familiarizing yourself with the generic layout of these forms and their common fields using the following articles:

PEGA ©2025 Pegasystems Inc

- Creating Rules
- Creating a Rule specialized by circumstance
- Copying a Rule or data instance
- Creating a Rule specialized by circumstance

This information identifies the key parts and options that apply to the record type that you are creating.

Create a Data Transform Rule by selecting **Data Transform** from the **Data Model** category.

A Data Transform Rule has two key parts:

Field	Description
Apply to	Select a class that this Data Transform applies to. The list of available class names depends on the Ruleset that you select. Each class can restrict applying Rules to an explicit set of Rulesets as specified on the **Advanced** tab of the class form.
Name	Enter a name that is a Java identifier. Begin the name with a letter, and use only letters, numbers, and hyphens. See How to enter a Java identifier. A conventional name for the initial, and typically the simplest, Data Transform in a class is `pyDefault`. This name is not reserved and has no special significance.

©2025 Pegasystems Inc

- Data Transforms
- Unit testing a Data Transform
- More about Data Transforms

Configuring a Data Transform

To make application development maintenance more convenient and efficient, set property values by using Data Transforms. When you create a Data Transform, you convert data from one format and class to another format and class. For example, if a customer enters a shipping address on an order form, you can configure a Data Transform to populate the billing address with the same details.

1. In the header of Dev Studio, click **Create** > **Data Model** > **Data Transform**.
2. In the **Label** field, enter a short description for your Data Transform.
   For Data Transforms that initialize Cases, use the `pyDefault` name.
3. **Optional:** To manually set the identifier of your Data Transform, in the **Identifier** section, click **Edit**.
   By default, the system automatically populates this field with a read-only value that is based on the sentence that you enter in the **Identifier** section. The system ignores spaces and special characters.
4. In the **Additional configuration options** section, select a Data Model format for your Data Transform:
   - If you want to use the Clipboard tool as your Data Model, select **Clipboard**.
   - If you want to use JSON as your Data Model, select **JSON**.
5. In the **Context** section, specify the Pega Platform Ruleset context for your Data Transform:
   a. Select an Application Layer in which you want to store the record.
   b. In the **Apply to** field, select the class to which this Data Transform applies.
   c. In the **Add to Ruleset** field, select the name of a Ruleset to which you want to add the record, and then, in the list, select the version number.
6. **Optional:** To override the default work item that your application associates with this development change, in the **Work item to associate** field, press the Down arrow key, and then select a work item.

©2025 Pegasystems Inc

For more information about default work items, see Setting your current work item.

7. Click Create and open.
8. On the **Definition** tab, configure details of the Data Transform, based on the selected Data Model:

Choices	Actions
Configure the Data Transform by using the Clipboard tool	a. In the Action column, select an action that you want to perform on the data. For more information about actions available for the Clipboard tool, see Data Transform actions for Clipboard. b. In the Target column, specify the target of the selected action. c. If available, in the Relation column, specify the relationship between the target and the source. The Relation column will show available selections based on the action selected in step a. d. In the Source column, specify the source of the selected action. e. **Optional:** To add more actions in your Data Transform, click Add a row, and then repeat steps 8.a through 8.d. f. **Optional:** To chain together this Data Transform and Data Transforms with the same name in any of its parent classes, select the

©2025 Pegasystems Inc

Choices	Actions
	Call superclass Data Transform checkbox. At run time, the system first performs the actions in the highest-level Data Transform.
Auto-map the Data Transform by using JSON	a. Select the Auto-map all data checkbox. **Result:** At run time, the system maps the JSON string to the Clipboard. b. In the **Top element structure** section, select whether the structure for your Data Transform is an object or an array. c. If your element structure is an array, in the Pagelist Property field, select the property to which to map the array, and then select the JSON elements.
Configure the Data Transform by using JSON	a. In the **Top element structure** section, select whether the structure for your Data Transform is an object or an array. b. If your element structure is an array, in the Pagelist Property field, select the property to which to map the

©2025 Pegasystems Inc

Choices	Actions
	array, and then select the JSON elements. c. In the **Action** column, select an action that you want to perform on the data. For more information about actions available for JSON, see Data Transform actions for JSON. d. In the **Clipboard** column, select the Clipboard value to use for JSON serialization or deserialization. e. If available, in the **Relation** column, specify the relationship between the target and the source. The **Relation** column will show available selections based on the action selected in step a. f. In the **JSON** column, enter the array, object, or simple field name from the JSON data. g. In the **Empty behavior** column, control the mapping results of a JSON Data Transform by selecting an empty behavior. The configuration choices include the following options: **Default value** The JSON Data Transform serializer maintains the preexisting value. This

©2025 Pegasystems Inc

Choices	Actions
	configuration supports backward compatibility with Pega Platform that are earlier than 8.5. Null Represents a null value in the JSON Data Transform when the associated *ClipboardPage / ClipboardProperty* exists with an empty value, Skip Skips mapping in JSON when the associated *ClipboardPage / ClipboardProperty* exists with an empty value. h. **Optional:** To add more actions to your Data Transform, click **Add element**, and then repeat steps 8.c through 8.g.

9. **Optional:** To hide all child rows, click **Collapse All**.
10. **Optional:** To show all child rows, click **Expand All**.
11. Click **Save**.

What to do next:

To make the Data Transform available for reuse in App Studio, mark it as a relevant record. As a result, complex and advanced Data Transforms can be authored in Dev Studio, and then reused in App Studio, for example, for Case

©2025 Pegasystems Inc

> Type design. For more information, see Marking a record as relevant and Populating Case data with a Data Transform.

- Data Transforms
- More about Data Transforms

Data Transform actions for Clipboard

To save time while processing your Cases, populate data in your application by using Data Transforms. When you use Data Transforms, you can convert data from one format and class to another format and class.

When you configure a Data Transform using the Clipboard Data Model, you select an action that you want to perform on the data. You can select the following actions:

Set
: Sets the target to equal the source. You can set single-value properties, top-level pages, embedded pages, page lists, and page groups. You typically use this action to initialize properties. If the target Pages that you specify do not exist on the Clipboard, the Set action creates the Pages.

Remove
: Deletes the target and any associated values from the Clipboard.

Update Page
: Sets the context for setting properties on the target page. Use to set properties on a page that is different from the Primary Page. If the target page does not already exist on the Clipboard, the system creates the page. If the source page is different from the Primary Page, from the Relation list, select with values from, and then specify the source page.

Apply Data Transform
: Applies another Data Transform to a Clipboard page that the previous action specifies in the current context.

Sort
: Creates custom ordered lists. You can sort a page list according to single-value properties in that page list. You can specify a sorting order for each property as

©2025 Pegasystems Inc

either ascending or descending order. In the Target column, click the View Sorting Properties icon to specify the properties on which to sort and the sorting order for each property.

Comment

Specifies descriptive text. Use this action to provide comments within the sequence. For example, you can explain the goal of a branch of steps.

When

Specifies a condition to evaluate to determine whether to apply the subsequent actions. If the condition evaluates to true, the system applies the action within the child rows. If the condition evaluates to false, the system continues with the action specified within the next row. If the action in the next row is Otherwise or Otherwise When, the system applies the action if the condition evaluates to true.

Otherwise When

Specifies another condition to meet before the system applies an alternate condition. If the condition evaluates to true, the system applies actions within the child rows. If the condition evaluates to false, the system continues with the action specified within the next row. You can use this action after you use a When or Otherwise When action.

Otherwise

Specifies the alternate actions to apply if the preceding When or Otherwise When action evaluates to false. You can only use this action after a When or Otherwise When action.

> **NOTE:** If you add When, Otherwise When, and Otherwise actions to your Data Transform, the system performs only one of these actions, depending on the first of these actions that evaluate to true.

Append to

Copies a page to the target. You can copy a page if the source and the target are of the same class, or if one of the classes is higher in the class hierarchy in the inheritance path.

©2025 Pegasystems Inc

Append and Map to

Adds a page to the target and maps properties and their source values to the target properties. The source and target can be of different classes.

For Each Page In

Iteratively applies actions specified in the subsequent rows to all the pages in the specified target. Ensure that the target is a Page List or a Page Group.

Exit For Each

Exits the current iteration defined by the preceding For Each Page action. You can use this action only as a child row of a For Each Page In action.

Exit Data Transform

Stops the Data Transform at the row that contains the Exit Data Transform action.

- Standard Data Transforms
- Data Transform actions for JSON

Data Transform actions for JSON

Save time while processing your Cases by providing data for your application through Data Transforms. When you apply a Data Transform, you populate target values with values from a source that you specify, so that you avoid providing the same data twice.

When you configure a Data Transform by using JSON, you select an action that you want to perform on the data. You can select the following actions:

Set

Sets the target to equal the source. You can set single-value properties, top-level pages, embedded pages, page lists, and page groups. You typically use this action to initialize properties. If the target Pages that you specify do not exist on the Clipboard, the Set action creates the Pages.

Auto-map

Automatically maps the JSON data to the Clipboard.

Update page

Sets the context for setting properties on the target page. Use to set properties on a page that is different from the Primary Page. If the target page does not already

©2025 Pegasystems Inc

exist on the Clipboard, the system creates the page. If the source page is different from the Primary Page, from the Relation list, select with values from, and then specify the source page.

Append and Map to

Adds a page to the target and maps properties and their source values to the target properties. The source and target can be of different classes.

Apply Data Transform

Applies another Data Transform to a Clipboard Page that the previous action specifies in the current context.

Comment

Specifies descriptive text. Use this action to provide comments within the sequence. For example, you can explain the goal of a branch of steps.

- Standard Data Transforms
- Data Transform actions for Clipboard

Data Transform form - Append and Map to action

Use the **Append and Map** to action to append a page to the target Page List mode property and set the context to that page for subsequent child actions to map properties on that page. The target and source can be of different **Applies To** classes.

Usage

The **Append and Map to** action is equivalent to doing an **Update Page** action using the `<APPEND>` keyword. When the system invokes the **Append and Map to** action, it creates a new page in the target Page List property, without creating any embedded properties in that new page. Subsequent child actions create the embedded properties on the page.

When you select the **Append and Map to** action, the system automatically displays a child **Set** action because no properties exist yet on the newly created page.

©2025 Pegasystems Inc

When you use a source that is a Page List or a Page Group property (using the `each page in` choice for the Relation), the **When** icon appears next to the `each page in` choice. Click this button to specify a When Condition Rule for the **Append and Map to** action, so that a page from the source is appended to the target only if the condition is true. For example, when your target is a Page List property of PreferredCustomers, and you want to map data from an AllCustomers source Page List property only for those customers that are "preferred" customers, click the **When** icon and specify a When Condition Rule named `IsPreferred`.

JSON Data Model format

When the Data Model format for a Data Transform is JSON, there are the following differences in behavior from the Clipboard Data Model format:

- The order of the steps is not guaranteed.
- The Data Transform is bidirectional and can be called in either direction using the *executionMode* parameter. This parameter indicates whether to use JSON as the source and the Clipboard as the target or use the Clipboard as the source and JSON as the target.
- There are two parameters for the JSON Transform. JSON Transforms cannot have custom parameters.
  - *jsonData* – Specifies the incoming or outgoing JSON data
  - *executionMode* – Specifies the direction of the transform, SERIALIZE or DESERIALIZE.
- The **Pages & Classes** tab is only used to specify the class of classless properties, such as pxResults.
- JSON Data Model format transforms cannot reference other top-level pages. You can only reference fields on the Primary Page.

In addition, the **JSON** field requires a JSON array name.

Example

Append and Map to `pyWorkPage`

©2025 Pegasystems Inc

Relationship settings

The following relationship settings are available when the Data Model format is Clipboard:

- **a new page** – The default. Use to create a new page in the target Page List, and subsequently create properties on that page.
- **an existing page** – Use to create a new page in the target Page List, and subsequently create properties on that page that are to be mapped from a source page.
- **each page in** – Use to create a new page in the target Page List, and subsequently create properties on that page that are to be mapped from a source Page List or Page Group.

The following relationship settings are available when the Data Model format is JSON. Select the setting that matches the structure of the JSON array.

- **An array of objects** – This is the only option that translates directly to the Clipboard. There are no restrictions on the number of child steps or siblings.
- **An array of scalars** – In JSON, an array of scalars has the format [1,2,3,...] or ["a","b","c",...]. When you choose this option, there is exactly one child step for the **Append and Map to** step. You cannot add siblings or disable the step. The step is a set step and the JSON side states, "each scalar element." You can specify the Clipboard field to map to.
- **An array of arrays** – In JSON, an array of arrays has the format [[...],[...],...], where each element is a nameless array. When you choose this option, there is exactly one child step for the **Append and Map to** step. You cannot add siblings or disable the step. The child step can have either an **Auto-map** or **Append and Map to** action. The JSON side states, "each array element." You can pick a Page List for the Clipboard side. If you add an **Append and Map to** step, the same options are available again (array of objects, array or scalars, and array of arrays), so that you can configure embedded child steps.

©2025 Pegasystems Inc

- A group of pages – Select this option to configure mapping to a property that contains an unordered group of embedded pages. You can map a group of pages in either direction, from the source to the target or the reverse. When you choose this option, you can set child steps to map the data for the Append and Map to step. You can add siblings, disable the steps, and perform the full set of actions available in the Action column.

Supported features

You can use the following items in the Append and Map to action:

- In the **Target** column, Page List mode properties
- In the **Source** column, Page, Page List, and Page Group mode properties
- Conditionalizing the Append and Map to action, when the source is a Page List or a Page Group (using the When icon)

Restrictions

You cannot specify a Page Group mode property as the target for the Append and Map to action. That is, you cannot append to a Page Group mode property by using this action.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Append to action

Usage

Use the Append to action to copy a page from the source to the target, or to copy all of the elements in a source Page List property to a target Page List property. The target must be of a class that is compatible with the source page. (To append a page to a

©2025 Pegasystems Inc

target having a class that is not compatible with the source's class, or to conditionally map source data, use the Append and Map to action.)

Example

In this example, `Shipments` and `ShipmentGroup` are properties having the same Page Class. `Shipments` is a Page List property, and `ShipmentGroup` is a Page Group property. The Append to action specifies to copy each page from `ShipmentGroup` to `Shipments`.

Append to `.Shipments` each page in `.ShipmentGroup`

Relationship settings

- `a new page` — The default. Use when you want to create a new page in the target Page List property.
- `an existing page` — Use when you want to copy a source page as the new page in the target Page List property.
- `each page in` — Use when you have a Page List or Page Group for the source, and want to copy pages in the Page List or Page Group as new pages in the target Page List.

Supported features

You can use the following items in the Append to action:

- In the Target, Page List mode properties
- In the Source, Page, Page List, and Page Group mode properties

Restrictions

You cannot specify a Page Group mode property as the target for the Append to action. That is, you cannot append to a Page Group mode property.

- Data Transforms

PEGA ©2025 Pegasystems Inc

- Creating a Data Transform
- More about Data Transforms

Data Transform form - Apply Data Transform action

Use the **Apply Data Transform** action to apply actions defined in another Data Transform (the applied Data Transform).

Usage

To apply the Data Transform to a specific page, specify a row by using an **Update Page** action before the **Apply Data Transform** action row. To apply the Data Transform to a specific page, precede the **Apply Data Transform** action with an **Update Page** action to identify the page.

JSON Data Model format

When the Data Model format is JSON, there are the following differences in behavior:

- The order of the steps is not guaranteed.
- The Data Transform is bidirectional.

Relation column

The **Apply Data Transform** action has no setting for the **Relation** field.

Parameters in the to-be-applied Data Transform

The applied Data Transform can have parameters defined on its **Parameters** tab. In the **Apply Data Transform** action row, click the **View Data Transform Parameters** icon to provide values for the applied Data Transform's input parameters (if any), or specify whether to pass the current parameter page.

When the **Pass current parameter page?** checkbox is selected, the two Data Transforms (the one that has the **Apply Data Transform** row and the applied Data Transform

©2025 Pegasystems Inc

specified in that row) share one parameter page. If the checkbox is not selected, the parameters on the parameter page for the applied Data Transform are not available to the other Data Transform, and vice versa.

If the current parameter page is shared, and the applied Data Transform has a parameter declared as `Out`, if the applied Data Transform updates the value of that parameter, the resulting value is available to the other Data Transform.

> **NOTE:** If the Data Transform that you are calling changes any values on the parameter page, those values will be lost if you do not select the **Pass current parameter page?** checkbox. For example, when calling a JSON Data Transform using the SERIALIZE value for the *executionMode* parameter, the resulting JSON data is written to the parameter page as a value of *jsonData.*

Restrictions

You cannot call a Clipboard Data Transform from a JSON Data Transform.

You cannot use the `Top` or `Parent` keywords in the entry field when specifying the applied Data Transform.

You cannot use Expressions when specifying the applied Data Transform.

When `$ANY` is used in the entry field (to make the class of the applied Data Transform assignable at run time), design-time validation does not check whether the specified Data Transform exists when you save this Data Transform form.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

©2025 Pegasystems Inc

Data Transform form - Auto-map action

Use the **Auto-map** action to contextually and automatically map data from JSON to the Clipboard or from the Clipboard to JSON.

Usage

Use the **Auto-map** action with a Page or Page List property that has a Data Model of the property's page class that exactly matches the JSON Data Model. The name and type of every child field is the same in both the JSON data and the Clipboard.

> **NOTE:** To ensure that the **Auto-map** action runs properly, as a best practice, define the desired Pega Platform Data Model in your application hierarchy.

For more information about mapping and JSON serialization and deserialization, see the following articles on Pega Community:

- *Using the mapping API for high-performance JSON serialization*
- *Using the mapping API for high-performance JSON deserialization*

Relation column

The **Auto-map** action has a fixed **Relation** field setting of `To`. That is, the **Auto-map** action always sets the target to the source and establishes no other sort of relationship between the two.

> **NOTE:** When serializing JSON, the source is Clipboard, and when deserializing JSON, Clipboard is the target.

©2025 Pegasystems Inc

Restrictions

The only property type values supported are:

- Page
- Page List

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - For Each Page In and Exit For Each actions

Usage

Use the For Each Page In action to have the system apply the subsequent actions to every page in a target Page List or Page Group mode property. After the For Each Page In action row, the system goes down the subsequent children rows and iteratively applies the actions in those rows to each page in the target.

Use the Exit For Each action to stop the iterations of the For Each Page In action. Actions at the same level as the Exit For Each row (its siblings) are not performed by the system.

The Exit For Each action is typically used as part of an overall conditional structure. For example, in the following shopping cart example, `Items` is a Page List property. if the user adds a product item (.PRODUCTNAME) and quantity to his shopping cart, the When action checks to see if that product is already in the cart, for each page of `Items` and if so, updates the quantity in the shopping cart with the additional quantity for that item. After the item is found in the cart, the Exit For Each action stops the system from iterating over the rest of the pages in `Items` and omits the actions for the Otherwise case (which are not needed once the item is found to exist in the cart).

PEGA ©2025 Pegasystems Inc

```
For Each Page In pyWorkPage.Items
  When .Name == Primary.PRODUCTNAME
    Set pyWorkPage.ItemToAdd.LocationFound equal to pyWorkPage.Items(<CURRENT
>).pxListSubscript
    Set pyWorkPage.ItemToAdd.Found equal to "true"
    Set pyWorkPage.ItemToAdd.Quantity equal to pyWorkpage.Items(<CURRENT>).Qu
antity
    Exit For Each
  Otherwise
    Set pyWorkPage.ItemToAdd.Found> equal to "false"
```

Example

```
For Each Page In .Customers
  When IsPreferred
    Set Discount equal to 15
  Otherwise When IsExisting
    Set Discount equal to 5
```

Relationship settings

The For Each Page In and Exit For Each actions have no settings for the Relation field.

Supported features

You can use the following items in the For Each Page In action:

- Page Lists
- Page Groups
- Parameters specified as `Page Name`

©2025 Pegasystems Inc

Restrictions

You cannot use Page List properties that have Lightweight specified in their Rule forms as sources.

You cannot iterate over a source. You can only iterate over target pages.

You cannot use the `<APPEND>` keyword in the For Each Page In action. Instead, follow the For Each Page action with an Append to action or Append and Map to action.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Remove action

Usage

The Remove action deletes the specified target and any associated values from the Clipboard.

Example

Remove `pyWorkPage.ItemToAdd.Quantity`

Relationship settings

The Remove action has no setting for the Relation field.

Supported features

You can use the following items in the Remove action:

- Properties
- Pages (top-level or embedded Clipboard Pages)
- Page Lists
- Page Groups

©2025 Pegasystems Inc

- Value Lists
- Value Groups
- Data Pages (pages created by Data Page Rules).
- `Primary` keyword and a property (for example, `Primary.Firstname`)

Restrictions

To remove a top-level page, the page must be specified on the Pages & Classes tab.

You cannot use `Top`, `Parent`, or `param` in the Remove action.

You cannot remove the Primary Page (the page of the Applies To class of the Data Transform).

You cannot remove a parameter (specified on the Parameters tab).

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Set action

Use the **Set** action to set the target equal to the source. When you specify a target page that does not yet exist on the Clipboard, the **Set** action creates the page.

Usage

To select a value for the target property from the set of values defined in Case instances for the property, click **Select values**.

JSON Data Model format

When the Data Model format is JSON, there are the following differences in behavior:

- The order of the steps is not guaranteed.
- The Data Transform is bidirectional.

©2025 Pegasystems Inc

Examples

```
Set .pyWorkIDPrefix equal to "W-"
Set .HardwareItems(1).Price equal to ""
```

Relation column

The Set action has a fixed **Relation** setting of `equal to`. That is, the Set action always sets the target equal to the source, and establishes no other sort of relationship between the target and source.

> **NOTE:** When serializing JSON, the source is Clipboard, and when deserializing JSON, Clipboard is the target.

Supported features Clipboard Data Model format

When the Data Model format is Clipboard, the Set action supports the target and source combinations shown in the following table. These features are not supported when the Data Model format is JSON.

The class of the target does not have to match the class of the source.

Target	Relation	Source	Example
Single Value Property	equal to	Property Reference	.Department
Single Value Property	equal to	Literal	"W-"
Single Value Property	equal to	Expression	.FIRSTNAME + " " + .LASTNAME
Single Value Property	equal to	Linked Property Reference	.pxCreateOperator.pyUserName

©2025 Pegasystems Inc

Target	Relation	Source	Example
Page	equal to	Page	OrderItems(1) equal to HardwareItems(1)

The source and target support the following items:

- Properties
- Literals
- Expressions (in the source column only)
- Parameters specified on the **Parameters** tab (for example, `param.Name`)
- Page Lists
- Page Groups
- Value Lists
- Value Groups
- Keywords: `Primary`, `<APPEND>`
- Function calls (in the source column only)

When the **Set** action is preceded by an **Update Page**, the context of the source is reset and all property references are relative to the new page.

Restrictions

The use of `Top` and `Parent` keywords is not supported.

You cannot set properties on Data Pages. Such properties cannot be specified as targets.

References to linked properties are read-only. Such properties can be specified as sources, but not as targets.

You cannot set reference properties (properties that have the **Reference Property** checkbox selected on the **Advanced** tab of their Property forms). They cannot be specified as targets.

PEGA ©2025 Pegasystems Inc

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Sort action

Usage

Use the Sort action to create custom ordered lists. A typical scenario is you are displaying a multi-column list and you want the columns automatically sorted according to values in one or more of the columns. With the Sort action, you can sort a Page List property according to single-value properties in that Page List. For each single-value embedded property, you can specify sorting in either ascending or descending order of that property. Click the **View sorting properties** icon to specify the properties on which to sort and the sorting order for each property.

If you specify more than one embedded property to sort on, the first property specified is given precedence at run time. For example, you want a section to display a list of product items in an order, and the page list is `Items`, with embedded properties:

- `Quantity` (Integer type)
- `TotalPrice` (Decimal type)

And you want to show the user the list of items sorted first by quantity from smallest to largest, and then by total price from highest to lowest within a given quantity. In the Sort By Property window, specify the `Quantity` property in ascending order and the `TotalPrice` property in descending order.

at run time, the displayed rows run from the lowest quantities to the highest, and rows that have the same quantity are displayed with the highest total price first, then the next, and so on.

©2025 Pegasystems Inc

Relationship settings

The Sort action has no setting for the Relation field.

Supported features

You can use the following items in the Sort action:

- In the Target, Page List mode properties
- As the embedded properties on which to sort by, single-value properties. For each sorted-by property, you can sort in ascending or descending order.

Restrictions

You cannot specify a Page Group mode property as the target for the Sort action. That is, you cannot sort by properties in a Page Group mode property.

You cannot sort by properties in the Page List that are not single-value properties. For example, you cannot sort by a Page property in a Page List property.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Update Page action

Use the **Update Page** action to set values on a target page that is different from the Data Transform's Primary Page.

Usage

The **Update Page** action sets the page context for the subsequent (children) steps. Using the **Update Page** allows you to:

- Avoid entering the target page name each time in the children **Set** actions

©2025 Pegasystems Inc

- Use source values from a page that is different from the Data Transform's Primary Page

For example, the following Data Transforms are equivalent:

```
Set pyWorkPage.CustomerType equal to "Existing"
```

Set `pyWorkPage.CustomerLevel` equal to "Bronze"

and

```
Update Page pyWorkPage
  Set .CustomerType equal to "Existing"
  Set .CustomerLevel equal to "Bronze"
```

In the second example, the **Update Page** action sets the page context to the *pyWorkPage*, so that the page does not have to be specified for the **Target** fields in the following child steps.

You can set the page context for the target only, or for both the target and source. To set the page context for the source, select `with values from` in the **Relation** field.

JSON Data Model format

When the Data Model format is JSON, there are the following differences in behavior:

- The order of the steps is not guaranteed.
- The Data Transform is bidirectional.

In addition, the **JSON** field requires a JSON object name.

Examples

```
Update Page pyWorkPage
  Set .ZipCode equal to "02139"
```

©2025 Pegasystems Inc

```
  Set .TelPrefix equal to "617"
Update Page pyWorkPage with values from Employees
  Set .ZipCode equal to .PostalCode
  Set .TelPrefix equal to .Prefix
```

Relation column

The following relationship settings are available when the Data Model format is Clipboard:

- blank – The default. Use when setting values on a target page using source values that are on this Data Transform's Primary Page (the Clipboard Page with the same class as the Data Transform's **Applies To** class).
- **With values from** – Use when the source values are on a page other than the Data Transform's Primary Page.

The following relationship settings are available when the Data Model format is JSON:

- **For both sides** – Set the page context for both the JSON and Clipboard sides.
- **For JSON only** – Set the page context for the JSON side only.
- **For Clipboard only** – Set the page context for the Clipboard side only.

JSON relationship settings example

This example illustrates how to use the relationship settings. The JSON for a customer is as follows:

```
{
                "id": 123456,
                "name": "John Doe",
                "email": "john.doe@example.com",
                "address": {
                    "addressLines": {
                    "line1": "1 example st",
```

©2025 Pegasystems Inc

```
                "line2": "example building"
            },
                "city": "Example",
                "zip": "02142"
            }
        }
```

The Clipboard structure is as follows:

```
Customer
        ID
        Name
        Email
    Address
        AddressLine1
        AddressLine2
        City
        Zip
```

The tree structures match except for the address line fields. `line1` on the JSON side is under the `address > addressLines` hierarchy, but its corresponding `AddressLine1` property on the Clipboard is directly under `Address`. In this case, use the **Update Page** action to update context only on one side. To map the entire structure, update the context for address on both sides, then update the context only on the JSON side, and then map individual properties, as shown in the following sample:

Update Page `Address` **For both sides** `address`

Update Page `No context change` **For JSON only** `addressLines`

Set `AddressLine1` **equal to** `line1`

©2025 Pegasystems Inc

Supported features

You can use the following items in the **Update Page** action:

- Page (top-level or embedded Clipboard Page)
- Page in a Page List
- Page in a Page Group
- Keywords: `<APPEND>`, `<LAST>`, `Primary`
- Expressions in the index (subscript) of an aggregate property

Restrictions

- You cannot use the `Parent` keyword.
- Do not use the `Top` keyword in Data Transforms. In some situations, using `Top` does not work if a top-level page is not clearly defined (for example, if a row has both source and target pages).
- You cannot use Data Pages (pages that are instances of Data Page Rules).

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - When, Otherwise, and Otherwise When actions

Usage

Use the When, Otherwise, and Otherwise When actions to conditionalize Data Transform actions, so that the system invokes the actions only when the specified conditions are true.

For example, given an ordering Process with a form where customers can select a checkbox to indicate whether their billing address is the same as their shipping address, you can use a When action to tell the system to perform a set of actions (children actions of the When) when that checkbox is selected

PEGA ©2025 Pegasystems Inc

(`BillAddressSameAsShipping` property is true). By using an Otherwise action after the When action, you can specify another set of Data Transform actions to take if the checkbox is not selected.

In the list of rows on the Definition tab, an Otherwise When action row must be preceded by a When action row at some point in the list. An Otherwise action row must be preceded by either a When action row or an Otherwise When action row.

Example

In the following example, if the user selects **Billing address same as shipping address** checkbox in the order form, the property `BillingAddressSameAsShipping` is set to "true". In that case, the Data Transform actions following the When action specify to update the `pyWorkPage.pyWorkParty(Customer).BillingInformation` page using the Address value from the `pyWorkPage.pyWorkParty(Customer).ShippingInformation` page (that is, use the shipping address for the billing address). If the checkbox is not selected, the children actions of the Otherwise action specify to set the billing information page values to null strings, so that the user has to enter the data.

```
When pyWorkPage.pyWorkParty(Customer).BillingAddressSameAsShipping == "true"
  Update Page pyWorkPage.pyWorkParty(Customer).BillingInformation with values fr
om pyWorkPage.pyWorkParty(Customer).ShippingInformation
    Set .Address equal to .Address
Otherwise
  Update Page pyWorkPage.pyWorkParty(Customer).BillingInformation
    Set .Address equal to ""
```

Relationship settings

The When, Otherwise, and Otherwise When actions have no settings for the Relation field.

©2025 Pegasystems Inc

Supported features

You can use the following items in the rows for the When and Otherwise When actions:

- When Condition Rules
- Expressions
- `Primary` keyword

Restrictions

Do not use `<` or `>` in an Expression. Use `<=` or `>=` instead.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Configuring the JSON Data Transforms settings column on the Data Transform form

Pega Platform displays the **Settings** tab on the Data Transform Rule form for the JSON Data Model format. Use the **Settings** tab to enable multidimensional array output and to specify fields to skip during auto-mapping. You might want to skip fields that do not have meaning outside of Pega Platform, such as *pyObjClass*.

> **Before you begin:**
>
> Create a Data Transform with a JSON Data Model format. For more information, see Configuring a Data Transform.

1. In the Data Transform that you create, click the **Settings** tab.
2. To enable multidimensional array output, select the **Enable multidimensional array output** checkbox.
   Select this option when auto-mapping is enabled on the **Definition** tab and you have a *Page List* property with only one non-skipped property that has the same

©2025 Pegasystems Inc

name as its parent. If you enable multidimensional array output, the *Page List* property will result in a multidimensional array in the JSON output. If you do not select the **Enable multidimensional array output** checkbox, the *Page List* property is displayed as an array of objects.

For example:

Multidimensional array option enabled

```
Clipboard page                    Resulting JSON
    .pxResults(1)                   {
     .pxResults(1)                   pxResults: [[{
     .Name = “John Smith”                  Name: “John Smith”
                                    }]]
                                   }
```

In this example, the *pxResults* field in the JSON output has two array characters, denoting the output as an array of arrays.

Multidimensional array option disabled

```
Clipboard page                    Resulting JSON
    .pxResults(1)                   {
     .pxResults(1)                   pxResults: [{
     .Name = “John Smith”                  pxResults: [{
                                    Name: “John Smith”
                                   }]
                                  }]
                                 }
```

In this example, the output is an array of objects, in which each element has a *pxResults* field.

3. In the **Date format for serialization** field, select the output format to use when serializing a Data Page:

©2025 Pegasystems Inc

- When you select **Pega API**, the system formats the output as:

  Date: `1996-07-26`

  Date Time: `1996-07-26T09:30:00.000Z`

  Time of Day: `09:30:00.000Z`

- When you select **Pega Internal**, the system formats the output as:

  Date: `19960726`

  Date Time: `19960726T093000.000 GMT`

  Time of Day: `093000`

4. To specify which fields to skip processing during auto-mapping, in the **Fields to skip when auto-mapping** section, click **Add field** to add a field to the list.
5. To specify the default auto-map empty behavior, select an empty behavior in the **Default Automap Empty Behavior** field.

> **NOTE:** Select this option when the **Auto-map all data** checkbox is selected on the **Definition** tab.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Data Transform form - Completing the Pages & Classes tab

To use page names in fields on the **Definition** tab, you must specify those pages on this **Pages & Classes** tab.

©2025 Pegasystems Inc

at run time, the system uses information from this tab to locate properties on the Clipboard.

Specifying the pages on the Pages & Classes tab also assists when working with the Definition tab at design time. For example, when a page is specified here, it is listed in the SmartPrompts in the fields on the Definition tab.

See How to Complete a Pages & Classes tab for more details.

> **NOTE:** For JSON Data Model format transforms, this page is only used to specify the class of classless properties, such as pxResults.

Click the **Add item** icon to add new rows to the array. Click the **Delete** icon to delete a row.

Field	Description
Page Name	Enter the name of a Clipboard Page referenced on the Definition tab.
Class	Specify the class of that page. Click the **Open** icon to open the class Rule, or to create the class if it does not already exist.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

Adding Data Transform to a Process

To save time, define a place in your Case where a Data Transform occurs by adding the Data Transform to a Process. You typically add a Data Transform on a connector between two Assignments.

PEGA ©2025 Pegasystems Inc

For example, while processing a purchase order, you can populate a shipping address with data that a user provides as a billing address.

Before you begin:

Define the Data Transform that you want to apply on a connector.

> **IMPORTANT:** Ensure that the Data Transform call from a Flow connector does not include any REST connectors. Invoke REST connector calls from the Integrator shape or through the Run Data Transform shape. REST connector calls should not happen from connectors in Flows because it might lead to errors during processing. Data Transforms in Flow connectors perform simple procedural logic, such as setting fields. Although it is possible to call REST connectors through Data Transforms, it is not the primary intention of those Rules. For more information, see Creating a Data Transform and Configuring a Data Transform.

You can also call a Data Transform automatically from a Case Lifecycle in a low-code way in App Studio. For more information, see Populating Case data with a Data Transform.

1. In the navigation pane of Dev Studio, click **Cases**, and then click the Case Type that you want to open.
2. On the **Workflow** tab, hover over a Process to which you want to apply a Data Transform, and then click **Configure Process**.
3. On the toolbar, click **Open Process**.
4. Double-click a connector between two Assignments where you want to add a Data Transform.
5. In the **Connector properties** dialog box, in the **Set properties** section, select **Apply Data Transform**.

©2025 Pegasystems Inc

6. In the **Data Transform** field, press the Down arrow key, and then select the Data Transform.
7. Click **Submit**.
8. Click **Save**.

- Referencing properties
- Populating Case data with a Data Transform
- Data Transforms

Standard Data Transforms

Naming convention

By convention, the Pega Platform includes a standard Data Transform named `pyDefault` for most standard concrete classes and several standard abstract classes. For example, the *Data-Admin-Operator-ID.pyDefault* Data Transform provides initial values for about 25 properties in the Operator ID data instance.

This is not a reserved name and has no special characteristics. You may create Data Transforms with this name or other names in any class or Ruleset.

Data Transforms that apply to classes derived from Work-

Whenever a new work item is created, the Pega Platform uses the *pyFlowName* property in the object to identify a flow Rule to start.

If you create a Data Transform named `pyDefault` with the Applies To key part class derived from the *Work-* base class, include a value for the *pyFlowName* property. In most cases, select the Call SuperClass Data Transform checkbox to apply the ancestor *pyDefault* Data Transforms also.

Set the value of this property to the Flow Name key part of a flow Rule. When a user starts a new flow in the context of that Work- class, the system uses this value, with Rule resolution to find the flow Rule to run.

©2025 Pegasystems Inc

Applies To	Data Transform	Purpose
Work-	*pyCopyDefault*	Determines which properties are copied when a user copies a work item.
Work-	*pyDefault*	Provides default initial values for properties in new work items.
Work-Cover-	*pyDefault*	Provides default initial values for properties in new work items that are covers.
Work-Folder-	*pyDefault*	Provides default initial values for properties in new work items that are folders.
Work-.RuleCheckin	*pyDefault*	Provides default initial values for properties in new work items for the Rule check-in flow.
PegaSample-SimpleTask	*pyDefault*	Demonstrates entry of a Simple Task work item, not part of a cover.
PegaSample-CustomerRequest	*pyDefault*	Demonstrates entry of a cover work item.
PegaSample-Task	*pyDefault*	Demonstrates entry of a covered work item.

Data Transforms that apply to System-User-MyRules class

By default, the order and link labels for reports in the Monitor Activity work area are controlled by Data Transforms with System-User-MyRules as the Applies To key part. Override these standard Data Transforms and then modify your Rule to reference and label reports as desired.

PEGA ©2025 Pegasystems Inc

Applies To	Data Transform	Purpose
System-User-MyRules	*AssignmentMonitoringReports*	Determines the order and labels of reports (that are available to all users) in the Monitor Assignments area of the Monitor Activity work area.
System-User-MyRules	*DataReports*	Not used in PRPC Version 6.
System-User-MyRules	*PerformanceAnalysisReports*	Determines the order and labels of reports (that are available to all users) in the Analyze Performance area of the Monitor Activity work area.
System-User-MyRules	*RuleReports*	Not used in PRPC Version 6.
System-User-MyRules	*WorkAnalysisReports*	Determines the order and labels of reports (that are available to all users) in the Analyze Process Quality area of the Monitor Activity work area.
System-User-MyRules	*WorkMonitoringReports*	Determines the order and labels of reports (that are available to all users) in the Monitor Processes area of the Monitor Activity work area.

Other standard Data Transforms

Applies To	Data Transform	Purpose

©2025 Pegasystems Inc

(various)	*pyTrackSecurityChanges*	Affects data retained in history for data objects or Rules.

- Data Transforms
- Creating a Data Transform
- More about Data Transforms

More about Data Transforms

Using a Data Transform, you can specify initial properties for objects of a specific class and also map properties (and their values) of an instance of a class to an instance of a different class.

For example, you can ensure that a special set of risk-related properties is present on all instances of class Data-Loan-Aircraft-SingleEngine. When a page of this class is created, you can instruct the system to apply a Data Transform, named Aircraft-High-Risk. The system places into the page the risk-related properties defined in the Data Transform. When the page is later saved into the database as a persistent instance, the properties are present with meaningful values.

Typically, you use the actions within a Data Transform to update and apply data to Clipboard Pages. Additionally, there are four activity methods that can optionally use a Data Transform as a parameter:

- Apply-DataTransform — Updates existing Clipboard Pages with values computed from a Data Transform.
- Page-New — Creates a new named page of a specified class. If a Data Transform is specified in this method, the Data Transform's actions are applied to the page.
- Page-Copy — Copies the contents of a Clipboard Page to a new named page or replaces the contents of an existing page. If a Data Transform is specified in this method, the Data Transform's actions are applied to the destination page before the source's contents are copied.

©2025 Pegasystems Inc

- Page-Change-Class — Creates a new page of the specified class, and then copies properties from another page into it, according to the method's Keep parameter and the specified Data Transform (if any).

The Apply-Model method is deprecated in version 6.2, and replaced by the Apply-DataTransform method. To use the Apply-DataTransform method to achieve the same result of using the Apply-Model method with the OverwriteProperties parameter set to `'false'`, use When actions in the Data Transform to prevent values from being applied when properties are already present on the target page. For example:

```
When !@PropertyHasValue(.FirstName)
Set .FirstName = "Bob"
```

In the preceding example, when the *FirstName* property has a value, the condition is not satisfied and the Set action is skipped.

Service Rules (*Rule-Service-* classes) and a few other Rule Types can reference a Data Transform. The Data Transform is processed and applied at run time to a newly created page.

Viewing the Java code of a Rule

Click **Actions** > **View Java** to view the generated Java of a Rule. You can use the Java code to debug your application or to examine how Rules are implemented.

Changing the value of a TimeOfDay property

When you use an Expression on TimeofDay property type values, the Expression converts the value into a BigDecimal where the whole number is the number of days and the fraction is the hour of the day expressed as a fraction of a day, for example, 12 hours is 0.5 (half of a day). The format is useful to know when you want to change the value of a property that is a Date, DateTime, or TimeofDayproperty type.

PEGA ©2025 Pegasystems Inc

For example, assume that you have a property of type TimeofDaythat is formatted as a standard time with hour, minute, and second without punctuation. To add 12 hours to the property:

1. Use a Set action to assign the property value to a second property that is of type TimeofDay.
2. On the right side of the Expression, enter the property reference plus the fraction. In this example, enter .5 to add 12 hours.

- Constants in Expressions
- Data Transforms
- Creating a Data Transform

Unit testing a Data Transform

You can use the **Run Rule** feature to test a Data Transform individually before testing it in the context of the application that you are developing. Additionally, you can convert the test run into a Pega unit test Case.

1. In the navigation pane of Dev Studio, click **Records** > **Data Model** > **Data Transform** , and then select the data transform that you want to open.
2. Click **Clipboard** to open the Clipboard and examine the pages that are generated by the unit test. For more information, see Clipboard pages created by the Run Rule feature.

- Data Transforms
- Viewing test Case results
- Exporting a list of test Cases

Creating a Data Transform in App Studio

Create a Data Transform in a low-code way to more efficiently process data in App Studio.

PEGA ©2025 Pegasystems Inc

Use a Data Transform to move data between objects, calculate values, change formats, and set default values. For example, in an order processing application you can set an order delivery address as the default mailing address for a customer account.

1. In the navigation pane of App Studio, click **Case Types**, and then click the Case Type that you want to open.
2. In the Process in which you want to call a Data Transform, click **Step** > **More** > **Automations** > **Run Data Transform**, and then click **Select**.
3. In the **Step** properties pane, in the **Run Data Transform** list, select **Define new Data Transform**.
4. In the **Data Transform Name** dialog box, provide a meaningful name, and then click **Create**.
5. Click the **Edit** icon.
6. In the **Action** list, select the operation that you want this row of the Data Transform to perform.
7. In the **Target** field, select the object that will be impacted by the action.
8. In the **Relation** field, specify the relationship between the target and the source.
9. In the **Source** field, specify the object that the action will use as the input.

The Edit Data Transform window

For more information, see Configuring a Data Transform

10. **Optional:** To define parameters, classes and other specifications, in the upper-right corner of the **Edit Data Transform** window, click **Open Data Transform in Dev Studio**. For more information, see Configuring a Data Transform

©2025 Pegasystems Inc

> ⚠ **IMPORTANT:** This step can be performed only by users with access to Dev Studio. Otherwise the **Open Data Transform in Dev Studio** button is not visible.

Related tasks

- Populating Case data with a Data Transform
- Using Data Transforms for automatic creation and prepopulation of Case participants
- Preventing validation errors in cascading drop-down lists
- Adding a Validation Rule to a Flow Action

Configuring a Data Transform as a pre/post processing for a flow action

Add a Data Transform directly to a flow action by using the Configure View feature in the Case Lifecycle window in App Studio.

Use the pre-processing Data Transform to populate fields in the user form with default values. For example, in a business trip Case you can configure the flow action to call a Data Transform to populate the event form with the most common purpose for a business trip.

Use the post-processing Data Transform to copy values from one page to another. For example, when the shipping address of the user is the same as the billing address, call a post-processing Data Transform to populate the shipping address page with the property values on the billing address page.

1. In the navigation pane of App Studio, click **Case Types**, and then click the Case Type that you want to open.
2. Click a step for which you want to configure a Data Transform, and then, in the **Step** properties pane, click **Configure view**.

©2025 Pegasystems Inc

3. On the **Pre/Post Processing** tab, in the **Pre processing Data Transform** list, select **Define new Data Transform**, and then follow steps 4 to 10 in Creating a Data Transform in App Studio.

The Pre/Post Processing tab of the Create dialog box

Related tasks

- Configuring pre-processing for a Flow Action
- Configuring post-processing for a Flow Action
- Populating Case data with a Data Transform
- Preventing validation errors in cascading drop-down lists

Decision tables

Decision tables derive values that have one of a few possible outcomes, where each outcome can result from a test of a condition that includes multiple variables. A decision table lists two or more rows, and each row contains test conditions, optional actions, and a result.Decision tables are appropriate for evaluations that include more elements compared to simple true/false evaluations. For example, you can use a decision table to estimate a car insurance payment based on the age of the driver, age of the car, and the make of the car.

You build decision tables by adding columns and rows. Each column represents a condition, while rows store corresponding values that the system tests at run time. For

©2025 Pegasystems Inc

example, you can build a decision table to determine shipment costs based on the following conditions:

- The value of the order
- The weight of the parcel
- Whether the shipment is to a foreign country

Each condition is a separate column in the decision table. Additionally, every decision table includes a column that stores results that the decision table returns if all values in a row evaluate to true.

Each row includes cells that store values that the system evaluates at run time. You can insert more than one value in a cell by adding an OR condition, and then the system continues processing a row if any of the values in the cell evaluate to true.

At run time, the system evaluates the rows starting at the topmost row:

- If any condition in a row evaluates to false, processing continues with the next row. The system ignores the **Return** column for that row.

  > **NOTE:** You can leave empty cells in a row. Empty cells evaluate to true except when a cell is empty and includes an OR condition. Then, the parser ignores that cell and parses only the cell that contains a value.

- If all the conditions in a row evaluate to true, the system processes the **Actions** and **Return** columns of that row. The next action depends on how you configure your decision table:
  - To return only one result, you can configure processing to end. The system then returns the value in the **Return** column as the value of the entire Rule.

    In scenarios that require only one result, such as estimating delivery costs, when you end processing after receiving the first result, you improve performance and avoid unnecessary computations.

©2025 Pegasystems Inc

- To return multiple results, you can configure processing to continue through all remaining rows, performing the Actions and Return calculations for any rows for which the conditions are all true.

  For example, when the decision table returns the email address of a person in a Process to approve an expense, the expense might require the approval of more than one person, based on the conditions. Such scenarios require more than one result.

- If all rows in the table evaluate to false, the system returns a default result.

For example:

In the following figure, the decision table calculates the shipment cost in an online shop. The cost depends on the value of the order, the weight of the parcel, and whether the shipment is to a foreign country. If the order value is less than $500, and the parcel weight is less than 3kg, and the shipment is to the same country, then the shipment cost is $20. If any of these values evaluate to false, the system moves to the next row. If all rows evaluate to false, the system returns the default value of $60 for the shipment cost:

	Conditions				Actions
	Order value <	Parcel weight in kilograms <	Shipment outside the country?		Return
if	500	3	False	→	20
else if	700	5	False	→	30
else if	1000	8	False	→	40
otherwise				→	60

Decision table

Applying decision tables

You can apply decision tables in the following elements of your application:

- In a flow Rule, you can reference a decision table in a Decision Shape .

©2025 Pegasystems Inc

For more information, see Adding Decisions to Processes.

- In an activity, you can evaluate a decision tree by using the *Property-Map-DecisionTable* method.

  For more information, see Creating an activity.

- A Declare Expression Rule can call a decision table.

  For more information, see Creating Declare Expression Rules.

- A collection Rule can call a decision table.

Category

Decision table Rules are instances of the *Rule-Declare-DecisionTable* class. Decision tables are part of the Decision category.

- Viewing Rule history

Creating decision tables

Better adjust to the varied factors in your business processes by creating a decision table. Decision tables test a series of property values to match conditions, so that your application performs a specific action under conditions that you define.

For example, you can define a decision table in your application to calculate a loan interest rate for a customer. The decision table evaluates the customer's credit score, type of account, and loan term, and then returns a result that matches values applicable to the customer.

1. Create a Rule to store the decision table:
   a. In the header of Dev Studio, click **Create** > **Decision** > **Decision Table**.
   b. In the **Label** field, enter a name that describes the purpose of the table.
   c. In the **Apply to** field, select the class in which you want to create the decision table.

©2025 Pegasystems Inc

d. Click **Create and open**.

For example:

The following figure shows a sample form to create a decision table:

Creating a decision table

2. **Optional:** To prepopulate the decision table with values, upload an .xls file that stores starting information:
   a. In the toolbar, click **Import**.
   b. In the **Upload file** dialog box, click **Choose file**.
   c. In the **Open** window, navigate to the file on your local machine, and then click **Open**.
   d. Click **Upload file**.

For example:

You can upload a decision table that stakeholders without access to Pega Platform™ prepare offline.

3. On the **Table** tab, in the **Conditions** column, click the header cell, as shown in the following figure:

©2025 Pegasystems Inc

An empty decision table

4. In the **Decision table property chooser** window, in the Property field, enter or select a property that you want to use as a condition.

   For example:

   Enter `pyCreditScore`.

5. In the Label field, enter the name of the property that you want to display in the header of the decision table.

   For example:

   Enter `Credit score`.

6. Select a comparison method:
   - To use a simple comparison, in the **Use operator** list, select the operator.

     For example, select >= to indicate that the value in the decision table must be equal to or greater than the value that the decision table evaluates at run time.

   - To specify a range for the condition property, select the **Use range** checkbox, and then define the start and end of a range.

©2025 Pegasystems Inc

For example, you can configure a property value to be greater than and lower than certain amounts.

7. Click **Save**.
8. **Optional:** To consider additional factors in a decision, add more condition properties:
   a. In the **Conditions** column, click a cell.
   b. In the toolbar, click the **Insert column after** icon.
   c. Define the condition property by repeating steps 3 through 7.

For example:

Add more columns to evaluate the customer account type and the loan term, as shown in the following figure:

	Conditions				Actions
	Credit score >=	Account type	Loan term		Return
if				→	
otherwise				→	

Decision table with three columns

9. In the **if** row, click the cell under a property, and then enter a value that the decision table evaluates at run time.
   If you configure two or more conditions, enter a value for at least one of the conditions. Your application ignores cells without values.
10. In the **Return** column, enter a return result.

For example:

You can configure a condition that if a credit score is greater than 500, the return result is a loan interest rate of 3%.

11. **Optional:** To insert an additional value in a cell, add an OR condition:

©2025 Pegasystems Inc

- To insert the OR condition before the current value, on the toolbar, click the **Insert OR before** icon.
- To insert the OR condition after the current value, on the toolbar, click the **Insert OR after** icon.

When an application evaluates the cell at run time, the application starts with the value in the top part of the cell.

For example:

The following figure shows a decision table that returns the rate interest for a loan. If a customer's credit score is equal to or greater than 500, the customer account type is Standard or Gold, the loan term is 12 months, and the rate interest is 3%.

	Conditions				Actions
	Credit score >=	Account type	Loan term in months		Return
if	500	Standard Gold	12	→	3%
otherwise				→	

Sample decision table

12. **Optional:** To remove a column from the table, select the column by clicking one of its cells, and then, in the toolbar, click **Delete column**.
    You cannot remove the **Conditions** and **Actions** columns.
13. **Optional:** To provide more outcomes, populate more rows with values:
    a. Click a cell in the **if** row.
    b. In the toolbar, click the **Insert row after** icon.
    c. Define the condition by repeating steps 3 through 10.

Result:

The newly added row displays the `else if` label. You can now insert rows before and after a row that you select.

©2025 Pegasystems Inc

14. **Optional:** To remove an **else if** row from the table, select the row by clicking one of its cells, and then in the toolbar, click **Delete row**.
    You cannot remove **if** and **otherwise** rows.
15. In the **otherwise** row, in the **Return** column, select or enter a property that defines an application behavior when no condition in the table returns a true value.

    For example:

    Configure your application to reject a Case.

16. **Optional:** To ensure that your application can process the table, check the table for conflicts by clicking **Show conflicts** on the toolbar.

    For example:

    If two rows are identical, the second row never evaluates to true and is unreachable.

    Result:

    A warning icon appears in rows that are unreachable or empty.

17. **Optional:** To increase the possibility of reaching a return value, improve the completeness of the table by clicking **Show completeness** on the toolbar.

    Result:

    The system automatically adds suggested rows to the decision table, that cover additional cases.

18. Click **Save**.

IMPORTANT: If you receive an error message that says that the code of a method exceeds the 65,535 bytes size limit, you must create the *DecisionTable/legacy/splitSizeLimit* dynamic system setting (DSS) in the

©2025 Pegasystems Inc

> *Pega-RULES* Ruleset, set its value to `true`, and then save the decision table again. This fix applies to decision tables for which you configure legacy Java generation instead of the standard Java generation.

19. **Optional:** To import the decision table to an .xls file, for example to share with stakeholders offline, in the toolbar, click **Export**.

Result:

At run time, your application processes rows in the table starting from the top row and returns a result for the row that is first to evaluate to true.

For example:

The following figure shows a decision table that calculates the interest rate for a loan. The decision table evaluates rows that include values that define the customer's credit score, account type, and loan term. The application returns the result of the first row that evaluates to true. If all rows evaluate to false, the application returns the result that the interest rate is 10%.

	Conditions				Actions
	Credit score >=	Account type	Loan term in months		Return
if	500	Standard Gold	12	→	3%
else if	300	Standard	12	→	5%
else if	150	Standard Basic	12 18	→	7%
otherwise				→	10%

Decision table to calculate interest rates

©2025 Pegasystems Inc

What to do next:

- Define additional options for the decision table. For example, decide whether an application continues evaluating the table after one row evaluates to true. For more information, see Configuring additional options for a decision table.
- To make the Decision table available for reuse in App Studio, for example, while authoring calculated fields, mark the decision table as relevant. For more information, see Marking a record as relevant.

- Calculating values with decision tables in App Studio
- Decision tables
- Viewing Rule history
- More about decision tables

Specifying pages and classes of a decision table Rule

To ensure that a decision table can access or update information on Clipboard pages, specify the names and classes of the pages. At run time, these pages contain the properties that a decision table references on other tabs of its Rule form.

Define the pages and classes of a Rule to:

- Set up Rule prompting during Rule configuration.
- Configure Rule validation.
- Apply default class for list elements.

For more information about pages and classes, see Defining the pages and classes of a Rule.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Decision Table**.
3. In the list of decision table instances, click the decision table that you want to edit.

©2025 Pegasystems Inc

4. On the Rule form, click the **Pages & Classes** tab.
5. In the **Page name** field, enter the name of the page that contains the property that this Rule references.
6. In the **Class** field, select the class of the page.
7. **Optional:** To add more pages and classes, click **Add item**, and then repeat steps 5 and 6.
8. Click **Save**.

For example:

The following figure shows a completed **Pages & Classes** tab of a decision table Rule:

Pages & Classes tab

- Creating decision tables
- Configuring additional options for a decision table
- Organizing Rules into classes

Configuring additional options for a decision table

Adjust how your decision table works by configuring additional options. For example, you can configure how a decision table behaves after one row evaluates to true, or define which elements of a decision table are available for edits.

©2025 Pegasystems Inc

Consider a scenario in which a user with a managerial role needs to update the results that the decision table returns. To avoid unintentional changes in the decision table, you can prevent users from manipulating columns and rows, and allow for editing only the results.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Decision Table**.
3. In the list of decision table instances, open the decision table that you want to edit.
4. On the **Results** tab, in the **Delegation options** section, configure the initial presentation and available options for the decision table:
   - To continue processing after one row in the decision table evaluates to true, select the **Evaluate all rows** checkbox.

     At run time, when a decision table evaluates all rows, the application performs the results from all rows that evaluate to true. If you leave this checkbox clear, the application performs the result from the first row that evaluates to true.

   - To disable row manipulation in the decision table, for example by adding or removing rows, clear the **Allowed to update row layout** checkbox.

     > **NOTE:** Users with Rule-editing privileges can update the cell values within an individual row, even if this checkbox is clear.

   - To disable column manipulation in the decision table, for example by adding or removing columns, clear the **Allowed to update column layout**.

     > **NOTE:** Users with Rule-editing privileges can update the cell values within an individual column even if this checkbox is clear.

©2025 Pegasystems Inc

- To prevent changes to the cell values, clear the **Allowed to change property sets** checkbox.
- To prevent access to the Expression Builder from any cell of the decision table, select the **Allowed to build Expressions**.

> **NOTE:** Users with Rule-editing privileges can add constants or property references in a row or column cell even if this checkbox is clear.

- To hide the **Result** column of the decision table to indicate that the decision table does not return an explicit value that represents the overall processing result, clear the **Allowed to return values** checkbox.

  If you select this checkbox, you can also restrict the list of possible return values. This checkbox is available only when you leave the **Evaluate all rows** checkbox clear.

5. **Optional:** To specify allowed results for the decision table, in the **Results** section, in the **Results defined by property** field, enter a property that lists the allowed results.

   > **For example:**
   >
   > In an online shopping application, you can select a property that lists countries to which the shop can send parcels, such as only countries within the European Union.

6. **Optional:** To match the allowed results that you provided in step 5 with specific values, map the results to properties:
   a. Expand the **Additional Allowed Results. When selected, each target property will be assigned the value specified** section.
   b. In the **Result** field, enter a value that the decision table can return at run time.

©2025 Pegasystems Inc

c. In the **Target property** field, enter a property that the system updates after the decision table returns the result that you specified in step 6.b.
d. In the **Value** field, enter a value to update the property that you specified in step 6.c.
e. **Optional:** To update more properties when the table returns a specific result, under the **Target property** field, click **Add item**, and then repeat steps 6.c and 6.d.
f. **Optional:** To match more results with properties, under the **Result** field, click **Add item**, and then repeat steps 6.b through 6.d.

For example:

If the decision table returns countries within the European Union to which an online shop can send parcels, you can define different shipping costs for different countries. When the decision table returns one of the countries as a result, the system automatically returns a property that specifies the shipping costs for that country. The following figure shows the configured **Results** section:

©2025 Pegasystems Inc

Defining allowed results

7. **Optional:** To set properties before the system starts processing the decision table, define properties to precalculate:
   a. Expand the **Preset Properties** section.
   b. In the **Property** field, enter a property that you want to set before processing of the decision table.
   c. In the text field, provide a value for the property by entering a constant, property name, Expression, or input parameter.
   d. To set more properties, click **Add a row**, and then repeat steps 7.b and 7.c.

> **For example:**
>
> You can enter an Expression that at run time automatically populates a property that holds a current date.

8. Click **Save**.

©2025 Pegasystems Inc

- Decision tables
- Creating decision tables
- Viewing Rule history
- More about decision tables
- Building Expressions with the Expression Builder

Calculating values with decision tables in App Studio

Respond to dynamically changing business situations by calculating values in your business processes automatically. With decision tables, you can define conditions that include multiple elements and result in different outcomes. Consequently, you deliver an application that helps users successfully resolve business processes in various and dynamic scenarios.

For example, you can create a decision table that calculates the loan rate based on the monthly income of a customer. You can add additional variables to the calculation, such as the loan or mortgage history of the customer. To ensure that a business process can reach its goal even in an unexpected scenario, you define a value that the decision table returns if all of the calculations evaluate to false.

Before you begin:

- Ensure that your application uses Constellation. For more information, see Application UI setup.
- Create fields that you want to compare at run time in your decision table. For more information, see Creating fields for capturing data.
- Create a calculated field for which you want to return results with a decision table. For more information, see Creating a calculated field.

When you create a decision table, you select a field that you want to compare against specific values, and then you define a method of comparison. You can compare one field against multiple values to find the result that best fits your business needs. Each

PEGA ©2025 Pegasystems Inc

column includes a field to evaluate, and each row includes values that an application uses to compare against the field. At run time, the application evaluates a decision table from top to bottom, starting with the row at the top of the table.

1. Navigate to the place in your application where you want to use a decision table.

For example:

You can use a decision table while creating a calculated field in the Data Model of a Case Type, as shown in the following figure:

Creating a calculated field that uses a decision table

©2025 Pegasystems Inc

2. In the **Evaluate** dialog box for a calculated field, click **Add column**.
3. In the **Column** list, select the field that you want to evaluate at run time.

   > **For example:**
   >
   > To include the monthly income of a customer in a calculation that determines the loan rate for the customer, select **Monthly income**.

4. In the **Comparator** list, select a comparator to use for evaluation.
   The list of comparators depends on the type of the calculated field that you want to edit.

   > **For example:**
   >
   > For a calculated field of the type integer, select **Between > and <** to indicate that a field in the **Column** list needs to be in a range of values at run time. In a sample scenario, the monthly income of a customer must be between 2,000 and 4,000 dollars.

5. **Optional:** To add more fields to compare at run time, repeat steps 2 through 4.

   > **For example:**
   >
   > Apart from the monthly income of a customer, you can add columns to check whether the customer has loans or mortgages.

6. Click **Submit**.
7. Click a cell in the column that you want to edit, and then enter a value to compare against the field at run time.

   > **For example:**
   >
   > In the **Monthly income** column, enter `2000` and `4000` to indicate that the monthly income of a customer must be between 2,000 and 4,000 dollars.

   If you have multiple columns, provide values for every column.

PEGA ©2025 Pegasystems Inc

8. In the **Return** column, click a cell, and then enter a value that the decision table returns if the condition in the row that you want to edit evaluates to true.

   > **For example:**
   >
   > To return a 4.25% loan rate for customers whose monthly income is between 2,000 and 4,000 dollars, in the **Return** column, in the row with the values of `2000` and `4000`, enter `4.25%`.

9. **Optional:** To provide more values against the field to evaluate at run time, click the **Add** icon, and then repeat steps 7 through 8.

   > **For example:**
   >
   > You can provide more values of monthly income and then associate them with different loan rates.

10. In the **Otherwise** row, in the **Return** column, click the cell, and then enter a value that the decision table returns if none of the rows evaluates to true.
11. **Optional:** To reuse the configured logic in the future, save the decision table:
    a. Above the header of the decision table, click **More options**.
    b. **Optional:** To provide a custom name for the decision table, in the **Add to library for reuse** dialog box, in the **Name** field, enter the label for the decision table.
       By default, the system uses `Evaluate` *[calculated field name]* as the name for a decision table.
    c. Click **Submit**.
12. Close the dialog box with decision table configurations by clicking **Submit**.

> **For example:**
>
> The following figure shows a decision table that calculates a loan rate based on a customer monthly income. The decision table also checks whether the customer has any loans or mortgages. If the monthly income fits any of the provided

PEGA ©2025 Pegasystems Inc

ranges, and if the customer does not have any loans or mortgages, the decision table returns an appropriate account type. If all the rows evaluate to false, the decision table returns a 5% loan rate.

Decision table that returns an account type

The following video shows how to create a decision table in App Studio while defining a calculated field:

Related concepts

- Building logic and calculating values in your application
- Decision tables

Related tasks

- Extending a Data Model for a Case

More about decision tables

When you learn more about decision tables, you can better understand how to use decision tables in a way that most efficiently meets your business scenarios. For example, you can understand how decision tables are related to selected methods and activities, and how these connections affect processing at run time.

©2025 Pegasystems Inc

Evaluating a decision table

In an activity, to evaluate a decision table and derive a value, your application can use the following objects:

- The *Property-Map-DecisionTable* method
- The standard *DecisionTable.ObtainValue()* Function
- The *pxEvaluateDecisionTable* utilities library
- The standard *@baseclass.DecisionTableLookup* activity

For more information about calling objects from an activity, see Configuring steps in an activity.

Property-Map-DecisionTable method

To evaluate a decision table and save the result as the value of a property, in an activity, call the *Property-Map-DecisionTable* method. As parameters, enter the target property name and the name of the decision table. For more information, see Property-Map-DecisionTable method.

DecisionTable.ObtainValue standard Function

In an activity, you can call the standard *DecisionTable.ObtainValue* Function to evaluate a decision table. Use the following syntax:

```
Lib(Pega-RULES:DecisionTable).ObtainValue(tools, myStepPage, DecisionTableName)
```

For more information about calling objects from an activity, see Configuring steps in an activity.

pxEvaluateDecisionTable utility library

In an activity, you can call the *pxEvaluateDecisionTable* utility library to evaluate a decision table. Use the following syntax:

PEGA ©2025 Pegasystems Inc

```
Lib(Pega-RulesEngine:Utilities).pxEvaluateDecisionTable(PrimaryPageName, Decision
TableName)
```

For more information about calling objects from an activity, see Configuring steps in an activity.

Performance

Pega Platform does not limit the number of rows in a decision table. However, as a best practice to avoid slow performance when updating the form and also avoid the Java 64 KB code maximum, limit your decision tables to no more than 300 to 500 rows.

> ⚠ **IMPORTANT:** When you save a decision table, if you receive an error message that says that the code of a method exceeds the 65,535 bytes size limit, you must create the *DecisionTable/legacy/splitSizeLimit* dynamic system setting (DSS) in the *Pega-RULES* Ruleset, set its value to `true`, and then save the decision table again. This fix applies only to decision tables for which you configure legacy Java generation instead of the standard Java generation.

Special processing with Declare Expression calls

When you create a Declare Expression Rule, you can incorporate the result of a decision table by providing the result property of the decision table in the **Set Property To** field. Special processing occurs at run time when a property from the decision table is not present on the Clipboard. Such decision Rules fail with an error message, and the system returns the Otherwise value from the decision table instead.

Not declarative

Despite the class name, the *Rule-Declare-DecisionTable* Rule Type does not produce forward or backward chaining. Technically, decision tables are not a declarative Rule Type.

PEGA ©2025 Pegasystems Inc

- Decision tables
- Creating decision tables
- Viewing Rule history

Unit testing a decision table

You can test a decision table individually, before testing it in the context of the application that you are developing. Additionally, you can convert the test run to a Pega unit test Case. Unit testing a decision table involves specifying a test page for the Rule to use, providing sample data as the input, running the Rule, and examining the results.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **Decision Table**, and then select the decision table you want to test.
2. Click **Actions** > **Run**.
3. In the **Data Context** list, click the thread in which you want to run the Rule.
4. Select a method for creating the test page.
   - Select **Copy existing page** to copy values from a thread of an existing Clipboard Page to the main test page.
   - Select **Create or reset test page** to create a new test page or reset the values of an existing test page.
     - To apply a Data Transform to the values on the test page, click the link in the **Apply** field and then select a Data Transform.
     - To clear the Result pane, click **Reset Page**.
     - To switch the current context, select a thread in the **Data Context** list and then click **Switch Context**.
5. To display the test results, enter data in the Result panel and then click **Run again**.

> **Result:**
>
> The value that you enter and the result that is returned are the values that are used for the default decision result assertion that is generated when you convert this test to a test Case.

©2025 Pegasystems Inc

6. **Optional:** To view the pages that are generated by the unit test, click **Show Clipboard**. For more information, see Viewing Clipboard Pages created by unit testing a Rule.
7. To convert the test into a Pega unit test Case, click **Convert to Test**. For more information, see Creating unit test cases for Rules.
8. **Optional:** To view the row on the **Table** tab that produced the test result, click the **Result Decision Paths** link. If the **Evaluate All Rows** option on the **Results** tab is selected, all the rows that are true are highlighted.

- Debugging decision trees with the Tracer
- Unit testing individual Rules
- Decision tables
- Running a unit test Case
- Viewing test Case results
- Exporting a list of test Cases
- Creating decision tables
- Viewing Rule history
- More about decision tables

Debugging decision tables with the Tracer

If your decision table Rule does not give you the results that you expect, and you cannot determine the problem by running the Rule and examining the Clipboard Pages, run the Tracer tool. With the Tracer, you can watch each step in the evaluation of a decision table as it occurs.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**.
2. Select the Ruleset that contains the Rule to be traced.
   a. In the Tracer window, click **Settings**.
   b. In the **Event Types to Trace** section, select the **Decision Table** checkbox.
   c. Select the Ruleset that contains the Rule to be traced.
   d. Clear the other Rulesets to avoid memory usage from tracing events occurring in other Rulesets.

©2025 Pegasystems Inc

   e. Click **OK**.
3. Return to the main Portal and run the decision table.
4. Watch the Tracer output as the Rule runs.

- Unit testing a decision table

Decision trees

Decision trees evaluate a set of conditions to return the result. Decision trees evaluate values against different properties and test conditions, and a true result can lead to additional comparisons. For example, you can create a decision tree that evaluates different conditions during a recruitment process. If a condition evaluates to true, the decision tree evaluates the next condition until reaching the final result. If any condition evaluates to false, the recruitment process ends.

When you build a decision tree, you add rows that your application evaluates at run time, starting with the top row. If the row evaluates to true, the application performs one of the following actions that you specify:

- Returns a specified value
- Continues evaluation
- Returns the `otherwise` result and finishes the evaluation

If all rows evaluate to false, the application returns the `otherwise` result and finishes the evaluation.
Consider a scenario in which you use a decision tree to check whether a job candidate is eligible for a given job offer. If the candidate's location equals EMEA, the decision tree moves on to more specific conditions, such as job experience and proficiency in a foreign language. If any of the rows evaluates to false, the decision tree returns the result that the candidate is not eligible for the offer, as the following diagram shows:

©2025 Pegasystems Inc

Schema of a decision tree

The following figure shows the same decision tree defined in Dev Studio:

Decision tree in Dev Studio

©2025 Pegasystems Inc

Nested branches

For more complex scenarios that can result in more than two outcomes, you can define a decision tree with nested branches. For example, in a recruitment process in a sample organization, most candidates might not require a background check. However, for candidates for the legal department, a background check is required. Additionally, if the candidate might potentially work with sensitive data, a criminal record check becomes mandatory. The following figure shows a diagram visualizing this scenario:

Decision tree with nested branches

The following figure shows the same decision tree defined in Dev Studio:

©2025 Pegasystems Inc

Decision tree with nested branches

Where referenced

Rules of four other types can reference decision trees:

- In a flow Rule, you can reference a decision tree in a Decision Shape . For more information, see Adding Decisions to Processes.
- In an activity, you can evaluate a decision tree by using the *Property-Map-DecisionTree* method. For more information, see Configuring steps in an activity.
- A Declare Expression Rule can call a decision tree. For more information, see Defining Expressions in Declare Expression Rules.
- A collection Rule can call a decision tree. For more information, see About Collection Rules.

After you complete initial development and testing, you can delegate selected Rules to line managers or other non-developers. Consider which business changes might require Rule updates and whether delegation to a user or group of users is appropriate. For more details, see Delegating a Rule or Data Type.

Category

Decision tree Rules are instances of the *Rule-Declare-DecisionTree* class. They are part of the Decision category.

PEGA ©2025 Pegasystems Inc

- Viewing Rule history

Creating decision trees

Calculate a value from a set of properties or conditions where true comparisons can lead to additional comparisons, organized and displayed as a tree structure, by creating a decision tree. For example, you can create a condition that checks whether the location of a job candidate is equal to a specific city. If the condition is true, your application evaluates additional conditions, such as work experience and education.

A decision tree continues processing until reaching a value that you specify as the **return** value. To ensure that the decision tree always returns a result, you specify the **otherwise** row that the decision tree uses as a result if it doesn't reach the **return** value.

Creating the decision tree Rule

1. In the header of Dev Studio, click **Create** > **Decision** > **Decision Tree**.
2. In the **Label** field, enter a name that describes the purpose of the decision tree.

   For example:

   Enter `IsCandidateEligible`.

3. In the **Apply to** field, select the class in which you want to create the decision tree.
4. Click **Create and open**.

   For example:

   The following figure shows a sample decision tree Rule form:

©2025 Pegasystems Inc

Creating a decision tree

Defining conditions in the decision tree

5. Click the default row to start defining conditions.
6. Define a condition and a result:

Choices	Actions
Define a single condition	a. In the first field, enter a property or a value. b. In the drop-down list, select a comparator. c. In the next field, enter a property or value that your application compares against the first property or value. d. In the **then** list, select **return**.

©2025 Pegasystems Inc

Choices	Actions
	e. In the last field, enter a property or value result that you want your application to return. **For example:** If you want a reporting manager to review any job application from candidates with > than 10 years of job experience, you can create the following condition and result: `if .JobExperience > 10 then return Reporting Manager`, as the following figure shows: *Defining a single condition* **Result:** When a decision tree reaches the **return** value, the tree stops processing, taking the **return** value as the result.

©2025 Pegasystems Inc

Choices	Actions
Define nested conditions	a. In the first field, enter a property or a value. b. In the drop-down list, select a comparator. c. In the next field, enter a property or value that your application compares against the first property or value. d. In the **then** list, select **continue**. e. Define a nested condition by providing a property or value, a comparator, and a result. **For example:** If the job experience of a job candidate is greater than 10 years, then your application checks whether the candidate has a master's degree, as shown in the following figure: *Nested condition in a decision tree*
Evaluate a property	a. In the first field, enter a property or a value.

©2025 Pegasystems Inc

Choices	Actions
NOTE: To select this option, on the Configuration **tab select the** Allow selection of 'evaluate property' option **checkbox.**	b. In the drop-down list, select a comparator. c. In the next field, enter a property or value that your application compares against the first property or value. d. In the **then** list, select **evaluate**. e. In the text field, enter a property that you want to use as the context for nested comparisons below the current row.
Call another declarative Rule **NOTE: In order to select this option, on the** Configuration **tab, select the** Allow selection of 'call decision' option **checkbox.**	a. In the first field, enter a property or a value. b. In the drop-down list, select a comparator. c. In the next field, enter a property or value that your application compares against the first property or value. d. In the **then** list, select the Rule type that you want to use. You can choose from decision tables, decision trees, and map values. e. In the text field, enter the name of the Rule that you want to use. You can press the Down Arrow key to view the list of available Rules.

©2025 Pegasystems Inc

Choices	Actions
	Result: The system evaluates the declarative Rule that you provide. The result of the Rule becomes the result of this decision tree, and evaluation ends. At run time, if this decision tree evaluates in a backward-chaining context (the *AllowMissingProperties* parameter to the method is true), the system evaluates the called Rule in the same way.

7. **Optional:** To create complex conditions, click **Add row**, and then repeat step 6.
8. In the **otherwise** section, define the behavior of your application if all of the conditions evaluate as false:

Choices	Actions
Return a value	a. In the `otherwise` row, select **Return**. b. In the **Default return value** field, enter a value that you want to use.

©2025 Pegasystems Inc

Choices	Actions
	For example: The following figure shows a decision tree that returns the `NotEligible` value if all of its rows evaluate to false: *Decision tree to evaluate job candidates*
Perform an action **NOTE: To select this option, on the** Configuration **tab select the** Allow selection of additional return actions **checkbox. The default action is setting a property value.**	a. In the `otherwise` row, click Take action. b. Click Actions. c. Click Add a row. d. Define an action by setting a value for the action property.
Return a value and perform an action **NOTE: In order to select this option, on the** Configuration **tab, select the** Allow selection of additional return actions	a. In the `otherwise` row, select Return. b. In the Default return value field, enter a value that you want to use. c. Click Take actions. d. Click Add a row.

©2025 Pegasystems Inc

Choices	Actions
checkbox. The default action is setting a property value.	e. Define an action by setting a value for the action property. **For example:** Change a Case Status by defining the following action: `Set pyUpdateCaseStatus equal to Resolved-Rejected`, as in the following figure: *Additional action in a decision tree*

9. **Optional:** To ensure that your application can process the tree, check the tree for conflicts by clicking **Show conflicts** on the toolbar.
   The presence of unreachable rows of the tree does not prevent you from saving the Rule. Comparisons involving two properties, such as *Width* and *Length* are ignored in this analysis. Conflicts are also checked when you save the Rule form, and when you use the Guardrails landing page for the application. Conflicts do not prevent the Rule from being validated or run, but could indicate that a Rule does not work as intended.

   For example:

   If one row checks whether the work experience is greater than 5 years, and the second row checks whether the work experience is greater than 3

©2025 Pegasystems Inc

years, the second row never evaluates to true because the first row includes the second row condition.

Result:

A warning icon appears in rows that are unreachable or empty:

Warning in a decision tree

10. **Optional:** To increase the possibility of reaching a return value, test for completeness of the tree by clicking **Show completeness** on the toolbar.

Result:

The system automatically adds suggested rows of the decision tree that cover additional cases:

©2025 Pegasystems Inc

Adding rows automatically

11. Click **Save**.

- Decision trees
- Rules management
- Viewing Rule history
- More about decision trees

Specifying pages and classes of a decision tree Rule

To ensure that a decision tree can access or update information on Clipboard pages, specify the names and classes of the pages. At run time, these pages contain the properties that a decision tree references on other tabs of its Rule form.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Decision Table**.
3. In the list of decision tree instances, click the decision tree that you want to edit.
4. On the Rule form, click the **Pages & Classes** tab.
5. In the **Page name** field, enter the name of the page that contains the property that this Rule references.

©2025 Pegasystems Inc

Decision tree Rules can apply to embedded pages that appear within top-level pages with various names. In such cases, you can use the keywords `Top` or `Parent` in the page name here.

6. In the **Class** field, select the class of the page.
7. **Optional:** To add more pages and classes, click **Add item**, and then repeat steps 5 and 6.
   Optionally, you can add a row with the keyword `Top` as the page name, to identify a top-level page. The `Top` keyword allows you to use the syntax `Top.propertyref` to identify properties on other tabs of this Rule form.
8. Click **Save**.

For example:

The following figure shows a completed **Pages & Classes** tab of a decision tree Rule:

Pages & Classes tab

- Decision trees
- Creating decision trees
- Viewing Rule history
- More about decision trees

©2025 Pegasystems Inc

Controlling output and actions for decision trees

To provide more accurate business logic in your application, restrict the possible values returned by a decision tree. You can also control the actions that other users can take while developing decision trees to ensure the accuracy of performed calculations.

The fields in this section impact the initial presentation and available options on the Decision tab of the decision tree.

For example, you can prevent users from calling specific Function aliases or adding new nodes to the tree structure. This helps you customize the development experience for delegated users, such as line managers, who may not require access to the full set of decision tree options.

All users, including delegated users, can remove these restrictions if they hold a Rule editing privilege.

Field	Description
Allow changes to Function lists	Select this checkbox to allow users to change the Function aliases called by each tree node on the Decision tab. Clear this checkbox to hide the Function alias picker on the Decision tab. Users with Rule editing privileges can still update the constant values in each tree node. With this option selected you can: • Populate the Functions Allowed list to restrict the Function aliases a user can select. Function aliases commonly used by managers include: *CompareTwoValues*,

©2025 Pegasystems Inc

Field	Description
	allEntriesSatisfyCondition, and *anyEntrySatisfiesCondition*. • Leave the Functions Allowed list empty to let users select any available Function alias. • Open any Function alias in the Functions Allowed list.
Allow adding of nodes to the decision tree	Select this checkbox to allow users to append and insert top-level tree nodes on the Decision tab. Clear this checkbox to hide the **add icon** on the Decision tab.
Allow selection of 'evaluate property' option	Select this checkbox to allow users to evaluate the value of a property from a tree node on the Decision tab. Clear this checkbox to hide the **evaluate** option from **then** drop-down list on the Decision tab. You must have the **Allow adding of nodes to the decision tree** option selected before you can change the state of this checkbox.
Allow selection of 'call decision' option	Select this checkbox to allow users to call a map value, decision tree, or decision table from a tree node on the Decision tab.

©2025 Pegasystems Inc

Field	Description
	Clear this checkbox to hide decision Rules from the list of available options in the **then** statement of the Decision tab. You must select the **Allow adding of nodes to the decision tree** option before you can change the state of this checkbox.
Allow selection of additional return actions	Select this checkbox to make the **Take Action** option visible on the Decision tab. Users can take action within each tree node or as part of the **otherwise** statement on the Decision tab. With this option selected, you can: • Populate the **Allowed Action Functions** list to restrict the Function aliases a user can call from an action. The *setPropertyValue* Function alias is commonly used by managers. • Leave the **Allowed Action Functions** list empty to let users call any available Function alias. • Open any Function alias in the **Allowed Action Functions** list.

Results

©2025 Pegasystems Inc

Use the options in this section of the tab to define the possible values this decision tree can return. You can also specify a list of preset properties that are calculated before the decision tree runs.

To define allowed results:

1. Enter a property or linked property name in the **Results defined by property** field.

   This property must use table validation because the table values are used to populate the **Result** field.

2. Select a value from the **Result** list.
   Alternatively, you can enter a string value without quotes to supplement the existing values from the property that uses table validation.
3. Define a list of **Target Property** and **Value** pairs that are set when the decision tree returns the corresponding **Result**.

   You can enter a constant, property name, or Expression in the **Value** fields.

4. Repeat steps 2 and 3 as necessary.

   At run time, the system sets target properties using the order you specify.

To define preset properties:

5. Enter a property name in the **Property** field.
6. Enter a constant, property name, Expression, or input parameter in the **Value** field.
7. Click the **add icon** and repeat this process for as many properties as are required.

   These properties are set before the Decision tab is processed.

- Decision trees
- Creating decision trees
- Viewing Rule history

©2025 Pegasystems Inc

- More about decision trees

More about decision trees

Decision trees test sets of values and conditions that can lead to other comparisons, enhancing business logic in your application. To take a full advantage of decision trees, explore additional information about them, such as how they can affect the performance of your application.

Overriding the property

Decision tree evaluation may be based on a known property identified on the **Configuration** tab, or on a parameter supplied in the *Property-Map-DecisionTree* method, or both.

If you leave the **Property** field blank after selecting the **Allow selection of 'evaluate property' option** checkbox on the **Configuration** tab, evaluation is always based on the parameter. If a parameter is supplied, the parameter value is used even when the **Property** field is not blank.

Standard Functions

As an alternative to the *Property-Map-DecisionTree* method, you can use the following standard Functions to evaluate a decision tree:

```
@(Pega-RULES:DecisionTree).ObtainValue(tools, myStepPage, decisiontree, inputpro
perty)
```

```
@(Pega-RULES:DecisionTree).ObtainValue(tools, myStepPage, decisiontree, inputpro
perty, bAllowMissingProperties)
```

Decision tree Rules can also be evaluated as part of a collection Rule (*Rule-Declare-Collection* Rule Type). For more information, see About Collection Rules.

©2025 Pegasystems Inc

Performance

The number of nodes in a decision tree is not limited. However, as a best practice, to avoid slow performance when updating the form and also avoid the Java 64 KB code maximum, limit your decision trees to no more than 300 to 500 rows.

Special processing with Declare Expression calls

When a Declare Expression Rule has the **Set Property To** field set to **Result of decision tree**, special processing occurs at run time when a property referenced in the decision table is not present on the Clipboard. Ordinarily, such Decision Rules fail with an error message; in this case the value provided in the **Otherwise** field of a decision tree is returned instead.

For more information, see Declare Expression Rules.

Declarative processing

Despite the class name, the *Rule-Declare-DecisionTable* Rule Type does not produce forward or backward chaining. Technically, decision trees are not declarative Rules.

For more information about forward and backward chaining, see Declare Expression Rules.

Custom Function aliases

You can use **Configuration** tab to enable custom Function aliases instead of the default comparison. For some of those aliases, you can click **Select values** and select from a list of values that are available for the selected property.

- Application debugging by using the Tracer tool
- Decision trees
- Creating decision trees
- Viewing Rule history

©2025 Pegasystems Inc

Configuring property evaluation in a decision tree

The run-time result of a decision tree can depend on the value of a property or the optional, third parameter of the Property-Map-DecisionTree method.

The following fields are visible when the **Allow selection of 'evaluate property' option** checkbox on the Configuration tab is selected. Use them to configure the property used by **evaluate** nodes in the decision tree.

Field	Description
Data Type	Choose `String`, `Number`, or `Boolean` to specify how the system evaluates the comparisons defined on the Decision tab when an optional parameter value is supplied in the Property-Map-DecisionTree method. The Data Type value you select affects comparisons on the Decision tab when the system obtains the input value as a method parameter. For example, if the method parameter is "007" and the Data Type is `String`, then a comparison of "007" < "7" is true. If the method parameter is "007" and the Data Type is `Number`, then the comparison of "007" < "7" is false. The Data Type is ignored when the property identified in the next field is used at run time. In that case, comparisons depend on the type of that property.

©2025 Pegasystems Inc

Field	Description
	This Data Type is independent of — and need not match — the type of the property to contain the decision tree result (the first parameter to the Property-Map-DecisionTree method). For example, you can evaluate comparisons of inputs based on numbers and return a result property of type `Text`.
Property	Optional. Enter a `Single Value` property reference, or a literal value between double quotes. (If your property reference doesn't identify a class, the system uses the Applies To portion of the key to this decision tree as the class of the property). at run time, if the value of the third parameter to the Property-Map-DecisionTree method is blank, the system uses the value of this property for comparisons.
Label	Optional. Enter a text label for the input property. This label appears on the Decision tab. Choose a meaningful label, as certain users may see and update only the Decision tab.

About Decision Trees

- Decision trees
- Creating decision trees
- Viewing Rule history
- More about decision trees

PEGA ©2025 Pegasystems Inc

Unit testing a decision tree

You can use the **Run Rule** feature to test a decision tree individually before testing it in the context of the application that you are developing.

You specify a test page for the Rule to use, provide sample data as the input, run the Rule, and examine the results. Additionally, you can convert the test run to a Pega unit test Case.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **Decision Tree**, and then select the decision tree you want to test.
2. Click **Actions** > **Run**.
3. In the **Data Context** list, click the thread in which you want to run the Rule.
4. Select a method for creating the test page.
   - Select **Copy existing page** to copy values from a thread of an existing Clipboard Page to the main test page.
   - Select **Create or reset test page** to create a new test page or reset the values of an existing test page.
     - To apply a Data Transform to the values on the test page, click the link in the **Apply** field and then select a Data Transform.
     - To clear the Result pane, click **Reset Page**.
     - To switch the current context, select a thread in the **Data Context** list and then click **Switch Context**.
5. To display the test results, enter data in the Result panel and then click **Run again**.

> **Result:**
>
> The value that you enter and the result that is returned are the values that are used for the default decision result assertion that is generated when you convert this test to a test Case.

©2025 Pegasystems Inc

6. **Optional:** To view the pages that are generated by the unit test, click **Show Clipboard**. For more information, see Clipboard pages created by the Run Rule feature.
7. To convert the test into a Pega unit test Case, click **Convert to Test**. For more information, see Creating unit test cases for Rules.
8. **Optional:** To view the row on the **Decision** tab that produced the test result, click the **Result Decision Paths** link.

- Unit testing individual Rules
- Decision trees
- Running a unit test Case
- Viewing test Case results
- Exporting a list of test Cases
- Creating decision trees
- Viewing Rule history
- More about decision trees

Debugging decision trees with the Tracer

If your decision tree does not give you the results you expect and you cannot determine the problem by running the Rule and examining the Clipboard Pages, run the Tracer tool. With the Tracer you can watch each step in the evaluation of a decision tree as it occurs.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**.
2. Select the Ruleset that contains the Rule to be traced.
   a. In the Tracer window, click **Settings**.
   b. In the **Event Types to Trace** section, select the **Decision Tree** checkbox.
   c. Select the Ruleset that contains the Rule to be traced.
   d. Clear the other Rulesets to avoid memory usage from tracing events occurring in other Rulesets.
   e. Click **OK**.
3. Return to the main Portal and run the decision tree.

©2025 Pegasystems Inc

4. Watch the Tracer output as the Rule runs.

- Unit testing a decision tree

Declare Expression Rules

Declare Expression Rules automatically calculate values based on other values that are available in your application. When you include Declare Expression Rules in calculations, you increase automation and efficiently reuse resources.

For example, you can use a Declare Expression Rule to automatically calculate a tax amount for an order in an online shop, based on the prices of items that the order includes.

Declare Expression Rules use two methods of calculating values: forward chaining and backward chaining. Starting with Pega Platform version 8.3, Pega Platform automatically applies the most suitable calculation method to each business scenario.

Forward chaining

Forward chaining occurs when the target property changes because the input values in the calculation change. Consider a scenario in which a customer creates an order in an online shop by selecting items that sum up to $60. A shopping application calculates the tax value that is 10% of the order amount and updates the target property with the value of $6. When the customer adds another item to the shopping cart, the application adds the price of the new item to the calculation. Consequently, input values change. The application performs forward chaining and updates the target value, which is the tax amount, based on the updated input values.

For example:

Consider a scenario in which customers provide width and length of a room to calculate the room area to order a correct number of floor tiles. Each time a

PEGA ©2025 Pegasystems Inc

customer changes the width or length of the room, the shopping application automatically updates the room area by performing forward chaining.

For example:

The following figure shows a shopping order for which an application calculates the tax amount based on the quantity and prices of ordered items. When a customer changes the quantity by adding two items, the total price changes, so the application performs forward chaining to calculate the updated tax amount.

Forward chaining

Backward chaining

Backward chaining occurs when an application seeks values of input parameters to calculate a target property. For example, in an online shopping application, when a customer provides a code to get a discount on specific items in a shopping order, the application seeks properties that hold prices of the selected items, and then updates the total order amount. In backward chaining, target values remain unchanged if input

©2025 Pegasystems Inc

values change, and the application performs calculation only after receiving a call for a target property.

Backward chaining directly corresponds to the *Property-Seek-Value* method. The *Property-Seek-Value* method can access the computational relationships among properties in a Declare Expression Rule to identify source values that are needed to compute a missing property value.

> **For example:**
>
> Consider a scenario in which a customer wants to fetch a prevailing foreign exchange rate and a tax rate at the time of quotation. The customer needs to access data from an external system. A declare expression that includes backward chaining fetches data from the external system instantly when the customer wants to access the foreign exchange rate and the tax rate.

Backward chaining is also applicable in the following specific scenarios:

- Some of the fields in the expression are from Data Pages that are outside of the current page structure. For example, consider a scenario in which you want to sum up values of multiple sales opportunities, but each opportunity is a separate and independent page.
- A declare expression invokes a Function that has no parameter, such as *calculateEmployeeCount()*. In such cases with no parameter, change tracking does not happen and the system applies backward chaining.

Users are no longer required to explicitly specify forward or backward chaining. Users can switch to legacy Expressions if they want to explicitly specify a change tracking method.

Category

Declare Expression Rules are instances of the *Rule-Declare-Expressions* Rule Types that are part of the Decision category.

PEGA ©2025 Pegasystems Inc

> **NOTE:** Do not confuse Declare Expression Rules with simple Expressions. Expressions — which contain syntax that includes constants, Function calls, operators, and property references — are often applicable in addition to Declare Expression Rules.

- Expressions
- Viewing Rule history
- Property-Seek-Value method

Creating Declare Expression Rules

Create Declare Expression Rules to automatically calculate values in your application. Declare Expression Rules take advantage of values that exist in your system to calculate values of target properties. As a result, you increase automation of calculations in your application.For example, in an online shopping order, an application can automatically calculate the tax value based on prices of the items and the information that tax equals a specific percentage of the item price. By implementing Declare Expression Rules, the application recalculates the tax value every time the customer updates the shopping order.

1. In the header of Dev Studio, click **Create** > **Decision** > **Declare Expression**.
2. In the **Target Property** field, enter a property that you want to calculate, preceding the property name with a period.

   > **For example:**
   >
   > For the *TaxValue* property, enter `.TaxValue`.

3. **Optional:** To restrict search for the result of the declare expression to a specific page, in the **Page Context** field, enter a Data Page that stores available results.

©2025 Pegasystems Inc

For example:

To situate a target property on the `.pyItemNames()` embedded page, enter `.Invoices.pyOrders().pyItems().pyItemNames()`. Leave the parentheses blank because a declare expression applies to all elements on a page.

For more information, see Data Pages overview.

4. In the **Context** section, select the Application Layer in which you want to store the Rule.
5. In the **Apply to** field, enter the class that stores the target property.

NOTE: When you create a declare expression, you can first provide the class, and then provide the target property in the **Target Property** field. If you select a class first, the system limits the available target properties to the properties that the selected class stores.

6. In the **Add to Ruleset** field, enter a Ruleset to store the Rule.
7. Click **Create and open**.

What to do next:

Define a declare expression. For more information, see Defining Expressions in Declare Expression Rules.

- Declare Expression Rules
- Viewing Rule history
- More about Declare Expression Rules

©2025 Pegasystems Inc

Defining Expressions in Declare Expression Rules

Define how your application calculates target properties by building Expressions. When you build Expressions, you create relations between properties so that an application can automatically update property values at run time. As a result, you increase the level of automation and flexibly respond to changing circumstances.

For example, you can create a Declare Expression Rule that automatically calculates shipping costs based on the total weight of all items that a customer orders in an online shop. When the customer adds items to the order and the total weight changes, a Declare Expression Rule automatically updates the shipping costs.

> **Before you begin:**
>
> Create a Declare Expression Rule. For more information, see Creating Declare Expression Rules.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Declare Expression**.
3. In the list of instances of declare expressions, open the instance that you want to edit.
4. **Optional:** To provide additional information about the Rule, on the **Expressions** tab, in the **Overview** section, enter a brief description:
   a. In the **Description** field, briefly explain the purpose of the Rule.

      > **For example:**
      >
      > Enter `This declare expression calculates the tax value that is 6% of the total price of all items in the shopping cart in an online shopping application.`

   b. In the **Usage** field, briefly explain the scenarios and elements of your application where the declare expression applies.

©2025 Pegasystems Inc

For example:

Enter `This declare expression calculates the tax value in the checkout page of a shopping application.`

TIP: Providing additional information about a declare expression is very useful to quickly share details of the Rule with other developers.

5. In the **Build Expressions** section, click **Add condition**.
6. In the **IF** field, enter or select a property to evaluate at run time.
   The list includes properties from the Apply To class that you selected during Rule creation. For more information, see Creating Declare Expression Rules.
7. In the list of comparators, select a comparator to apply during property evaluation.
8. In the text field, provide a property to evaluate against the property from the **IF** field.
9. **Optional:** To define more conditions, click **Add a row**, add a condition before or after the current condition, and then repeat steps 6 through 8.

   Result:

   By default, the system groups conditions by using the AND comparator. At run time, your application applies the result after all the conditions with the AND comparator evaluate to true.

10. **Optional:** If you have multiple rows, to apply the result of the Expression when any of the rows evaluates to true, click **AND**, and then click **OR**.

    For example:

    When the rows include price values of different items that a customer can add to an online shopping cart, and you select an **OR** comparator, the

©2025 Pegasystems Inc

> application applies the result of the Expression when the customer adds any of the products to the shopping cart.

11. In the **Then set** row for your target property, define how the declare expression determines the target property:
    - To determine the target property as a result of a string, in the list, select **Value of**, and then, in the text field, enter the string that calculates the property.
    - To determine the target property as a result of a decision tree, in the list, select **Result of Decision Tree**, and then, in the text field, enter a decision tree to compute at run time.
    - To determine the target property as a result of a decision table, in the list, select **Result of Decision Table**, and then, in the text field, enter a decision table to compute at run time.
    - To determine the target property as a result of a map value, in the list, select **Result of Map Value**, and then, in the text field, enter a map value to evaluate at run time.
12. **Optional:** To add more conditions to the Expression, click **Add condition**, and then repeat steps 6 through 11.

> **NOTE:** If you create multiple conditions, at run time, processing stops after your application determines the first Expression that evaluates to true.

13. In the **Set** row for the target property, provide that you want the system to apply if all the conditions evaluate to false:
    - To determine the target property as a result of a string, in the list, select **Value of**, and then, in the text field, enter the string that calculates the property.

©2025 Pegasystems Inc

- To determine the target property as a result of a decision tree, in the list, select **Result of Decision Tree**, and then, in the text field, enter a decision tree to compute at run time.
- To determine the target property as a result of a decision table, in the list, select **Result of Decision Table**, and then, in the text field, enter a decision table to compute at run time.
- To determine the target property as a result of a map value, in the list, select **Result of Map Value**, and then, in the text field, enter a map value to evaluate at run time.

14. Click **Save**.

For example:

The following figure shows a Declare Expression Rule that calculates a tax value in a shopping application when a customer reaches the `Checkout` page. Based on the prices of three types of items, the declare expression applies 6% of the total price as a tax. At run time, the declare expression updates the target property, which is a tax value, every time the user changes the quantity of items in a shopping order, because the total price changes as well. If the customer adds items with prices other than the prices that are included in the declare expression, the application calculates the tax value as 10% of the total price.

©2025 Pegasystems Inc

Sample declare expression

- Declare Expression Rules
- Creating Declare Expression Rules
- Viewing Rule history
- More about Declare Expression Rules

Specifying pages and classes for a declare expression

Define pages and classes that store properties to ensure that declare expressions can access these properties to correctly calculate values for your business scenario. As a result, you increase the level of automation in your application because automatic calculations can occur without user intervention.

Before you begin:

Create a declare expression. For more information, see Creating Declare Expression Rules.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Declare Expression**.

©2025 Pegasystems Inc

3. In the list of instances of declare expressions, click the Expression that you want to edit.
4. On the Rule form, click the **Pages & Classes** tab.
5. In the **Page name** field, enter the name of the page that contains the property that this Rule references.

> **NOTE:**
>
> To identify a top-level page, you can add a row with the keyword `Top` as the page name. With the `Top` keyword, you can use the syntax `Top.`*propertyref* to identify properties that you reference on other tabs of this Rule form.
>
> Declare Expression Rules can apply to embedded pages that appear within top-level pages with various names. In such cases, you can use the keywords `Top` or `Parent` in the page name. If the **Page Context** key part of this Rule is not blank, this tab automatically includes a definition of the `Top` and `Parent` keywords.

6. In the **Class** field, select the class of the page.
7. **Optional:** To add more pages and classes, click **Add item**, and then repeat steps 5 and 6.
8. Click **Save**.

- Defining the pages and classes of a Rule
- Declare Expression Rules
- Creating Declare Expression Rules
- Viewing Rule history
- More about Declare Expression Rules

©2025 Pegasystems Inc

More about Declare Expression Rules

This topic provides additional information about declare expression Rules.

Avoid direct updates to target properties

> ⚠ **CAUTION:** Do not change the value of a property computed by a Declare Expression Rule through the Property-Set method, application of a Data Transform, or user input into a form. As you develop your application, the Pega Platform attempts to detect and prohibit such situations.

For example, if someone creates a Declare Expression Rule that computes the property AverageWait, a developer tomorrow cannot save an activity that uses the Property-Set method to change the value of AverageWait.

However, if yesterday — before the Declare Expression Rule was defined — someone created and saved an activity that set the value of AverageWait, then after today both the activity and the Declare Expression Rule can execute to save the value. This practice is not recommended.

To reduce the risk of similar Rule conflicts, you can adopt a naming convention for properties designed to be computed in Declare Expression Rules, and create Declare Expression Rules early, using stub or dummy Expressions.

> ⚠ **CAUTION:** At run time, the system neither detects nor prevents the value of a target property from changing through user input, through the Property-Set method, or by Data Transforms. If the value changes through these or other direct means, the target property value can temporarily not equal the last computed Expression value. This situation is corrected the next time any input values for the Expression change.

PEGA ©2025 Pegasystems Inc

How this Rule runs with forward chaining

If you select `Whenever inputs change` in the Compute Values field, then each time the value of any property referenced in any Declare Expression Rule — or properties in other Rules (such as decision trees, decision tables, or map values) referenced in the Declare Expression Rule — changes, the system computes the values of the target property.

Ordinarily, code your activity to place all properties of interest on the Clipboard before the activity accesses the value of a property referenced in the key of a Declare Expression Rule. Your activity can create placeholder values (with Property-Set or with a Data Transform), or create the properties by opening instances with the Obj-Open method.

If the Declare Expression Rule contains a non-blank Page Context field, the Expression is evaluated at run time only when the Clipboard contains a page matching that full context.

When more than one Declare Expression Rule is to run, you cannot control or predetermine the order in which the multiple Declare Expression Rules run.

Forward chaining does not create new embedded pages. Declare expressions that have a non-blank **Page Context** field and that use `top` or `parent` keywords do not run unless at least one embedded page exists at each level of the page context.

When a property is the target of two Declare Expression Rules, one context-free and the second with a context, the Rule with the context runs, and the system ignores the context-free Rule. If two context-free Declare Expression Rules reference the same property by using distinct property reference forms, for example *workpage.targetproperty* and *.targetproperty*, the Rule with the longer reference runs, and the system igonres the Rule with the shorter reference.

PEGA ©2025 Pegasystems Inc

Sample Rules for backward chaining (goal seeking)

The Property-Seek-Value method uses Declare Expression Rules to compute a property value on request. For an example, the standard flow action named *Work-.VerifyProperty* calls the standard activity *Work-.VerifyProperty*. If you use this flow action in a flow Rule, a user can select it to cause the system to use goal-seeking to compute the value of the *pyResolutionCost* property.

If the backward chaining process fails, it can indicate a property with no current value that if set could aid in the computation. Your flow can then prompt a user for help or for a value that can allow the computation to complete.

Primary Page

During execution of a Declare Expression Rule, the page on which the Rule operates temporarily becomes the Primary Page. The page keyword `PRIMARY` and the results of the *tools.getPrimaryPage()* PublicAPI method reflect this change. When the Rule execution completes, the Primary Page of the calling activity resumes as primary.

Embedded properties

Declare expressions do not support displaying values of target properties in user interface if the target property is an embedded property and if the calculation engages forward chaining. Declare expressions always display property values if the calculation uses backward chaining. During forward chaining, the system might render the UI before populating the Clipboard, and the properties are not visible in the UI. To show updated values, define refresh conditions in the UI to get new values from the server when the values change. For more information, see Defining refresh conditions for UI areas.

Declarative Network Analysis gadget

In Dev Studio, click **Configure** > **Case Management** > **Business Rules** > **Declarative Network** to view and interact with the declarative expression Rules in your application.

©2025 Pegasystems Inc

Alias Function Rules

The contents of the selection lists on the Expressions tab depend on property alias Rules and alias Function Rules.

Testing and debugging Declare Expression Rules

Using the Tracer tool, you can watch the evaluation of a Declare Expression Rule. Start the Tracer tool and select a requestor session. Click **Settings** and check the Declare Expression box in the Event Types to Trace section. Also check the Ruleset that contains the Rule you want to trace.

The statistic Tracked Property Changes on the full details page of the Performance tool shows how many property changes have occurred (for the current requestor since log-in) that are tracked for declarative Rules computations.

Like other Rules, Declare Expression Rules won't evaluate as expected if the Rulesets needed for correct execution are not available to the requestor at run time. For an example, see the Pega Community article *Declarative Rules require access to correct input property Rules*.

Workstation display of calculated values using AJAX

When appropriate, your application can recompute the value of target properties (presented as read-only fields) immediately as a user changes an input value on a user form or flow action form, rather than later when the form is submitted. Users can see the new value immediately.

For example, the target property can represent an order total amount for a sales order. As a user enters and revises sales details, the total changes immediately as user focus leaves an input field.

> **NOTE:** This feature can improve user productivity and accuracy, while also reducing the number of server interactions and HTTP traffic required to

©2025 Pegasystems Inc

> complete a valid input form. Consider whether and where such interactive operation can simplify the user's task of completing complex input forms in your application.

To implement this capability:

1. Use this feature on flow actions or harnesses that use the SmartFrames layout and JSP tags.
2. Include at least one input to the Expression as another field or flow action. You can't place the input in the flow action form and the target in the harness form or vice versa.
3. Select the Enable Expression Calculation? box on the HTML tab of the Flow Action form or Harness form.
4. Test.

This feature uses AJAX between the browser client and the server.

Special processing for map value, decision table, and decision tree calls

When a Declare Expression Rule has `Result of decision table`, `Result of decision tree` or `Result of map value` for the Set Property To field, special processing occurs at run time when a property referenced in that Decision Rule is not present on the Clipboard. Ordinarily such decision Rules fail with an error message; in this case the system returns the Otherwise value instead. For details, see the Pega Community article *Troubleshooting: Declarative Expression does not execute when a Decision Rule provides no return value*.

Advanced debugging

Use the Tracer tool to detect that a Declare Expression Rule executes when expected. For more detailed debugging help, use logging categories.

For more information, see Creating custom log categories for debugging.

PEGA ©2025 Pegasystems Inc

OnChange Rules

Declare Expression Rules do not evaluate during the execution of an activity of type `OnChange`. Such executions are typically brief.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

- Application debugging by using the Tracer tool
- Declare Expression Rules
- Creating Declare Expression Rules
- Viewing Rule history

Declare Expression form - Completing the Change Tracking tab

Complete this tab to determine the conditions that cause automatic recomputation of the Expression. These conditions affect how often the computation on the **Expressions** tab is performed.

> **NOTE:** Declare expressions that are created as of Pega Platform™ 8.3 do not include the **Change tracking** tab. To use the legacy declare expression **Change tracking** tab, click **Actions** > **Use legacy Expression**.

> **NOTE:** Change tracking applies to properties on the Primary Page — the page that matches the Applies To key part of the Rule. Changes to properties on other pages referenced in the Expression or on the Pages & Classes tab are not tracked and do not trigger recomputation.

Additionally, the following operations that can change a property value do not cause change tracking to occur:

©2025 Pegasystems Inc

- Saving a Rule form that contains a property
- Retrieving a property with the Obj-List or Obj-Browse method
- Retrieving a property with the RDB-List method, unless the ApplyDeclaratives parameter of that method is selected

NOTE: Forward chaining operates only when the source properties are not marked as invalid. If a property has an associated message, forward chaining halts.

NOTE: Changes made to a property value within a Java step of an activity do cause backward changing computations to start, but do not cause forward chaining computation.

Field	Description
Target Property Data	Select to determine how often Pega Platform recomputes the value of the Target Property (identified in the second key part).
Calculate Value	Select an option to determine which events cause this Rule to run: **Whenever inputs change** (Default). Forward-chaining recomputation. The target property is not recomputed when the property is used as a source, only when one of the Expression inputs change. If the target property is not present on the page when needed or is present but has no value, the Declare Expression Rule does not run.

©2025 Pegasystems Inc

Field	Description
	When used if no value present Compute when the value is null, blank, zero, or does not yet appear on the page using backward chaining. Assuming the computation results are not null, later requests for the target property do not cause the Declare Expression Rule to run. **When used if property is missing** Recomputation using backward chaining when the target property is not present on the Clipboard. **Whenever used** Just-in-time recomputation even when the property already has a value, using backward chaining. **CAUTION:** This choice ensures that the target property value matches the computed value at the start of every activity step, by evaluating the Rule upon every read access of the target property. Because this choice can be costly in terms of performance, a warning icon appears when you save the Rule form.

©2025 Pegasystems Inc

Field	Description
	NOTE: During backward chaining computations, if the Otherwise section of a decision tree or decision table, or the Default row in a map value referenced in a Declare Expression Rule can be computed, but properties needed for other parts of the form are not defined, the Otherwise value is returned as the value of the Rule. **When applied by a Rule Collection** The target property is computed only when the defined Declare Expression Rule is invoked by a collection Rule. The chained Expressions also are evaluated if they too are set to "When applied by a Rule Collection." For example, if the defined Rule is TotalCostAfterTax=TotalCost*(1+TaxRate) where TotalCost=ItemCost*Quantity , then this Expression is run only when a collection references this Rule. In this case, TotalCost is also calculated if the properties on it were set to "When applied by a Rule Collection."

©2025 Pegasystems Inc

Field	Description
	This option supports self-referential Expressions, for example, `MyProperty=.MyProperty+1` . **NOTE:** This option is compatible with Pega Platform releases earlier than 7.1.6. **When invoked procedurally** The target property is computed only when an independent collection Rule directly invokes the target property. If the defined Declare Expression Rule is chained into another Expression that is set to "When invoked procedurally," the chained Expression will not run. For example, if the defined Rule is TotalCostAfterTax=TotalCost*(1+TaxRate) where TotalCost=ItemCost*Quantity , then this Expression is run only when a collection references this Rule. In this case, TotalCost is not calculated if the properties on it were set to "When invoked procedurally." This option supports self-referential Expressions, for example,

©2025 Pegasystems Inc

Field	Description
	NOTE: As a best practice, select `Whenever used` when the Expression involves values from properties on multiple pages.
Additional Dependencies	Optional. Enter property references to identify additional properties (in the same page context) to be tracked. Order is not significant. If not blank, changes to any of the properties cause this Rule to execute. This array appears only when you select `Whenever inputs change` for the Calculate Value field.
Context Execution Behavior	Select one of three settings to further define which situations cause this Expression to be evaluated: **Only when the top-level page is of the Applies to class** To execute this Rule only if the property is on a top-level page. **NOTE:** Do not select this option if the Applies To key part of this Rule identifies a class derived from the *Embed-*base class; by definition,

©2025 Pegasystems Inc

Field	Description
	pages of such classes are never at the top-level. **When the top-level page is of the Applies To class, or one of the following** To restrict execution to contexts in which the top-level page matches the Applies To class, one of a specified list, or any descendent classes in the class hierarchy. **Regardless of any pages it is embedded in** To execute the Rule regardless of the page context, making this Rule completely "context-free". **NOTE:** If you select this option, you cannot use the `Top` or `Parent` keywords in the computations on the Expressions tab, either explicitly or in decision Rules referenced on that tab. **CAUTION:** The third selection can produce high execution frequencies. Select the most restrictive value that meets your application requirements.

©2025 Pegasystems Inc

Field	Description
	NOTE: If this Pega Platform system was updated from Version 5.2 or earlier, execution of context-free Expressions is disabled by default. Enable them through changes to the `prconfig.xml` file or Dynamic System Settings. See More about Declare Expression Rules.
Execute this Expression	
Allowed top-level classes	This array appears only if you select the second choice (`When the top-level page..` .) for the Execute this Expression field. Enter a list of classes. Execution occurs if the top-level page has a class that exactly matches any class on this list, or is a descendent of one of the classes on this list. Do not list the Applies To key part here; that class is included automatically.

- Declare Expression Rules
- Creating Declare Expression Rules
- Viewing Rule history
- More about Declare Expression Rules

©2025 Pegasystems Inc

Supported and unsupported configurations in simplified declare expressions

As of Pega Platform™ 8.3, the **Change tracking** tab has been removed from the declare expression Rule form to simplify Expression configuration. In these simplified declare expressions, some **Target Property Data** and **Context Execution Behavior** configurations are not supported.

> **NOTE:** To access the **Change tracking** tab, click **Actions** > **Use legacy Expression**.

Supported configurations

In simplified declare expressions the target property's value is only calculated whenever used. Additionally, Expressions always run regardless of the context of the pages in which they are contained.

Unsupported configurations

The following **Target Property Data** field options are not supported for simplified declare expressions. For detailed information about these options, see Declare Expression form - Completing the Change Tracking tab.

- Whenever inputs change
- When used, if no value present
- When used, if property is missing
- When applied by a Rule Collection
- When invoked procedurally

The following **Context Execution Behavior** field options are not supported for simplified declare expressions. For detailed information about these options, see Declare Expression form - Completing the Change Tracking tab.

PEGA ©2025 Pegasystems Inc

- Only when the top-level page is of the Applies To class
- When the top-level page is of the Applies To class, or specified Allowed Top-Level Classes

Page context changes

In simplified declare expressions, context-bound Rules are not supported.

Non-atomic targets

Non-atomic targets are not supported in simplified declare expressions, for example, .page.targetProp.

Related concepts

- Declare Expression Rules

Related reference

- Declare Expression form - Completing the Change Tracking tab

Viewing the declarative network

Quickly gain insight into the relationships between declarative properties, which your system calculates automatically based on the value of other properties. By accessing the declarative network you can view and test declarative Rules, which reduces processing errors and development effort.

1. In the header of Dev Studio, click **Configure** > **Case Management** > **Business Rules** > **Declarative Network**.
2. On the **Declarative Network Analysis** tab, click **application**, and then select the application whose declarative network you want to review.

©2025 Pegasystems Inc

> **Result:**
>
> The system displays the declarative Rules that your application uses, divided by class.

3. Review the relationship between the declarative Rules by performing any of the following actions:
   - To display the Rule and related properties in a pop-up tree diagram, click the **Display this top-level declarative network** icon.
   - To open the declare expression for the Rule in a new tab, click the **Display declare expression** icon.
   - To open the target property of the declarative Rule in a new tab, click the property name.
   - To open the class to which a Rule belongs in a new tab, click the class name.

- Testing Rules in the Declarative Network Viewer

Testing Rules in the Declarative Network Viewer

Reduce the number of processing errors by testing calculated properties in the Declarative Network Viewer. By determining the value of a property automatically (declaratively), you reduce the number of manual calculations required and improve system efficiency. The viewer is a tool that displays the relationship between declarative properties based on changes to other property values. You can also use the viewer to test whether the properties calculate correctly, and set up tests for future use.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **Declare Expression**.
3. In the list, select the declare expression that you want to test.
4. In the declare expression, click **Actions** > **Run**.
5. In the **Test Page** section, select the context to use for the test:
   a. In the **Data Context** list, click the thread in which you want to run the Rule.

©2025 Pegasystems Inc

   b. **Optional:** To discard all previous test results and start from a blank test page, click **Reset Page**.
6. In the section with the declare expression that you want to test, define the value of the properties that the system uses to calculate the declarative property:
   a. Click a property that you want to define, and then enter its value in the **Property** field.
   b. Click **Update**.
   c. Repeat steps 6.a through 6.b for all properties in the declare expression.

   > **Result:**
   >
   > The value of the declare expression changes to reflect the updated properties. You can now see whether the declarative property calculates correctly.

7. **Optional:** To save the settings as a test scenario, click **Convert to test**, and then complete the **New** tab.
   For more information, see Creating unit test Cases for Rules.

- Viewing the declarative network

Declare OnChange Rules

Create Declare OnChange Rules to run an activity automatically at activity step boundaries when the value of a specified property changes. This capability provides a form of automatic forward chaining.

The following tabs are available on this form:

- OnChange Properties

For example, a Declare OnChange Rule can update a year-to-date counter property (an integer) to track how many times a work item status changed in value. Another Declare OnChange Rule can compute the average dollar amount of work items entered by a team, in real time.

©2025 Pegasystems Inc

Declare OnChange Rules can force all processing on a work item to be suspended pending an independent review of the situation. The independent review is supported by one or more flows. This feature can support compliance, fraud detection, and quality control staffs.

Where referenced

No other Rules explicitly reference Declare OnChange Rules. After you save a Declare OnChange Rule, it is run immediately and as needed.

Access

Use the Application Explorer to access OnChange Rules that apply to work types in your application. Use the Records Explorer to list all OnChange Rules available to you.

Category

Declare OnChange Rules are instances of the *Rule-Declare-OnChange* class. They are part of the Decision category.

- Policy overrides and suspended work items
- Viewing Rule history

Creating Declare OnChange Rules

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing Rules by creating a copy in a specific Ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Create a Declare OnChange Rule by selecting `Declare OnChange` from the `Decision` category.

Key parts:

©2025 Pegasystems Inc

A Declare OnChange Rule has two key parts:

Field	Description
Apply to	Select a class for this Rule. At run time, a Clipboard Page of this class must be a top-level page. The properties to be watched may be in this class (or in the class of an embedded page). You cannot use a Rule-Declare-* class or any ancestor of the Rule-Declare- class (including *@baseclass*) here. You cannot use a class derived from the *Code-* or *Embed-* class here.
Identifier	Enter a unique name for this OnChange Rule within the class. Begin the name with a letter and use only letters, numbers, the ampersand character, and hyphens. No other Rules explicitly reference this **Identifier** value. However, because of normal class inheritance, a Declare OnChange Rule named OutofStock at one level in the class structure may override (and so prevent execution of) a Declare OnChange Rule named OutofStock at a higher level in the class structure.

Rule resolution

- Declare OnChange Rules
- Viewing Rule history
- More about Declare OnChange Rules

PEGA ©2025 Pegasystems Inc

Declare OnChange form - Completing the OnChange Properties tab

Define the conditions that will cause this Rule to be executed. Specify in an array the properties to be watched for change.

Properties, Conditions, and Action

Field	Description
Properties to Watch	This array identifies some of the properties that the Rule tracks. Order is not significant. See More about Declare OnChange Rules for information about the properties tracked by the Rule.
(1, 2, 3...)	Enter the desired property references. Each property entered is monitored for changes. Identify a property on the top-level page (with a dot followed by the page name) or on a page identified by the **Page Context** field on the Pages & Classes tab. • If you list more than one property, they must be on the same page, either the top-level page or a common embedded page. • If you list more than one property, when two or more of the properties change value (for example, within a single step of an activity) this `OnChange` activity runs only once.

©2025 Pegasystems Inc

Field	Description
	• Start each property reference with a dot. You can't reference a property on a page other than the page corresponding to the Applies To key part of the Rule or on a page identified by the **Page Context** field on the Pages & Classes tab.
Conditions	

When	Optional: Identify a When Condition Rule to be evaluated at the time a property value changes.
	
	The system uses the Page Context value on the Pages & Classes tab (if not blank) as the Applies To key part of the When Condition Rule. If the Page Context value is blank, the system uses the Applies To key part of this OnChange Rule.
	
	Alternatively, you can enter a simple Boolean Expression here, for example in one of these formats:
	
	`property = "constant value"`
	`property1 > property2`
	
	You can reference linked properties in the Expression. Do not use a complex Java Expression or an Expression that calls a Function here.

©2025 Pegasystems Inc

Field	Description
	When this OnChange Rule used to suspend work item processing, this When Condition is known as the business exception.
Choose Action...	
Choose Action	Select: • `Suspend Work Object` — To cause all executing flows for a work item to be suspended. See Understanding policy overrides and suspended work items. • `Call Activity` — For all other OnChange processing.

When True Run and When False Run

Complete these fields when you select `Call Activity` in the Choose Action field.

Field	Description
When True Run	
Activity	Enter the Activity Name key part of an activity to execute when any of the specified properties change (and the test in the When field evaluates to true). Choose an activity with an Activity Type of `OnChange` (recorded on the Security tab). If the activity has input parameters, click **Params** to enter literal constant values for them. The system uses the Page Class value on the Pages & Classes tab (if not blank) as the

©2025 Pegasystems Inc

Field	Description
	Applies To key part of the activity. If the Page Context value is blank, the system uses the Applies To key part of this OnChange Rule. See More about Declare OnChange Rules for more about the activity.
When False Run	
When False Run	Optional. Enter the Activity Name key part of an activity to execute when any of the specified properties change and the test in the When field evaluates to false. Choose an activity with an Activity Type of `OnChange` (recorded on the Security tab). If the activity has input parameters, click **Params** to enter literal constant values for them. See More about Declare OnChange Rules for more about the activity.

Suspend Section

Complete these fields when you select `Suspend Work Object` in the Choose Action field. See Understanding policy overrides and suspended work items.

Field	Description
Suspend Section	
Policy Override Flow to run	Enter the Flow Name key part of a flow to execute to support review of the suspended flow. Ensure that the flow meets the requirements listed in How to implement

PEGA ©2025 Pegasystems Inc

Field	Description
	business exception processing with policy overrides.
Error message	Enter the third key part of a field value Rule with first key part *@baseclass* and second key part *pyMessageLabel*, or enter literal text between double quotes to identify this source of policy override reviews.

- Declare OnChange Rules
- Creating Declare OnChange Rules
- Viewing Rule history
- More about Declare OnChange Rules

Declare OnChange form - Completing the Pages & Classes tab

Use this tab to list the pages referenced by name in the OnChange tab. Specify the pages in an array containing these fields. See How to Complete a Pages & Classes tab for basic instructions.

1. About
2. New
3. OnChange Properties;
4. Pages & Classes
5. History
6. More...

If the watched properties are an aggregate or are on an embedded page, complete the Page Context Data fields.

PEGA ©2025 Pegasystems Inc

Field	Description
Pages and Classes	
Page Name	Optional. Name of a page referenced on the OnChange Properties tab. You can identify named pages here to support evaluation of a When Condition Rule referenced on the OnChange Properties tab. However, all tracked properties must be on the page corresponding to the Applies To key part, or a context within that page.
Class	Optional. Select the class of that page.
Page Context Data	
Page Context	Optional. Leave this blank if the monitored properties are `Single Value` properties that appear on a top-level page. Otherwise, identify a `Page List` or `Page Group` property reference on the page. You can supply literal index (subscript) values or property-reference index values. Omit any index to indicate that all values of the index are acceptable.
Page Class	Optional. Identify the class of the page or pages that are identified in the Page Context field. You cannot use the keywords `$ANY`, `$NONE` or `$CLASS` here.

About Declare OnChange Rules

©2025 Pegasystems Inc

- Declare OnChange Rules
- Creating Declare OnChange Rules
- Viewing Rule history
- More about Declare OnChange Rules

More about Declare OnChange Rules

Pega Platform tracks properties listed in Declare OnChange Rules for changed values. After each activity step, and after forward chaining for Declare Expression Rules and constraints Rules occurs, tracked changes start the activity specified on the OnChange Properties tab.

Properties tracked by the OnChange Rule

This Rule tracks the following properties:

- Properties identified in the Properties to Watch field
- Properties used in the When Condition Rule identified in the When field
- Properties used in the decision Rules referred by the When Condition Rule

Restrictions for OnChange activities

The activity identified on the OnChange Properties tab must have an Activity Type of `OnChange`. It can call or branch to other activities, but only if they too have an Activity Type of `OnChange`.

Your activity can examine the `Value List` property *pyChangedProperties* to discover the property that changed value. This property is on the page named *pyDeclarativeContext*, of class *Code-Pega-DeclarativeContext*. This page exists only while the activity runs.

The Primary Page passed to OnChange activities is the top-level page.

©2025 Pegasystems Inc

OnChange processing may start as the result of a user HTTP Post operation. If users complete an HTTP form that changes the value of a property tracked by a Declare OnChange processing, this change is detected.

Use care not to start an infinite processing loop within declarative Rules. For example, in an OnChange activity, do not update any of the properties that caused the activity to start. You can to update other properties in the OnChange activity.

- Declare Expression Rules do not evaluate during the execution of an `OnChange` activity. However, do not attempt in an OnChange activity to overtly set the value of a property that is the target of a Declare Expression Rule.
- Similarly, constraints Rules do not evaluate during the execution of an OnChange activity.

Primary Page

During execution of a Declare OnChange Rule, the page on which the Rule operates becomes the Primary Page. The page keyword `PRIMARY` and the results of the *tools.getPrimaryPage()* PublicAPI method reflect this change. When the Declare OnChange Rule execution completes, the Primary Page of the calling activity resumes as primary.

Performance

Design the OnChange activity to execute quickly, as some properties may change values often. Remember that properties may change values (and thus cause the Rule to run) during development and testing tasks, such as the preview of a harness or flow action form.

Declare OnChange Rules may be expensive (computing several values) compared with Declare Trigger Rules that test the same property. Consider which Rule Type better meets your needs. For example, if a value changes every few seconds but is part of an object that is saved only hourly on average, a Declare Trigger Rule computes a new value only hourly, not in real time.

PEGA ©2025 Pegasystems Inc

Testing and debugging Declare OnChange Rules

Using the Tracer, you can watch the evaluation of a Declare OnChange Rule. Start the Tracer and select a requestor session. Click **Settings** and check the Declare OnChange box in the Event Types to Trace section. Also check the Ruleset that contains the Rule to be traced.

The statistic Tracked Property Changes on the full details page of the Performance tool shows how many property changes have occurred (for the current requestor since log-in) that are tracked for declarative Rules computations.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

- Application debugging by using the Tracer tool
- Declare OnChange Rules
- Creating Declare OnChange Rules
- Viewing Rule history

How to implement business exception processing with policy overrides

Through OnChange Rules and special flows, your application can create application-wide temporary pauses in flow processing — for example, to detect and review exceptions, suspect data or special Cases — without altering the flows. This provides an alternative to use of tickets.

For a background on the benefits and purposes of this capability, see Policy overrides and suspended work items.

Configuration overview

This facility requires three components:

PEGA ©2025 Pegasystems Inc

- A property or properties present in a work item that can signify a business exception.
- An OnChange Rule that responds to a business exception by promptly suspending all Assignments on all currently executing flows for the work item.
- A flow that supports independent review of the work item, allowing the reviewer to record a final verdict.
- A reviewer — an operator holding the *Work-.ReviewPolicyOverrides* privilege — who assesses the business exception and records, in the work item history, a verdict allowing normal flow processing to resume or end.

1. Define the business exception as a When Condition Rule

   1. Identify the property or properties that are involved in the test.
   2. Create a When Condition Rule that is true only when the property or properties assume an unlikely, suspect, or potentially erroneous values. Choose a condition that is rare; that for most work items is false at all times from initial entry through resolution.

2. Create a policy override review flow

1. Create a flow Rule to support review of work items that become suspended when the business exception occurred. On the New dialog box, select *PolicyOverride* in the Template field. Design the flow to meet these criteria:

   - Applies to the class of the suspended work item (or a parent class).
   - Does not create a new work item.
   - Does not change the work item status.
   - Calls, as the last shape before the End shape, the standard flow *Work-.FinishPolicyOverride*.

This flow may be simple — containing only the Start, End Flow, and Subprocess shapes — or it may require multiple Assignments performed by multiple operators. Use the standard flow *Work-.PolicyOverride* as a starting point.

PEGA ©2025 Pegasystems Inc

In the SubProcess Properties panel, complete four flow parameters for the *Work-.FinishPolicyOverride* flow, to control the email Correspondence it produces. You can send Correspondence when the review results in a Deny verdict, and when it results in an Allow verdict.

3. Create an OnChange Rule

The OnChange Rule detects the business exception and starts the review flow.

1. Select the work type of the work items as the Applies To class.
2. Complete the OnChange Properties tab. In the When field, reference the When Condition Rule created in step 1 above.
3. In the Choose Action field, select `Suspend Work Object`.
4. Identify the review flow in the Policy Override flow to run field.
5. Enter text such as "Too many payees" or the key to a message Rule in the Error Message field, characterizing the reason for the suspension.

4. Adjust the routing

By default, the *Work-.FinishPolicyOverride* flow creates an Assignment on the Worklist of the operator administrator@org.com, where org.com is the organization owning the work item.

1. Override the router activity *Work-.ToPolicyOverrideOperator* to route this Assignment to a person (or one of a group) who is appropriate to act as a reviewer.

2. Ensure that the Operator ID of this person acquires the privilege *Work-.ReviewPolicyOverrides*. The standard role *PegaRULES:SysAdm4* conveys this privilege, but you can create other roles appropriate to your application and organization.

Use the Tracer tool to test such flows.

5. Ensure that reviewers complete their Assignments

©2025 Pegasystems Inc

Each person who acts as a reviewer must review and act on Assignments which have the status `Pending-PolicyOverride`. To verdict, they can choose `Allow` or `Deny` as the flow action.

If they choose `Allow`, processing resumes and any suspended Assignments reappear in Worklists or Work Queues. If they choose `Deny`, the work item status changes to `Resolved-Revoked`. Affected users may be notified by email, if configured in the flow. In the unusual case that two or more reviews are underway for one suspended work item, the status changes occur only after all the reviews are complete.

6. Periodically monitor outstanding flow errors

Suspended work items remain suspended until all open reviews are completed.

To see a report of suspended work items, in the header of Dev Studio, click **Configure** > **Case Management** > **Tools** > **Work Admin** > **Suspended Work in Current Work Pool**.

To see a list of Assignments in suspense, in the header of Dev Studio, click **Configure** > **Case Management** > **Tools** > **Work Admin** > **Suspended Work Assignments**.

NOTE:

- This capability illustrates how procedural Rules — such as the flows — can operate with declarative Rules — such as the OnChange Rules — to directly support business processing. Pegasystems identifies this distinctive combination as the Pro-Dex capability.
- The **Show Policy Override** button appears at run time on flows (created or updated in PRPC Version 4.2 SP6 and later) for users who have the *@baseclass.FlowViewProDex* privilege. This button does not appear on flow actions for which the Auto- generated HTML box on the HTML tab is not selected.

Process category

©2025 Pegasystems Inc

- Declare OnChange Rules

Declare Trigger Rules

To define correlations between events in your Case Types, create Declare Trigger Rules to cause an activity to run when a specified event takes place in a Case. By creating Declare Trigger Rules, you automate your business processes and flexibly respond to dynamic business needs.

Declare Trigger Rules are a part of the declarative Rules that implement the declarative programming paradigm in Pega Platform.

For each Declare Trigger Rule, Pega Platform monitors database operations for objects that you specify as the Applies To class, and concrete classes that are derived from that class, such as a class that stores your Case Type. When a change affects a specified property, Pega Platform uses the Change Tracking feature to monitor updates to the property values and run triggers accordingly.

For example, a Declare Trigger Rule can invoke an activity each time a user changes a postal code in their personal details. By changing their postal code, the user modifies a *Data-Party.pyPostalCode* property in an instance of the *Data-Party-Person* class. The activity that you specify in the Declare Trigger Rule sends an email message to the customer service representative (CSR) whose territory includes the new address. Similarly, a Declare Trigger Rule can implement a form of logging or audit history for a class by recording the date, time, and other facts. You can run Declare Trigger Rules in the following scenarios:

- Page Save
  - In context
  - In background
- Page Delete
- Page Commit

©2025 Pegasystems Inc

Nested Declare Trigger Rules

To resolve complex use cases, you can nest a Declare Trigger Rule within another Declare Trigger Rule. Nested Declare Trigger Rules work in the context of the other Declare Trigger Rule. For example, you can define a Declare Trigger Rule that sends an email to a CSR when a customer updates their address details. Then you can create and nest another Declare Trigger Rule that creates a document with the updated details after the Case moves to the next Stage. You can only nest Rules of the *Save* and *Save and* types that run immediately. Nesting Rules in the context of the *Commit* type Rules might result in adding an excessive number of deferred operations during one database transaction. To avoid recursion during nesting of declare triggers, you can only nest a trigger in a trigger that is not already running on a page in the execution stack. For example, if a Page A save leads to a save of Page B and a trigger on Page B again leads to a save of Page A, the declare trigger does not run again on the Page A save.

The following figure presents how nested Declare Trigger Rules start events in a Case:

Triggering events in a Case

©2025 Pegasystems Inc

To nest triggers, set the value of the *declaratives/nestedTriggersEnabled* dynamic system setting to true. For more information, see Creating a dynamic system setting.

Applying Declare Trigger Rules

Immediately after you save the Rule, your application runs a Declare Trigger Rule according to your provided configuration.

- Viewing Rule history

Creating Declare Trigger Rules

Define correlations between events in your Case Types by creating Declare Trigger Rules. Declare Trigger Rules run activities as a response to a specified event in a Case. As a result, you provide flexible applications that precisely meet your business needs.For example, you can define a Declare Trigger Rule that sends an email to a customer service representative (CSR) after a customer changes a postal address in a Case.

Declare Trigger Rules invoke activities when your application creates, updates, or deletes a class in the database. Operations on classes are related to actions in a Case, such as saving a Case or updating information in a Case. You can also use Declare Trigger Rules to track property changes, and for auditing purposes.

1. In the header of Dev Studio, click **Create** > **Decision** > **Declare Trigger**.
2. In the **Label** field, provide a descriptive name for the Rule that you want to create.
3. In the **Context** section, specify where your application stores the Rule:
   a. From the list of Application Layers, select an application to store the Rule.
   b. In the **Apply to** field, enter the class to which the Rule applies.
      At run time, the system monitors property values in this class and classes that inherit from this class.
   c. In the **Add to Ruleset** list, select a Ruleset and a Ruleset version in which to store the Rule.
4. Click **Create and open**.

©2025 Pegasystems Inc

Result:

The Rule form opens.

5. On the Triggers tab, in the Trigger when an instance is list, select a type of event to trigger the Rule:

Choices	Actions
Run the Rule when an application deletes an instance in the specified class	Select Deleted.
Run the Rule when an application saves an instance of the specified class	Select Saved.
Run the Rule when a Commit method occurs for a saved instance of the specified class	Select Committed Save.
Run the Rule when a Commit method occurs for a deleted instance of the specified class	Select Committed Delete.
Run the Rule when an application saves an instance of the specified class and specified property values in the class change	a. Select Saved and. b. In the **One of these properties was modified** section, in the Property field, enter a property that you want your application to track. **For example:** To track changes of the postal address, enter `Data-Party.pyPostalAddress`.

©2025 Pegasystems Inc

Choices	Actions
	c. **Optional:** To retain the initial value of the property that changes, in the **Copy Value To (optional)** field, enter a property that stores the initial value. Retaining the initial values is useful for auditing purposes and makes the previous property value available to the second and subsequent runs of the Rule. Copying occurs only if the trigger activity runs, and after the trigger activity completes. d. To track changes to more properties, click **Add a row**, and then repeat steps 5.b and 5.c.

> **NOTE:** The Declare Trigger Rule has other properties that are tracked under the following conditions:
> - When you enter the property in the **One of these properties was modified** section.
> - When you use an identified When Condition Rule in the When Field.
> - When you refer to the When Condition Rule in the Decision Rules.

6. To run the Rule only when a Case meets specified conditions, in the **Condition** section, in the **When** field, enter a When Condition Rule.
   The system evaluates the conditions at run time and runs the Declare Trigger Rule only when the When condition evaluates to true.

©2025 Pegasystems Inc

7. In the **Trigger activity** section, in the **Name** field, enter the activity that the Rule invokes.
8. In the **Execute** list, determine how the activity runs:
   - To run the activity during forward chaining, before the commit completes, select **Immediately**.
   - To run the activity in a new requestor, outside of the context of forward chaining, select **In Background On Copy**.

> **NOTE:** Ensure that the activity that you select to run in the background includes the Commit method.

When you select an activity of a different type than *Trigger*, ensure that the preconditions and transitions in the activity do not use When condition Rules and do not call Functions. For more information about activities, see Creating an activity.

> **CAUTION:** Although your application can contain multiple *Trigger* activities for the one class that is specified in the **Apply to** field, you cannot control the order in which two or more triggered activities run. Create activities that provide correct results when running in any order, and when running either independently or simultaneously.

9. Click **Save**.

- Declare Trigger Rules
- Viewing Rule history
- More about Declare Trigger Rules

©2025 Pegasystems Inc

Declare Trigger form - Completing the Pages and Classes tab

Use this tab to list the Clipboard Pages referenced by name in the **Triggers** tab. See How to Complete a Pages & Classes tab for basic instructions.

Field	Description
Pages and Classes	
Page Name	Optional. Enter the name of a page referenced on the Triggers tab. Optionally, add a row with the keyword `Top` as the page name, to identify a top-level page. The Top keyword allows you to use the syntax Top.propertyref to identify properties on other tabs of this Rule form.
Class	Optional. Select the class of the page or pages that are referenced on the Triggers tab. You cannot use the keywords `$ANY`, `$NONE` or `$CLASS` here. The system completes the first row of this array, using the Applies To key part of the Rule as the class, and leaving the Page Name field blank. Do not alter that row.
Page Context Data	
Page Context	Optional. Leave this blank if the properties watched by this Rule (and recorded on the Triggers tab) are `Single Value` properties found on a top-level page.

©2025 Pegasystems Inc

Field	Description

	Otherwise, identify a `Page List` or `Page Group` property reference on the page. You can supply literal index values or property-reference index values. Omit any index to indicate that all values of the index are acceptable. These are acceptable:
	`Invoices.pyOrders(2).pyItems("Manuals").pyItemNames`
	`Invoices.pyOrders().pyItems().pyItemNames`

Class	Optional. Select the class of the page or pages that are identified in the Page Context field. You cannot use the keywords `$ANY`, `$NONE` or `$CLASS` here.

About Declare Trigger Rules

- Declare Trigger Rules
- Creating Declare Trigger Rules
- Viewing Rule history
- More about Declare Trigger Rules

More about Declare Trigger Rules

Create Declare Trigger Rules to cause an activity to run when instances of a specific class are created, updated, or deleted in the database. This implements a form of forward chaining.

©2025 Pegasystems Inc

Trigger activities

The Primary Page passed to Trigger activities is the top-level page corresponding to the Applies To class of the Rule.

An activity of type `Trigger` may alter properties, call Functions and execute other Rules, but do not perform database commits. Take care in declarative processing not to specify processing that produces infinite looping.

When you choose `In Background on Copy` in the Execute field, the triggered activity runs in a child requestor in parallel to the current requestor. This means that:

- The triggered activity does not appear in your Tracer session
- The triggered activity cannot access Clipboard Pages other than the Primary Page
- Within the triggered activity, the Primary Page has no name

Primary Page

During execution of a Declare Trigger Rule, the page on which the Rule operates temporarily becomes the Primary Page. The page keyword `PRIMARY` and the results of the *tools.getPrimaryPage()* PublicAPI method reflect this change.

When the Declare Trigger Rule execution completes, the Primary Page of the calling activity resumes as primary.

Testing and debugging

Using the Tracer, you can watch the evaluation of a Declare Trigger Rule if the Execute field value is `Immediate`:

1. Start the Tracer and select a requestor session.
2. Click **Settings** and check the Declare Trigger box in the Event Types to Trace section.
3. Select the Ruleset that contains the Rule to be traced.

PEGA ©2025 Pegasystems Inc

History change auditing

Declare Trigger Rules can automatically update the history of a work item, Rule, or data object when certain properties change.

For work item change tracking, use the Field Auditing gadget, on the Work History landing page.

For data or Rule tracking, see Auditing field-level changes to security Rule and data instances.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

- Application debugging by using the Tracer tool
- Declare Trigger Rules
- Creating Declare Trigger Rules
- Viewing Rule history

Map values

Use a map value to create a table of number, text, or date ranges that converts one or two input values, such as latitude and longitude numbers, into a calculated result value, such as a city name. Map values greatly simplify decisions based on ranges of one or two inputs. Use a map value to record decisions based on one or two ranges of an input value. A map value uses a one- or two-dimensional table to derive a result.

Through cascading — where one map value calls another — map values can provide an output value based on three, four, or more inputs.

Complete the **Configuration** tab before the **Matrix** tab.

Where referenced

Rules of five types can reference map values:

©2025 Pegasystems Inc

- In a flow, you can reference a map value in a decision task, identified by the Decision shape in a flow.
- In an activity, you can evaluate a map value using the Property-Map-Value method or Property-Map-ValuePair method.
- A Declare Expression Rule can call a map value.
- A map value can call another map value.
- A collection Rule can call a map value.

Access

Use the Application Explorer to access the map values that apply to work types in your application. Use the Records Explorer to list all the map values that are available to you.

After you complete initial development and testing, you can delegate selected Rules to line managers or other non-developers. Consider which business changes might require Rule updates and if delegation to a user or group of users is appropriate. For more information, see Delegating a Rule or Data Type.

Category

Map values are part of the Decision category. A map value is an instance of the *Rule-Obj-MapValue* Rule Type.

Map values

Create a map value by selecting **Map Value** in the **Decision** category.

A map value has two key parts:

Field	Description
Apply to	Select a class that this map value applies to.

©2025 Pegasystems Inc

Field	Description
	The list of available class names depends on the Ruleset that you select. Each class can restrict applying Rules to an explicit set of Rulesets as specified on the **Advanced** tab of the class form. Map value Rules can apply to an embedded page. On the Map Value form, you can use the keywords `Top` and `Parent` in property references to navigate to pages above and outside the embedded page. If you use these keywords, include the class and absolute name — or a symbolic name using `Top` or `Parent` — on the **Pages & Classes** tab.
Identifier	Enter a name that is a valid Java identifier. Begin the name with a letter and use only letters, numbers, and hyphens. See How to enter a Java identifier.

When searching for instances of this Rule Type, the system uses full Rule resolution, which:

- Filters candidate Rules based on a requestor's Ruleset list of Rulesets and versions
- Searches through ancestor classes in the class hierarchy for candidates when no matching Rule is found in the starting class
- Finds circumstance-qualified Rules that override base Rules
- Finds time-qualified Rules that override base Rules

- Map values

Completing the Matrix tab

Map Value form - Completing the **Matrix** tab

This tab contains a table of one column (for a one-dimensional map value) or two or more columns (for a two-dimensional map value). The order of rows and columns is important. Rows are evaluated from left to right, and columns from top to bottom.

©2025 Pegasystems Inc

Complete the **Configuration** tab before updating the **Matrix** tab. Labels that you enter on the **Configuration** tab appear on the **Matrix** tab to guide your input.

To limit possible results to values in a fixed list of constant values, complete the **Configuration** tab before the **Matrix** tab.

Adding rows and columns

You can add new rows and columns by clicking **Configure rows** and **Configure columns**, respectively. You can also perform these actions from the **Configuration** tab.

Assessing completeness and consistency

Optionally, you can use these buttons to determine whether the map value is complete and consistent (based on a static evaluation).

Button	Results
Show Conflicts	Mark with a warning icon any cells of the matrix that are unreachable. For example, if two rows are identical, the second row can never evaluate to true and so cannot affect the outcome of the Rule. Click the warning icon on a row to highlight with an orange background the other cells that cause a cell to be unreachable. The selected row is highlighted with a yellow background. A map value that contains no such unreachable rows is called consistent. Conflicts are also checked when you save the form, and when you use the Guardrails

©2025 Pegasystems Inc

Button	Results
	landing page to run the guardrails check for the application. Conflicts do not prevent the Rule from validating or executing, but may indicate that the Rule does not implement the intended decision.
Import	After you export a map value, you can make changes in the .xlxs file and import the updated file. The map value Rule form is updated with the changes you made. You must import the same file that you exported. You can change the name of the exported file and import the renamed file. However, you cannot import a file different from the one you exported. The **Default** row and first rows are locked in the exported file. You cannot delete these rows, and you cannot insert rows when you select these rows. The **Default** column is locked in the exported file. You cannot delete this column, and you cannot insert columns when you select this column.
Export	Exports the map value in .xlxs format. After you make your changes and save this file, you can import it with your changes.

©2025 Pegasystems Inc

Button	Results
Show Completeness	Automatically add suggested rows that cover additional cases and reduce or eliminate the situations that fall through to the Default row . Suggested additions appear with a light green background. They are only suggestions; you can alter or eliminate them.

Entering row and column header

Each row and column has a header that defines both a label and a comparison. To create, review or update a row or column header:

1. Click **Configure rows** or **Configure columns**. A pop-up window appears.
2. Select a comparison operator: `<` `>` `=`, `>=`, `<=` or `is missing`. (If you omit an operator, the system assumes =.) Select `is missing` to detect that a property is not present — not that it is present but has the null value.
3. Enter an Expression. For each Expression, you can enter a literal value, a property reference or a more complex Expression, including Function calls.

The keyword `Default` always evaluates to true and appears as the final choice at the end of each row and column. You can complete values for the `Default` row or leave them blank.

Completing a cell

If you completed a list of literal constant values on the **Configuration** tab, select one of those values for each cell.

Otherwise, enter an Expression in the cell — a constant, a property reference, a Function call, or other Expression. For guidance while entering Expressions, click the **Expression Builder** to start the Expression Builder. (You can enter complex

©2025 Pegasystems Inc

Expressions and use the Expression Builder only if the **Allowed to Build Expressions?** checkbox is selected on the **Configuration** tab.)

If a cell is blank but is selected by the runtime evaluation, the system returns the null value as the value of the map value.

Cascading map values with Call

One map value cell can reference another map value as the source of its value. Type the word call followed by the name (the second key part) of another map value with the same first key part. SmartPrompt is available. Click the **Open** icon to open the other map value.

If, at run time, this map value executes in a backward-chaining mode (that is, the *AllowMissingProperties* parameter of the *Property-Map-Value* method is `True`), the called map value also runs in this mode.

- Map values
- Building Expressions with the Expression Builder

Completing the Configuration tab

Complete the fields on this tab to guide your inputs on the **Matrix** tab and define the possible values returned by this map value.

Security

The following options impact the initial presentation and available options on the Matrix tab.

For example, you can prevent users from accessing the Expression Builder or modifying the column layout of the map value. This helps you customize the development experience for delegated users, such as line managers, who may not require access to the full set of decision table options.

PEGA ©2025 Pegasystems Inc

> **NOTE:** All users with a Rule-editing privilege, including delegated users, can remove these restrictions.

Field	Description
Allow updating of the matrix configuration in delegated Rules	Select this checkbox to allow users to modify the rows and columns of the **Matrix** tab. Clear this checkbox to prevent users from updating row or column configuration. Users with Rule-editing privileges can still change values within the cells of the **Matrix** tab.
Allow use of the Expression Builder on the matrix view	Select this checkbox to allow access to the Expression Builder from any cell on the **Matrix** tab. Clear this checkbox to hide the Expression Builder icon. Users with Rule-editing privileges can still add constants or property references in a row or column cell.

Input Rows

See Configuring rows and columns in a map value.

Input Columns

See Configuring rows and columns in a map value.

Results

©2025 Pegasystems Inc

Use the options in this section of the tab to define the possible values that this map value can return. You can also specify a list of preset properties that are calculated before the map value runs.

To define allowed results:

1. Enter a property or linked property name in the **Results defined by property** field.

   This property must use table validation because the table values are used to populate the **Result** field.

2. Select a value from the **Result** list.

   Alternatively, you can enter a string value without quotes to supplement the existing table values.

3. Define a list of **Target Property** and **Value** pairs that are set when the map value returns the corresponding **Result**.

   You can enter a constant, property name, or Expression in the **Value** fields.

4. Repeat steps 2 and 3 as necessary.

At run time, the system sets target properties using the order you specify.

To define preset properties:

1. Enter a property name in the **Property** field.

2. Enter a constant, property name, Expression, or input parameter in the **Value** field.

3. Click the **add** icon, and then repeat this process for as many properties as required.

These properties are set before the rows and columns on the **Matrix** tab are processed.

PEGA ©2025 Pegasystems Inc

About Map Values

Completing the Pages & Classes tab

Identify what is known about the class of each page that is referenced on other tabs. See How to Complete a Pages & Classes tab for basic instructions.

Field	Description
Page Name	Optional. Enter the name of a Clipboard Page referenced on the Matrix or Configuration tab. Optionally, add a row with the keyword `Top` as the page name, to identify a top-level page. The Top keyword allows you to use the syntax Top.propertyref on other tabs of this Rule form to identify properties. Map value Rules can apply to embedded pages that appear within top-level pages with various names. In such cases, you can use the keywords `Top` or `Parent` in the page name here.
Class	Optional. Select the class of that page.

About Map Values

More about map values

Map values can be updated as needed to reflect changing business conditions, without the need to ask a skilled developer to modify complex activities or other Rules.

©2025 Pegasystems Inc

Uploading an Excel spreadsheet to start

If you have in advance an Excel spreadsheet in XLS file format that contains useful starting information for a map value, you can incorporate (or "harvest") the XLS file and the information it contains directly into the new Rule.

Evaluating

Both rows and columns contain a Type field (set on the Headers tab). The system makes comparisons according to the data type you recorded on the Headers tab, converting both the input and the conditions to the specified data type.

At run time, the system evaluates row conditions first from top to bottom, until one is found to be true. It then evaluates column conditions for that row, left to right, until one is found to be true. It returns the value computed from that matrix cell.

Where map values are used

You can reference map values in the following places:

- In activities that use the Property-Map-Value method or the Property-Map-ValuePair method. These methods evaluate a one-dimensional or two-dimensional map value, compute the result, and store the result as a value for a property.
- In other map values, through a `Call` keyword in a cell.
- Through a standard Function *ObtainValuePair()* in the *Pega-RULES:Map* library.
- On the Rules tab of a collection.

Input parameters in called map values

If a map value is evaluated through a Decision Shape on a flow, or one of the two methods noted above, the input value or values may be literal constants or may be property references, recorded in the flow or in the method parameters.

PEGA ©2025 Pegasystems Inc

However, if a map value is evaluated by a Call from a cell in another map value, the evaluation always uses the Input Property on the Header tab. Nothing in the Call can override this source.

Special processing with Declare Expression calls

When a Declare Expression Rule has `Result of map value` for the **Set Property To** field, special processing occurs at run time when a property referenced in the decision table is not present on the Clipboard. Ordinarily, such Decision Rules fail with an error message; in this case the default value is returned instead.

Performance

Pega Platform does not limit the number of nodes in a map value. However, as a best practice to avoid slow performance when updating the form and also avoid the Java 64KB code maximum, limit your map values to no more than 300 to 500 rows.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

Parent class

Through directed inheritance, the immediate parent of the *Rule-Obj-MapValue* class is the *Rule-Declare-* class. However, despite the class structure, this Rule Type does not produce forward or backward chaining. Technically, it is not a declarative Rule Type.

Standard Rules

The Pega Platform includes a few standard map values that you copy and modify. Use the Records Explorer to list all the map values available to you.

Applies To	Map Name	Purpose
Data-	*SetCorrPreference*	Selects a Correspondence type based on an input value. For example, if the input value is "Home Address", the

©2025 Pegasystems Inc

Applies To	Map Name	Purpose
		Correspondence type result is *Mail*. Allows outgoing Correspondence to be sent based to on available addresses, for a *Data-Party* object.
Work	*Officers*	Can associate an Operator ID or email addressee with officers based on the titles CEO, CFO, COO, and VP. (Cells are blank in the standard Rule.)

- Decision trees
- Decision tables

Unit testing a map value

You can test a map value individually, before testing it in the context of the application that you are developing. Additionally, you can convert the test run to a Pega unit test Case. Testing a map value involves specifying a test page for the Rule to use, providing sample values for required parameters, running the Rule, and then examining the test results.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **Map Value**, and then click the map value that you want to test.
2. Click **Actions** > **Run**.
3. In the **Test Page** pane, select the context and test page to use for the test:
   a. In the **Data Context** list, click the thread in which you want to run the Rule. If a test page exists for the thread, then it is listed and is used for creating the test page.
   b. To discard all previous test results and start from a blank test page, click **Reset Page**.
   c. To apply a Data Transform to the values on the test page, click the Data Transform link, and then select the Data Transform you want to use.

©2025 Pegasystems Inc

4. Enter sample values to use for required parameters in the **Results** pane and then click **Run Again**.

   > **Result:**
   >
   > The value that you enter and the result that is returned are the values that are used for the default decision result assertion that is generated when you convert this test to a test Case.

5. **Optional:** To view the pages that are generated by the unit test, click **Show Clipboard**.
6. To convert the test into a Pega unit test Case, click **Convert to Test**. For more information, see Creating unit test cases for Rules.
7. **Optional:** To view the row that produced the test result, click a **Result Decision Paths** link.

- Map values
- Unit testing individual Rules
- Using the Clipboard tool
- Opening a unit test Case
- Application debugging by using the Tracer tool

Configuring rows and columns in a map value

Complete the fields in the Input Rows and Input Columns sections of the **Configuration** tab to guide your inputs on the **Matrix** tab of a map value.

Evaluation of a map value can be based on the value of properties (specified here as the Row Property and Column Property), or on the value of parameters specified in a method.

If you leave the Property fields blank, the method must specify parameter values that match or are converted to the Data Type values on this tab.

PEGA ©2025 Pegasystems Inc

When the Property fields are not blank but the activity step used to evaluate the Rule specifies a parameter, the parameter value in the activity step is used, not the property value.

Input Rows

Field	Description
Row Parameter	
Data Type	Select `String`, `integer`, `double`, `Boolean`, `Date`, or `DateTime` to control how the system makes comparisons when a row parameter is supplied. It uses the Java `compareTo( )` method when comparing two dates or two strings. For example, if the method parameter is "007" and the Data Type is `String`, then a comparison of "007" < "7" is true. If the method parameter is "007" and the Data Type is `Number`, then the comparison of "007" < "7" is false. For Booleans, only the "=" comparison is available. The Data Type field is ignored (and becomes display-only on the form) when the Row Property property is the source of a value for the map value. Comparisons in that case depend on the type of that property.
Row Property	

©2025 Pegasystems Inc

Field	Description
Property	Optional. If this map value is to obtain the row input value from a property, select or enter a property reference or linked property reference. If you leave this blank, the calling method must supply a parameter value for the row. For a map value that is "called" by another map value, this field is required.
Label	Enter brief text that becomes a row name on the Matrix tab.

Input Columns

Select `none` as the Column Parameter Data Type when defining a one-dimensional map value.

Complete these optional fields to define a two-dimensional map value, which can be evaluated by the Property-Map-ValuePair method.

Field	Description
Column Parameter	
Data Type	Select `String`, `integer`, `double`, `Boolean`, `Date`, or `DateTime` to define a two-dimensional map value and to control how the system makes comparisons when a column parameter is supplied. It uses the Java `compareTo()` method when comparing two dates or two strings.

©2025 Pegasystems Inc

Field	Description
	To create a one-dimensional map value, select `none`. For Booleans, only the "=" comparison is available. The Data Type field is ignored (and becomes display-only on the form) when the Column Property property is the source of a value for the map value. Comparisons in that case depend on the type of that property.
Column Property	
Property	Optional. If this map value is to obtain a column input value from a property, select or enter a property reference or linked property reference. If you leave this blank but use a two-dimensional matrix, the calling method must supply a parameter value for the column. For a two-dimensional map value that is called by another map value, this field is required.
Label	Enter brief text that becomes a column name on the Matrix tab.

About Map Values

©2025 Pegasystems Inc

Validations for ensuring data integrity

While you collect input from Pega Platform™ users, validations help you ensure that the data is correct. Validation is critical for successfully processing Cases because it helps end users understand the requirements for their input.

Pega Platform provides the following types of validation:

- Validate Rule
- Edit Validate Rule
- Validation by using a Constraints Rule
- Validation through a Property Rule
- Validation at a Section Rule

The most important Rule of validation: keep validations out of the UI layers

One of the important best practices related to validation is to keep it out of the UI layers. It might seem easier to keep the validation at the control level, but it makes it harder in the future. Validations that are embedded in the UI layer do not work when a Case is processed through other Channels, such as APIs and conversational Channels. You can avoid this issue by keeping validations out of the UI layer.

Determine the correct type of validation to use

It is important to use the correct type of validation to use for a particular context. Determine what type of validation to use in your Pega Platform application by using the following guidelines:

-
    - If a validation is necessary for a field in every place that uses the field, use Field validation.
    - If you want to apply validations for a field that is calculated from other fields, use Constraints Rules.

©2025 Pegasystems Inc

- If a particular validation is needed only in a form while users enter data, use form validation.
- If you want to prevent the user from entering a Stage if some fields are not captured, use Stage validation.
- If you want to validate the same conditions each time while saving a Case, use validation on Case save.
- If you need to apply a validation before creating a Case, use validation on Case creation.

Field validation

If a validation for a field is applicable at every place in the application that uses the field, use field-level validation. Field-level validations are automatically added when you create fields of one of the following types in App Studio:

- Email
- URL
- Phone

If a user enters a value with an incorrect format, an error is displayed. The validation does not require any additional actions from the user. Additional restrictions for text fields can be added at the property Rule. For example, you can define the maximum number of characters that a user can enter in a field in a form. To add other field-level validations, you can create or reuse an Edit Validate Rule and associate it with the field Rule. Use Edit Validate Rules only for format limitations or value range limitations. Do not use Edit Validate Rules for comparing one field against another, because Edit Validate Rules are client-side validations and the other inputs might not be available when the Edit Validate Rule runs. For more information, see About Edit Validate rules.

Constraints Rules

Field-level validations are useful when an end user enters a value in a field. However, in a scenario where the value of a field is calculated from other fields in which a user enters the value, you can validate the field declaratively, whenever its value changes. If

©2025 Pegasystems Inc

you use a Constraints Rule, when the specified field value changes, the Constraints Rule checks whether the value still falls within the expected range. Do not use this type of validation for fields in which users enter values. For more information, see Constraints Rules.

Form validation

When you design a Case Type, ensure that a form meets all the required conditions before the form is submitted and the Case moves to the next step. By validating the values of various fields in the form, application developers can ensure that the end user enters the correct data at run time. For example, in a leave application form, the leave start date must be earlier the end date.

When you configure a form in App Studio, you define the validation that applies to the form, which includes defining a set of conditions and an error message. When the set of conditions returns true, the defined error message is displayed for the corresponding field after the user submits the form (Assignment). For more information, see Validating input values on a form.

When you create a validation in the view configuration window for an Assignment, in the background, App Studio creates a Validate Rule. The auto-generated Validate Rule is referred from the flow action of the Assignment. For advanced validation configurations that are not supported in App Studio, you can open the corresponding Validate Rule and update the Rule form.

For more information, see About Edit Validate rules.

> ⚠ **IMPORTANT:** Remember to keep validations out of the UI layer. Form validation provides the most flexibility because it is applicable for the Case through any Channel, including web, mobile, or API. However, if you use the option to mark a field as mandatory in the view configuration window in App Studio, the validation is restricted to the UI layer. When the Case is processed

©2025 Pegasystems Inc

through an API or other Channels that do not use this UI, the mandatory option is not applicable and can cause problems.

Stage validation

Moving to the next Stage relies on capturing data, receiving the required approvals, or the occurrence of a specific event. It is important that you define validations that notify the user when required conditions are not met before transitioning to a Stage.

For more information, see Defining the entrance criteria for a Stage.

NOTE: Internally, this model-driven Stage validation creates a *pyOnStageEntry* Validate Rule for the Case Type. For advanced Stage validation configurations that are not supported in App Studio, you can open the *pyOnStageEntry* Validate Rule and update the Rule accordingly.

Validation on Case creation

If you want to ensure that users complete some of the fields, or want to mandate relationships between some of the fields for a Case that is created, you must define validation on Case creation. For more information, see Configuring Case-wide validation for correct data entry in fields.

Validation on Case save

If you want to validate a condition whenever the Case is saved, you can use validation on Case creation or update. For more information, see Configuring Case-wide validation for correct data entry in fields.

Difference between validation on Case save and form validation

If you want to check a validation condition in all the Assignments across the Case Type, define it as Case validation. Otherwise, configure form-level (Assignment-level)

©2025 Pegasystems Inc

validations. Both types of validation can also coexist: you can define the common validation conditions in the Case save validation, and define the form-specific validation at the form validation.

Related tasks

- Validating input values on a form
- Defining the entrance criteria for a Stage

Related information

- https://docs.pega.com/bundle/platform-88/page/platform/case-management/validating-case-data.html

When Condition Rules

A When Condition Rule evaluates a Boolean logic statement that involves comparisons among field values, to return true or false. As a result, you deliver flexible software that adjusts to changing business circumstances.

For example, a When Condition Rule returns true if a job applicant has more than ten years of experience, and if the applicant speaks French fluently. As a result, an application performs a specific action, such as starting an optional Process in a Case.

You can apply When Condition Rules in multiple places in your application. To create more sophisticated logic, you can create multiple conditions, and then apply grouping that meets your business requirements.

Elements of When Condition Rules

When Condition Rules are Boolean Expressions that consist of three elements:

- An input value to compare at run time, for example, the value of a field that a user completes.
- A comparator, for example `is greater than` or `is equal to`.

©2025 Pegasystems Inc

- A value to compare against the input value from the condition. This value can be a constant that you enter or a reference to another field.

For example:

Consider a scenario in which an HR worker can start a background check of a job candidate only if the candidate applies for a managerial position. You can create a When Condition Rule `Position is equal to "Manager"`, where `Position` corresponds to a field with which a user interacts at run time. If the user selects `Manager` as the field input, the When Condition Rule evaluates to true. The following figure presents this When condition:

Sample When Condition

Input for When Condition Rules

As an input value for a When Condition Rule, you can use any of the following information:

- A field value
- A Configuration
- Another When condition

As you build a When condition by applying another When condition, you define whether the input When condition needs to evaluate to true or false at run time instead of selecting a comparator and a test value.

©2025 Pegasystems Inc

For example, you can create a When Condition Rule `CaseStatus is true`, where `CaseStatus` is another When condition defined as `CaseStatus is equal to "Pending-Review"`. In this scenario, `CaseStatus is true` evaluates to true if the Case Status at run time is `Pending-Review`. For more intuitive development experience, you can easily preview the When condition that you apply as an input value, as shown in the following figure:

When condition preview

Grouping conditions

You can group conditions by applying AND and OR conjunctions. When you apply AND conjunctions, all grouped conditions need to evaluate to true for a When Condition Rule to return true. For example, in an application to review job candidates, if you create the conditions `French skill is equal to "10"` and `Work experience is greater than or equal to "10"`, and you use an AND conjunction, both conditions need to evaluate to true for the When condition to return true. If only one condition evaluates to true, for example, if a candidate has more than ten years of work experience, but the French skill is below 10, overall the When condition evaluates to false.

If you group conditions by using an OR conjunction, any grouped condition needs to evaluate to true for the When Condition Rule to return true. For example, if you create the conditions `German skill is equal to "10"` and `French skill is equal to "10"` and you use an OR conjunction, the When condition evaluates to true if a candidate fluently speaks German or French.

©2025 Pegasystems Inc

You can mix comparators to meet your business requirements. For example, you can create a When Condition Rule that determines that a job candidate needs to have at least ten years of work experience and needs to be fluent in either French or German, as shown in the following figure:

Mixed conjunctions

To resolve complex business scenarios, you can use advanced logic to group conditions by entering a logic string. When you apply advanced logic, every condition has an integer assigned that you use in the logic string. In logic strings, you can subgroup conditions and apply nested levels of grouping. A sample logic string is `1 AND (2 OR (3 AND 4))`. In advanced logic, you can also use the `NOT` conjunction.

Referencing When Condition Rules

Rules of many other Rule Types can reference When Condition Rules. For example, you can reference When Condition Rules in activities, Processes, and through the `<pega:when>` JSP tag in HTML and XML Rules in your applications based on UI Kit.

For example, you can use a When Condition Rule to define when an optional Process starts in a Case. For more information, see Conditionally starting a Process.

Delegation

After you complete initial development and testing, you can delegate selected When Condition Rules to line managers or other non-developers. Consider which business

©2025 Pegasystems Inc

changes might require Rule updates and if Rule delegation to a user or group of users is appropriate.

For more information, see Delegating a Rule or Data Type.

Conditions in activity steps

Each activity step can reference one or several When Condition Rules, as preconditions for a method, or transitions between the completed method and the next step.

As a precondition, a When Condition Rule determines whether the method in the activity step runs. Referenced in a transition, a When condition determines which activity processing runs after the completion of the current activity step.

For more information, see Creating an activity and Configuring steps in an activity.

Standard Function

To run a When Condition Rule in an Expression or in the context of Java code, call the standard Function *callwhen()* in the Pega-RULES *Default* library.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

Category

When Condition Rules are part of the Decision category. A When Condition Rule is an instance of the *Rule-Obj-When* Rule Type.

pyWorkPage

The use of *pyWorkPage* inside a When condition is not supported, in particular when used in Embedded Data list of record fields.

- Performing unit testing on a When Rule

©2025 Pegasystems Inc

Creating a When Rule

Evaluate a Boolean logical statement that involves comparisons among values of properties, to return true or false, by creating a When rule.

For example, you can create a When rule to prompt a user to enter a delivery address only when the user specifies that the delivery address varies from the address of residence.

> **NOTE:** The use of *pyWorkPage* inside a When condition is not supported.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **When**.
3. On the **When** tab, click **Create**.
4. In the **Label** field, enter a description of the When rule.
   Create names of when condition records that logically follow the word when.

   > **For example:**
   >
   > Enter `IsTaskCompleted`.

5. **Optional:** To enter a when expression that evaluates whether a value is true or false, click **Show additional options**, and then enter the expression.

   > **For example:**
   >
   > `.BalanceRemaining > 0.0`

6. **Optional:** To enable editing the rule in the legacy mode, click **Show additional options**, and then select the **Use legacy authoring mode** checkbox.
   By default, you edit conditions in the condition builder. For more information, see Defining conditions for a When Rule.
7. In the **Context** section, select an application to which the rule applies.

©2025 Pegasystems Inc

8. In the **Apply to** field, select a class to which the rule applies.
9. In the **Add to ruleset** field, select a ruleset for the rule.
10. **Optional:** To enable traceability of the rule, in the **Current work item** section, enter the work item to associate with the rule, for example, a bug ID.
11. Click **Create and close**.

Defining conditions for a When Rule

Enter or revise a When Rule that can be expressed as a single Boolean Expression or the conjunction (AND or OR) of one or more Boolean Expressions.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **When**.
2. From the list of When instances, select a When Rule that you want to edit.
3. On the Rule form, click the **Conditions** tab.
4. In the first field, select a value or a property from the list.
   You can choose from Configurations, fields, and When conditions. If you use a When condition, you define whether the input When condition needs to evaluate to true or false at run time instead of selecting a comparator and a test value, as shown in the following figure:

When condition preview

5. In the second field, select a comparator from the list.
6. In the third field, enter a value that you want to compare with the value from the first field.
   - Enter the value.
   - Click **Select values**, and then select the value.

©2025 Pegasystems Inc

The system compares the value of this field to the value of the corresponding field or condition in the first column.

> **For example:**
>
> `Work status is equal to Done`

7. **Optional:** To treat empty values for *Date* and *DateTime* properties as properties with null value, click **Select values**, and then select the **Use null if empty** checkbox.

> **NOTE:** If you clear the checkbox, the application uses the current date as the value of an empty *Date* or *DateTime* property.

8. **Optional:** To specify additional conditions and define the relationships between them by performing the following actions:
   a. Click the **Add a row** icon and add as many conditions as you need.
   b. In the drop-down lists between each pair of conditions, specify how these conditions relate to each other.
      You can group conditions with **AND** or **OR** operators.
   c. In the drop-down list in the upper right corner, specify how you want to evaluate condition groupings
      - **Group ANDs** - Select this option if you want conditions linked with the **AND** operator to be evaluated as a group, and conditions linked with the **OR** operator to be evaluated individually. With this grouping, the grouping of condition 1 **AND** condition 2 **OR** condition 3 **OR** condition 4 is evaluated as (1 AND 2) OR 3 OR 4. That is, either conditions 1 and 2 must both be true, or either one of conditions 3 and 4 must be true.
      - **Group ORs** - Select this option if you want conditions linked with the **OR** operator to be evaluated as a group, and conditions linked with the **AND** operator to be evaluated individually. With this grouping, the grouping of condition 1 **AND** condition 2 **OR** condition 3 **OR** condition 4

©2025 Pegasystems Inc

is evaluated as 1 AND (2 OR 3 OR 4). That is, both condition 1 and one of conditions 2, 3, and 4 must be true.

- **Use advanced logic** - Select this option if you want to define a mix of grouping by the **AND** and **OR** operator. Use the **Logic string** field to specify the condition grouping.

> **NOTE:** If you change a condition that uses advanced logic back to **Group ANDs** or **Group ORs**, all the groupings that you previously defined are reset.

> **TIP:** Conditions that are evaluated as a group are displayed on a single block of gray background.

9. Click **Save**.

> **What to do next:**
>
> To make the created When Rule available for reuse in App Studio, for example, when defining conditionally processing Stages and Steps, mark the Rule as relevant. For more information, see Marking a record as relevant.

- Building Expressions with the Expression Builder
- Creating a When Rule
- Conditionally starting a Process
- Defining Conditions for skipping a Stage

Editing a When Rule in the legacy mode

Apply Functions and more advanced logic in When Rules by editing the conditions in the legacy authoring mode.

©2025 Pegasystems Inc

Before you begin:

Enable the legacy authoring mode for the Rule that you want to edit. For more information, see Creating a When Rule.

By default, when you create and edit When Rules, you use the Condition Builder view. For more information on how to define conditions, see Defining conditions for a When Rule.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **When**, and then open the Rule that you want to edit.
2. On the Rule form, double-click the link below the **When** node.
3. In the first field, enter a property, a literal constant, or a Function call.
4. In the second field, select a comparator.
5. In the third column, provide a value that you want to compare with the value from the first field:
   - Enter the value.
   - Click **Select values**, and then select the value.

   For example:

   `Status work is equal to Done`

6. **Optional:** To configure advanced conditions, press the Down arrow button, and then select a condition from the list.
7. Click **Submit**.
8. **Optional:** Develop the Rule by clicking **Actions**, and then performing one of the following procedures:
   - To update the condition, click **Edit**.
   - To add another condition, click **Insert condition**.
   - To create a group of nested conditions, click **Insert group**.
   - To delete the Expression node, click **Delete**.
9. Click **Save**.

©2025 Pegasystems Inc

Result:

The system displays the conditions and logical operators that you enter on the **Advanced** tab in the **Condition** array and the **Logic String** field.

- When Condition Rules

Configuring advanced options of a When Rule

Define complex logic strings and use Functions in conditions by configuring advanced options of a When Rule.

For example, if you create conditions 1, 2, 3, and 4, you can define an advanced logic string for the conditions, such as `1 AND (2 OR (3 AND 4))`.

NOTE: As a best practice, use the **Conditions** tab to configure and update the Rule. Your configuration populates the conditions array and the **Logic String** field on this tab. If you add, change, or delete rows, or edit the **Logic String** field, you can no longer use the **Conditions** tab. Updating values in conditions or using the **Options** area does not disable that tab.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **When**, and then open a Rule that you want to edit.
2. On the Rule form, click the **Advanced** tab.
3. In the drop-down list, select an Expression type.
4. Enter a label for the Expression, for example, A.
5. Complete an Expression.
6. If you want to add more Expressions, click **Add condition**, and then repeat steps 3 trough 5.
7. In the **Logic string** field, enter the Boolean logic operations performed on the Conditions array to evaluate if the condition is true or false at run time.

©2025 Pegasystems Inc

If you created the conditions on the **Conditions** tab, the system populates this field by using the AND and OR operators, in addition to the group hierarchies that are defined on the condition tree.

You can use `and`, `or`, and `not` in the statement. Use parentheses to control the order of evaluation.

For example:

```
(A and B) or (C and not D)
```

Result:

When you save the Rule, if the form contains only one test row, the system inserts the label for that test row in the **Logic string** field. If you have more than one row, the system requires all rows to be true. If you want a different outcome, revise and save the statement again.

8. **Optional:** In the **Advanced options** section, in the **Allowed Functions** field, select a Function alias that restricts the available conditions to the one in this field by clicking **Add Function**.
   If you add multiple Functions, the top entry is the default one.
9. Click **Save as**.

- When Condition Rules
- Creating a When Rule

Specifying pages and classes of a When Rule

Ensure that a When Rule accesses or updates information on Clipboard Pages by specifying the page name and class of the pages. At run time, these pages contain the properties that are referenced on the other tabs of a When Rule.

Define the pages and classes of a Rule to:

©2025 Pegasystems Inc

- Set up Rule prompting during Rule configuration.
- Configure Rule validation.
- Apply default class for list elements.

For more information about pages and classes, see Defining the pages and classes of a Rule.

1. In the navigation pane of Dev Studio, click **Records** > **Decision** > **When**, and then open the Rule that you want to edit.
2. On the Rule form, click the **Pages & Classes** tab.
3. In the **Page name** field, enter the name of the page that contains the property that is referenced in this Rule.
4. In the **Class** field, select the class of the page.
5. **Optional:** To add more pages and classes, click **Add item**, and then repeat steps 3 and 4.
6. Click **Save**.

- When Condition Rules
- Creating a When Rule

More about When Condition Rules

When Condition Rules help you build application logic so that you can deliver software that dynamically responds to changing business scenarios. When Condition Rules compare the value of one property reference against a constant, or against the value of another property reference. If you define more than one comparison, you can combine the results with `AND`, `OR`, and `NOT` operations to determine the final true/false outcome.

Conditions in activity steps

Each activity step can reference one or several When Condition Rules, as preconditions for a method, or transitions between the completed method and the next step.

©2025 Pegasystems Inc

As a precondition, a When Condition Rule determines whether the method in the activity step runs or not. Referenced in a transition, a When condition determines what activity processing runs next after the completion of the current activity step.

For more information, see Creating an activity and Configuring steps in an activity.

Standard Function

To run a When Condition Rule in an Expression or in the context of Java code, call the standard Function *callwhen()* in the Pega-RULES *Default* library.

You can view the generated Java code of a Rule by clicking **Actions** > **View Java**. You can use this code to debug your application or to examine how Rules are implemented.

- When Condition Rules
- Creating a When Rule

Performing unit testing on a When Rule

For more efficient and detailed debugging, you can test a When Rule individually before testing it in the context of the application that you develop. You can also convert the test into a Pega unit test Case to validate application data by comparing the expected property values to the actual values that the test returns.In a continuous delivery environment, Pega unit testing gives you feedback on the quality of your applications to ensure that you can quickly identify and correct issues.

By default, when you run the When Rule, the result assertion uses the input value that you enter and the result that the test returns. The system generates the assertion when you convert this test to a test Case.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **When**.
3. In the list of When Rule instances, click the Rule that you want to test.
4. In the Rule form header, click **Actions** > **Run**.

©2025 Pegasystems Inc

5. In the **Run** window, in the **Run context** pane, in the **Thread** list, select the thread in which you want to run the Rule.
6. In the **Page** list, select whether to copy parameter values from an existing page or to create an empty page:
   - To use parameter values from a Clipboard Page in the selected thread, click **Copy existing page**, and then, in the **Page to copy** field, select the page that you want to use.

     For example, you can select a Clipboard Page to test specific values.

   - To start a test from a page that contains no parameter values, click **Empty test page**.

> **NOTE:** The system runs the Rule instance on the *RunRecordPrimaryPage* Clipboard Page, regardless of the page that you select from this list. If you convert this test run to a test Case and the *RunRecordPrimaryPage* Clipboard Page requires initial values, configure the Clipboard to populate the page with initial values. For more information, see Setting up your test environment.

7. **Optional:** To apply a Data Transform to copy values for the test, select the **Apply Data Transform** checkbox, and then select the transform to apply.
8. **Optional:** To change the values that the Rule uses during the test, enter the new values on the main test page.
9. Start the test by clicking **Run**.
10. **Optional:** To convert the test run into a Pega unit test Case, click **Convert to test**, and then configure the test Case.
    For more information, see Creating unit test cases for Rules.
11. **Optional:** To view the pages that the unit test generates, click **Clipboard**.
    For more information, see Clipboard pages created by the Run Rule feature.

©2025 Pegasystems Inc

What to do next:

If the Rule has errors, debug the Rule by using the Tracer tool. For more information, see Debugging When Rules with the Tracer tool.

- When Condition Rules
- Unit testing individual Rules
- Using the Clipboard tool
- Viewing test Case results
- Application debugging by using the Tracer tool

Debugging When Rules with the Tracer tool

If your When Rule returns unexpected results, and you cannot determine the problem by running the Rule and examining the Clipboard Pages, run the Tracer tool. Using Tracer, you can watch each step in the evaluation of a When Rule, and as a result, you can find and then fix issues in your application faster.For example, you can use Tracer if a When Rule returns false but you expect the Rule to return true.

1. In the footer of Dev Studio, click **Tracer**.
2. In the header of the **Tracer** window, click **Settings**.
3. In the **Events to trace** section, select **When Rules**:
   - To trace events when the Rule starts running, select the checkbox in the **Start** column.
   - To trace events when the Rule ends running, select the checkbox in the **End** column.

   You can select both **Start** and **End** checkboxes.
4. In the **Rulesets to trace** section, select the Ruleset that contains the Rule that you want to trace.
5. **Optional:** To avoid memory use from tracing events that occur in other Rulesets, clear the checkboxes for other Rulesets.
6. Click **OK**.
7. Return to Dev Studio, and then run the When Rule.

©2025 Pegasystems Inc

> **NOTE:** Keep the **Tracer** window open by minimizing the window.

8. In the **Tracer** window, watch the Tracer output as the Rule runs.
9. **Optional:** To obtain more information, click a row in the Tracer output.
10. Analyze values and information present while the When Rule runs, and then adjust relevant elements of your application.
    For more information, see Defining conditions for a When Rule.

- Performing unit testing on a When Rule

Defining Conditions in the Condition builder

Use the condition builder to create conditions that define the behavior of your application, or to use for propositions evaluated by a proposition filter. You can save custom conditions to the condition library for future use.

> **Before you begin:**
>
> - If you want to use a configuration as a value to compare at run time to specify application behavior, create a configuration. For more information, see Creating Configurations.
> - If you want to evaluate an answer to a question at run time, create the question. For more information, see Adding a question to a question page.
> - If you want to use a Prediction to evaluate at run time, create and configure the Prediction for Case Management. For more information, see Predicting Case Outcomes.
> - If you want to reuse a When condition as a value to compare at run time, mark the When condition as a relevant record. For more information, see Marking a record as relevant.

1. In App Studio, navigate to a condition builder that defines the application element that you want to edit.

PEGA ©2025 Pegasystems Inc

For example:

The condition builder for refreshing a layout is on the configuration pane of that layout. For more information, see Defining refresh conditions for UI areas.

2. **Optional:** For proposition filters, to remove all previously added criteria from the condition builder, click **Actions** > **Clear all criteria**.
3. In the **Configure condition** section, in the list of values, select a value to evaluate:
   - To select a configuration that defines application behavior at run time, click **Configurations**, and then select the configuration from a Configuration Set.
   - To select a field from a Data Model in your application, click **Fields**, and then select the field.
   - To select a question to evaluate the answer at run time, click **Questions**, and then select the question.
   - To select a Prediction, click **Predictions**, and then select the Prediction.
     If the Prediction has a binary outcome, for example, the probability of a fraudulent claim, the following fields are available:
     - **Probability** (value between 0 and 1, for example, 0.01)
     - **Segment** (cutoff probability labels, if defined in the Prediction, for example, **Abnormal** and **Normal**)

     If the Prediction has a continuous outcome, for example, future credit card transactions, the **Value** field is available and contains the predicted value.

   - To select a when condition from your application, click **When conditions**, and then select the condition.

     TIP: To view the logic of the When condition that you want to use, click **Preview this condition**, as in the following figure:

©2025 Pegasystems Inc

Previewing a condition

4. In the comparator list, select a comparator.
5. In the value field, enter or select a value that your application compares with the field in the Data Model or a when condition.

   For example:

   `Status work is equal to Done`

6. **Optional:** To add more conditions, click the **Add a row** icon, and then repeat steps 3 through 5.
7. If you add multiple conditions, between the rows, select the **and** or **or** operator to define how to evaluate the conditions.

   You can group conditions using the **and** or **or** operators.

   Result:

   If you select **and**, the condition evaluates to true when all of the rows evaluate to true. If you select **or**, the condition evaluates to true if at least one of the rows evaluates to true.

8. In the list in the upper-right of the page, select one of the following options to specify how to evaluate condition groupings:

   - To evaluate the conditions that are linked with the **AND** operator as a group, and to evaluate the conditions that are linked with the **OR** operator individually, click **Group ANDs**.

©2025 Pegasystems Inc

With this grouping, the grouping of condition 1 **AND** condition 2 **OR** condition 3 **OR** condition 4 is evaluated as (1 AND 2) OR 3 OR 4. That is, either conditions 1 and 2 must both be true, or either one of conditions 3 and 4 must be true.

- **Group ORs** - Select this option if you want conditions linked with the **OR** operator to be evaluated as a group, and conditions linked with the **AND** operator to be evaluated individually.

  With this grouping, the grouping of condition 1 **AND** condition 2 **OR** condition 3 **OR** condition 4 is evaluated as 1 AND (2 OR 3 OR 4). That is, both condition 1 and one of conditions 2, 3, and 4 must be true.

- To define a grouping by both the **AND** and **OR** operators, click **Use advanced logic**, and then specify the condition grouping in the **Logic string** field.

> **NOTE:** If you change a condition that uses advanced logic for the **Group ANDs** or **Group ORs**, all the groupings that you previously defined are reset.

> **TIP:** Conditions that are evaluated as a group are displayed on a single block of gray background, so that you can identify them more easily.

9. **Optional:** To reuse the condition in the future, save the condition to the library:
   - For conditions in a workflow, click **More** > **Add to when condition library**.
   - For conditions in a proposition filter, click **Actions** > **Save to library**.

> **NOTE:** The **Save to library** option is not available if the conditions include either Strategy Rules or when conditions with parameters. The option is also disabled for proposition filters which include

©2025 Pegasystems Inc

> default criteria that are created using the **Use advanced logic** option.

> **Result:**
>
> The conditions are saved to a new When Rule in the top level class of the proposition filter. The system automatically registers the When Rule as a relevant record for use in the condition builder.

- Conditionally starting a Process
- Defining Conditions for skipping a Stage
- Displaying optional Actions conditionally
- Displaying supporting Processes Conditionally
- Adding Decisions to Processes

Expression Builder

The Expression Builder is a tool that guides you in authoring arithmetic and logical Expressions in a natural language format. For example, you can compute an employee's salary by defining a calculation that uses properties, such as the number of years employed, employee review rating, and job grade level.

You can open the Expression Builder in multiple locations throughout Dev Studio by clicking the **Build an Expression** icon. For example, you can enter Expressions in activities, Data Transforms, constraints, and most other places that compute a value at run time.

Accessing the Expression Builder

©2025 Pegasystems Inc

For more information about using the tool, see Building Expressions with the Expression Builder.

The Expression Builder provides the following features:

- Build Expressions in an integrated text editor
- Add Functions and properties into the text editor
- Search for properties and non-internal Functions.
  The following figure shows the list of results for a keyword search:

Searching for properties and Functions in the Expression Builder

- Select properties and Functions while entering text in the text editor

©2025 Pegasystems Inc

The following figure shows some example suggestions:

Selecting properties and Functions from a text editor prompt

- A navigation pane that displays Functions in available libraries and properties within a selected page
- Automatic bracket matching
- Syntax validation
  The following figure shows an example of syntax validation:

Syntax validation in the Expression Builder

- Building Expressions with the Expression Builder
- Expressions

Building Expressions with the Expression Builder

Create calculations that save you time by automatically providing values in your application by authoring Expressions in the Expression Builder. The Expression Builder eases the construction of Expressions in your application by providing a large input area, prompting for Functions, prompting for properties, and adding the ability to test your results.

For more information about the Expression Builder, see Expression Builder.

An Expression is a single text element that, after evaluation at run time, usually returns a single value. Use Expressions to perform operations on data, such as performing

©2025 Pegasystems Inc

mathematical operations, comparing date and time values, manipulating text, transforming data, and converting data types. For example, you can create an Expression to calculate the total salary of an employee using their base salary, a bonus that depends on their years of experience, and another bonus dependent on the number of certifications that the employee has.

1. In Dev Studio, navigate to a place in your application where you want to build an Expression.

   For example:

   In an activity, you can use the Expression Builder to build the Function to get the current date time. You can then use the Function in a *Property-Set* method to set Case data such as *SubmitDateTime*.

   The following figure shows the location of the **Build an Expression** icon in a Rule form:

Accessing the Expression Builder

2. In the **Expression Builder** dialog box, build an Expression:

Choices	Actions
Enter the Expression manually	In the text field, enter an Expression. You can include combinations of constants, Functions, arithmetic operators, logical operators, comparison operators, and property references. For

©2025 Pegasystems Inc

Choices	Actions
	more information, see Examples of Expressions. **TIP:** When you press the Period (.) key, the Expression Builder prompts a list of the fields available in the Data Page in whose context you currently work. When you press SHIFT+2 (@), the Expression Builder prompts a list of the available Functions.
Browse for and apply a Function	a. In the header of the **Expression Builder** dialog box, click **Browse**. b. In the navigation pane, click **Functions** to expand a list of available Function libraries. c. Click a Function library to expand the available Functions. **For example:** Click **DateTime** to view Functions connected to calculating date and time. d. Click the plus icon to add the Function to your Expression.

©2025 Pegasystems Inc

Choices	Actions
	When you add the Function to the text area, the Function does not include arguments. **For example:** When you add the *addToDate* Function, the text `addToDate(` appears in the text area. **TIP:** Hover over a Function to see a tooltip with information about each Function, as shown in the following figure: *Sample Function* **NOTE:** The Expression Builder displays only non-internal Functions. To display all Functions in the system, see Displaying all

©2025 Pegasystems Inc

Choices	Actions
	Functions in the Expression Builder. e. If the Function requires an argument, enter the argument value. **For example:** To add a day to the current date, after you select **addToDate** in the previous step, enter `@CurrentDateTime()@CurrentDateTime(),"1","","","","")`. **NOTE:** When using the *@CurrentDate* function, use the lowercase year format (`@CurrentDate(yyyyMMdd,"Asia/Tokyo")`). The uppercase format (`@CurrentDate(YYYYMMdd,"Asia/Tokyo")`

©2025 Pegasystems Inc

Choices	Actions
	refers to ISO year-week date and might produce incorrect results in the last week of the year.
Browse for and apply a property	a. In the header of the **Expression Builder** dialog box, click Browse. b. In the navigation pane, click Properties to expand a list of available properties organized by context. You can select properties from the following types of pages: **Current Page** Includes a list of properties from your current Case context. **Custom Pages** Includes properties from custom pages defined on the Pages & Classes tab of the Rule that you edit. **Data Pages** Includes a list of properties available from application Data Pages.

©2025 Pegasystems Inc

Choices	Actions
	Parameter Page Includes properties defined as parameters on the Rule that you edit. c. Click the selected context to expand the available properties. Some areas, such as Data Pages or custom pages, might include nested Data Pages. d. Drag the property to the text field to add the property to your Expression. **TIP:** Hover over the property to view additional information, such as the property name, property label, and the class that includes the property. **For example:** Click the Parameter Page page to display a list of parameters on the current Rule, and then click the plus icon to add the parameter to your Expression.

©2025 Pegasystems Inc

Choices	Actions
	The Expression Builder

3. **Optional:** To validate the correctness of the Expression, test the Expression:
   a. In the header of the **Expression Builder** dialog box, click Test.
   b. If your Expression includes properties or parameters, in the **Test data** pane, enter the values that you want to use for testing.

Result:

The **Result** section, which is displayed with a blue background in the following figure, includes the results of the computation. You can also test a part of the Expression by selecting a fragment that you want to test, as shown in the following figure. If you test a part of the Expression, the result appears in the **Result of selection** section.

©2025 Pegasystems Inc

Testing an Expression

If the Expression includes errors, the Expression Builder displays errors in the **Error** section, as shown in the following figure. You can hover over the section to view the full error message.

Error in an Expression

4. Click **Submit**.

Result:

The value of the Expression becomes available in the Rule form.

©2025 Pegasystems Inc

Expressions

An Expression is a single text element that produces a string value. Users create both SQL Expressions (or Functions) for use in report definitions and Java Expressions for use in activities and other Rules.

The information in this topic discusses creating Java Expressions.

Expressions in Pega Platform look similar to formulas in Microsoft Excel but are based on Java language conventions.

You can type in an Expression, or (in many places) use the Expression Builder to guide you in entering an Expression. Expressions can include constants, property references, operators for arithmetic and logical operations, parentheses to control the order of evaluation, and Functions.

Expressions can perform operations on data, including:

- Performing mathematical operations
- Comparing property values
- Manipulating text
- Transforming data
- Converting data types (casting)

> **NOTE:** Expressions, which are used in many situations, differ from Declare Expression Rules, which create Expressions that are evaluated automatically. Declare Expression Rules contain Expressions, but so do many other Rule Types.

Expressions in Rule forms

You can enter Expressions in activities, Data Transforms, Constraints Rules, and most other places that compute a value at run time.

PEGA ©2025 Pegasystems Inc

In Rule forms, the sorts of Expressions you can enter into fields in the forms depend on the defined behaviors of the Rule form's fields and what sorts of entries they allow. The field in which an Expression appears determines whether the Expression is part of:

- The right-hand side of an Assignment to a property
- The computation of a value
- A Boolean (or logical) Expression (yielding a true/false value)
- A property reference containing an Expression as a subscript

You can also use Expressions in most fields that allow a constant value. Similarly, you can use Expressions to supply the subscript value of a list or group element within a property reference.

You cannot use Expressions in fields that require property references as the destination of assigning a value. For example, you cannot use an Expression in the Target column of a Set action in a Data Transform or in the Property-Set method parameter array.

Expressions in HTML and XML streams

Through the <when> JSP tag and Java scriptlets (or the <when> directive and Java directive), you can include Expressions in HTML-based Rules or XML Stream Rules. You can use these in HTML Rules, XML Stream Rules (*Rule-Obj-XML* Rule Type) and Correspondence Rules (*Rule-Obj-Corr* Rule Type).

- Building Expressions with the Expression Builder
- Examples of Expressions

Examples of Expressions

Expressions provide a wide range of computations. Some examples of Expressions are:

Constants

An Expression can be as simple as a single literal constant.

PEGA ©2025 Pegasystems Inc

```
"Good Evening"
142
true
20050705
0x143F871A
```

Single property references

An Expression can reference a single property, identifying the page on which it is found. In the context of an activity, an Expression can reference a parameter or local variable.

```
.Price
parameters
MortgageLoan.Application.ZIPCode
primary.pyLabel
```

Aggregate property references

You can identify aggregate properties or parts of aggregate properties. The property mode of the target must match the result of the Expression.

```
MortgageLoan.Application.Address(4)
Globe.Division(7).Unit("West")
Invoices.pyOrders(2).pyItems("Manuals").pyItemNames
```

Linked property references

You can identify properties accessed through linked properties, using the syntax `.property1.property2.`, where property1 is the linked property and property2 is a property reference in the linked class.

©2025 Pegasystems Inc

> **NOTE:** Note the initial period character before the property name. When no initial period is present, the system assumes that property1 is a page name.

```
.pyUpdateOperator.pyLabel
.pyUpdateOperator.pySkillsPrimary(6)
```

Arithmetic, logical, and comparison operators

You can use most Java operators for arithmetic, string operations, comparisons, and conditions. Use parentheses to control the order of evaluation.

```
.Price * (1+(.Tax/100)) + ShipInfoPage.ShippingCost
.pyEffortActual >= .pyEffortForecast
3.14159*.radius *.radius
```

Function calls

Your Expressions can call built-in Functions, Functions in standard libraries, and Custom Functions.

```
@SUM(.SubComponents().Price)
@Pega-RULES:MapTo.Function(argument1, argument2)
```

All combined

Expressions can incorporate all the elements described in this topic together:

```
@SUM(.SubComponents(38+.Offset).Price)
```

- Expressions

©2025 Pegasystems Inc

Constants in Expressions

An Expression can consist of only a constant — also called a literal value — or can include constants, operators, property references and Functions.

Use these guidelines to enter constants. The value you type may depend on the Type of the property. Include all quote characters as shown. (Don't use Microsoft Word Smart Quote characters; use only the ASCII double quote character `"` to identify constants.)

Type	Instructions	Example
`TrueFalse`	Enter "true" or "false" or "T" or "F." In most cases, such properties appear as a checkbox or radio buttons. Avoid the null value "" for a `TrueFalse` property; in many but not all settings, it is the same as "false".	"true"
`Text`	Enter two double quotes for the null string. For normal text, enter printable characters (other than double quotes) between two double quotes. You can include single quotes within the double quotes.	"" "Hello World" "It's a beautiful day" "Make\tmy\tday."

©2025 Pegasystems Inc

Type	Instructions	Example
	Use the two character sequence \" to include a double quote. Use Java character escapes to include tab characters (\t), backspace (\b), newline (\n), form feed (\f), carriage return (\r), backslash (\\), and \zzz for the Latin-1 character identified by the octal value zzz, between 000 and 377.	
`Identifier`	Enter a text value not containing double quote characters, newline characters, or carriage return characters.	MarketPrice ProductID <X-95$?? >
`Integer`	Enter digits 0-9 and an optional plus or minus sign. Do not include commas or a decimal point. Enter pairs of hex digits (0 to F) preceded by 0x. for a hexadecimal value.	-443 1 432329894 0xFFF4 0x143F871A 014371
`Decimal`	Enter any number of decimal digits, a single comma or period, and additional decimal digits.	437.1 3.14159

©2025 Pegasystems Inc

Type	Instructions	Example
	Don't include more than one punctuation mark. Don't start a value with a period; use a leading zero.	
`Password`	Use any non-blank printable character. Don't include spaces or control characters. Case is significant.	4X9KK6u
`Date`	When a date constant appears alone in an Expression, include double quotes to prevent the value from being interpreted as an integer. Enter two or four digits for the year, two digits for the month, two digits for the day. If the year contains two digits, the system uses an algorithm to determine a "reasonable" century based on the current date. During 2006, the system prefers dates between 1997 and 2097. The valid Expression	"19990512" "20021231" "990101"

©2025 Pegasystems Inc

Type	Instructions	Example

	`12/10/2006`
	
	is an Expression involving two divisions and three integers, whereas
	
	`"12/10/2006"`
	
	is an American-style date.
	
	The empty string and "" are equivalent to "19700101" or January 1, 1970. In settings where this is not desirable, processing can test for and block this date value.	

`Double`	Enter an optional sign, digits, decimal point, digits, the letter E and an exponent. All parts of the literal are optional except for the decimal point and one digit.	123.45 0.04 6.02E23
`DateTime`	Enter a value using this pattern: `yyyyMMddtHHmmss.SSS zzz`. Enter: • Two or four digits for the year	20060415T115959.123 GMT

©2025 Pegasystems Inc

Type	Instructions	Example
	• Two digits for the month • Two digits for the day • The single `T` (or t or period . or comma ,) • Two digits for the hour • Two digits for the minutes • Two digits for the seconds • A period • Three digits for the milliseconds • One space • The literal `GMT` for the zone. The year, month, and day are required. Separate the date and time portions of the value with either `T`, `t`, a single space, or a punctuation mark such as a period or a comma. If you include milliseconds, separate the seconds from the milliseconds with either a period (`.`) or a comma (`,`). Use the GMT time zone only.	

©2025 Pegasystems Inc

Type	Instructions	Example
	The time portion is optional. If you omit the time portion, follow the guidelines above for Date constants, by enclosing the value within parentheses. If the year contains only two digits, the system uses an algorithm to determine a "reasonable" century based on the current date. During 2007, the system prefers dates between 1998 and 2098.	
`Time of Day`	Enter two digits for the hour, from 00 to 23. Optionally, enter two digits for the minute, from 00 to 59. If you include six digits, the system interprets the last two as seconds. You can include an optional colon to separate the segments of this value. Do not enter a time zone.	`000000` for midnight `12:00:00` for noon `23:59:59` for one second before midnight

- Expressions
- Data Transforms

©2025 Pegasystems Inc

Functions in Expressions

Functions extend the power of Pega Platform™ Expressions. An Expression can contain many Function calls, including calls to built-in Functions, standard Functions, and Custom Functions that you create. Use the Function Builder to help find the Function that you need.

This topic relates to creating and using Java Functions. For information about creating and using SQL Functions in Report Definition Rules, see About Report Definitions.

- Expressions

Methods for calling a Function

The following forms call a Function.

Fully qualified Function call

The fully qualified syntax for calling a Function identifies both the Ruleset and the library:

```
@(RuleSet:libraryname).functionname(arg1, arg2... argn)
```

For example:

```
@(MyInventory:Sort).CardType("Gold")
```

Using fully qualified references eliminates the risk of Function overrides by a library or Ruleset.

By fully qualifying a Ruleset, standard Ruleset resolution does not apply. Double-check your requirements before using this type of Function call.

©2025 Pegasystems Inc

> **NOTE:** A deprecated syntax used before Version 04-01 is still available as an alternative to using a fully qualified Function call:

```
lib(ruleset:libraryname).functionname(arg1, arg2... argn)
```

When using this syntax, "ruleset" may be omitted (defaults to `Pega-RULES`), and "libraryname" may be omitted (defaults to `default`).

Library qualified Function call

The library qualified syntax for calling a Function identifies the library name:

```
@LibraryName.FunctionName(arg1, arg2... argn)
```

Using library qualified references provides a more readable syntax than the fully qualified syntax, and prevents accidental library conflicts that can occur with unqualified calls.

For example, if two libraries with the same name are located in different Rulesets, when using library qualified notation, the library from the Ruleset listed higher in the operator runtime Ruleset list is selected.

Additionally, using this notation helps to avoid multiple "suitable found" instances, such as when the same Function exists in two different libraries but in the same Ruleset.

With library qualified references, standard Ruleset resolution applies, allowing the Function to be resolved from any Ruleset. This provides the capability to override a Function in a different Ruleset name, similar to how other Rule Types can be overridden.

Unqualified Function call

The unqualified syntax for calling a Function calls a Function Rule:

©2025 Pegasystems Inc

```
@FunctionName(arg1, arg2... argn)
```

The unqualified syntax is the simplest to read, but does not identify a Ruleset or a library for the Function. Your system may contain many Functions that match the Function name.

> **NOTE:** When using this syntax, if another developer creates a Function with the same name and signature in another library or Ruleset, their call may be affected.

Pega Platform uses your Ruleset list to find the best one to execute, using the following algorithm:

1. Collect a list of the *Rule-Utility-Function* Rules belonging to the Rulesets listed in the user's Ruleset list that have the same name and same number of parameters.
2. Exclude Functions defined in any library named default. (The use of the default library is deprecated.)
3. If the list is empty, no such Functions exist. The system reports an error.
4. If more than one such Function exists (because the same name is overloaded), select the best match by comparing the data types of the parameters, using the promotion Rules for Pega Platform data types.
5. If the *Rule-Utility-Function* Rule has not supplied data types for its parameters or return value, use standard Java promotion Rules to match parameters. As a last resort, substitute `double` for the preferred `BigDecimal` type if necessary to get a match.
6. If more than one Function matches the selected name and fully decorated signature, then the Function from the higher listed Ruleset is executed.

Java call

Functions when compiled extend the Java class:

©2025 Pegasystems Inc

`com.pegarules.generated.` *myruleset_mylibrary*

where the Ruleset name and library name are converted to lowercase. To call a Function from within Java, use the following syntax:

```
result = com.pegarules.generated.
myruleset_mylibrary.MyFunction( params )
```

using the exact case for the Function name.

For example, to call the CardType() Function in the Sort library of the MyInventory Ruleset, use:

```
myinventory_sort.CardType("Gold")
```

Pega Platform constructs the Java class name for a *Rule-Utility-Library* by concatenating the Ruleset name, an underscore and the library name, with the resulting string converted to lowercase and characters not valid in a Java identifier translated to an underscore.

Built-in and When() Functions

Four Functions are part of the Expression language. All other names following an at-sign are references to Function Rules, instances of the *Rule-Utility-Function* Rule Type. An Expression can use any of these built-in Functions:

Syntax	Description
These Functions are part of the Expression language. Their names are reserved and cannot be overridden. These Functions may	

©2025 Pegasystems Inc

Syntax	Description
only be invoked using the preferred ('@') syntax.	
@if(boolean expression, result1, result2)	Evaluates the condition in parameter one and then, if true, evaluates and returns parameter two, otherwise evaluates and returns parameter three. Equivalent to the Java ternary operator "?:." Only the one appropriate result parameter is evaluated.
@and(e1, e2, ...)	Logical AND operation of the (arbitrary number of) parameters (Evaluated left-to-right, stopping when the result is known)
@or(e1, e2, ...)	Logical OR operation of the (arbitrary number of) parameters (Evaluated left-to-right, stopping when the result is known)
General purpose	
@when(name)	Evaluates a When Condition Rule in the context of the Primary Page of the Rule in which the Expression appears. The parameter can be an Expression that evaluates to a text value.
@when(name, pagename)	Evaluates the specified When Condition Rule using the supplied `ClipboardPage` PublicAPI object (not a page name) as the Primary Page for evaluation. Either parameter can be an Expression that evaluates to the correct data type.

©2025 Pegasystems Inc

Syntax	Description
	NOTE: The Primary Page for evaluation is supplied as a `ClipboardPage` object, not a page name, so that embedded pages (or pages whose name is not known) can be evaluated.

You can use a `Value List`, `Value Group`, or `Page Group` property as an argument to the @SUM(), @MIN(), @MAX() and @AVERAGE().

Arithmetic Functions

These basic arithmetic Functions are available.

Syntax	Description
Scalar arithmetic	
@abs(num)	Absolute value
@sqr(num)	Square
@sqrt(num)	Square root
@exp(num)	Exponential
@ceil(num)	Ceiling, least integer not greater than

©2025 Pegasystems Inc

Syntax	Description
@floor(num)	Floor, greatest integer not less than
@round(num)	Rounded to the nearest integer
Aggregate arithmetic	
@sum(num1, num2)	Sum
@max(num1, num2)	Maximum
@min(num1, num2)	Minimum
@mode(num1, num2)	Modal value
@stddev(num1, num2)	Standard deviation
@average(num1, num2)	Average

You can use a `Value List`, `Value Group`, or `Page Group` property as an argument to the @SUM(), @MIN(), @MAX() and @AVERAGE().

DateTime Functions

The following Functions operate with date and time values. A `DateTime` property type represents an instant in time while `Date` and `TimeofDay` property types represent values. Adding (the BigDecimal/double values corresponding to) a `Date` and a `TimeofDay` together similarly represents an instant in time, but in an unknown or arbitrary time zone.

©2025 Pegasystems Inc

Syntax	Description
These Rule-Utility-Function Rules are similar to the corresponding Functions in Microsoft Excel.	
@date(year, month, day)	Returns a BigDecimal value (Pega Platform Date) corresponding to th int parameters specified. The year is a four digit year, and is not adjusted for a “base date.” The parameters are interpreted leniently (a in Microsoft Excel) through the use of the Java Calendar class.
@datevalue(string)	Returns a BigDecimal value (Pega Platform Date) corresponding to th parsed value of the string specified. Only the date portion of the string is considered.
@day(double) @day(BigDecimal))	Returns the int value corresponding to the day of the month (1 to 31) when the double parameter is interpreted as a Pega Platform Date.
@month(double) @month(BigDecimal)	Returns the Java int value corresponding to the month of the year (0 t 11) when the double parameter is interpreted as a Pega Platform Date The corresponding Excel Function also returns 0 to 11.
@today() @today(selection)	Returns the BigDecimal value (Pega Platform Date) corresponding to the current date.
@weekday(double) @weekday(BigDecimal)	Returns the int value corresponding to the day of the week (Sunday = to Saturday = 7) when the double parameter is interpreted as a Pega Platform Date.
@year(double) @year(BigDecimal)	Returns the int value corresponding to the year when the double parameter is interpreted as a Pega Platform Date.
@hour(double) @hour(BigDecimal)	Returns the Java int value corresponding to the hour of the day (0 to 2 when the double parameter is interpreted as a Pega Platform Time of Day.
@minute(double) @minute(BigDecimal)	Returns the Java int value corresponding to the minute of the hour (0 59) when the double parameter is interpreted as a Pega Platform Tim of Day.

©2025 Pegasystems Inc

Syntax	Description
@second(double) @second(BigDecimal)	Returns the Java int value corresponding to the second of the minute to 59) when the double parameter is interpreted as a Pega Platform Time of Day.
@time(hour, minute, second)	Returns a BigDecimal value (Pega Platform Time of Day) correspondir to the int parameters specified.
@timevalue(string)	Returns a BigDecimal value (a Pega Platform Time of Day value) corresponding to the parsed value of the string specified. Only the tim portion of the string is considered.
The following Functions facilitate conversion between BigDecimal/ double values derived from DateTime properties and Date or Time of Day properties. Use these Functions to adjust the double value resulting from the sum of a Date and Time of Day, or the double value resulting from a DateTime property into a (possibly different) known time zone. The return type of these Functions is the same as the type of the first argument.	
@toGMT(double, zone) @toGMT(BigDecimal, zone)	Returns a double/BigDecimal value (Pega Platform DateTime) in the GMT time zone corresponding to the double value interpreted in the “zone” time zone. Use to convert the sum of a Date and Time of Day to DateTime. If zone is null, then the time zone of the requestor is used.
@toLOCAL(double, zone) @toLOCAL(BigDecimal, zone)	Returns a double/BigDecimal value that is the result of treating the double value as a Pega Platform DateTime (in the GMT time zone) and converting it to the time zone. You can then treat the resulting value a

©2025 Pegasystems Inc

Syntax	Description
	Date and Time of Day values in the specified time zone. If zone is null, then the time zone of the requestor session is used. For example, if bostonTime is a double value that represents a date and time in the Eastern Standard Time zone, you can find the corresponding time in India by using: toLOCAL(toGMT(bostonTime, “EST”), “IST”).

Calling an activity in an Expression

The standard Function *callActivity( )* in the *Pega-RULES Utilities* library calls an activity, but returns only void, not a value result. You can run an activity in an Expression, but only for its side effects. Identify the Primary Page, the activity, and the parameter page. For example:

```
@Pega-RULES:Utilities.callActivity(pyWorkPage, MyActivity, tools.getParameterPage())
;
```

Calling an overloaded Function

Pega Platform allows multiple Function Rules to be defined with the same name, in the same Ruleset, version, and library, if they have different signatures. The signature of a Function is computed automatically from the name, the position and data types of parameters, and the return type. Functions that have multiple variants are called overloaded.

When calling an overloaded Function, you do not need to specify which variant to use. The system determines which to use based on the types and positions of parameters you supply.

For example, assume one Function named Round is defined with two variants:

- Round(double: d) — Returns an int by rounding the input value d, a double
- Round(double: d, places: n) — Returns a double by rounding the value to n decimal places

©2025 Pegasystems Inc

If your call includes two parameters, the system uses the second variant.

Displaying all Functions in the Expression Builder

Build advanced Expressions that automate calculations in your application by displaying internal Functions and libraries from your entire system in the Expression Builder. By default, the Expression Builder displays only non-internal libraries and Functions. You can display all Functions in the system by overriding the *pyShowInternalLibrary* When Rule setting.

Internal Functions are not displayed by default, because they are subject to change without notice. Use internal Functions with caution. A Function is internal if the *pyMethodStatus* property of the library that stores the Function has the value of `Internal`. To check the *pyMethodStatus* property, view the XML or the clipboard page of the Rule. For more information, see Viewing the XML of a Rule and Using the Clipboard tool.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Decision** category, and then click **When**.
3. In the list of instances of When Rules, click the **pyShowInternalLibrary** Rule.

> **TIP:** To find the Rule faster, you can filter the Rules by name.

4. In the Rule form header, click **Save as**.
5. On the **Save As When: Show Internal Library** form, in the **Context** section, in the **Add to Ruleset** list, select your application Ruleset and Ruleset version.
6. Click **Create and open**.
7. On the **Conditions** tab, in the **When...** section, click **false**.
8. Click **Actions** > **Edit**, as shown in the following figure:

©2025 Pegasystems Inc

Editing the When Rule

9. In the **Condition** dialog box, enter `true`, and then click **Submit**.
10. Click **Save**.

> **Result:**
>
> When you author Expressions in the Expression Builder, you can browse through all Functions and libraries in the system.

Related concepts

- When Condition Rules

Defining URL patterns for Cases

Share your work items with users of the same application by defining URL patterns for the work items in your application, for example, for Cases or other work items. Simple, meaningful URLs make collaboration more efficient for users. For example, they can bookmark a Case for quick access, and then send a direct link to the Case to other users.

Before you define URL patterns for work items, your application must contain a URL Mappings Rule. Each application can have only one URL Mappings Rule, whose name is always *pyDefault*. You define URL patterns by editing this Rule.

©2025 Pegasystems Inc

> **NOTE:** This procedure focuses on configuring URL patterns for Cases. Configuring URL patterns for other objects, such as reports and landing pages, requires advanced configuration and creating your custom activities.

1. Create a URL Mappings Rule in your application:

   > **NOTE:** If your application already contains a URL Mappings Rule, go to step 2.

   a. In the header of Dev Studio, click **Create** > **Technical** > **URL Mappings**.
   b. In the **Context** section, in the **Add to ruleset** list, select the name and version of a Ruleset to store the URL Mappings Rule.
   c. Click **Create and open**.
   d. Click **Save**.
2. In the navigation pane of Dev Studio, click **Records**.
3. Expand the **Technical** category, and then click **URL Mappings**.
4. In the list of URL Mappings instances, click *pyDefault*.
   Each application has only one valid URL Mappings Rule, whose name is always *pyDefault*.
5. In the **Cases** section, start configuring a URL pattern:
   - To modify an existing pattern, click the **Switch to edit mode** icon, as shown in the following figure:

Switch to edit mode icon on the URL Mappings form

©2025 Pegasystems Inc

- To add a URL pattern to a work item, click **Add URL alias**.

6. In the **Identifier** field, enter a name for the URL alias that uniquely identifies it in all patterns in the Rule.

> **For example:**
>
> Enter `OpenWorkForReview`.

7. **Optional:** To generate a valid URL for an object, apply the URL pattern to the object:
   a. Select the **Map path elements for URL generation** checkbox.
   b. In the **Class** field, specify the class of the object for the properties mapping.
8. In the **Path element type** column, select **Constant**.
   The first path element must be a constant.
9. In the **Value** field, enter text that is a constant element in the URL pattern.

> **For example:**
>
> Enter `Review`.

10. **Optional:** To add more path elements to the URL pattern, click **Add path element**.
    a. To define text, in the **Path element type** column, select **Constant** and then, in the **Value** field, enter the text.
    b. To define a parameter, in the **Path element type** column, select **Parameter**.
    c. If you selected **Map path elements for URL generation** in step 7, in the **Property mapping** column, enter a clipboard property to map to the URL path element.

> **NOTE:** You can preview the URL pattern that you create below the **Build resource path** label, as shown in the following figure:

©2025 Pegasystems Inc

Preview of a resource path and the configured path elements

11. **Optional:** To make this URL pattern the default for objects in specific classes, in the **Make this resource path the default for the following classes** section, click **Add class**, and then in the text field, enter the name of the class.
The class can be the same class that you entered in step 7, or any of the child classes.

> **NOTE:** The option is available only if you selected **Map path elements for URL generation** in step 7.

©2025 Pegasystems Inc

12. Click Next.
13. In the **Define processing activity** dialog box, in the Class field, enter `Work-`, and then, in the Activity field, enter `pxOpenByKeys`.

Result:

The system identifies activity parameters and displays them in the table, as shown in the following figure:

Mapping an activity to a URL mapping Rule

14. In the **Value** column, enter the values for the processing activity parameters. Map your URL path elements (parameters) to the processing activity parameters.

For example:

In step 10, if you defined a URL path element as the `{param1}` parameter, in the Value field for an activity parameter, enter `{param1}`. For example, In the Value field for the `WorkID` parameter, enter `{param1}`. As a result, because `{param1}` is assigned to the `pyCaseID` property, at run time the

©2025 Pegasystems Inc

URL contains the Case ID. The following figure shows parameters with sample values:

Mapping a processing activity to the URL mapping

15. **Optional:** To pass an additional parameter to the activity that the current activity invokes, click **Add parameter**, and then repeat step 14.
16. Click **Finish**.
17. **Optional:** To add a URL pattern to another category, expand the category, and then repeat steps 5 through 16.
18. On the **URL Mappings** form, click **Save**.

> **NOTE:** If you make any changes to an activity referenced in a URL Mapping, you must re-save the individual URL Mapping. This step is necessary irrespective of any change in the individual URL Mappings rule.

©2025 Pegasystems Inc

For example:

You can use the mappings to generate URLs that follow the patterns that you define by configuring a **Copy shareable link** button in a case. The following figure shows the results of the configuration at run time:

Copying a simplified URL for a case

For more information, see Theme-Cosmos: Open work object in Full portal on Copy sharable Link.

Related information

- https://support.pega.com/support-doc/url-mappings-troubleshooting-faqs

Developing applications in Branches

Develop multiple features simultaneously without overwriting work and causing conflicts by implementing branched application development. As a result, you speed up the development process without the risk of overwriting the work.

Using Branches helps teams manage parallel development of both large and small-scale projects. Multiple teams or developers in a single team can work simultaneously on Rules in the application without concern for interfering or overwriting work. Working independently in an isolated space is a best practice in Pega Platform™ low-code application development. For example, in a small-scale project scenario, a developer can create one Branch to work on form properties and another to make changes in the application UI.

©2025 Pegasystems Inc

Branch Rulesets

Branch Rulesets are based on existing Rulesets and contain Rules that are in development in the associated Branch. By creating Branch Rulesets, teams can manage the independent development of features that share the same Rulesets. For more information about Rulesets, see Rulesets and Organizing Rules into Rulesets

Developing Application in Branches

Branch operations

To ensure the quality of your application, you can work with Branches in multiple ways. For example, establish Branch reviews and use Pulse to collaborate with other users to address the issues. Pack Branches into a JAR file or remove or deleted them from the system. After Branch development is complete, you can lock the Branch to ensure that no further modifications occur, and then merge it into a targeted Ruleset to make changes available in the development application. For more information, see Branch operations.

PEGA ©2025 Pegasystems Inc

Create a development application

Creating a development application is a best practice in branched development, especially for large-scale projects. A development application can provide an uninterrupted development process. Because you create a development application by copying the application, therefore the names of Classes and Rulesets remain unchanged. For more information, see Creating a development application

- **Implementing branched application development**
- **Branch operations**
- **Branch reviews**
- **Merging Branches into target Rulesets**
- **Branches and unlocked Rulesets**
- **Branches and Branch Rulesets**
- **Developing applications in Workspaces**

Implementing branched application development

For faster delivery of your products, configure your application development to use Branches. By implementing Branches, team members can work simultaneously on multiple features, without the risk of interrupting work of different team members.For example, team members can develop different Service-Level Agreement Rules or work on multiple bugs in parallel by using separate Branches.

To use Branches in your application, create a development application that is built on your base application. You create and save Rules in your development application, which results in creating Branch Rulesets. You can then work with Branches in several ways, such as creating Branch reviews and merging Branches.

To configure your application to use Branches and Branch Rulesets, complete the following steps:

PEGA ©2025 Pegasystems Inc

1. Create a development application that is built on your base application.
   For more information, see Creating a development application.
2. Add Branches to your application, into which you can save the Rules that you develop.
   For more information, see Adding development Branches to your application.
3. Work with Branches in several ways that meet your business requirements. For example, create a review, merge Branches, or delete Branches from the system.
   For more information, see Branch operations.

- **Creating a development application**
- **Adding development Branches to your application**
- **Adding system Branches**
- **Creating Branch Rulesets**
- Setting Branch development preferences
- Branches and Branch Rulesets

Creating a development application

To speed up application development, you can support branched development by creating a development application that is built on the production application. By providing Branches for development, you ensure that different teams and team members can work on different features simultaneously, without a risk of creating errors and conflicts.For example, different team members can work on bug fixes that correspond with two separate features but refer to the same base Rule. During development, conflicts might occur. Edits made by one team member might impact the work of other developers. Creating a development application is not mandatory but can provide an uninterrupted development process, especially for large or multiple development teams that work on the same application.

©2025 Pegasystems Inc

For branched development, create a new application by copying an existing application. As a result, the names of classes and Rulesets remain unchanged. Consequently, you can conveniently merge your changes after the development in a Branch is complete.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. In the Rule form header, click the Down arrow next to the Save button, and then click Save as.
3. In the **Application Record Configuration** section, in the Label field, enter a name of your development application.
   Team application names typically reflect the base application and team name or the focus of the team.

   > **For example:**
   >
   > For a base MyBankingApp application, the development team that works on the layout of the user Portal can name the development application MyBankingAppPortal.

4. Create a unique application Rule by changing the identifier:
   a. In the **Identifier** section, click Edit.
   b. In the dialog box that is displayed, in the text field, enter a new identifier.
   c. Click OK.
5. **Optional:** To create another version of a development application that already exists, in the Version field, enter a new version number.
6. In the **Context** section, in the Add to Ruleset list, select a Ruleset to store your edits.
   The Ruleset that the system prompts by default is the Ruleset in your production application. If you select a different Ruleset, you are unable to add your changes directly to the production application.
7. Click Create and open.

©2025 Pegasystems Inc

What to do next:

- In your development application, add the main application as a built-on application. For more information, see Adding built-on applications.
- Provide access to your application for development team members. For more information, see Learning about Access Groups.

- Building your first application

Adding development Branches to your application

To avoid conflicts and errors when team members work simultaneously, add development Branches to your application so that team members can work simultaneously on multiple features without risking conflicts and errors that can arise from working on the same Rules.

For example, one team member can fix bugs in the application UI while another developer fixes invalid Service-Level Agreement Rules. If both Rules refer to the same base Rule, working in separate Branches prevents conflicts during development.

Before a Pega Platform™ update, you can create an Update Branch to collect application Rules in a Branch that you can subsequently update in a single process after a Pega Platform update. By using the Update Branch, you avoid manual interaction with specific application Rule updates during or after a software update. Use this update strategy to simplify the merging of these updated Rules, which then makes these changes available to other developers in the application when they merge their changes into the application. For more information about a software update example, see Creating an unlocked application ruleset to manage updated rules.

1. In the header of Dev Studio, click the name of the application, and then click Definition.
2. In the **Development Branches** section, click Add Branch.
   You can add Branches that contain Rulesets only from your current application.

©2025 Pegasystems Inc

3. In the **Add a Branch ID** dialog box, in the **Branch name** field, enter a Branch name:
   - To create a new Branch, enter a unique name that starts with a letter.

     Ensure that the Branch name reflects the purpose so that your development team can easily identify the correct Branch in which to save changes during application development.

   - To reuse an existing Branch from your system, press the Down arrow key, and then select a Branch that you want to use.
4. Click **Submit**.
5. **Optional:** If you have multiple Branches, you can specify how the system selects the Rules for Rule resolution by reordering the list of Branches.
   For more information, see Reordering Branches.

   The system selects the Rules from the top Branches first.

6. **Optional:** If you use Live UI, to facilitate application development so that less technical users can modify common properties of a control or the layout of Rules in the Branch, set up and manage a run-time Branch.
   For more information, see Enabling run-time branching and editing.
7. Create Rules and add them to your Branch.
   The system automatically creates Rulesets where you save your Rules. For more information about working with Rules in Branches, see Rule development in Branches.

Result:

When you develop your application, you can select the Branch and Ruleset into which you want to save new Rules.

- **Final Rules and utility Functions in branched development**
- Branch operations

PEGA ©2025 Pegasystems Inc

Final Rules and utility Functions in branched development

To add Rules to Branches, when you create a Rule, you can select the Branch where you want to save your Rule. Branch Rulesets are automatically created.

Working with Rules in the Branch Ruleset is the same as in standard Rulesets, except for the following differences:

- Final Rules – You can typically modify final Rules only in the Ruleset and class in which they are declared final. This limitation means you cannot normally copy final Rules to another Ruleset that either has the same Applies To class or is located within the subclass of the Ruleset of the final Rule. For Branch Rulesets that are branched from the originating Ruleset of the final Rule, this limitation is removed. You can check out a final Rule from its original Ruleset into a Branch Ruleset that is branched from the owning Ruleset. For more information, see Checking out a Rule to a Branch.
- Utility Functions – Some Function references in the system use the Ruleset name as part of the reference. To allow for Rules to appropriately reference Functions that exist in Rules in a Branch Ruleset, the system's Function lookup considers the Branches in your Ruleset list when resolving the references.

- **Developing Branches with libraries and Functions**

- Branch operations

Developing Branches with libraries and Functions

Libraries hold Custom Functions for your application, which can supplant and extend standard Functions. For example, the OperatorHasPrivilege Function verifies that a user has a specific privilege.

However, you cannot create libraries in Branch Rulesets. Libraries are based on the Ruleset name only, not the Ruleset version. Because you must merge Branches into a particular Ruleset version, you cannot create libraries directly in Branch Rulesets.

PEGA ©2025 Pegasystems Inc

To use libraries and Functions in Branches, perform the following steps:

1. Create a library and save it in the base Ruleset.
2. Create the Functions that the library contains and save them into a Branch Ruleset.
3. Merge the Branch so that libraries and Functions are contained within the Branch.

- Final Rules and utility Functions in branched development
- Merging Branches into target Rulesets

Adding system Branches

You can add Branches that already exist on your system to your application. You cannot add a Branch that contains branched versions of a Ruleset that is not in your application stack.

1. In the navigation pane, click **App**, and then click **Branches**.
2. Right-click the application to which you want to add a system Branch and select **Add Branch from this system**.
3. Press the **Down Arrow** and select the system Branch that you want to add.
4. Click **Submit**.
5. If you are using multiple Branches, reorder the list of Branches so that it matches the order in which Rules should be resolved.
   For more information, see Reordering Branches.
6. Create Rules and add them to your Branch.
   When you create Rules, you can select the Branch and Ruleset into which you want to save them. Rulesets are automatically created. For more information, see Final Rules and utility Functions in branched development.

- Adding development Branches to your application

©2025 Pegasystems Inc

Creating Branch Rulesets

Provide your development team with an ability to work on multiple features in parallel by creating Branch Rulesets. When you save Rulesets in different Branches, team members can work on isolated Rulesets without impacting development of other features. As a result, you avoid overwriting results and generating errors.

> **Before you begin:**
>
> Enable Branch development. For more information, see Branches and Branch Rulesets.

1. In the header of Dev Studio, click **Create** > **SysAdmin** > **Ruleset**.
2. Create a new Ruleset or a Ruleset version:
   - To create a new Ruleset, in the **Ruleset Name** field, enter a unique name for your Ruleset.
   - To create a new Ruleset version, in the **Ruleset Name** field, press the Down arrow key, and then select the Ruleset for which you want to create a new version.

   > **Result:**
   >
   > The system autopopulates the version and description of your Ruleset.
3. Select the **Branch Ruleset** checkbox.
4. In the **Branch ID** list, select a Branch to store your Ruleset.
5. In the **Ruleset to Branch** list, select a Ruleset that you want to Branch.
6. Click **Create and open**.

> **Result:**
>
> The system creates a new Ruleset version and saves it in the Branch.

PEGA ©2025 Pegasystems Inc

What to do next:

Configure version settings for the Ruleset. For more information, see Configuring Ruleset version settings.

Related tasks

- Adding development Branches to your application

Branch operations

After you create Branches and develop Rules in Branch Rulesets, you can work with Branches in a number of ways. For example, you can create Branch reviews with other users, delete Branches form the system, and lock Branches before you merge them.

- **Setting Branch development preferences**
- **Viewing Branch quality and Branch contents**
- **Reordering Branches**
- **Locking a Branch**
- **Packaging a Branch**
- **Removing a Branch from an application**
- **Deleting a Branch from the system**

- Branches and Branch Rulesets
- Integrating with file and content management systems

Setting Branch development preferences

Define Branches where you want to save the changes that you make in App Studio, so that you can work on a feature without affecting other parts of your application.

©2025 Pegasystems Inc

For example, you can create a sandbox Branch where you can develop and test different Rules, and then merge or discard your changes. You can also create multiple Branches to work on different features or bugs within one application.

> **NOTE:** As a best practice, use Workspaces instead of Branch development preferences to perform private edits in your application. By default, Workspaces replace Branch development preferences. To set Branch development preferences, toggle Workspaces off. For more information, see Developing applications in Workspaces.

1. In the header of App Studio, click the **Toggle Branch development** icon.
2. In the **Branch development preferences** dialog box, toggle the Branch development on.
3. **Optional:** From the list, select an Application Layer where you want to make changes.
4. Define the Branch to save your changes:
   - From the list, select a Branch.
   - If you need a new Branch, click **Create a new Branch**, and then complete the dialog box.
5. Click **Submit**.
   By default, Dev Studio sets the context of new and updated Rules to the highest Branch in your application.

- Final Rules and utility Functions in branched development

Viewing Branch quality and Branch contents

You can view information about your Branch, such as the Rules that it contains and whether Branches have been reviewed. You can also view quality metrics such as guardrail warnings for information about the health of your Branch.

1. In the navigation pane of Dev Studio, click **App**.

©2025 Pegasystems Inc

2. Click **Branches**.
3. Click a Branch, and then click the **Content** tab to view the Rules that the Branch contains.
   You can also see if a Branch has been reviewed and can open a review in the Branch.
4. Click the **Branch quality** tab to view guardrails, merge conflicts, and information about the unit tests that are configured on the Rules in the Branch.

- **Understanding Branch quality metrics**

- Branches and Branch Rulesets
- Understanding unit test Cases

Understanding Branch quality metrics

You can view quality metrics to monitor the health of your Branch.

Branch quality metrics include the following information:

Guardrails

Monitor the number and severity of guardrail violations that were found in your Branch. By investigating and resolving the violations, you can improve your Branch's overall quality.

The **Guardrails** section displays the following information:

- **Weighted score**: The score calculated by comparing guardrail warnings (weighted by severity and justification) against the total number of Rules in the Branch.
- **Warnings**: The total number of guardrail warnings in the Branch.
- **Severe**: The number of severe guardrail warnings in the Branch.

For more details about individual guardrail violations, select the **Warnings** tab.

PEGA ©2025 Pegasystems Inc

Test coverage

Check how many of your Branch Rules are covered by tests of any kind.

The **Test coverage** section displays the following information:

- Rules covered: The percentage of the Branch Rules covered by tests.

For more details about individual Rules, select the Uncovered Rules tab.

Unit testing

Monitor unit test results and see how many of your Branch Rules have unit test Cases written for them.

The **Unit testing** section displays the following information:

- Rules with unit tests: The percentage of the Branch Rules with unit tests.
- Test pass rate: The percentage of unit tests written for the Branch Rules that passed.

For more details, select the Failed unit tests and Rules without unit tests tabs.

Merge conflicts

Investigate any potential merge conflicts to identify problems with your Branch as early as possible.

The **Merge conflicts** tab displays the following information:

- Total: The total number of potential merge conflicts between the Branch and the main application.
- Name: The name of a Rule that is causing a potential merge conflict.
- Rule Type: The type of Rule that is causing a potential merge conflict.
- Base Rule: The fallback Rule that is selected by the conflicted Rule's resolution.

PEGA ©2025 Pegasystems Inc

Warnings

Investigate individual guardrail violation warnings to identify problems related to your Branch.

The **Warnings** tab displays the following information:

- Total: The total number of guardrail warnings in the Branch.
- Severe: The number of severe guardrail warnings in the Branch.
- Moderate: The number of moderate guardrail warnings in the Branch.
- Rule name: The name of a Rule that is causing a guardrail warning.
- Rule Type: The type of Rule that is causing the guardrail warning.
- Ruleset: The Ruleset that contains the Rule that is causing a guardrail warning.
- Severity: The severity status of the guardrail warning.
- Warning type: The type of the guardrail warning.
- Justified: The justification status of the guardrail warning.
- Warning message: The guardrail warning message.
- Introduced by: The user who introduced the Rule that is causing the guardrail warning.

Uncovered Rules

Investigate individual Rules that are still uncovered by tests of any kind.

The **Uncovered Rules** tab displays the following information:

- Executable Rules: The total number of executable Rules in the Branch.
- Uncovered Rules: The number of Branch Rules that are not covered by tests.
- Rule name: The name of a Rule that is not covered by unit tests.
- Rule Type: The type of Rule that is not covered by unit tests.
- Class: The class of the Rule that is not covered by unit tests.
- Ruleset: The Ruleset that contains the Rule not covered by unit tests.

PEGA ©2025 Pegasystems Inc

Failed unit tests

Determine the causes of individual unit test failures.

The **Failed unit tests** tab displays the following information:

- Total unit tests: The total number of unit tests in the Branch.
- Failed unit tests: The number of failed unit tests in the Branch.
- Test Case name: The name of a failed unit test Case.
- Rule Type: The type of Rule that failed the unit test Case.
- Rule name: The name of the Rule that failed the unit test Case.
- Class: The class of the Rule that failed the unit test Case.
- Last run: The time when the failed unit test Case was last run.
- Result: The result of the failed unit test Case.
- Run history: The run history of the failed unit test Case.

Rules without unit tests

Investigate the details of Rules that are supported by unit testing but that still do not have unit test Cases written for them.

The **Rules without unit tests** tab displays the following information:

- Supported Rules: The number of Branch Rules that are supported by unit tests.
- Rules without unit tests: The number of supported Rules without unit tests.
- Rule name: The name of the Rule without a unit test.
- Rule Type: The type of Rule without a unit test.
- Ruleset: The Ruleset that contains the Rule without a unit test.

- Viewing Branch quality and Branch contents
- Justifying an application warning
- Base Rules

PEGA ©2025 Pegasystems Inc

Reordering Branches

If you use multiple Branches in your application, ensure that the order of the Branches in the application Rule form matches the order in which Rules should be resolved for this application.

1. To open the application Rule form, click the application name in the Dev Studio header, and then click **Definition**.
2. On the **Definition** tab, in the **Development Branches** Branches section, drag a Branch to move it. When you log in to this application, the system assembles the Ruleset list and Branch Rulesets according to the sequence in which the Branches are listed.
3. **Optional:** If you are using Live UI, you can set up and manage a run-time Branch so that less technical users can modify common properties of a control or a layout on Rules in the Branch. See Enabling run-time branching and editing for more information.
4. Click **Save**.

- Adding development Branches to your application
- Branches and Branch Rulesets

Locking a Branch

Before you merge a Branch, lock the Branch after development is complete so that no further modifications can be made to the Rulesets within the Branch. When you lock a Branch, all Rulesets within the Branch are locked.

To ensure that developers follow best practices of locking a Branch, you can override the *AllowMergeWhenUnLocked* When Rule. As a result, locking Branches is mandatory before running the Merge Branches wizard. For more information, see Branch merge process customization.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.

©2025 Pegasystems Inc

3. Right-click the Branch that you want to lock and select **Lock**.
4. Enter a password to lock the Branch.
5. Reenter the password to confirm it.
6. Click **Lock**.

- Branches and Branch Rulesets

Packaging a Branch

To retain a copy of your Branch, you can package the contents of a Branch in a .jar file. Data instances that are tagged with the Rulesets that are associated with the Branch and history record are included in the package.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.
3. Right-click the Branch that you want to package and click **Package**. A browser window displays the progress of the packaging process.
4. Click the link in the browser window to save the .jar file to a local directory. If any Rules in the Branch Rulesets are checked out, the latest version of the checked-in Rule is included in the package.

- Branches and Branch Rulesets

Removing a Branch from an application

When you remove a Branch, you remove an association between a Branch and an application. Removed Branches and their Rulesets are not available when you create or copy records in the application.

> **NOTE:** Removing a Branch does not delete its contents from the system. To delete a Branch from the system, see Deleting a Branch from the system.

1. In the navigation pane of Dev Studio, click **App**.

©2025 Pegasystems Inc

2. Click **Branches**.
3. Right-click the Branch that you want to remove from your application and click **Remove from application**.

- Branches and Branch Rulesets

Deleting a Branch from the system

You can delete an entire Branch and its contents from the system. You can also package the Branch in a ZIP file and download it before deleting it.

1. Check in any records that are checked out.
2. In the navigation pane of Dev Studio, click **App**.
3. Click **Branches**.
4. Click the Branch that you want to delete, as shown in the following figure:

Opening a Branch

5. In the **Brnach** form header, click **Actions** > **Delete from system**, as the following figure shows:

©2025 Pegasystems Inc

Deleting a Branch

6. **Optional:** If there are Rulesets in the Branch, and you want to save them in a ZIP file that you can download, select the **Save Branch contents in a ZIP fie for download** checkbox, as shown in the following figure:

Selecting Rulesets to download

7. In the **Delete Branch** dialog box, click **Delete**, and then, when the process is complete, click **Done**.

- Branches and Branch Rulesets

PEGA ©2025 Pegasystems Inc

Branch reviews

To increase the quality of your application, you can create reviews of Branch contents to improve Branch quality by, for example, ensuring that the Rules are guardrail-compliant. You can assign Branch reviews to other users, use Pulse to collaborate on reviews, and close reviews after you have addressed any issues.

If you have a Default email account configured, you are notified when reviews are assigned to you and when reviews are closed. For more information, see Configuring outbound email.

If Pulse notifications are configured, you can receive emails when a new comment is added or when a reply is posted to a new comment. For more information, see Configuring Pulse email notifications.

You can override the following When Rules to allow merges when Branches are unlocked or not reviewed:

- pyAllowMergeWhenNotReviewed: Override to disallow merges when Branches have not been reviewed.
- pyAllowMergeWhenUnlocked: Override to disallow merges when Branches are not locked.

- **Creating a Branch review**
- **Modifying a Branch review**
- **Deleting a Branch review**
- **Closing a Branch review**
- **Reopening a Branch review**
- **Reviewing Branches**

PEGA ©2025 Pegasystems Inc

- Branches and Branch Rulesets

Creating a Branch review

You can create a review for a Branch so that you and other review team members can collaborate to improve the Branch quality before you merge the Branch. Only one Branch review can be open at a time.

If you have a Default email account configured, you are notified when reviews are assigned to you and when reviews are closed. For more information, see Configuring outbound email.

Complete the following steps:

1. As a best practice, check in any Rules that are checked out.
2. In the navigation pane of Dev Studio, click **App**.
3. Click **Branches**.
4. Click the Branch for which you want to create the review.
5. On the **Content** tab, click **Create review**.
6. In the **Create Review** dialog box, in the **Objective** field, enter the purpose of the review.
7. In the **Due** field, specify the date on which the review should be completed.
8. In the **User ID** field, press the Down arrow key and select the user ID of a reviewer.
9. Add more reviewers by clicking the **Plus** sign and repeating the previous step.
10. Click **Create**.

- Branch reviews
- Branches and Branch Rulesets

Modifying a Branch review

For open Branch reviews, you can modify the list of users who are assigned to the review.

©2025 Pegasystems Inc

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.
3. Click the Branch for which you want to modify the Branch review.
4. On the **Content** tab, in the **Current review** section, click the review to open it.
5. Select **Actions** > **Manage review team**.
6. Add or remove reviewers, as appropriate.
7. Click **OK**.

- Branch reviews
- Branches and Branch Rulesets

Deleting a Branch review

You can delete both open and closed Branch reviews.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.
3. Click the Branch for which you want to delete the review.
4. On the **Content** tab, in the **Current reviews** section, click the review to open it.
5. Click **Actions** > **Delete**.

- Branch reviews
- Branches and Branch Rulesets

Closing a Branch review

You can close a Branch review after all comments have been addressed.

If you have a Default email account configured, you are notified when reviews are assigned to you and when reviews are closed. For more information, see Configuring outbound email.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.

©2025 Pegasystems Inc

3. Click the Branch for which you want to close the review.
4. On the **Content** tab, in the **Current review** section, click the review to open it.
5. Click **Actions** > **Close**.
6. A dialog box is displayed if Branches in the Rule are checked out. Click **Yes** to close the review.
7. Click **Submit**.

- Branch reviews
- Branches and Branch Rulesets

Reopening a Branch review

You can reopen a Branch review after the review was closed, for example, if a reviewer inadvertently closed it.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.
3. Click the Branch for which you want to reopen the Branch review.
4. On the **Content** tab, in the **Previous reviews** section, click the review to open it.
5. Select **Actions** > **Reopen**.
6. In the dialog box, click **Yes** to confirm that you want to reopen the review.

- Branch reviews
- Branches and Branch Rulesets

Reviewing Branches

If you are a Branch reviewer, you can open reviews and inspect the Rules within the Branch improve the quality of the Branch, for example, before you merge its contents. You can also communicate with other users in Pulse to work collaboratively on Branch reviews.

If you have a Default email account configured, you are notified when a review is assigned to you and when reviews are closed. For more information, see Configuring

©2025 Pegasystems Inc

outbound email. If Pulse notifications are configured, you can receive emails when a new comment is added or when a reply is posted to a new comment. For more information, see Configuring Pulse email notifications.

1. In the navigation pane of Dev Studio, click **App**.
2. Click **Branches**.
3. Click the Branch that has the review that you want to open.
4. On the **Content** tab, in the **Current review section**, click the review to open it. You can perform the following actions:
   - Filter the list of Rules in the Branch by Rule Type using the drop-down list.
   - Click **General comments** to post and view Pulse comments about the Branch.
   - Click a Rule to collaborate with other users in Pulse about the specific Rule. You can open the Rule form on this page.

- Branch reviews
- Branches and Branch Rulesets

Merging Branches into target Rulesets

When Rule development in a Branch is complete, make the changes available by merging Branches into a target Ruleset of the development application. As a result, your team provides required solutions in a timely and efficient way, without the risk of overriding or losing work.

> **Before you begin:**
>
> Check all Rules into their base Rulesets before you merge Branches. For more information, see Checking in a Rule.

As a best practice, the system administrator creates a new Ruleset version for the base Ruleset, independent of the wizard. Individual teams develop Rules in their specific Branches, and then merge those Branches into the existing base Ruleset version that

PEGA ©2025 Pegasystems Inc

the administrator provides. As a result, administrators have better control over versions.

Before a Pega Platform™ update, you can update your Branch to avoid manual interaction with specific application Rule updates during or after a software update. Use this update strategy to simplify the merging of these updated Rules, which then makes these changes available to other developers in the application when they merge their changes into the application. For more information about a software update example, see Creating an unlocked application ruleset to manage updated rules.

1. **Optional:** To save time and resolve any potential issues before merging Rulesets, check whether any conflicts might occur.
   For more information, see Viewing Branch information.
2. **Optional:** To ensure that Branches include only changes that you want to merge, lock a Branch after development is complete.

   > **NOTE:** If the Merge Branches wizard encounters an error and fails during the merge process, the Rules that were merged before the error are merged into the target Ruleset. The merge process skips the remaining Rules. Consequently, the target Ruleset is incomplete and only partially merged. To avoid such behavior, as a best practice, lock Branches before merging. For more information, see Locking a Branch.

   To ensure that developers follow best practices of locking a Branch, you can override the *AllowMergeWhenUnLocked* When Rule. As a result, locking Branches is mandatory before running the Merge Branches wizard. For more information, see Branch merge process customization.

3. In the navigation pane of Dev Studio, click **App**.
4. In the App Explorer, click the **Branches** tab.
5. Select Branches that you want to merge:

©2025 Pegasystems Inc

Choices	Actions
Merge a single Branch	a. Right-click the Branch, and then click Merge. b. In the **Merge Branch** dialog box, click Proceed.
Merge multiple Branches	a. Right-click your application name, and then click Merge multiple Branches. b. In the **Select Branches to Merge** dialog box, select the Branches that you want to use. c. Click OK.

6. If conflicts or warnings occur, on the Merge Branches tab, review information about any issues.
   You must resolve conflicts before you can merge Branches. For more information, see Conflicts and warnings in the Merge Branches wizard.
7. In the **Target Ruleset** section, in the list of versions, select the base Ruleset version into which you want to merge Rulesets:
   - To create a new Ruleset version in the base Ruleset during the merge, select Create new version.
   - To reuse an existing Ruleset, select the Ruleset version.
8. Click Merge.

> **Result:**
>
> When a merge is complete, you can view the details about the outcome, the number of merged Rules, and the source and target Rulesets.

- **Conflicts and warnings in the Merge Branches wizard**
- **Branch merge process customization**

©2025 Pegasystems Inc

- Branches and Branch Rulesets

Conflicts and warnings in the Merge Branches wizard

The Conflicts and Warnings window displays an expandable list of Rules that the Merge Branches wizard has identified as having conflicts, warnings, or both. To open this window, click the displayed number of conflicts and warnings on the main wizard screen.

> **NOTE:** The displayed report of conflicts and warnings depends on the target Ruleset version selected for the merge. For example, merging a Rule into target Ruleset version 01-01-01 might display no conflicts and one warning, while merging that Rule into target Ruleset version 01-01-02 might display one conflict and no warnings. You must select the target Ruleset version for the merge to get an accurate report of the potential conflicts and warnings.

When there are resolved conflicts, the report refreshes to indicate how many are resolved.

- **Conflicts in the Merge Branches wizard**
- **Resolving conflicts in the Merge Branches wizard**
- **Addressing warnings in the Merge Branches wizard**
- Merging Branches into target Rulesets

Conflicts in the Merge Branches wizard

Before the wizard proceeds with the merge, you must indicate in this window that you have resolved all conflicts reported by the wizard. Conflicts occur for the following situations:

PEGA ©2025 Pegasystems Inc

Conflict situation	Example of how this occurs
The Rule in a lower base Ruleset version has been recently updated.	Base Ruleset XYZ has two locked Ruleset versions, 01-01-01 and 01-01-02. TeamA branches XYZ into their Story-A Branch, and TeamB branches XYZ into their BugFix-1 Branch. TeamB's work is a bug fix for version 01-01-01, and they complete their work and merge their BugFix-1 Branch with the changed Rules for base Ruleset XYZ into 01-01-01. Then TeamA starts the wizard to merge their Story-A Branch by creating a new version for base Ruleset XYZ. The wizard reports a conflict because the Rule was updated in Ruleset version 01-01-01 (a lower version) by TeamB prior to TeamA's merge.
The Rule in the selected target (base) Ruleset version has been recently updated.	Base Ruleset XYZ has two locked Ruleset versions, 01-01-01 and 01-01-02. TeamA branches XYZ into their Story-A Branch, and TeamB branches XYZ into their Story-B Branch. TeamB completes their work and merges their Story-B Branch into 01-01-02. Then TeamA starts the wizard to merge their Story-A Branch into 01-01-02 also. The wizard reports a conflict because the Rule was updated in the same Ruleset version 01-01-02 (TeamA's selected target version) by TeamB prior to TeamA's merge.
The same Rule exists in multiple Branches that are merged using the wizard to merge multiple Branches.	Base Ruleset XYZ has one version, 01-01-01. TeamA has Branches Story-A and Story-B, which have the same Rule. Story-A is the topmost Branch in the hierarchy. Story-A and

©2025 Pegasystems Inc

Conflict situation	Example of how this occurs
	Story-B are being merged using multibranch merge. The wizard reports a conflict for Story-A Branch because Story-B has the same Rule and will be merged before Story-A. To resolve this conflict, move all your changes for this Rule from the Story-B Branch to the Rule in Story-A.

- Resolving conflicts in the Merge Branches wizard
- Conflicts and warnings in the Merge Branches wizard
- Merging Branches into target Rulesets

Resolving conflicts in the Merge Branches wizard

To resolve the conflicts for a Rule, complete the following steps:

1. Examine the conflicts reported for the Rule in the **Conflicts and Warnings** window.
2. Click **Compare** next to the first conflict to examine the differences between your Branch copy of the Rule and the one in the base Ruleset. Alternatively, click the Ruleset name to open the base Rule and directly examine it.
3. Review and resolve all of the differences between the base Rule and your Branch Rule.
    - If the changes are minor (such as a typo), open your Branch Rule and apply the same changes to it so that it matches the base Rule.
    - If the changes are more complex, or if they conflict with the choices or logic in your Branch Rule, contact the individual who modified the base Rule and negotiate what the final set of changes should be and the testing procedure for testing them. After reaching a mutual decision, make the agreed-upon changes in your Branch Rule and test.

©2025 Pegasystems Inc

4. When changes to your Branch Rule are completed and tested, select the Mark as Resolved checkbox to indicate you have resolved the conflict.

Result:

After all conflicts reported in the **Conflicts and Warnings** window are marked resolved, you can click **OK** to close the window and return to the Merge Branches wizard to continue with the merge process.

In the main screen of the wizard, you can see if all conflicts are resolved by verifying if the number of Resolved conflicts matches the number of reported conflicts for all of the Branch Rulesets.

- Conflicts in the Merge Branches wizard
- Comparing Rule versions

Addressing warnings in the Merge Branches wizard

Warnings are primarily informational and are provided to make you aware of how merging your Branch might impact other teams or might not behave exactly as you think it will.

Warning situation	Example of how this occurs	Recommended steps
A Rule with the same keys as the Branch Rule exists in a higher Ruleset version in the target Ruleset.	You are merging the Branch Rule into target Ruleset version 01-01-01, and the corresponding base Rule also exists in target Ruleset version 01-01-02.	1. Click **Compare** next to the warning to examine the differences between your Branch copy of the Rule and the one in the other base

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		Ruleset version. Alternatively, click the Ruleset name next to the caution icon to open the other Rule and directly examine it. 2. Review all differences between the Rules, and use the Related Rules display to assess the impact of the differences. If the features in the Rule in the higher Ruleset version does not apply to your target Ruleset version, and your changes in the lower target are not present in, or will not impact, the one in the higher Ruleset version, it is likely safe to proceed with your merge. Merge into the lower Ruleset version. The changes in the Rule will work at that

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		Ruleset version level. Those applications that use the higher Ruleset versions might not see your merged changes. For changes in your Branch Rule that you determine need to be ported to the Rule in the higher Ruleset version, define a new development application and matching Branch using the higher Ruleset version, and copy the base Rule from the higher Ruleset version into that Branch Ruleset. Make your changes in that Branch, test, and merge to the higher Ruleset version.
The corresponding base Rule exists in another	Another team has branched the same base Ruleset that you are merging, and has a	1. Click **Compare** next to the warning to examine the

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
Branch Ruleset in the system.	copy of the base Rule in their Branch Ruleset.	differences between your Branch copy of the Rule and the one in the other Branch. Alternatively, click the Ruleset name next to the caution icon to open the other Rule and directly examine it. 2. Review all differences between the Rules, and use the Related Rules display to assess the impact of the differences. 3. Contact the owner of the other Branch, or the author of the other branched Rule, and describe the changes you made to the Rule in your Branch and how they can test for the changes. Best practice is they would validate your changes after your merge your Branch and they obtain your changes

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		and include them in their Branch. 4. Complete the merge of your Branch.
The corresponding base Rule exists in another Branch Ruleset, which is branched from a different Ruleset than your Branch Ruleset.	In this situation, the Rule exists in another Branch Ruleset which will later be merged into a different Ruleset than the one the corresponding base Rule is in. A team has Branched a base Ruleset that is different than the one you are merging, and has copied the base Rule that corresponds to your Branch Rule into their Branch Ruleset.	1. Click **Compare** next to the warning to examine the differences between your Branch copy of the Rule and the one in the other Branch. Alternatively, click the Ruleset name next to the caution icon to open the other Rule and directly examine it. 2. Review all differences between the Rules, and use the Related Rules display to assess the impact of the differences. 3. Contact the owner of the other Branch, or the author of the other branched Rule, and discuss why a Rule

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		with the same name exists in two different Rulesets, and determine whether there are reasons for both to exist or whether one should be removed. • If the base Rulesets are for different purposes and will not be run concurrently, then continue with your merge. • If the teams agree the Rulesets are to be combined, negotiate an appropriate conclusion and perform agreed-upon changes. Depending on the outcome, complete your

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		merge if appropriate.
A Rule with the same keys has been updated somewhere else in the system.	A Rule exists in the system that has the same keys as the Rule you are merging, and the other Rule has been updated.	1. Click **Compare** next to the warning to examine the differences between your Branch copy of the Rule and other Rule. Alternatively, click the Ruleset name next to the caution icon to open the other Rule and directly examine it. 2. Review all differences between the Rules, and use the **Related Rules** display to assess the impact of the differences. 3. Contact the owner of the other Ruleset, or the author of the other Rule, and discuss why a Rule with the same name exists in two different Rulesets, and determine whether

©2025 Pegasystems Inc

Warning situation	Example of how this occurs	Recommended steps
		there are reasons for both to exist or whether one should be removed. • If the base Rulesets are for different purposes and will not be run concurrently, then continue with your merge. • If the teams agree the Rulesets are to be combined, negotiate an appropriate conclusion and perform agreed-upon changes. Depending on the outcome, complete your merge if appropriate.

- Conflicts and warnings in the Merge Branches wizard

©2025 Pegasystems Inc

Branch merge process customization

You can enhance the merge process for your development team in various ways to comply with your organization's policies and procedures. By using extension points, you customize the standard Branch merge process. For example, you can add postprocessing behavior that sends an email notification to the project lead with the status of the merge process. In another example, you can make locking a Branch before merge mandatory to avoid merge conflicts and Rule overrides.

The merge process and its extension points

When development in a Branch is complete and stable, you can use the Merge Branches wizard to merge the changes in one or more Branches to the base Rulesets.

> **TIP:** As a best practice, use the customizable and model-driven pipelines of Deployment Manager to validate a Branch before a merge. For more information, see Deployment Manager for model-driven DevOps.

The merge process is defined by the standard flow Rule *pzMergeInstances* in the *PegaAccel-Management-Refactor-Instances-Merge* class.

©2025 Pegasystems Inc

©2025 Pegasystems Inc

For preprocessing and postprocessing behavior, the process has two extension points (as subprocesses), to which you can add custom steps.

By saving a copy of the standard Rule and customizing your copy, you can override the generic behavior. For more information about making a copy of the standard Rule, see the Creating a class to organize your customizations section.

The standard merge process provides extension points, at which you can provide new behavior:

pyPreMergeInstances (subprocess)
: To add preprocessing behavior. By customizing a copy of *pyPreMergeInstances*, you can provide processing that occurs before the Merge Branches wizard starts, such as verifying that a user's operator ID is performing the merge instead of a generic account ID.

pyPostMergeInstances (subprocess)
: To add postprocessing behavior. By customizing a copy of *pyPostMergeInstances*, you can provide processing that occurs after the Merge Branches wizard completes, such as sending an email notifying the team about the status of the merge.

pyValidateMergeOptions (activity)
: To customize validation on the inputs into the user interface of the Merge Branches wizard.

pyAllowMergeWhenUnLocked (When Rule)
: To make locking Branches before running the Merge Branches wizard mandatory. By customizing a copy of *pyAllowMergeWhenUnLocked* and setting its value to **False**, you minimize the risk of the Merge Branches wizard failures due to some change that another developer might introduce when the merge process is in progress.

> **NOTE:** If the Merge Branches wizard encounters an error and fails during the merge process, the Rules that were merged before the error are merged into the target Ruleset. The merge process skips the remaining Rules. Consequently, the target Ruleset is incomplete and only partially

©2025 Pegasystems Inc

> merged. To avoid such behavior, as a best practice, lock Branches before merging. For more information, see Locking a Branch.

You can also modify the UI for the merge process by overriding the following Sections and When Rules in the Merge Branches wizard:

pyMergeInstancesRuleSetListFooter (Section)
: To customize the pre-merge screen by adding fields or information to the bottom of the merge wizard.

pyConfirmContent (Section)
: To customize the confirmation screen. You can modify this Section to show a message to the user after the user submits the merge or when a user returns the merge work object to review.

pyRestrictedTargetOptions (When Rule)
: To apply custom logic for merge options, such as Hide Merge Options, which determines which target Rulesets and passwords are visible.

pyMergeButtonDisabled (When Rule)
: To apply custom logic for when the merge is enabled on the UI, such as the Disable Merge Button option.

Creating a class to organize your customizations

The recommended action before extending the Merge wizard is to create a class to store all your extensions. By creating a class to add your customizations, you can expose different merge behaviors and enable users to fall back to an existing behavior.

> **TIP:** You can create a new component to expose your merge wizard customizations, which provides a way to package and share your customization between different applications. For more information, see Components.

©2025 Pegasystems Inc

To safely organize your customizations in a class in which you can override the extension points, ensure that:

- The class is a child of the *PegaAccel-Management-Refactor-Instances-Merge* class.
- You update the *pySetCustomMergeExperience* activity to conditionally set the class of the Primary Page by using the page-set-class method.
- If multiple merge wizard implementations are required, include this logic in the highest Ruleset version available.

For more training materials, see the Merging a development Branch in 8.6 challenge on Pega Academy.

Related concepts

- Branches and Branch Rulesets

Related tasks

- Merging Branches into target Rulesets

Branches and unlocked Rulesets

Based on your development model, you can build your application through branched development or by using unlocked Rulesets. Before you select an approach, analyze which option suits your requirements best.

Working in Branches

Branches allow developers to work in separated areas, which helps avoid the problem of overwriting someone else's changes and minimizes the occurrence of conflicts and errors. When a developer or developers make changes in a Branch, these changes have no effect on other Branch Rulesets or base application Rulesets. As a result, developers can work on multiple features in parallel, even if the features use the same base Rules. This saves time and accelerates application development.

PEGA ©2025 Pegasystems Inc

As a best practice, to start branched development, create team applications by copying your base application. When every team has a separate application, implementing and testing changes is more convenient compared to a scenario in which multiple teams work on the same Ruleset in multiple Branches. For more information, see Creating a development application.

When the development is complete, teams merge changes to the base application. By merging, developers can select only those changes that they intend to implement in the base application, and discard other Rules. Merging provides greater control over changes that developers implement in the base application. For more information, see Merging Branches into target Rulesets.

Working in unlocked Rulesets

Unlocked Rulesets are available to multiple developers at the same time. When developers work in an unlocked Ruleset, the implemented changes immediately affect other developers who use the affected Rules in the same Ruleset. This approach might lead to work overwrites, errors, and unexpected results. A solution to minimize the risk of overwrites is Rule checkout. When a developer checks out a Rule, the Rule is unavailable to other developers for editing. However, after implementing their changes, a developer checks in a Rule, and only then do the changes affect other Rules in the Ruleset. For more information, see Checking out a Rule and Checking in a Rule.

Application development for larger projects

The most common scenario is application development that involves multiple teams working on multiple features. In such scenarios, branched development is the most suitable choice. Developers can implement changes simultaneously, because changes in one Branch have no effect on other Branch Rulesets, which minimizes errors and work overwrites. Parallel development of multiple features saves time and helps developers create software faster.

Branched development involves merging, which developers can use to select Rules to include in the base application, and remove unnecessary Rules. Also, validation of new

PEGA ©2025 Pegasystems Inc

features in an environment that does not include the ongoing work of other developers is more effective and reliable. With branched development and merging, the process of reverting changes is easier than when teams work in unlocked Rulesets. Merging is also applicable in situations in which changes require approval, for example from a manager or a team lead.

Rule checkout during branched development is a best practice. By checking out Rules, developers avoid situations in which two or more team members edit the same Rule. Rule checkout also creates a Rule history, which developers can use to analyze changes in the Rule, or to revert the Rule to a specific state.

Branched development supports continuous integration and continuous delivery (CI/CD) pipelines, so that developers deliver well-tested software through an agile and iterative development process. For more information about CI/CD pipelines, see Understanding continuous integration and Understanding continuous delivery.

Application development for a single developer and small-scale teams

When only one person works on an application, or the project involves a small and communicative team of up to four or five members, branched development and Rule checkout are not required, because the implemented changes do not affect other developers. For a small development team, communicating the changes that the developers intend to implement is crucial to avoid work overwrites and errors. This business scenario is untypical and usually applies in situations when a developer builds an application for learning purposes, or creates an application as a citizen developer. Typically, citizen developers have limited technical background and focus on more granular applications. Similar scenarios include the very early stages of application development process, when a client and a software company define requirements for a new application.

> **NOTE:** Rule checkout creates a history of all changes that occur in a Rule. Developers can use this history to analyze the edits in a Rule, or to roll back

©2025 Pegasystems Inc

> the Rule to a specific state. As a best practice, check out the Rules that you want to edit every time you intend to edit them.

Working in unlocked Rulesets is a good approach for development projects that include very limited CI/CD pipelines. As a best practice, if you intent to move an application to a production environment, implement branched development first.

- Branches and Branch Rulesets
- Defining the security of a Ruleset
- Checking out a Rule
- Checking in a Rule

Branches and Branch Rulesets

Branches help you manage work in development environments in which multiple teams contribute to a single application. You use Branches to develop software simultaneously in a version-controlled environment. For example, a team can develop a feature in one Branch while another team develops another feature in a different Branch, even if the teams share the same Rulesets.

When you use Branches, different development teams can work without interfering with changes that other teams make in an application, even if multiple teams share one base Ruleset. For example, Team A can develop Service-Level Agreements for a Case Type while Team B fixes UI bugs in the application. When you provide separate Branches for both teams, you minimize the risk of errors and work overwrites that might arise when both teams work on the same Ruleset simultaneously.

To organize Rules better during your development process, you can categorize Rules into Branch Rulesets. Branch Rulesets have the following characteristics:

- Branch Rulesets are branched from Rulesets in your application.
- Branch Rulesets contain Rules that are in active development in an associated Branch.

PEGA ©2025 Pegasystems Inc

- Branch Rulesets have a naming convention that includes a Ruleset ID, the word `Branch`, and a Branch name, for example, `SLAS_Branch_TeamABranch`, where `SLAS` is a Ruleset ID, `Branch` indicates a Branch Ruleset, and `TeamABranch` is a Branch name.

Branching is especially useful in large development projects when multiple teams work simultaneously on the Rules in an application, and members of one team view each others work, but isolate their development changes from other teams. Consequently, Rule changes that one team makes do not affect the other teams until the changes are stable, conflicts are resolved, and the approval is granted to make the new improvements available to the entire development project team.

Branched development includes the following main tasks:

1. You create a development application that your team can use to develop new features. You create a development application by copying an existing application to ensure that the names of classes and Rulesets remain unchanged.

   For more information, see Creating a development application.

2. You add development Branches to the development application. Teams use Branches to develop features independently.

   For more information, see Adding development Branches to your application.

3. As a best practice, you create a Branch review to ensure that Rules in your Branches are guardrail-compliant and meet your business objectives.

   For more information, see Branch reviews.

4. You make new features available in your application by merging changes from Branches into a target Ruleset. You now can resolve conflicts and address warnings that might arise among different branched versions of the same Ruleset.

   For more information, see Merging Branches into target Rulesets.

PEGA ©2025 Pegasystems Inc

For example:

The following figure shows a sample scenario in which multiple development teams create Service-Level Agreements (SLAs) and resolve UI bugs in a Loan Requests application. An administrator creates two development applications: one for the UI teams and one for the SLAs teams. In each application, the administrator creates two Branches that contain Branch Rulesets. As a result, two teams can resolve UI bugs and two teams can develop SLA Rules without affecting another team. After the development is complete, the administrator first reviews and then merges Branches into target Rulesets. As a result, the Loan Requests application includes new features from the simultaneous work of four development teams.

Branched development

Branches and unlocked Rulesets

An alternative to branched development is working in unlocked Rulesets. Unlocked Rulesets are suitable for projects that typically involve one team, and team members immediately need to view changes that any team members make. When a developer saves a change in an unlocked Ruleset, the change immediately affects the work of other developers on that application.

©2025 Pegasystems Inc

During branched application development, a change that a developer makes in a Branch Ruleset affects the work that other developers do on the same development application only after merging changes into a target Ruleset.

For more information, see Branches and unlocked Rulesets.

- Adding development Branches to your application

Developing applications in Workspaces

Workspaces help you develop your application faster and more seamlessly, decreasing the risk of Rule conflicts and overrides when multiple developers work on the same application. Changes that you make in a Workspace are private and not visible to any other team member until you share them to a Branch.

Workspaces are operator-specific, which means that a Workspace and the changes that the Workspace includes are visible only to the Operator using that Workspace. You make the changes visible to other team members by sharing them to a Branch, and then generally available by merging the Branch to an application, and then releasing the application, as the following scheme shows:

Stages of feature development

Workspaces have the following characteristics:

- Each Workspace has only one owner - the operator in whose context the Workspace is created.
- Each Workspace is exclusively associated with one application.

©2025 Pegasystems Inc

- Within one application, each operator with developer access has one Workspace created by default.
- Each operator can create multiple Workspaces within one application.

  As a best practice, before creating a new Workspace, consider reusing an existing Workspace to perform the work.

- Each Workspace shows only the changes that it includes, and does not show changes that other operators perform in other Workspaces.
- Operators can switch between their own Workspaces.
- Operators can open changed Rules directly from the list of changes in a **Workspace** pane.

  Some Rule types can be only opened in Dev Studio. The ability to open those Rules depends on whether an operator can access Dev Studio. For more information, see Granting Portal access to an Access Group.

- Workspaces are available both in Dev Studio and App Studio.
- Workspaces are turned off by default. To start using Workspaces, toggle them off by using the *EnableWorkspace* toggle. For more information, see Toggling features on and off.
- When an operator creates a new Rule in Dev Studio and Workspaces are enabled, in the **Context** section, the operator can select the Application Layer in which to store the Rule, as shown in the following figure:

©2025 Pegasystems Inc

Setting context for the Rule

For more information, see Adding built-on applications and Creating Rules.

Related concepts

- Learning about Access Groups
- Role permissions and privileges in Access Manager

Enabling Workspaces

Start working on your private changes by enabling Workspaces in your authoring environment.

©2025 Pegasystems Inc

Before you begin:

By default, Workspaces are turned off in your authoring environment. To start using Workspaces, turn the *EnbaleWorkspace* toggle on. For more information, see Toggling features on and off.

1. In the header of App Studio, click the **Toggle Workspace development** icon to open the **Workspace** pane.
2. In the **Workspace** pane, turn on the **Enable Workspace** switch.

Result:

The **Toggle Workspace development** icon displays a green dot. When you enable Workspace development for the first time, your default Workspace opens in the **Workspace** pane. Otherwise, the Workspace that you used previously opens, as in the following figure:

©2025 Pegasystems Inc

Operator-specific Workspace

A Workspace includes the following elements:

Number	Description
1	A unique name
2	A menu to perform various operations, such as switching between Workspaces
3	A button to refresh a list of changes
4	A button to share changes to a Branch

©2025 Pegasystems Inc

Number	Description
5	A list of modified Rules, including a Rule name, a modification type, and a Rule type

Creating a Workspace

Work on multiple features within one application seamlessly by creating multiple Workspaces. All changes that you perform within a Workspace are private and have no effect on other Rules in your application or changes that other developers make. For example, you can work privately on a bug fix, and share your changes to the team Branch only when your work is completed.

1. In the **Workspace:** *Workspace name* pane, click Workspace menu > New Workspace, as shown in the following figure:

Creating a new Workspace

2. In the **New Workspace** pane, in the Name field, enter a meaningful name for your Workspace.

   For example:

   Enter BUG-3876.

3. Click Save.

©2025 Pegasystems Inc

Result:

The **Workspace** pane displays the newly created Workspace.

Switching between Workspaces

Work privately on another feature in your application by switching your Workspace.

NOTE: As a best practice, reuse a Workspace after sharing your changes to a Branch. Switch Workspaces only when it is necessary to keep feature development separate.

In the **Workspace:***Workspace name* pane, click Workspace menu > Switch Workspace > *Workspace name*, as shown in the following figure:

Switching Workspaces

Sharing changes with your team

After you finish performing changes in your private Workspace, share your work with your team by moving your changes to the team's Branch. Additionally, you can secure the Branch with a password to ensure that no conflicts or unexpected Rule overrides occur.

1. In the **Workspace** pane, click Share changes.

©2025 Pegasystems Inc

2. In the **Share changes** dialog box, select the Branch:

Choices	Actions
Create a new Branch	a. In the Share to Branch list, select Create new. b. Click Create a new Branch. c. In the Branch name field, enter a descriptive name for your Branch. **For example:** Enter `UIBugFixes`. Branch names can only contain letters, numbers, and dashes. d. **Optional:** To provide more information about the Branch, in the Description field, enter a brief description. e. Click Submit.
Select an existing Branch	a. In the Share to Branch list, select a Branch to store your changes. b. If the selected Branch is locked, in the Provide a lock password field, enter the password to unlock the Branch. After sharing changes the Branch is again automatically locked.

3. **Optional:** To avoid Rule conflicts and overrides, set a password for the Branch:
   a. Select the Lock Branch after sharing changes checkbox.
   b. In the Provide a lock password and Confirm lock password fields, enter identical passwords.

©2025 Pegasystems Inc

4. Click **Submit**.

For example:

The following figure shows the **Share changes** dialog box with information provided to share changes from a Workspace to a Branch, with locking of the Branch:

Publishing changes to a Branch

Switching Workspaces off

If your scenario does not require you to make changes privately, you can switch the Workspaces off. As a result, App Studio automatically saves your changes in the most appropriate Ruleset.

In the **Workspace:** *Workspace name* pane, click **Workspace menu** > **Switch off Workspace**, as shown in the following figure:

©2025 Pegasystems Inc

Switching Workspace off

You can switch Workspaces back on at any time. For more information, see Enabling Workspaces.

Delivering application documentation

Deliver application documentation to stakeholders for transparent application development, and to end users to ensure that they have enough information to independently use your software.

Documents for stakeholders

Create application documentation for stakeholders to deliver details about your application, such as an application overview, the case types, Data Model, and personas. As a result, the stakeholders are well informed about how your application represents and handles relevant business processes. For greater flexibility, you can decide which elements to include in the documentation.

To adjust the documents further to your individual needs, you can use Dev Studio to apply individual styling and structure to the output. For example, you can bold the information that you want to emphasize.

©2025 Pegasystems Inc

To meet your specific needs, you can also export a Data Model in your application to a file, for example, to share with stakeholders offline.

Creating an application document in App Studio

Application guides for developers and administrators

Give developers and administrators guidance to work with your application independently by creating meaningful application guides. For greater clarity, organize the content of your guides into logical chapters, and then populate the chapters with relevant tasks. For example, you can prepare a guide to building logic in your application. Then, you can create chapters on different logic types, such as decision trees and decision tables. In a chapter about configuring decision tables, you can add a task about prepopulating a decision table with values that are applicable to your

©2025 Pegasystems Inc

organization. You can provide more information by attaching URLs, linking rules or data instances, or by providing links to external documentation.

The following figure shows a sample application guide on creating application logic, with one completed task:

©2025 Pegasystems Inc

Application guide

©2025 Pegasystems Inc

Guided tours for your users

Showcase application features to your users by creating a guided tour. A series of pop-up windows that describe features in real-time help users work with your application independently and reach their goals quicker. For example, you can create a guided tour that describes features in an end-user portal, such as Case Manager.

> **NOTE:** Only Pega Platform™ applications based on UI Kit and Theme Cosmos support guided tours. Applications based on Constellation do not support this feature. For more information about UI architecture, see User experience.

Start documenting your application by performing the following actions:

- **Documenting your application**
- **Creating a guide for application developers and administrators**
- **Describing features to end users**

Documenting your application

Create documentation to engage stakeholders and give guidance to the end users and developers who interact with your application. You can generate a physical document or provide embedded information, based on your audience.

Use the following techniques to document your application:

- **Creating project documents for stakeholders**
- **Exporting a Data Model into a document**
- **Contents of application overview and Data Model documents**
- **Creating project documents for stakeholders in Dev Studio**
- Creating a report

PEGA ©2025 Pegasystems Inc

- Creating advanced reports

Creating project documents for stakeholders

Create an application overview to educate stakeholders about the features in your application and to review the development status of your project. To provide most relevant information, customize the application overview by including only specific chapters.

1. In the navigation pane of App Studio, click **Overview**.
2. In the **Application documents** section, click **Export**.
3. **Optional:** To modify the outline, in the **Generate overview document** window, click **Show outline**, and then:
   - To include a chapter in the outline, select the checkbox next to the name of the chapter.
   - To change the order of the items in the outline, move the chapters.
4. Click **Generate**, and then perform one of the following actions:
   - To save the document locally, click **Download**.
   - To change the content of the document, click **Regenerate document**, and then perform steps 3 and 4.

- Adopting feature-driven development

Exporting a Data Model into a document

Share information about a Data Model in your application with stakeholders and developers so that they can quickly check if the Data Model includes all relevant information.

For example, if the aim of your application is to investigate various loan requests, you can quickly check the information that is collected for every Case Type, such as *Mortgage request* or *Car loan request*.

1. In the navigation pane of App Studio, click **Overview**.

©2025 Pegasystems Inc

2. Click **Actions** > **Export Data Model**.
3. In the **Export Data Model** dialog box, select a format for the file.
4. Click **Generate**.

- Adopting feature-driven development

Contents of application overview and Data Model documents

Provide stakeholders with high level information about your application by generating application overview and Data Model documentation. Customize the document to include only relevant information.

You can include the following details in the application overview:

- Case Types
  - Case Type details
    - Graphic representation of Stages in Case Types
    - Case wide optional Processes
    - Case wide optional actions
  - Stages in Case Type details
    - Statuses associated with each Stage
    - Processes associated with each Stage
    - Stage wide optional Processes
    - Stage wide optional actions
    - Conditions for skipping a Stage
- Data types
  - Systems of records
  - References between data types and Case Types
- Roles
  - Number of operators for each role
  - Web Channels available for each role
  - Pages available for each role for each web Channel

©2025 Pegasystems Inc

- Channels and interfaces
  - Roles that use a specific Channel as a default Channel
  - Roles with access to a specific Channel
- Features that are defined in the application

A Data Model document contains the following information that you can export to `.xml` and `.pdf` files:

- All data types and Case Types with descriptions
- References between data type and Case Types
- System of records for each data type
- Fields for each data type and Case Type with the following details:
  - Label
  - ID
  - Type
  - Field group type, if applicable
  - List items, if applicable
  - **Is calculated** field
  - Calculation details, if calculated
  - Description

Related tasks

- Exporting a Data Model into a document

Creating project documents for stakeholders in Dev Studio

Create project documents to educate stakeholders about the features in your application and to review the development status of your project.

©2025 Pegasystems Inc

Before you begin:

Ensure that your application uses features instead of specifications. For more information about the legacy documents that support specifications, see previous versions of the help on Community.

1. **Optional:** Align the look and feel of your document with your brand.
   - Changing the style of a project document
   - Changing the structure of a project document
2. In the header of Dev Studio, click **Configure** > **Application** > **Tools** > **Document**.
3. In the **Select a document type** section, choose a document type:
   - To generate a document that provides high-level information, such as the application description, Case Types, Data Model, and roles, click **Application overview**.
   - To generate a document that provides detailed information on application data, click **Data Model**.
   - To generate a document that provides traceability from unfinished features to open work items, and a brief product overview, click **Gap analysis**.
4. To control which features you document, select a filter option from the **Features to include** list.
5. To exclude a chapter from the document, clear the checkbox next to the chapter name.
6. Click **Generate document**.
7. Click **Document generation completed**.
8. Click the document name to open the document in Microsoft Word.

- **Changing the style of a project document**
- **Changing the structure of a project document**

©2025 Pegasystems Inc

- Creating a guide for application developers and administrators
- Creating guidance for developers

Changing the style of a project document

Use HTML styles to align the look and feel of project documents with your brand.

> **Before you begin:**
>
> Review your document template for custom styles, because these styles override the HTML styles that you provide. For more information, see Changing the structure of a project document.

1. Use search to find and open the standard *@baseclass.pyRTADocStyle* HTML Rule.
2. Click **Save as** > **Specialize by class or Ruleset**.
3. In the **Add to Ruleset** field, select a Ruleset and version in your application.
4. Click **Create and open**.
5. On the **HTML** tab, enter your custom styles in the `<style>` element.
   For example, you can change the font, font size, or colors.
6. Click **Save**.

Changing the structure of a project document

Use a template to change align the structure of project documents with your brand.

1. Use search to find and open the standard `pyNextGenDocTemplate` file.
2. Click **Download file**.
3. In Microsoft Word, change the structure of the template.

   For example, you can insert a page or add a header for your company logo.

4. Save the `pyNextGenDocTemplate.docx` file to your local machine.
5. On the File form, click **Upload file**.
6. Click **Choose File**, and then navigate to your local file.

©2025 Pegasystems Inc

7. Click **Open**.
8. Click **Upload file**.
9. Click **Save**.

Creating a guide for application developers and administrators

Create an application guide to direct application developers or administrators through a set of tasks. By providing instructional information, you can help users learn about the functionality that your application provides, configure complex settings, or extend features with limited assistance.

You add structure to an application guide by organizing related tasks into chapters. Each task can include instructional text and a link to supporting information, such as an external document or a landing page.

For example, you can create an application guide for migrating locally stored information to a database server. In the chapter for configuring your database, you can add tasks for creating tables, views, and users. In the task for creating users, you can supplement the instructions with a document that lists database privileges.

1. Create a Rule in the **Application Definition** > **Application Guide** Rule category. For more information, see Creating Rules.
2. Add chapters, tasks, and supporting information to the application guide, and then add the guide to a Portal.

- Creating project documents for stakeholders in Dev Studio
- Creating guidance for developers

Adding a chapter to an application guide

To organize related tasks, add a chapter to an application guide.

©2025 Pegasystems Inc

> **TIP:** To support translation, use a paragraph to define your chapter description. For more information, see Creating paragraph Rules.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, click **Add Chapter**.
3. In the **Chapter name** field, enter text that describes the purpose of this chapter.
4. Provide a chapter description.
   - To reference a paragraph:
     - In the **Description** list, select **Paragraph**.
     - In the **Paragraph** field, press the Down Arrow key, and then select the paragraph that describes the chapter.
   - To manually enter a description:
     - In the **Description** list, select **Rich Text**.
     - In the rich text editor, enter and format your chapter description.
5. Click **Submit**.
6. Click **Save**.

- Documenting your application

Adding a task to a chapter in an application guide

To provide step-by-step information in an application guide, add a task to a chapter.

> **TIP:** To support translation, use a paragraph to define your task description. For more information, see Creating paragraph Rules.

©2025 Pegasystems Inc

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click **Add task**.
4. In the **Task name** field, enter text that describes the outcome of the task.
5. Provide instructions to complete the task in the task description.
   - To reference a paragraph:
     - In the **Description** list, select **Paragraph**.
     - In the **Paragraph** field, press the Down Arrow key, and then select the paragraph that describes the task.
   - To manually enter a description:
     - In the **Description** list, select **Rich Text**.
     - In the rich text editor, enter and format your task description.
6. Click **Submit**.
7. Click **Save**.

- Creating project documents for stakeholders in Dev Studio

Adding supporting information to an application guide

Add supporting information to an application guide to direct users to the resources that relate to their current task.

Use the following techniques to add supporting information to an application guide:

- Documenting your application

©2025 Pegasystems Inc

Linking a URL to an application guide

Link a URL to your application guide to integrate a task with an external resource.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Open URL**.
5. In the **Display text** field, enter a label for the link that users click in the task.
6. In the **URL** field, enter the web address of your supporting information.
7. Click **Submit**.
8. Click **Save**.

- Documenting your application

Linking a Rule or data instance to an application guide

Link a Rule or data instance to your application guide give users direct access to internal resources.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Open Rule/data**.
5. In the **Display text** field, enter a label for the link that users click in the task.
6. In the **Rule Type** field, enter the class that defines the type of Rule that you are linking.

   For example, all Data Transforms are instances of the *Rule-Obj-Model* class.

7. In the **Instance Name** field, use one of the following methods to select your Rule.
   - Press the Down arrow key, and then select the fully qualified name of a Rule.

©2025 Pegasystems Inc

- Enter the name of a data instance.

8. Click **Submit**.
9. Click **Save**.

- Creating a guide for application developers and administrators

Calling a guided tour from an application guide

Call a guided tour from your application guide to make a task more interactive.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Launch tour**.
5. In the **Display text** field, enter a label for the link that users click in the task.
6. In the **Action to take** field, select **Start tour**.
7. In the **Tour applies to** field, press the Down Arrow key, and then select the class to which the guided tour applies.
8. In the **Tour name** field, press the Down Arrow key, and then select the name of a guided tour.
9. **Optional:** Change the look and feel of the tour stops in the guided tour.
   a. Select the **Override default tour stop container template** checkbox.
   b. In the **Template name** field, press the Down Arrow key and then select the name of a section that defines the layout and controls for each tour stop.
10. Click **Submit**.
11. Click **Save**.

- Creating a guide for application developers and administrators

PEGA ©2025 Pegasystems Inc

Calling a wizard from an application guide

Call a wizard from your application guide to provide well-defined steps for a complex task.

> **Before you begin:**
>
> Ensure that your wizard, or Screen Flow, inherits from the *PegaAccel-* class.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Run wizard**.
5. In the **Display text** field, enter a label for the link that users click in the task.
6. In the **Wizard** field, press the Down arrow key, and then select the class of your wizard.
7. Click **Submit**.
8. Click **Save**.

- Creating a guide for application developers and administrators

Linking a help topic to an application guide

Link a standard help topic to your application guide to integrate a task with supporting information.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Open help URL**.
5. In the **Display text** field, enter a label for the link that users click in the task.

©2025 Pegasystems Inc

6. In the **Relative path** field, enter the file path to the help topic, relative to the base path that you provide on the System Settings landing page.
7. Click **Submit**.
8. Click **Save**.

- Creating a guide for application developers and administrators

Linking a landing page to an application guide

Link a landing page to your application guide to give users direct access to internal tools.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. Add a link to the task.

> **TIP:** For an example configuration of the following fields, open the *Pega-Landing.pyMainMenu* navigation Rule by searching for it or by using the Application Explorer.

   a. In the **Action type** list, select **Open landing page**.
   b. In the **Display text** field, enter a label for the link that users click in the task.
   c. In the **Action** field, select **Display**.
   d. In the **Name** field, enter a label for the tab that displays the landing page.
   e. In the **Class** field, press the Down Arrow key, and then select a class that defines harnesses.
   f. In the **Harness Name** field, select the harness that the landing page displays.
   g. Decide which tab has focus when users open the landing page.
   h. Enter the tab name in one of the following fields, based on the tab's position in the landing page hierarchy:

©2025 Pegasystems Inc

- Level A
- Level B
- Level C

5. Click **Submit**.
6. Click **Save**.

- Creating a guide for application developers and administrators
- Adding action sets to a control
- Available UI actions

Linking a custom section to an application guide

In addition to other types of supporting information, such as URLs and help topics, you can link a section to your application guide to change the presentation or available actions in a task.

1. Open an application guide by searching for its name or by navigating to the **Application Definition** > **Application Guide** Rule category in Dev Studio.
2. On the **Definition** tab, expand a chapter in the **Chapter name** column.
3. Click the **Gear** icon next to a task name.
4. In the **Action type** list, select **Custom section**.
5. In the **Action value** field, press the Down Arrow key, and then select a section in the *Embed-Guide-Task* class path that is not in a *Pega-* Ruleset.

> **TIP:** To support users in App Studio, ensure that the first action in your section calls the *pega.desktop.closeAllDocuments* API.

6. Click **Submit**.
7. Click **Save**.

©2025 Pegasystems Inc

- Creating a guide for application developers and administrators
- Adding action sets to a control
- Adding action sets to a control
- Available UI actions

Adding an application guide to a Portal

Add your application guide to one or more Portals so that end users can access it.

> **Before you begin:**
>
> Add the *pxLaunchAppGuide* control to your Portal.

1. In the Dev Studio header, click the name of your current application, and then click **Definition** to open the Application form.
2. Click the **Documentation** tab.
3. If there are no guides defined in the Application Guides section, click **+ Add guide**.
4. In the field that is displayed, press the Down arrow key, and then select the name of your application guide.
5. In the **Available in** column, click **Portals**.
6. Click **+ Add Portal**.
7. In the field that is displayed, press the Down arrow key, and then select the name of a Portal.
8. Click **Submit**.
9. On the Application form, click **Save**.

- Creating a guide for application developers and administrators

Describing features to end users

You can use a guided tour to showcase application features in real time. By providing information in a sequence of pop-up windows, you can help users discover functionality that is available on their current screen.

©2025 Pegasystems Inc

> **NOTE:** Only Pega Platform™ applications that you build on the traditional UI architecture support guided tours. Applications that you build on Constellation do not support this feature.

For example, you can create a tour that gives example use cases for the different tools in a web Channel.

1. **Creating content for a guided tour**
2. **Associating content with a feature in a guided tour**
3. **Integrating a guided tour with your application**

- Documenting your application

Creating content for a guided tour

You can use sections to create the content for each pop-up window, or tour stop, in a guided tour. By using a common format, you can associate your content with application features more quickly.

> **NOTE:** Only Pega Platform™ applications that you build on the traditional UI architecture support guided tours. Applications that you build on Constellation do not support this feature.

1. Create a section that contains the text for the tour stop.

   You can add action controls to the section, however, all elements must be read-only. For more information, see Creating Sections.

2. Find a section in a form or Portal, that contains the application feature for your tour stop.
   - Search for the section by name.

©2025 Pegasystems Inc

- In the Application Explorer, navigate to the section by class.
- Use Live UI to inspect sections on the screen.

3. On the Section form, click an element, such as a field or layout.
4. If you are in full section editing mode, click the **Gear** icon.
5. In the **Cell properties** dialog box, enter a unique string identifier in the **Tour ID** field.
6. Click **Submit**.
7. Click **Save**.

Next topic:
Associating content with a feature in a guided tour

- Live UI
- Documenting your application

Associating content with a feature in a guided tour

You can associate content with a feature to create a tour stop in a guided tour.

> **NOTE:** Only Pega Platform™ applications that you build on the traditional UI architecture support guided tours. Applications that you build on Constellation do not support this feature.

1. Find and open a guided tour, by searching for it or by using the Application Explorer.
2. If there are no guided tours available, create one in the **User Interface** > **Guided Tour** Rule category.

   For more information, see Creating Rules.

3. On the **Definition** tab, click **+ Add tour stop**.
4. In the **Tour stop anchor** list, select a type of identifier, and then enter an ID in the text field that identifies an application feature.

©2025 Pegasystems Inc

- **Tour ID** - A unique string that you set on the **General** tab of the **Cell Properties** panel on the Section form.
- **Custom css selector** - A string that you define, which contains a pattern for finding elements on the screen.

5. In the **Title** field, enter the text that is displayed above the content for your tour stop.
6. In the **Tour stop section** field, press the Down Arrow key, and then select the section that defines your content.
7. In the **Show when** list, select an option to control when the tour stop is displayed at run time.
8. If your tour stop relies on the presence of a specific element, select a type of identifier, and then enter an ID for that element in the fields that are displayed.
9. **Optional:** To change the order of your tour stop, drag it to a different position on the form.
10. Click **Save**.

Previous topic:
Creating content for a guided tour
Next topic:
Integrating a guided tour with your application

- Creating project documents for stakeholders

Integrating a guided tour with your application

You can use one or more controls to integrate a guided tour with your application. By defining the conditions that start, stop, or restart your tour, you can respond to the actions that users take on the screen.

©2025 Pegasystems Inc

NOTE: Only Pega Platform™ applications that you build on the traditional UI architecture support guided tours. Applications that you build on Constellation do not support this feature.

Before you begin:

Ensure that the users in your target audience have write permission for the *Data-Preference-Operator* class, because guided tours do not start unless their history can be recorded.

1. Add a control to the section that starts the tour.
   a. Find and open a section to start the guided tour, by searching for it or by using the Application Explorer.
   b. On the **Design** tab of the Section form, add any field, or control, to the section. For more information, see Creating Sections.
   c. Click the field to edit it.
   d. If you are in full section editing mode, click the **Gear** icon.
   e. In the **Cell properties** dialog box, click **Change**, and then click **Other** in the **Custom** control category.
   f. In the autocomplete field, enter one of the following controls:
      - *pxGuidedTourAutoStart* – Starts the tour one time per user.
      - *pxGuidedTourAutoStartOncePerSession* – Starts the tour one time per session, to support users who share the same log-in credentials.
2. Associate the control with your guided tour.
   a. In the **Cell properties** dialog box, click **Parameters**.
   b. In the **TourClass** field, press the Down Arrow key, and then select the class of your guided tour.
   c. In the **TourName** field, press the Down Arrow key, and then select the name of your guided tour.

©2025 Pegasystems Inc

   d. **Optional:** To start the tour every time that the control is displayed, select the AutoLaunchAlways checkbox. This option is available for the *pxGuidedTourAutoStart* control only.
3. Click Submit.
4. Click Save.
5. **Optional:** To support restarting the guided tour, add actions to other elements on the screen.
   a. Follow step 1 to open the Cell Properties dialog box for another section or another element in the same section.
   b. Click Actions, and then click Create an action set.
   c. Click Add an event, and then click the type of event, such as mouse click, that initiates the action.
   d. Click Add an action > All actions > Manage guided tour.
   e. In the Action to take list, select Start tour.
   f. In the Tour applies to field, enter the class of your guided tour.
   g. In the Tour name field, enter the name of the guided tour.
   h. **Optional:** To restart the tour from the beginning instead of the last visited tour stop, clear the Always continue where left off checkbox.
   i. Click Submit.
   j. Click Save.

Previous topic:

Associating content with a feature in a guided tour

- Adding action sets to a control

Troubleshooting tools and techniques

Maintain the good quality and health of your application by using the troubleshooting tools and techniques that Pega Platform™ provides. You can view application metrics to locate any issues and then fix them, so that you deliver top quality software.

©2025 Pegasystems Inc

Maintaining the good health of your application

Establish and maintain the good health of your application by implementing the following techniques:

Compliance score

Pega Platform calculates the compliance score by comparing warnings weighted by severity and justification to the total number of Rules in your application. You can quickly access and review Rules that generate warnings to improve your compliance score. Keep the compliance score at 90 or above to ensure that your application is in good health by following application development best practices, such as avoiding custom code and completing the security checklist. The following figure shows the compliance score metrics:

Compliance score

For more information, see Improving your compliance score.

Unit testing

Unit testing individual Rules helps you test and then fix rules that you develop before moving your application to the production environment. Thanks to unit testing, you can spot erroneous Rules even before encountering an unexpected application behavior. For faster application development, you can create reusable test cases to test multiple rules.

For more information, see Unit testing individual Rules.

©2025 Pegasystems Inc

Finding and fixing issues

Use the following tools that Pega Platform provides to determine and fix issues in your application, and as a result, improve its quality:

The Tracer tool

The Tracer tool traces events in your application, such as activities, Data Transforms, Rules, and Processes. By running events and tracing them in real time, you can conveniently spot which parts of your application hinder processing and need fixing. The following figure shows sample Tracer results:

The Tracer tool

For more information, see Application debugging by using the Tracer tool.

The Pega Clipboard tool

The Clipboard tool stores snapshots of Data Pages that your application processes. As a result, you can quickly check the properties and their values that are engaged in the application processing to determine which property value causes errors. You can also modify property values on the clipboard to test data before entering it at run time. The following figure shows the clipboard with a node of Data Pages on the left and property values on the right:

©2025 Pegasystems Inc

The Clipboard tool

For more information, see Clipboard tool.

My Alerts display

By viewing alerts generated in your system, you can quickly determine the cause of errors in your application. The alert log displays summary information for each alert, such as the date and time, the alert type, the work pool that the requestor uses, and the measured value of the Key Performance Indicator, recorded in seconds as a count, or in bytes.

The following figure shows a sample alert log:

©2025 Pegasystems Inc

	Date and Time	Alert Type	Value	Interaction	Work Pool	Last Input	First Activity
	11/8/23, 5:43:08 AM	Browser Interaction	2.66(s)	32	OS96BZ-Mortgage-Work	Activity=Data-Portal.ShowSelectedPortal	Rule-Obj-Activity:pzRunActionWrapper
	11/8/23, 5:43:08 AM	DB Time	1.99(s)	0	OS96BZ-Mortgage-Work	NA	Rule-Obj-Activity:pzWarningsIntroduced
	11/8/23, 5:43:08 AM	DB Time	1.59(s)	0	OS96BZ-Mortgage-Work	NA	Rule-Obj-Activity:pzWarningsIntroduced
	11/8/23, 5:43:07 AM	DB Time	0.69(s)	32	OS96BZ-Mortgage-Work	Activity=Data-Portal.ShowSelectedPortal	Rule-Obj-Activity:pzRunActionWrapper
	11/8/23, 5:43:05 AM	DB Time	0.54(s)	30	OS96BZ-Mortgage-Work	Activity=pzRunActionWrapper	Rule-Obj-Activity:pzRunActionWrapper
	11/8/23, 5:43:04 AM	Browser Interaction	10.10(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:43:04 AM	DB Time	1.46(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:43:02 AM	DB Time	1.53(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:43:01 AM	DB Time	1.70(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:59 AM	DB Time	1.53(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:57 AM	DB Time	1.53(s)	23	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:53 AM	Browser Interaction	10.03(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:53 AM	DB Time	1.49(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:52 AM	DB Time	1.75(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:50 AM	DB Time	1.50(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:48 AM	DB Time	1.48(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:47 AM	DB Time	1.50(s)	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:45 AM	Lightweight List copy	31	21	OS96BZ-Mortgage-Work	Activity=@baseclass.doUIAction	Rule-Obj-Activity:doUIAction
	11/8/23, 5:42:43 AM	DB Time	0.54(s)	18	OS96BZ-Mortgage-Work	Activity=pzRunActionWrapper	Rule-Obj-Activity:pzRunActionWrapper
	11/8/23, 5:42:42 AM	Browser Interaction	2.40(s)	17	OS96BZ-Mortgage-Work	default portal: Data-Portal.ShowDesktop	Rule-Obj-Activity:ShowDesktop

1 2 3 next>

This Session Only | All My Sessions | Security Alerts

My Alerts display

For more information, see My Alerts display.

Explore the following articles to learn more about available troubleshooting tools and techniques:

- **Improving your compliance score**
- **Unit testing individual Rules**
- **Troubleshooting newly created Rules**
- **Application debugging by using the Tracer tool**
- **Clipboard tool**
- **Analyzing application performance with PAL**
- **My Alerts display**
- **Viewing and resolving errors**
- **Viewing in-progress and completed wizards**

PEGA ©2025 Pegasystems Inc

- **Creating connector simulators**
- **The DateTime parse tester tool**
- **About the bulk Revalidate and Save tool**
- **Application Structure landing page**

Related concepts

- Designing applications for reuse and extension

Improving your compliance score

Follow development best practices to improve your compliance score. By eliminating risks, such as custom code or degraded performance, you can improve quality and resolve issues before your application goes into production.

To maintain your application quality, try to achieve a compliance score of 90 or greater. If your application's score becomes less than 80, take immediate action to address the issues.

You access the Compliance score landing page by clicking **Configure** > **Application** > **Quality** > **Guardrails** > **Compliance Score** in the header of Dev Studio, as the following figures show:

©2025 Pegasystems Inc

Opening Compliance score landing page

Compliance score

The Compliance score landing page includes also a score for a Security Checklist. For more information about tasks that you complete for the Security Checklist, see Security Checklist.

1. **Analyzing a compliance score**

©2025 Pegasystems Inc

2. **Resolving an application warning**
3. **Justifying an application warning**
4. **Reviewing user interface components**
5. **Exporting compliance data**
6. **Scheduling a compliance report**

- Compliance score logic
- Monitoring project compliance
- Viewing application quality metrics

Analyzing a compliance score

Analyze your compliance data to find the root cause of a low compliance score. By taking action when your application's score is less than 80, you can improve application quality more quickly.

> **Before you begin:**
>
> Encourage your team to check in their work so that you have complete data.

1. Find your compliance score.
   a. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Guardrails** > **Compliance Score**.
   b. In the **Compliance score** section, review your weighted score.
2. **Optional:** To broaden the scope of your compliance data, include built-on applications.
   a. Click **application**, and then select one or more built-on applications.
   b. Click **Apply**.
3. Identify the highest priority issues that affect your score.
   a. Click the **Compliance Details** tab.

©2025 Pegasystems Inc

   b. In the **Warning impact** section, click the values in the **Resolve now** column.
   c. Review the list of Rules with severe warnings.
   d. In the **Application risk introduced by operator** section, take note of the user with the highest risk percentage.
   e. Click the percentage, and then review the list of Rules with warnings that the user added to your application.
4. Identify the types of Rules with the most warnings.
   a. Click the **Warning Summary** tab.
   b. In the **# Rules with warnings** column, review the distribution of warnings across the different types of Rules.
   c. Expand the row for the Rule Type with the highest number of warnings.
   d. **Optional:** For more information about a specific Rule, click a Rule name in the **Rule** column.
5. Review how your compliance score has changed over time.
   a. Click the **Compliance Score** tab.
   b. In the **Rules modified on or after** field, enter a date, and then click **Apply date filter**.

> **What to do next:**
>
> Use the findings from your analysis to focus your improvements on specific Rules and warnings, and coach individuals on development best practices.

Next topic:
Resolving an application warning

- Monitoring project compliance
- Compliance score logic

Resolving an application warning

Resolve warnings to improve application quality and ensure that you follow development best practices.

©2025 Pegasystems Inc

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Guardrails** > **Compliance Score**.
2. In the **Compliance score** section, click the number in the **Rules with unjustified warnings** field.
3. In the **Rule name** column, click a Rule to open it.
4. In the header of the Rule form, click **Review/Edit**.
5. Review the warning message to understand which best practice your application does not follow, how it affects your compliance score, and what to do next.
6. Click **OK**.
7. Update the Rule by following the guidance from the warning message.
8. Click **Save**.

Previous topic:
Analyzing a compliance score
Next topic:
Justifying an application warning

Justifying an application warning

Justify a warning to explain why your application is exempt from a best practice, or guardrail. By documenting which warnings are intentionally unresolved, you can improve your compliance score.

> **Before you begin:**
>
> Ensure that your Access Role is associated with the *pxAllWarningJustification* privilege.

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Guardrails** > **Compliance Score**.

©2025 Pegasystems Inc

2. In the **Compliance score** section, click the number in the **Rules with unjustified warnings** field.
3. In the **Rule name** column, click a Rule to open it.
4. In the header of the Rule form, click **Review/Edit**, and then review the warning to confirm that your application can tolerate the associated risk.
5. Click **Add Justification**.

> **NOTE:** You cannot justify some warnings, because the risk that they introduce is too severe. For these cases, consider resolving the warning instead.

6. In the field that is displayed, enter text that explains why your development team chooses to leave this warning unresolved.
7. Click **OK**.
8. Click **Save**.

> **What to do next:**
>
> Periodically review the list of Rules with unjustified warnings in your application because imported Rules can display new warnings when your development team resaves them.

Previous topic:
Resolving an application warning
Next topic:
Reviewing user interface components

- Resolving an application warning

©2025 Pegasystems Inc

Reviewing user interface components

Find out which UI elements in your application are high-maintenance. By analyzing the reports on the UI issues landing page, you can learn how to more efficiently render process-heavy components, and save development time during future updates.

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **App Studio Compliance** > **User Interface**.
2. In the **Built on applications** section, select one or more applications whose UI components you want to review.
3. Click **Apply**.
4. Review the report and follow the suggestions for updating UI components.
5. **Optional:** To open a section that contains UI issues in a new tab, expand the relevant report header, and then click the section name.

   > **For example:**
   >
   > Your application contains sections that do not use a design template. To address this issue, expand the **No design template** node, click the name of the section that you want to fix, and apply a design template to that section in the section editor.

Previous topic:
Justifying an application warning
Next topic:
Exporting compliance data

Related tasks

- Improving your compliance score

Exporting compliance data

Export compliance data to create reports that you can download and distribute later. By manually collecting data, you can control which metrics to share with stakeholders.

©2025 Pegasystems Inc

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Guardrails**.
2. To export your compliance score:
   a. Click the **Compliance Score** tab.
   b. **Optional:** To capture data from a specific point in time, select a date from the **Rules modified on or after** field, and then click **Apply date filter**.
   c. Click **Export to PDF**.
3. To export the distribution of warnings by type, age, and owner:
   a. Click the **Compliance Details** tab.
   b. **Optional:** To capture data from a specific point in time, select a date from the **Rules modified on or after** field, and then click **Apply date filter**.
   c. Click **Export to PDF**.
4. To export a list of Rules with warnings:
   a. Click the **Warning Details** tab.
   b. **Optional:** To capture Rules from a specific class, click **classes**, enter a class name in the field that is displayed, and then click **Apply**.
   c. Click **Export to Excel**.

> **What to do next:**
>
> Open and review each file that you export, because the filters that you set on the Application Guardrails landing page are not persistent.

Previous topic:
Reviewing user interface components
Next topic:
Scheduling a compliance report

Scheduling a compliance report

Schedule a compliance report to automatically inform stakeholders about changes to your application's compliance score over time.

©2025 Pegasystems Inc

Before you begin:

Ensure that the *DefaultNotify* email account is configured. For more information, see Viewing email accounts.

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Guardrails**.
2. Based on the type of data that you want to include in your report, click one of the following tabs:

   Compliance Score
   Summarizes the factors that contribute to your compliance score

   Compliance Details
   Categorizes the warnings in your application by type, age, and owner

   Warning Summary
   Depicts the distribution of warnings in your application by severity and Rule Type

   Warning Details
   Lists more information about the Rules that introduced the warnings

3. Click **Schedule report**.
4. In the **Email Scheduling** dialog box, select the **Enable email notifications** checkbox.
5. In the **Send the report** list, select the frequency at which your application sends the report data in an email.
6. Click the **Add** icon.
7. In the field that is displayed, press the Down Arrow key, and then select the name of a stakeholder.

   If the stakeholder is not in the list, enter an email address in the field instead.

8. Click **Submit**.

Previous topic:

©2025 Pegasystems Inc

Exporting compliance data

- Exporting compliance data

Monitoring project compliance

Monitor the collective compliance score for your application and built-on applications to determine how well your project follows development best practices.

1. In the header of Dev Studio, click **Configure** > **Application** > **Quality** > **Dashboard**.
2. On the **Compliance Score** tab, click **Enable historical compliance score**.
3. Periodically review the graph that is displayed, to monitor how your score changes from week to week.

> **What to do next:**
>
> If your compliance score is not displayed, review the log files for relevant error messages.

Previous topic:
Scheduling a compliance report

- Logging for VM-based deployments and containerized deployments on Kubernetes

Compliance score logic

A standard formula determines your compliance score. By understanding the logic and variables in this formula, you can focus on the development changes that improve application quality the most.

The following figure presents the logic that assigns weights to the warnings in your application.

©2025 Pegasystems Inc

$$\max\left(0{,}1 - \left(\frac{10Sev1u + 5Sev2u + 2Sev1j + Sev2j}{Total\ Remaining\ Rules}\right)\right) \times 100$$

Variable definitions:

Sev1u
: Severe, unjustified warnings

Sev2u
: Moderate, unjustified warnings

Sev1j
: Severe, justified warnings

Sev2j
: Moderate, justified warnings

The sum of the variables is divided by the number of Rules with informational warnings or no warnings. The result is a positive, integer percentage.

Compliance scores apply only to the Rules in your application and imported Rules that you resave. Property Rules and all Rules in *Pega** Rulesets do not affect your compliance score.

Previous topic:
Scheduling a compliance report

Related tasks

- Analyzing a compliance score
- Improving your compliance score

Unit testing individual Rules

An incorrect Rule configuration in an application can cause delays in Case processing. To avoid configuration errors such as incorrectly routed Assignments, unit test individual Rules as you develop them. To expedite future Rules testing, you can create reusable test Cases from the unit test.

©2025 Pegasystems Inc

You can test a Rule with test data that you provide by clicking **Actions** > **Run** on the Rule form toolbar. 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 is not available.

> **NOTE:** Some Rules can be unit-tested without the **Run** option available. For example, to perform a unit test on a Case Type, you can use the **Record test Case** action. For more information, see Understanding unit test Cases

The appearance of the **Run Rule** window varies across Rule Types, so how you run a Rule varies by its type. In general, however, the Rules run by using data from a test page that you define for the test.

Tasks involved in defining the test page include the following:

1. Selecting a method for creating the test page – You can copy values from a thread of an existing Clipboard Page to the test page, create a new test page, or reset the values of an existing test page. For more information about Clipboard Pages, see Clipboard tool.
2. Applying Data Transforms – For a reusable and expedited method of making decisions and calculating values, you can apply Data Transforms to set values for the test page. 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 entering values when you run the Rule. For more information about Data Transforms, see Data Transforms.
3. Manually entering test data – In some Cases, you can manually enter values to use. If you enter values for a test, the values that you enter override values on the test page.
4. Specifying how service Rules run – For services, you also specify whether the service Rule is to run in your session or is to run as a newly created service requestor. If the service is configured to run as an authenticated user, you are prompted for a user name and password.

PEGA ©2025 Pegasystems Inc

> **NOTE:** To test a circumstance Rule, ensure that the circumstances are correct for the Rule. Otherwise, the system tests the base Rule.

When you run the Rule, the system uses Rule resolution. If you unit test a Rule, and there is a higher version of the Rule, the system runs the higher version.

After you run the test, you can also convert the test to a reusable test Case that you can run at any time. For more information about using unit test Cases, see Understanding unit test Cases.

- **Unit testing service Rules**

Troubleshooting newly created Rules

If there are issues with copying a Rule or data instance, see the following information for troubleshooting help.

- If your newly created Rule is not being executed, it might be related to the Rule resolution algorithm which operates differently on different Rule Types. See the help topics *Completing the Create form* for each Rule Type to understand specifics of Rule resolution for that type.
- Many standard Rules have Work- as the **Apply to** key part, but you cannot create additional Rules with Work- as the **Apply to** key part. When you're copying such a Rule, the Save As form defaults the name of the container class for the current work pool as the **Apply to** class.
- The Save As form is restricted to those users who hold the privilege *@baseclass.ToolbarFull* or *@baseclass.ToolbarSaveAs*.
- The availability of your new Rule is not specified on the Save As form.
- The status of your new Rule may be automatically cleared if the following criteria are met:
    - The original Rule status is not internal
    - One of the following conditions apply
        - The specified Ruleset is not a Branch of the original Rule

©2025 Pegasystems Inc

    - You are changed the Apply To class
    - The original Rule has circumstances defined
- You can specialize a Rule or create a circumstance of it.

- Copying a Rule or data instance
- Setting Rule status and availability

Application debugging by using the Tracer tool

To provide a top-quality experience for your software users, test and debug your applications by using the Tracer tool. You can test and debug multiple resources, for example, activities, Data Transforms, Decision Rules, service Rules, parse Rules, and processes. You can also troubleshoot offline-enabled applications from a browser by testing scripts when the application is running.For example, you can trace the events that are related to queue processing to ensure that your application correctly queues tasks to process in the background. As a result, you increase the efficiency of your application and save time because you avoid further debugging that untraced events might cause.

You can select which Rulesets, Rules and events to trace; set breakpoints and watch variables; trace reference properties; select which requestor session to trace, and add custom events to Tracer output.

You can also trace services and listeners anywhere in the cluster by using the RuleWatch trace.

When you set up your trace conditions and start tracing, you can view the Tracer output in the Tracer window or save the Tracer output to your local system, so that you can analyze your system offline or share the results with other developers.

> **NOTE:** When using this tool for Customer Service chat interactions, be aware that multiple requestors can serve a single browser session.

©2025 Pegasystems Inc

> **NOTE:** The Tracer tool is not available during a test coverage session.

Tracer for Constellation applications

Apart from the standard Pega Platform applications and your custom applications, Tracer also supports applications that you build by using the Constellation. Because Constellation applications are stateless and rely on DX APIs, the Tracer results window reflects the scheme in which your application interacts with a browser during a browser session. For greater clarity, the results window groups events by request ID, because multiple requests can be active simultaneously, as in the following example:

Tracer results for Constellation

You can also use the Tracer tool to debug applications directly from a user Portal, for example, for applications that are already in production, as the following figure shows:

©2025 Pegasystems Inc

Opening Tracer from a web portal

> **NOTE:** To open the Tracer tool from a user Portal, ensure that you have the *OpenDeveloperForm* and *clipboardViewer* privileges.

For relevant training materials, see the Debugging application errors module on Pega Academy.

- **Tracer best practices**
- **Tracer disk usage FAQ**
- **Configuring trace conditions**
- **Tracing and capturing events**
- **Tracing services**
- **Viewing Tracer results in the Tracer window**
- **Offline debugging by using Pega-TracerViewer**
- Constellation in traditional UI
- Cluster-wide tracing of service Rules
- Estimating test coverage

©2025 Pegasystems Inc

Tracer best practices

Use Tracer best practices to make tracing easier and more efficient.

Trace only one requestor session per operator and node

Do not attempt to start the Tracer when another requestor session on the same Pega Platform node is using the same Operator ID, or is running the Tracer.

Turn off HTTP compression

Ordinarily, Pega Platform uses HTTP compression to reduce the length of text messages sent to Internet Explorer. HTTP compression is also used in SOAP messages. This might complicate Tracer debugging because compression makes it difficult to interpret the messages.

To disable HTTP compression, set the value of the *prconfig/enable-compression/default* Tracer dynamic system setting to `false` and its owning Ruleset to *Pega-Engine*. This change takes effect the next time your system starts. For more information, see Creating a dynamic system setting.

Reset this value to enable compression when debugging is complete.

Domain Name System lookups required

In some Pega Platform installations, the `ClientHostName` value appears only as a numeric Internet Protocol (IP) address. The true `ClientHostName` is visible only when the application server is enabled to perform Domain Name System lookups:

- For Tomcat-based systems, set the parameter `enableLookups` (in the `server.xml` file) to "true" to allow DNS lookups.
- For Oracle Weblogic, add the parameter `ReverseDNSAllowed="true"` to the `config.xml` file to allow DNS lookups.

©2025 Pegasystems Inc

Duplicate trace error

If the node hosting your trace session fails and you attempt to trace the same service again, you might encounter a duplicate trace error. Clear stale trace sessions using Admin Studio.

- Tracer event queue overflows
- Adjusting the buffer size of the Tracer header
- Reducing Tracer event output
- Configuring dynamic system settings

Tracer disk usage FAQ

For effective debugging, Tracer creates a file with the details of traced events so that you can make an informed decision about processes in your application. When you trace a high number of events, the file size can become extensive. To avoid issues with disk usage, Tracer sets a limit on the file size. As a result, tracing your application has no impact on running of your system.

To which deployments of Pega Platform does the limit apply?

The limit on disk usage applies to on-premises Pega Platform deployments, Pega Cloud deployments, and Pega Platform deployments on client-managed clouds.

What is the disk usage limit?

The disk usage limit is 10 GB per node. The limit is the same for all nodes in a system, and the disk limit check occurs at the node level. Multiple sessions on one node share the disk usage limit.

In a sample scenario, four Tracer sessions run on two nodes. Sessions S1 and S2 run on node N1, and sessions S3 and S4 run on node N2. The disk usage limit defines Tracer behavior in the following way:

PEGA ©2025 Pegasystems Inc

- When the session S1 reaches the limit of 10 GB, S1 stops and the system deletes the respective file of event details.
- Sessions S2, S3, and S4 continue to run.

What happens when the file size reaches the disk usage limit?

When the event file reaches the size of 10 GB, Tracer behaves in the following way:

- Tracing of the session that reaches the disk usage limit stops.
- The system deletes the event file. You can then only download error event messages.
- In the Tracer window, you can view the header of the previous Tracer session without the full event details.

Does the system reclaim the disk space after stopping a Tracer session?

Yes, the system reclaims the disk space after the file size in a session reaches 10 GB.

Does the Tracer stop running after reaching the disk usage limit?

Yes, the Tracer stops a tracing session that reaches the disk usage limit. For service Rule tracing, the system generates a disk write event. Consider a scenario in which sessions S1 and S2 run on the same node. The node already uses 9 GB of disk space. If session S1 tries to use another 1.5 GB, the system stops session S1. Session S2 continues to run.

How does service Rule tracing work with the disk usage limit?

Service Rule tracing continues to run even after the event files reach the size of 10 GB, however, the system deletes the file that exceeded the limit and you can view only event error messages.

PEGA ©2025 Pegasystems Inc

Can I increase or decrease the disk usage limit for Tracer?

Yes, you can change the disk usage limit by creating and then editing the *tracer/disk/maxusage* dynamic system setting. The minimum value for the disk usage limit is 5 GB.

> **NOTE:** Change the default limit value only in justified situations, for example, when you do not have 10 GB free space in your system, or when 10 GB per node is not sufficient for your scenario.

For more information about dynamic system settings, see Configuring dynamic system settings.

Does the system delete the event file when the file size reaches 10 GB?

Yes, the system deletes the event file after the file size reaches 10 GB. The Tracer file that you can later download contains only error messages.

Does each Tracer session have a separate event file?

Yes, each Tracer session has a separate event file. If you have multiple Tracer sessions that run on one node, the total maximum size of all event files is 10 GB.

How does the system maintain the disk usage limit?

The system maintains the disk usage limit at the node level. Multiple Tracer sessions share the usage when running on one node.

In a sample scenario, sessions S1 and S2 run on node N1, while session S3 runs on node N2. Session S1 occupies 9 GB in an event file, and session S2 uses 700 MB. Session S2 needs 200 MB to write details about the next event to trace. The system allows the action because the total space that sessions S1 and S2 need in the files is less than 10 GB. Next, session S2 tries to write details about the next event that require another 200 MB. The system stops session S2 and sends the message that the file size

PEGA ©2025 Pegasystems Inc

reaches the Tracer disk usage limit for node N1. The system deletes the event file for session S2 and users can then only view the error messages. In the meantime, session S3 tries to write event details that require one GB in the event file. The system allows the action because every node has a separate limit of 10 GB.

What happens if a Tracer session starts and a browser crashes, or the Tracer does not receive a close event before the event file size reaches 10 GB?

The Tracer session remains active and writes events to the file. However, the system treats the session as stale and deletes the session automatically after an hour. You can change the timeout value after which the system deletes stale sessions by editing the *tracer/session/timeout* dynamic system setting.

> **NOTE:** The *tracer/session/timeout* dynamic system settings affects also pause sessions. The system deletes paused sessions after the timeout value passes.

For more information about editing dynamic system settings, see Configuring dynamic system settings.

What can I do when the file size constantly reaches the 10 GB limit, even if I do not run any Tracer sessions that use disk space?

Verify with your system administrator that you close all Tracer sessions that are not necessary at the moment. The list of operators that run Tracer sessions is available in the error message after reaching the disk usage limit. Also, ensure that you select only necessary Rules and events to trace instead of tracing all available events. If you do not need to view Clipboard Page content, selecting the **Abbreviate Events** checkbox in the Tracer settings drastically reduces the disk usage.

PEGA ©2025 Pegasystems Inc

How can I reclaim disk space when another operator starts a Tracer session and forgets to close the session?

You can check the other operators who run Tracer sessions. The list of operators is available in the error message that you receive after the event file size reaches the disk usage limit. You can also close Tracer sessions from Admin Studio.

Related concepts

- Managing requestor pools
- Monitoring system health
- Offline debugging by using Pega-TracerViewer

Related tasks

- Viewing Tracer results in the Tracer window
- Configuring trace conditions
- Tracing services

- Managing system resources

Configuring trace conditions

Before you run the Tracer tool, optionally configure trace conditions, based on the activity, Data Transform, Rule, and so on, that you are testing and debugging. If you do not configure trace conditions, the default conditions are used.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. **Optional:** Configure Tracer settings. Select which events, event types, Rulesets, and pages to trace, break conditions, and options for the amount of output that you want the Tracer to generate.
3. **Optional:** Send the Tracer output to a file. This option is useful if you want to work offline with the Tracer results instead of using the Tracer window.

©2025 Pegasystems Inc

4. **Optional:** Set breakpoints to stop the Tracer at specific steps in activities.
5. **Optional:** Set watch variables to detect when processing has changed the value of a property.
6. **Optional:** Select a requester session other than your own current session to trace. By default, the Tracer traces your own requester session.

- **Configuring Tracer settings**
- **Sending Tracer output to a file**
- **Setting breakpoints in the Tracer tool**
- **Setting watch variables in the Tracer tool**
- **Selecting a session to trace**
- Tracing and capturing events
- Tracer best practices

Configuring Tracer settings

Maintain the quality of your application and quickly locate any issues by tracing selected events with the Tracer tool. Analyze the Tracer results conveniently by defining what information you want to include in the output.For example, you can specify when you want the Tracer tool to pause processing, or how many events you want to display in the output.

1. In the footer of Dev Studio, click the Tracer icon.
2. On the **Tracer** toolbar, click Settings.
3. In the **Tracer Settings** window, in the **Events to trace** section, select checkboxes of events that you want to include in the Tracer output:
   - To trace when an event starts, select the checkbox in the Start column.
   - To trace when an even ends, select the checkbox in the End column.
4. In the **Break conditions** section, select the conditions under which you want the Tracer to pause processing:

©2025 Pegasystems Inc

- To pause processing on the first Java exception, select **Exception**.
- To pause processing when a step ends with a Fail status in the *pxMethodStatus* property value, select **Fail Status**.
- To pause processing when a step ends with a Warn status in the *pxMethodStatus* property value, select **Warn Status**.

> **NOTE:** If you build your application on Constellation, Tracer does not support break conditions.

5. In the **General options** section, select the options that determine the quantity of output in specific conditions:
   - To include the Java class details from those properties in the Tracer output, if processing has properties of mode Java Pages, select **Expand Java Pages**.

     This option might significantly slow processing.

   - To capture the status of local variables for Begin and End events of activities and their steps, select **Local Variables**. Tracer creates an output row each time the system sets the value of a local variable in an activity.

     For local variables with a value of Java null, the Tracer output displays the string `null – No Value` for the value of the local variable.

   - To reduce the performance impact by limiting the amount of Clipboard detail for each row in the Tracer window, select **Abbreviate Events**.
6. In the **Event types to trace** section, select event types that you want the Tracer to monitor.
7. **Optional:** To add additional event types, in the **Event Type** field, enter the event type, and then click **Add**.
8. In the **Rulesets to trace** section, select the Rulesets that you want to trace.
9. **Optional:** To trace additional pages, in the **Pages to trace** section, in the **Page name** field, enter a page name, and then click **Add**.

©2025 Pegasystems Inc

While the Tracer session runs, the Tracer watches the pages that you add. If the Tracer locates the pages, links to their content display in the **Properties on Page** window when you open an event from the Tracer output.

> **NOTE:** When you add multiple pages, the Tracer might work slower.

The option to add pages is unavailable if you select **Abbreviate Events** in the **General options** section.

10. In the **User interface** section, in the **Max Trace Events to Display** field, enter the number of trace events to display in the output.
11. Click **OK**.

- **Tracer events to trace**
- **Tracer event types to trace**
- **Rulesets to trace**
- **Reducing Tracer event output**
- **Setting the number of lines to display in Tracer**
- **Adjusting the buffer size of the Tracer header**
- **Tracer event queue overflows**
- Troubleshooting newly created Rules

Tracer events to trace

You can select the following events to trace in the **Tracer Settings** dialog box. When selected, the Tracer output includes these events. Events are most relevant for debugging activities rather than flows or declarative Rules.

©2025 Pegasystems Inc

Events to Trace	Description
Access Deny Rules	Displays a line at the start of each run of an Access Deny Rule.
Activities Start	Displays a line at the start of each activity.
Activities End	Displays a line at the completion of each activity.
Activity Steps Start	Displays a line at the start of every activity step and step iteration.
Activity Steps End	Displays a line at the completion of every activity step.
Data Transforms Start	Displays a line at the start of each Data Transform.
Data Transforms End	Displays a line at the completion of each Data Transform.
Data Transform Actions Start	Displays a line at the start of each Data Transform action.
Data Transform Actions End	Displays a line at the completion of each Data Transform action.
Exception	Displays a line for every Java exception. In most cases, processing continues after the exception.
Push notification	For Subscriptions, Tracer displays the following information: • Notification Channel • Subscription Class • Subscription Condition • Subscriber Data For Publish, Tracer displays the following information: • Notification Channel name • Source Page • Published Page • Stack trace of publish

©2025 Pegasystems Inc

Events to Trace	Description
When Rules Start	Displays a line at the start of every When Condition evaluation, including preconditions and transitions.
When Rules End	Displays a line at the completion of every When Condition evaluation, including preconditions and transitions. If selected, Tracer output shows the results of all When Rule executions from all Rulesets, regardless of your selected Rulesets.

- Tracing and capturing events
- Configuring Tracer settings
- Configuring trace conditions

Tracer event types to trace

You can trace the progress of various events that occur in the system by selecting event types to trace in **the Tracer Settings** dialog box. Examples of Rules that produce events are flows, declarative Rules, and decision trees Rules. You can also trace events related to services and database operations.

For declarative Rules, rows of the Tracer output show the operation of the declarative network, that is, the forward and backward chaining computations that occur automatically.

Label	Description
Event Type	Enter an additional event type as instructed by Pega Global Client Support and click **Add** to enable tracing for this type.
ADP Load	Trace the background thread (requestor) that loads a declare page asynchronously.
Adaptive Model	Trace Adaptive Models.

©2025 Pegasystems Inc

Label	Description
Alert	Include Alert log messages produced by your Thread in the Tracer output.
Asynchronous Activity	Trace asynchronous Activity.
Auto Populate Properties	Trace auto-populated properties.
Automation	Trace Automation Rules.
CaseType	Trace Case Type Rule calculations.
DB Cache	Show hits and misses for the Rule cache and the conclusion cache.
DB Query	Show each SQL query sent to the PegaRULES databases and its results, including expansion of the BLOB column.
Data Pages	Show Data Page Activity, including instantiations from properties, calls to data sources, and responses to those calls.
Declare Collection	Trace collection Rules.
Declare Constraint	Trace Constraints Rules.
Declare DecisionMap	Trace map values (start only).
Declare DecisionTable	Trace decision tables (start only).
Declare DecisionTree	Trace decision trees (start only).
Declare Expression	Trace Declare Expression Rules.
Declare Index	Trace Declare Index Rules.
Declare OnChange	Trace Declare OnChange Rules.
Declare Trigger	Trace Declare Trigger Rules.
Flow	Trace a flow. Rows of the Tracer output identify each flow start and end, each flow

©2025 Pegasystems Inc

Label	Description
	shape start and end, and the connectors (arrows on the flow diagram).
GenAI	Trace GenAIConnect Rules and Coach Rules.
Interaction	Start and end of a user interaction, as defined by the Performance tool.
Linked Page Hit	Trace operations on linked pages that cause retrieval of the target page from a cache.
Linked Page Miss	Record cache misses for linked properties.
Locking	Trace locking operations. Tracer output shows each lock acquired and each lock released.
Log Messages	Record each run of the Log-Message method that has the SendtoTracer parameter selected.
Parse Rules	Trace Parse XML, Parse Structure, and Parse Delimited Rules. Other parse Rule Types are not traced.
Predictive Model	Trace Predictive Model Rules.
Query resolution	Trace report queries to identify queries that run against the database instead of Eleasticsearch.
SOAP Messages	Trace detailed SOAP request and response messages for SOAP, SAP, and .NET connectors and services.
Scorecard	Trace scorecard Rules.
Services	Trace service Rules and service Activities.
Strategy	Trace strategy Rules.

PEGA ©2025 Pegasystems Inc

Label	Description
Stream Rules	Trace all types of stream Rules.

- Tracing and capturing events
- Configuring trace conditions
- Configuring Tracer settings
- About Collection Rules
- Property form: Completing the General tab - Value modes
- Log-Message method
- Tracing services

Rulesets to trace

You can select the Rulesets to trace in the **Tracer Settings** dialog box. The dialog box lists all of the Rulesets that you can access, in the order they appear in your Ruleset list, based on your Access Group and other sources.

The Tracer always enforces Ruleset access controls. Whether you trace your own session or another user's session, Tracer output includes only information about the execution of Rules in the Ruleset that you selected.

The *Pega-* Rulesets listed last are base Pega Platform Rulesets. You can select them for tracing, but the Tracer might produce large quantities of output. You can trace flows in your application Rulesets without selecting the *Pega-ProCom* Ruleset or other base Rulesets.

- Tracing and capturing events
- Configuring Tracer settings
- Configuring trace conditions

Reducing Tracer event output

You can reduce the amount of Clipboard detail that is sent to the Tracer by using the **Abbreviate Events** option. Reducing the amount of output improves the performance

©2025 Pegasystems Inc

of the requestor session that is being traced so that elapsed time statistics during tracing are closer to the normal values, that is, the values when processing is not being traced.

When you abbreviate events, some watch variables might not operate correctly, and the **Pages to Trace** and **Local Variables** features are disabled. Do not abbreviate events if debugging requires these features.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. Click **Settings**.
3. In the **GENERAL OPTIONS** section, select **Abbreviate Events**.
4. Click **OK**.

- Setting the number of lines to display in Tracer

Setting the number of lines to display in Tracer

You can change the number of lines that are displayed in the Tracer window to manage network and workstation load. By default, the Tracer displays the 500 most recent trace lines. You can set a higher value, which requires additional workstation memory. If you only want to view Tracer output as an XML file and save it to your local system, set the number of lines to display to zero (0) to reduce the network and workstation load.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. Click **Settings**.
3. In the **USER INTERFACE** section, set **Max Trace Events to Display** to the number of lines to display. When Tracer output reaches this limit, the oldest lines are dropped from the window to allow newer lines to be displayed.
4. Click **OK**.
5. Restart the Tracer tool for the changes to take effect.

- Reducing Tracer event output

©2025 Pegasystems Inc

Adjusting the buffer size of the Tracer header

You can adjust the buffer size of the Tracer header to increase the limit for unprocessed events. By default, the system saves up to 50,000 items for unprocessed events during a Tracer operation. If the buffer exceeds this limit, Tracer processing ends.

1. Configure the *prconfig/tracer/queue/header/limit/default* dynamic system setting with the owning Ruleset *Pega-Engine* and the value of a nonnegative integer. For more information, see Creating a dynamic system setting.
2. Stop and restart (or redeploy) the server node to enable the change.

- Reducing Tracer event output
- Tracer best practices
- Updating dynamic system settings by using Java methods
- Configuring dynamic system settings

Tracer event queue overflows

Each Tracer session uses a file buffer that holds up to 500 events not yet processed for display. Some complex tracing situations reach this limit and additional events are discarded.

The following message is displayed:

```
Event queue is overflowing on server, events will be discarded.
```

To avoid this situation, select fewer Rulesets or event types, and then repeat the operation.

If this message occurs often, you can increase the file buffer size so that it can contain more events. If the problem persists after you have set a larger buffer size, your application may contain an infinite loop.

Various conditions can cause the event queue to fill up. For example, if the Tracer and the traced requestor session are running on one client workstation, then both sessions

PEGA ©2025 Pegasystems Inc

share the normal limit of two HTTP operations to the server. If the traced operations involve many HTTP operations, then the Tracer contends with other browser operations for access to the server. This contention can cause the Tracer to drain the event queue more slowly, causing an event queue overflow.

- Adjusting the buffer size of the Tracer header
- Reducing Tracer event output
- Tracer best practices

Sending Tracer output to a file

You can send the Tracer output to a file if you want to work offline with the Tracer results instead of viewing the output interactively in the Tracer window. In addition, you can suppress the event window to reduce network traffic and workstation load when you only want to view the output offline.

By default, the system saves the output as an .xml file. When tracing in a cluster, and more than one node reported events for a given RuleWatch, the system saves the output as a .zip file that contains an .xml file for each node that reported events. If only one node reported events, then the system saves the output as an .xml file.

For information about viewing the output file and to download a Windows viewer for Tracer output .xml files, see the article *Using the Tracer tool to summarize Tracer XML output in Pega 7*.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. Click **Settings**.
3. **Optional:** To suppress the event window, in the **USER INTERFACE** section, set **Max Trace Events to Display** to 0.
4. Click **OK**.
5. Perform the work that you want to trace.
6. Click **Save**. The Tracer output file is saved in the local Downloads folder.

©2025 Pegasystems Inc

What to do next:

Analyze the output offline by using the Pega-TracerViewer tool. For more information, see Offline debugging by using Pega-TracerViewer.

- Configuring trace conditions
- Tracing and capturing events

Setting breakpoints in the Tracer tool

You can set breakpoints to pause processing for specific steps in activities. When you set a breakpoint for an activity, the Tracer window acquires focus when you or the selected requestor connection starts the activity.For example, if a step in an activity triggers a Data Transform, you can set a breakpoint before the step so that you can analyze and understand how the system updates the values in the Data Transform.

NOTE: If you build your application on Constellation, Tracer does not support breakpoints. Changing configurations for breakpoints in Constellation applications might cause issues.

NOTE: You cannot view the Clipboard when the system reaches a breakpoint during processing because the requestor that the Tracer traces pauses.

1. In the footer of App Studio, click the **Open runtime toolbar** icon, and then click the **Tracer** icon.
2. On the **Tracer** toolbar, click **Breakpoints**.
3. In the **Class Name** field, select the Applies To class that stores an activity to trace.
4. In the **Activity Name** field, select an activity.
5. In the **Where to Break** list, select one of the following options:
   - To stop processing when the activity starts, select **1**.
   - To stop processing before a specific step, select the step number.

©2025 Pegasystems Inc

- To stop processing before every step, select **all steps**.

6. Click **Set Break**.
7. **Optional:** To add more breakpoints, repeat steps 3 through 6.
8. **Optional:** To remove a breakpoint, in the **Remove** column, select the checkbox, and then click **Remove**.
9. Click **Close**.

- Configuring trace conditions
- Tracing and capturing events

Setting watch variables in the Tracer tool

Set a watch variable to detect when processing changes the value of a property. After you set a watch variable, you can minimize the Tracer window and continue working. Focus returns to the Tracer window when the property value changes.

For example, if you watch a Case Status, you can see who changes the status of a particular Case, and when. You can use this information when a Case unexpectedly fails to successfully reach a resolution.

> **NOTE:** If you build your application on Constellation, the Tracer does not support watch variables. Changing configurations for watch variables in Constellation applications might cause issues.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. In the **Tracer** toolbar, click **Watch**.
3. In the **Page Name** field, enter the name of the page that stores the property to watch, or enter a parameter for the parameter page.
4. Enter the property name without a period in front of it in the field, or select to watch this page for page messages.
5. Select variables to watch:

©2025 Pegasystems Inc

- To watch a single property, in the **Property Name**, enter the property to watch.
- To watch the page for page messages, select the **Or Page Messages**.

6. Click **Set Watch**.
7. To watch more properties, repeat steps 3 through 6
8. **Optional:** To remove a watch variable, select the checkbox in the **Remove** column, and then click **Remove**.
9. Click **Close**.

- Configuring trace conditions
- Tracing and capturing events

Selecting a session to trace

By default, the Tracer traces your own requestor session. You can trace another session by using the Remote Tracer. The Remote Tracer lists every requestor that is connected to your Pega Platform server.

The first letter of the session ID identifies the requestor type:

- H – Web browser-based users
- B – Background users such as agents, service requestors other than for Service Portal Rules, and daemons
- A – External applications
- P – Access through Service Portlet Rules

You cannot trace listener processing because listeners operate as Java threads, not full requestor sessions. Use remote logging to debug listeners.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. Click **Remote Tracer**.
3. Select the session that you want to trace.
4. Click **Okay**.

©2025 Pegasystems Inc

- Configuring trace conditions
- Tracing and capturing events

Tracing and capturing events

After you have configured the trace conditions, you can begin tracing and capturing events. As you perform the work that you want to trace, the Tracer displays the traced events according to the selected trace conditions. Each event is a row in the Tracer window.

1. In the footer of Dev Studio, on the developer toolbar, click **Tracer**, or on an activity, Data Transform, or service Rule form, click **Actions** > **Trace**.
2. Click **Play** to start tracing.
3. Move or minimize the Tracer window so that you can see the Dev Studio to perform the work that you want to trace.
4. Perform the work that you want to trace, or when tracing a requestor session other than your own, wait for the system to perform the work in that session.
5. If you have set a breakpoint for an activity, and the activity starts, the Tracer window acquires focus. Click **Continue** to resume tracing. Tracing automatically resumes after one hour.
6. **Optional:** To remove a breakpoint:
   a. Click **Breakpoints**.
   b. Select the **Remove** checkbox.
   c. Click **Remove**.
7. If the value of a watch variable changes, the Tracer window acquires focus and the Property Inspector is displayed in the lower left of the window. To view the full Property Inspector display, place your pointer at the line until a yellow arrow appears and drag the dispaly up. The following fields are displayed:

   Reference – Specifies the page and property being watched.

©2025 Pegasystems Inc

Old Value – If you clicked **Continue** and the Tracer finds that the value has changed, the previous value is displayed.

Current Value – Displays the current value of the property.

8. Click **Continue** to resume tracing. Tracing automatically resumes after one hour.
9. To end the watch:
   a. Click **Watch**.
   b. Select the **Remove** checkbox.
   c. Click **Remove**.
10. Click **Pause** to stop tracing.

- Configuring trace conditions
- Tracer best practices

Tracing services

You can use the Tracer to monitor any active requestor session. However, a service usually runs in a new requestor session with a requestor ID that is not created until the service begins processing. At that point, the processing that the service performs in that requestor session occurs so quickly (in less than one second), that it can be hard to catch the event to trace it.

In this situation, it is recommended that you use the Trace Open Rule feature to trace the service Rule. You can use this feature to trace a service request that is invoked from an external client application.

The ability to trace services comes with a performance impact even when you are not actively tracing, therefore it is recommended that you disable service tracing in production environments and during performance testing unless you are troubleshooting a related issue.

Service Rule tracing is only enabled in environments where the *trace/cluster/ServiceRuleWatchMaxProductionLevel* Dynamic System Settings is greater than or equal

©2025 Pegasystems Inc

to the production level. By default, *trace/cluster/ServiceRuleWatchMaxProductionLevel* is set to 4, which means that the ability to trace service Rules would not work in a production level 5 environment until *trace/cluster/ServiceRuleWatchMaxProductionLevel* is increased to 5.

- Owning Ruleset: `Pega-RulesEngine`
- Setting Purpose: `trace/cluster/ServiceRuleWatchMaxProductionLevel`
- Value: `<greater than or equal to the production level, for example, 5>`

To use the Trace Open Rule feature to trace and debug a service Rule:

1. In the navigation pane of Dev Studio, click **Records**.
2. In the **Records** explorer, click **Integration-Services**, and then a service Rule Type.
3. In the list of service Rule instances, select the service Rule that you want to trace.
4. Click **Actions** > **Trace**.
5. In the Tracer window, click **Settings** to open the Tracer Settings window.
6. In the **Event Types to Trace** section, select **Services**.
7. In the **Rulesets to Trace** section, select the Rulesets that contain the service Rule and service activity.
8. Click **OK** to close the Tracer Settings window.
9. Run the Rule.
10. Watch the Tracer window as the Rule runs. The window includes lines for the start and end of each service Rule execution, lines for the start and end of request mapping, lines for the start and end of response mapping, and (if these options are selected in the Tracer Settings panel) lines for the start and end of parse Rules, XML Stream Rules, and HTML Rules.

Result:

The Tracer offers the following trace options for services:

©2025 Pegasystems Inc

- Services – The Tracer adds steps for when the service invocation begins and ends. Nested within those steps are entries that show when the inbound data mapping begins and ends, and when the outbound data mapping begins and ends.
- Parse Rules – The Tracer adds steps when a parse Rule (delimited, structured, or XML) begins and ends processing.
- Stream Rules – The Tracer adds steps when an HTML Rule or XML Stream Rule begins and ends processing.

- **Cluster-wide tracing of service Rules**

- Configuring trace conditions
- Cluster-wide tracing of service Rules

Cluster-wide tracing of service Rules

You can trace service Rules across a cluster. Cluster-wide tracing makes it easier for you to troubleshoot services in clustered configurations where a load balancer dynamically assigns service requests to nodes in the cluster. This functionality is especially useful for debugging mobile applications, RESTful Web Services, and other stateless service executions.

When tracing across the cluster, the trace feature only captures one trace at a time, which might result in some concurrent service executions not being traced. All other concurrent service executions in the cluster continue unimpeded. When the trace in progress completes, the next new service execution in the cluster is traced. For example, you might see traces for invocation one on node A, invocation two on node C, and invocation four on node B. This selective tracing approach ensures that tracing with a breakpoint will not stop all service executions across the cluster.

You can selectively disable cluster-wide tracing based on production level to eliminate performance impact in production by using the `trace/cluster/ServiceRuleWatchMaxProductionLevel` dynamic system setting. Set the value to

©2025 Pegasystems Inc

the maximum production level for which you want to allow RuleWatch tracing for Rule-Service- data instances. The Trace option will not be shown in the **Actions** menu for the Rule form on systems with a production level higher than the dynamic system setting's value.

> **NOTE:** Only one user can trace a service Rule at any given time. If a user is tracing a service Rule on one node and another user tries to trace the same service Rule on either the same node or a different node, an error message is displayed.

- Tracing services

Viewing Tracer results in the Tracer window

To maintain the quality of your application, you can use the Tracer window to interact with Tracer results. Each row in the window represents an event. The Tracer records selected events from the Rule executions, database operations, and other event types that you select when you configure the Tracer.

The tracer tool also supports stateless applications that you build on Constellation and that rely on Digital Experience (DX) APIs. In Constellation applications, multiple API requests can be active simultaneously during a single browser session. For greater clarity, the Tracer window categorizes the events by request ID.

1. In the footer of Dev Studio, click the **Tracer** icon.
2. In the **Tracer** window, analyze the Tracer output:
    - To learn more about the event and view a Java stack trace if an exception occurred, click the value in the **Line**, **Rule#**, or **Step** column.
    - To view the properties that were on the step page when the step began, click the value in the **Step Page** column.
    - To open the corresponding Rule, click the value in the **Name** column.

©2025 Pegasystems Inc

- To view the contents of the parameter page, click the value in the **Line** column, and then click **Parameter Page Name**.
- To view the contents of the Primary Page, click the value in the **Line** column, click **Primary Page Name**.

For example:

The following figures present the Tracer windows for a stateful, classic Pega Platform application and stateless application that relies on Constellation:

Tracer

Play Clear | Settings Breakpoints Watch Remote Tracer Save

LINE	THREAD	INT	RULE#	STEP METHOD	STEP PAGE	STEP	STATUS	EVENT TYPE	ELAPSED	NAME	RULESET
- Events for Requestor: HHGS0GN5D9OI6YWHZCCVF93S0C7NF1SV3A Node: pega-web-75c577797-8cx2b											
24458	PS2__TABTH...	105	673	When Rule Evaluation	pyWorkPage		False	When End	0.0000	Work- hasPreviousAssignment	Pega-ProcessEngine 08-01-01
24457	PS2__TABTH...	105	673	When Rule Evaluation	pyWorkPage			When Begin		Work- hasPreviousAssignment	Pega-ProcessEngine 08-01-01
24456	PS2__TABTH...	105	673	Property-Set	pyWorkPage	4		Step Begin		Work- pxSetActionProperties	Pega-ProcessEngine 08-05-01
24455	PS2__TABTH...	105	673	Call GetFlowData	pyWorkPage	3	GOOD	Step End	0.0450	Work- pxSetActionProperties	Pega-ProcessEngine 08-05-01
24454	PS2__TABTH...	105	674		pyWorkPage			Activity End	0.0390	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24453	PS2__TABTH...	105	674	Call pyGetFlowDataDefaults	pyWorkPage	12	GOOD	Step End	0.0030	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24452	PS2__TABTH...	105	676		pyWorkPage			Activity End	0.0010	Work- pyGetFlowDataDefaults	Pega-ProcessEngine 08-01-01
24451	PS2__TABTH...	105	676		pyWorkPage			Activity Begin		Work- pyGetFlowDataDefaults	Pega-ProcessEngine 08-01-01
24450	PS2__TABTH...	105	674	Call pyGetFlowDataDefaults	pyWorkPage	12		Step Begin		Work- GetFlowData	Pega-ProcessEngine 08-03-01
24449	PS2__TABTH...	105	674	Call GetFlowDataDefaults	pyFlowData	11	GOOD	Step End	0.0110	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24448	PS2__TABTH...	105	675		pyFlowData			Activity End	0.0030	Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24447	PS2__TABTH...	105	675	Call Code-Flow-Navigation.ShowHidePAD...	pyFlowData	1	Skip Step / exit re...	Step End	0.0020	Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24446	PS2__TABTH...	105	675	pyFlowData.pyClassName=="PegaAccel-...	pyFlowData		False	When End	0.0000	Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24445	PS2__TABTH...	105	675	pyFlowData.pyClassName=="PegaAccel-...	pyFlowData			When Begin		Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24444	PS2__TABTH...	105	675	Call Code-Flow-Navigation.ShowHidePAD...	pyFlowData	1		Step Begin		Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24443	PS2__TABTH...	105	675		pyFlowData			Activity Begin		Code-Flow-Navigation GetFlowDataDefau...	Pega-DecisionScience 08-01-01
24442	PS2__TABTH...	105	674	Call GetFlowDataDefaults	pyFlowData	11		Step Begin		Work- GetFlowData	Pega-ProcessEngine 08-03-01
24441	PS2__TABTH...	105	674	Call pzGetSurveyFlowData	pyWorkPage	10	Skip Step / exit re...	Step End	0.0030	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24440	PS2__TABTH...	105	674	When Rule Evaluation	pyWorkPage		False	When End	0.0010	Work- pzIsSurveyFlow	Pega-Survey 08-01-01
24439	PS2__TABTH...	105	674	When Rule Evaluation	pyWorkPage			When Begin		Work- pzIsSurveyFlow	Pega-Survey 08-01-01
24438	PS2__TABTH...	105	674	Call pzGetSurveyFlowData	pyWorkPage	10		Step Begin		Work- GetFlowData	Pega-ProcessEngine 08-03-01
24437	PS2__TABTH...	105	674	Java	pyWorkPage	8	Jump To Step	Step End	0.0020	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24436	PS2__TABTH...	105	674	@hasMessages(myStepPage)	pyWorkPage		False	When End	0.0010	Work- GetFlowData	Pega-ProcessEngine 08-03-01
24435	PS2__TABTH...	105	674	@hasMessages(myStepPage)	pyWorkPage			When Begin		Work- GetFlowData	Pega-ProcessEngine 08-03-01
24434	PS2__TABTH...	105	674	Java	pyWorkPage	8		Step Begin		Work- GetFlowData	Pega-ProcessEngine 08-03-01
24433	PS2__TABTH...	105	674	Property-Set	pyWorkPage	7	GOOD	Step End	0.0010	Work- GetFlowData	Pega-ProcessEngine 08-03-01

Tracer results

Tracer

Pause Clear | Settings Breakpoints Watch Remote Tracer Save

LINE	THREAD	INT	RULE#	STEP METHOD	STEP PAGE	STEP	STATUS	EVENT TYPE	ELAPSED	NAME	RULESET
+ Events for Requestor: HPZQMOQVRUKXGDSSLYRGN1YKGN4JARVUCA											
+ Resource: /application/v2/ui_lists/pyMyWork-D_pyMyWorkList/personalizations											
+ Resource: /application/v2/pages/pyMyWork											
+ Resource: /application/v2/messages											
+ Resource: /application/v2/pages/pyHome											
+ Resource: /application/v2/data_views/D_RequestApprovalList											
+ Resource: /application/v2/ui_lists/6a55e093s6ca1dfd43ca35fc24e7f183/personalizations											
+ Resource: /application/v2/data_views/D_RequestApprovalList/count											
+ Resource: /application/v2/data_views/D_RequestApprovalList/metadata											
- Resource: /application/v2/pages/RequestApprovalList Interaction: GET-pages-10-44-53-072-AX8O4EYIW4QS1KU6BL6Z2JOG73I3KC7JBA Node: fd85ea5f8aab0393a924dc951d794cc0 (vagrant)											
1166	STANDARD	6			ShowPageResponse_XZTWN			Data Transform End	0.0020	@baseclass pzTranslate...	Pega-API 08-04-01
1165	STANDARD	6			ShowPageResponse_XZTWN			Data Transform Be...		@baseclass pzTranslate...	Pega-API 08-04-01
1164	STANDARD	6	28		D_pzParameters			Activity End	0.0100	Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1163	STANDARD	6	28	Property-Set	D_pzParameters	5	GOOD	Step End	0.0000	Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1162	STANDARD	6	28	Property-Set	D_pzParameters	5		Step Begin		Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1161	STANDARD	6	28	Page-Remove	RulePage	4	GOOD	Step End	0.0010	Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1160	STANDARD	6	28	Page-Remove	RulePage	4		Step Begin		Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1159	STANDARD	6	28	Apply-DataTransform	D_pzParameters	2	GOOD	Step End	0.0080	Code-Pega-List pzLoadP...	Pega-SystemArchitect 0...
1158	STANDARD	6			D_pzParameters			Data Transform End	0.0050	Code-Pega-List pzSetPar...	Pega-SystemArchitect 0...
1157	STANDARD	6		For Each Page In	D_pzParameters.pxResults	4		Action End	0.0020	Code-Pega-List pzSetPar...	Pega-SystemArchitect 0...
1156	STANDARD	6		When	D_pzParameters.pxResults(4)	4.2		Action End	0.0000	Code-Pega-List pzSetPar...	Pega-SystemArchitect 0...
1155	STANDARD	6		Set .pyClassLabel	D_pzPara... <-- D_pzPara...	4.2.1		Action End	0.0000	Code-Pega-List pzSetPar...	Pega-SystemArchitect 0...
1154	STANDARD	6		Set .pyClassLabel	D_pzPara... <-- D_pzPara...	4.2.1		Action Begin		Code-Pega-List pzSetPar...	Pega-SystemArchitect 0...

Tracer results for Constellation

©2025 Pegasystems Inc

- Constellation in traditional UI
- Troubleshooting newly created Rules

Tracer results window

The Tracer helps you debug your applications and improves the quality of your software for users. By analyzing results in the Tracer window, you can obtain more information about how your application runs in the system, as well as find and fix any issues.

The way the tracer results window groups information depends on the type of your application. If you use a standard Pega Platform application, the header row of the tracer results window contains the Pega node hash and computer name. For cluster-wide tracing of service Rules, the tracer displays a separate section for each node that reported events.

For applications that rely on the Constellation, the Tracer results window groups the data by request ID. Because in the Constellation applications multiple requests can be active simultaneously, grouping events by requests ID provides more clarity, so that you can find relevant information faster.

The tracer records selected events from the Rule executions, database operations, and other event types that you select when you configure the Tracer. Each row represents an event and is color-coded in the following way:

- Gray rows represent activity processing.
- Orange rows represent events from flow, decision, or declarative Rules.
- Light blue rows represent PegaRULES database and cache operations.

The following table contains columns that you find in the Tracer results window. The values of some columns change depending on the type of event that you trace.

©2025 Pegasystems Inc

Column	Description
Line	Number of events traced, where 1 marks the oldest event.
Thread	The Thread object for the step.
Int	The interaction number for the step. The system starts tracing the total number at login and resets upon logout.
Rule#	Count of distinct activities that are traced. The count is not reset to zero if you clear all the events. When a single activity runs again later, the previously assigned number is repeated. Rules other than activities are not assigned a number.
Step Method	For an activity, the value indicates the method that you associate with a step. For a declarative Rule or decision Rule, the value indicates the start or end of a computation. For a When Condition Rule or Boolean Expression, the value identifies the Rule name or the Expression. When you trace a database operation, this column lists the specific operation performed on the database, for example commit, insert, or update.

©2025 Pegasystems Inc

Column	Description
Step Page	Name of the step page. If the Step Page column of this step is blank, the window displays the =unnamed= value. Step pages that have messages set on them, such as error messages, have a bright orange background. When you trace a database operation, this column lists the database table affected by the database operation in the Step Method column.
Step	Step number of this step. If two or more rows appear with the same step number, an iteration is in process at that step.
Status	Status of the method in the step that the system derives from the *pxMethodStatus* property, such as `Good`, `Fail`, or `Warn`. A red background marks steps that fail and that are not addressed by a transition. `Exit Iteration` marks the end of an iteration step. In this context, a red Fail row indicates an unhandled exception condition. If a method returns a `Fail` status but the step contains a transition, the Tracer row displays the status as `Good` and has a gray background. This behavior is consistent with the processing status that the next activity step to run

©2025 Pegasystems Inc

Column	Description
	perceives. The Good status reflects that the activity notes any existing error condition. When you trace a database operation, this column lists the number of bytes involved in a write to the database.
Watch	Properties for which you set up a watch variable.
Event Type	Type of event or Rule, such as `Step Begin`, `Step End`, `Activity End`, `Constraint`, `Expression`, `DecisionTree`, or `MapValue`. The `Begin` and `When End` events identify the start of a When Condition Rule or a similar test, such as in a precondition or transition.
Elapsed	For `Step End` and `Activity End` rows, the value defines the elapsed time in seconds for the step. A Tracer operation might decrease the time interval value.
Name	Full name of the Rule that Tracer traces, with all key parts. When you trace a database or cache operation, the **Name** and **Ruleset** columns are combined. The text in this combined column provides details on the Rules that the system searches for in the cache, or on the size of the database operation.

©2025 Pegasystems Inc

Column	Description
Ruleset	Ruleset and version that contains the Rule that Tracer traces.

- Constellation in traditional UI
- When Condition Rules
- Decision trees
- Decision tables
- Creating an activity
- Viewing Tracer results in the Tracer window

Offline debugging by using Pega-TracerViewer

The Pega-TracerViewer tool presents and summarizes Tracer output data, that is saved in an .xml file, in an interactive table or tree format. By using TracerViewer, you can analyze the Tracer data offline to understand what happens on your system, determine where performance bottlenecks occur, and diagnose additional issues.

> **NOTE:** The Pega-TracerViewer requires Java for launch and is provided as-is in an open source GitHub repository. The viewer is not a licensed Pegasystems product. The `README.md` file might include information about additional prerequisites, for example a Java version that the latest Pega-TracerViewer requires. Ensure that you download and use the latest version of the Pega-TracerViewer tool. For more information, see Pega-TracerViewer.

After you run the Tracer tool on Pega Platform™, save the output as an `.xml` file to your local system, and then open it in the viewer.

Key features of Pega-TracerViewer

By using the TracerViewer, you can perform the following actions:

- View the Tracer data offline after the test is complete.

©2025 Pegasystems Inc

- Organize the data in a table or in tree views.
- Sort the data, for example, by the slowest call.
- Search the Tracer tool output data.
- View Performance tool statistics for each interaction.
- View alerts and SQL statements in the context of where and when they occur.
- Group traced events by a key, such as a unique SQL statement, stream, activity, or method.

Starting with the 3.3 version, the TracerViewer supports standard Pega Platform applications that rely on Constellation. The output depends on the type of application that you trace. For Constellation, the output additionally includes DX API interaction IDs and DX API paths.

Related concepts

- Logging for VM-based deployments and containerized deployments on Kubernetes
- My Alerts display

Related reference

- Tracer best practices

Debugging applications by using Pega-TracerViewer

Fix issues, prevent bottlenecks, and analyze events in your application by using the Pega-TracerViewer tool. You can generate Tracer output in an `.xml` format so that you can continue debugging offline or share the results with other developers.

The viewer runs as an executable file from a directory on your operating system.

Obtaining and starting Pega-TracerViewer

©2025 Pegasystems Inc

1. Go to Pega-TracerViewer, and refer to the `README.md` file for instructions on how to launch the Pega-TracerViewer.
2. Download the latest version of the viewer from https://github.com/pegasystems/pega-tracerviewer/releases.
3. Extract the files into a directory on your local operating system.
   Ensure that you maintain the viewer's directory structure.

Viewing Tracer output data in the TracerViewer

4. Start the Tracer tool and configure the tool to capture the information about the areas that you want to investigate, for example, the event types to trace.
   For more information, see Configuring Tracer settings, Setting breakpoints in the Tracer tool, and Setting watch variables in the Tracer tool.

   > **NOTE:** If you plan to analyze the results in the tree view, in the Trace Options window, for events such as activities that have Begin and End options, either select or clear both checkboxes to ensure that the viewer nests the events correctly in the tree view.

5. From Pega Platform™, run the Process that you want to investigate.

   > **For example:**
   >
   > If you want to investigate the performance of a specific business Process, run that Process. As you perform actions on the system, the Tracer tool captures data according to the criteria that you set in step 4.

6. In the **Tracer** window, click **Save** to download the .xml file to your downloads directory.
7. In the header of the TracerViewer, click **File** > **Load Pega Tracer File** File.
8. In the file directory window, navigate to the .xml file with the Tracer output, and then open the file.

©2025 Pegasystems Inc

Result:

You can view the Tracer results in offline mode, as displayed in the following figures:

LINE	TIMESTAMP	THREAD	INT	R...	STEP METHOD	STEP PAGE	S...	STATUS	EVENT TYPE	EVENT NAME
68	2019-01-28 14:45:17,366	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
67	2019-01-28 14:45:17,366	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
66	2019-01-28 14:45:17,365	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
65	2019-01-28 14:45:17,364	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
64	2019-01-28 14:45:17,364	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
63	2019-01-28 14:45:17,363	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
62	2019-01-28 14:45:17,362	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		True	When	When End
61	2019-01-28 14:45:17,362	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
60	2019-01-28 14:45:17,361	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
59	2019-01-28 14:45:17,361	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
58	2019-01-28 14:45:17,360	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
57	2019-01-28 14:45:17,360	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
56	2019-01-28 14:45:17,359	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
55	2019-01-28 14:45:17,358	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
54	2019-01-28 14:45:17,358	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...		False	When	When End
53	2019-01-28 14:45:17,357	Tracer	18	0	When Rule Evaluation	D_pzTracerSettingsForRequestor [No...			When	When Begin
52	2019-01-28 14:45:17,164	Tracer	18	12		=unnamed=			Activity	Activity End
51	2019-01-28 14:45:17,163	Tracer	18	12		=unnamed=	1	GOOD	Step	Step End
50	2019-01-28 14:45:17,162	Tracer	18	12		=unnamed=	1		Step	Step Begin
49	2019-01-28 14:45:17,160	Tracer	18	12		=unnamed=			Activity	Activity Begin
48	2019-01-28 14:45:17,157	Tracer	18	11		=unnamed=			Activity	Activity End
47	2019-01-28 14:45:17,157	Tracer	18	11		=unnamed=	1	GOOD	Step	Step End

TracerViewer results

©2025 Pegasystems Inc

TracerViewer for Constellation

Related concepts

- Logging for VM-based deployments and containerized deployments on Kubernetes

Related tasks

- Viewing and resolving errors

Related reference

- Tracer best practices

Clipboard tool

Every connected Pega Platform requestor (including browser-based users, even unauthenticated guest users) has an associated temporary memory area on the server known as the Clipboard. The Clipboard has a hierarchical structure, consisting of nodes

©2025 Pegasystems Inc

known as pages, most of which have a name and an associated class. Pages act as buffers or temporary copies of object instances (of that class) that are copied from, or might later be stored into, the Pega Platform database or another database.

Use the Clipboard tool when developing and debugging to:

- Examine property values and messages associated with them
- Find the information necessary to reference a property — the page and property names
- Quickly update, add, or delete page properties
- Quickly start activities
- Search for properties or property values in all or selected pages

> **NOTE:** When using this tool for Customer Service chat interactions, be aware that a single browser session can be served by multiple requestors.

Access the Clipboard tool by clicking **Clipboard** in the developer toolbar, which is located in the footer of Dev Studio. As you interact with the Clipboard tool at your workstation, you see a static snapshot copy of your Clipboard, which resides in memory on the server.

Required privileges

The Clipboard tool is available only to users who have access to the *@baseclass.clipboardViewer* privilege. Action menu items that update the Clipboard contents are available only to users who hold the *@baseclass.clipboardViewerUpdate* privilege.

The standard Access Role `PegaRULES:SysAdm4` provides these privileges.

©2025 Pegasystems Inc

Clipboard tool for debugging

Several methods manipulate the Clipboard. As you execute activities that affect your Clipboard, you can examine the results with the Clipboard tool.

For example, when you execute an activity that contains a Page-New method, you can see the resulting named top-level page in the User Pages section. If you execute an activity that uses the Property-Set method (on a named Clipboard Page), you can see the new values.

Mobile Clipboard

Pega Mobile offers a Clipboard view similar to the Clipboard that you use on your Apple Safari desktop emulator in Dev Studio. Useful for trouble-shooting and debugging your mobile app, the Clipboard is grouped by User, Declare, Linked Property, and System pages.

The Mobile Clipboard is available only to users who have access to the *@baseclass.clipboardViewer* privilege. The Access Roles `PegaRULES:SysAdm4` and `PegaRULES:ProArch4` provide these privileges.

In the Safari desktop emulator, access the Clipboard by using a tap and hold gesture on the mobile app. Display the Clipboard for the main content panel and modal/overlay windows. Tap and hold does not display the Clipboard for the main navigation pane, the toolbar menu, or a menu bar in a section. You can also add the Clipboard as a navigation menu item.

- **Using the Clipboard tool**
- **Setting property values with the Clipboard tool**
- **Adding a user page to the Clipboard**
- **Measuring Clipboard size**
- **Indirect pages**

©2025 Pegasystems Inc

- **Viewing Clipboard Pages created by unit testing a Rule**
- **Troubleshooting long-lived Clipboard Pages**
- Application debugging by using the Tracer tool

Using the Clipboard tool

The Clipboard tool displays two panels.

- The left panel displays, for a selected thread, the page structures in a hierarchical tree format. As you interact with Dev Studio, you create new Threads in a variety of ways. These include starting user Portals, opening Rules or landing pages in Portal tabs, or creating work items in a process or a wizard.
- The right panel contains the names and values of Single Value, Value List, and Value Group properties and messages for one page selected in the left panel.

To interact with the Clipboard:

1. Select a Thread from the Clipboard header. Every Thread in your requestor session has a Clipboard context. Each open tab on Dev Studio has a dedicated Thread. Thread names correspond to their tab names; work item Threads are identified by their IDs. By default, the thread for the selected tab appears when you open the Clipboard.
2. Select an aggregate object in the left panel, which may be of mode Page, Page List, Page Group, Value List, or Value Group.
3. Review or interact using menus and links in either panel.

Review page structure in the left panel

The left panel presents the entire Clipboard for the currently selected Thread as a tree, growing from the left.

©2025 Pegasystems Inc

Click a name or expand arrow to display any node in the left panel to view the page structure it contains. Click again or click a collapse arrow to hide) the pages within a page.

The Clipboard contains three groups of pages:

- `User Pages` – Top-level pages created by your normal processing, sorted alphabetically by page name. For more information on adding user pages to facilitate testing, see Adding a user page to the Clipboard.
- `Data Pages` – Read-only Data Pages (created if necessary when accessed, defined by Data Pages Rules).
- `Linked Property Pages` – Read-only pages retrieved because a harness, section, flow action or other Rule contained a property reference based on a linked property. These pages are read-only, and never updated. They are removed automatically whenever your requestor performs any Commit operation.
- `System Pages` – Top-level created pages created during authentication. These correspond to the organization, division, Access Group, application Rule, and organization unit of the operator. None of the pages are available to guest users.

Three single, system-maintained pages are always present and are displayed when you click the **Show Advanced** link at the bottom of the tree:

- `pxThread` – Known as the thread page, it provides the context of Clipboard.
- `pxRequestor` – Known as the requestor page, it contains information about your Access Roles, Ruleset list, and HTTP protocol parameters.
- `pxProcess` – Known as the process page, it contains information about the server's Java Virtual Machine and operating system.

Each top-level page may contain other pages that in turn may contain other pages, and so on. For most pages, the class of the page appears in parentheses after the page name and a single space. For group and list properties, additional parentheses appear around subscripts.

PEGA ©2025 Pegasystems Inc

Every embedded page is defined by a property. Properties of mode `Page`, `Page Group`, and `Page List` may appear as embedded on the Clipboard. Pages may contain properties (of any mode) and messages.

The Branch structure of the tree in the left panel directly corresponds to the fully qualified name of a property reference. For example, the property reference:

```
pyWorkPage.pxFlow(VacationRequest).pyConfirmationNote
```

identifies a `Single Value` property named `pyConfirmationNote` embedded on a page named `pxFlow("VacationRequest")`, an element of a `Page Group` named `pxFlow` on the top-level user page named `pyWorkPage`.

Square brackets and the term [Refers to ..] indicate a reference property. The text in the square brackets identifies the non-reference page.

Using the Tools menu

Click the Tools menu on the Clipboard header to do the following:

- Select **Analyze** to review a tabular report showing the approximate size in kilobytes of each Clipboard Page, the page name and class, the number of accesses (read or update), and date and time of the last access, and passivation history. If the *Collect Details* mode is enabled, the display contains stack traces showing how pages were created or deleted and historical information about pages no longer on the Clipboard.

- Select **Collect Details** to enable, or disable, automatic collection of detailed Clipboard size and access information. A check mark next to this menu option indicates that data collection is active. A pop-up window confirms your menu action. While enabled, select the *Analyze Clipboard* option to review the size and use of each page on the Clipboard, including pages that were removed during the data collection period.

©2025 Pegasystems Inc

> **NOTE:** To eliminate unnecessary processing, disable this facility except when you need the additional detail it provides.

Using right-click command menu

You can perform the following right-click commands on a page you select in the tree. Options that update the Clipboard are available only to users who have the hold the `clipboardViewerUpdate` privilege.

Command	Description
Refresh Page	Select to cause the Clipboard tool to access and redisplay only a single aggregate property and its elements.
Delete Page	Select to delete a page. Confirm the deletion in the resulting dialog box. **NOTE:** You cannot delete a Data Page.
Show XML	Select to display the page's underlying XML. Use this command to see `pz` properties and other Clipboard entries that aren't truly properties.
Show JSON	Select to display the page's underlying JSON, which is sent to the mobile client.
Execute Activity	Select to test a Rule without the need to create a test activity that creates the pages and properties expected as starting

©2025 Pegasystems Inc

Command	Description
	conditions for the Rule. The selected page becomes the Primary Page for the activity. An Execute Activity dialog appears and displays the class of the selected page in the Page Class field. 1. Enter the Activity Name key part. 2. Click **Get Params** to enter parameter values. 3. Click **Execute Activity** to run it.

Review property values in the right panel

When you select a page in the left panel, all `Value List`, `Value Group`, and `Single Value` properties on that page appear in the right panel, sorted by property name. Property messages appear in red text.

Click a property name to open its form in Dev Studio.

Values of `TextEncrypted` properties are encrypted. They appear as blank in this display.

The following options are available in the right panel:

Option	Description
Refresh	Click to cause the Clipboard tool to access and redisplay the properties on the current page.
Edit	Select to add a single value property to a page, modify an editable property value, or

©2025 Pegasystems Inc

Option	Description
	delete an editable property from the page. When you select **Edit**, the following occurs: • Editable property values appear in text boxes • The Add option appears above the properties list • The Delete icon appears at the end of a row containing an editable property When you are finished with your edits, click **Save** to keep your updates and save the page to the memory, or click **Cancel** to cancel them. **NOTE:** You cannot edit a read-only Data Page outside of the Data Page load and post-activity process.
Add	To add a single value property to a page, select the Edit option, select a row in the list, and click **Add**. Enter the property name and value in the Add properties dialog box. **NOTE:** You cannot add an embedded property.

©2025 Pegasystems Inc

Option	Description
Save	This option is available when you click **Edit**. Click **Save** to save your updated page to the memory. **CAUTION:** Pega Platform does not validate properties on the page. This action can introduce invalid data into the memory.
Delete	This option is available when you click **Edit**. Select a row containing an editable property and click to delete it from the page.
Discard	This option is available when you click **Edit**. Click **Discard** to cancel your updates.
Actions	Click to select the *Show XML*, *Show JSON*, or *Execute Activity* commands as described previously.

CAUTION: Changing your Clipboard contents or structure may affect the integrity of your system or your application results. Change Clipboard values with this tool only in a debugging situation. Deleting properties or saving altered pages may introduce invalid data into the memory.

Page and property messages

Clipboard page names with errors appear in red text on the left panel. Click the name to view messages associated with properties on this page in the right panel.

©2025 Pegasystems Inc

A page message is a text Clipboard value that is generated by the system and associated with a page. Similarly, a property message is a text Clipboard value that is generated by the system and associated with a property. These messages can convey error conditions, progress, or exceptions to a user.

Although messages appear as values on the Clipboard, they are not defined through properties. Generally, a Clipboard Page containing messages cannot be saved, because typically, the message indicates that the page, or a property on it, is invalid. The value of a property might not meet the requirements of a permanent instance of the page's class because of missing or incorrect data.

Property messages are associated with a single property and value. They might indicate that the property value is not valid. Page messages are associated with an entire page.

When a workstation user submits an HTML form, previous page messages corresponding to the input are cleared, and property messages are cleared for any value that changed.

Properties and pages not found on the Clipboard

The following items are not visible on the Clipboard:

- Properties with Internal status (properties with names that start with `pz`). These standard properties support internal Pega Platform operations. To view these properties you can override the *@baseclass.pyShowInternalProperty* When Rule or right-click in the left panel and select *Show XML*.
- Primary Pages of some activities. The parameter page of an activity. Use the Tracer to view these pages.
- Properties of mode `Java Object`, `Java Object Group`, and `Java Object List`.

> **NOTE:** The `Application` page contains much, but not all, of the properties that make up the requestor's application Rule.

©2025 Pegasystems Inc

- Clipboard tool
- Adding a user page to the Clipboard
- Page names and reserved pages

Setting property values with the Clipboard tool

Maintain the best quality of your application by testing it with the Clipboard tool. To test or debug your application, set values for the properties that your application logic uses, and then check whether the application logic behaves as expected.For example, to test the logic of a child Case Type, before you finish creating its parent Case Type, you can provide a sample of the data that the child Case Type normally inherits from its parent.

For relevant training materials, see the Clipboard data module on Pega Academy.

1. In App Studio, navigate to an element that you want to test.

   For example:

   To test a child Case, in the navigation pane of App Studio, click **Case Types**, click a Case Type that you want to open, and then click **Preview**.

2. In Dev Studio, navigate to an element that you want to test.

   For example:

   To test a child Case, in the navigation pane of Dev Studio, click **Case Types**, click a Case Type that you want to open, and then click **Preview**.

3. In the footer of App Studio, click **Toggle runtime toolbar** > **Clipboard**.

   Result:

   The **Clipboard Viewer** window opens. The navigation pane lists pages in the memory that are associated with the thread that you select in the **Thread** list.

©2025 Pegasystems Inc

4. In the footer of Dev Studio, click **Clipboard**.

> **Result:**
>
> The **Pega Clipboard** window opens. The navigation pane lists pages in the memory that are associated with the thread that you select in the **Thread** list.

5. In the navigation pane, click the page that stores the values that you want to edit.
6. In the **Clipboard page** section, click **Edit**.
7. In the Value column, provide a value for the property that you want to test or debug.

> **For example:**
>
> Set the value of a property to `false`, as shown in the following figure:

Editing property values in the Clipboard tool

8. Click **Save**.
9. Populate the Process that you want to test or debug with the new Clipboard data.

> **For example:**
>
> Bring the Case to a resolution.

10. Verify that the field values match the test values.

©2025 Pegasystems Inc

- Clipboard tool
- Using the Clipboard tool

Adding a user page to the Clipboard

Quickly exercise your application logic in different scenarios by manually adding user pages in the Clipboard tool to mock the state of the application. For testing purposes, you can add user pages manually when your application is still in development. As a result, development efforts can continue in parallel rather than in sequence, which helps you deliver working software more rapidly.For example, in an insurance claim Case, you want to test a Process that includes a Load Data Page step, but the Case currently lacks a Data Model. In this situation, you can use the Clipboard tool to create mock pages to verify that the Process runs correctly against a proposed model.

You add a user page during application development in the following situations:

- To avoid building logic into your application that provides mock values for application validation.
- To manually test your application against a data structure that is not available yet.

For relevant training materials, see the Clipboard data module on Pega Academy.

1. In App Studio, navigate to an element that you want to test.

   For example:

   To test a Case, in the navigation pane of App Studio, click **Case Types**, click a Case Type that you want to open, and then click **Preview**.

2. In the footer of App Studio, click **Toggle runtime toolbar** > **Clipboard**.

   Result:

   The **Pega Clipboard** window opens. The navigation pane lists pages in memory that are associated with the selected thread.

©2025 Pegasystems Inc

3. In the **Thread** list, select the thread in which you want to add a user page.
4. In the navigation pane, right-click **User Pages**, and then click **Add Page**.
5. In the **Add New Page** dialog box, in the **Page Name** field, enter a name for the new page.
6. **Optional:** To create a list of properties based on a specific class, in the **Page Class** field, enter or select the class to which the user page belongs.
   Otherwise, the page that you create is classless.
7. **Optional:** To provide values in the user page by using a Data Transform, in the **Data Transform** field, specify the Data Transform that supplies the initial values for the user page.
8. If the Data Transform sends parameter values to the user page, you can provide these values by clicking **Get Params**, and then entering the parameter values that you want to set in the appropriate fields.

For example:

Set the value of a field to the name of a product, as shown in the following figure:

Adding a user page with a Data Transform

©2025 Pegasystems Inc

9. Click **Submit**.

> **What to do next:**
>
> Manually set the values for particular properties. For more information, see Setting property values with the Clipboard tool.

- Clipboard tool
- Using the Clipboard tool

Measuring Clipboard size

The Clipboard display shows the contents of the Clipboard, but not its size in bytes. Large Clipboards can affect performance because memory in the Java Virtual Machine (JVM) supporting the Pega Platform™ holds the Clipboards of all requestors.

You can use the Performance tool to see the size of your Clipboard in bytes, or to track the growth and contraction of your Clipboard over time.

1. In the footer of Dev Studio, on the developer toolbar, select **Performance**.
2. Click **Add reading with Clipboard Size**.

Use Admin Studio to determine the size of each requestor's Clipboards and the size of each page. For more information, see the Admin Studio help.

1. Click Admin Studio > **Resources** > **Requestors**.
2. In the options menu, click **Analyze Clipboard**.

- Clipboard tool
- Using the Clipboard tool

©2025 Pegasystems Inc

Indirect pages

An indirect page is a page that the system finds by searching the Clipboard at run time. The page reference to an indirect page starts with the keyword `prompt` followed by the page name.

> **NOTE:** This type of page is deprecated but still supported for Rules that used this feature prior to Pega 7.1.

With a Call or Branch instruction, the calling activity can identify parameters to the target activity by using indirect page references. For example, if you identify a page as promptALPHA, the system searches the Clipboard for pages named ALPHA.

You cannot use symbolic page names (such as `primary` or `param`) after the keyword `prompt` ; an explicit page name is required.

You can use indirect pages in these Rule Types:

- Activities
- Correspondence
- Harness
- HTML
- JSP
- Paragraph
- Parse Infer
- Parse Structured
- XML

On the Pages & Classes tab of these forms, indicate that a page referenced in the Rule is an indirect page by setting the Mode field to `prompt` .

- Page names and reserved pages

©2025 Pegasystems Inc

Viewing Clipboard Pages created by unit testing a Rule

After running a Rule, you can open the Clipboard tool and examine the output as it appears on the resulting Clipboard Pages.

The following Clipboard Pages are created when you unit test a Rule:

- RuleToRun — The Clipboard representation of the Rule that you tested.
- runRulePage — Holds the output from the Rule.
- temp_ pages — Pages created or copied by the Run Rule feature when it ran the Rule. The names of these pages begin with the literal temp_.
- pySimulationDataPage — For service Rules, a page of the helper class *Data-Admin-IS-ClientSimulation*. Contains information about the simulated request and response.

- Unit testing individual Rules

Troubleshooting long-lived Clipboard Pages

Clipboard pages that remain unused for long periods could indicate a design or implementation flaw that can hurt performance. For example, the application created a page but neglected to use the Page-Remove method to remove the page after it was no longer needed.

When building or testing an application, you can discover whether your own processing has created such "orphan" or "near-orphan" page.

1. In the footer of Dev Studio, in the developer toolbar, click **Clipboard**.
2. After the Clipboard opens, select **Tools** > **Analyze**.
3. In the resulting grid of data, look in the **Est Size (KB)**, **Last Passivation**, and **Last Activation** columns to identify any pages that were automatically passivated but never reactivated.
4. Research whether such pages are needed at all, and whether they can be removed at specific points in the application, to reduce Clipboard size.

PEGA ©2025 Pegasystems Inc

- Reporting on passivated requestors

Analyzing application performance with PAL

Improve the performance of your application with insights from the Performance Analyzer of Logs (PAL) tool. For example, you can use the PAL tool to measure the amount of system resources that your application uses when processing a single requestor session.

PAL supports both applications that you build on traditional UI architecture and Constellation. By using PAL, you can analyze processing that occurs in Dev Studio, as well as Pega Digital Experience (DX) API sessions. For more information about DX API, see the Pega Digital Experience (DX) API Community page.
You can also use PAL to debug Constellation applications directly from a user web portal, as shown in the following figure:

Opening PAL from a web portal

As a result, you can analyze actions with performance issues, identifying APIs that consume significant system resources.

> **NOTE:** To open PAL from a user Portal, ensure that you have the *OpenDeveloperForm* and *clipboardViewer* privileges.

©2025 Pegasystems Inc

> **NOTE:** In case of node passivation, the UI might stop being functional. To fix the issue, close the tab of the PAL profiler and launch it again from the user portal.

1. In the header of Dev Studio, click **Configure** > **System** > **Performance** > **PAL**.

   > **Result:**
   >
   > The PAL landing page opens, with data divided into two sections, **Requestor Performance Analyzer** and **Background Performance Analyzer**. You can also access cumulative PAL information by clicking **Performance** in the footer of Dev Studio. Then, the data that the PAL tool gathers is displayed in a new window.

2. Clear any previously recorded entries by clicking **Reset data**.
3. Above the PAL data, add a marker on which to base your analysis:
   - To add a simple reading, click **Add reading**.
   - To add a reading that includes Clipboard data, but takes longer to compute, click **Add reading with Clipboard size**.

   > **NOTE:** Adding reading with Clipboard data is available only for applications that you build with traditional UI architecture.

   > **TIP:** When adding a reading, the best practice is to define points that identify what occurred during that reading. For example, use one reading fo each Flow Action or screen render, depending on what process you want to measure.

4. Perform actions that you want to analyze, for example, launch the Work Portal and create a Case.
5. View additional information about the readings:

©2025 Pegasystems Inc

- To view information about the first reading, click **INIT**.
- To view the difference between the last two full readings, click **DELTA**.
- To view information about all readings, click **TOTAL**.

The following figure shows a window with cumulative PAL results:

Add reading | Reset | Save | Show graph | Help

TOTAL	53	53	116.36	17.68	1.88	74.56	0.00	7.35	27.82	0.00	0.00	0.00	83	4,034	18,283	2,578	1,792	0	6,702	26	2,017,604	3,251,636,464
	Int #	Int Count	Server Elapsed	RA Elapsed	Rule I/O Elapsed	RDB I/O Elapsed	Connect Elapsed	Other I/O Elapsed	Total CPU	RA CPU	Rule CPU	Other CPU	RA Count	Rule Count	Total Rules Used	Activity Count	RDB I/O Count	Connect Count	Other I/O Count	Alert Count	Total Bytes	Memory Allocated
DELTA	53	20	39.50	0.12	0.41	26.27	0.00	2.50	7.80	0.00	0.00	0.00	2	361	3,124	658	796	0	3,197	11	484,268	635,680,208
DELTA	33	7	12.56	9.86	0.19	0.41	0.00	0.80	8.44	0.00	0.00	0.00	5	1,642	1,554	142	257	0	274	5	139,267	1,015,479,680
INIT	26	26	64.30	7.70	1.27	47.88	0.00	4.06	11.58	0.00	0.00	0.00	76	2,031	13,605	1,778	739	0	3,231	10	1,394,069	1,600,476,576

PAL results

For more information about reading performance data, see the Pega Academy article Analyze performance data.

6. View additional information about a selected reading:
   - To view information about a DX API session, click the corresponding DX API.
   - To view information about processing that occurs in Dev Studio, click **Studio interactions**.

   The following figure shows a list of DX API sessions and Studio interactions that you can analyze:

©2025 Pegasystems Inc

	Int #	Int Count	Server Elapsed	RA Elapsed	Rule I/O Elapsed	RDB I/O Elapsed	Connect Elapsed	Other I/O Elapsed
TOTAL	53	20	39.50	0.12	0.41	26.27	0.00	2.50
/application/v2/portals/WebPortal	1	1	1.01	0.03	0.04	0.05	0.00	0.24
/application/v2/data_views/D_pxAvailableCaseTypesForPortal	1	1	1.03	0.00	0.13	0.06	0.00	0.18
/application/v2/data_views/D_pyGetCurrentUserNotificationsCount	1	1	0.73	0.00	0.01	0.06	0.00	0.18
/application/v2/pages/pyHome	1	1	3.44	0.00	0.04	0.10	0.00	0.32
/application/v2/data_views/D_pyMyWorkList	1	1	0.95	0.00	0.00	0.13	0.00	0.11
/application/v2/data_views/D_pyMyFollowedCases	1	1	0.91	0.00	0.00	0.05	0.00	0.17
/application/v2/feeds/pyDashboardFeed	1	1	1.63	0.00	0.02	0.06	0.00	0.40
/application/v2/mention_types	1	1	0.47	0.00	0.00	0.03	0.00	0.13
/application/v2/data_views/D_pyC11nOperatorsList/metadata	1	1	1.06	0.00	0.01	0.25	0.00	0.21
Studio Interactions	44	11	28.29	0.06	0.16	25.48	0.00	0.57

DX API sessions

PAL data divided into DX API sessions and Studio interactions

7. **Optional:** To download the results as a spreadsheet, click **Save data**.

Related reference

- Tools for debugging Constellation

My Alerts display

You can view the current Alert log on the server node produced by your own requestor session by using the My Alerts display. Click **Issues** on the Pega Platform™ developer toolbar, which is located in the footer of Dev Studio.

> **NOTE:** This display is limited to the current Alert log; alerts in older logs are not visible. Depending on your log file system settings, a new log may start each time a node is started, at a specific time each day, or on some other basis.

The following summary information is displayed for each alert:

©2025 Pegasystems Inc

- **Date and Time** – Date and time of the alert, converted from GMT to the time zone of the server.
- **Alert Type** – A text description of the alert type. For example, the text `BrowserInteraction` corresponds to alert type PEGA0001, interaction times. Other types are `Database` (`PEGA002` to `PEGA007`).
- **Value** – The measured value of the Key Performance Indicator, recorded in seconds, as a count, or in bytes.
- **Interaction** – The sequence number of the browser-to-server interaction since the requestor session began, as indicated by the Performance tool display.
- **Work Pool** – Work pool that the requestor is using, or `none`.
- **Last Input** – A portion of the URL received in the most recent interaction. Typically the text Stream= or Activity= and the name of an activity or stream Rule run by the URL.
- **First Activity** – For alerts from interactive requestors, the first activity or stream run by this requestor in the current interaction. (This activity or stream may or may not have caused the alert.)

The alerts are ordered by date and time; the first row is the most current. Click the arrow in the first column to expand the row to view more details about the alert.

> **NOTE:** When using this tool for Customer Service chat interactions, be aware that a single browser session can be served by multiple requestors.

Display options

By default, 20 alerts are displayed on the page.

- Click a numbered link, **next>**, or **<previous** to scroll back and forth through the list.
- Click **This Session Only** to limit the display to alerts produced by your current requestor session.

©2025 Pegasystems Inc

- Click **All My Sessions** to include alerts from the current session and other sessions with your current Operator ID.
- Click **Performance** or **Security** to toggle the display between security-related alerts (SECnnnn) and performance-related events (PEGAnnnn).
- Click the **Options** link to view or set log filtering criteria. You can set the following options:
  - **Lines per page** – Enter the number of alerts to display on a page. The valid values are 1 through 200.
  - **Number of pages presented** – Set a maximum number of pages to present as numbered links, between 2 and 20.
  - **Filter by** – Optional. Enter a text string to limit the display to only alert lines containing an exact match anywhere within the line. Leave blank for no filtering. Case is not significant. For example, enter Smith@Alpha.com to find lines containing this value, or containing SMITH@alpha.com.
  - Click **Apply** to update the display with the new filter criteria.

- **Parsing the alert log**
- **Suppressing sensitive data in alerts**
- **Summarizing the alert log**
- **Alerts**
- **Alert format**

Parsing the alert log

You can import the alert log into Microsoft Excel for viewing, searching, sorting, or other analysis using the Text Import Wizard in Microsoft Excel. Fields in the alert log are delimited by an asterisk.

1. Start Microsoft Excel.
2. Click **File** > **Open**.

©2025 Pegasystems Inc

3. Navigate to the `PEGA-ALERT-ZZZZ.log` file and click **Open**.
4. Click Delimited in the **Original data type** group.
5. Click **Next**.
6. Select **Other**.
7. Enter a single asterisk as the delimiter.
8. Click **Next**.
9. Click **Finish**.

- Alert format
- Summarizing the alert log

Suppressing sensitive data in alerts

The alert log identifies the names of activities and other Rules that have been executed. By default, PEGA0002 to PEGA0007 alerts also show parameter values and SQL statements containing property values. You can configure the `prconfig/suppressInserts/default` and `prconfig/includeParameterPage/default` dynamic system settings (DSS) so that parameter values and other potentially sensitive data values are omitted in prepared statement inserts:

1. Add the following DSS:
   - *prconfig/alerts/database/operationTimeThreshold/suppressInserts/default* with the value of `true` and the owning Ruleset *Pega-Engine*
   - *prconfig/alerts/general/includeParameterPage/default* with the value of `false` and the owning Ruleset *Pega-Engine*

   For more information, see Creating a dynamic system setting.

2. Stop and restart or redeploy the system.

- Alerts
- Parsing the alert log
- Summarizing the alert log
- Alert format

©2025 Pegasystems Inc

- Configuring dynamic system settings

Summarizing the alert log

You can parse, consolidate, and summarize alert logs using the PegaRULES Log Analyzer or services such as Elastic Stack.

For information on the PegaRULES Log Analyzer, see the Pega Community articles *Understanding the PegaRULES Log Analyzer* and *How to use the PegaRULES Log Analyzer*.

- Alerts
- Parsing the alert log
- Suppressing sensitive data in alerts
- Alert format

Alerts

You can view or download the current alert log to see important notifications about the functioning of your system.

Threshold values in dynamic system settings (DSS) affect the operation and criteria of many alerts.

If an article about an alert lists default `prconfig.xml` settings, as a best practice, you can change those settings by using DSS. For example, to enable the PEGA0001 alert, you create the *prconfig/alerts/browser/interactionTimeThreshold/enabled/default* DSS, and then set its value to `true`. For more information, see Creating a dynamic system setting.

> **NOTE:** These changes take effect the next time your system starts.

For more information about alerts in Pega Platform, see Alerts overview.

- My Alerts display

©2025 Pegasystems Inc

- Alert format
- Parsing the alert log
- Suppressing sensitive data in alerts
- Summarizing the alert log
- Configuring dynamic system settings

Alert format

Each line of the log contains multiple fields, delimited by a single asterisk character. Each line contains the following fields. The first column of the table is the Excel column after you have imported the file into Excel.

Column	Field	Description
A	Time stamp	Date and time of the alert event in UTC format.
B	Version	Displays the header version for the alert. For example, "8".
C	Message ID	Identifier of alert type, for example `PEGA0011`.
D	KPI Value	Observed duration in milliseconds or count of value, known as the key performance indicator.
E	KPI Threshold	Threshold value for this alert type, in milliseconds or count.
F	ServerID	Node ID for the node that is the source of the alert event.
I	Requestor ID	Session identifier; the first letter (A, B, H, or P) identifies the requestor type. • A - requestor is being used by an application • B - batch requestor used by agent processing • H - requestor is being used by a user (HTTP interaction)

©2025 Pegasystems Inc

Column	Field	Description
		• P - requestor is used for portlet support
J	User ID	Operator ID of requestor, or `none`.
K	Work pool	Work pool that the requestor uses, or `none`.
L	Rule application name version	Rule application and version used at the time of the alert.
M	Encoded Ruleset	Hash code identifying the requestors Ruleset list.
N	Checkout Enabled	`Y` indicates that the activity in the Activity field is in a personal Ruleset; that a developer has checked out the activity.
O	Interaction number	For alerts from browser sessions, the interaction number (as it appears on the Performance tool display).
P	Correlation ID	ID used to filter messages arriving on the request queue. By default, the requestor ID.
Q	Sequence	Sequence number for this alert in the log; unique within this log.
R	Thread	Internal name of the PRThread instance, or NA if this alert is not from a Thread.
S	Pega Thread Name	The name of the Pega thread on which the alert happened.
T	Logger	Java class that produced the alert instance.
U	Stack	An indicator in the engine process that shows the processing state when the message occurred. (Not available for all alerts.)
V	Last Input	For alerts from interactive requestors, a portion of the URL received in the most recent interaction.

PEGA ©2025 Pegasystems Inc

Column	Field	Description
		Typically the text Stream= or Activity= and the name of an activity or stream Rule run by the URL.
W	First Activity	For alerts from interactive requestors, the first activity or stream run by this requestor in the current interaction. (This activity or stream may or may not have caused the alert.)
X	Last Step	If the First Activity field identifies an activity, this field identifies the step number last completed, when available.
Y	Reserved	Reserved.
Z	Reserved	Reserved.
AA	Reserved	Reserved.
AB	Reserved	Reserved.
AC	Trace List	Colon-delimited string containing the most recent 50 operations of this requestor before the alert event. Each operation consists of two subfields in the format number:value:.
AD	PAL data	For some Message ID types, a set of selected of Performance tool statistics at the time of the alert. A colon delimited string in the format propertyname:value:. Values are in milliseconds, bytes, or counts.
AE	Primary Page class	The class of the Primary Page at the time of the alert.
AF	Primary Page name	The name of the Primary Page at the time of the alert.
AG	Step page class	The class of the step page at the time of the alert.

©2025 Pegasystems Inc

Column	Field	Description
AH	Step page name	The name of the step page at the time of the alert.
AI	Pega stack	The Pega stack at the time of the alert.
AJ	Parameter page	The Clipboard page data found on the parameter page at the time of the alert.
AK	Line	A text description of the alert.

- Parsing the alert log

Viewing and resolving errors

Errors appear if you save a form and the system detects one or more errors.

You must correct any errors before you can save the form by completing the following steps:

1. Use the information displayed in the error section of the header to understand what the issue is and how to resolve it.
2. Use the indicator icons to quickly see which tab has an issue and what fields on that tab are involved in the error condition.
3. Make your changes to the affected areas on the form and click **Save**.
   The header automatically updates to reflect the new error count. After you resolve all errors, errors no longer appear in the header, and you can save the form.

Viewing in-progress and completed wizards

You can find all wizards that have been started on the **All Wizards** landing page. You can use this landing page to clean up incomplete wizards.

1. In Dev Studio, click **Configure** > **Application** > **Tools** > **All Wizards**.
2. Enter keywords in the fields to filter results. By default, only unfinished wizards are listed.

©2025 Pegasystems Inc

3. Click a wizard name to open it.
4. Follow the steps of the wizard to resolve it.

Creating connector simulators

With the Connect Simulator functionality, test connect Rules in flows and from Data Pages before the actual data source is implemented, or when it is not available.

To simulate a connector:

1. Create an activity that sets values to the properties that are to be returned by the connector. For more information, see Creating an activity.
2. On the **Service** tab of the **Connector** Rule form, click the **Simulations** button to access the **Simulations** form.
3. On the **Simulations** form, click **Add a row**.
4. To enable simulation, in the **Simulation Activity** field, enter or select an activity.
5. Select either the **Global** or **User Session** options.
6. Click **Submit**.

   When flow execution reaches the connector shape, the activity is called.

Disabling simulations

When you no longer need to simulate the connector, you can temporarily ignore all simulators by accessing the **Connect Simulator** window and clicking the **Clear Local** or **Clear Global** buttons.

You can also disable or enable simulations on the Connectors landing page.

Connect Simulator Properties

Field	Description
Simulation Activity	Activity name (second key part) of an activity that simulates the results of the connector

©2025 Pegasystems Inc

Field	Description
	Rule identified by the previous fields in this row.
Global	Enables system-wide use of the simulator whenever flow execution reaches the connector shape. When enabled along with a User Session, the User Session overrides the Global simulator.
User Session	The simulation occurs only when the flow is started by the logged-in user. The User Session overrides a Global simulator.
	Remove the Simulation Activity.
Clear Local	Click to clear the User Session option from all Simulation Activities.
Clear Global	Click to clear the Global option from all simulation activities. This option is not available if the production level of the system is 5.
Apply Changes	Click to apply changes to the Connect Simulator.
Close	Click to close the Connect Simulator form.

The DateTime parse tester tool

Use the Date/Time Parse Tester tool to test whether a text input matches the DateTime patterns used by Pega Platform™, which are based directly on Java standards.

1. Use the Application Explorer to open the standard activity *Code-Pega-Parse.DTParseTester*.
2. Click **Actions** > **Run**. A test input form appears.
3. In the **Page** list, click **Empty test page**.

©2025 Pegasystems Inc

4. Leave the parameter values blank. Click **Run**.
5. Complete the input fields on the form:

Field	Description
Date/Time Pattern	Enter a pattern. The count of pattern letters identifies a format: • G — Era designator, for example AD • y — Year, for example 2018 • M — Month in year, for example July or 07 • d — Day in month, for example 10 • h — Hour in A.M./P.M. format, 1 to 12 • H — Hour in day format, 0 to 23 • m — Minute in hour, for example 30 • s — Second • S — Millisecond • E — Day in week, for example Tuesday • Z — Time zone, for example Pacific Standard Time
Test Case	Enter text that may contain a date or date/time formatted with the pattern.
Locale	Select: • Use default locale — The current locale of your requestor session is used. • Specify a locale — Select a locale from a list.
Timezone	Select:

©2025 Pegasystems Inc

Field	Description
	• Use default timezone — The current locale of the server node is used (not the time zone in your workstation or operator ID instance) • Specify a timezone — Select a locale from a list.

6. Click **Test Parsing**.
7. The system searches the Test Case text for a conforming date or date/time value. If found, it converts the value into a Pega Platform internal format and displays the results.

- Internationalization and localization

About the bulk Revalidate and Save tool

You can force revalidation in bulk of selected Rules or data instances of a single type.

For example, you can revalidate all operator ID instances, or all flows in a specific Ruleset and version.

The Revalidate and Save tool is useful in various situations, including the following:

- After you import a .zip file archive of Rules that may depend on Rules previously present, or that were exported from a lower-version Pega Platform system.
- To confirm that Rules pass current validation requirements.
- To propagate the effects of changes you made to some Rules to other Rules that reference those Rules.

This operation is similar to opening and then saving each selected Rule or data instance. If an instance fails validation, the previous version is unchanged.

Every saved Rule or data instance was previously validated at least once, at the time it was most recently saved. The instance was valid at that time in the environment,

PEGA ©2025 Pegasystems Inc

system, and context in which it was saved. Revalidation is desirable to confirm that nothing in the current environment makes the object invalid.

If an instance passes validation, the History tab is updated with your operator ID, the current date and time, and current system in the Updated row. For Rule instances, a history detail instance is added with your memo and the notation Bulk Update.

Starting the bulk revalidation tool

To revalidate Rule or data instances:

1. In Dev Studio, click **Configure** > **System** > **Release** > **Upgrade** > **Validate**.
2. Complete the form. See Help — Using the Revalidate and Save tool for instructions.
3. Click **Run** to begin revalidation.
4. As it executes, the Revalidate and Save tool marks valid Rules (or data instances) with a checkmark; Rules not valid are marked with a red X. Hold the mouse pointer over the red X to see the error message. Click the **Open** icon to open the instance.

NOTE:

- If an instance fails validation, the previously saved instance remains and is unmodified.
- Depending on the number of Rules or data instances selected, this processing may require minutes or hours to complete.
- Note any Rule or data instances that fail validation. After making corrections, you can click **Clear Status** and rerun the test.
- For some Rule Types, validation of an instance may fail because another instance is invalid. Running the Revalidate and Save tool a second time may allow instances that failed the first time to validate.
- You can revalidate Rules that belong to a locked Rulesets version, excluding Ruleset versions that form the Pega Platform product. The History tab and history details are updated to reflect a successful

©2025 Pegasystems Inc

revalidation; other aspects of the Rule (as visible on the form) do not change.
- Some Rules may present warning messages. Warning messages do not indicate that the Rule failed to validate or save. Use the Guardrails landing page to see Rules that have warnings.

- **Validation tool**
- **Using the Revalidate and Save tool**

Validation tool

Use the Validation tool to check the validity of Rules in your application. This tool ensures that all Rules defined in an application are valid and contain valid Rule references based on a specified application context.

The Validation tool can help you improve the quality of your application and should be used regularly. It is available for Rulesets using both Application Validation (AV) mode and Ruleset Validation (RV) mode.

Use the Validation tool:

- To identify how changes to some Rules will impact other Rules that reference those Rules. For example, if you delete or change the return type of a Rule by running the Validation tool, you will know how many Rules became invalid by that change.
- To confirm that Rules pass current validation requirements
- After critical changes or milestones, such as:
  - Changes to a Ruleset list in the application Rule
  - Changes to a built-on application
  - Before lock/export
  - After import
  - Regularly during development

©2025 Pegasystems Inc

How the Validation tool works

The Validation tool is available only to users who have access to the *ApplicationValidation* privilege. The Access Role `PegaRULES:SysAdm4` provides these privileges. The tool will quickly validate the Rules in your application, Ruleset, or Ruleset version without re-saving them. Only Rules that can be called at run time are validated; withdrawn and circumstanced Rules are not validated.

Running the Validation tool

1. Click **Configure** > **Application** > **Tools** > **Validation**.
2. In the **Select Validation** drop-down menu, select the application or individual Ruleset and (optionally) the version you want to validate.
3. Click **Run Validation**.

> **NOTE:** If you want to validate a component, ensure that the component is unlocked. Validating a locked component results in an error. For more information, see Enabling components.

A progress bar appears, indicating:

- **Total Rules** – Number of Rules associated with application
- **Processed Rules** – Number of Rules validated at given point in time
- **Invalid Rules** – Number of invalid Rules found at given point in time

Click **Cancel** at any time to stop the validation process.

You can only run the Validation tool for the selected application. To run for another application, click the **Reset Validation** button.

After validation is complete, a results grid displays a list of any Rules that failed validation, along with their associated error messages.

PEGA ©2025 Pegasystems Inc

- Expand a Rule name to display the Rule's validation error.
- Click the Rule Name to open the Rule.

> **NOTE:** When you open the Rule from the results grid, it will not contain the error messages from the validation process. To re-validate the Rule, save the Rule.

Related tasks

- Enabling components

Using the Revalidate and Save tool

You can identify the class of the instances (such as Rules or data instances) to be revalidated, and optionally a Ruleset and Version, and list instances that meet these criteria. You can also bulk validate the Rules you select.

Completing the form

Complete this form to identify the objects to be revalidated.

Label	Description
Class	Select the type of object to be listed.
Ruleset Name	Optional. If you chose a Rule Type, you many select a Ruleset to restrict which the Rules are listed.
Ruleset Version	If you selected a **Ruleset Name**, select to limit the Rules to be listed to a single version. Select a version number.

©2025 Pegasystems Inc

Label	Description
During Update Move to?	If you selected a version number, select to cause the revalidated Rules to be placed in a higher version number. Rules that fail validation are not altered. Select the new version number, from those already defined for the selected Ruleset.
Memo	Enter a brief text memo describing the purpose of this bulk revalidation. This memo appears in the detailed history of instances that are successfully revalidated.
List	Click to list the objects that meet the criteria you have entered.
Run	Click to begin processing the selected instances.
Close	Click to close the Revalidate and Save form.

Controls

Columns of the list include an **Update?** checkbox, the key of the object, a Ruleset and version (for Rules), and the status. Check at least one box before starting the bulk validation:

- Select the **Update?** checkbox for a row to include this row in the validation.
- Click **Check All** to check the Update? box in all rows.
- Click **Uncheck All** to clear the Update? box in all rows.
- Click **Inverse** to invert the Update? box value in all rows.

©2025 Pegasystems Inc

Click **Run** to begin revalidation. Click **Stop** to pause revalidation. Results are presented in the **Results** column as a check mark or red X. Hold the mouse pointer over the red X to review at least one error.

Click the **Open** icon in any row to review or correct any that fail.

After making corrections, you can click **Clear Status** and rerun the validation test.

Results

A dynamic display identifies the key of the object being processed; a counter shows total rows and rows that failed validation.

This operation is similar to opening and saving each Rule or data instance. (In addition, some but not all Ruleset prerequisites and class restrictions are checked.) If an instance fails validation, the previous saved version is unchanged.

If an instance passes validation, the **History** tab is updated with your operator ID, the current date and time, and current system in the **Updated** row. For Rules, a history detail instance is added with your Memo and the notation `Bulk Update`.

Warning messages may appear. Warning messages do not indicate that the Rule failed to validate or save. Use Guardrails landing page to see all warnings.

You can revalidate Rules that belong to a locked Rulesets version, excluding Ruleset versions that form Pega Platform. The **History** tab and history details are updated to reflect the successful revalidation; other aspects of the Rule (as visible on the form) do not change.

- About the bulk Revalidate and Save tool

Application Structure landing page

The Application Structure landing page helps you to manage and understand the Rulesets, Rules, Access Groups, and operators that make up your Pega Platform

PEGA ©2025 Pegasystems Inc

application. Options on this page display referencing applications, allow you to invite collaborators to the development process, and provides access to other applications on the system (for operators with permission to do so).

Access this landing page from Dev Studio by clicking **Configure** > **Application** > **Structure**.

- **Ruleset Stack tab**
- **Ruleset Prerequisites tab**
- **Referencing Applications tab**
- **Access Groups & Users tab**
- **Other Applications tab**

Ruleset Stack tab

Access the **Ruleset Stack** tab by clicking **Configure** > **Application** > **Structure** > **Ruleset Stack**.

This tab lists the Rulesets and Ruleset versions that make up your current application including those inherited from any built-on applications. It also allows you to lock versions and roll (increment) them to higher versions.

The display is organized by Current Application followed by the Built on Application(s) and is filtered by Ruleset and version, lock status, total number of Rules, and number of checked-out Rules.

To open an application Rule in the list, click its name.

Field	Description
Ruleset	The Rulesets and versions in the application. To open a Ruleset, click its name.

©2025 Pegasystems Inc

Field	Description
Lock	Contains an icon indicating the lock status of Ruleset versions for the application. • — Indicates that the version for any given Ruleset is locked. Also indicates a locked application Rule. • — Indicates that the version for any given Ruleset is unlocked. Also indicates an unlocked application Rule. • — Warns you that a locked Ruleset version has lower unlocked versions. Click the icon to view a list that shows you which versions are unlocked. To lock an unlocked version, select it in the list, and its Rule form opens where you can lock the version. As a best practice, leave only the highest version of a Ruleset unlocked. If the warning condition exists, you cannot roll (increment) a lower unlocked version using Lock and Roll functionality. • — Warns you that this Ruleset version does not exist but appears in your application Rule. To correct, remove the version. Click an icon to open a window that displays a list of versions, and for each version, the lock status and the number of total and checked out versioned and non-versioned Rules, including Ruleset, class, and organization instances.
Checked Out	The total number of checked-out versioned and non-versioned Rules.
All Rules	The total number of versioned and non-versioned Rules. Click a number to open a window listing the Rule Types and counts comprising the total. Click a row (including Total) in the list to drill down to a list of Rule instances comprising a Rule Type.
Lock and Roll	Click to display the Application Lock and Roll window where you can review, lock current Ruleset versions, and roll (increment) versions in a single step. You can optionally update your current application Rule to

©2025 Pegasystems Inc

Field	Description
	reflect the new versions, or create a new application Rule that contains the new Ruleset versions.
Package and Export	For the Current Application only. Click to start the Application Package wizard, to create a product Rule for this application.

In the window, select the application Ruleset versions you want to lock and roll to a higher version.

Field	Description
Lock	Select the checkbox to indicate you want to lock the Ruleset version and roll to a new version. Locking prevents a developer from saving new Rule instances to this version, or updating or deleting existing instances. A icon appears if there is a condition that might prevent you from locking the Ruleset version. For example: A higher locked version exists or a version currently has Rules checked out. A icon appears if the Ruleset version is already locked.
Ruleset Version	The Ruleset and current version. Click the + icon next to the Prerequisites label to display the Current and New prerequisites for the version. To open the Rule form for a current Ruleset and version, click its name in the Current list.

©2025 Pegasystems Inc

Field	Description
	When you lock a Ruleset version and select the Roll checkbox, you can modify the new prerequisites. By default, they are the same as the current ones. You can select different prerequisites, add additional ones, and delete ones from the New list. To add another Ruleset version to the list, click the **Add a row** icon. To open the Rule form for a Ruleset in the New list, click the **Open the Rule Form** icon next to its name. To delete a Ruleset from the New list, click the **Delete** icon next to its name.
Password	Appears when you select the Lock checkbox. Enter the password for the Ruleset version.
Roll	Displays when Lock is selected. Used to roll the Ruleset to a higher version.
Roll to Version	Displays the version that the Ruleset will be rolled to unless you indicate another version number. Defaults to the next logical number in the Ruleset numbering sequence.
Description	Current description of the Ruleset version.
Run	Select an option to run against the selected Ruleset versions. • **Do not update my application** — Locks and rolls the versions without updating the application Ruleset list. • **Update my Application to include the new Ruleset Versions** — Locks and rolls

©2025 Pegasystems Inc

Field	Description
	the versions and updates the current application Ruleset list. Prompts with an application description (current by default, which you can edit). Also prompts for a password if the current application is locked. Enter the application password. • **Create a new version of my Application** — Locks and rolls the versions and creates a new application Rule. Prompts with the new application version (one increment higher by default, which you can edit), and with an application description (current by default, which you can edit). Also prompts for a password if the current application is locked. Enter the application password.

- Application Structure landing page

Ruleset Prerequisites tab

Access the Ruleset Prerequisites tab by clicking **Configure** > **Application** > **Structure** > **Ruleset Prerequisites**.

This tab displays a tree view of Rulesets and versions and their prerequisites in the current application.

©2025 Pegasystems Inc

- Click the + icon to expand the tree to show the prerequisite Rulesets and versions. A version in blue text indicates that it is listed as a top-level version in the application Ruleset list.
- Click the name of a Ruleset in the tree to open and view its Ruleset form.

> **NOTE:** Rulesets using Application Validation mode do not have prerequisites.

- Application Structure landing page

Referencing Applications tab

Access the Referencing Applications tab by clicking **Configure** > **Application** > **Structure** > **Referencing Applications**.

This tab lists other applications that reference the current application.

Field	Description
Name and Version	This (referencing) application name and version.
Access Groups	The total number of Access Groups associated with this application.
Operators	The total number of operators associated with the Access Groups.
Ruleset	This application Ruleset.

- Application Structure landing page

Access Groups & Users tab

Access the Access Groups & Users tab by clicking **Configure** > **Application** > **Structure** > **Access Groups And Users**.

©2025 Pegasystems Inc

Use this tab to:

- See the list of Access Groups that provide access to the current application.
- See the list of users that belong to an Access Group.
- Invite people to collaborate in developing this application.

Field	Description
Invite Collaborator	Click to invite people (via email) to collaborate in developing the application. For example, a Lead Business Analyst can invite other business analysts to edit and add specifications and requirements to the in-progress application. They might also invite subject matter experts to review the specifications. Upon clicking the link, the Invite Collaborator window opens for you to specify the information needed to send email invitations to the invitees. See Invite Collaborator window — Inviting collaborators.
Access Group	The name of an Access Group referenced by the application. Select a name to see the list of users that reference that Access Group, and so have that level of access into the application.
Operator Name	The names of operators who reference the Access Group.
Operator ID	The IDs of operators who reference the Access Group.

- Application Structure landing page

©2025 Pegasystems Inc

Other Applications tab

Access the Other Applications tab from the header of Dev Studio by clicking **Configure** > **Application** > **Structure** > **Other Applications**.

Administrators (operators with the *PegaRULES:SysAdm4* role or a role based on that role) can use this tab to lock and roll applications in the Pega Platform system other than those in the Ruleset stack of the current application. For example, when upgrading a Pega Platform it is considered best practice to lock all applications and create new Ruleset versions before resuming development.

This page displays all applications in the system that contain an unlocked Ruleset version and lists the Rulesets and Ruleset versions that make up each application including those inherited from any built-on applications.

To open an application Rule in the list, click its name.

Field	Description
Ruleset	The Rulesets and versions in the application.
Lock	Contains an icon indicating the lock status of Ruleset versions for the application. • — Indicates that the version for any given Ruleset is locked. Also indicates a locked application Rule. • — Indicates that the version for any given Ruleset is unlocked. Also indicates an unlocked application Rule. • — Warns you that a locked Ruleset version has lower unlocked versions. Click the icon to view a list that shows you which versions are unlocked. To lock an unlocked version, select it in the

©2025 Pegasystems Inc

Field	Description
	list, and its Rule form opens where you can lock the version. As a best practice, leave only the highest version of a Ruleset unlocked. If the warning condition exists, you cannot roll (increment) a lower unlocked version using Lock and Roll functionality. • ⚠ — Warns you that this Ruleset version does not exist but appears in your application Rule. To correct, remove the version. Click an icon to open a window that displays a list of versions, and for each version, the lock status and the number of total and checked out versioned and non-versioned Rules including Ruleset, class, and organization instances.
Checked Out	The total number of checked-out versioned and non-versioned Rules.
All Rules	The total number of versioned and non-versioned Rules. Click a number to open a window listing the Rule Types and counts comprising the total. Click a row (including Total) in the list to drill down to a list of Rule instances comprising a Rule Type.
Lock and Roll	Click to display the Application Lock and Roll window where you can review, lock current Ruleset versions, and roll (increment) versions in a single step. You can optionally

©2025 Pegasystems Inc

Field	Description
	update your current application Rule to reflect the new versions, or create a new application Rule that contains the new Ruleset versions.

In the window, select the application Ruleset versions you want to lock and roll to a higher version.

Field	Description
Lock	Select the checkbox to indicate you want to lock the Ruleset version and roll to a new version. Locking prevents a developer from saving new Rule instances to this version, or updating or deleting existing instances. A icon appears if there is a condition that might prevent you from locking the Ruleset version. For example: A higher locked version exists or a version currently has Rules checked out. A icon appears if the Ruleset version is already locked.
Ruleset Version	The Ruleset and current version. **NOTE:** Rulesets using Application Validation mode do not have prerequisites.

©2025 Pegasystems Inc

Field	Description
	Click the + icon next to the Prerequisites label to display the Current and New prerequisites for the version. To open the Rule form for a current Ruleset and version, click its name in the Current list. When you lock a Ruleset version and select the Roll checkbox, you can modify the new prerequisites. By default, they are the same as the current ones. You can select different prerequisites, add additional ones, and delete ones from the New list. To add another Ruleset version to the list, click the **Add a row** icon. To open the Rule form for a Ruleset in the New list, click the **Open the Rule Form** icon next to its name. To delete a Ruleset from the New list, click the **Delete** icon next to its name.
Password	Appears when you select the Lock checkbox. Enter the password for the Ruleset version.
Roll	Displays when Lock is selected. Used to roll the Ruleset to a higher version.
Roll to Version	Displays the version that the Ruleset will be rolled to unless you indicate another version number. Defaults to the next logical number in the Ruleset numbering sequence.
Description	Current description of the Ruleset version.
Run	Select an option to run against the selected Ruleset versions.

©2025 Pegasystems Inc

Field	Description
	• **Do not update my application** — Locks and rolls the versions without updating the application Ruleset list. • **Update my Application to include the new Ruleset Versions** — Locks and rolls the versions and updates the current application Ruleset list. Prompts with an application description (current by default, which you can edit). Also prompts for a password if the current application is locked. Enter the application password. • **Create a new version of my Application** — Locks and rolls the versions and creates a new application Rule. Prompts with the new application version (one increment higher by default, which you can edit), and with an application description (current by default, which you can edit). Also prompts for a password if the current application is locked. Enter the application password.

- Application Structure landing page

©2025 Pegasystems Inc

Pega GenAI Autopilot in application development

Pega GenAI Autopilot™ provides an enhanced and accelerated application development process in App Studio. In a conversational way, Pega GenAI Autopilot provides detailed instructions and information on building and editing applications, so that you can deliver your software faster.

> **NOTE:** Pega GenAI Autopilot is currently a preview feature that you can use after switching on the *EnableGenerativeAI* toggle. For more information, see Toggling features on and off and contact your Account Executive or Client Success Manager for more information.

Getting started with Pega GenAI Autopilot

To jump-start your application development, you can select a conversation starter that is provided on the Pega GenAI Autopilot landing page. You can also enter your own questions to obtain the information that you require for your specific use case. Pega GenAI Autopilot can respond with a sequence of steps that you need to perform, and useful information for developing the application element in question. You can also find links that lead to documentation on the topic.

The following figure shows a sample conversation in Pega GenAI Autopilot:

©2025 Pegasystems Inc

Conversing with Pega Autopilot

To accelerate your application development even further, Pega GenAI Autopilot generates buttons that open some authoring experiences for you, for example, a New Application wizard, or a dialog box to create a Case Type, as shown:

©2025 Pegasystems Inc

Creating a Case Type from Pega Autopilot

As a result, you can continue developing your application without changing context, immediately implementing suggested steps. In this example, you can also use Pega GenAI Autopilot to suggest a Case Lifecycle and Data Model for you, directly in the dialog box.

For more complex scenarios, Pega GenAI Autopilot can provide answers that include more than one topic. For example, if you want to create a Portal, Pega GenAI Autopilot provides links to create both mobile and web portals, as the following figure shows:

©2025 Pegasystems Inc

Different options in Pega Autopilot

By clicking the thumb icons under the most recent message in the conversation, you provide feedback on how useful the generated response is.

Managing conversations

For better organization of your conversations, you can rename them so that you can find a specific thread quicker:

©2025 Pegasystems Inc

Renaming a thread

You can also search for a conversation by typing a word or a phrase that is part of the conversation's name.

Pega GenAI Autopilot in Case Type creation

You can now further automate and accelerate creation of your Case Types by exploring and engaging the following solutions that Pega GenAI Autopilot offers:

Creation of data reference and Embedded Data fields

When you create a Case Type with Pega GenAI Autopilot, AI-generated Embedded Data and data reference fields are suggested for you. In the **Suggested fields** window, you can view and edit the fields before submission.

For more information, see Creating a Case Type with Autopilot.

©2025 Pegasystems Inc

Creation of Case Type workflow

During Case Type creation with Pega GenAI Autopilot, you can view and modify AI-generated workflow, that includes Stages and Steps related to Case Type name that you entered, as the following figure shows:

Generating a Case Type with Pega Autopilot

For more information, see Creating a Case Type with Autopilot.

Creation of Views

When you author an empty View from the Step configuration pane, you can involve Pega GenAI Autopilot to populate the form with suggested fields, based on the names of the Case Type, Stage, and Step.

©2025 Pegasystems Inc

Populating a View with Pega Autopilot

For more information, see Configuring forms.

Generation of email content

When you use a Send `Email` Step in your Case Type, you can use Pega GenAI Autopilot to generate message content, based on the names of the Case Type, Stage, and Step. Before submission, you can view, edit, and regenerate the message, as shown in the following figure:

©2025 Pegasystems Inc

Creating correpondence with Pega Autopilot

For more information, see step 6 in Sending automatic emails from Cases.

©2025 Pegasystems Inc