

Pega Customer Decision Hub Implementation Guide

Dynamically manage cross-channel conversations from action design to fulfillment, drive revenue, and expand customer lifetime value by implementing Pega Customer Decision Hub.

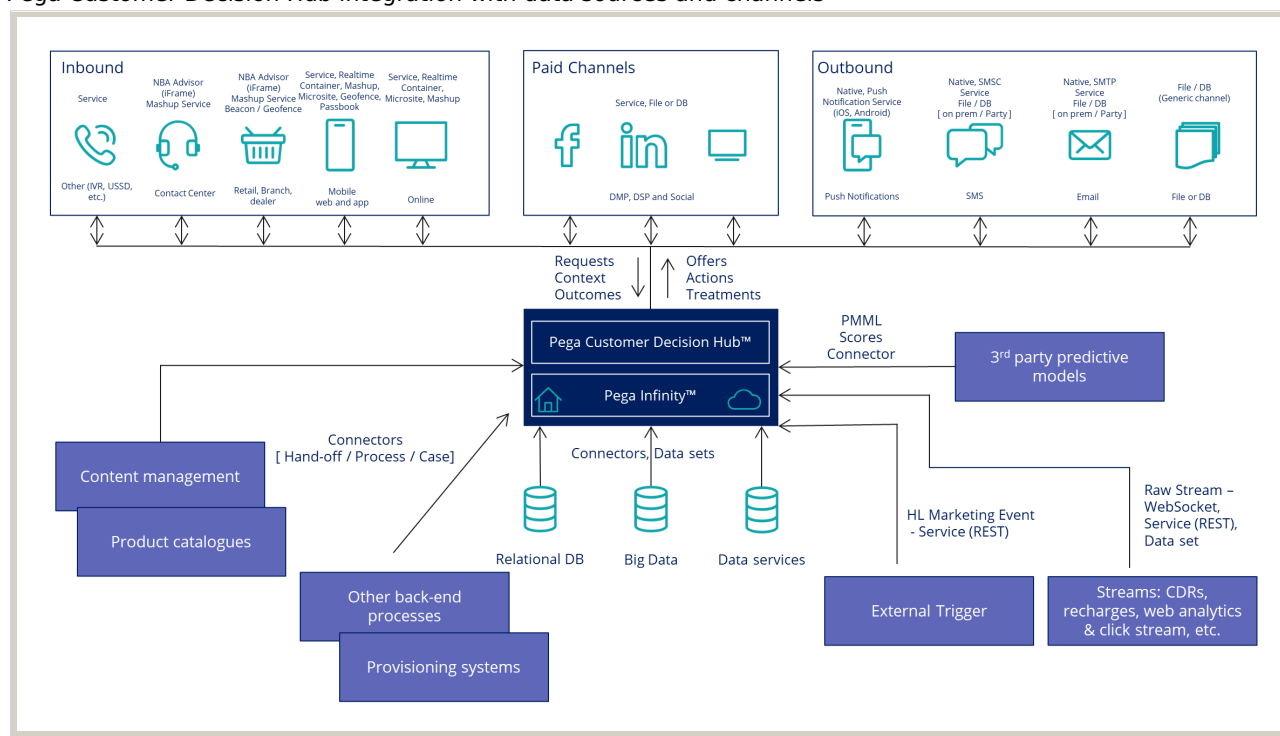
Pega Customer Decision Hub is a comprehensive customer engagement automation solution that delivers inbound action management and outbound marketing campaigns on a single platform. Pega Customer Decision Hub uses a unique combination of predictive and adaptive analytics, real-time decisioning, and business process management to meet your business needs.

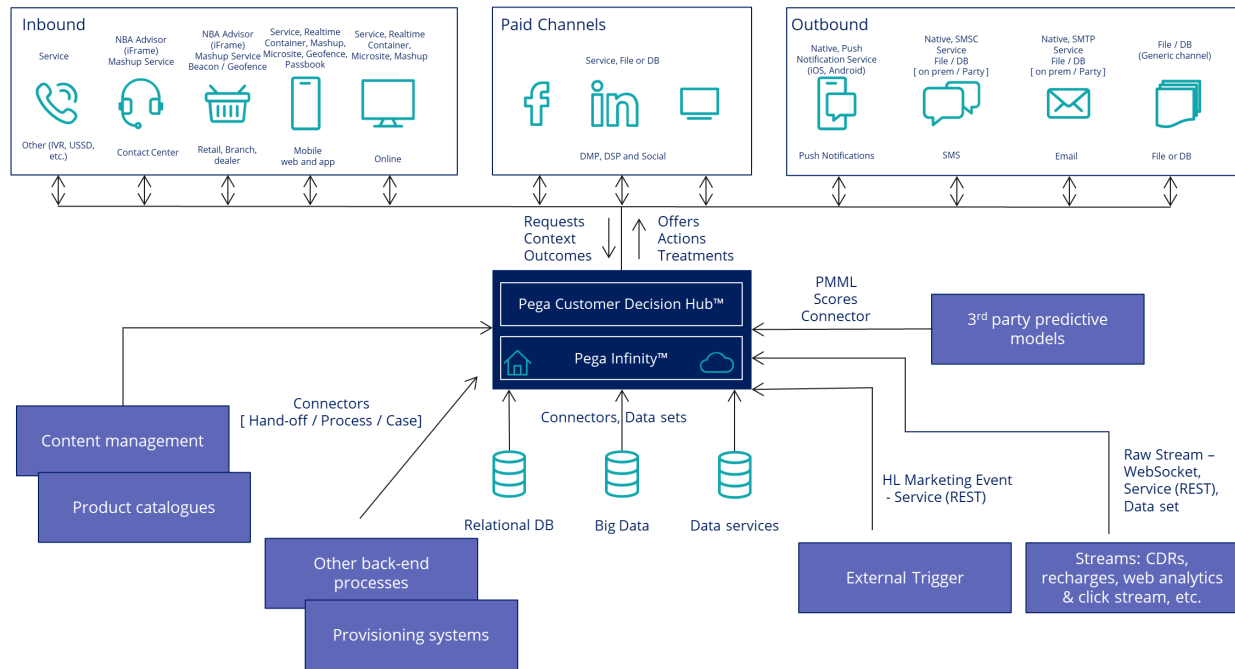
How Pega Customer Decision Hub integrates with your systems

Before you start your implementation, it is helpful to understand how Pega Customer Decision Hub integrates with your data sources and engagement channels, and how the application is structured to support daily operations and change management.

Pega Customer Decision Hub supports multiple connections to data sources and provisioning systems to obtain information about your customers and products, and other data that you use to run your business. The application also integrates with various inbound, outbound, and paid channels to help you engage with customers through marketing campaigns, sales, and support services.

Pega Customer Decision Hub integration with data sources and channels





How to use this guide

This implementation guide presents tasks in the sequence in which they are commonly performed. However, the sequence varies based on business priorities. Follow the procedures in this guide to complete the tasks for the first minimum lovable product (MLP) release. Also, use this guide during iterative releases as you configure and extend more features on top of the MLP release.

Intended audience

A project team that implements a Pega Customer Decision Hub application can use this implementation guide to enable and extend the application features to meet your business requirements. The procedures in this guide are intended for system architects who are familiar with Pega Platform and aware of your organization's data model.

Additional resources

To prepare for your Pega Customer Decision Hub implementation, review the following information:

For information about the basic functionality of Pega Customer Decision Hub, Pega Platform, and business change management, see the following courses:

- [Pega Customer Decision Hub Foundation](#)
- [Low-Code App Builder](#)
- [Business Agility in 1:1 Customer Engagement](#)

For information about the supported browsers, see [Pega Platform Support Guide Resources](#).

Implementing Pega Customer Decision Hub

Determine the next best action for every customer and drive an effective return on marketing investment by implementing the Pega Customer Decision Hub application in your system.

Start by deploying Pega Customer Decision Hub and Pega 1:1 Operations Manager in the Development (SOR) environment. Migrate the entire application to higher environments: Staging, Business Operations Environment (BOE), and Production. Configure an orchestrator and set up the data pipelines between the environments. Finally, test your implementation by running a sample change request through the BOE.

Before you begin:

1. Plan your implementation by reviewing the architecture and deployment options for Pega Customer Decision Hub. For more information, see [Deployment architecture for Pega Customer Decision Hub](#).
2. Complete the prerequisites for your deployment type. For more information, see [Prerequisites for Pega Customer Decision Hub](#).

Configuring the Development environment

1. Create and configure the Pega Customer Decision Hub implementation layer application.
For more information, see [Creating the Pega Customer Decision Hub implementation layer application](#).
2. Create and manage your customer data model.
For more information, see [Managing data](#).
3. **Optional:** To implement your application for a specific industry, apply accelerators.
For more information, see [Applying industry-specific accelerators](#).
4. Create and configure the Pega 1:1 Operations Manager implementation layer application and the overlay application.
For more information, see [Implementing Pega 1:1 Operations Manager](#).
5. Configure engagement channels.
For more information, see [Configuring engagement channels](#).
6. Package changes for the entire implementation application.
For more information, see [Packaging an application for migration](#).
7. Lock artifacts and open a new version.
For more information, see [Locking the release](#).

Configuring higher environments

8. Deploy the implementation application to the Staging, BOE, and Production environments.
For more information, see [Importing the packaged application](#).
9. Verify that the implementation application works in all environments.
10. Configure operators in the BOE:
 - a. Create operations team operators to log in to Pega 1:1 Operations Manager.
 - b. Create Revision Manager and Strategy Designer operators to log in to Customer Decision Hub.For more information, see [Creating operators in the Business Operations Environment](#).

Next steps

11. Configure an orchestrator tool, such as Deployment Manager, for data migration pipelines..
Note: Deployment Manager is used by default on Pega Cloud. For non-Pega Cloud implementations, you can use other tools, such as Jenkins.

For more information, see [Configuring Deployment Manager for data migration pipelines](#).

12. Test your implementation.
Verify whether your environments and pipelines work as designed by running a change management test case and deploying the associated revision to production.

For more information, see [Testing your implementation](#).

13. Configure the artifacts.
The business team can define the necessary artifacts, by using the Next-Best-Action Designer, which provides an easy-to-use, guided way to define, manage, and monitor next-best-actions across inbound and outbound channels.

For more information see, [Understanding Next-Best-Action Designer basics](#).

Deployment architecture for Pega Customer Decision Hub

To support ongoing application development and business operations, a Pega Customer Decision Hub implementation requires a deployment architecture with four environments and the change pipelines for deploying changes to the environments.

Deployment options

Some aspects of the implementation process differ depending on the type of environment where you deploy your Pega Customer Decision Hub application. Review the available deployment options and related implementation considerations. Pega Customer Decision Hub supports the following deployment options:

Pega Cloud Services

If your Pega Customer Decision Hub application is deployed in a Pega Cloud services environment, the Pega Cloud services team provisions the hardware for your environment and installs all of your required software. The process of creating your application is streamlined and much of the technical configuration is done for you, except for steps that only you can complete. Pega Cloud services manages deployments for security, stability, and premium support. For more information, see [Learning about Pega Cloud Services](#).

If you deployed your application in a Pega Cloud services environment, some tasks in this guide do not apply to your application. This is noted in the description of each task.

Deployment on premises

If you deploy your application on a server on the premises of your own organization, you are responsible for provisioning the hardware for your environment.

Deployment on third-party cloud

For third-party cloud deployments, the hardware is provisioned by the cloud provider.

Environments

A Pega Customer Decision Hub implementation requires the following environments:

Development

A system of record (SOR) for rules and artifacts, used by developers to make enterprise changes to your application.

Staging

A testing environment for testing changes to your application before they are deployed to production.

Business Operations Environment (BOE)

An environment that business teams use to plan, build, test, and optimize their changes, for example, create or modify actions and treatments, or introduce new predictive models. For more information, see [Business operations use cases](#).

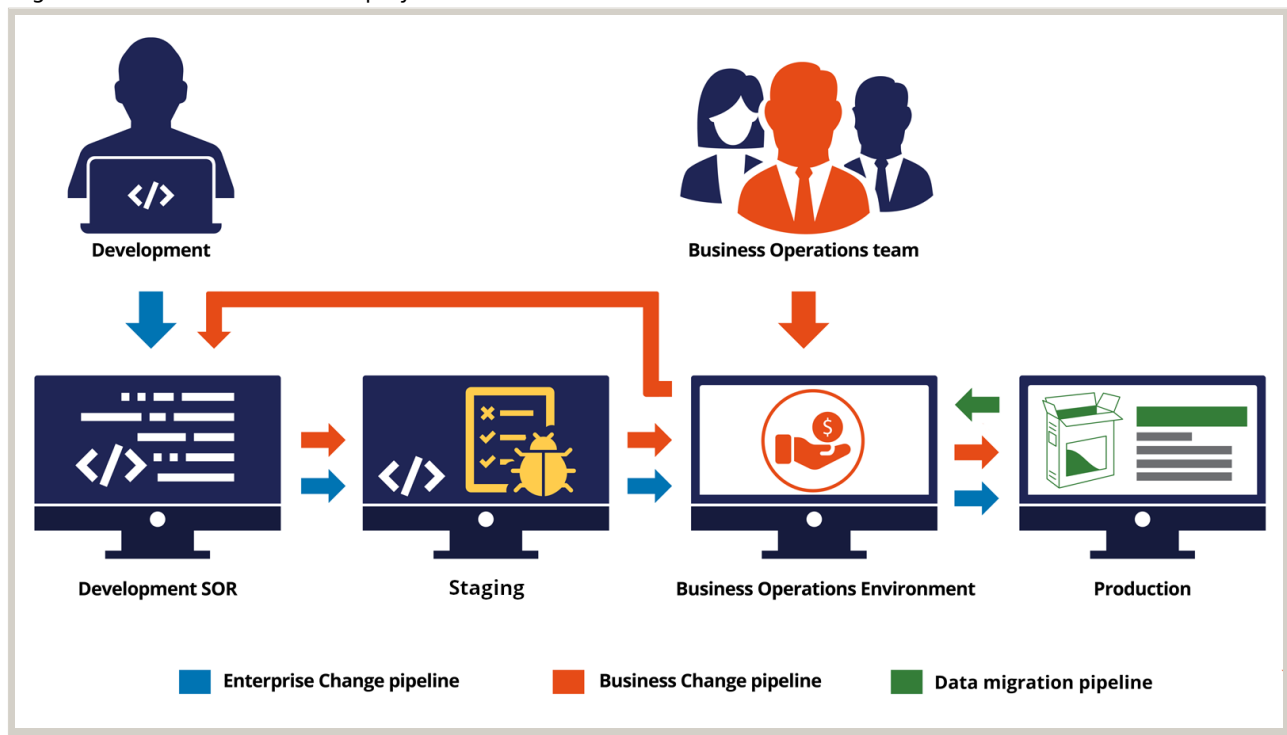
Production

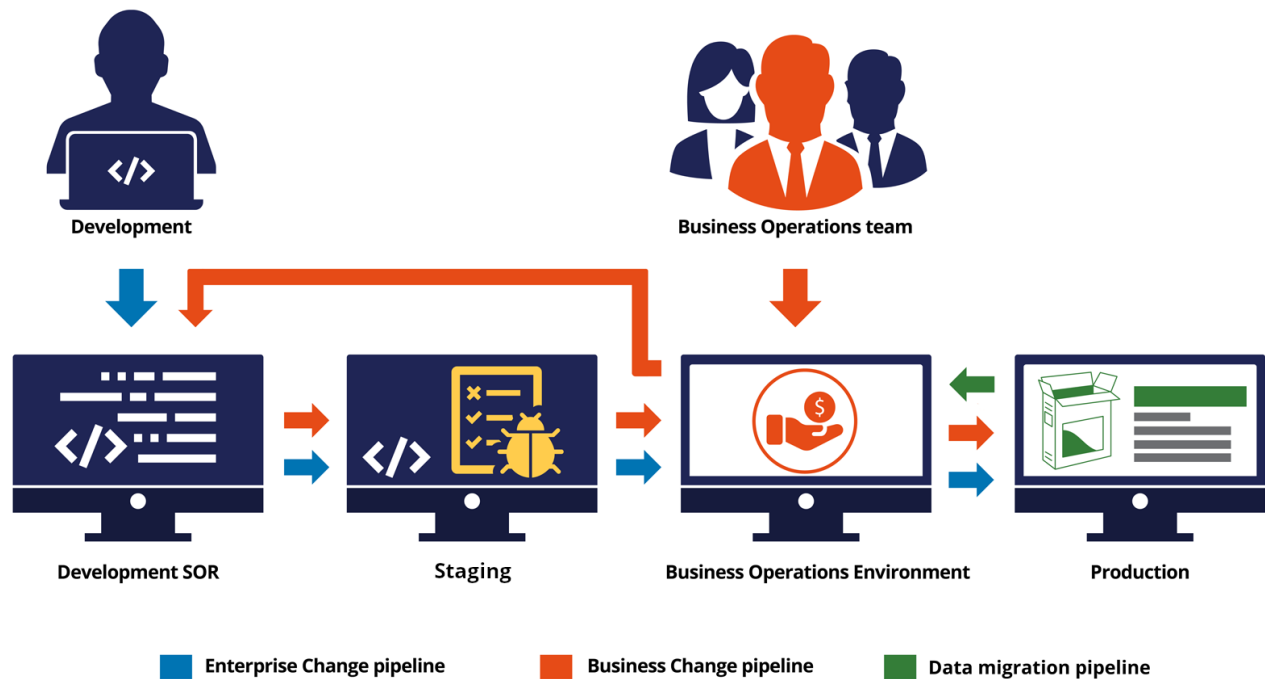
The latest stable live version of your system that is available for end users.

Note: Starting with Pega Customer Decision Hub 8.6, a BOE environment with Pega 1:1 Operations Manager is required to use the product as designed. To make business changes to a live system with Pega Customer Decision Hub, you require the Pega 1:1 Operations Manager application in a BOE environment. Pega 1:1 Operations Manager is a module of Pega Customer Decision Hub that business teams use to make these changes.

The following diagram shows the deployment architecture for a system with Pega Customer Decision Hub with the required environments and the change pipelines that connect these environments.

Pega Customer Decision Hub deployment architecture





Pipelines

As a best practice, the change management process in an organization involves the following types of changes:

Enterprise changes

Enterprise changes involve changing the application itself, for example, by modifying the user interface. Enterprise changes are done by developers in a Development system of record (SOR) environment, and then pushed to other environments in the pipeline.

Business changes

Business changes involve business-as-usual operations such as the creation of new actions, treatments, customer segments, and so on. This type of change is triggered by business users in a BOE environment, and then pushed to other environments by using the Business Change pipeline.

Data migration

The automated data migration pipeline provides you with the means to test how the changes that you make to decision strategies affect their results, without running these strategies in a live production environment. You can sample a selection of real customer and internal Pega Platform data, and then migrate the sample to the BOE environment. As a result, the simulations are based on the migrated production data, that is, customer details, adaptive model data, and interaction history, which helps you assess how your changes are reflected in real-life scenarios. For more information, see [Data migration pipeline overview](#).

Orchestrator

Your system also requires an orchestrator to manage the migration of changes through the change pipelines. In Pega Cloud services deployments, the default orchestrator is Deployment Manager.

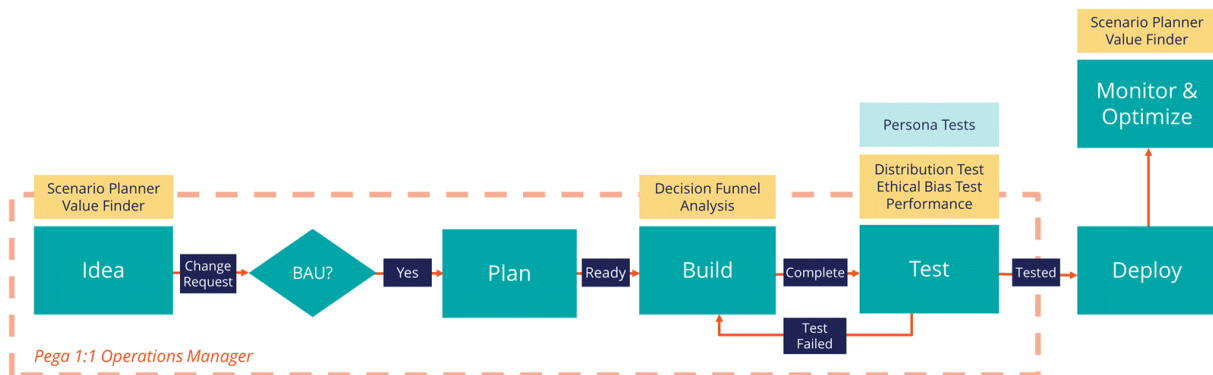
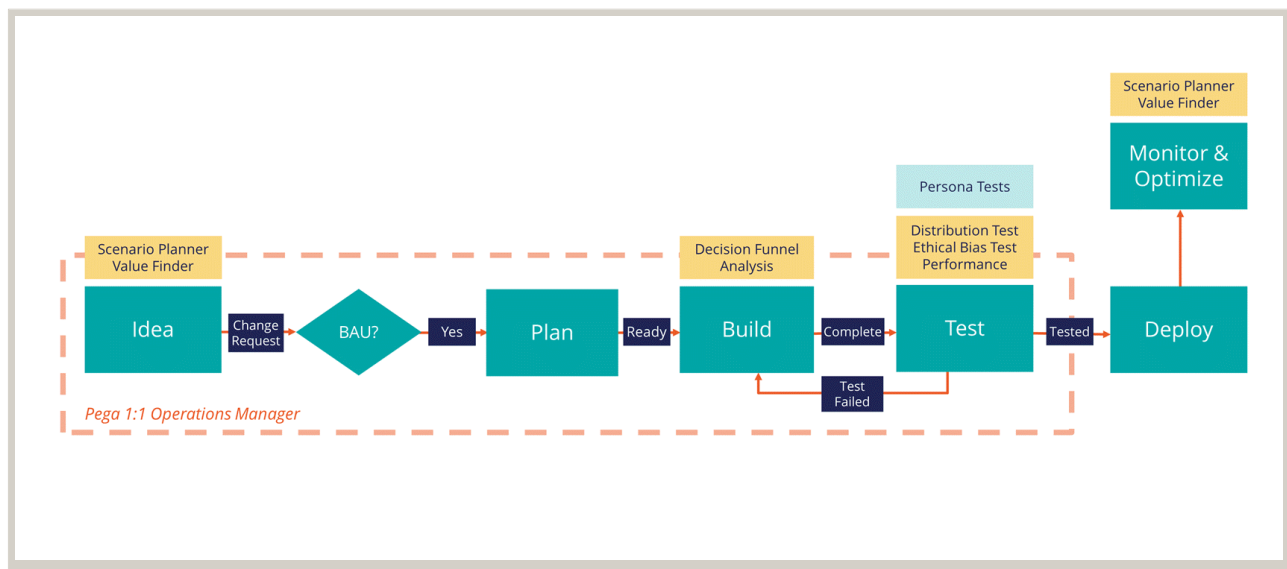
- [Business operations use cases](#)
- [Implementing Pega Customer Decision Hub](#)

Business operations use cases

Use the Business Operations Environment (BOE) to test the configuration, request changes, review and approve revisions, and deploy the revisions to the Production environment. Learn more about common tasks that the business team performs in the BOE environment.

The following diagram represents the BOE workflow:

BOE workflow



Value Finder analysis

Environment	Portal	User
BOE	Customer Decision Hub	Next-best-action designer

1. The next-best-action designer runs monthly Value Finder simulations with the latest data that they sampled from the Production environment.
2. The next-best-action designer analyzes the results, looking for value opportunities.
3. The next-best-action designer identifies groups of under-served customers (without relevant actions) or customers without any actions.
4. The next-best-action designer raises findings with the team lead and next-best-action analyst in the governance forum.

Scenario Planner analysis

Environment	Portal	User
BOE	Customer Decision Hub	Next-best-action designer

1. The next-best-action designer reviews the insights from Value Finder and uses the Scenario Planner to investigate the opportunities to improve customer engagement.
2. The next-best-action designer runs a Scenario Planner simulation with the latest data that they sampled from the Production environment.
3. The next-best-action designer analyzes the results to see how close the revenue is to meeting the target. The Scenario Planner results confirm that revenue is not on target and targeting the group of underserved customers could help

meet the target.

4. The next-best-action designer gives feedback to business owners for consideration.

Idea, plan, and build

Environment	Portal	User
BOE	Pega 1:1 Operations Manager	Business owner, team lead, next-best-action specialist

1. The business owner produces and submits an idea to a team lead.
2. The team lead considers whether the idea is a business-as-usual (BAU) change that the business content team can process and requests the confirmation of requirements.
3. The business lead supplies the requirements and the team lead approves them.
4. The next-best-action specialist completes the build.
5. The team lead approves the build for deployment. A Pega 1:1 Operations Manager change request is pushed to Revision Manager.

Testing

Environment	Portal	User
BOE	Customer Decision Hub	Next-best-action designer
BOE	Pega 1:1 Operations Manager	Team lead (revision manager)

1. The next-best-action specialist tests the revision by running the following simulations:
 - A decision funnel simulation to ensure that pass rates of engagement policies are in the right area
 - A distribution simulation to check overall distribution of actions
 - An ethical bias simulation to look for bias in the decision framework
 - A value finder simulation to verify whether value is now targeted correctly
2. The team lead runs a revision performance test to identify any overhead.
3. The team lead approves the revision.
4. The team lead initiates the deployment of the revision.
5. The revision is merged successfully.

Deployment

Environment	Portal	User
Orchestrator	Deployment Manager	Deployment manager
BOE	Pega 1:1 Operations Manager	Team lead (revision manager)

1. The deployment manager monitors the progress of the revision through the pipeline.
2. The revision awaits deployment to the Production environment.
3. The deployment manager initiates the deployment of the revision to Production.
4. The revision is deployed to the Production environment successfully.
5. After the successful deployment of the revision to Production, the team lead marks the revision as completed.

Verification

Environment	Portal	User
BOE	Customer Decision Hub	Next-best-action designer

1. The next-best-action designer runs a Value Finder simulation with the latest data that they sampled from the Production environment.
2. The business user analyzes the results to ensure that the value opportunity that the revision was to address no longer appears.
3. If the business user identifies any new gaps in customer engagement, the change request and revision process repeats.

Prerequisites for Pega Customer Decision Hub

Before you start your Pega Customer Decision Hub implementation, complete the following tasks.

Deployment on Pega Cloud

1. Optional:

Complete the "Discover" phase of the Inbound for Digital Channels Microjourney.

The "Discover" outlines the initial tasks required to successfully implement Pega Customer Decision Hub.

For more information, see "Discover" in the [MLP Backlog: Cross/Upsell – Digital](#) book.

2. Verify access to the cloud environment.

For more information, see [Pega Cloud Services](#).

Deployment on premises and third-party cloud

1. Optional:

Complete the "Discover" phase of the Inbound for Digital Channels Microjourney.

The "Discover" outlines the initial tasks required to successfully implement Pega Customer Decision Hub.

For more information, see "Discover" in the [MLP Backlog: Cross/Upsell – Digital](#) book.

2. Set up your environments:

- Development (SOR)
- Staging
- Business Operations Environment (BOE)
- Production

For more information, see [Deployment architecture for Pega Customer Decision Hub](#).

3. Install Pega Customer Decision Hub.

For more information, see [Pega Customer Decision Hub Installation Guide](#).

- [Implementing Pega Customer Decision Hub](#)
- [Deployment architecture for Pega Customer Decision Hub](#)

Managing changes in Pega Customer Decision Hub environments

Pega Customer Decision Hub is a Pega application intended to be configured and extended in a development environment, and then migrated to higher environments. At the same time, changes to business artifacts are made in the Business Operations Environment. These changes follow a separate change management process that allows clients to support two works streams.

This document provides detailed guidance for delivery teams looking to enable this with Customer Decision Hub using Pega 1:1 Operations Manager. The document also describes DevOps best practices that ensure your application can be maintained in the future.

Initial deployment and migrating entire applications to higher instances

Pega Customer Decision Hub implementation requires a deployment architecture with four environments and the change pipelines for deploying changes to the environments.

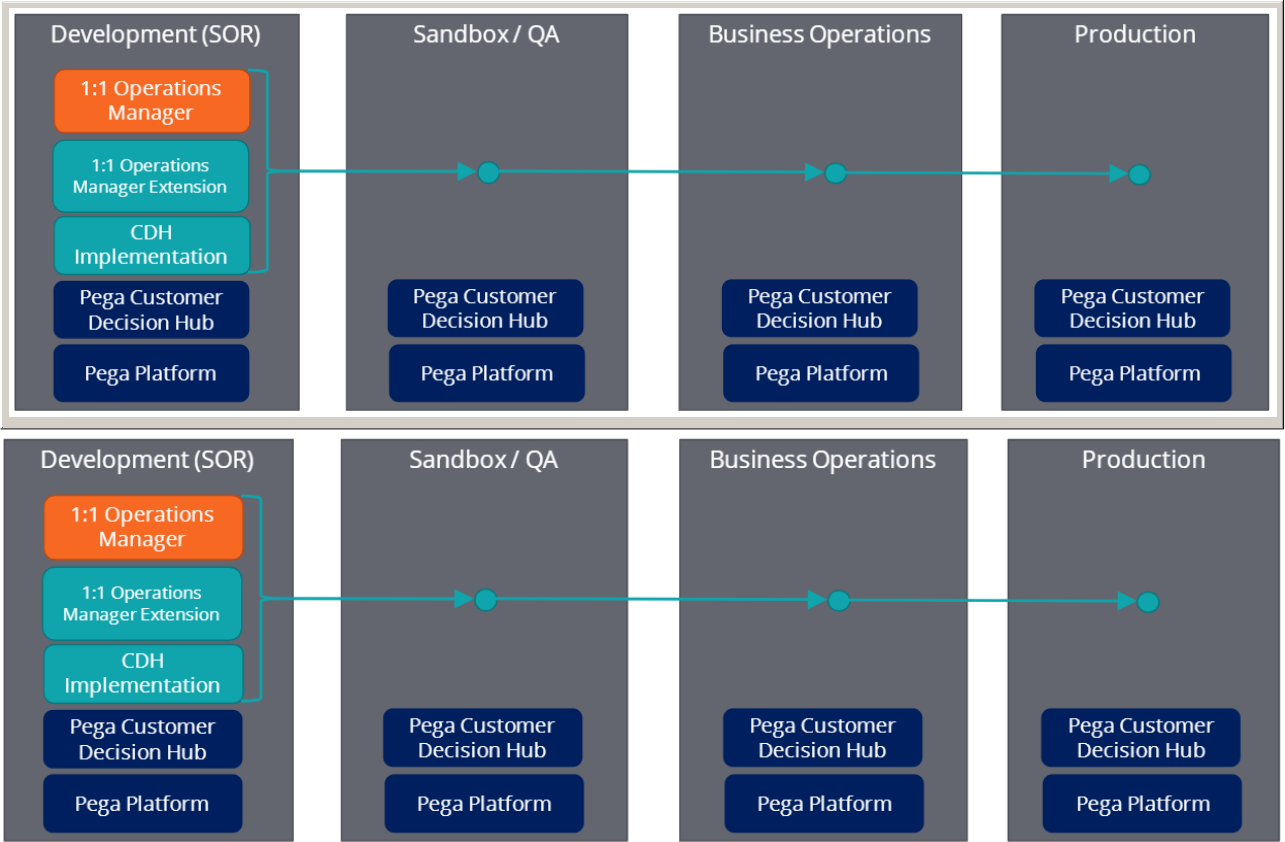
Configure the Development (system of records) environment to include the following applications:

Application name	Built on application	Users	Notes
1:1 Operations Manager overlay application	• Customer Decision Hub • 1:1 Operations Manager extension	NBA Specialist, Team Leader, Business User, Customer Decision Hub Manager, Revision Manager	This is the end-user runtime application. Enhancements to 1:1 Operations Manager cannot be made in this application.
1:1 Operations Manager extension	• 1:1 Operations Manager • Pega Platform (DevOps)	Developer, QA Tester	Enhancements to 1:1 Operations Manager are done here. These include changes to UI, cases, stages, flow and assignment routing.
		Customer Decision Hub Admin, Customer Decision	Enhancements to core Customer Decision Hub are done here. These include

Customer Decision Hub Application name	- Customer Decision Hub - Built on application - Pega Platform (DevOps)	Hub Manager, Customer Decision Hub Analyst	Context Dictionary Notes configuration, integrations, custom attributes, and others.

As a best practice, configure all three applications in the Development environment. Then, once all applications are validated and smoke tested, migrate them using the initial deployment pipeline.

Before initial deployment: Customer Decision Hub is installed on the standard environments

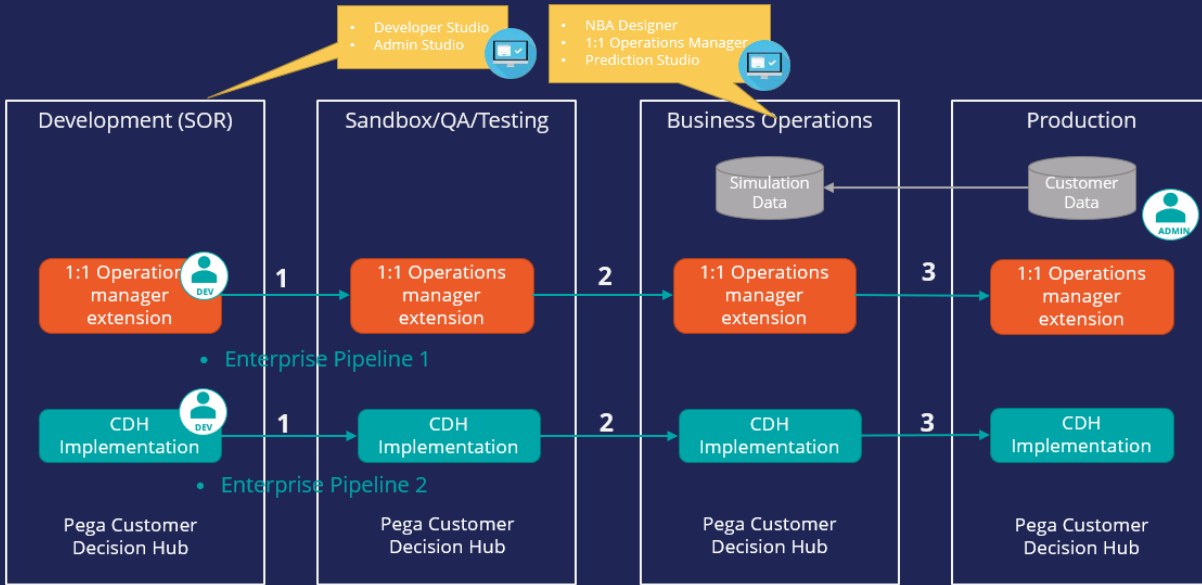


Enterprise changes

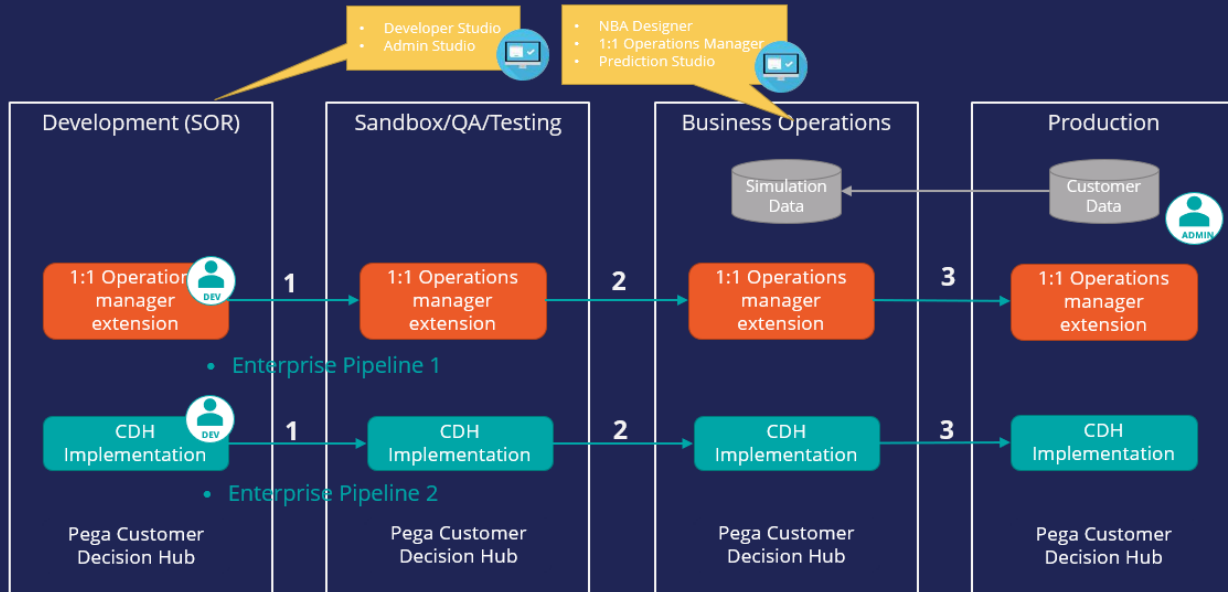
Enhancements to application functionality are made through standard developer practices like branching and ruleset lock and roll. Enhancements can be made for both, the Customer Decision Hub implementation application, and the 1:1 Operations Manager extension application. These enhancements are typically handled in the Rules ruleset, or in client-specific rulesets. For the 1:1 Operations Manager extension application, enhancements are handled in client-specific rulesets. No changes to the overlay application should be migrated from development after the initial deployment. These changes are migrated to higher environments using the Enterprise pipeline.

Enterprise changes

Ongoing deployment of applications



Ongoing deployment of applications



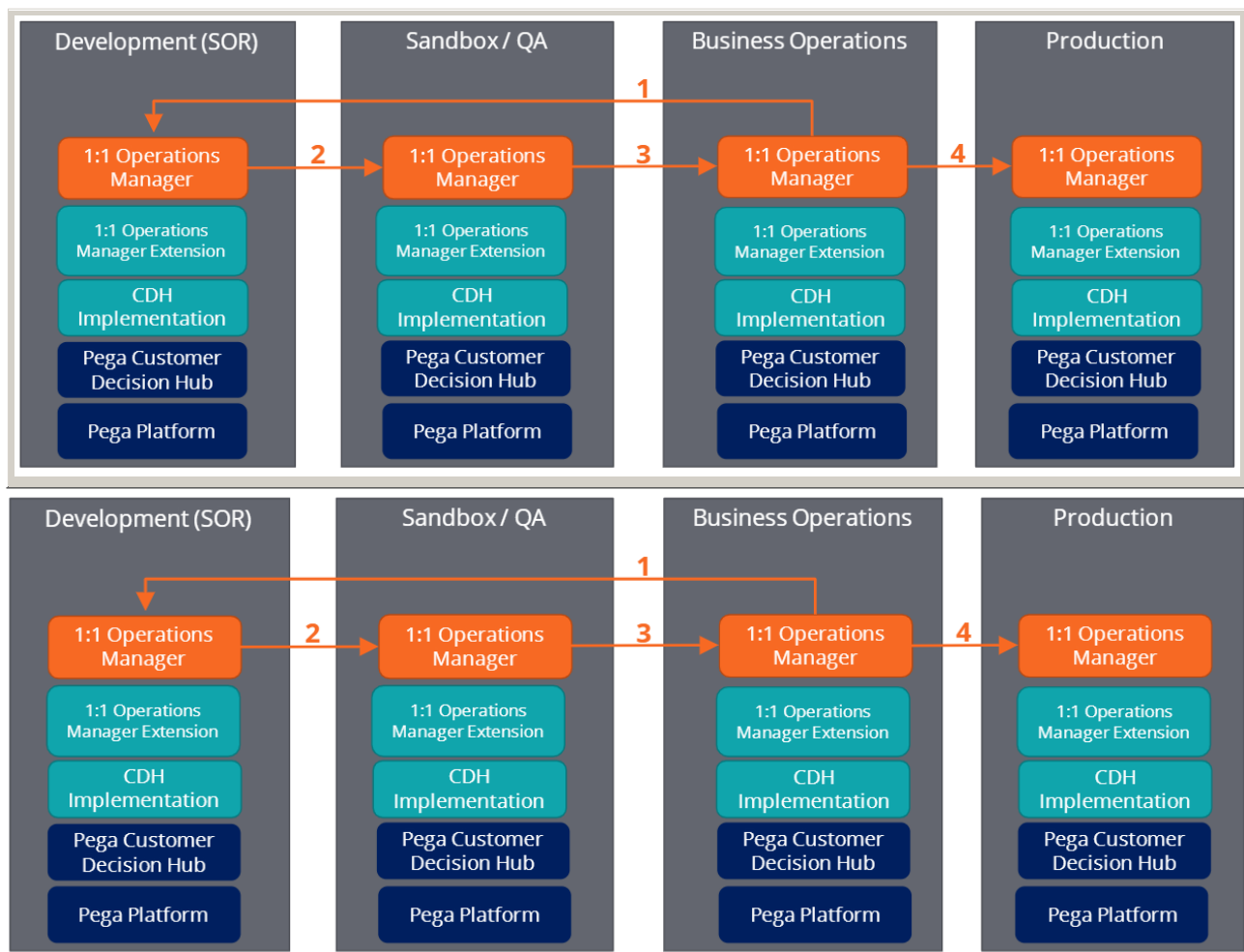
Business changes

Ongoing business changes include changes to next best actions, treatments, models, Next-Best-Action Designer configurations, contact policy, and similar artifacts. These changes are completed in the Business Operations Environment and should be quickly tested in lower environments.

The changes are handled through the Revision functionality, and are packaged automatically for you when using 1:1 Operations Manager.

Once users start using 1:1 Operations Manager, you need to include Next-Best-Action Designer under change management as well. The artifacts should be first deployed to Development, then higher environments to Production. Testing for these changes would be done in each environment by running specific tests that are appropriate in that environment. These changes are migrated using the Business change pipeline.

Business changes



Orchestrator

Your system also requires an orchestrator to manage the migration of changes through the change pipelines. In Pega Cloud services deployments, the default orchestrator is Deployment Manager.

Deployment pipelines

Customer Decision Hub supports the following primary types of pipelines from a DevOps perspective:

Enterprise changes

Enterprise changes involve changing the application itself, for example, by modifying the user interface. Enterprise changes are done by developers in a Development system of record (SOR) environment, and then pushed to other environments in the pipeline.

Business changes

Business changes involve business-as-usual operations such as the creation of new actions, treatments, customer segments, and so on. This type of change is triggered by business users in a BOE environment, and then pushed to other environments by using the Business Change pipeline.

Data migration

The automated data migration pipeline provides you with the means to test how the changes that you make to decision strategies affect their results, without running these strategies in a live production environment. You can sample a selection of real customer and internal Pega Platform data, and then migrate the sample to the BOE environment. As a result, the simulations are based on the migrated production data, that is, customer details, adaptive model data, and interaction history, which helps you assess how your changes are reflected in real-life scenarios. For more information, see [Data migration pipeline overview](#).

Pipelines are configured using [Pega Deployment Manager](#), or can be replicated using your own DevOps solution.

For more information about the recommended pipeline setup, see [Pipelines](#).

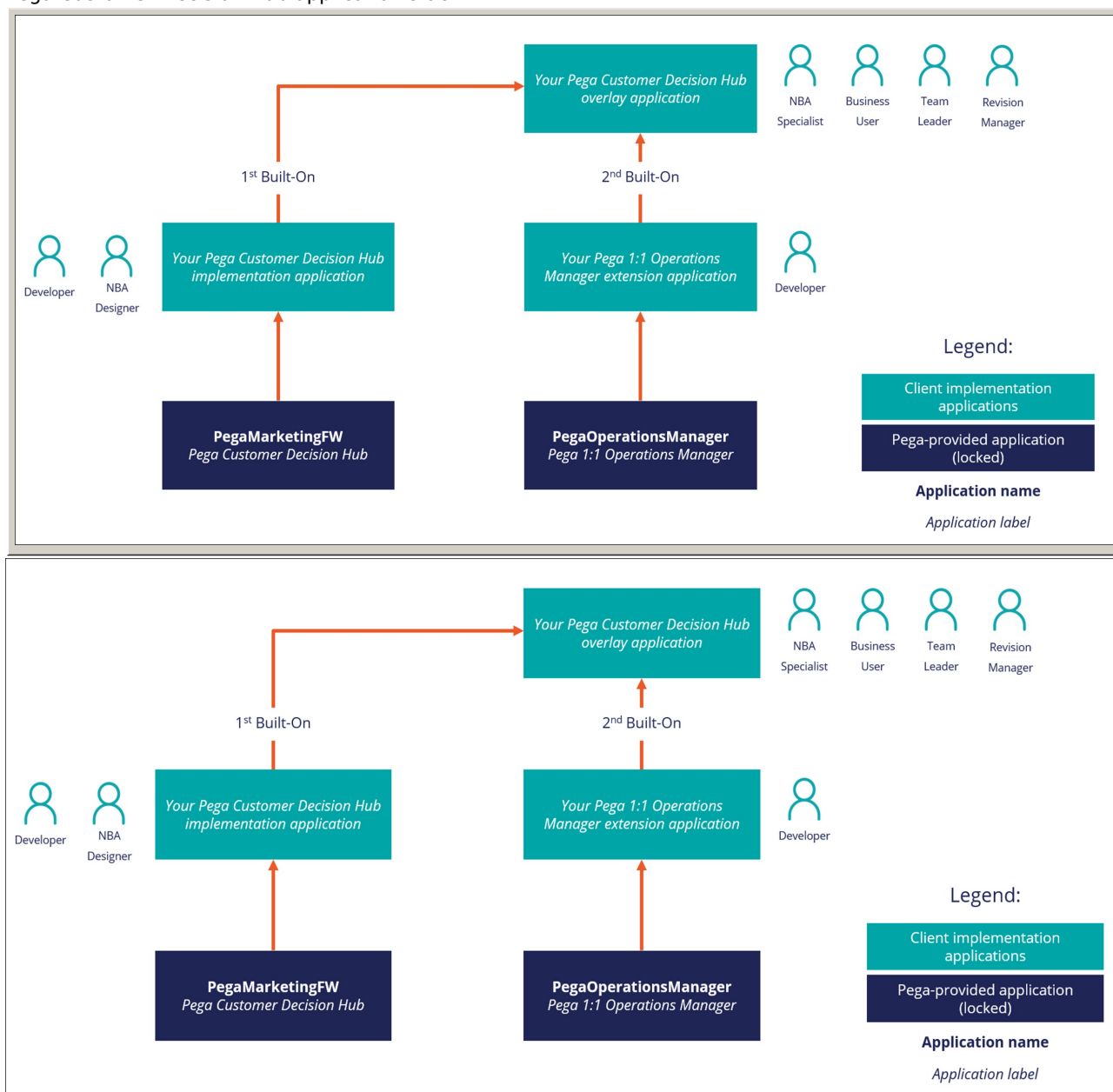
- [Configuring the Development environment](#)
- [Configuring higher environments](#)

Configuring the Development environment

Start your implementation by deploying Pega Customer Decision Hub and Pega 1:1 Operations Manager in the Development environment.

The following diagram represents the application layers that Pega Customer Decision Hub application depends on. The upper layers inherit features from a lower layer.

Pega Customer Decision Hub application stack



Your Pega Customer Decision Hub overlay application

The overlay application contains all the decisioning artifacts that a business operations team can change. This includes artifacts that are part of Pega Customer Decision Hub implementation layer and that are selected in Pega Customer Decision Hub overlay creation process.

Your Pega Customer Decision Hub implementation application

This is your extended Pega Customer Decision Hub application. This layer includes extensions and customizations of the out-of-the-box features of Pega Customer Decision Hub, such as the components that extend the implementation application for the communications and financial industries.

Your Pega 1:1 Operations Manager extension application

This application contains all the extensions and overrides that you made to the base Pega Customer Decision Hub application.

Pega Customer Decision Hub

This layer contains the out-of-the-box features of Pega Customer Decision Hub.

Pega 1:1 Operations Manager

This layer contains the out-of-the-box features of Pega 1:1 Operations Manager.

To configure the Development environment, perform the following procedures:

- [Creating the Pega Customer Decision Hub implementation layer application](#)

- [Managing data](#)
- [Implementing Pega 1:1 Operations Manager](#)
- [Configuring engagement channels](#)
- [Locking the release](#)
- [Implementing Pega Customer Decision Hub](#)
- [Deployment architecture for Pega Customer Decision Hub](#)

Creating the Pega Customer Decision Hub implementation layer application

Extend and customize the out-of-the-box features of Pega Customer Decision Hub by creating the implementation layer application.

For example, the [implementation layer application](#) contains the customer class for storing your customer data, a hierarchy of groups and issues and engagement policies for determining next best actions, and rules for loading customer data into Pega Customer Decision Hub from external systems.

Getting started:

Complete the initial setup for your [deployment type](#):

Pega Cloud services deployment

Create an administrator operator and run the First-time Setup Process for your application.

For instructions, see .

On-premises or client-managed cloud deployment

Create your application by running the New Application wizard.

For instructions, see [Creating your Pega Customer Decision Hub application for on-premises and third-party cloud deployment types](#).

What to do next:

When you complete the initial setup, perform the following procedures:

1. [Verifying installed rulesets](#)
2. [Optional: Implementing your Pega Customer Decision Hub application in a CRM configuration](#)
3. [Creating operator accounts](#)
4. [Updating the Manager field for your organization units](#)
5. [Optional: Defining the security model and organization structure](#)
6. [Configuring access groups and roles](#)
7. [Configuring Microsite settings](#)
8. [Optional: Granting permission to configure ethical bias policies](#)
9. [Optional: Enabling Pega Customer Decision Hub Marketing Campaign Edition](#)
10. [Verifying the System Health page](#)

Creating your Pega Customer Decision Hub application for on-premises and third-party cloud deployment types

If your application is not deployed on a Pega Cloud Services environment, you must create it by using the New Application wizard.

Before you begin: Access the Organization landing page and create a customer organizational structure. For more information, see [Adding an operator by using the organizational chart](#).

When you run the New Application wizard for Pega Customer Decision Hub, it creates a set of Pega-provided access groups. You must then create your own operators and add them to the access groups that you create. **Caution:** If a Pega Customer Decision Hub application already exists in your system, running the New Application wizard overwrites some of the existing application settings. This will cause issues with your existing application. Having more than one active Pega Customer Decision Hub application on the same Pega Platform instance is not supported. This includes custom applications built by you, and sample applications provided by Pega.

1. Log in to Dev Studio using the operator ID, for example, *administrator@pega.com*, and the password that you specified for that operator.
2. In the header of Dev Studio, click **Application New Application** .
3. Follow the instructions in the New Application wizard.
For more information, see [Creating applications](#).
4. Click **Go to app** .

Result:

The New Application wizard creates the application class structure for you. You can reuse the classes and rule sets that the wizard creates in your applications. When you implement your applications, you can then apply class hierarchy and inheritance principles to help you improve the maintainability and overall efficiency of your application.

For more information, see [Class layers](#) and [Class hierarchy and inheritance](#).

What to do next: To fully set up your application, in the top-right corner of App Studio, click **Help Application Guides**, and then follow the setup steps.

Verifying installed rulesets

The following lists the highest level ruleset versions available after you run the New Application wizard. Verify that your system includes these ruleset versions.

Note: If your system does not contain the following rulesets, the application was not created correctly.

1. Log in as an administrator created for your implementation application.
2. In Dev Studio, click **Configure Application Structure RuleSet Stack**.
3. Verify that the following ruleset versions are listed:
 - *your implementation application:01.01.01*
 - *your implementation application-Artifacts:01-01-01*
 - *your implementation application-NBADTests:01-01-01*
 - *your implementation application-Rules:01-01-01*
 - *your implementation application:01-01-01*
 - *your implementation applicationInt:01-01-01*

Optional: Implementing your Pega Customer Decision Hub application in a CRM configuration

Pega Customer Decision Hub is ready to be used with the [Pega Customer Relationship Management suite of applications](#) to share data and service cases across the Pega Customer Service, Pega Sales Automation, and Pega Customer Decision Hub applications.

Important: The Pega Customer Decision Hub application should be created on a different Pega environment with its own database and application servers separate from the ones used by Pega Customer Service and Pega Sales Automation applications. The Pega Customer Decision Hub application is a back-end analytical application with a typically smaller number of active users compared to the operational Pega Customer Service and Pega Sales Automation applications. Keeping the operational and analytical clusters separate helps ensure easier maintenance and decreases upgrade complexity.

Creating operator accounts

Create operator accounts for users of your implementation application, and then assign them to the predefined user groups.

The New Application wizard for Pega Customer Decision Hub creates the following default access groups:

your implementation application:CDHAdmins

This is an access group for administrators. The default portal for operators in this access group is the App Studio workspace in administrative mode for the Pega Customer Decision Hub portal. This access group also grants access to Admin Studio, Prediction Studio, and the Pega Customer Decision Hub portal.

your implementation application:CDHAnalysts

analysts and other marketing users. The default portal for operators in this access group is the Pega Customer Decision Hub portal. This access group also grants access to Prediction Studio.

your implementation application:CDHManagers

This is an access group for managers. The default portal for operators in this access group is the Pega Customer Decision Hub portal. This access group also grants access to Prediction Studio.

1. Log in as a Pega Platform administrator.
2. Create an operator account using one of the above access groups:
For more information, see [Creating operator IDs](#).
3. Complete the **Profile** tab.
For more information, see [Fields for operator contact information and application access](#).
4. On the **Security** tab, in the **Access settings** section, configure access settings for your operator ID:
 - a. Select **Allow rule checkout**.
 - b. Click **Update password**, enter the password for this operator, and then click **Submit**.
 - c. Click **Save**.
5. Log off and log back in as the operator that you created.
6. In the operator preferences, disable the **Auto checkout on save as** setting.
For more information, see [Setting your user preferences](#).

Tip: After creating the first operator record, open it in Dev Studio, and then click **Save Save as**. This operation creates a new operator record with all the settings and preferences that you previously entered.

What to do next: After you create operator accounts for users of your implementation application, you can invite them to

log in at any time. To invite users to your application:

1. Log in to the Pega Customer Decision Hub portal as a user in the *your implementation application*:CDHAdmins access group, and then click **Dev Studio App Studio** .
 2. In App Studio, click **Users People Invite people to your application** .
- [Template access groups and roles](#)

Template access groups and roles

Template access groups are provided with the Pega Customer Decision Hub and are used to generate new access groups when the user runs the new application wizard.

Template access groups

Access group name	Application where it is used
<i>CDHTemplate:OpsDeveloper</i>	Pega 1:1 Operations Manager
<i>CDHTemplate:NBASpecialist</i>	Pega 1:1 Operations Manager
<i>CDHTemplate:OpsMgrAdmin</i>	Pega 1:1 Operations Manager
<i>CDHTemplate:TeamLead</i>	Pega 1:1 Operations Manager
<i>CDHTemplate:BusinessUser</i>	Pega 1:1 Operations Manager
<i>CDHTemplate:CDHAdmins</i>	Pega Customer Decision Hub
<i>CDHTemplate:CDHManagers</i>	Pega Customer Decision Hub
<i>CDHTemplate:CDHAnalysts</i>	Pega Customer Decision Hub
<i>CDHTemplate:CDHInstall</i>	Pega Customer Decision Hub

CDHTemplate or *MKTTemplate* in the name of the access group will be automatically replaced with the name of your application. For example, if you name your application *DevCo* during the new application wizard, the access groups will be named *DevCo:Admins*, *DevCo:CDHAnalysts* and so on.

Associating access groups with roles

The access groups used by these operators in the application are associated with roles. Each of these roles should be built with the associated *PegaNBAM_FW* role as a dependent role. This ensures that any new privilege granted in Pega Customer Decision Hub will be granted to the appropriate operators automatically when upgrading your environment.

To verify that you access groups are associated with the correct role:

1. Open your access group to see its list of available roles.
2. Open the role.
3. Click **Manage dependent role** and verify that you see one of the *PegaNBAM_FW* roles as a dependent role.
4. If required, add the *PegaNBAM_FW* role to get the latest list of privileges.

Note: Some access groups are associated with multiple roles. The order of these roles defines the additional permissions or privileges for the access group. The **Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access** option is disabled for these groups, so that the permissions for these roles can accumulate.

To remove permissions, use one of the following rules:

- To remove all permissions for a given rule, use an *Access Deny* rule. For more information, see [Understanding Access Deny rules](#).
- For a more granular permission edit, use an *Access of Rule to Object* rule. For more information, see [Understanding Access of Rule to Object rules](#).

Updating the Manager field for your organization units

The New Application wizard creates the organization structure for your application. For each organization unit the wizard creates, specify the manager operator.

1. Log in as a Pega Platform administrator.
2. In the navigation pane, click **Records Organization Organization Unit** . **Result:** A list of all existing organization units is displayed.
3. Click on a unit to select the manager operator.
4. In the **Manager** field in the **Staff** section, enter or select one of the operators with the *your implementation application*:CDHManagers access group.

5. Click **Save**.
6. Repeat the procedure for every unit in your organization.

Optional: Defining the security model and organization structure

Define the authorization and authentication strategies for your application.

Authentication

Proves to the application that you are who you say you are.

Authorization

Determines the functions that you can perform in the application. This corresponds to an access group and role configuration.

Security planning involves defining authorization and authentication strategies for your application. It is a best practice to create new access groups and roles that are based on the default access groups and roles that come with the product.

Security planning also involves setting up the organization structure and operator attributes. The application provides security in the form of access settings and denial rules. Many integration rules also incorporate authentication.

For more information about the additional aspects of security, enroll in the *Lead System Architect* course on Pega Academy.

1. [Authentication schemes](#)
2. [Defining your authentication scheme](#)
3. [Defining your access groups](#)
4. [Defining access roles and privileges](#)
5. [Defining the work groups](#)
6. [Defining the workbaskets](#)

Authentication schemes

The Pega Platform offers the following authentication types:

PRBasic

Based on passwords in the Operator ID data instances and the login form. This is defined by the HTML `@baseclass.Web-Login` rule, which your application can override.

PRSecuredBasic

Similar to PRBasic, but passes credentials by using Secure Sockets Layer (SSL) with Basic HTTP authentication. The login form is defined by the HTML `@baseclass.Web-Login-SecuredBasic` rule, which your application can override.

PRCustom

Supports access to an external LDAP directory or a custom authentication scheme.

PRExtAssign

Supports external assignments (Directed Web Access).

J2EEContext

Specifies that the application server in which the Pega Platform is deployed uses JAAS to authenticate users.

Defining your authentication scheme

Your site can use a centralized, automated means of maintaining operator data instead of maintaining it manually in your application.

1. Discuss the authentication schemes with your site's security and application server teams.
2. Determine the appropriate authentication type.

For more information on authentication scheme planning, see [Authentication](#).

Defining your access groups

Define the access groups that you want to add to your application.

1. Identify additional access groups that are needed for your application.
2. Identify portals associated with these access groups.

For more information, see [Viewing access groups and operators](#).

Defining access roles and privileges

You can associate one or more roles with an access group. Roles are additive. The more roles that you add to an access group, the more authorization there is. Privileges can be associated with one or more roles.

1. Determine which roles are needed for your application. You can use the Pega Customer Decision Hub roles as a starting point.
2. Determine which privileges to associate with each role.
3. Associate each role with an access group.

For more configuration information, see [Groups and roles](#).

Defining the work groups

A work group determines which work queues you can access.

1. In the header of Dev Studio, click **Configure Org & Security Tools Work Groups**.
2. Review and modify the list if necessary.

For more information, see [Teams](#).

Defining the workbaskets

A work queue is a queue of open assignments in the application.

1. In the header of Dev Studio, click **Configure Org & Security Tools Work Queues** to display a list of work queues.
2. Review and modify the list if necessary.

For more information, see [Initial Work queues](#).

Configuring access groups and roles

It is a best practice to create new access groups and roles that are based on the default access groups and roles that come with the product. This insulates your application from changes if the roles and groups change permissions in a future release. Refer to the roles and groups that you identified during the procedures.

To update an access group, complete the following steps:

1. Click **Dev Studio Configure Application Structure Access Groups and Users**.
2. Select the access group that you want to copy.
3. Click **Save as** to save the role to your application.
It is recommended that you name the role with a different name. Update the roles to use the ones that you created for your application.
4. Click **Create and open**.
5. In the **Definition** tab, map the **Name** and **Version** field details with your application and click **Save**.
6. **Optional:** To update an access role, complete the following steps:
 - a. Click **Dev Studio Configure Application Structure Access Groups and Users**.
7. Select the access role that you want to copy.
8. Click **Save as** to save the role to your application.
Pega recommends that you give the role a different name.
9. Click **Create and open**.
10. Click **Save**.

Configuring Microsite settings

If you have installed the MS.war file on a different machine than the one on which you have deployed Pega Customer Decision Hub, you need to modify the `index.html` file.

Note: Only operators who have the role PegaMarketing_FW:MicrositeAdmin can create Microsites. By default, analysts do not have this role. If an operator requires the ability to create Microsites, associate the operator with an access group that contains the PegaMarketing_FW:MicrositeAdmin role (see Pega Platform help or go to Pega Community for instructions on configuring access groups).

1. Navigate to the directory where the MS.war file has been deployed. **For example:** Tomcat\webapps\MS
2. Edit the `index.html` file and prepend the full URL to your system:
 - `http://hostname:port/prweb?pyActivity=pzIncludeMashupScripts`
 - `http://hostname:port/prweb/PRRestService/PegaMKTMicrosite/Services/IncludeMicrositeScripts`
3. In Dev Studio, open the application rule associated with the PegaNBAM:Agents access group.
4. In the application rule, go to the **Integration & Security** tab.
5. In the **Mashup security** section, add a new **Trusted Origin** and add the URL for the location of the deployed MS.WAR file.
6. If you previously configured a content security policy in the **Content security** section, edit the policy and allow access from the website where the MS.war file has been deployed.
You must update the **Frame-Ancestors** section of the policy on the **Policy Definition** tab, and potentially other sections, if required by the Microsite.
7. Click **Configure System General Systems, Nodes, Requestor**.
8. In the **Requestors** table, click the BROWSER requestor for your current system.
Note: Your current system name is displayed in the **System Name** field above the table.
9. Click **Save**.
10. Create a new operator account. In the **Access Group** section on the **Profile** tab, enter or select <your application>:Microsites.
11. Create the following Dynamic System Settings to override the default configuration settings in the Microsite gadget:
The owning ruleset for all of the settings is *PegaMKT-Engine*.
 - **microsite/applicationname** - This is the application name which the Microsite gadget uses when connecting to

Pega Customer Decision Hub. By default, it is the name of the application associated with the access group on the PegaMKTMicrosite service package.

- **microsite/URL** - This is the URL which the Microsite gadget uses when connecting to Pega Customer Decision Hub. The URL should point to an IAC servlet on the public node. The URL format is `http://<hostname>:<port>/prweb/IAC`.
- **microsite/deployedURL** - This is the URL where the MS.War file was deployed. The URL format is `http://<hostname>:<port>/MS`.
- **microsite/systemid** - This is the system ID which the Microsite gadget uses when connecting to Pega Customer Decision Hub. By default, it is the system ID defined on the system which the URLs point to.
- **microsite/defaultclass** - If the Site parameter of the Microsite URL is not defined, this setting purpose is used instead.
- **microsite/parameters/useridentifier** - This is the user which the Microsite gadget uses to authenticate itself with Pega Customer Decision Hub. Enter the name of the operator you created in step 11 into the **Value** field.
- **microsite/parameters/skinname** - This is the name of the skin used by the Microsite. **Note:** If you are using your own skin to update the included styles tab of the skin rule, select the checkbox **Include legacy CSS**. Without the legacy CSS, your skin will not be displayed on multi-node environments.
- **microsite/parameters/additional** - This is a comma-separated list of additional parameters that can be passed to the gadget.

Optional: Granting permission to configure ethical bias policies

Configure access privileges to grant selected operators the right to define ethical bias policies.

Ethical bias policies provide a way of testing your strategies for unwanted bias, as well as checking how extensive the bias is. For example, you can test whether your strategies generate biased results by sending more actions to male rather than female customers.

To edit the ethical bias policies, enable the *decision/simulation/biasCheck* dynamic system setting, and then grant the *pzBiasPolicyConfiguration* to an access group.

1. Log in to Pega Customer Decision Hub as an operator with access to Dev Studio.
2. In the navigation pane of Dev Studio, click **Records SysAdmin Dynamic System Settings**.
3. In the list of dynamic system settings, click the *decision/simulation/biasCheck* setting.
4. Set the value of the *decision/simulation/biasCheck* setting to **true**.
5. Click **Save**.
6. In the navigation pane of Dev Studio, click **Records Security Access Groups**.
7. Edit the access group that you want to have the right to configure ethical bias policies for your organization.
Tip: This type of access is usually granted to administrators. If required by company policies, you can create a new access group specifically for the purpose. For more information, see [Creating an access group](#).
8. In the **Available roles** section, click **Add role**.
9. Enter a name for the new role, for example, *your implementation application:MarketingBias*, and then click the **Open** icon.
10. On the **Create Access Role Name** page, click **Create and open**.
11. Select **Inherit privileges within class hierarchy**, and then click **Save**.
12. On the **Role** tab, in the **Access class** table, click **Add a row**.
13. In the **Add access Role Object** window, in the **Class** field, enter `@baseclass`.
14. In the **Access Controls** section, set the value of all controls to 5.
15. In the **Privileges** section, click **Add item**, and then enter the following settings:
 - a. In the **Name** field, enter `pzBiasPolicyConfiguration`.
 - b. In the **Level** field, enter 5.
16. Click **Save**.
17. On the access group tab, click **Save**.

Result: Operators associated with the access group can now configure ethical bias policies in the Pega Customer Decision Hub portal. For more information, see "Ethical Bias Policies" in the *Pega Customer Decision Hub User Guide* on the [Pega Customer Decision Hub product page](#).

Optional: Enabling Pega Customer Decision Hub Marketing Campaign Edition

With Pega Customer Decision Hub Marketing Campaign Edition, you can define segments and schedule traditional batch campaigns, delivering messages across direct mail, email, and SMSs.

Pega's recommended approach for outbound customer engagement is to use the [outbound schedule](#) option available within the Next-Best-Action Designer, referred to as the *next-best-action approach*. This approach allows you to use the same customer engagement policies consistently across all inbound and outbound communication channels. Further, this approach allows you to use Pega 1:1 Operations Manager, which makes consistent management of actions and engagement policies even easier. The filtering is done at the group and action level with very few segments used only to limit the starting population for the schedule. This results in very few or even just one outbound schedule (the primary schedule).

The next-best-action approach is different compared to the product-centric, segment-based, batch and blast approach or the campaign approach, for which you need the additionally licensed Marketing Campaign Edition component. In the campaign approach you end up creating tens or hundreds of individual campaigns, each with their own filtering criteria and schedule.

To start using the Marketing Campaign Edition, you need to enable the **Campaigns** menu in your implementation application.

1. Log in to Dev Studio with an administrator account.
2. In the navigation pane of Dev Studio, click **Records SysAdmin Application Settings**.
3. On the **Application Settings** tab, search for the *EnableCampaignEdition* setting in the *PegaMKT-Engine:<version>* ruleset.
4. Click the *EnableCampaignEdition* row.
5. Save the *EnableCampaignEdition* setting to the implementation ruleset:
 - a. On the **Application Settings: Campaign edition licensed customers only** tab, click **Save as**.
 - b. On the **New** tab, in the **Add to ruleset** field, enter the name of the implementation ruleset.
 - c. Click **Create and open**.
6. On the **Edit Application Settings: Campaign edition licensed customers only** tab, set all the production level values to **True** by selecting the check boxes.
7. Check in your changes:
 - a. Click **Check in**.
 - b. In the **Check-in comments** field, describe the changes. **For example:** Enabled marketing campaign edition
 - c. Click **Check in**.

Verifying the System Health page

After building your implementation features, check the System Health page for any errors or warnings.

1. Log in to the Pega Customer Decision Hub portal.
2. Click **Configuration Settings System Health**.
3. Review and address any errors or warnings.

Managing data

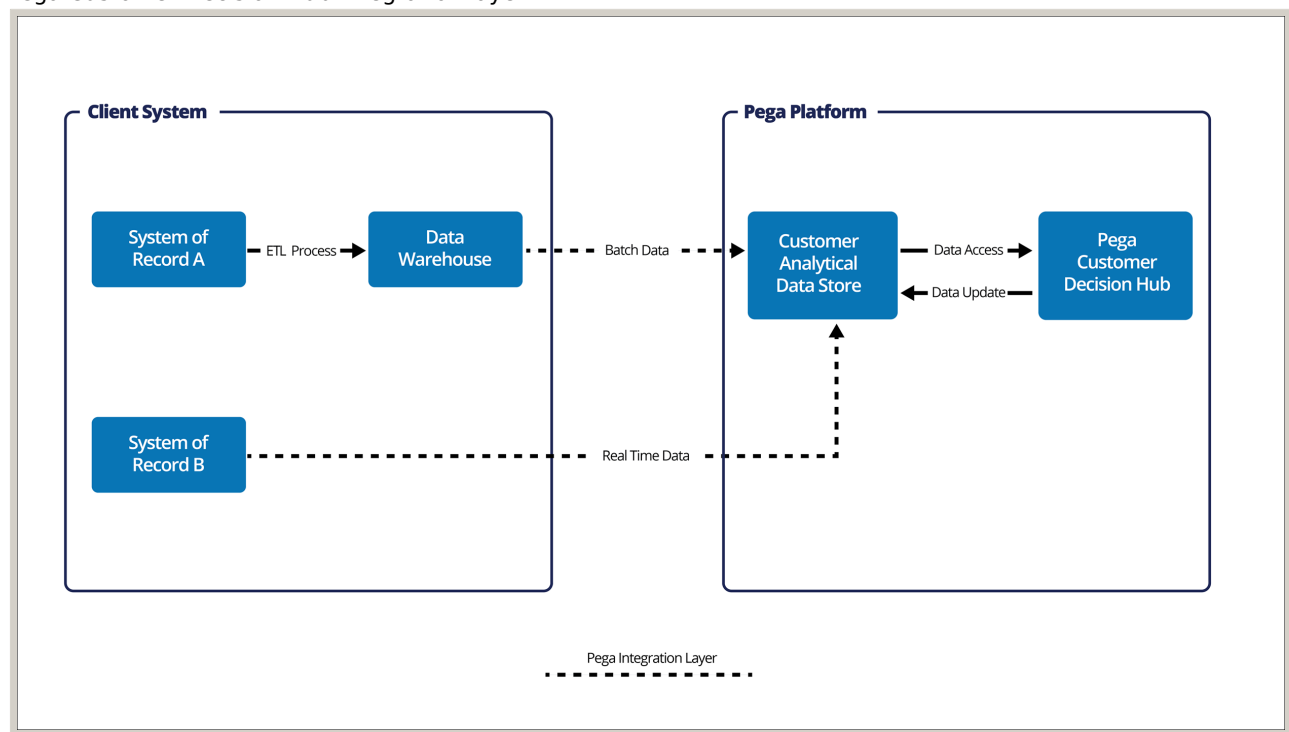
Create and manage your customer data model. Make sure that you identify the customer data that correlates to your business use case and that is aligned with the problem that you want to solve.

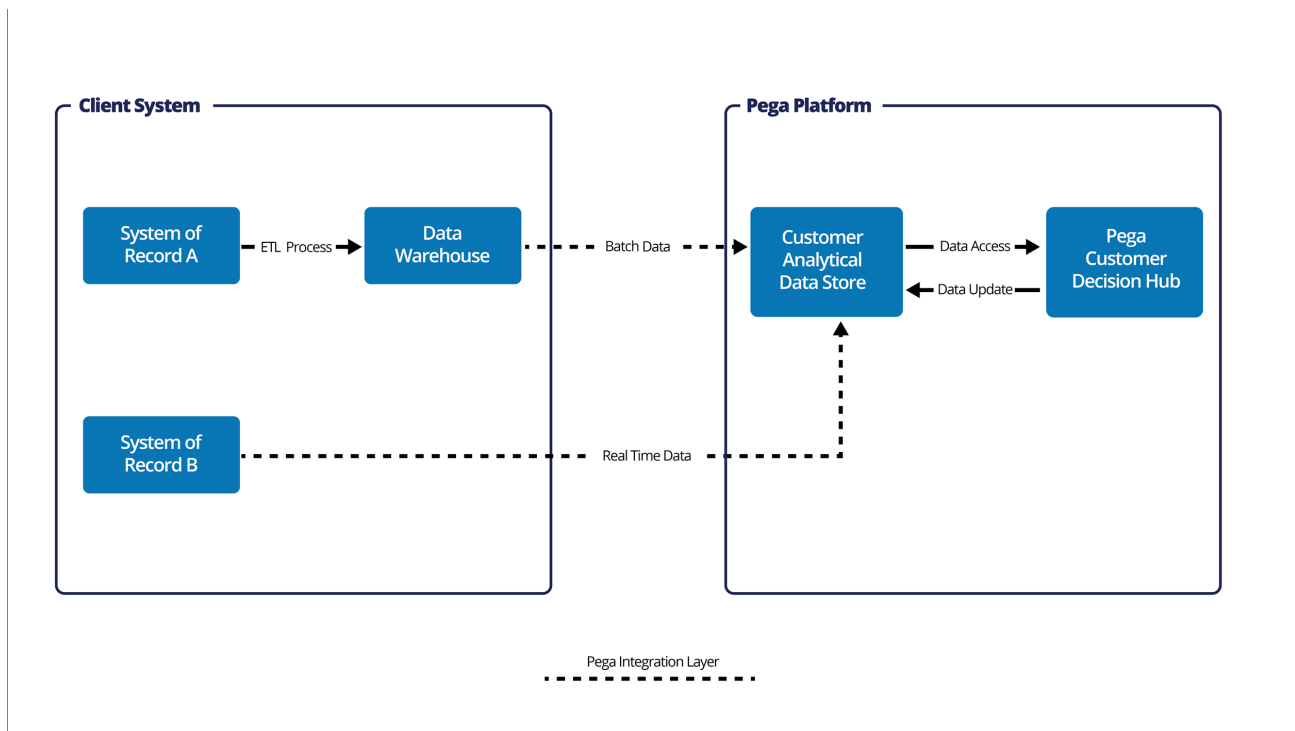
Data Model

The Pega Customer Decision Hub application integrates your customer data to provide next best actions, execute marketing campaigns, and learn customer behavior using predictive and adaptive analytics.

The following diagram represents a typical Pega Customer Decision Hub implementation:

Pega Customer Decision Hub integration layer





The data in your systems of record is usually extracted to a data warehouse or data lake for internal purposes. Typically, only a portion of this data is required by the Pega Customer Decision Hub application. Pega supports accessing data through database ETL processes, batch file imports, or real-time services such as Kafka, Kinesis, or REST-based streams. Your project objectives and data latency requirements determine your data access implementation.

After the source data is identified, data is loaded to the out-of-the-box customer analytical data store. This process restructures the customer data as flattened data objects that maximize performance and scalability while ensuring data integrity. Designing the customer analytical data store is an important part of project initiation and should be done in conjunction with your business and implementation partners.

The customer analytical data store is also referred to as the Customer Analytics Record (CAR or xCAR). The xCAR is a flat, structured record of the customer data in a format that is optimized for performance and real-time analytics in one-to-one customer engagement. The result is a solution that provides more accurate and faster business decisions and that presents customers with messages and offers that are more attractive and better tailored to their needs.

Deployment options for storing customer data

When implementing Pega Customer Decision Hub, you can improve performance or reduce deployment effort by choosing either a Decision Data Store (DDS) implementation or a combined relational and DDS implementation for your customer analytical data store.

Relational implementation

In a relational-only databases solution, your customer analytic data store has the highest level of compatibility with Pega Platform functionalities. Relational implementation is the most tested and proven solution.

Note: This is the default recommended solution for organizations with 100 million or less customers and requires you to work with your database administrators to index and optimize your relational tables. Contact the Pega team to discuss best approach if you have more than 100 million customers.

Select this deployment option for simulation testing, as it relies on data segmentation.

For more information, see [Simulation testing](#).

Combined relational and DDS implementation

With the combined relational and DDS databases, you can reduce implementation effort and maximize reuse, particularly if you deploy applications that need to support audience-driven marketing campaigns, or multi-level decisioning.

In this solution, your customer analytic data store is contained in both SQL and NoSQL data stores and are replicated and maintained within Pega Platform for maximum flexibility.

Decision Data Store implementation

In a DDS-only solution, your customer analytic data store is contained in Pega Platform by using NoSQL technology, which allows for handling large volumes of data and, as a result, helps to avoid performance issues.

Select this deployment option when:

- You have performance requirements of near real-time responses or a database that exceeds over a hundred million of customer records.
- You want to deploy your application using only the inbound Pega Customer Decision Hub (a centralized component that handles business needs) or in analytics-driven decision processes. **Note:** Do not use the DDS implementation for outbound marketing.

The DDS does not support:

Segmentation rules

A Pega Customer Decision Hub feature with which you can create a static list of customers to use in traditional outbound campaigns.

Exclusion lists

Lists that specify entities that do not receive actions and offers.

Using accelerators to configure the customer class

Pega Customer Decision Hub requires a customer class to support the loading of customer data. For organizations in the telecommunications, financial services, healthcare, or insurance industries, the recommended way to implement a customer class is through the use of Pega-provided data model (xCAR) accelerators.

Accelerators help organizations go live with their applications in a shorter time and provide guidance on best-in-class Pega practices.

The following data model accelerators are available for Pega Customer Decision Hub implementations:

[Communications Customer Analytical Data Model](#)

The communications customer analytical data model in the form of an extended Customer Analytics Record (xCAR) is a multilevel, structured record of the customer data in a format that is optimized for performance, real-time analytics, and one-to-one marketing. The xCAR provides a basis for faster, more accurate business decisions and for more attractive offers that are better tailored to the customers' needs.

[Financial Services Customer Analytical Data Model](#)

This component holds the data model for financial services in the form of an extended Customer Analytics Record (xCAR) that includes data entities and attributes to support banking use cases.

[Healthcare Customer Analytical Data Model](#)

This component holds the healthcare customer analytical data model (xCAR) that includes data entities and attributes to support healthcare use cases.

[Insurance Customer Analytical Data Model](#)

This component holds the insurance customer analytical data model (xCAR) that includes data entities and attributes to support insurance use cases.

1. Go to [Pega Marketplace](#) and download the data model for your industry.
2. Install the component by doing the following steps:
 - a. In the header of Dev Studio, click **Configure Application Distribution Import**.
 - b. On the **Import Wizard** tab, click **Choose File**, and then select the file that contains your xCAR data model. For example, the healthcare xCAR data model file is `PegaCDHxCARforHC.zip`.
 - c. After the file has been successfully imported, in the header of Dev Studio, click the name of your implementation application, and then click **Definition**.
 - d. Click **Manage Components**.
 - e. In the pop-up window, select the xCAR data model component that you imported.
 - f. Select **Enable**, and then click **OK**.
 - g. On the application form for your implementation application, click **Save**.
 - h. In the header of Dev Studio, click **Configure Application Structure Ruleset Stack**.
 - i. Verify that the ruleset corresponding to the imported file is above `PegaMarketing_FW:8` in your application stack.
3. Save the shipped customer class into your implementation ruleset by using the **Save as** option.
4. Open the customer class that you saved into your implementation ruleset.
5. In the **Class inheritance** section, select the **Find by name first (Pattern)** check box, and then as the parent class, enter the shipped customer class.
6. [Create the database table rule](#) that maps the class to the customer table.
Note: To help ensure future compatibility, save the shipped database table into your own implementation ruleset, and then map the class to the new database table.
7. For any associated data or additional contexts, save the page list properties that are used to link your data together into your implementation ruleset, and then update the page definition of your implementation class.
8. Define a new [association rule](#) to associate your classes.
Note: If you are adding associated data, you do not need to manually create the association rule. It will be done automatically for you when you add your associated data in Customer Profile Designer, as long as the property that you are associating on is the key of the class.
9. [Update the context dictionary](#) based on your new classes, page list properties, and [static](#) or [stream data](#) associations.
10. If you want to add additional properties, do the following steps:
 - a. Add the property to the new implementation class.

- b. Expose the relevant column in the customer table.
- c. Update the **External mappings** tab of your class definition.
- d. Add the property to relevant records and categories as needed.

For example:

The following video demonstrates how to use the xCAR data model component to configure the customer class while implementing your Pega Customer Decision Hub application:

Adding associated static data in the Customer Profile Designer

In addition to creating data aggregates, you can also use the Customer Profile Designer to associate static data, such as database tables, with customer contexts. For example, you can associate an investment portfolio table with the Account Owner context.

Caution: The relationship with associated entities can be one-to-one or one-to-many. Pega Customer Decision Hub does not support many-to-many relationships.

1. In Dev Studio, create a Data Set rule for the associated entity. For more information, see [Data Set rules - Completing the Create, Save As, or Specialization form](#).
Important: Ensure that the type of the data set is Database Table.
2. In the Pega Customer Decision Hub interaction portal, click **Data Profile Data Sources**.
3. Select the data set that you want to associate with a customer context by clicking **Add Data set**.
4. In the **Add data set** window, select the data set and click **Add**.
5. Open the Customer Profile Designer by clicking **Data Profile Designer**.
6. Click **Edit**.
7. Next to the context with which you want to associate the data set, click **Add associated data Data set** and then select the data set from the list.
Tip: A single data source can only be associated with a single customer context. You cannot associate the same data source with multiple contexts.
8. Click **Save. Result:** The data set is now associated with the customer context. You can review the context in the Context Dictionary.

What to do next: For a property on an associated entity to be available as criteria in a [segment](#), it must be added to the [External Mappings](#) of the appropriate context class.

Adding stream data aggregates in the Customer Profile Designer

You can associate customer records with aggregate stream data by using the Customer Profile Designer.

Starting in Pega Customer Decision Hub version 8.6, you can manage associated data in Customer Profile Designer. Customer Profile Designer allows users to define their customer profile data with aggregate properties and summaries, and enables Pega Customer Decision Hub to handle real-time aggregation based on specified aggregation logic and source data. These aggregates can then be associated with the extended customer analytical record (xCAR) data through the new Customer Profile Designer landing page, and used as predictors in adaptive models or as conditional properties in engagement policies.

Customer Profile Designer is available in Pega Customer Decision Hub 8.6 as a fully working early preview version. The functionality will be further expanded in future releases.

Creating the data class for Customer Profile Designer summaries

To add a new association for an entity (for example, *PurchaseData*) to a customer context, first create the class and service integration for the data source that you want to aggregate.

1. In Dev Studio, create a class for the source data that you want to use for the associated entity. For more information, see [Organizing rules into classes](#).
For the purposes of being the data class for your source data, your class should inherit from *Data-* and be defined as a concrete class. Your class key should be the identifier value of the source data and your class should be saved to your Pega Customer Decision Hub implementation ruleset. **Note:** If the property you select to be your class key is not available, you may need to create the property in the *Data-* class. **Tip:** To avoid problems when creating data summaries, ensure that both the class name and ID properties are short.
2. Define the properties of your source data in this class.
3. Create a Data Set rule for the associated entity.
 - a. In Dev Studio, click **Records Data Model Data Set**.
 - b. Click **Create**.
 - c. Provide a **Label**, and then select the **Stream** data type.
 - d. Enter the **Applies to** class of the source data class that you created in step 1.
 - e. Select to save your data set into your Pega Customer Decision Hub implementation ruleset.
 - f. Set the **Partition key** to be the same as your class key.
 - g. Save and check in your changes.
 - h. **Optional:** Create a data flow that starts from your source data set and ends in an abstract shape, and then define and add a data transform to help transform your data as needed.

4. Create the service integration by doing the following steps:
 - a. In Dev Studio, create an integration service class. For more information, see [Organizing rules into classes](#). For the purposes of being an integration service class, your class should be defined as an abstract class and saved into your Pega Customer Decision Hub implementation ruleset, or your integration ruleset, if it is separate from the implementation ruleset.
 - b. Create a *Page* property in your source data class.
 - c. Create a service activity that saves into your source stream data set. **For example:** If your previous page property is defined as *ClickStreamPage*, and your source data set is defined as *ClickStream*, your service activity would be defined as in the following figure:

Label	Method	Step page	Description
1. Loop When v	DataSet-Execute	.ClickstreamPage	Save the data from the clickstream page into the clickstream dataset

Method Parameters

Data set Clickstream	Operation Save
☐ Save list of pages defined in a named page	

[+ Add a step](#) [Collapse all steps](#)

Label	Method	Step page	Description
1. Loop When v	DataSet-Execute	.ClickstreamPage	Save the data from the clickstream page into the clickstream dataset

Method Parameters

Data set Clickstream	Operation Save
☐ Save list of pages defined in a named page	

[+ Add a step](#) [Collapse all steps](#)

- d. Create a [service package rule](#).
- e. Create a [REST service](#).
Define the Resource class of your integration class in the REST service, and set your service activity as the POST implementation activity. Configure your POST method to map to JSON based on the Page property in your integration class.

Message data
Description
Map to
Map to key
Mapping method

Message data
Description
Map to
Map to key
Mapping method

Defining the data source on the Profile Data Sources landing page

The Profile Data Sources landing page is used for managing the rules associated with your Customer Profile Designer aggregation. After creating the data class, use the Profile Data Sources page to define the aggregated data source.

1. In the Pega Customer Decision Hub interaction portal, click **Data Profile Data Sources**.
2. Select the data set that you want to associate with a customer context by clicking **Add Data set**.
3. In the **Add data set** window, select the data set that you created, and click **Add**.
4. Create a summary to aggregate the events in the data set by performing the following steps:
 - a. Click **Add Summary**.
 - b. In the **Create summary** window, select the data set to aggregate, and then click **Create**.**Note:** You can create summaries for stream data sets with one key, or data flows with a stream source and an abstract destination. Stream data sets and data flows with more than one key are not visible in the **Create summary** window.

Note: You can define a maximum of 3 summaries with 120 data aggregates each. Each aggregate must have a unique name.

✕

Edit aggregation

Name *

Accessed via mobile last 1 day

ID

AccessedViaMobileLast1Day

Aggregation

Function

Count ▾ of all records

Time window

Last ▾ 1 Days ▾

Conditions

Aggregate where ?

Event ▾ is equal to ▾ "Access" [Select values](#) +

and ▾

DeviceType ▾ is equal to ▾ "Mobile" [Select values](#) +

Cancel

Submit

- ×

Edit aggregation

Name *

Accessed via mobile last 1 day

ID

AccessedViaMobileLast1Day

Aggregation

Function

Count

▼

of all records

Time window

Last

▼

1

Days

▼

Conditions

Aggregate where ?

Event

▼

is equal to

▼

"Access"

Select values

+

and ▼

DeviceType

▼

is equal to

▼

"Mobile"

Select values

+

Cancel

Submit

- a. In Dev Studio, click **Configure Decisioning Decisions Data flows Real-time processing**.
- b. Find the data flow for Customer Profile Designer that lists the number of successfully processed records.
- c. Verify that the number matches the number of records that you sent in through the REST service.
5. Confirm the aggregation by opening the summary data set and browsing the data:
 - a. Open your summary data set.

Tip: The summary data set should have the same name as the summary rule that you created in the **Summaries** tab of the **Profile Data Sources** landing page.
 - b. Click **Actions Run**.
 - c. Select **Browse**, and then click **Run**.

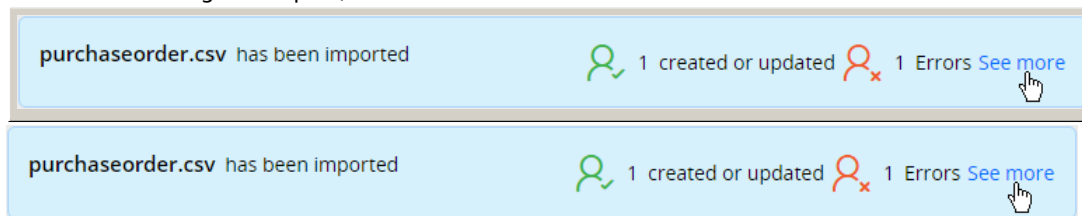
Manually importing customer data

In addition to the automated data ingestion process, you can use the import mechanism on the **Records** tab of Customer Profile Designer to import multiple instances using a .csv file.

To initiate an import, click the **Import** link above the instances grid. **Note:** The Import link is disabled if a data import is already in progress for the data type.

At a high level, the import process has the following three steps:

1. In the Pega Customer Decision Hub interaction portal, click **Data Profile Designer**.
2. Click on the context where you want to import customer records, for example, **Subscriber**.
3. Upload a .csv file and select one of the following import purposes:
 1. **Add or update** - Use this option to create new records for those instances in the import file that do not currently exist in the system, and to update records for those instances that do exist.
 2. **Add only** - Use this option to only create new records for those instances in the import file that do not currently exist in the system. Records for existing instances will not be updated.
 3. **Delete** - Use this option to delete matching records from the system.
4. Click **Map fields** to select the field to use for matching records, and specify the mappings for all fields to import.
5. Click **Import records** to start the import process. **Result:** When importing large files, the import progress can take some time. The system displays the status of the import at the top of the Data Management landing page. Once the import completes, the system displays the number of instances that were added or updated by the import. If there were errors during the import, the number of erroneous records is also shown.



6. To download a .csv file containing the records which failed to import, click **See more**.
The last column in this file contains a description of the error that occurred.
7. **Optional:** Click the **Export** link above the list of imported records to export them to a .csv file.
8. **Optional:** To manually add, update or remove records, perform one of the following actions:
 - **Add an instance** - To add an instance, click the **Add record** link at the bottom of the grid. Fill in all necessary fields and click anywhere outside the input row. To cancel your changes and discard the new record, click the delete icon at the end of the row.
 - **Modify an instance** - To modify an instance, click anywhere in the desired row. The row will toggle to editable mode. After making the desired changes, click anywhere outside the row to save the modified instance.
 - **Delete an instance** - To delete an instance, click the delete icon at the end of the row.

Note: The key value of an instance cannot be modified.

Note: Access to add, modify, or delete data instances is managed by standard Pega Platform security privileges. Administrators can use these privileges to grant or restrict access as needed.

Manually configuring the customer class

For industries other than telecommunications, finance, healthcare, or insurance, you must manually create and configure your customer class.

The customer class represents the customer entity in Pega Customer Decision Hub. The base customer class is *PegaMKT-Data-Customer*, which is backed by the *MKT_CUSTOMER* table in the outbound marketing schema. Pega Customer Decision Hub utilizes the external mappings defined on the customer class to load up customer data.

Typically, to support loading of customer data, you must configure a class that extends the base customer class.

What to do next: To configure the customer class, complete the following steps:

1. [Creating your customer class](#)
2. [Mapping a customer class to a database table](#)
3. [Optional: Creating a class key property for an extended customer class](#)

4. [Defining customer contexts for multilevel decisioning in Pega Customer Decision Hub](#)
5. [Extending and updating the CustomerSearchResultGrid section](#)
6. [Extending and updating the MKTProfileHeaderSearchOverlay section](#)
7. [Optimizing properties to speed up segment creation](#)

Creating your customer class

To support loading customer data, create a customer class that extends the base customer class.

1. Define your customer class.
 - a. Define a name for the customer class.
For more information, see [Creating classes](#).
 - b. Identify the customer class key. The default key property is *CustomerID*.
 - c. Identify the ruleset for the customer class. **For example:** *<your-application>-Rules*
 - d. Identify the associated entities (for example, *PurchaseData*) for this customer class.
2. In the Dev Studio header search text field, enter PegaMKT-Data-Customer. From the results, select the PegaMKT-Data-Customer class.
3. Click **Save as**.
4. Enter the name for the new customer class, and then click **Create and open**. **For example:** MyCo-Data-Customer
5. In the **Keys** section, ensure that only the correct key property is listed.
The default customer class key is *CustomerID*.
6. In the **Class inheritance** section, select the **Find by name first (Pattern)** check box, and then as the parent class, enter PegaMKT-Data-Customer.
7. On the **External Mapping** tab of the class rule, specify the properties available on the customer record and their backing columns in the customer table. Ensure that these mappings satisfy the following guidelines:
 - The column name does not exceed 27 characters
 - The table contains no empty rows
8. Click **Save**.

Mapping a customer class to a database table

After creating and configuring the class rule for your customer, create a database table instance to map your customer class to a database entity (table or view) in the marketing schema by completing the following steps.

1. In the **Records** menu, expand the **SysAdmin** category.
2. Right-click the **Database Table** node, and then select **+Create**.
3. Enter the name of your customer class and a short description, and then click **Create and open**.
4. In the **Database** section, enter details about the backing database entity. Work with the database administrator to find out what the details are.
Tip: By default, the backing database of Pega Customer Decision Hub is *ExternalMKTDData*.
5. Click **Save**.
6. Validate the mapping between your customer class and the specified database table by clicking **Test connectivity**.

What to do next: Note: Whenever changes are made to either the class's external mappings or the columns in the backing database entity, to ensure that the changes are propagated correctly, re-save the database table instance.

Optional: Creating a class key property for an extended customer class

By default, the key of the customer class is the *CustomerID* property that is used to store customer IDs. If your extended customer class uses a different class key, for example, *AccountID*, then you must create two copies of the *CustomerID* property and rename them to the same name as your custom class key.

1. In the Dev Studio header search text field, search for and select the *CustomerID* property that applies to the *Data-* class.
2. Click **Save as**.
3. In the **Label** field, enter the name of the property that you identified in [Creating your customer class](#).
4. In the **Add to ruleset** section, select your Rules ruleset.
5. Click **Create and open**.
6. In the Dev Studio header search text field, search for and select the *CustomerID* property that applies to the *PegaMKT-* class.
7. Repeat steps [2](#) to [5](#).

What to do next: After defining the new property, select it as the key for your extended customer class. For more information, see [Creating classes](#).

Defining customer contexts for multilevel decisioning in Pega Customer Decision Hub

Pega Customer Decision Hub uses customer contexts and associated data to represent various levels of customer contacts and related data entities. Use the Context Dictionary to review or update the default contexts, configure a different primary context, or add an additional customer context.

The [Context Dictionary](#) identifies the customer entities, the relationships between these entities, and any associated data to be used by Pega Customer Decision Hub.

By default, unless you import any accelerators such as [xCAR data model](#), the primary context is Customer, which is mapped to the *<your organization name>-Data-Customer* class, and the prospect context is Prospect, which is mapped to the *<your organization name>-Data-Customer-Prospect* class.

As part of implementing Pega Customer Decision Hub, review the available customer contexts and decide whether they fulfill your requirements. **Tip:** Consider the following questions:

- Do the default contexts cover the types of customer entities with which you communicate?
- What is the [primary context](#) for your application?
- What is the [relationship between different customer contexts](#) in your application?
- What type of [additional data should be associated](#) with each customer context?

1. In the navigation pane of App Studio, click **Settings Context Dictionary**.
2. Review the primary context.
 - a. Click the **Configure** icon by the name of a context.
 - b. Review the available properties.
 - c. If you defined a new extended customer class, select it in the **Class** field.
 - d. Review the associated data.
If you want to define additional associated data for your base customer class, see [Adding associated static data in the Customer Profile Designer](#).
3. **Optional:** To add a new context, click **Add context** and specify the name, customer class, and optional exclusion list.
4. **Optional:** To set your new context as the primary Context, click **Change primary** and select the context in the **Select Primary Context** dialog box.
Caution: Changing the primary context affects any segments and campaigns that use it. If you have already defined any segments or campaigns, recreate them manually after changing the primary context to ensure that they continue working.
5. **Optional:** If you want to select a non-default location for context storage, select it in the **Context storage** section. By default, the data resides in a relational database supported by Pega Customer Decision Hub in the form of a database table. If you plan to use segmentation, all data must be synchronized between the decision datastore and the relational tables.
6. Click **Save**.

Extending and updating the CustomerSearchResultGrid section

After creating a custom customer class, you must extend the rules that are related to list segments. This is necessary so that your application can use customer data from your extended customer class.

CustomerSearchResultGrid is the default section that can be used to display a segment in a grid based on a segment identifier.

1. In the Dev Studio header search text field, enter `customersearchresultgrid`, and then select the **CustomerSearchResultGrid** section from the search results.
2. Click **Save as**.
3. In the **Apply to** field, enter the custom customer class that you configured in [Creating your customer class](#).
4. In the **Add to ruleset** section, select your Rules ruleset.
5. Click **Create and open**.
6. Click **Check out**.
7. On the **Pages & Classes** tab, in the **Class** column, replace the base customer class name with the customer class name that you configured in [Creating your customer class](#).
8. On the **Design** tab, on the **Table Layout** panel, click the **Gear** icon.
9. Click **Submit**.
10. Click **Save**.
For more information on list segments, see the *Pega Customer Decision Hub User Guide* on the Pega Customer Decision Hub product page.

Extending and updating the MKTProfileHeaderSearchOverlay section

After creating a custom customer class, extend the rules that are related to customer profile. Update the *MKTProfileHeaderSearchOverlay* section to use the extended customer class and the equivalent properties. This is necessary for the search result to display as expected.

MKTProfileHeaderSearchOverlay is the section that hard codes the customer identifier and name properties to those in the out-of-the-box customer class.

1. In the Dev Studio header search text field, enter `mktprofileheadersearchoverlay`, and then select the **MKTProfileHeaderSearchOverlay** section from the search results.
2. Click **Save as**.
3. In the **Apply to** field, enter the Data-Portal class.
4. In the **Add to ruleset** section, select your Rules ruleset.
5. Click **Create and open**.

6. Click **Check out**.
7. On the **Pages & Classes** tab, for the *MKTProfileCustomerList.pxResults* page name, replace the value in the **Class** column with the customer class name that you configured in [Creating your customer class](#).
8. On the **Design** tab, click the **Table** layout panel, and click the **Gear** icon.
9. Click **Submit**. **Result:** The class displayed in the table layout should refresh to the custom customer class.
10. For each property field in the table:
 - a. Click the property field, and then click the **Gear** icon.
 - b. Update the **Link property** field to the corresponding property for the custom customer class.
 - c. Click the **Actions** tab.
 - d. Expand the first **Refresh-Other** section action.
 - e. In the **Data Transform** section, update the values passed into the *CustomerID* and *FullName* parameters to their respective properties from the custom customer class.
 - f. Expand the second **Refresh other** section action.
 - g. In the **Data Transform** section, update the value passed into the *CustomerID* parameter to its corresponding property from the custom customer class.
 - h. Click **Submit**.
11. Click **Save**.
12. Click **Check in**.
13. In the modal dialog box, type in the comment, and then click **Check in**.
For more information on customer profile, see the *Pega Customer Decision Hub User Guide* on the [Pega Customer Decision Hub product page](#).

Optimizing properties to speed up segment creation

To improve value retrieval performance when selecting segment criteria from a list, you can add and remove optimizations for properties which you plan to use to create segments. When you select criteria values from a list in a segment rule, you use the **Select items** dialog box to select one or more values for the criterion. If the property has many values, populating the **Select items** dialog box can result in poor performance. By optimizing the property, the number of values that are queried is reduced, resulting in improved retrieval performance.

1. In the **Application Explorer**, in the search text field, enter *Data-Admin-DistinctValueIndex* and press **Enter**.
2. Click **Data-Admin-DistinctValueIndex** to open the *Data-Admin-DistinctValueIndex* class.
3. Click **Create** to create a distinct value index instance of the property that you want to optimize. You must define the property as a text, integer, or decimal type.
 - a. In the **Short description** field, enter a brief description of the instance.
 - b. In the **Class name** field, enter the name of the class that the property is associated with. The class cannot have security restrictions defined for it.
 - c. In the **Property name** field, enter the name of the property whose values you want to retrieve.
 - d. Click **Save**. **Result:** An agent calls the **populateDistinctValues** activity, which searches the specified class and caches the distinct values of the property in a table.
4. Use the optimized property in a segment. For more information, see "Selecting Criteria Values from a List" in the *Pega Customer Decision Hub User Guide* on the Pega Customer Decision Hub product page.

Configuring the prospect class

If you want to use Pega Customer Decision Hub to support your customer acquisition efforts by making next-best-action decisions for potential customers, enable your customer class to allow the import of customer prospects.

Important: This procedure is not required if you used an xCAR accelerator component to create your data model. It is only necessary if you manually created your customer class.

Customer prospects are potential customers of your organization. Pega Customer Decision Hub can decision on their next-best-actions in the same way as for existing customers. To make sure that your customer class can load customer prospect data, do the following steps:

1. In Dev Studio, in the left navigation pane, click **App**.
2. In the **Classes** field, enter the name of your customer class, for example, *MyCo-Data-Customer*.
3. Right-click on the class name, and then click **Create Data model Property**.
4. In the **Label** field, enter *IsProspect*.
5. Ensure that the class selected in the **Apply to** field is your customer class.
6. Click **Create and open**.
7. In the **Property type** section, click **change**, and then select **TrueFalse**.
8. Click **Save**.

Ingesting customer data into Pega Customer Decision Hub in Pega Cloud

To make the best, most relevant next-best-action decisions, Pega Customer Decision Hub needs to have up-to-date data about your customers, their demographics, product holding, activity data, and other information that may be stored in third-party systems.

Because this information is updated frequently, you must ensure that the updates propagate to Pega Customer Decision Hub. Not all data is updated at the same frequency. Based on the frequency of updates, customer data can be classified as:

Non-volatile data

For example, data attributes which are updated at regular intervals, such as on a weekly or monthly basis.

Volatile data

For example, real-time transactional updates, mobile and web transactions, and tags and deletions of customer records, which are updated in real time.

This article covers the mechanism to propagate updates to non-volatile data to Pega Cloud in the process that is known as data ingestion or import. For volatile data, create separate services that propagate updates in real time to ensure optimum results.

A key design principle to follow is that customer data required by Pega Customer Decision Hub should be available at rest. Avoid attempting to make real-time access at the time of making a next-best-action decision as this can substantially impact performance.

Understanding the ingestion process

Before ingesting customer data into Pega Cloud, ensure that the required data is extracted from the data stores and transformed into a structure that matches the customer data model as defined in the context dictionary. This article assumes that the data to be ingested into Pega Cloud is available in a CSV or JSON file format.

The following Pega Platform capabilities are used in the data ingestion process:

Pega Cloud SFTP service

To securely transfer files between your enterprise and your Pega applications that Pega Cloud services runs. The service uses the Pega Cloud File Storage repository for reliable and resilient storage. For more information, see .

Pega Cloud File Storage

To store the files on Pega Cloud. For more information, see .

File listeners

To monitor the files and initiate the data processing. For more information, see [Using file listeners](#).

Data flows

To ingest data. For more information, see [Processing data with data flows](#).

Pega cases

To orchestrate the end-to-end data ingestion process. Also, to meet additional requirements for auditing, reporting, error tracking, and to retry in case of failure. For more information, see [Introduction to case management](#).

Three types of files are involved in the data ingestion process:

Data files

Files that contain customer data and are generated by your extraction-transformation-load (ETL) team.

Manifest files

Files that contain metadata about the data files being transferred.

Token files

Files that are used to signal that the data file transfer is complete.

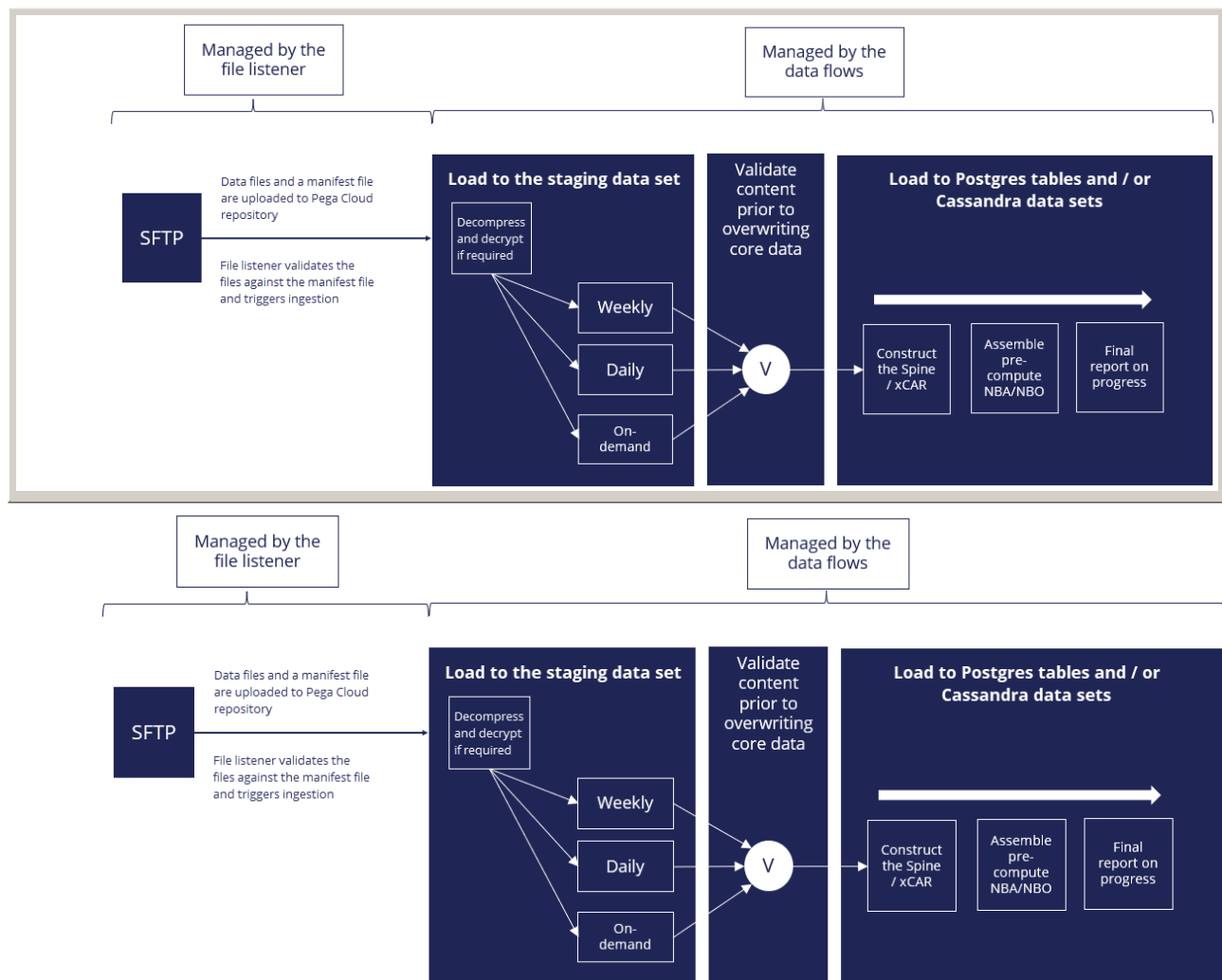
For more information, see [File types used in data ingestion](#).

The process starts when the data files and a manifest file are uploaded through the SFTP service into their respective directories in the Pega Cloud repository. A file listener that is configured to listen for the arrival of the manifest file runs a preprocessing check that validates the uploaded files against a manifest file. You can configure the manifest file to contain the file path and file-related metadata, such as information about the files to process, the total record count, and the number of records in each file, as well as the process type of the case that manages the ingestion process. If the preprocessing check passes, the file listener triggers the data ingestion.

Depending on your business requirements, you can configure several data flows to manage the data ingestion. In the following example, a staging data flow runs and ingests the data from the repository to a staging data set. After the data flow run, a postprocessing check validates the ingested data against the manifest file. If the validation passes, the data ingestion is considered successful. A second data flow runs and ingests the files from the staging data set to the core data set.

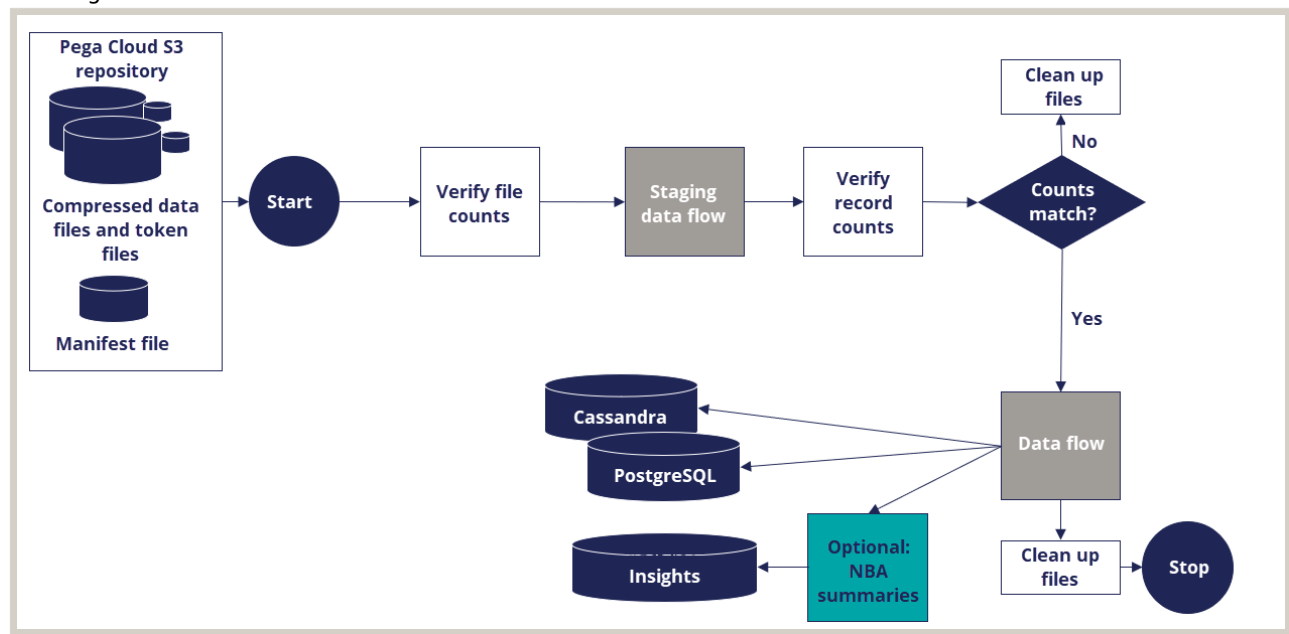
The following figure shows the main components, such as a file listener and data flows, that support the data ingestion process:

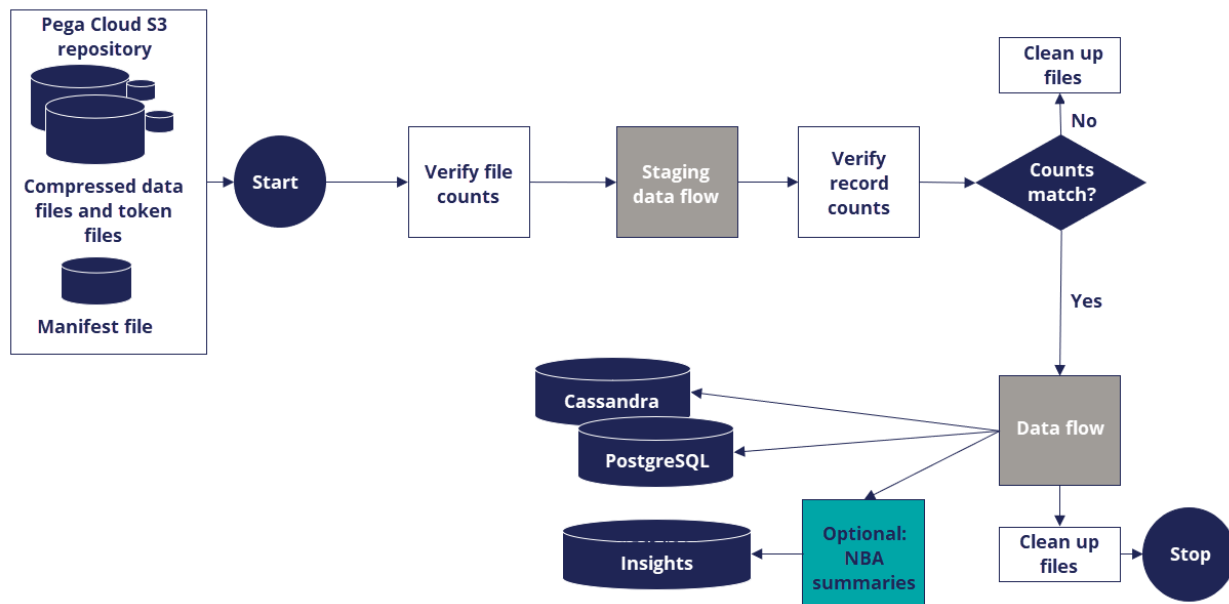
Data ingestion stages



The following figure shows a more detailed workflow of data ingestion, including the use of a manifest file to validate the transferred files.

Data ingestion workflow





To learn more, see [File types used in data ingestion](#) and [Best practices for data ingestion](#).

For a sample end-to-end implementation of the process, see [Configuring the data ingestion process](#).

File types used in data ingestion

Ingesting customer data into Pega Customer Decision Hub on Pega Cloud involves three types of files: data files, manifest files, and token files.

Before you configure the ingestion process, familiarize yourself with these file types, their functions, and formats. Establish the recommended file naming conventions and folder structures.

Data files

These files contain the actual customer data and are generated by the client's extraction-transformation-load (ETL) team. The files might contain new and updated customer data as well as data for purging or deleting existing records. One ingestion run can support one or more data files. A best practice is to divide large files into multiple smaller files to allow parallel transfer and faster processing. Files can be in CSV or JSON format, and can be compressed. .gzip and .zip compressions are supported. File encryption is supported, but requires custom Java coding.

The following table shows sample content of a CSV file for adding or updating customer data:

customer_id	first_name	last_name	middle_initial	clv_value	customer_type	billing_city	billing_state	billing_zip
811162399	Bill	Smith	O	3	I	TAYLOR	MI	48180
443470562	John	Kennel	F	11	I	PORTLAND	OR	97266

The following table shows sample content of a CSV file for deleting or purging customer data:

customer_id
811162389
443470534

Manifest files

The manifest file contains metadata about the data files being transferred. There is one manifest record for each ingestion or purge to be transferred. Examples of ingestion or purge: customer data ingestion, customer data purge, account data ingestion, account data purge. While processing, the file listeners listen for manifest files. Manifest files are in XML format.

The manifest file is backed by a data model which is a part of the *Data-Ingestion-Manifest* class.

The following example is a manifest file that is typically used:

```
<?xml version="1.0" ?>
<manifest>
```

```

<processType>CustomerDataIngest</processType>
<totalRecordCount>1300</totalRecordCount>
<files>
<file>
<name>CustomerDataIngest_MMDDYYYY_000.csv</name>
<size>149613</size>
<recordCount>700</recordCount>
</file>
<file>
<name>CustomerDataIngest_MMDDYYYY_001.csv</name>
<size>125613</size>
<recordCount>600</recordCount>
</file>
</files>
</manifest>

```

Field	Description	Example
processType	Type of data being loaded. This field also identifies the data flow to run.	CustomerDataIngest AccountDataIngest
totalRecordCount	Total number of records across all data files.	1300
recordCount	Record count for one file.	700 600
name	Name of the data file that needs to be agreed with the client and configured in the file data set. The name can have suffix substitution.	CustomerDataIngest_MMDDYYYY_000.csv CustomerDataIngest_MMDDYYYY_001.csv
size	Size for one file in bytes. This field is optional and used if there is a need to do file size validation. Additional process work is required if size validation must be performed.	149613 125613

Token files

Token files are used to signal that the data file transfer is complete. Token files are optional if your SFTP client application can be configured to send the manifest file after successfully transmitting all data files. Otherwise, token files are required. Token files are typically generated by the SFTP client application when the SFTP has completed the transfer of a file. There is one token for each file transferred. A token file does not need to contain any data, only the presence or absence of the file is checked.

The manifest file and the data files are transferred to the Amazon S3 location through SFTP. Manifest files are typically very small in size (a few kilobytes). The actual data files are typically quite large (multiple gigabytes). As a best practice, large files are divided into smaller files that can be transferred in parallel by the SFTP client application. As a result, the manifest files are transferred before the data files.

Pega Platform file listeners are configured to listen and process the manifest files. As manifest files arrive first, ahead of the data files, their processing completes almost immediately, which in turn kicks off the data flow process. However, the data flow process fails as the data file transmission has not been completed.

Hence, it is critical that the manifest file is sent last. If this cannot be guaranteed by your SFTP client application, then additional work by the SFTP team is required to create a token file and send it after every data file is successfully transferred. The ingestion case is configured to wait until all the token files have arrived before invoking the data flow to start loading the data.

For example, after the `DataIngestion_04182020_1.csv` is successfully transmitted, a token file `DataIngestion_04182020_1.csv.tok` or `DataIngestion_04182020_1.csv.done` is created by the SFTP application.

File naming convention and folder structure

Both the file listeners and file data sets use pattern matching based on file names. As a result, the following file types, naming conventions, and folder structures must be agreed in advance:

- Naming convention of manifest files
- XML as the format of manifest files
- Naming convention of data files

- For CSV:
 - Header row (field names to be mapped to the Spine tables and xCAR data sets)
 - Delimiter

Best practices for data ingestion

Review these best practices for configuring data ingestion processes in your application.

Add data flow run details to the case review screen to help debug issues

Ingestion case review screen with data flow details

Ingestion (I-4005) RESOLVED-COMPLETED

Assignments

View all ☐



Nothing but space

✓ CHECKFORFILES >
 ✓ VALIDATEMANIFEST >
 ✓ UPLOADTOSTAGING >
 ✓ UPLOADTOXCAR >
 ✓ CLEANUP

[Information](#) | Audit

< Ingestion Details

Ingestion Type:	AccountDataIngest
No of Files Expected:	1
No of Files Found:	1

> Staging data flow details

< xCAR staging data flow details

Data flow ID:	DF-12008
Dataflow Status:	Completed
Total records updated in DB:	60

Ingestion (I-4005) RESOLVED-COMPLETED

Assignments View all ☐



Nothing but space

✓ CHECKFORFILES
✓ VALIDATEMANIFEST
✓ UPLOADTOSTAGING
✓ UPLOADTOXCAR
✓ CLEANUP

Information
Audit

⌵ Ingestion Details

Ingestion Type:	AccountDataIngest
No of Files Expected:	1
No of Files Found:	1

> Staging data flow details

⌵ xCAR staging data flow details

Data flow ID:	DF-12008
Dataflow Status:	Completed
Total records updated in DB:	60

Provide reports and access for the client's ETL and production support

teams

Provide insight into the execution status (which is normally set up as an agent) by following these best practices:

- Configure a case progress and monitoring report to identify execution statistics and status.
- Provide access to the Case Manager portal to both the client's ETL and production support teams.
- Schedule the report to be sent out as email attachment to the ETL and production support teams.

Case progress report

Displaying 50 records								
Ingestion Type	Case ID	Entered on	Case status	No of Files Expected	No of Files Found	Staging DataFlow ID	Staging Dataflow Status	Staging Records Processed
LineDataIngest	I-12016	4/10/20	Resolved-Withdrawn	1	1	DF-24053	Completed	789
AccountDataIngest	I-12015	4/10/20	Resolved-Withdrawn	1	0			
LineDataIngest	I-12014	4/8/20	Resolved-Withdrawn	1	1	DF-24044	Completed	789
LineDataIngest	I-12013	4/8/20	Resolved-Completed	1	1	DF-24042	Completed	789

Displaying 50 records								
Ingestion Type	Case ID	Entered on	Case status	No of Files Expected	No of Files Found	Staging DataFlow ID	Staging Dataflow Status	Staging Records Processed
LineDataIngest	I-12016	4/10/20	Resolved-Withdrawn	1	1	DF-24053	Completed	789
AccountDataIngest	I-12015	4/10/20	Resolved-Withdrawn	1	0			
LineDataIngest	I-12014	4/8/20	Resolved-Withdrawn	1	1	DF-24044	Completed	789
LineDataIngest	I-12013	4/8/20	Resolved-Completed	1	1	DF-24042	Completed	789

For more information, see [Creating a report](#).

Provide data to identify and resolve errors

Include the *CaseID*, *DataFlowID*, and the details that are available from the **Batch processing** landing page in error reports. This information provides key details to identify and resolve processing errors.

Check the S3 file count and correct file location regularly

The processing of the files in the S3 SFTP folders uses pattern matching to identify the files to be processed. The production support team must regularly verify that the files are being sent to the appropriate S3 locations and that the files in those locations match the pattern for the intended processing. The client's ETL teams regularly update their processing as data attributes are added or removed from the data sets transferred and mistakes can be made as the processing jobs that support those changes are updated.

Use parallel processing

File processing in Pega Platform provides several opportunities to process data in multiple parallel streams, which can substantially reduce processing time. To take advantage of the parallel processing of the data flows, divide large files into multiple smaller files, each with approximately the same number of records.

The number of concurrent partitions is determined by the **Number of threads** parameter that you set when creating a data flow run, as shown in the following figure:

Changing the number of threads for a data flow run

Decisioning: Data Flows

Refresh Help Close

Batch processing Real-time processing Single case processing

The execution of these data flows is completed when data in the main input is read entirely or when it is explicitly stopped. Use this landing page to create, manage and monitor batch data flow runs.

New

ID	Data flow	Service	Priority	Start time	Last processed	# Records processed	# Failures
No data flow work items created yet							

New: Data Flow Work Item

Submit

Associate a data flow with this run

Applies to

Data flow

Access group

UPlus:CDHAdmins

Service instance name

Batch

Number of threads

5

Priority

Medium

Decisioning: Data Flows

Refresh Help Close

Batch processing Real-time processing Single case processing

The execution of these data flows is completed when data in the main input is read entirely or when it is explicitly stopped. Use this landing page to create, manage and monitor batch data flow runs.

New

ID	Data flow	Service	Priority	Start time	Last processed	# Records processed	# Failures
No data flow work items created yet							

New: Data Flow Work Item

Submit

Associate a data flow with this run

Applies to

Data flow

Access group

UPlus:CDHAdmins

Service instance name

Batch

Number of threads

5

Priority

Medium

One thread is associated with each input file. If the input files are not approximately equal in size, processing takes more time as the entire process must wait until the last file is processed. For more information, see [Creating a batch run for data flows](#).

You can also manage the thread count setting on the Services landing page, as shown in the following figure:

Changing the thread count for a batch data flow service

Decisioning: Services Refresh Help Close

Decision Data Store Adaptive Decision Manager Data flow Real-time Data Grid Stream

Service Batch

Data flow nodes Edit settings Node details

View the current state of the data flow service, manage nodes and execute operations.

Status	Node name	IP address	Action
NORMAL	→ uplusdataflows	192.168.x.x	Execute

Edit batch settings X

Node type
Batch

Thread count
5

Maximum number of active runs
10

Cancel Submit

For more information, see [Configuring the Data Flow service](#).

For example, in a scenario where there are three nodes, you can process 15 files with 1 million records in each file faster than three files with 5 million records in each file or one file with 15 million records.

Archive files after processing

Archive the files in your Pega Cloud File Storage repository at the end of an ingestion run. If errors occur during an ingestion run and you have to reprocess the files, you can use the archived files and save time by not having to retransfer the files to the repository.

Configuring the data ingestion process

Ensure that the customer data in your Pega Customer Decision Hub application is always current. Propagate the changes in your customer data from an external system to Pega Customer Decision Hub through a process known as data ingestion, or import.

The following process shows the configuration of the repository, the manifest file, and the rules for loading customer data from CSV files onto a Pega Cloud SFTP server into the Account, Subscriber, and Product Holding classes. Use this example to understand which types of rules you need in order to configure your system for data ingestion and plan your own

configuration.

Rule type	Rule name	Ruleset
Data set	<i>UPlusImportAccount</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data set	<i>UPlusImportSubscription</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data set	<i>UPlusImportHolding</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data flow	<i>UPlusImportAccount</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data flow	<i>UPlusImportSubscription</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data flow	<i>UPlusImportHolding</i>	<i>UPlusCDH-Rules:01-01-01</i>
Case type	<i>Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>
Flow	<i>UploadToXCAR</i>	<i>UPlusCDH-Rules:01-01-01</i>
Property	<i>processType</i>	<i>UPlusCDH-Rules:01-01-01</i>
Property	<i>DataFlowClassName</i>	<i>UPlusCDH-Rules:01-01-01</i>
Property	<i>DataFlowName</i>	<i>UPlusCDH-Rules:01-01-01</i>
Property	<i>ManifestDetails</i>	<i>UPlusCDH-Rules:01-01-01</i>
Decision table	<i>UPlusIngestionProcessType</i>	<i>UPlusCDH-Rules:01-01-01</i>
Activity	<i>ExecuteIngestionDF</i>	<i>UPlusCDH-Rules:01-01-01</i>
Parse XML	<i>UPlusManifestParser</i>	<i>UPlusCDH-Rules:01-01-01</i>
Service package	<i>UPlusIngestionFileProcessing</i>	<i>UPlusCDH-Rules</i>
Service file	<i>UPlusProcessManifestFile</i>	<i>UPlusCDH-Rules:01-01-01</i>
File listener	<i>UPlusAccountFileListener</i>	<i>UPlusCDH-Rules</i>
File listener	<i>UPlusSubscriptionFileListener</i>	<i>UPlusCDH-Rules</i>
File listener	<i>UPlusHoldingFileListener</i>	<i>UPlusCDH-Rules</i>

What to do next: Configure the data ingestion process in your system by performing the following tasks:

1. [Configuring a repository to ingest customer data](#)
2. [Creating a manifest file to ingest customer data](#)
3. [Creating data sets to ingest customer data](#)
4. [Creating data flows to ingest customer data](#)
5. [Creating a case type to ingest customer data](#)
6. [Creating properties to ingest customer data](#)
7. [Creating a decision table to ingest customer data](#)
8. [Creating an activity to ingest customer data](#)
9. [Creating a Parse XML rule to ingest customer data](#)
10. [Creating a service package to ingest customer data](#)
11. [Creating a service file to ingest customer data](#)
12. [Creating file listeners to ingest customer data](#)
13. [Starting file listeners to ingest customer data](#)

Configuring a repository to ingest customer data

Configure a repository for storing the manifest file and data files that you want to ingest into your Pega Customer Decision Hub.

- If your system is deployed in a Pega Cloud services environment, configure your SFTP application to point to your S3 file repository as the root directory for file destination paths.
The Pega Cloud services team defines the S3 file repository for your system in advance. For more information, see .
- If your system is deployed in a third-party cloud, perform the following actions:
 - a. Create a repository.
For instructions, see [Creating a repository](#). **Note:**

If the repository that you create is to be shared across environments, you must manually update the root path to identify the environment. For example, if the same repository is to be shared by the development and QA environments:

- Create folders named Dev and QA in the SFTP root folder.
- Update the root path in the repository rule.

The following figure shows how a Dev folder is defined as the root path for the repository in the QA environment, for non-Pega Cloud implementations:

Sample repository configuration with a root path configured to Dev folder

Edit Repository: AWS Repository

ID AWSS3 RS CDHSample [Edit]

Definition

History

Repository type

AMAZON S3

Change

Configuration

Bucket*

pegamkt-dev-bucket

Authentication profile

AWS

☐ Server side data encryption with KMS managed keys

Root path

Dev

Edit Repository: AWS Repository
ID AWSS3 RS CDHSample [Edit]

Definition
History

Repository type



Change

Configuration

Bucket★

pegamkt-dev-bucket

Authentication profile

AWS

☐ Server side data encryption with KMS managed keys

Root path

Dev

- b. Configure your SFTP application to point to your S3 file repository as the root directory for file destination paths.

Creating a manifest file to ingest customer data

Provide metadata about the files that you want to transfer by creating a manifest file and uploading it to the Pega Cloud SFTP server.

The manifest file can include such information as files names and sizes, the number of records in each file, and the process type of the case that manages the ingestion process. In a system configured for ingestion, a file listener listens for the arrival of a manifest file, and then runs a preprocessing check that validates the uploaded files against the manifest file. If the preprocessing check is successful, the system starts the ingestion process.

1. Create the manifest file in XML format. **For example:**

```
<?xml version='1.0' encoding='UTF-8'?>
<manifest>
  <processType>AccountIngestion</processType>
  <totalRecordCount>1300</totalRecordCount>
  <files>
    <file>
      <name>account_data_pegas_20210531.csv.gz</name>
      <size>1437966</size>
      <recordCount>700</recordCount>
    </file>
    ...
  </files>
</manifest>
```

Field	Description	Example
processType	Type of data being loaded. This field also identifies the data flow to run.	AccountIngestion
totalRecordCount	Total number of records across all data files.	1300
recordCount	Record count for one file.	700
name	Name of the data file that needs to be agreed with the client and configured in the file data set. The name can have suffix substitution.	account_data_pegas_20210531.csv.gz

Field	Description	Example
size	Size for one file in bytes. This field is optional and used if there is a need to do file size validation. Additional process work is required if size validation must be performed. Note: The file path in the manifest file name needs to be relative to the repository configured in the data set. For example, if your file is in the root folder of your repository, you can directly access the file name, if not, you will need to include the folder structure as a part of the name.	1437966

2. Upload the manifest file to the Pega Cloud SFTP server.

- [File types used in data ingestion](#)

Creating data sets to ingest customer data

Create data sets to hold the files that you want to transfer.

The goal of this procedure is to create the following rules with sample parameter values:

Rule type	Rule name	Ruleset
Data set	<i>UPlusImportAccount</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data set	<i>UPlusImportSubscription</i>	<i>UPlusCDH-Rules:01-01-01</i>
Data set	<i>UPlusImportHolding</i>	<i>UPlusCDH-Rules:01-01-01</i>

Important: When you configure these rules in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Data Model** category, and then click **Data Set**.
3. On the **Data Set** tab, click **Create**.
4. On the **New** tab, configure the following parameters:
 - a. In the **Label** field, enter *UPlusImportAccount*.
 - b. In the **Type** field, select **File**.
 - c. In the **Apply to** field, enter the name of the applicable customer class, for example, *UPlusCDH-Data-Comms-Account*.
 - d. In the **Add to ruleset** fields, select ***UPlusCDH-Rules*** and ***01-01-01***.
5. Click **Create and open**.
6. In the **Data source** section, select **Files on repository**.
7. In the **Connection** section, enter the name of your Pega Cloud [repository](#), for example, *pegacloudrepository*.

Data source and connection configuration

The screenshot shows the 'Edit Data set: UPlusImportAccount' configuration window. The breadcrumb path is 'CL UPlusCDH-Data-Comms-Account > ID UPlusImportAccount > RS UPlusCDH-Rules:01-01-01'. The 'File' tab is active. In the 'Data source' section, 'Files on repository' is selected. In the 'Connection' section, the 'Repository configuration' field contains the value 'pegacloudrepository'.

Edit Data set: UPlusImportAccount [Available]

CL UPlusCDH-Data-Comms-Account ID UPlusImportAccount RS UPlusCDH-Rules:01-01-01

File Mapping Specifications History

Data source

☒ Files on repository
☐ Embedded file

Connection

Repository configuration ★
 pegacloudrepository

8. In the **File configuration** section, configure the parameters for the [manifest file](#):
 - a. Select **Use a manifest file**.
 - b. In the **Manifest file path** field, enter the path to your manifest file.
 - c. If the files are compressed, select the **File is compressed** checkbox, and then select the compression type.

Manifest file configuration

File configuration

☐ Use a file path
☒ Use a manifest file

Manifest file path ★ ⓘ
 /csvtransfer/manifest.xml [Preview file](#)

☒ File is compressed

Compression type ★
 GZip

☐ Custom stream processing ⓘ

File configuration

☐ Use a file path
☒ Use a manifest file

Manifest file path ★ ⓘ
 /csvtransfer/manifest.xml [Preview file](#)

☒ File is compressed

Compression type ★
 GZip

☐ Custom stream processing ⓘ

9. In the **Parser configuration** section, configure the following parameters:
 - a. In the **File type** field, select the type of files to be ingested, for example, CSV.
 - b. Click **Configure automatically** to automatically configure the parser settings for the files in your data set.

Parser configuration

Parser configuration

File type★
 CSV Configure automatically

☒ File contains header

Delimiter character★
 Comma (,)

Supported quotation marks
 None

Date Time format
 yyyy-MM-dd HH:mm:ss

Date format
 yyyy-MM-dd

Time Of Day format
 HH:mm:ss

Parser configuration

File type★
 CSV Configure automatically

☒ File contains header

Delimiter character★
 Comma (,)

Supported quotation marks
 None

Date Time format
 yyyy-MM-dd HH:mm:ss

Date format
 yyyy-MM-dd

Time Of Day format
 HH:mm:ss

- Click the **Mapping** tab, and then adjust the mapping between the columns in the files to be ingested and the corresponding properties in the customer class.

Sample mapping

File Mapping Specifications History

Create missing properties << < Page 1 of 2 > >>

Column	Property
customerID	.CustomerID  
CustomerType	.AccountStatus  
billing_city	.pyCity  
billing_state	.pyState  
billing_zip	.pyPostalCode  

File
Mapping
Specifications
History

Create missing properties
Page 1 of 2

Column	Property
customerID	.CustomerID
CustomerType	.AccountStatus
billing_city	.pyCity
billing_state	.pyState
billing_zip	.pyPostalCode

11. Click **Save**.

12. Repeat steps 3 through 11 to create the other two data sets:
 - UPlusImportSubscription
 - UPlusImportHolding

Creating data flows to ingest customer data

Create data flows to process the files from the source data sets to the destination data sets.

The goal of this procedure is to create the following rules with sample parameter values:

Rule type	Rule name	Apply to	Ruleset
Data flow	UPlusImportAccount	UPlusCDH-Data-Comms-Account	UPlusCDH-Rules:01-01-01
Data flow	UPlusImportSubscription	UPlusCDH-Data-Comms-Subscription	UPlusCDH-Rules:01-01-01
Data flow	UPlusImportHolding	UPlusCDH-Data-Comms-Holding	UPlusCDH-Rules:01-01-01

Important: When you configure these rules in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Records**.

2. Expand the **Data Model** category, and then click **Data Flow**.

3. On the **Data Flow** tab, click **Create**.

4. On the **New** tab, configure the following parameters:

a. In the **Label** field, enter *UPlusImportAccount*.

b. In the **Apply to** field, enter the name of the applicable customer class, for example, *UPlusCDH-Data-Comms-Account*.

c. In the **Add to ruleset** fields, select *UPlusCDH-Rules* and *01-01-01*.

5. Click **Create and open**.

6. Double-click the source shape in the diagram, and then configure the following parameters:

a. In the **Source** field, select **Data set**.

b. In the **Data set** field, enter the name of the appropriate [data set](#) that you configured earlier, which in this example was *UPlusImportAccount*.

Configuring the source data set

Source configurations

Name

UPlusImportAccount

Results in

UPlusCDH-Data-Comms-Account

Import data from

Source *

Data set

Data set *

UPlusImportAccount

Data set type

File

Cancel

Submit

Source configurations

Name

UPlusImportAccount

Results in

UPlusCDH-Data-Comms-Account

Import data from

Source *

Data set

Data set *

UPlusImportAccount

Data set type

File

Cancel

Submit

7. Double-click the destination shape in the diagram, and then configure the following parameters:
- In the **Destination** field, select **Data set**.
 - In the **Data set** field, enter *Account*.
 - In the **Save options** field, select **Insert new and overwrite existing records**.

Configuring the destination data set

Destination configurations

Name

Account

Results in

UPlusCDH-Data-Comms-Account

Output data to

Destination

Data set

Data set

Account

Data set type

Database Table

Database type

RDBMS

Save options

☐ Only insert new records

☒ Insert new and overwrite existing records

Cancel

Submit

Destination configurations

Name

Account

Results in

UPlusCDH-Data-Comms-Account

Output data to

Destination

Data set

Data set

Account

Data set type

Database Table

Database type

RDBMS

Save options

☐ Only insert new records

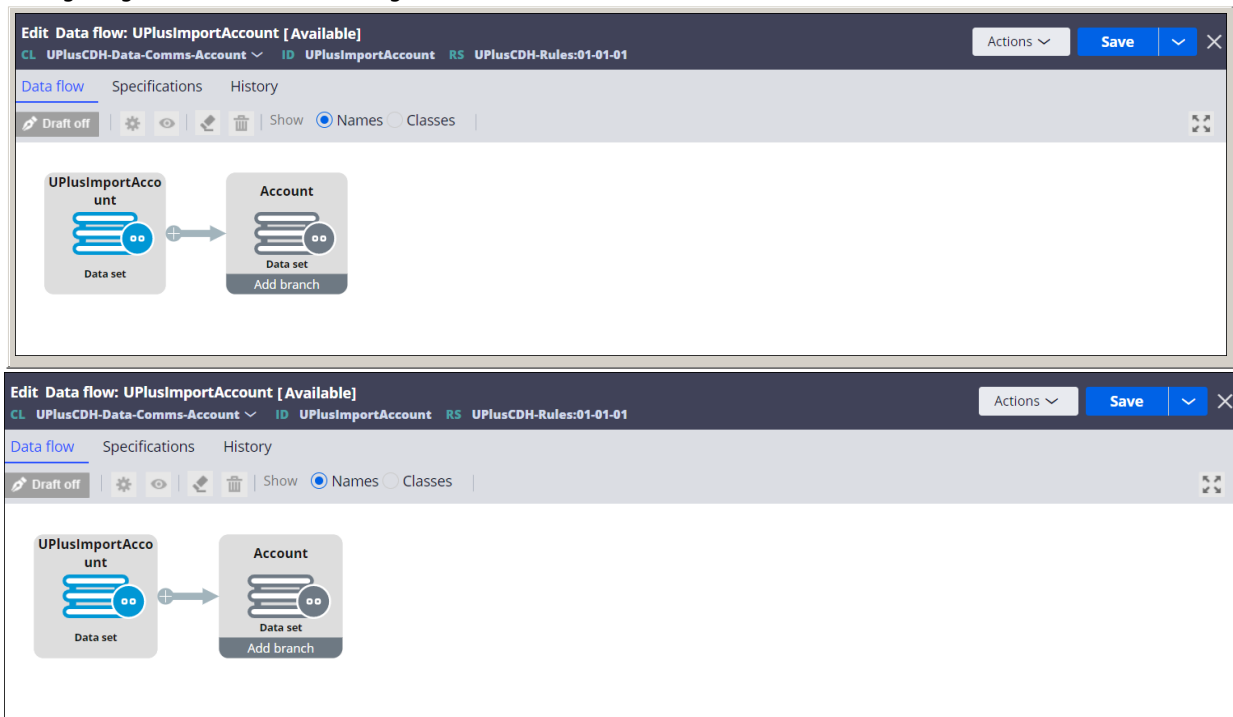
☒ Insert new and overwrite existing records

Cancel

Submit

8. Click **Save**.

Configuring a data flow for data ingestion



9. Repeat steps 3 through 8 to create the other two data flows:
 - *UPlusImportSubscription*
 - *UPlusImportHolding*

Creating a case type to ingest customer data

Manage the data ingestion process by configuring an ingestion case type. In the case type, you can specify when to start the ingestion process and how to handle errors.

The goal of this procedure is to create the following rules with sample parameter values:

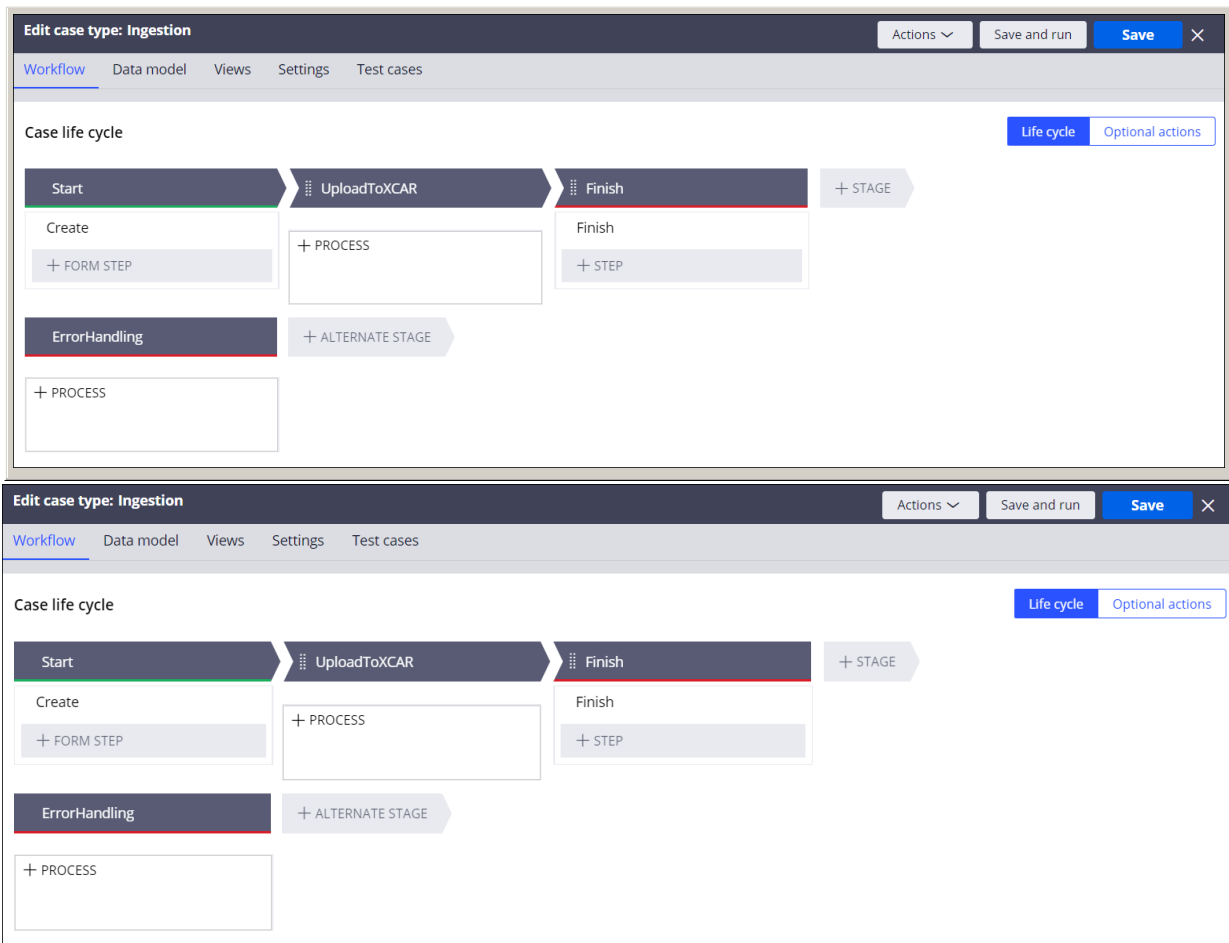
Rule type	Rule name	Ruleset
Case type	<i>Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>
Flow	<i>UploadToXCAR</i>	<i>UPlusCDH-Rules:01-01-01</i>

Important: When you configure these rules in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Case types**.
2. Click **Add a case type**.
3. In the **Name** field, enter *Ingestion*.
4. Expand the **Advanced Settings** section, and then configure the following parameters:
 - a. In the **Ruleset** field, select ***UPlusCDH-Rules***.
 - b. In the **Ruleset version** field, select ***01-01-01***.
5. Click **Submit**.
6. Add stages to the case type with the following settings:

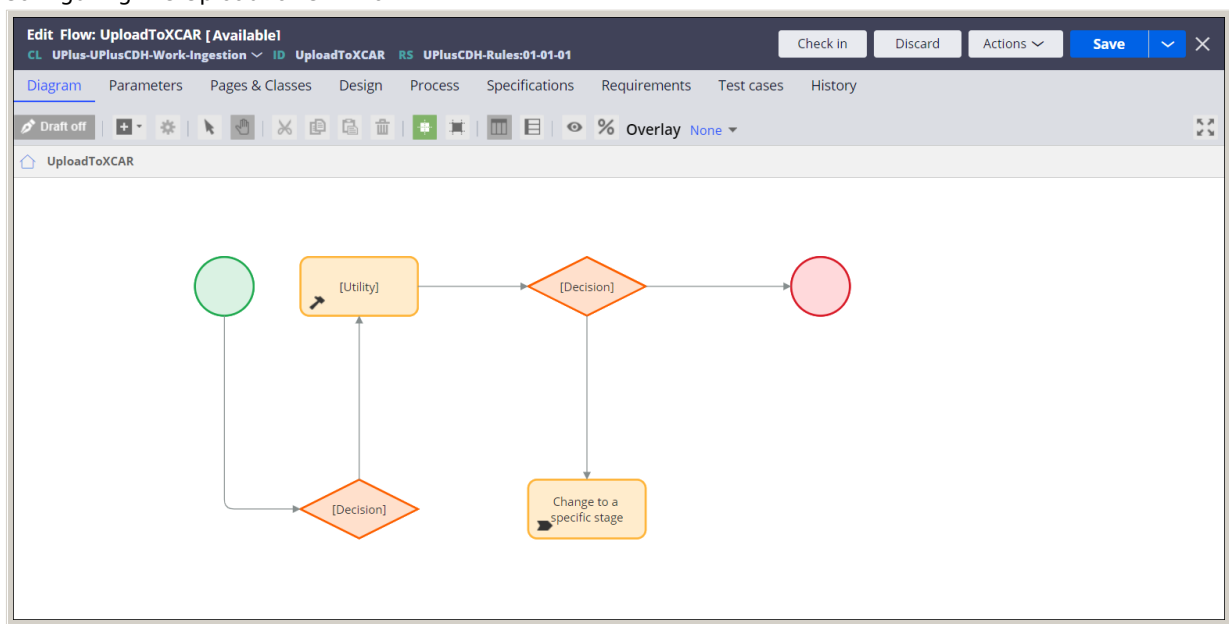
Settings	Start stage	UploadToXCAR stage	Finish stage	ErrorHandling stage
When all processes in this stage are complete		Automatically move to next stage	Resolve the case	Resolve the case
Set case status on stage entry	New	Pending-Importing	Completed	Resolved-Failed
Resolution status			Resolved-Completed	Resolved-Failed

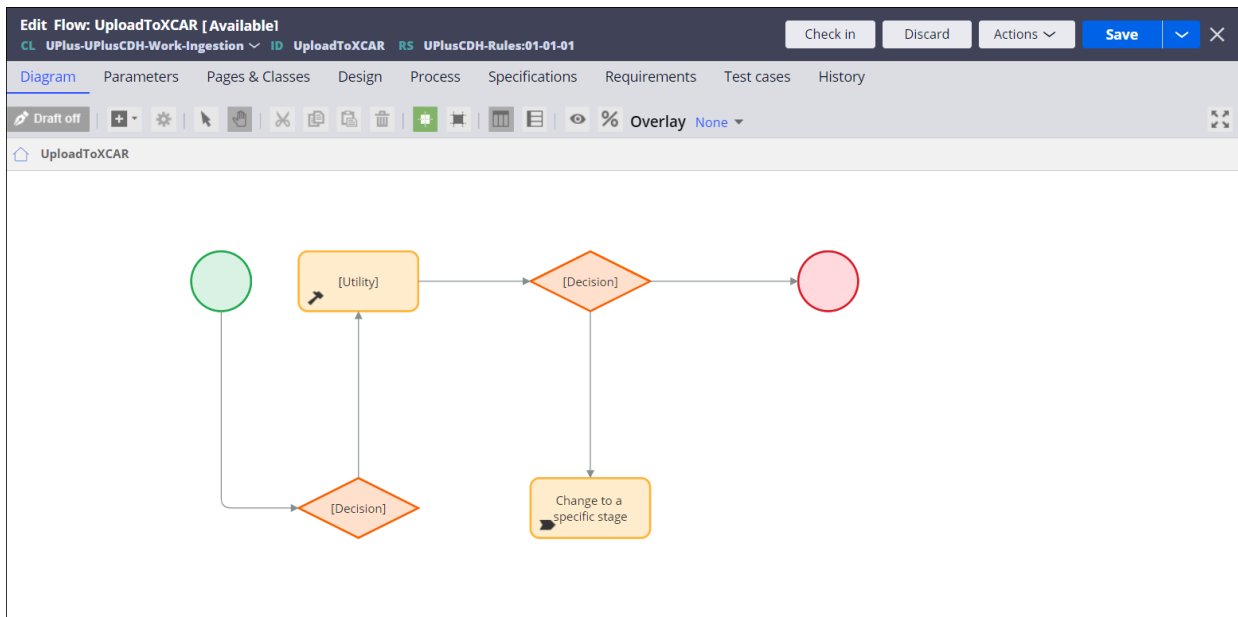
Adding stages to the ingestion case type



7. Hover over the **UploadToXCAR** process, click the **Configure process** icon, and then click **Open process**.
8. On the **Create Flow** tab, configure the following parameters:
 - a. In the **Label** field, enter a short description of the flow, for example, Upload to XCAR.
 - b. In the **Apply to** field, select the ingestion work class, which in this example is **UPlus-Work-Ingestion**.
 - c. In the **Add to ruleset** field, select **UPlusCDH-Rules**.
9. Click **Create and open**.
10. Add components to the process flow and connect them as shown in the following figure:

Configuring the UploadToXCAR flow





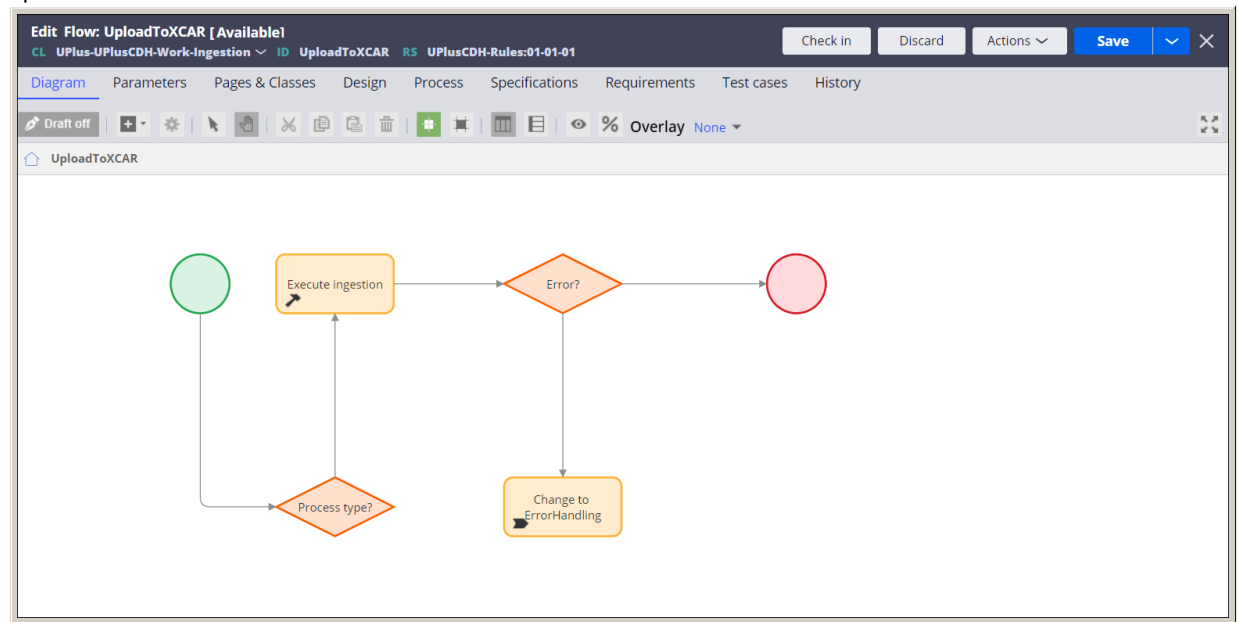
Note: The second *Decision* shape and the *Change to a specific stage* shape are optional. You can use them to configure error handling in your ingestion case type.

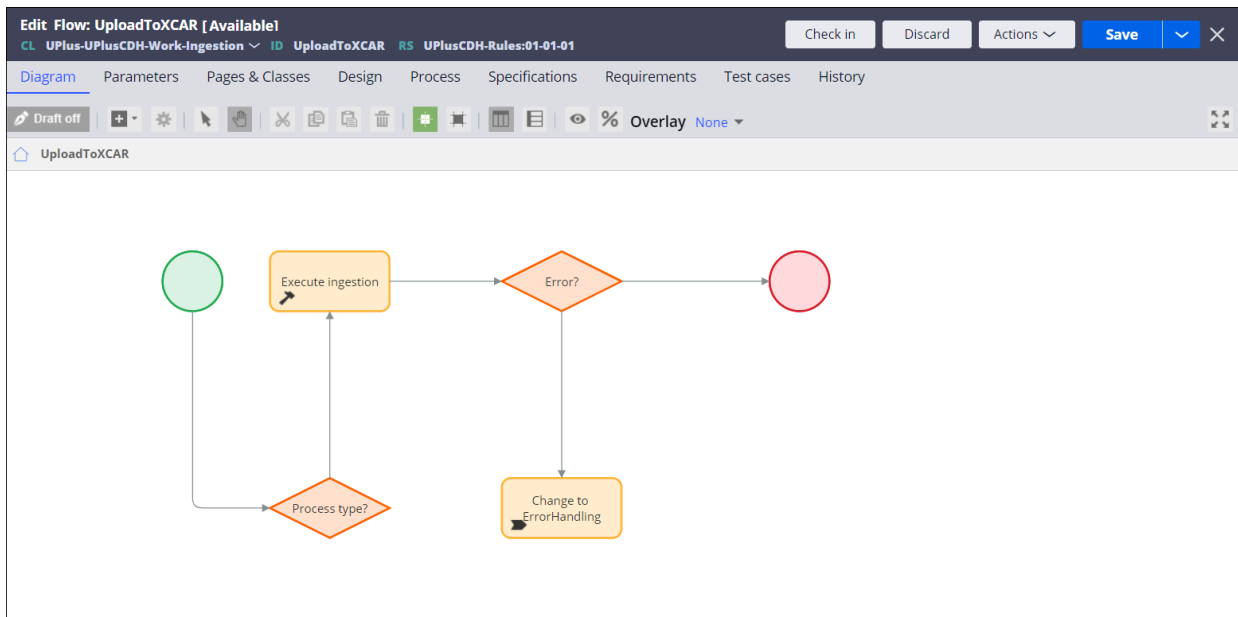
11. Double-click and rename each component as follows:

Component	Name
Decision (first)	Process type?
Utility	Execute ingestion
Decision (second)	Error?
Change to a specific stage	Change to ErrorHandling

Result:

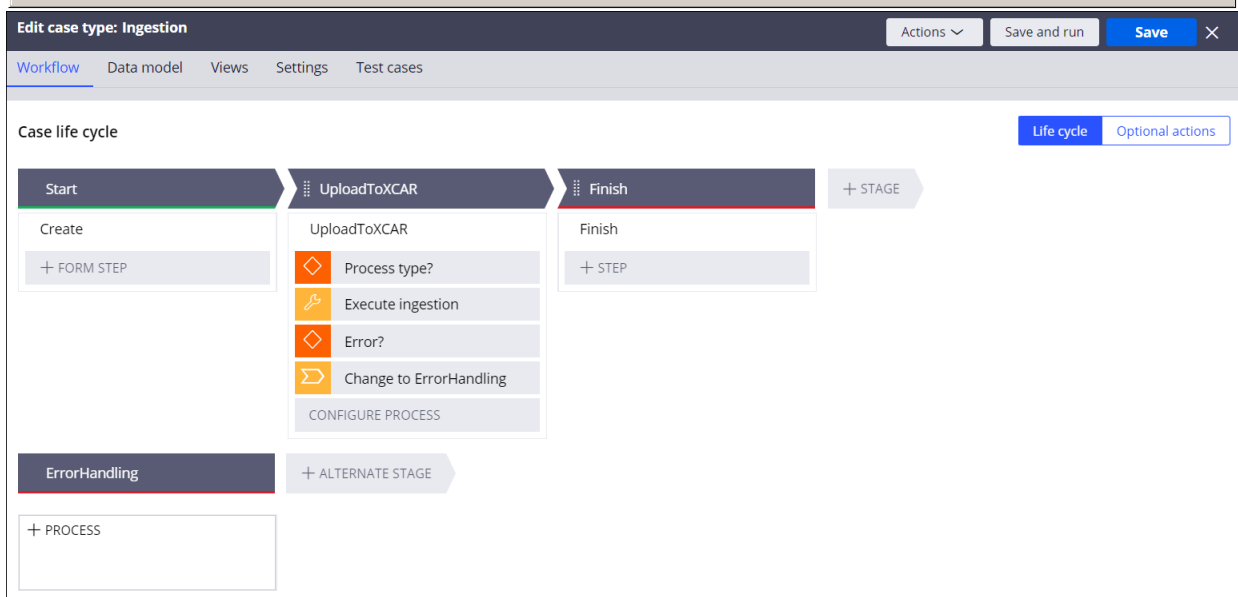
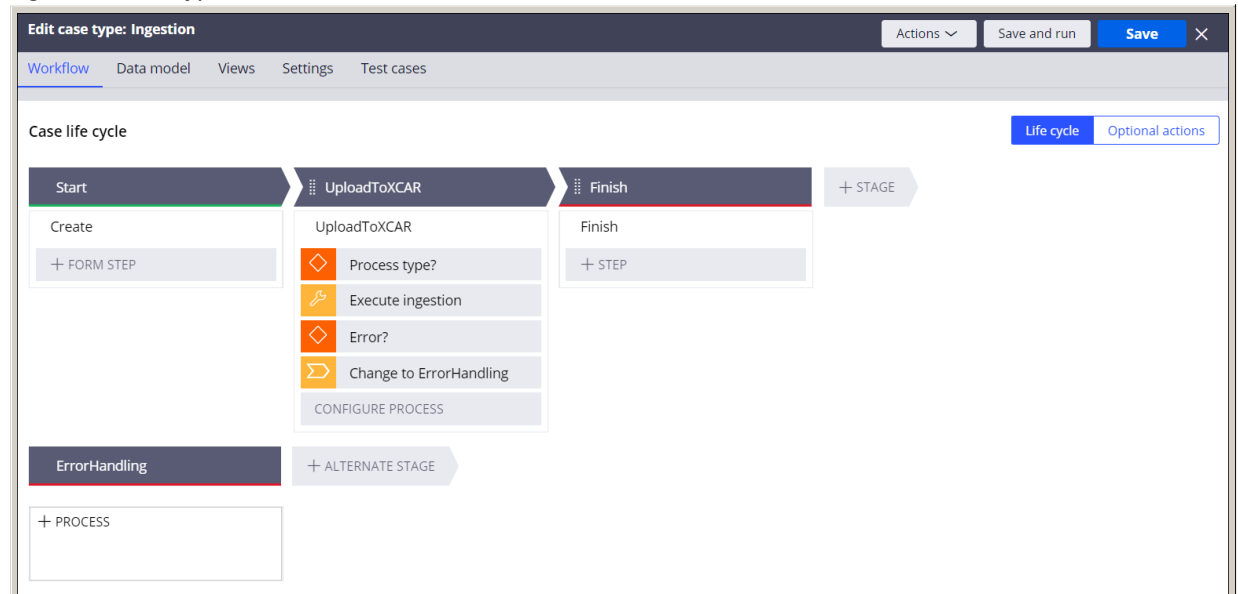
UploadToXCAR flow





In the case workflow view, the process flow in the UploadToXCAR stage contains the following steps:

Ingestion case type



12. Click **Save**.

What to do next:

Configure the components of the UploadToXCAR flow:

1. Create the *processType*, *DataFlowClassName*, *DataFlowName*, and *ManifestDetails* properties that are required to configure other rules for data ingestion (a decision table, an activity, and a Parse XML rule). For more information, see [Creating properties to ingest customer data](#).
2. Configure the **Process type?** decision shape in the *UploadToXCAR* flow.

Use this decision shape to select the ingestion data flow to run based on the value of the *processType* [property](#). This property identifies the type of files to be loaded, for example, files that contain customer data for the Account, Subscriber, or Product Holding class. The system obtains the value of the *processType* property from the [manifest file](#). For more information, see [Creating a decision table to ingest customer data](#).

3. Configure the **Execute ingestion** utility shape in the *UploadToXCAR* flow.

Use this utility shape to define an activity to perform the required processing and run the ingestion [data flows](#). For more information, see [Creating an activity to ingest customer data](#).

4. **Optional:** Configure the **Error?** and **Change to ErrorHandling** shapes in the *UploadToXCAR* flow.

You can configure these two shapes to handle errors that might occur while the files are ingested into the target data set. The configuration of these components is not covered in this tutorial.

- [Best practices for data ingestion](#)

Creating properties to ingest customer data

Create properties that are required to configure other rules for data ingestion (a decision table, an activity, and Parse XML rule).

The goal of this procedure is to create the following rules with sample parameter values:

Rule type	Rule name	Apply to	Ruleset	Type
Property	<i>processType</i>	<i>Data-Ingestion-Manifest</i>	<i>UPlusCDH-Rules:01-01-01</i>	Text
Property	<i>DataFlowClassName</i>	<i>UPlus-Work-Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>	Text
Property	<i>DataFlowName</i>	<i>UPlus-Work-Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>	Text
Property	<i>ManifestDetails</i>	<i>UPlus-Work-Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>	Single page

Important: When you configure these rules in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Data Model** category, and then click **Property**.
3. Create the *processType* property:
 - a. On the **Property** tab, click **Create**.
 - b. On the **New** tab, in the **Label** field, enter *processType*.
 - c. In the **Apply to** field, enter *Data-Ingestion-Manifest*.
 - d. In the **Add to ruleset** fields, select **UPlusCDH-Rules** and **01-01-01**.
 - e. In the **Property type** section, leave the **Text** type selected.
 - f. Click **Create and open**.
 - g. Click **Save**.

processType property

Edit Property: processType [Available]

CL Data-Ingestion-Manifest > ID ProcessType > RS UPlusCDH-Rules:01-01-01

Actions Save Close

General Advanced Specifications History

Property type

Text (change)

4. Create the *DataFlowName* property:
 - a. On the **Property** tab, click **Create**.
 - b. On the **New** tab, in the **Label** field, enter *DataFlowName*.
 - c. In the **Apply to** field, enter the name of the ingestion work class, which in this example is *UPlus-Work-Ingestion*.
Note: The system creates the ingestion work class automatically when you create the ingestion case type. To check what the ingestion work class is in your system, in the navigation pane of Dev Studio, click **App**, and then expand the following categories: **Ingestion SysAdmin Class**.
 - d. In the **Add to ruleset** fields, select *UPlusCDH-Rules* and *01-01-01*.
 - e. In the **Property type** section, leave the **Text** type selected.
 - f. Click **Create and open**.
 - g. Click **Save**.

DataFlowName property

5. Create the *DataFlowClassName* property:
 - a. On the **Property** tab, click **Create**.
 - b. On the **New** tab, in the **Label** field, enter *DataFlowClassName*.
 - c. In the **Apply to** field, enter the name of the ingestion work class, which in this example is *UPlus-Work-Ingestion*.
 - d. In the **Add to ruleset** fields, select *UPlusCDH-Rules* and *01-01-01*.
 - e. In the **Property type** section, leave the **Text** type selected.
 - f. Click **Create and open**.
 - g. Click **Save**.

DataFlowClassName property

6. Create the *ManifestDetails* property:

- On the **Property** tab, click **Create**.
- On the **New** tab, in the **Label** field, enter `ManifestDetails`.
- In the **Apply to** field, enter the name of the ingestion work class, which in this example is `UPlus-Work-Ingestion`.
- In the **Add to ruleset** fields, select ***UPlusCDH-Rules*** and ***01-01-01***.
- In the **Property type** section, click **change**, and then select **Single Page**.
- In the **Page definition** field, enter `Data-Ingestion-Manifest`.
- Click **Create and open**.
- Click **Save**.

ManifestDetails property

The image shows two screenshots of the 'Edit Property: ManifestDetails' dialog box. The top screenshot shows the 'General' tab with 'Property type' set to 'Single Page' and 'Page definition' set to 'Data-Ingestion-Manifest'. The bottom screenshot is identical but shows the 'Save' button highlighted.

Creating a decision table to ingest customer data

Create a decision table to define a set of conditions under which the system runs a specific ingestion process using the appropriate data flows, or reports an error.

The goal of this procedure is to complete the following tasks:

- Create the following rule with sample parameter values:

Rule type	Rule name	Ruleset	Class
Decision table	<code>UPlusIngestionProcessType</code>	<code>UPlusCDH-Rules:01-01-01</code>	<code>UPlus-Work-Ingestion</code>

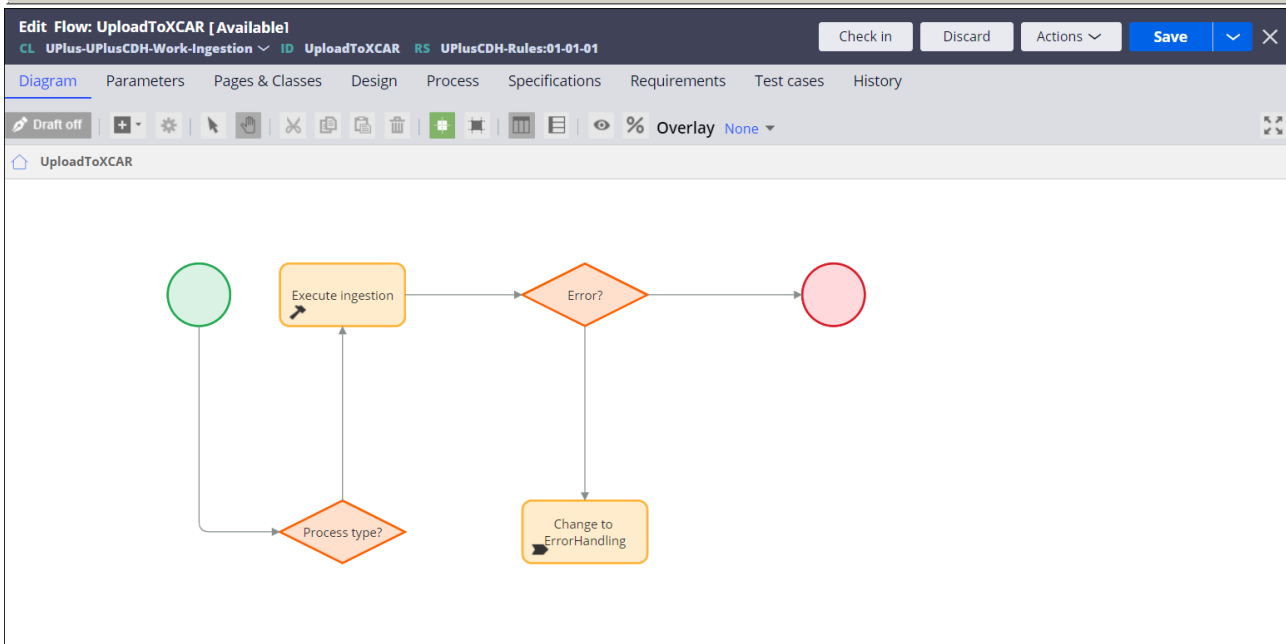
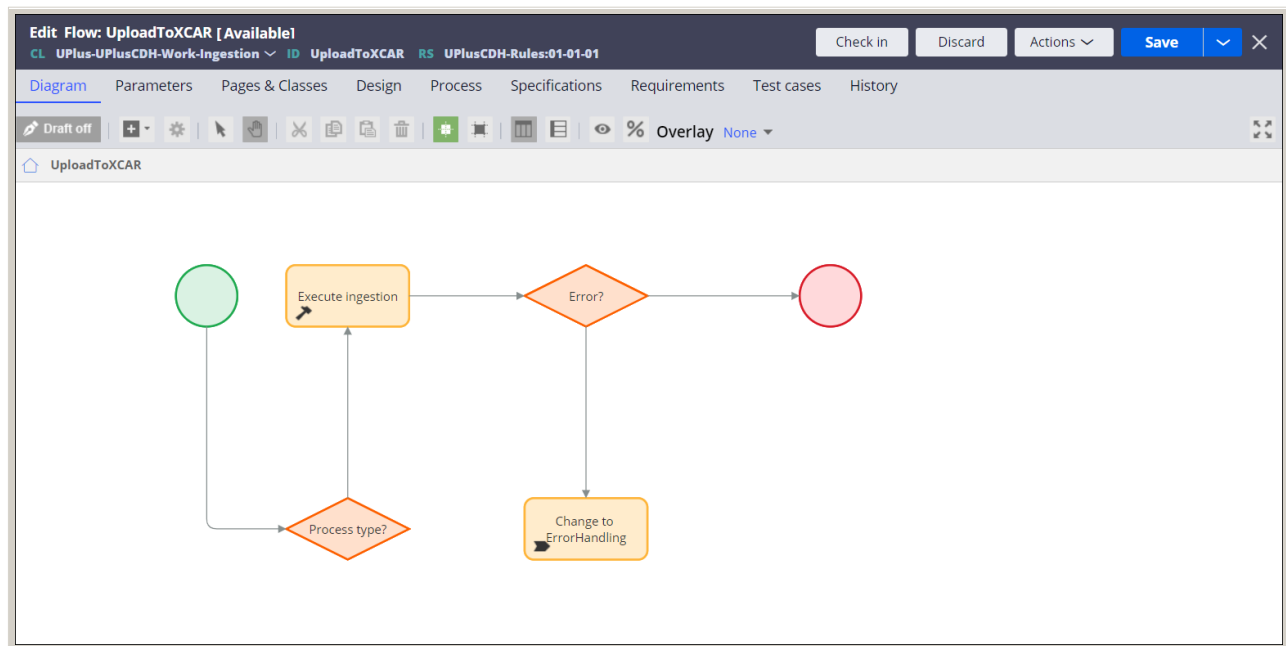
Important: When you configure this rule in your system, use the values that are appropriate for your system.

- Add the rule to the **Process type?** decision shape in the process flow in the *Ingestion* case type.

Before you begin: Create the *Ingestion* case type. For more information, see [Creating a case type to ingest customer data](#).

The starting point for this procedure is the process flow that you added to the *Ingestion* case type, as shown in the following figure:

UploadToXCAR flow



1. On the **Edit Flow: UploadToXCAR** tab, double-click the **Process type?** decision shape.
2. In the **Type** field, select **Decision Table**.
3. Click the **Crosshairs** icon to the right of the **Rule** field.
4. On the **New** tab, configure the following parameters:
 - a. In the **Label** field, enter *UPlusIngestionProcessType*.
 - b. In the **Apply to** and **Add to ruleset** fields, leave the default values.
5. Click **Create and open**.
6. On the **Table** tab, create the following decision table:

Tip: The decision table uses the [data flows](#) and [properties](#) that you created earlier.

	processType	Return	DataFlowClassName	DataFlowName
if	AccountIngestion	NA	UPlusCDH-Data-Comms-Account	UPlusImportAccount
else if	SubscriptionIngestion	NA	UPlusCDH-Data-Comms-Subscription	UPlusImportSubscription
else if	HoldingIngestion	NA	UPlusCDH-Data-Comms-Holding	UPlusImportHolding
otherwise		NA	Error	Error

Decision table for selecting the ingestion process

Decision properties

Decision:

Automatically evaluate a business decision to determine how this case progresses.

Details

Provide a decision to be made in the current process

Type

Rule*

Result in

Audit note

☐ Enable navigation link

10. Click **Submit**.
11. On the **Edit Flow: UploadToXCAR** tab, click **Save**

Result: You have configured the first decision shape. You can now configure the utility shape by creating an activity to ingest customer data.

Creating an activity to ingest customer data

Create an activity to define a sequential set of steps to perform the required processing and run the ingestion data flows.

The goal of this procedure is to complete the following tasks:

1. Create the following rule with sample parameter values:

Rule type	Rule name	Ruleset	Class
Activity	<i>ExecuteIngestionDF</i>	<i>UPlusCDH-Rules:01-01-01</i>	<i>UPlus-Work-Ingestion</i>

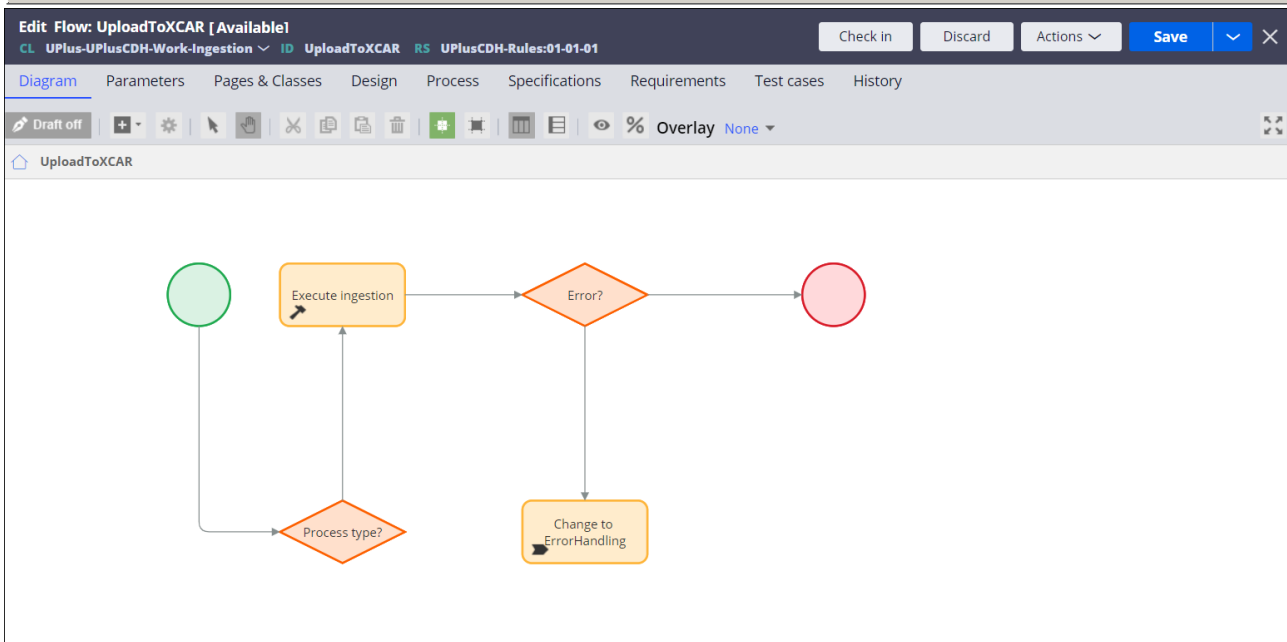
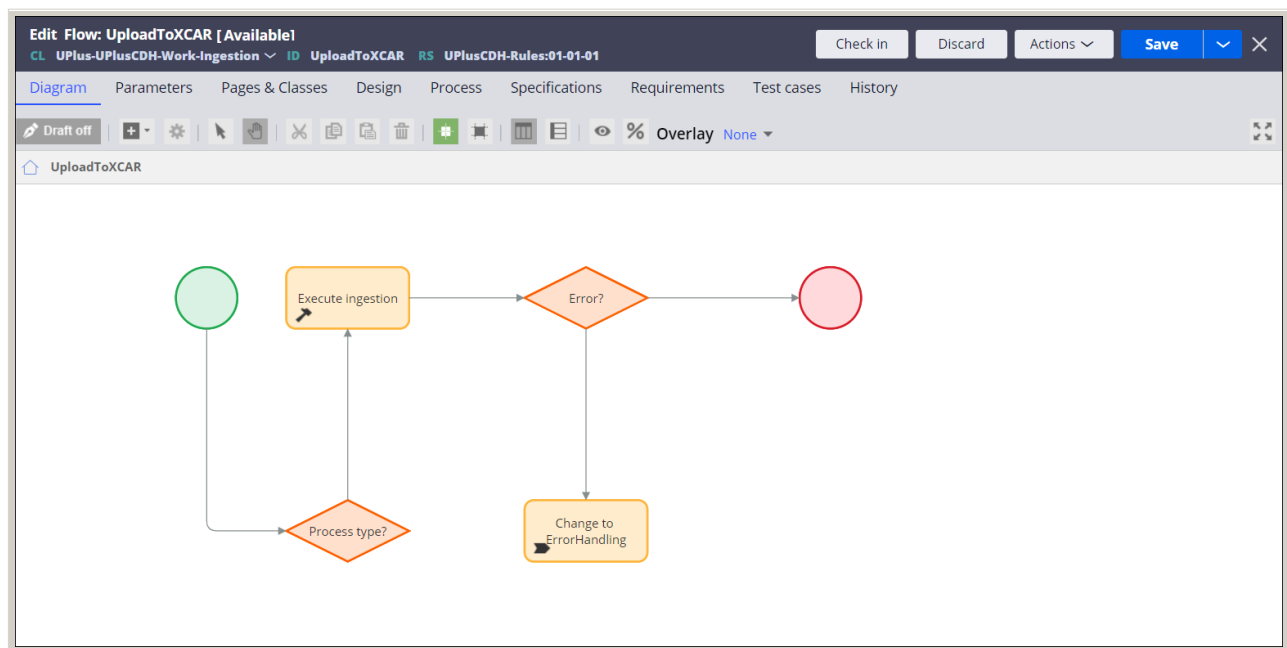
Important: When you configure this rule in your system, use the values that are appropriate for your system.

2. Add the rule to the **Execute ingestion** utility shape in the process flow in the *Ingestion* case type.

Before you begin: Create the *Ingestion* case type. For more information, see [Creating a case type to ingest customer data](#).

The starting point for this procedure is the process flow that you added to the *Ingestion* case type, as shown in the following figure:

UploadToXCAR flow



1. On the **Edit Flow: UploadToXCAR** tab, double-click the **Execute ingestion** utility shape.
2. Clear the **Rule** field.
3. Click the **Crosshairs** icon to the right of the **Rule** field.
4. On the **New** tab, configure the following parameters:
 - a. In the **Label** field, enter *ExecuteIngestionDF*.
 - b. In the **Apply to** and **Add to ruleset** fields, leave the default values.
5. Click **Create and open**.
6. Click the **Pages & Classes** tab.
7. Add the *RunOptionsPage* with the following parameters:
 - a. In the **Page name** field, enter *RunOptionsPage*.
 - b. In the **Class** field, enter *Data-Decision-DDF-RunOptions*.
8. Add the *Progress* page with the following parameters:
 - a. Click the **Add item** icon to add a new row.
 - b. In the **Page name** field, enter *Progress*.
 - c. In the **Class** field, enter *Data-Decision-DDF-Progress*.

RunOptionsPage and Progress page configuration

Edit Activity: ExecutelIngestionDF [Available]
 CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps Parameters **Pages & Classes** Security Specifications History

Page name	Class
RunOptionsPage	Data-Decision-DDF-RunOptions
Progress	Data-Decision-DDF-Progress

+

Edit Activity: ExecutelIngestionDF [Available]
 CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps Parameters **Pages & Classes** Security Specifications History

Page name	Class
RunOptionsPage	Data-Decision-DDF-RunOptions
Progress	Data-Decision-DDF-Progress

+

9. Click the **Parameters** tab.
10. In the **Parameters** section, add a parameter with the following settings:
 - a. In the **Name** field, enter sleepTime.
 - b. In the **Data type** field, select **Integer**.
 - c. In the **Required**, ensure that **No** is selected.
 - d. In the **In/Out**, ensure that **In** is selected.

SleepTime parameter configuration

Edit Activity: ExecutelIngestionDF [Available]
 CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps **Parameters** Pages & Classes Security Specifications History

Parameters

Name	Description	Data type	Required	In/Out	Default value	Smartprompt type	Validate as
1 sleepTime		Integer	No	In			

+

Edit Activity: ExecutelIngestionDF [Available]
 CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps **Parameters** Pages & Classes Security Specifications History

Parameters

Name	Description	Data type	Required	In/Out	Default value	Smartprompt type	Validate as
1 sleepTime		Integer	No	In			

+

11. In the **Local variables** section, add a variable with the following settings:
 - a. In the **Name** field, enter dfstatus.
 - b. In the **Data type** field, ensure that **String** is selected.

dfstatus parameter configuration

Edit Activity: ExecutelIngestionDF [Available]
 CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps **Parameters** Pages & Classes Security Specifications History

Local variables

Name	Description	Data type
1 dfstatus		String

Edit Activity: ExecutelngestionDF [Available]

CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelngestionDF RS: UPlusCDH-Rules:01-01-01

Steps Parameters Pages & Classes Security Specifications History

Local variables

Name	Description	Data type
1 dfstatus		String

12. On the **Steps** tab, configure the following steps:

Tip: The activity uses the *DataFlowClassName* and *DataFlowName* [properties](#) that you configured earlier.

Step number	Method	Step page	PropertiesName	PropertiesValue
1	Page-New	RunOptionsPage		
2	Property-Set		param.sleepTime	30
			RunOptionsPage.pyAppliesTo	.DataFlowClassName
			RunOptionsPage.pyRuleName	.DataFlowName
3	Call pxRunDDFWithProgressPage	RunOptionsPage		
4	Property-Set	Progress	local.dfstatus	@(PegaMKT-Engine:PegaMKTUtilities).monitorDFProgress(tools,param.sleepTime)
			Param.DataFlowID	Progress.pyWorkItemID
5	Page-Remove	RunOptionsPage		

Steps configuration in the ExecutelngestionDF activity

Edit Activity: ExecutelngestionDF [Available]

CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecutelngestionDF RS: UPlusCDH-Rules:01-01-01

Steps Parameters Pages & Classes Security Specifications History

Label	Method	Step page	Description
1. ☐ Loop When >	Page-New	RunOptionsPage	Create a page
2. ☐ Loop When >	Property-Set		Set property values

Method Parameters

*PropertiesName	*PropertiesValue
param.sleepTime	30
RunOptionsPage.pyAppliesTo	.DataFlowClassName
RunOptionsPage.pyRuleName	.DataFlowName

3. ☐ Loop When >	Call pxRunDDFWithProgress	RunOptionsPage	Call activity pxRunDDFWithProgressPage
4. ☐ Loop When >	Property-Set	Progress	Set property values

Method Parameters

*PropertiesName	*PropertiesValue
local.dfstatus	@(PegaMKT-Engine:PegaMKTUtilities).monitorDFProgress(tools,param.sleepTime)
Param.DataFlowID	Progress.pyWorkItemID

5. ☐ Loop When >	Page-Remove	RunOptionsPage	Remove page(s)
---	-------------	----------------	----------------

+ Add a step Collapse all steps

Edit Activity: ExecuteIngestionDF [Available] Actions Save ✕

CL: UPlus-UPlusCDH-Work-Ingestion ID: ExecuteIngestionDF RS: UPlusCDH-Rules:01-01-01

Steps Parameters Pages & Classes Security Specifications History

	Label	Method	Step page	Description	
1.	☐ Loop When >	Page-New	RunOptionsPage	Create a page	Jump
2.	☐ Loop When v	Property-Set		Set property values	Jump

Method Parameters ?

*PropertyName	*PropertiesValue	
param.sleepTime	30	
RunOptionsPage.pyAppliesTo	.DataFlowClassName	
RunOptionsPage.pyRuleName	.DataFlowName	

+

3.	☐ Loop When >	Call pxRunDDFWithProgress	RunOptionsPage	Call activity pxRunDDFWithProgressPage	Jump
4.	☐ Loop When v	Property-Set	Progress	Set property values	Jump

Method Parameters ?

*PropertyName	*PropertiesValue	
local.dfstatus	@(PegaMKT-Engine:PegaMKTUtilities).monitorDFProgress(tools,param.sleepTime)	
Param.DataFlowID	Progress.pyWorkItemID	

+

5.	☐ Loop When >	Page-Remove	RunOptionsPage	Remove page(s)	Jump
----	--------------------------------------	-------------	----------------	----------------	------

+ Add a step Collapse all steps

13. Click **Save**.
14. Go back to the **UploadToXCAR** tab.
15. In the **Rule** field, select the **ExecuteIngestionDF** activity that you created.

Adding the activity rule to the utility shape

Utility properties

Utility:

Perform automated processing in your case.

Automation details
 Define automation

Select type of rule

☒ Activity
☐ Automation

Rule*

Audit note

☐ Enable navigation link

Cancel Submit

Utility properties

Utility:

Perform automated processing in your case.

Automation details
Define automation

Select type of rule

☒ Activity
☐ Automation

Rule *

Audit note

☐ Enable navigation link

Cancel
Submit

- Click **Submit**.
- On the **Edit Flow: UploadToXCAR** tab, click **Save**.

Creating a Parse XML rule to ingest customer data

Create a Parse XML rule to read the manifest file and map the nodes in the manifest file to properties in Pega Platform.

The goal of this procedure is to create the following rule with sample parameter values:

Rule type	Rule name	Apply to	Ruleset
Parse XML	<i>UPlusManifestParser</i>	<i>UPlus-Work-Ingestion</i>	<i>UPlusCDH-Rules:01-01-01</i>

Important: When you configure this rule in your system, use the values that are appropriate for your system.

- In the navigation pane of Dev Studio, click **Records**.
- Expand the **Integration-Mapping** category, and then click **Parse XML**.
- On the **Parse XML** tab, click **Create**.
- On the **New** tab, configure the following parameters:
 - In the **Label** field, enter *UPlusManifestParser*.
 - In the **Root Element Name** field, enter *manifest*.
Note: *manifest* is the root element of the [manifest file](#) to be parsed by this Parse XML rule.
 - In the **Apply to** field, enter the name of the ingestion work class, which in this example is *UPlus-Work-Ingestion*.
Note: The system creates the ingestion work class automatically when you create the ingestion case type. To check what the ingestion work class is in your system, in the navigation pane of Dev Studio, click **App**, and then expand the following categories: **Ingestion SysAdmin Class** .
 - In the **Add to ruleset** fields, select *UPlusCDH-Rules* and *01-01-01*.
- Click **Create and open**.
- On the **Mapping** tab, configure a mapping that matches the structure of your [manifest file](#):
Note: The mapping uses the *processType* and *ManifestDetails* [properties](#) that you configured earlier.

Node Name	Context Page	Context Class	Property	Node Namespace
manifest	.ManifestDetails	Data-Ingestion-Manifest		UPlusManifestParser
processType			.processType	
totalRecordCount			.pyTotalRecordCount	

Node Name	Context Page	Context Class	Property	Node Namespace
recordCount		Embed-Data-FileDetails	.pyRecordCount	
name			.pyName	
size			.pySize	

Parse XML rule for processing the manifest file

The interface shows a configuration window for 'UPlusManifestParser'. It includes tabs for 'Mapping', 'Pages & Classes', 'XML', 'Specifications', and 'History'. A tree view on the left lists the following nodes and their corresponding properties:

- manifest .ManifestDetails
 - processType .processType
 - totalRecordCount .pyTotalRecordCount
- files
 - file (.pyFilesList)
 - name .pyName
 - size .pySize
 - recordCount .pyRecordCount

7. Click **Save**.

Creating a service package to ingest customer data

Create a service package to group service rules that are designed to be developed, tested, and deployed together. The service package determines the security scheme and access for the services in the package.

The goal of this procedure is to create the following rule with sample parameter values:

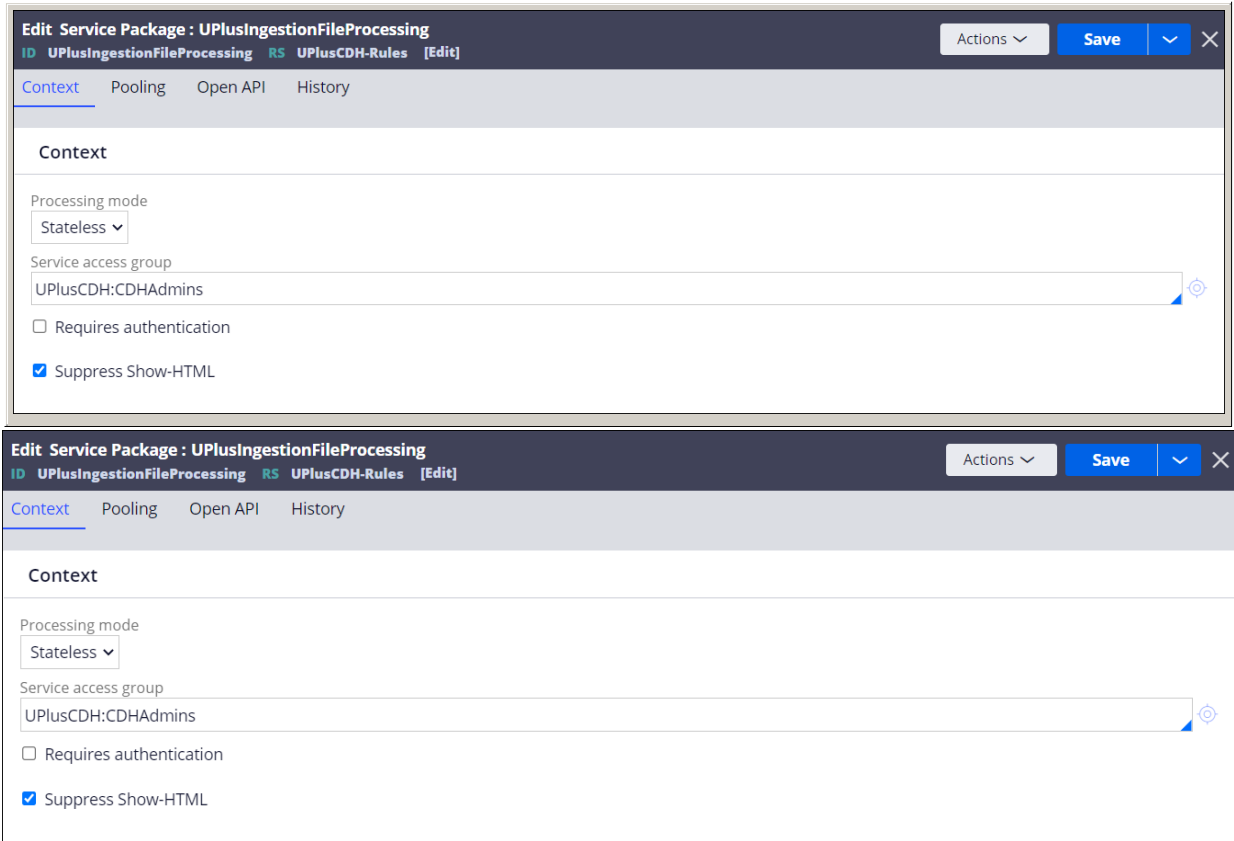
Rule type	Rule name	Ruleset
Service package	<i>UPlusIngestionFileProcessing</i>	<i>UPlusCDH-Rules</i>

Important: When you configure this rule in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Integration-Resources** category, and then click **Service Package**.
3. On the **New** tab, configure the following parameters:
 - a. In the **Short description** field, enter *UPlusIngestionFileProcessing*.
 - b. In the **Service Package Name** field, enter *UPlusIngestionFileProcessing*.

- 4. Click **Create and open**.
- 5. In the **Service access group** field, enter *UPlusCDH:CDHAdmins*.
- 6. Clear the **Requires authentication** checkbox.
- 7. Click **Edit** to edit the associated ruleset, and then change the ruleset to *UPlusCDH-Rules*.

Service package configuration for data ingestion



- 8. Click **Save**.

Creating a service file to ingest customer data

Create a service file to define a package of rules for processing the manifest file.

The goal of this procedure is to create the following rule with sample parameter values:

Rule type	Rule name	Ruleset	Work class
Service file	<i>UPlusProcessManifestFile</i>	<i>UPlusCDH-Rules:01-01-01</i>	<i>UPlus-Work-Ingestion</i>

Important: When you configure this rule in your system, use the values that are appropriate for your system.

- 1. In the navigation pane of Dev Studio, click **Records**.
- 2. Expand the **Integration-Services** category, and then click **Service File**.
- 3. On the **Service File** tab, click **Create**.
- 4. On the **New** tab, configure the following parameters:
 - a. In the **Label** field, enter *UPlusProcessManifestFile*.
 - b. In the **Customer Package Name** field, enter the name of the [service package](#) that you created earlier, which in this example was *UPlusIngestionFileProcessing*.
 - c. In the **Customer Class Name** field, enter Default.
 - d. In the **Add to ruleset** fields, select *UPlusCDH-Rules* and *01-01-01*.
- 5. Click **Create and open**.
- 6. On the **Service** tab, in the **Primary page class** field, enter the name of the ingestion work class, which in this example is *UPlus-Work-Ingestion*.
Note: The system creates the ingestion work class automatically when you create the ingestion case type. To check what the ingestion work class is in your system, in the navigation pane of Dev Studio, click **App**, and then expand the following categories: **Ingestion SysAdmin Class**.
- 7. Click the **Method** tab, and then in the **Processing method** field, select **file at a time**.
- 8. Click the **Request** tab, and then configure the following parameters:
 - a. In the **Map to** field, enter XML ParseRule.
 - b. In the **Map to key** field, enter the name of the [Parse XML rule](#) that you configured earlier, followed by the name of the root element of your [manifest file](#), which in this example is *UPlusManifestParser manifest*.
 - c. In the **Final activity** field, enter svcAddWorkObject.

Service file configuration for data ingestion

Edit Service File: UPlusProcessManifestFile [Available]
ID UPlusIngestionFileProcessing • Default • UPlusProcessManifestFile RS UPlusCDH-Rules:01-01-01
Actions Save

Service Method Request Specifications History

Processing prolog

Initial activity

Parse segments

Record type	Once?	Map to	Map to key	Auth?	Activity	Keep page?
1	☐	XML ParseRule	UPlusManifestParser ma	☐		☐

Processing epilog

Final activity

svcAddWorkObject

Edit Service File: UPlusProcessManifestFile [Available]
ID UPlusIngestionFileProcessing • Default • UPlusProcessManifestFile RS UPlusCDH-Rules:01-01-01
Actions Save

Service Method Request Specifications History

Processing prolog

Initial activity

Parse segments

Record type	Once?	Map to	Map to key	Auth?	Activity	Keep page?
1	☐	XML ParseRule	UPlusManifestParser ma	☐		☐

Processing epilog

Final activity

svcAddWorkObject

9. Click **Save**.
10. In the **Processing epilog** section, configure the following parameters:
 - a. In the **FlowType** field, enter pyStartCase.
 - b. In the **organization** field, enter UPlus.
 - c. In the **workPage** field, enter pyWorkPage.

Processing epilog configuration

Processing epilog

Final activity

svcAddWorkObject

Parameters

FlowType ? pyStartCase
organization ? UPlus
workPage ? pyWorkPage
CoverHandle ?
returnNextActionInfo ? ☐
checkFieldSecurity ? ☐
pageInstructions ?
SkipCreateView ? ☒

Processing epilog

Final activity
svcAddWorkObject

Parameters

FlowType	pyStartCase
organization	UPlus
workPage	pyWorkPage
CoverHandle	
returnNextActionInfo	☐
checkFieldSecurity	☐
pageInstructions	
SkipCreateView	☒

11. Select the **SkipCreateView** checkbox.
12. Click **Save**.

Creating file listeners to ingest customer data

Create file listeners to listen for the arrival of the manifest file in your repository, run a preprocessing check to validate the files against the manifest file, and if the check is successful, start the ingestion process.

The goal of this procedure is to create the following rules with sample parameter values:

Rule type	Rule name	Ruleset
File listener	<i>UPlusAccountFileListener</i>	<i>UPlusCDH-Rules</i>
File listener	<i>UPlusSubscriptionFileListener</i>	<i>UPlusCDH-Rules</i>
File listener	<i>UPlusHoldingFileListener</i>	<i>UPlusCDH-Rules</i>

Important: When you configure these rules in your system, use the values that are appropriate for your system.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Integration-Resources** category, and then click **File Listener**.
3. On the **File Listener** tab, click **Create**.
4. On the **New** tab, configure the following parameters:
 - a. In the **Short description** field, enter a meaningful description of the file listener, for example, *File listener for Account class*.
 - b. In the **Listener Name** field, enter a name for the file listener, for example, *UPlusAccountFileListener*.
5. Click **Create and open**.
6. In the **Listener nodes** section, configure the following parameters:
 - a. In the **Startup option** field, select **NodeClassification based startup**.
 - b. In the **Classifications** section, click the **Add a row** icon to add a row.
 - c. In the **Node Type** field, enter *BackgroundProcessing*.

Listener nodes configuration

Listener nodes

☐ Block startup

Startup option
NodeClassification based startup

Classifications

Node Type	
1	BackgroundProcessing

+

Listener nodes

☐ Block startup

Startup option
NodeClassification based startup ▾

▼ **Classifications**

Node Type	
1	BackgroundProcessing 

+

7. In the **Source properties** section, in the **Source location** field, enter the path to the folder in your repository where the files to be ingested will be stored.
8. In the **Source name mask** field, enter the source name mask for the file listener.
 - For *UPlusAccountFileListener*, enter *account_manifest*.xml*.
 - For *UPlusSubscriptionFileListener*, enter *subscription_manifest*.xml*.
 - For *UPlusHoldingFileListener*, enter *product_holding_manifest*.xml*.

Source properties configuration

Source properties

Source location ★
file://pegacloudrepository:/csvtransfer

Source name mask ★
account_manifest*.xml

☒ Disable case sensitivity

The character cases of the location and mask will be ignored.

Source properties

Source location ★
file://pegacloudrepository:/csvtransfer

Source name mask ★
account_manifest*.xml

☒ Disable case sensitivity

The character cases of the location and mask will be ignored.

9. Select the **Disable case sensitivity** checkbox.
10. In the **Listener properties** section, configure the following parameters:
 - a. In the **Concurrent threads** field, enter 1.
 - b. In the **Service package** field, select the name of the [service package](#) that you created earlier, which in this example was **UPlusIngestionFileProcessing**.
 - c. In the **Service class** field, select **Default**.
 - d. In the **Service method** field, select the name of the [service file](#) that you configured earlier, which in this example was **UPlusProcessManifestFile**.

Listener properties configuration

Listener properties

Concurrent threads
1

Service package
UPlusIngestionFileProcessing ▾

Service class
Default ▾

Service method
UPlusProcessManifestFile ▾ 

[Test connectivity](#)

Listener properties

Concurrent threads
1

Service package
UPlusIngestionFileProcessing ▾

Service class
Default ▾

Service method
UPlusProcessManifestFile ▾ 

[Test connectivity](#)

11. Click the **Process** tab.
12. In the **Cleanup** section, in the **File operation** field, select **Keep**.
13. Click **Save**.
14. Repeat steps 3 through 13 to create the other two file listeners:
 - *UPlusSubscriptionFileListener*
 - *UPlusHoldingFileListener*

Starting file listeners to ingest customer data

Start the file listeners to enable the ingestion process in your system. File listeners listen for the arrival of a manifest file in your repository, validate the files to be transferred against the manifest file, and if the validation is successful, start the ingestion of files.

1. Switch to Admin Studio.
2. In the navigation pane of Admin Studio, click **Resources Listeners**.
3. In the **Start / restart listener** field, select **Start new instance**.
4. In the field below, enter the name of a [file listener](#) that you configured earlier, for example, *UPlusAccountFileListener*.
5. Click **Apply**.

Starting a new instance

6. Verify that the new listener instance appears in the **Active listeners** list without any errors.

Active listeners

Active listeners										Restart	Stop
☐	Name	Instance	Type	Last access	Node	Requests	Errors	Status	Actions		
☐	UPlusAccountFileListener	FILE-Thread-66	File Listener	6/4/21 4:02 AM	crmmarketingenablement_i_0f45ea	0	0	Sleeping	Actions		
☐	UPlusAccountFileListener	FILE-Thread-67	File Listener	6/4/21 4:02 AM	crmmarketingenablement_i_0f45ea	0	0	Sleeping	Actions		

Active listeners										Restart	Stop
☐	Name	Instance	Type	Last access	Node	Requests	Errors	Status	Actions		
☐	UPlusAccountFileListener	FILE-Thread-66	File Listener	6/4/21 4:02 AM	crmmarketingenablement_i_0f45ea	0	0	Sleeping	Actions		
☐	UPlusAccountFileListener	FILE-Thread-67	File Listener	6/4/21 4:02 AM	crmmarketingenablement_i_0f45ea	0	0	Sleeping	Actions		

7. Switch to Dev Studio.
8. In the header of Dev Studio, click **Configure Decisioning Decisions Data Flows Batch Processing**.
9. Verify that your file listener successfully triggered the data flow and processed the files from the SFTP server.
10. Repeat steps 1 through 9 to start the other two file listeners:
 - *UPlusSubscriptionFileListener*
 - *UPlusHoldingFileListener*

Exporting customer data from Pega Customer Decision Hub

Organizations frequently need to export data such as transactional data, analytical and monitoring data, or interaction history results out of Pega Customer Decision Hub for further processing.

For example, the results of next-best-action decisions are recorded in Pega Customer Decision Hub as interaction history for each customer. You can export interaction history data out of Pega Customer Decision Hub and use it for reporting in external business intelligence tools that create insight beyond the scope of the Pega application.

Understanding the export process

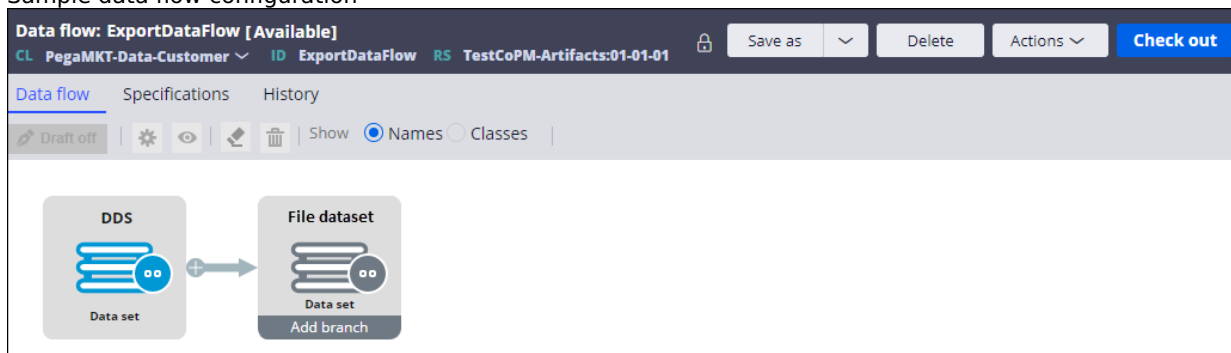
You can define a data flow that exports interaction history records into a File data set. By running the data flow, you can export the data out of Pega Customer Decision Hub and into a specified repository.

Configuring the export process

To export the data associated with daily interactions into a Pega Cloud File Storage and use this data for additional analysis, configure a data flow that exports the interaction history data into a File data set.

1. Create a file data set with a file path that leads to the location in which you want to export the interaction files. For more information, see [Creating a File data set record for files on repositories](#).
2. Create a data flow with the data set that you want to export as the source and the file data set that you created in step 1 as the destination.

Sample data flow configuration



For more information, see [Creating a data flow](#).

3. Run the data flow.

For more information, see [Triggering the customer data ingestion process](#).

Result: During each run, the data flow exports interaction history data into files in the specified repository. The file names include a date and time stamp, as configured in the file data set. Additionally, a .meta file is created that contains the metadata of each file saved in the repository, as in the following example:

```
<?xml version='1.0' encoding='UTF-8'?>
<file>
  <name>records_2020-02-21_002003_00000.json</name>
  <size>28773955</size>
  <recordCount>11000</recordCount>
</file>
```

Exporting interaction history data from the relational database

You can export interaction history data from Pega Customer Decision Hub into files in the specified repository. You can then send the exported interaction history files to downstream operational, reporting, and analytics systems and combine it with other data to provide more comprehensive reporting.

In on-premise implementations, you can do a straight database (DB) extraction or DB-to-DB transfers. However, these methods are not available for systems implemented in Pega Cloud.

The alternative method described in this article relies on data flows and a file data set. It is suitable for all deployment types, including systems that run in Pega Cloud.

Prerequisites

To export interaction history data to files in the repository, your system must be on Pega Platform 8.3.3 or later.

Interaction history structure

Pega Customer Decision Hub stores interaction history data in the relational database. The data is divided into classes that represent the Fact table and the associated dimensions. Because of this structure, exporting interaction history requires the creation of a class, in which you can combine the Fact and dimension data before exporting it to a file. For more information about the interaction history tables and classes, see [Pega Customer Decision Hub Interaction History data model](#).

Designing the export process

To start exporting interaction history data into files in the repository, you need to create several artifacts:

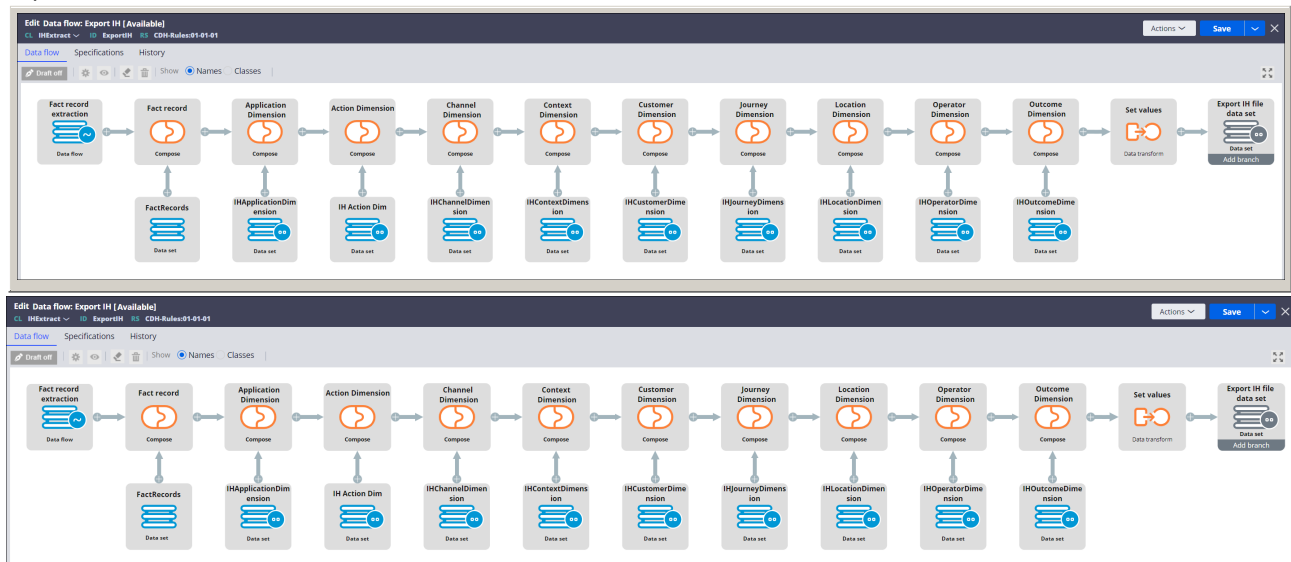
- A [class](#) that you can use to combine the Fact and Dimension data. Let us call it the *IH Extract* class.
- The primary [data flow](#) that combines all the data for a single interaction history record in the *IH Extract* class, and then exports this data into a file in the repository (see example below).
- The secondary data flow that retrieves the Fact ID and Dimension record IDs for the desired record, and then feeds them into the primary data flow (see example below).
- A [file data set](#) that is used as the destination for the primary data flow. This file data set specifies a file path that points to a folder in the repository in which the exported files are saved. The file data set also defines the parameters of the exported files, such as their format (CSV or JSON), date and time settings, and the property mapping.
- **Optional:** If you want to run the export data flow on a regular basis, you need to create two additional artifacts:
 - A [wrapping activity](#) that calls the export data flow.
 - A [job scheduler](#) that runs the wrapping activity according to the specified schedule.

Data flow examples

The *Export IH* data flow in the following figure is an example of the primary data flow that you can create to export interaction history data. For input, the primary data flow uses the secondary data flow, from which it retrieves a list of Fact ID and Dimension record IDs. Next, it combines the Fact ID and Dimension record IDs with relevant records retrieved from data sets in the Fact and Dimension classes, and adds them to the *IH extract* class. Finally, the data flow exports the relevant records into a file, using the file data set.

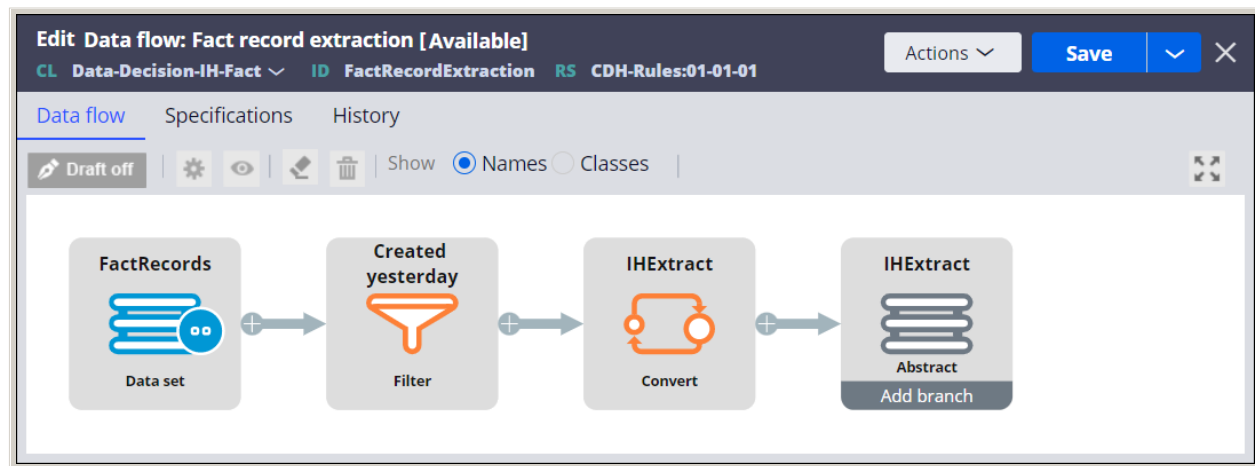
Note: To view the full-sized figure, double-click the image, and open it in a new tab.

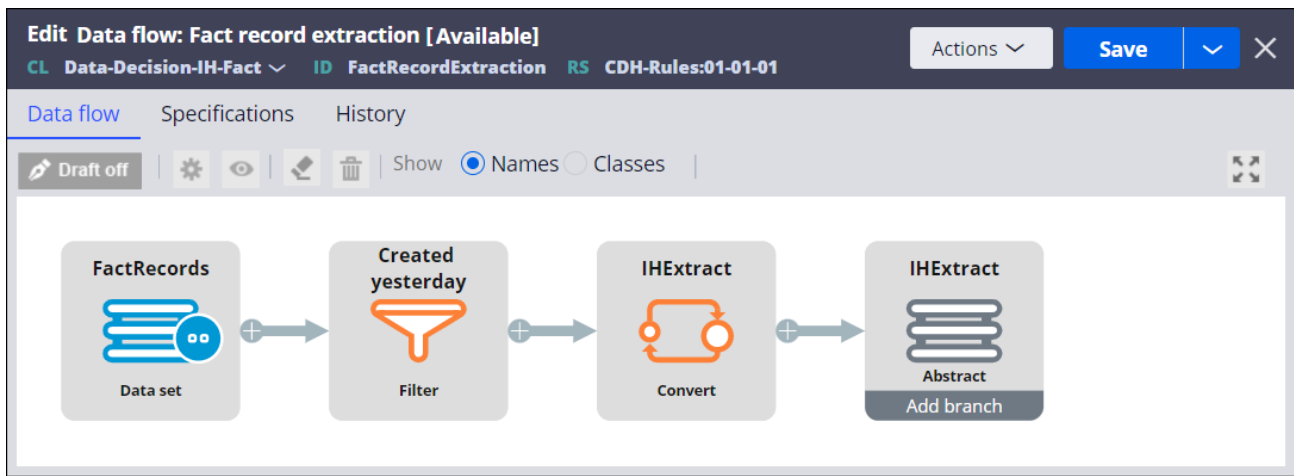
Export IH data flow



The secondary data flow (*Fact record extraction* in the following example) contains a filter that selects the records that were created on the previous day. The data flow uses a data set as the source, but you can also use a report definition or even an abstract source, depending on the desired outcome. The output of the data flow belongs to the *IH Extract* class and consists of all the Fact and Dimension IDs. The primary data flow (*Export IH*) uses this data to pull the relevant interaction history records.

Fact record extraction data flow





Final considerations

- You can transfer the exported file to a downstream system using a file listener and a connect-ftp rule.
- You can add more properties to the extracted file by changing the mapping in the file data set.
 - You can add any existing interaction history properties (both Fact and Dimension) right away.
 - To add new properties, you first need to add them to interaction history by following the standard process, and then you can add them to the file data set for export.
- Note:** Downstream systems may require changes to ingest the new properties.
- In the *Export IH* data flow, you can add simple data transformations using a data transform, just before the file data set output.
- You can run the export manually for different purposes, for example, to do a full extract, extract a specific action, or a specific campaign run. To do that, follow these steps:
 - Modify the secondary data flow, for example, by adding or removing filters.
 - Change the file data set destination to prevent interfering with scheduled data transfers.
 - On the primary data flow page, click **Actions Run**.

Recommended: Materializing interaction history summaries

Pega Customer Decision Hub contains interaction history summaries, which are shipped without preaggregated data. To save processing time, turn on preaggregation for each interaction history summary by materializing them.

Preaggregated data sets save processing time because they include the latest interactions. Data sets that are not preaggregated do not include the latest interactions and therefore they query the database. **Note:** For more information about interaction history and its data model, see [Pega Customer Decision Hub Interaction History data model](#).

- Log in as an operator with access to the **Summaries** landing page. Access to the Summaries landing page is limited to users whose access group includes the CustomerDecisionHub:DataAdmin user role. For more information about access roles, see [Defining access roles and privileges](#).
- In App Studio, click **Data Summaries**.
- On the Summaries landing page, find the data set for which you want to turn on preaggregation, and click **More** : **Materialized**.

Applying industry-specific accelerators

Accelerate your Pega Customer Decision Hub implementation by applying industry-specific components for communications, financial services, healthcare, or insurance. Through the use of components, you can quickly configure your customer analytical data model with business-friendly properties and define a typical industry configuration for Next-Best-Action Designer.

Applying communications accelerators

Speed up your implementation of Pega Customer Decision Hub for the communications industry and get more value from your investment in Pega Customer Decision Hub by applying accelerators. Accelerators help communications providers go live with their applications in a shorter time and provide guidance on best-in-class Pega practices.

The following accelerators are available for Pega Customer Decision Hub implementations for the communications industry:

[Communications Customer Analytical Data Model](#)

The communications customer analytical data model in the form of an extended Customer Analytics Record (xCAR) is a multilevel, structured record of the customer data in a format that is optimized for performance, real-time analytics, and one-to-one marketing. The xCAR provides a basis for faster, more accurate business decisions and for more attractive offers that are better tailored to the customers' needs.

[Pega Next-Best-Action Advisor connector for Pega Customer Decision Hub](#)

The connector integrates Pega Customer Decision Hub with Pega Next-Best-Action Advisor. This component is required if you want to use your Pega Customer Decision Hub application with Pega Next-Best-Action Advisor, Pega Customer Service for Communications, or both.

[Next-Best-Action Designer Template for Pega Next-Best-Action Advisor](#)

The Next-Best-Action Designer template defines a standard configuration to accelerate your implementation of Pega Next-Best-Action Advisor with Pega Customer Decision Hub. The template includes a typical communications-based hierarchy of groups and issues and engagement policies that you can customize to ensure that your customers receive the right actions in a timely manner. The template was designed by using the knowledge and experience gained from successful implementations of Pega Customer Decision Hub in agent-assisted channels for communications providers.

Before you begin: Determine which accelerators you need by reviewing the following implementation scenarios:

Common implementation scenarios for communications

SCENARIO	IMPLEMENTATION WITH			REQUIRED COMPONENTS		
	Pega Customer Service for Communications	Pega Next-Best-Action Advisor	Data model for communications	Communications Customer Analytical Data Model (xCAR)	Pega Next-Best-Action Advisor Connector for Pega Customer Decision Hub	Next-Best-Action Designer Template for Pega Next-Best-Action Advisor
1	NO	NO	NO	NOT REQUIRED	NOT REQUIRED	NOT REQUIRED
2	✓	NO	NO	NOT REQUIRED	✓	NOT REQUIRED
3	NO	✓	NO	NOT REQUIRED	✓	✓
4	✓	✓	NO	NOT REQUIRED	✓	✓
5	NO	NO	✓	✓	NOT REQUIRED	NOT REQUIRED
6	✓	NO	✓	✓	✓	NOT REQUIRED
7	NO	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓

SCENARIO	IMPLEMENTATION WITH			REQUIRED COMPONENTS		
	Pega Customer Service for Communications	Pega Next-Best-Action Advisor	Data model for communications	Communications Customer Analytical Data Model (xCAR)	Pega Next-Best-Action Advisor Connector for Pega Customer Decision Hub	Next-Best-Action Designer Template for Pega Next-Best-Action Advisor
1	NO	NO	NO	NOT REQUIRED	NOT REQUIRED	NOT REQUIRED
2	✓	NO	NO	NOT REQUIRED	✓	NOT REQUIRED
3	NO	✓	NO	NOT REQUIRED	✓	✓
4	✓	✓	NO	NOT REQUIRED	✓	✓
5	NO	NO	✓	✓	NOT REQUIRED	NOT REQUIRED
6	✓	NO	✓	✓	✓	NOT REQUIRED
7	NO	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓

1. Go to [Pega Marketplace](#) and download the components that you want to add to your application.
2. Install the components by following the instructions that are provided with each component.

Applying financial services accelerators

Speed up your implementation of Pega Customer Decision Hub for financial services and get more value from your

investment in Pega Customer Decision Hub by applying accelerators. Accelerators help financial institutions go live with their applications in a shorter time and provide guidance on best-in-class Pega practices.

The following accelerators are available for Pega Customer Decision Hub implementations for financial services:

[Financial Services Customer Analytical Data Model](#)

This component holds the data model for financial services in the form of an extended Customer Analytics Record (xCAR) that includes data entities and attributes to support banking use cases.

[Next-Best-Action Designer Template for Retail Banking](#)

Next-Best-Action Designer template defines a standard configuration to accelerate your implementation of Next-Best-Action Designer and Pega Customer Decision Hub in retail banking.

1. Go to [Pega Marketplace](#) and download the components that you want to add to your application.
2. Install the components by following the instructions that are provided with each component.

Applying healthcare accelerators

Speed up your implementation of Pega Customer Decision Hub for healthcare services and get more value from your investment in Pega Customer Decision Hub by applying accelerators. Accelerators help healthcare institutions go live with their applications in a shorter time and provide guidance on best-in-class Pega practices.

The following accelerators are available for Pega Customer Decision Hub implementations for the healthcare industry:

[Healthcare Customer Analytical Data Model](#)

This component holds the healthcare customer analytical data model (xCAR) that includes data entities and attributes to support healthcare use cases.

[Next-Best-Action Designer Template for Healthcare](#)

The Next-Best-Action Designer template defines a standard configuration to accelerate your implementation of Next-Best-Action Designer and Pega Customer Decision Hub for healthcare services.

1. Go to [Pega Marketplace](#) and download the components that you want to add to your application.
2. Install the components by following the instructions that are provided with each component.

Applying insurance accelerators

Speed up your implementation of Pega Customer Decision Hub for insurance services and get more value from your investment in Pega Customer Decision Hub by applying accelerators. Accelerators help insurance providers go live with their applications in a shorter time and provide guidance on best-in-class Pega practices.

The following accelerators are available for Pega Customer Decision Hub implementations for insurance services:

[Insurance Customer Analytical Data Model](#)

This component holds the insurance customer analytical data model (xCAR) that includes data entities and attributes to support insurance use cases.

[Next-Best-Action Designer Template for Insurance](#)

The Next-Best-Action Designer template defines a standard configuration to accelerate your implementation of Next-Best-Action Designer and Pega Customer Decision Hub for insurance services.

1. Go to [Pega Marketplace](#) and download the components that you want to add to your application.
2. Install the components by following the instructions that are provided with each component.

Implementing Pega 1:1 Operations Manager

Pega 1:1 Operations Manager is a module of Pega Customer Decision Hub that non-technical teams use in the Business Operations Environment (BOE) to manage business change requests and quickly deploy them to production.

Pega 1:1 Operations Manager facilitates the business change process by empowering business users to request changes in a dedicated, user-friendly application. To learn more about requesting and deploying business changes through Pega 1:1 Operations Manager, see [Business operations use cases](#).

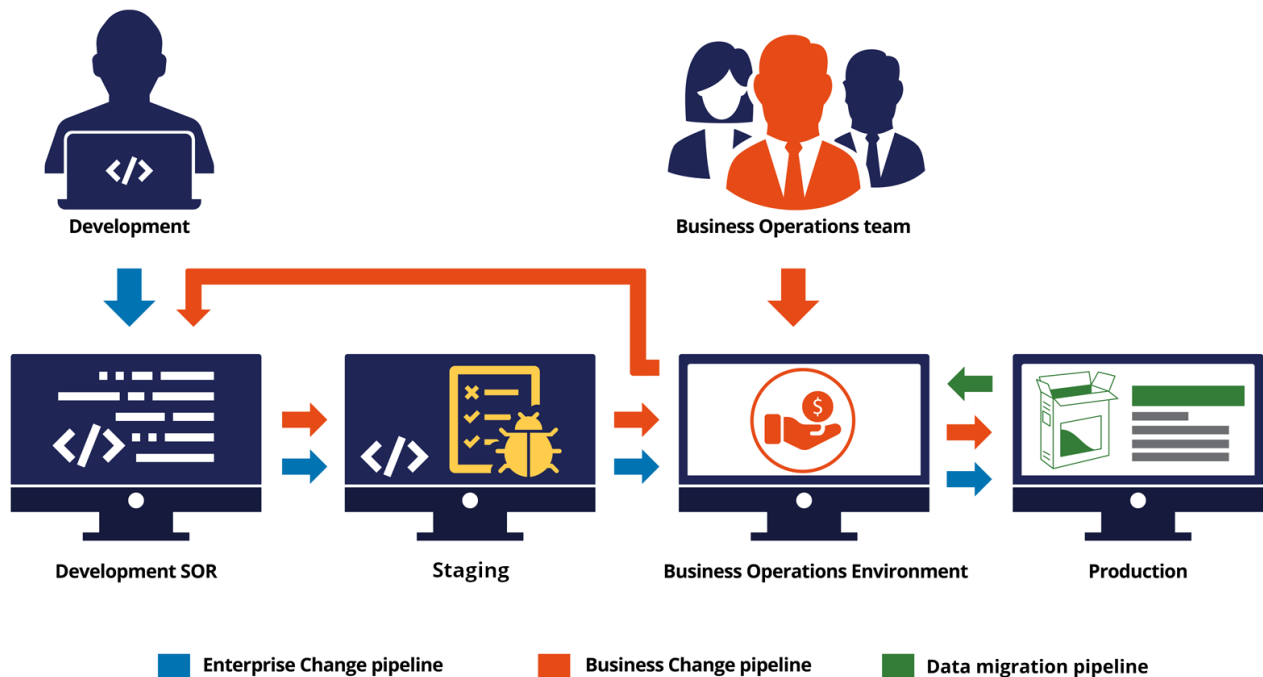
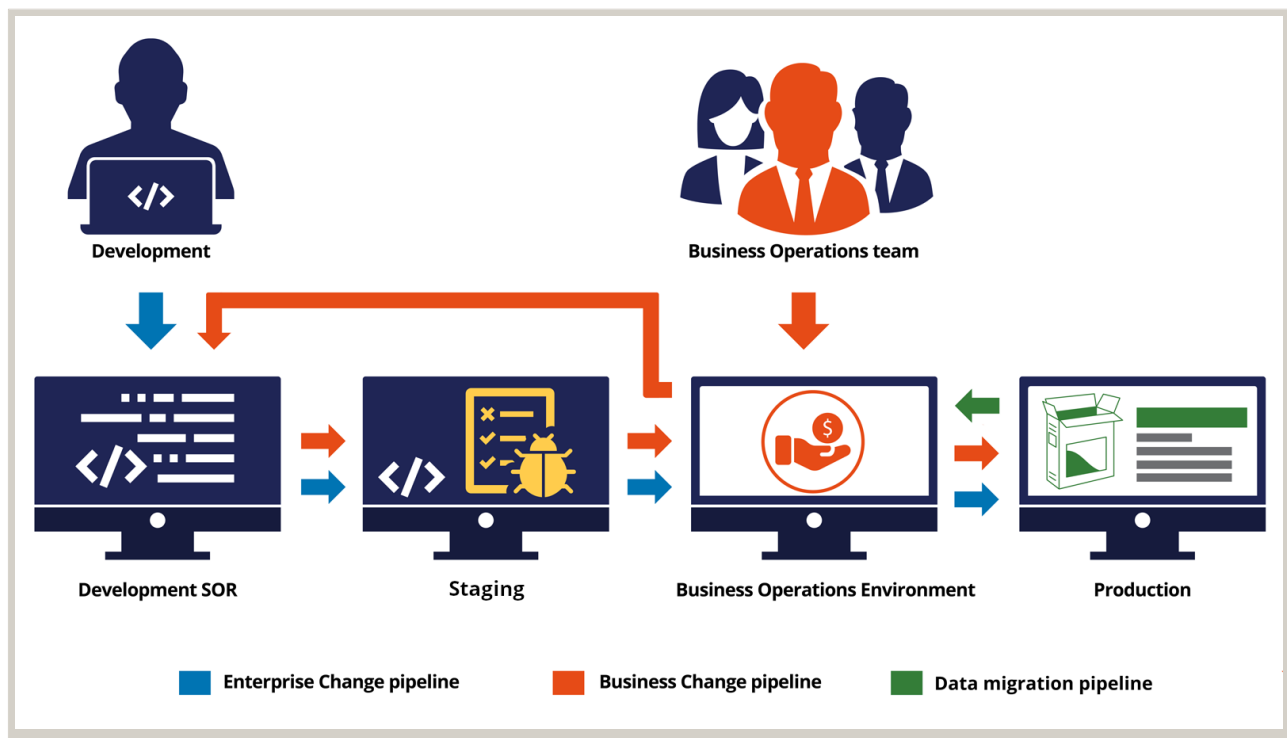
Deployment architecture

To make business changes to a live system with Pega Customer Decision Hub, you need the Pega 1:1 Operations Manager application in the BOE environment, in which business users can access Pega 1:1 Operations Manager and request business changes.

Pega Customer Decision Hub and Pega 1:1 Operations Manager are first implemented in the Development environment. Then, both applications are packaged and migrated to the Staging, BOE, and Production environments.

For more information about the four environments and the data pipelines that are required for a system with Pega Customer Decision Hub, see [Deployment architecture for Pega Customer Decision Hub](#).

Pega Customer Decision Hub deployment architecture



Understanding the Pega 1:1 Operations Manager implementation process

At a high level, implementing Pega 1:1 Operations Manager consists of the following stages:

1. Setting up Pega 1:1 Operations Manager in the Development environment:
 1. Creating the Pega 1:1 Operations Manager implementation application in its own implementation layer.
 2. Creating an overlay application for holding your artifact changes.
 3. Extending and customizing the Pega 1:1 Operations Manager implementation application to meet your requirements, for example, by extending the approval process.
2. Migrating the Pega 1:1 Operations Manager application to the Staging, BOE, and Production environments.

To set up Pega 1:1 Operations Manager, perform the following procedures:

1. [Prerequisites for Pega 1:1 Operations Manager](#)
2. [Creating the Pega 1:1 Operations Manager implementation application](#)
3. [Creating the overlay application for Pega 1:1 Operations Manager](#)
4. [Creating users in Pega 1:1 Operations Manager](#)

5. [Configuring a team in Pega 1:1 Operations Manager](#)
6. [Updating the PegaMKT-Work-Ops email account](#)
7. [Verifying Pega 1:1 Operations Manager implementation](#)
8. [Building features](#)

Prerequisites for Pega 1:1 Operations Manager

Before you start your Pega 1:1 Operations Manager implementation, complete the following tasks.

1. Create the implementation application for Pega Customer Decision Hub.

For more information, see [Creating the Pega Customer Decision Hub implementation layer application](#).

Resources

To prepare for your Pega 1:1 Operations Manager implementation, ensure that you understand the Revision Management functionality. For more information, see [Managing business changes](#).

Creating the Pega 1:1 Operations Manager implementation application

In your development environment, run the New Application wizard to create your application.

1. Log in to your Pega Customer Decision Hub as an operator with in the *CDHAdmins* access group for your Pega Customer Decision Hub implementation application.
2. In the Dev Studio header, click the name of your current application and click **New Application**.
3. Select **Pega Operations Manager**.
4. Follow the New Application wizard instructions.
For more information, see [Creating applications](#). **Important:** On the **Define the application and organization name** screen, enter the same organization name as for your Pega Customer Decision Hub implementation application.
Result: The New Application wizard creates the Pega 1:1 Operations Manager implementation application and a set of access groups for the application.
5. Click **Done**.

Creating the overlay application for Pega 1:1 Operations Manager

Create an application overlay to hold the business-as-usual changes requested by business users through the Pega 1:1 Operations Manager.

Before you begin: Note: Pega Customer Decision Hub automatically generates an overlay application with a name that starts with the letters RTC. You must create one additional overlay by using the **Create New Application Overlay** wizard. If you have already created an additional application overlay for Pega Customer Decision Hub, do not create a new overlay now.

1. Log in to your Pega Customer Decision Hub implementation application as an operator with access to Dev Studio.
2. Make sure that the application is pointing to your Pega Customer Decision Hub implementation application.
3. Create an application overlay by using the **Create New Application Overlay** wizard.
 - a. Click **Configure Decisioning Infrastructure Revision Management Application Overlays**.
 - b. Click **New**.
 - c. Click **Create**.
 - d. Enter a unique name for the overlay. **For example:** Use a format like <your organization> Business Changes, for example, U+ Business Changes. The application short name is automatically truncated, for example, to UBusiness. To edit the short name, click on it. **Important:** If you change the short name for the overlay, make sure to also change the Revision rule set name to the same value. For example, if you change the short name to BizChanges, you should also set the Revision rule set name to BizChanges.
 - e. **Optional:** Change the revision rule set name.
 - f. In the drop-down list, select your Pega 1:1 Operations Manager implementation application.
 - g. Click **Next**.
 - h. Click **Next**.
 - i. Select a single rule that will be included in your overlay, for example, the decision data rule for a particular issue or group.
Tip: Unlike with standard Revision Management, in Pega 1:1 Operations Manager, you do not need to manually add more than one rule to the overlay. The system will automatically add any other rule that you modify into the overlay application.
 - j. Click **Include for revision management**.
 - k. Click **Next**.
 - l. Verify that all given details are correct, and then click **Create**.

For more information about application overlays, see [Creating application overlays](#).

Creating users in Pega 1:1 Operations Manager

Review the list of available user roles and access groups, and create operator accounts for your users.

While creating operator accounts, make sure to enable the **Allow rule check out** option on the **Security** tab of the operator account. For more information about creating operator accounts, see [Creating an operator ID](#). **Important:**

Because Pega 1:1 Operations Manager follows best practice guidelines for case management, the same operator may not combine different user roles. For example, the same operator may not request a change, and then approve their own change request. To execute the entire change request process, make sure to create operator accounts in the *BusinessUser*, *TeamLead* / *RevisionManager*, and *NBASpecialist* access groups.

You must assign at least one operator to the *TeamLead* access group as their default access group, otherwise Pega 1:1 Operations Manager will not work correctly.

User roles	User tasks	Access groups	Default portal
Administrator	Maintaining and administering the application after it is implemented	<i>your Pega 1:1 Operations Manager implementation application:OpsMgrAdmin</i> Note: An operator with the <i>OpsMgrAdmin</i> access group is required to complete the implementation process.	Dev Studio
Developer	Extending and customizing the Pega 1:1 Operations Manager implementation application	<i>your Pega 1:1 Operations Manager implementation application:OpsDeveloper</i> Note: An operator with the <i>OpsDeveloper</i> access group is required to complete the implementation process.	App Studio
Manager / team lead / scrum master	- Approving and sizing the change requests - Reviewing the changes done for the change request - Resolving the change request	<i>your Pega 1:1 Operations Manager implementation application:TeamLead</i>	Pega 1:1 Operations Manager portal
Business user	- Creating change requests - Updating the details on a change request	<i>your Pega 1:1 Operations Manager implementation application:BusinessUser</i>	Pega 1:1 Operations Manager portal
Next-Best-Action specialist	Making the changes required to implement the change request	<i>your Pega 1:1 Operations Manager implementation application:NBASpecialist</i>	Pega 1:1 Operations Manager portal
Next-Best-Action designer	Modifying the Next-Best-Action configuration	<i>your Pega 1:1 Operations Manager overlay application:StrategyDesigner</i>	Pega Customer Decision Hub portal
Revision manager	Making changes for any decisioning rule or artifact by using the Pega Platform Revision Management feature	<i>your Pega 1:1 Customer Decision Hub overlay application:RevisionManager</i>	Pega Customer Decision Hub portal

Tip: As a best practice, for Pega 1:1 Operations Manager users, only grant access to Revision Portal to your revision manager and selected operators in the *your Pega 1:1 Operations Manager implementation application:TeamLead* access group.

Configuring a team in Pega 1:1 Operations Manager

After you create the operator accounts, an operator in the TeamLead access group must log in to the Pega 1:1 Operations Manager portal and create the teams, or work groups, for the users.

Team leaders can facilitate the processing of change requests by creating or editing work teams directly in the Pega 1:1 Operations Manager portal, without the need to switch to Dev Studio. Assigning team members like NBA Specialists to a work team provides the ability to share and distribute the team workload through a simple, intuitive interface.

1. Log in to Pega 1:1 Operations Manager as an operator in the *OpsAdmin* access group.
2. In the left navigation pane, click **Teams**.
3. Create the following teams:

Team leaders

Members of this team can do the following operations on change requests:

- Withdraw
- Change stage
- Approve for build
- Approve for deployment

These operations are available for all change requests except *Change prediction* requests which are view-only.

The manager of this team should belong to the *<your Pega 1:1 Operations Manager implementation application>:OpsMgrAdmin* or *<your Pega 1:1 Operations Manager implementation application>:TeamLead* access group.

NBA specialists

Members of this team can do the following operations on a change request:

- Work on tasks associated with change requests in the Build and Test stages

The manager of this team should belong to the *<your Pega 1:1 Operations Manager implementation application>:TeamLead* access group.

Business users

Members of this team can do the following operations on a change request:

- Create and submit new change requests

The manager of this team should belong to the *<your Pega 1:1 Operations Manager implementation application>:TeamLead* access group.

4. Open the teams that you created and remove any pre-populated work queues on each team.
5. Add the following default work queues for each team:
 - Team leaders - **Pending approval** and **Pending deployment**
 - NBA specialists - **Pending build**
 - Business users - **Pending details**
6. To add members to a team, click the **gear** icon.
7. Search for the operator that you want to add to the team, and then click **Add**.
Important: Make sure that the operator that you are adding is not associated with any other teams or work groups. As a best practice, in Pega 1:1 Operations Manager, the same operator may not combine different user roles.
8. In the **Edit members** window, make sure to select the check box **Set as the primary team for added operator**.
9. Click **Submit**.

What to do next: For more information about managing teams, see [Creating a team](#).

Updating the PegaMKT-Work-Ops email account

Edit the *PegaMKT-Work-Ops* email account, which Pega 1:1 Operations Manager uses to send out notifications.

1. Log in to Dev Studio by using an operator ID with the *OpsMgrAdmin* access group and the password that you specified for that operator.
2. Click **Configuration Integration Email Email Accounts**.
3. Fill out the required information for the Sender account.
4. **Optional:** Fill out the required information for the Receiver account.
This account is used to receive responses to emails sent from the Sender account.
5. Click **Save**.

Verifying Pega 1:1 Operations Manager implementation

To verify that your Pega 1:1 Operations Manager is correctly implemented, validate the application stack, confirm that you can access the Pega 1:1 Operations Manager portal, and test the change request creation.

1. Log in as an operator in the *<your Pega Customer Decision Hub implementation application>:CDHAdmins* access group.
2. Verify that the overlay application is created.
3. Verify that operators in the following access groups can access the application:
 - *<your Pega 1:1 Operations Manager implementation application>:OpsManagerAdmin*
 - *<your Pega 1:1 Operations Manager implementation application>:BusinessUser*
 - *<your Pega 1:1 Operations Manager implementation application>:TeamLead*
 - *<your Pega 1:1 Operations Manager implementation application>:NBASpecialist*
4. Make sure that all the access groups point to the overlay application.
5. Open each access group and verify that the respective access roles are available in the **Available Roles** section.
6. On the **Advanced** tab check if the work pool is *PegaMKT-Work-Ops*.
7. For the following access groups ensure the **Default destination** field contains the *CDH-Artifacts* ruleset:
 - *<your Pega 1:1 Operations Manager overlay application>:StrategyDesigner*
 - *<your Pega 1:1 Operations Manager overlay application>:RevisionManager*
8. Click **Configure Decisioning Decisions Proposition Management Hierarchy**.
9. Verify that the built-on Pega Customer Decision Hub application has at least one issue and one group in the **Proposition Management** hierarchy.
10. Verify that the overlay SR points to the SR class of your main application.

11. In Dev Studio, click on the name of your overlay application, and then click **Definition**.
12. Verify that the applications are listed in the following orders:
 - Your Pega Customer Decision Hub implementation application
 - Your Pega 1:1 Operations Manager implementation application

Note: The Pega Customer Decision Hub implementation application must be higher on the list of built on applications than the Pega 1:1 Operations Manager implementation application.
13. In the **Presentation** section, verify that the selected skin is **CosmosSkin**.
14. On the **Case & data** tab, verify that the following case types are available:

ChangeRequest	OpsCR-	<i>PegaMKT-Work-Ops-ChangeRequest</i>
CreateAction	OpsCA-	<i>PegaMKT-Work-Ops-CreateOffer</i>
UpdateAction	OpsUA-	<i>PegaMKT-Work-Ops-UpdateOffer</i>

15. Switch to the Pega 1:1 Operations Manager portal and verify that the teams on the **My Teams** page are configured according to your requirements.

Building features

The implementation team enables or extends features to meet your business requirements.

To configure application features, review the topics in the following sections:

- [Selecting the default action flow for actions created from Pega 1:1 Operations Manager change requests](#)
- [Enabling business users to request changes to action bundles](#)
- [Enabling revision settings](#)
- [Enabling peer review](#)
- [Extending the change request approval process](#)
- [Extending custom action attributes](#)
- [Extending the treatment sections in Pega 1:1 Operations Manager](#)
- [Creating change requests with the Pega API](#)
- [Guidelines for extending Pega 1:1 Operations Manager](#)

Selecting the default action flow for actions created from Pega 1:1 Operations Manager change requests

All actions requested by business users in Pega 1:1 Operations Manager are created with the same default flow settings. Choose between flow settings based on a dynamic action template, a specific action, or an empty action flow with no shapes.

1. Log in to Pega 1:1 Operations Manager as an operator in the *TeamLead* access group.
2. In the navigation pane, click the operator icon, and then click **Operations Manager Settings**.
3. Select default action flow for outbound execution.
 - **None** - actions are created with an empty flow that the NBA Specialist can customize manually.
 - **Reference flow from another action** - actions are created by using an existing action as a template. The new action reuses the flow defined in the template action. To modify the action flow, the NBA Specialist must edit the original template action. **Note:** If you select a different action to use as a template while a change request is in the Build stage, the action flow for the change request will be updated accordingly. If a change request is already in the Test or Deploy stage, selecting a different action template will not affect the change request.
 - **Copy flow from another action** - actions are created by copying an existing action. The new action has a flow with the same shapes as the copied action, but it does not reference the copied action directly. The NBA Specialist can edit the new action flow directly without modifying the copied action.
4. Click **Save**.

Enabling business users to request changes to action bundles

Team leaders can enable business users to request updates to action bundles. To activate this option, edit the Pega 1:1 Operations Manager settings.

1. Log in to Pega 1:1 Operations Manager as an operator in the *TeamLead* access group.
2. In the left navigation pane, click the user icon, and then click **Operations Manager Settings**.
3. In the **Data capture** section, select **Enable bundling**.

Result: Business users requesting new actions through Pega 1:1 Operations Manager can now specify if the action should be included in an existing action bundle.

Enabling revision settings

Specify the default Revision Management user for creating new revisions with Pega 1:1 Operations Manager.

1. Log in to Pega 1:1 Operations Manager as an operator in the *TeamLead* access group.
2. In the left navigation pane, click the user icon, and then click **Operations Manager Settings**.

3. In the **Revision settings** section, select an operator with Revision Manager access rights.

Enabling peer review

Increase the quality of your NBA Specialists' work product by enabling a peer review process for changes resulting from a Pega 1:1 Operations Manager change request.

1. Log in to Pega 1:1 Operations Manager as an operator in the *TeamLead* access group.
2. In the left navigation pane, click the user icon, and then click **Operations Manager Settings**.
3. In the **Enable Peer-review process** section, click **Enable peer review**.

Extending the change request approval process

By default, the approval process for change requests created in the Pega 1:1 Operations Manager consists of approval by the team lead in the Idea stage, and then by the Next-Best-Action specialist, in the Test stage. If you want other users, like the reporting manager or the work group manager, to approve a change request instead, extend the change request flow with an additional approval step.

Before you begin: Ensure that manager and work group data for operators involved in the processing of change requests are complete and up-to-date. For more information, see [Fields for operator teams, work queues, and schedules](#).

1. Log in to Dev Studio as an operator with the *Developer* access group.
2. In the navigation pane of Dev Studio, click **Case types Operations change request**.
3. In the **Idea** step, click **Configure process**.
4. Click **Open Process**.
5. Click **Save as** and save the rule into your implementation ruleset.
6. On the **Submit Idea** flow diagram, right-click the **Approve/Reject Change Request** step, and then click **View properties**.
7. To route the request for additional manager approval within the same organizational structure, in the **Approval flow type** list, select **Cascading**.
8. In the **Approval based on** list, select **Reporting structure**.
9. In the **Approval to be completed by** list, select one of the following options:
 - **Reporting manager** - The request must be approved by the manager to whom the request creator reports. This is the manager that you selected in the **Reports to** field on the **Work** tab of the request creator's operator account.
 - **Work group manager** - The request must be approved by the manager of the work group to which the request creator belongs. This is the default work group specified in the **Work** tab of the request creator's operator account.

For more information, see [Fields for operator teams, work queues, and schedules](#).

10. Specify how many managers in the request creator's organizational hierarchy must approve the request by selecting one of the following options:
 - **One** - Only the reporting manager or the work group manager must approve the request.
 - **All** - All managers in the request creator's organizational hierarchy must approve the request.
 - **Custom** - Specify the number of levels that must approve the change request, as determined by the evaluation of When rules. Specify manager at each approval level and define a cut-off with a When rule for each manager.
11. Click **Submit**.

Extending custom action attributes

To enable additional Action properties in Pega 1:1 Operations Manager, extend the underlying Pega 1:1 Operations Manager implementation application.

1. **Optional:** If the decision data rule that corresponds to the issue and group where you want to add a custom attribute is not included in a revision, do the following steps:
 - a. Log in to your Pega Customer Decision Hub application as an operator with access to Dev Studio.
 - b. In Dev Studio, click **Configure Decisioning Infrastructure Revision Management Application Overlays**.
 - c. Click on the overlay application to view it.
 - d. In the **Rules available for revision management** section, click **Edit available rules**.
 - e. Select one of the following rules:
 - If you want to add a custom attribute for all issues and groups, select the top-level rule of the type Next-best-Action Model.
 - If you want to add a custom attribute for a specific issue and group, select the Decision Data rule that corresponds to the issue and group.
 - f. Click **Include for revision management**, and then click **Save**.
 - g. In the Pega Customer Decision Hub portal, on the **Revision Management** landing page, create a new change request, and include the rule that you enabled for revision management in step 1f.
 - h. Click **Next**, and then submit the change request.
 - i. Open the change request that you created, and then do the following steps:
 - If you want to add a custom attribute for all issues and groups, open the included Next-best-Action Model rule, and then click **Taxonomy Property Add property Static**.
 - If you want to add a custom attribute for a specific issue and group, open the included Decision Data rule that corresponds to the issue and group, and then click **Form fields Add field**.

- j. Define the property according to your requirements.
- k. Complete the change request and package the revision.
2. Log in to the Pega 1:1 Operations Manager application.
3. Open the *CustomAttributesPresent* When rule and click **Save as** to save the rule into your Pega 1:1 Operations Manager implementation layer.
4. Set the condition for the *CustomAttributesPresent* rule to *true*.
5. If the custom SR property is at the top-level SR class level or at group level, save the property into the Pega 1:1 Operations Manager implementation layer and set the **Applies-to** class to *Data-pxStrategyResult*.
6. Save the *ActionCustomAttributes* and *NBA_ConfigurationExtAttribute* sections into Pega 1:1 Operations Manager implementation layer and edit them to include additional custom attributes.
7. To display custom attributes in the **Details** tab of a Pega 1:1 Operations Manager change request, save the *AttributesTabInner* section into your Pega 1:1 Operations Manager implementation layer with a new name for both the classes *PegaMKT-Work-Ops-ChangeRequest* and *PegaMKT-Work-Ops-CreateOffer*, and then edit it to include custom attributes on the **Details** tab.

Extending the treatment sections in Pega 1:1 Operations Manager

To enable additional treatment properties in Pega 1:1 Operations Manager, extend the Custom Treatment Element section.

The changes required to extend treatments must be handled in both the Pega Customer Decision Hub and Pega 1:1 Operations Manager implementation applications.

1. Log in to your overlay application as a user in the *RevisionManager* access group.
2. Create a standard change request that includes the treatment Decision Data rule for the treatment that you want to extend.
3. Add the additional treatment property to the form field of the treatment Decision Data rule.
Note: The property must belong to the *Data-pxStrategyResult* class.
4. Complete the change request.
5. Log in to your Pega Customer Decision Hub application as a user in the *CDHAdmins* access group.
6. In Dev Studio, find the appropriate section for the treatment type that you want to extend, and save it into your Pega Customer Decision Hub implementation layer:

Extension section name	Treatment type
<i>CustomEmailTreatmentElement</i>	Email
<i>CustomWebTreatmentElement</i>	Web
<i>CustomMobileTreatmentElement</i>	Mobile
<i>CustomSMSTreatmentElement</i>	SMS
<i>CustomPushTreatmentElement</i>	Push
<i>CustomAssistTreatmentElement</i>	Agent assisted

7. Add the additional treatment property to the extension section.
8. Log in to the Pega 1:1 Operations Manager application as a user in the *OpsDeveloper* access group.
9. Save the properties that you created in step 3 into the Pega 1:1 Operations Manager implementation layer.
10. Add the properties to the extension section and When rule in the following way:

Treatment type	Section rule	When rule
Email	<i>EmailTreatmentForm</i>	<i>EmailCustomAttributesPresent</i>
Web	<i>WebTreatmentForm</i>	<i>WebCustomAttributesPresent</i>
Mobile	<i>MobileTreatmentForm</i>	<i>MobileCustomAttributesPresent</i>
SMS	<i>SMSTreatmentForm</i>	<i>SMSCustomAttributesPresent</i>
Push	<i>PushTreatmentForm</i>	<i>PushNotificationCustomAttributesPresent</i>
Agent Assisted	<i>AgentAssistedTreatmentForm</i>	<i>AgentAssistedCustomAttributesPresent</i>
Other	<i>OtherTreatmentForm</i>	Not applicable

Result: When you create a change request in Pega 1:1 Operations Manager, after completing the Build stage, the new treatment properties are visible on the associated Decision Data rule form.

Creating change requests with the Pega API

You can integrate Pega 1:1 Operations Manager with an external application, for example, Atlassian's Jira or Pega Agile Studio. In this way, you can continue to use the external application for managing and approving ideas, and then send the change requests that you create in the external application directly to Pega 1:1 Operations Manager by using the Pega API.

Change request creation

You create new change requests using the POST method.

Required parameters

- *caseTypeID* - Specify the class name of the Change Request class.
- *caseTypeID* - Specify the process name. The default value is *pyStartCase*.
- *content* - Specify additional case type properties related to the case type.

Request URL

`https://<host:port>/prweb/api/v1/cases`

Sample request

```
{"caseTypeID":"PegaMKT-Work-Ops-ChangeRequest",
"processID":"pyStartCase",
"content":
{"pyLabel":"changeRequestLabel",
"Name":"Change request name"}}
```

Sample response

```
{"ID":"PEGAMKT-WORK-OPS OPSCR-244",
"nextPageID": "Confirm",
"pxObjClass": "Pega-API-CaseManagement-Case"}
```

Successful response code

201

Change request updates

You update change requests using the GET and PUT method of the *cases{ID}* Pega API.

Required parameters

- *ID* - Specify the ID of the change request that you want to update.
- *If-Match* - Specify the ETag value. To get the ETag value, make a REST call to any method (for example, *GET /cases/{ID}*) with a dummy If-Match value, then check the ETag value in the response header.
- *requestBody* - Specify the properties that you want to update. To check the input structure for *requestBody*, call the *GET /cases/{ID}* method.

Request URL

`https://<host:port>/prweb/api/v1/cases`

Sample request

```
ID:"PEGAMKT-WORK-OPS OPSCR-244",
If-Match:"20190808T102253.902 GMT",
requestBody:
{"content":
{"pyLabel":"changeRequestLabel",
"Name":"Change request name"}}
```

Successful response code

204

Guidelines for extending Pega 1:1 Operations Manager

Some areas of Pega 1:1 Operations Manager are commonly extended during the implementation process, while others should be left as default for most implementations.

Recommended extension points

Review the following areas of Pega 1:1 Operations Manager and customize them to match your requirements:

- The mapping of properties and the UI of change request phases
- SLA details
- Routing
- Notifications
- Approval process
- Additional, non-default change request types
- Testing steps

Customizing the UI of change request phases

To customize the UI of the Plan and Build stage of a Pega 1:1 Operations Manager change request, do the following steps:

1. Save the corresponding wrapper section into your implementation layer.

For example, to customize the Channels screen, save the *Channels* section into your implementation layer.

The wrapper section contains two sections:

- Default - For example, *DefineChannels*
 - Customizable extension - For example, *DefineChannelsCustomization*
2. Save the default section with the same name as the customizable section.

For example, save the *DefineChannels* section into your implementation layer as *DefineChannelsCustomization*.

3. Change the visibility of the default section to **Never**.
4. Change the visibility of the customizable extension section to **Always**.
5. Customize the extension section as required.

Areas which should be left as default

In most cases, do not modify the following areas of Pega 1:1 Operations Manager:

- The built-in class definition and the class-to-table mapping
- The Build change request phase
- The integration between Pega 1:1 Operations Manager and the Revision Management functionality of Pega Platform
- Default change request types

Configuring engagement channels

Set up your engagement channels, such as real-time containers to manage content on Web pages, email and SMS accounts, and push notifications.

1. [Implementing real-time containers](#)
2. [Configuring channels](#)
3. [Enabling channels](#)
4. [Adding additional channels to Next-Best-Action Designer](#)
5. [Configuring additional channels for use with volume constraints](#)
6. [Optional: Configuring additional channels for use with contact policies](#)
7. [Defining real-time container triggers](#)
8. [Defining real-time event triggers](#)
9. [Configuring a recurring schedule for Next-Best-Action](#)
10. [Creating treatment eligibility criteria](#)
11. [Optional: Configuring repository storage for marketing images](#)
12. [Optional: Integrating Pega Customer Decision Hub with Adobe Experience Manager](#)
13. [Optional: Implementing Paid Media Manager](#)
14. [Implementing Pega Next-Best-Action Advisor](#)

Implementing real-time containers

Real-time containers manage content in real-time channels, for example, on web pages. When a container is loaded, for example, when an individual views an ad for an action in a predefined placement on the Web channel, it triggers Next-Best-Action.

To prepare your application to use real-time container triggers for your actions, do the following tasks:

1. [Modifying pool settings for PegaMKTContainer service package](#)
2. [Returning additional properties in a real-time container response](#)
3. [Modify Service Level Agreement settings for the Container service](#)

Modifying pool settings for PegaMKTContainer service package

PegaMKTContainer is a [Pega service package](#) that encompasses real-time container services. Pool settings in a service package determine the size and behavior of pools to service requestors, which in turn affect the amount of system resources used.

Update the default values for requestor pool settings to meet your system's needs.

1. Log-in to your Pega Customer Decision Hub implementation application with an operator who has access to Dev Studio.
2. In the navigation pane of Dev Studio, click **Records**.
3. Expand the **Integration-Resources** category, and then click **Service Package**.
4. Click the *PegaMKTContainer* row to open the service package.
5. Click the **Pooling** tab, and then modify the requestor pool settings:
 - a. In the **Maximum idle requestors** field, enter value 10.

This setting determines the maximum number of idle requestors that can be in the pool for container services.

- b. In the **Maximum active requestors** field, enter value 10.

This setting determines the number of concurrent requestors the system can create and put into use for the services in this package.

The default value for the two settings is -1. When you modify these settings for the first time, use 10 as a starting value. Revisit these settings from time to time and update them to meet your system's needs. Leaving the values to -1 could result in performance issues and can sometimes bring down the Decision hub nodes. For more information on configuring requestor pools, see [Configuring a requestor pool for a service package](#) and [Managing requestor pools](#)

6. Save the service package.

Returning additional properties in a real-time container response

You can make additional properties available for real-time container responses.

1. Create a Proposition or Strategy property under the top-level SR class.
Proposition properties define and store the value on the Action ruleform. Strategy properties are temporary and calculated as part of strategy execution. **Tip:** Proposition properties should be created using the Proposition Management landing page. Strategy properties are created from within a strategy ruleform.
2. Make the property available to real-time container data flows.
For more information, see [Making custom Strategy Result properties available for real-time container data flows](#).
3. Extend the Interaction History table for the CaptureResponse service.
For more information, see [Extending the Interaction History table for the CaptureResponse service](#).
4. Set the custom property in a real-time container response.
For more information, see [Setting the custom property in a real-time container response](#).
5. **Optional:** Add custom properties for a supporting decision that uses an alternate SR class.
For more information, see [Returning properties of an alternate SR class in a real-time container response](#).

Making custom Strategy Result properties available for real-time container data flows

Every property output from a decision strategy associated with real-time containers in the Next-Best-Action Designer must be identified in the *pyDefaultWorkingSet* data transform. Update this data transform if you use real-time containers and if you have configured custom action properties.

1. In Dev Studio, create a new data transform called *pyDefaultWorkingSet* in your top level SR class.
2. On the **Definition** tab of the data transform, select the check box **Call superclass data transform**.
Enabling the **Call superclass data transform** option ensures that all properties defined in the built-in *pyDefaultWorkingSet* data transform are available. You do not need to define them again in your new data transform.
3. Add any properties which you want to make available as output properties from Strategy in the generated data flows, as shown in the following example:

An example configuration of the *pyDefaultWorkingSet* data transform

	Action	Target	Relation	Source
▼ • 1	Append and Map to	defaultWorkingSetPage.pyWorkingSet	a new page	
• 1.1	Set	.pyPropertyName	equal to	"AccountType"

	Action	Target	Relation	Source
▼ • 1	Append and Map to	defaultWorkingSetPage.pyWorkingSet	a new page	
• 1.1	Set	.pyPropertyName	equal to	"AccountType"

- a. In the first row, in the **Action** column, enter: Append and Map to
- b. In the first row, in the **Target** column, enter: defaultWorkingSetPage.pyWorkingSet
- c. In the second row, in the **Action** column, enter: Set
- d. In the second row, in the **Target** column, enter: .pyPropertyName
- e. In the second row, in the **Source**, enter the name of the property which you want to make available.
4. On the **Pages & Classes** tab, enter the following information:
 - a. In the **Page name** field, enter: defaultWorkingSetPage
 - b. In the **Class** field, enter: Data-MO-DDF-DecisionStrategy
5. Click **Save**.

Extending the Interaction History table for the CaptureResponse service

Extend the Interaction History table to support custom properties for the CaptureResponse service.

1. In Dev Studio, create the properties that you want to add in the appropriate interaction history class (Fact or Dimension). **For example:** Create an AccountID property in the *Data-Decision-IH-Fact* class.
2. Create matching columns in the appropriate interaction table (Fact or Dimension).
Note: If you use a database editor to add the columns, verify that they have been properly created.
For example: Create an AccountID column in the *pr_data_ih_fact* table.
3. Copy the *Data-Decision-IH-Configuration.pyInteractionHistoryConfiguration* data transform to your Artifacts ruleset and

update it to add the new properties that you created in step 1.

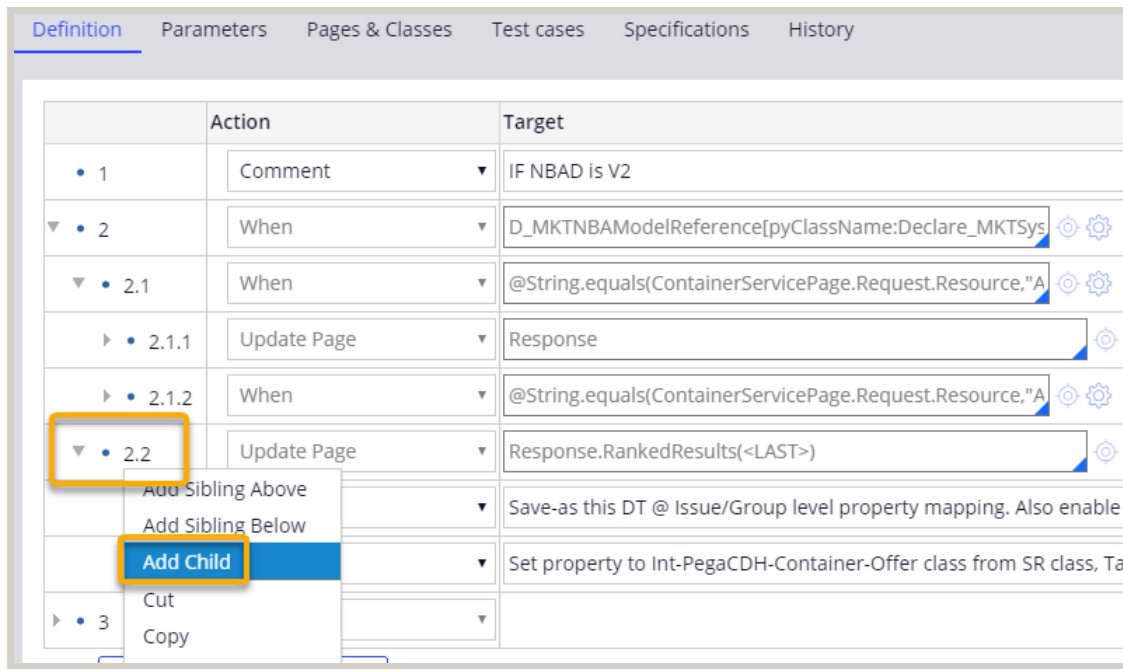
4. Click **Decisioning Decisions Proposition Management Hierarchy**.
5. Create the matching properties, for example, *AccountID*.
Note: By default, this creates columns with a length of 32 characters. If required, edit the property and set the **Max Length** to a higher value.
6. Copy the *Data-Decision-IH-Configuration.MKTInteractionHistoryExtension* data transform to your Artifacts rule set and update it to add the new properties that you created in step 1.
Tip: Follow the pattern documented in the **Comment** field.
7. Re-save the corresponding Database Table rule for the interaction class where you added the properties. **For example:** Re-save the *Data-Decision-IH-Fact* database table.

Setting the custom property in a real-time container response

After you add the new property to the Strategy Result class, ensure that it is returned as part of the Container service JSON response.

The Container service response structure can return two types of results: *RankedResults*, that is, the actual strategy results, and *Actions*, that is, the category or type of results. An example of an Action result is *Make sales offer*.

1. In Dev Studio, create a property under the appropriate class.
Tip: *Int-PegaCDH-Container-Offer* is usually the recommended class to use.
2. Save the *ExtendedContainerProperties* Data Transform rule into your implementation ruleset.
In the **Apply to** field, select the top-level SR class or the corresponding Issue or Group class.
3. On the **Definition** tab of the new *ExtendedContainerProperties* rule, map the property that you created in the *Int-PegaCDH-Container-Offer* class to a property in your top-level SR class.
 - a. Right-click row **2.2**, and then click **Add child**, as shown in the following figure:



Definition Parameters Pages & Classes Test cases Specifications History				
	Action	Target		
• 1	Comment	IF NBAD is V2		
▼ • 2	When	D_MKTNBAModelReference[pyClassName:Declare_MKTSys		
▼ • 2.1	When	@String.equals(ContainerServicePage.Request.Resource,"A		
▶ • 2.1.1	Update Page	Response		
▶ • 2.1.2	When	@String.equals(ContainerServicePage.Request.Resource,"A		
▼ • 2.2	Update Page	Response.RankedResults(<LAST>)		
			Add Sibling Above	
			Add Sibling Below	
			Add Child	
			Cut	
			Copy	
▶ • 3				

- b. In the new row, in the **Action** column, select **Set**.
- c. In the **Target** column, enter the property that you created in the *Int-PegaCDH-Container-Offer* class, for example, *AccountType*.
- d. In the **Source** column, enter the property that you created in your top-level SR class, for example, *AccountType*.
- e. Click **Save**.
4. To extend the Action results that the container returns, do the following steps:
 - a. In Dev Studio, create a property under the *Int-PegaCDH-Container-Action* class.
 - b. On the **Definition** tab of the new *ExtendedContainerProperties* rule, right-click on row 2.1, and then click **Add child**.
 - c. In the new row, in the **Action** column, select **Set**.
 - d. In the **Target** column, enter the property that you created in the *Int-PegaCDH-Container-Action* class.
 - e. In the **Source** column, enter the property that you created in your top-level SR class.
5. If you previously associated the real-time container with any level of Next-Best-Action hierarchy, do the following steps to ensure that the configuration is updated:
 - a. In App Studio, click **Next-Best-Action Designer Channels**.
 - b. Click **Edit**.
 - c. In the **Triggers** section, remove the real-time container association by clicking **Configure**, and then **Remove**.
 - d. Click **Apply**.
 - e. Click **Save**.
 - f. In the **Triggers** section, restore the real-time container association by clicking **Configure**, and then **Add**.
 - g. Click **Apply**.
 - h. Click **Save**.

Returning properties of an alternate SR class in a real-time container response

In order to add custom properties to the Container response for a supporting decision that uses an alternate strategy result class, do the following steps:

1. In Dev Studio, create a property under the alternate SR class by using the **Create property** method.
Note: Set the **pyDecisioningItem** custom field for this property to *StrategyProperty*.
2. In Dev Studio, create a new data transform called *pyDefaultWorkingSet* in the alternate SR class.
3. On the **Definition** tab of the data transform, select the check box **Call superclass data transform**.
Enabling the **Call superclass data transform** option ensures that all properties defined in the built-in *pyDefaultWorkingSet* data transform are available. You do not need to define them again in your new data transform.
4. Add any properties which you want to make available as output properties from Strategy in the generated data flows, as shown in the following example:

An example configuration of the *pyDefaultWorkingSet* data transform

	Action	Target	Relation	Source
▼ • 1	Append and Map to	defaultWorkingSetPage.pyWorkingSet	a new page	
• 1.1	Set	.pyPropertyName	equal to	"AccountType"

	Action	Target	Relation	Source
▼ • 1	Append and Map to ▼	defaultWorkingSetPage.pyWorkingSet	a new page ▼	
• 1.1	Set ▼	.pyPropertyName	equal to	"AccountType"

- In the first row, in the **Action** column, enter: Append and Map to
 - In the first row, in the **Target** column, enter: defaultWorkingSetPage.pyWorkingSet
 - In the second row, in the **Action** column, enter: Set
 - In the second row, in the **Target** column, enter: .pyPropertyName
 - In the second row, in the **Source**, enter the name of the property which you want to make available.
- On the **Pages & Classes** tab, enter the following information:
 - In the **Page name** field, enter: defaultWorkingSetPage
 - In the **Class** field, enter: Data-MO-DDF-DecisionStrategy
 - Click **Save**.
 - In Dev Studio, create a property under the *Int-PegaCDH-Container* class.
 - Save the *TransformSROOutput* Data Transform rule into your implementation ruleset.
In the **Apply to** field, select the alternate SR class.
 - On the **Definition** tab of the new *TransformSROOutput* rule, map the property that you created in your alternate SR class.

The following image shows an example configuration of the *TransformSROOutput* data transform.

	Action	Target	Relation	Source	
▼ • 1	Append and Map ▼	ContainerServicePage.Response.RankedResult	a new page ▼		
• 1.1	Set ▼	.ClassIdentifier	equal to	"Eligibility"	Select values + ⚙️
• 1.2	Set ▼	.EligibilityType	equal to	.EligibilityType	Select values + ⚙️

	Action	Target	Relation	Source	
▼ • 1	Append and Map ▼	ContainerServicePage.Response.RankedResult	a new page ▼		
• 1.1	Set ▼	.ClassIdentifier	equal to	"Eligibility"	Select values + ⚙️
• 1.2	Set ▼	.EligibilityType	equal to	.EligibilityType	Select values + ⚙️

- Clear the **Call superclass data transform** checkbox.
- Save and check in your changes.

Modify Service Level Agreement settings for the Container service

The Container REST service triggers real-time container processing. You can modify the default Service Level Agreement settings for the Container service, for example, to change the maximum number of consecutive violations which the Service allows before it invokes a fallback service activity.

- Save the Container service into your implementation ruleset.
- In the **Processing options** section, edit the following settings as required:
 - **Maximum duration for service activity (milliseconds)** - The maximum time for a service activity to run. If a service exceeds the specified value, it logs a Service Level Agreement violation alert. The default value for this setting is 500.
 - **Maximum consecutive violations** - The maximum number of consecutive Service Level Agreement violations for the Container service. If the number is exceeded, the interval service returns the default set of actions for the Container service for a length of time defined in the **Retry interval (Seconds)** setting. The default value for this setting is 3.
 - **Retry interval (seconds)** - The time during which the interval service returns the default set of actions for the Container service. After the retry interval ends, the interval service tries to invoke the actual service activity. The default value for this setting is 10.

Configuring channels

Channels are a means of communication that customers can use to interact with a company. Pega Customer Decision Hub includes integrated, multi-channel support capabilities for managing email, SMS, push notifications and web-based service interactions. Additionally, in-person interactions can be conducted using a mobile device, such as a tablet computer.

Pega Customer Decision Hub allows you to build your process once and reuse it in any channel.

Configure channels by performing the following actions:

- [Configuring an outbound email account](#)
- [Configuring an outbound SMS account](#)
- [Configuring an inbound SMS account](#)
- [Configuring SMS delivery receipts](#)
- [Defining push notifications](#)
- [Configuring push notifications](#)

- [Specifying delivery time frames](#)

Configuring an outbound email account

Configure the predefined email accounts which Pega Customer Decision Hub uses to send out emails, including messages related to actions or campaigns. Optionally, you can also set up additional email accounts for action-related emails.

Note: Prepackaged connectors for email service providers (ESPs) [Mailgun](#), [SendGrid](#), and [Mailjet](#) are available on [Pega Marketplace](#). Install the connectors to help reduce deployment time by automating the configuration of an email account.

1. In App Studio, click **Settings Channels Email**.
2. Click the **Edit** icon by a predefined email account to configure it.
By default, Pega Customer Decision Hub includes the following outbound email accounts:
 - **Default** - This account is used to send out most types of emails, including password reset emails, Email and Passbook Treatments in actions, and other internal and external notifications. It is not used for Campaign-related emails.
 - **PegaMKT-Work** - This account is used to send Campaign-related emails, including Campaign Suspended and Campaign Approved notifications.
 You must provide the required connection data for both accounts to ensure that emails to customers and internal users are properly delivered.
3. **Optional:** If you want Email, Passbook and Multi Treatments in actions to be sent from a non-default email address, click **Add new** to add a new email account.
Note: After configuring the new email account, select it as the delivery account for the Send Email, Send Passbook, and Send Multi shapes in the actions where you want to use the non-default email address for correspondence.
4. **Optional:** In the **Account Availability** section, specify delivery time frames for email related to actions.
Select one of the following options:
 - **Always Available** - The account is always available to deliver messages.
 - **Specify Range** - The account only delivers messages during the specified time range. Specify the following value for the time range:
 - **From** - Start value for the time range, for example, 9.00 AM
 - **To** - End value for the time range, for example, 7.00 PM
 - **Time Zone** - Time zone for the start and end time values. By default, Pega Customer Decision Hub uses the time zone of the current user.**Note:** Delivery time frames do not apply to emails which are not related to actions, for example, work object-related notification.

Configuring an outbound SMS account

Configure the predefined outbound SMS accounts which Pega Customer Decision Hub uses to send out SMS messages, including messages related to actions. Optionally, you can also set up additional SMS accounts for action-related messages.

1. In App Studio, click **Settings Channels SMS Outbound**.
2. Click the **Edit** icon by the **Default** account to configure it.
3. In the **Setup** tab, fill out the required connection details:
 - **Host Address** - The address of the SMS center host, for example, my.smsc.com.
 - **Host Port** - The port number of the SMS center host, for example, 1041.
 - **User ID** and **Password** - Authentication credentials for the SMS account.
 - **Test Number** - The number which is used to receive test messages for this account.
 - **Sender's Number** - The number which is displayed as the sending number for messages sent from this account.
4. **Optional:** In the **Maximum Connections** field, enter the maximum number of connections available at the same time.
5. **Optional:** Select **Track delivery receipt** to enable delivery receipts. For more information, see [Configuring SMS delivery receipts](#).
6. **Optional:** Set **Delivery mode** to **Asynchronous** to send messages in bulk using the asynchronous option.
Configure the following options:
 - **Unacknowledged queue size** - The maximum queue size for messages that were delivered to a consumer but not yet acknowledged.
 - **Wait on rejection (seconds)** - The interval to wait for the message to be rejected.
 - **SMS messages per second** - The number of messages to be sent out per second.
 - **Acknowledgement timeout (milliseconds)** - The interval before a message is considered unacknowledged.
7. **Optional:** In the **Account Availability** section, specify delivery time frames for SMS messages related to actions.
Select one of the following options:
 - **Always Available** - The account is always available to deliver messages.
 - **Specify Range** - The account only delivers messages during the specified time range. Specify the following value for the time range:
 - **From** - Start value for the time range, for example, 9.00 AM
 - **To** - End value for the time range, for example, 7.00 PM
 - **Time Zone** - Time zone for the start and end time values. By default, Pega Customer Decision Hub uses the time zone of the current user.**Note:** Delivery time frames do not apply to messages which are not related to actions, for example, work object-related notification.
8. **Optional:** In the **Advanced** tab, configure advanced account settings, such as response timeout, character encoding, and reconnect options.

9. **Optional:** In the **Additional settings** tab, specify the name and value for any additional SMPP configuration settings which are not available in the **Setup** and **Advanced** tabs.

Property name	Property value	
unackedQueueSize	5000	

[+ Add row](#)

Property name	Property value	
unackedQueueSize	5000	

[+ Add row](#)

10. Click **OK** and then click **Test** to test your settings.
The test message is sent to the **Test Number** specified in step 3.
11. **Optional:** If you want SMS and Multi Treatments in actions to be sent from a non-default SMS account, click **Add connection** to add a new SMS account.
Note: After configuring the new SMS account, select it as the delivery account the Send SMS and Send Multi shapes in the actions where you want to use the non-default SMS account for messaging.

Configuring an inbound SMS account

Configure an inbound SMS account which Pega Customer Decision Hub uses to listen for Customer responses to actions, and then specify how to process inbound SMS messages that are received by this inbound SMS account.

When the inbound SMS account receives a message, Pega Customer Decision Hub checks the sender's phone number against the Customer table to find the Customer who sent the message. If no matching Customer is found, the message is treated as an error and moved to the *UnhandledInboundSMS* workbasket for further processing.

You can configure the inbound SMS account to parse the SMS text to determine the response, and then associate each response with a different action. In a simpler scenario, you can associate all responses to a particular account with a single action.

1. In App Studio, click **Settings Channels SMS Inbound**.
2. Click **Add connection**.
3. In the **Connection Setup** tab, fill out the required connection details:
 - **Account name** - A unique name for the inbound SMS account, for example, InboundSMSMain. The name must contain only letters and numbers.
 - **Host address** - The address of the SMS center host, for example, my.smsc.com.
 - **Host port** - The port number of the SMS center host, for example, 1041.
 - **Inbox address** - The address of the SMS inbox on which Pega Customer Decision Hub should listen for responses.
 - **IP address** - The IP address of the SMS center host.
 - **User ID** and **Password** - Authentication credentials for the SMS account.
4. **Optional:** In the **Additional settings** tab, specify the name and value for any additional SMPP configuration settings which are not available in the **Connection Setup** tab.

Property name	Property value	
unackedQueueSize	5000	

[+ Add row](#)

Optionally, configure additional SMPP properties here, by providing property name and value

Property name	Property value	
unackedQueueSize	5000	

[+ Add row](#)

5. In the **Response behaviors** tab, specify how the application should process inbound SMS messages.
 - a. **Optional:** Specify what type of messages this inbound SMS account will receive.
The following options are available:
 - **Messages and delivery receipts** - The SMS account can receive incoming messages, as well as delivery receipts for SMS messages which Pega Customer Decision Hub sends to Customers.
 - **Delivery receipts only** - The SMS account can receive only delivery receipts.
 For more information, see [Configuring SMS delivery receipts](#).
 - b. Select how Pega Customer Decision Hub should match received messages to responses:
 - **One response** - Select this option to assign the same action and response value for all SMS messages received on this account.
 - **Decision table** - Select this option to select one or more decision tables which associate the received message with a positive or negative action response. Each result from the decision table can be associated with the same action or with different individual actions.
 - c. Configure the necessary settings for matching received messages to responses.
 - If you selected the **One response** option, in the **Responses for inbound messages** section, select an action and a response. Depending on the selected response, all incoming messages are treated as either acceptance or rejection for the selected action.
 - If you selected the **Decision table** option, in the **Manage decision tables** section, configure how the incoming messages should be matched to actions, and then select the decision table for parsing the response. For more information, see [Using decision tables to manage SMS action responses](#).
 - d. Specify how Pega Customer Decision Hub should process responses to actions which are not active for a given Customer:
 - **Push it to a workbasket** - Select this option to prevent Customers from responding to actions which are not active for them. Responses to inactive actions are treated as an error and moved to the *UnhandledInboundSMS* workbasket for further processing. You can access this workbasket in the Case Manager portal.
 - **Start the action for the customer** - Select this option to initiate a new action for the Customer if they respond to an action which is not active for them.
6. Click **OK**.
7. Toggle the control in the **Running** column to start listening for incoming messages.
The **Connection status** column indicates the current status of the account.

- [Using decision tables to manage SMS action responses](#)

Using decision tables to manage SMS action responses

If you selected the **Decision table** option while configuring response behavior for your inbound SMS account, you must select one or more decision tables which associate inbound SMS messages with a positive or negative action response. Each result from the decision table can be associated with the same action or with different individual actions.

1. In Dev Studio, configure a decision table in the *PegaMKT-Work-Offer* class. Below are some considerations for accessing message and Customer data in decision tables:
 - Customer data is available through the *.Customer* page property under the action class.
 - Inbound message data is available through the *.InboundResponseData.SMS* page property under the action class. The *.InboundResponseData.SMS* page includes properties for the incoming number (*.SourceNumber*) and the message content (*.MessageBody*).
 - The below screenshot shows a predefined *InboundSMSParsingSample1* decision table, which you can use as template to create your own decision tables.

Table Results Parameters Pages & Classes Test cases Specifications History				
        Select values Show conflicts Show completeness Export Import				
Conditions			Actions	
	◦ Message contains offer name	◦ Message contains Yes	◦ Message contains No	Return
◦ if	true	true		→ Interested
◦ else if			true	→ Not Interest
otherwise				→ Maybe

Table	Results	Parameters	Pages & Classes	Test cases	Specifications	History
-------	---------	------------	-----------------	------------	----------------	---------

							Select values	Show conflicts	Show completeness	Export	Import
--	--	--	--	--	--	--	---------------	----------------	-------------------	--------	--------

	Conditions			Actions	
	Message contains offer name	Message contains Yes	Message contains No		Return
if	true	true		→	Interested
else if			true	→	Not Interest
otherwise				→	Maybe

Note: If you want to use more than one decision table, do not specify a return value for the **otherwise** row in the decision tables. Leaving the **otherwise** row blank allows Pega Customer Decision Hub to move on to evaluating the next decision table in the list.

- In the **Manage decision tables** section of the **Response behaviors** tab, select one of the following options:
 - One action** – Select this option if all messages received on this account should be associated with a single action.
 - Individual actions** – Select this option if this account receives responses to multiple actions. In this case, specify each action name alongside the response value in the decision table grid.
- Optional:** If you selected the **One action** option, use the **Action** field to specify the action with which all incoming messages should be associated.
- Select a decision table and click **Add decision table**.
- For each decision table result, select the response value and, if applicable, an action name.

Add a decision table
 Add decision table

Name	Offer	Response
InboundSMSParsingSample1		
• Interested	ProcessInboundOffers	Accept ▼
• Not Interested	ProcessInboundOffers	Reject ▼
• Maybe	ProcessInboundOffers	Reject ▼

Add a decision table
 Add decision table

Name	Offer	Response
InboundSMSParsingSample1		
• Interested	ProcessInboundOffers	Accept ▼
• Not Interested	ProcessInboundOffers	Reject ▼
• Maybe	ProcessInboundOffers	Reject ▼

- Optional:** Select and add another decision table.
When an inbound message arrives, Pega Customer Decision Hub evaluates the decision tables from top to bottom until it finds a matching decision result.

Configuring SMS delivery receipts

You can enable delivery receipts for the SMS messages that Pega Customer Decision Hub sends out.

Note: If you enable SMS delivery receipts, the receipts will be processed and included in a Customer's Interaction History.

- In App Studio, click **Settings Channels SMS Outbound**.
- Edit or create an outbound SMS account for which you want to enable delivery receipts.
- On the Setup tab of the SMS account, select **Track delivery receipt**.
- Click **Settings Channels SMS Inbound**.
- Edit or create an inbound SMS account which corresponds to the outbound SMS account where you enabled delivery receipts.
The inbound account must have the same connection setup data as the outbound account.
- On the Response behaviors tab, specify what type of messages this inbound SMS account will receive.
 - Messages and delivery receipts** - The SMS account can receive incoming messages, as well as delivery receipts for SMS messages which Pega Customer Decision Hub sends to Customers.
 - Delivery receipts only** - The SMS account can receive only delivery receipts.

Defining push notifications

Pega Customer Decision Hub can send push notification messages to registered apps. Push notifications can be sent while running actions.

For more information, see [Configuring application settings for push notifications](#).

1. For every app, define the following information.
 - Name of the app - how the app is referred to within the Pega Customer Decision Hub application, for example, in action flows
 - Description
 - Platform of the app - select Android or iOS
 - Unique ID - used for validating push notification service requests, for example, for registering and un-registering app users
2. Define additional information.
 - For an Android app, define the Google key.
 - For an iOS app, define the following additional information.
 - App certificate - an Apple .p12 certificate
 - Certificate name - used for storing the certificate in the system
 - Certificate password - used to validate the certificate
 - Certificate type
 - Monitor Apple's notification feedback service?

Configuring push notifications

You can register an app so it can receive push notification messages from the Pega Customer Decision Hub application. Push notifications allow a server to send (push) a message to the client. For example, mobile apps can send notification messages to their users.

1. Log in to the Pega Customer Decision Hub portal.
2. Complete the instructions in [Configuring application settings for push notifications](#).

Specifying delivery time frames

You can specify delivery time frames for outbound SMS and email accounts. The time frames only apply to the delivery of SMS and email treatments in actions. Other delivery, such as work object-related correspondence, is not affected by these time restrictions.

1. In the **ACCOUNT AVAILABILITY** section, click the **Specify Range** button.
2. Select the start value **From** and the end value **To** of the time range.
3. Ensure that the correct time zone is selected from the drop-down list.
4. Click **OK**.

Enabling channels

Use the on/off toggles to indicate which inbound and outbound channels are active and should be considered for Next-Best-Action.

1. In the **Channels** tab of the Next-Best-Action Designer, in the **Channels** section, review the active toggles.
2. If you want to disable a channel for Next-Best-Action, set the related toggle to OFF.
If a channel is toggled off, its status is stored in a Decision decision data rule (DDR) and the related part of the strategy framework is turned off. Setting the toggle to on reactivates the related part of the strategy framework.

Adding additional channels to Next-Best-Action Designer

In addition to the built-in channels provided by Pega Customer Decision Hub, you can implement additional channels by extending the *ChannelSettings* decision data rule. For example, if you send out physical product catalogs to your customers, you can create a Direct Mail channel for these types of treatments.

1. In Dev Studio, click **Records Decision Decision data**.
2. Open the *ChannelSettings* decision data rule that resides in the *Data-Decision-Request-Customer* class.
3. Click **Check out**.
4. Click **New**.
5. Configure the channel details:
 - **Name**- For example, Direct Mail.
 - **Description**- An optional description of the channel, for example, Used for product catalogs related to the Household line.
 - **Channel icon**- An optional icon to visually identify the channel in Next-Best-Action Designer. Available icons include **pi-forward**, **pi-mail**, **pi-picture**, **pi-mobile-phone**, **pi-notification**, **pi-money**, **pi-headset**, and **pi-chat-typing**.

- **ModelName**- Enter NBA.
- **SettingName**- A unique identifier for the channel, for example, *DirectMail*.
- **SettingValue**- Set to *true* to enable the new channel, or *false* to create the channel without enabling it.

For more information about enabling channels, see [Enabling channels](#).

6. Save and check in your changes.

Result: The new channel now appears in the **Channels** tab of Next-Best-Action Designer. If it is enabled, you can also use it when configuring contact policies in the **Constraints** tab of Next-Best-Action Designer.

- [Creating treatments for additional channels](#)

Creating treatments for additional channels

After creating the new channel in Next-Best-Action Designer, you can configure treatments for that channel.

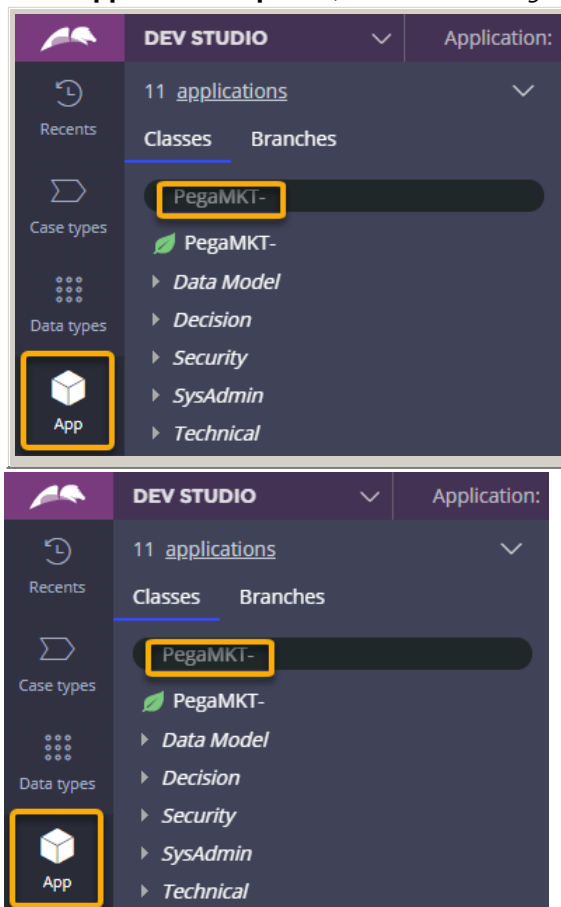
1. In the Pega Customer Decision Hub portal, click **Content Treatments**.
2. Click **Create Other**.
3. In the **Channel** list, select the new channel that you created.
4. Fill out all other fields required to configure the treatment.

What to do next: For more information about treatments, see [Defining treatments for actions](#).

Configuring additional channels for use with volume constraints

Volume constraints limit the number of actions which are offered to customers, for example, to comply with corporate or regulatory guidelines. Volume constraints can apply per action, per action property, or per channel. If the default channels do not fulfill your requirements, you can create additional channels for use with volume constraints.

1. Log in to Dev Studio.
2. In the **Application Explorer**, switch to the *PegaMKT*- class, as in the following figure:



3. Click **Data Model Field Value**.
4. Open a field value, for example, **Channel Email**.
5. Click **Save as**.
6. Enter a label for the new channel, for example, *Telemarketing*.
7. In the **Add to ruleset** list, select the ruleset for your custom rules. By default, this is *<your implementation application>-Rules*.
8. Click **Create and open**.

9. Click **Check out**.
10. In the **Translate To** field, enter the name of the new channel, as in the following figure:

Home Channel Email Channel Telem...

Edit Field Value: Telemarketing [Available]

CL PegamKT- ID Channel • Telemarketing R

Localized label Specifications History

Translate from
Telemarketing

To
Telemarketing

Extra whitespaces will be collapsed to single space by b

Home Channel Email Channel Telem...

Edit Field Value: Telemarketing [Available]

CL PegamKT- ID Channel • Telemarketing R

Localized label Specifications History

Translate from
Telemarketing

To
Telemarketing

Extra whitespaces will be collapsed to single space by b

11. In the **History** tab, in the **Description** field, enter a new description for the channel, for example, Field value for the Telemarketing channel.
12. Save and check in your changes.
13. In the **Application Explorer**, switch to the *PegaMKT-System-Setting* class.
14. Click **Technical Activity**.
15. Open the **SetChannelPropertyMappings** activity.
16. Click **Private edit**.
17. In the **Parameters** tab, add the new field value in one of the unused rows, as in the following figure:

Steps Parameters Pages & Classes Security Test cases Specifications History

Parameters

	Name	Description	Data type	Required	In/Out	Default value	Smartprompt type	Validate as
1	Channel1	Name of channel to be represented by Channel	String	No	In	CallCenter		
2	Channel2	Name of channel to be represented by Channel	String	No	In	DirectMail		
3	Channel3	Name of channel to be represented by Channel	String	No	In	Email		
4	Channel4	Name of channel to be represented by Channel	String	No	In	Facebook		
5	Channel5	Name of channel to be represented by Channel	String	No	In	SMS		
6	Channel6	Name of channel to be represented by Channel	String	No	In	Twitter		
7	Channel7	Name of channel to be represented by Channel	String	No	In	Passbook		
8	Channel8	Name of channel to be represented by Channel	String	No	In	Telemarketing		
9	Channel9	Name of channel to be represented by Channel	String	No	In			
10	Channel10	Name of channel to be represented by Channel	String	No	In			

Steps **Parameters** Pages & Classes Security Test cases Specifications History

Parameters

	Name	Description	Data type	Required	In/Out	Default value	Smartprompt type	Validate as
1	Channel1	Name of channel to be represented by Channel	String	No	In	CallCenter		
2	Channel2	Name of channel to be represented by Channel	String	No	In	DirectMail		
3	Channel3	Name of channel to be represented by Channel	String	No	In	Email		
4	Channel4	Name of channel to be represented by Channel	String	No	In	Facebook		
5	Channel5	Name of channel to be represented by Channel	String	No	In	SMS		
6	Channel6	Name of channel to be represented by Channel	String	No	In	Twitter		
7	Channel7	Name of channel to be represented by Channel	String	No	In	Passbook		
8	Channel8	Name of channel to be represented by Channel	String	No	In	Telemarketing		
9	Channel9	Name of channel to be represented by Channel	String	No	In			
10	Channel10	Name of channel to be represented by Channel	String	No	In			

Note: The parameter value is case-sensitive.

18. Click **Save**.

19. Click **Actions Run**.

Result: You can now use the new channel in your volume constraints and contact policies, as in the following figure:

Describe which Channels to Constrain

Limit the number of customers that can be contacted via specific outbound channels.

Enabled?	Channel	Reset Interval	Minimum Volume	Maximum Volume	Remaining
☒	Telemarketing	Manually	50		Unlimited

Describe which Channels to Constrain

Limit the number of customers that can be contacted via specific outbound channels.

Enabled?	Channel	Reset Interval	Minimum Volume	Maximum Volume	Remaining
☒	Telemarketing	Manually	50		Unlimited

Optional: Configuring additional channels for use with contact policies

Contact Policy rules use an internal mapping scheme to support up to 10 channels to which the contact policy can apply. This mapping is not externally exposed. However, you may need edit the mapping if you [created a new channel](#) and want to use it in your contact policies.

1. Log in to Dev Studio.
2. Review the default channel mappings by using the [Clipboard tool](#) to inspect the `channelPrefixMapping` property on the `Declare_MKTSysSystemSetting` node-level page.

Example channel mappings in the `channelPrefixMapping` property

Thread: Segments

Search within this thread

- User Pages
- Data Pages
 - Thread
 - Requestor
 - Node
 - D_CDHSUBJECTDictionary
 - D_ProspectCount (PegaMKT-Data-Prospect)
 - D_pyVLExemptions (@baseclass)
 - D_pzAutoUpdatePredictionsList (Code-Pega-List)
 - D_pzListTimezones (Code-Pega-List)
 - D_pzLoadPortalName
 - D_pzUsageEvent
 - D_RepositorySettings (Embed-PegaMKT-System)
 - Declare_AuthPolicies (Data-Admin-System)
 - Declare_MKTSysSetting (PegaMKT-System)
 - ClassKeys
 - DataRelationships

Data Page: Declare_MKTSysSetting

Edit Refresh Actions

Property	Value
AvailableBudget	13840104.81
channelPrefixMapping	{callcenter=Channel1, sms=Channel5, email=Channel3, push=Channel7, retail=Channel8}
ConfigName	Default
ContactSelectionMode	LIST
CustomerClassName	PegaMKT-Data-Customer
DBType	PostgreSQL
DecisionBatchSize	250
DefaultCustomerLatitude	.Latitude
DefaultCustomerLongitude	.Longitude
DisableCampaignOfferValidation	false
EmailResendAttempt	3
EnablePartition	false
ExtendedCustomerClass	true
ImplRuleSetName	BaseApp-Rules

Thread: Segments

Search within this thread

- User Pages
- Data Pages
 - Thread
 - Requestor
 - Node
 - D_CDHSUBJECTDictionary
 - D_ProspectCount (PegaMKT-Data-Prospect)
 - D_pyVLExemptions (@baseclass)
 - D_pzAutoUpdatePredictionsList (Code-Pega-List)
 - D_pzListTimezones (Code-Pega-List)
 - D_pzLoadPortalName
 - D_pzUsageEvent
 - D_RepositorySettings (Embed-PegaMKT-System)
 - Declare_AuthPolicies (Data-Admin-System)
 - Declare_MKTSysSetting (PegaMKT-System)
 - ClassKeys
 - DataRelationships

Data Page: Declare_MKTSysSetting

Edit Refresh Actions

Property	Value
AvailableBudget	13840104.81
channelPrefixMapping	{callcenter=Channel1, sms=Channel5, email=Channel3, push=Channel7, retail=Channel8}
ConfigName	Default
ContactSelectionMode	LIST
CustomerClassName	PegaMKT-Data-Customer
DBType	PostgreSQL
DecisionBatchSize	250
DefaultCustomerLatitude	.Latitude
DefaultCustomerLongitude	.Longitude
DisableCampaignOfferValidation	false
EmailResendAttempt	3
EnablePartition	false
ExtendedCustomerClass	true
ImplRuleSetName	BaseApp-Rules

In the figure above, channels 1, 3, 5, 7, and 8 are mapped. Channels 2, 4, 6, 9, and 10 are unused. **Note:** The above figure shows an example configuration. Channel mappings may be different in your application, so you must review the mappings on your system before proceeding.

3. Select an unused channel mapping that you can use to map the new channel, and take a note of how the channels are currently mapped.
4. Run the *SetChannelPropertyMappings* activity under the *PegaMKT-System-Setting* class and enter the current and new channel mappings as parameters for the activity.

Caution: The *SetChannelPropertyMappings* activity replaces the current channel map with a new map. The current channel mappings are not populated or validated by the activity. Because of that, you must enter the current channel mappings exactly as they exist in your system, to avoid issues resulting from incorrect mappings. The following figure shows the parameter values for the previously mapped channels 1, 3, 5, 7, and 8, as well as the new mapping for channel 9 (whatsapp).

Run Activity: SetChannelPropertyMappings

Clipboard

Trace

Run

CL PegaMKT-System-Setting ID SetChannelPropertyMappings

This activity can only be called from another activity because the "Allow invocation from browser" is disabled on the security tab.

Parameter		Value
Channel1	?	callcenter
Channel2	?	
Channel3	?	email
Channel4	?	
Channel5	?	sms
Channel6	?	
Channel7	?	push
Channel8	?	retail
Channel9	?	whatsapp
Channel10	?	

Run Activity: SetChannelPropertyMappings

Clipboard

Trace

Run

CL PegaMKT-System-Setting ID SetChannelPropertyMappings

This activity can only be called from another activity because the "Allow invocation from browser" is disabled on the security tab.

Parameter		Value
Channel1	?	callcenter
Channel2	?	
Channel3	?	email
Channel4	?	
Channel5	?	sms
Channel6	?	
Channel7	?	push
Channel8	?	retail
Channel9	?	whatsapp
Channel10	?	

- After the activity runs successfully, review the channel mappings in the [Clipboard tool](#) to confirm that the channel map has been correctly updated.
 - To ensure that the new channel is available as a selectable option on the Contact Policy rule, [create a new Field Value](#) with the following keys:
 - Identifier** - <your_channel_name>
 - Field Name** - Channel
 - Apply to** - PegaMKT-Work-Offer
- For example:**

Field Value: WhatsApp [Available]
CL PegaMKT-Work-Offer **ID** Channel • WhatsApp

Localized label

Specifications

History

Translate from

WhatsApp

To

WhatsApp

Field Value: WhatsApp [Available]
CL PegaMKT-Work-Offer **ID** Channel • WhatsApp

Localized label

Specifications

History

Translate from
WhatsApp

To

WhatsApp

7. Verify that the new channel shows in the list of available channels when you create a new contact policy, as in the following figure:

Policy Starts: 7/27/2021  Policy Ends:  ☒ Available

Please specify the maximum number of times a can be contacted per channel.

Channel	Contact(s) per	Duration	
☒ Call Center ☐ Email ☐ Push Notification ☐ Retail ☐ SMS ☒ WhatsApp		Daily 	

Policy Starts: 7/27/2021  Policy Ends:  ☒ Available

Please specify the maximum number of times a can be contacted per channel.

Channel	Contact(s) per	Duration	
☒ Call Center ☐ Email ☐ Push Notification ☐ Retail ☐ SMS ☒ WhatsApp		Daily 	

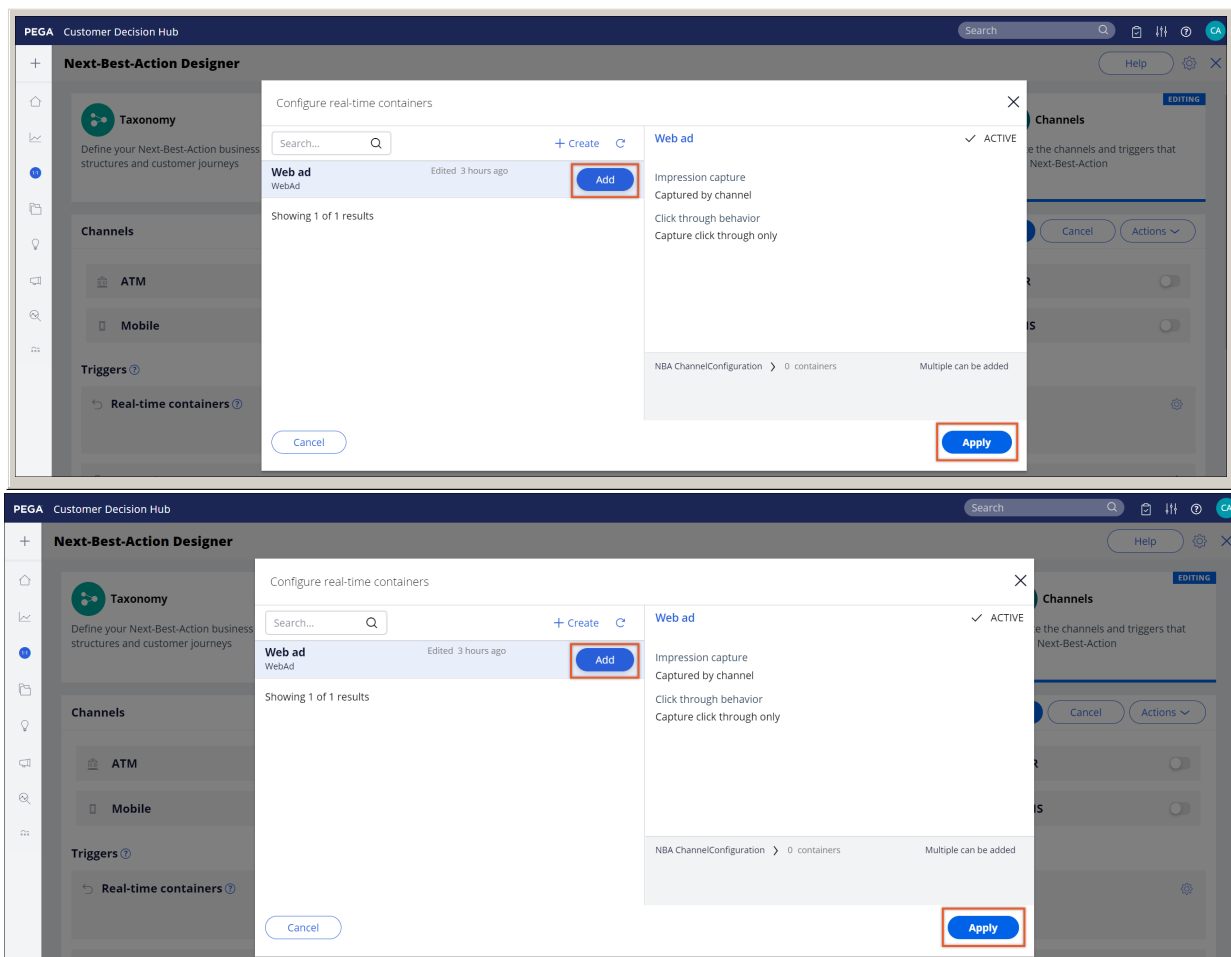
8. Save the contact policy and test the end-to-end behavior to verify that it works as expected.

Defining real-time container triggers

Real-time containers manage content in real-time channels. Adding a real-time container as a trigger enables it to be called by an external channel. When a container is loaded, for example, when an individual views an ad for an action in a predefined placement on the web channel, it triggers the next best action. For more information, see [Managing content in real-time channels with real-time containers](#).

1. Define a real-time container. For more information, see [Managing content in real-time channels with real-time containers](#).
2. In the **Channels** tab of the Next-Best-Action Designer, click **Edit**, and then in the **Triggers Real-time containers** section, click **Configure** icon.
3. Select the real-time container that you want to add as a trigger and click **Add Apply**.

Adding a real-time container as trigger



4. In the **Business structure level** list, associate the container with the business issue and group.
Note: Before adding real-time containers as triggers, work with your system architect to make sure your the design time configuration of your access group is pointing to highest (and only) unlocked artifacts ruleset version.
5. Click **Submit**.
6. Click **Save**.

Defining real-time event triggers

Define real-time events to respond to real-time customer engagement opportunities triggered by an external event. For example, you can define a real-time event which triggers Next-Best-Action after an individual signs up for a loyalty card with your retail store. For more information, see [Responding to real-time customer engagement with real-time event triggers](#).

1. Define a real-time event. For more information, see [Responding to real-time customer engagement with real-time event triggers](#).
2. In the **Channels** tab of the Next-Best-Action Designer, in the **Triggered by Events** section, click **Configure**.
3. Select the event that you created.
4. In the **Business structure level** list, select the business issue and group with which the event is associated.
5. Click **Submit**.
6. Click **Save**.

Configuring a recurring schedule for Next-Best-Action

Configure a recurring schedule for running Next-Best-Action using an audience that represents all customers and a daily schedule. The outbound schedule initiates any outbound actions, for example, campaigns or audience updates on external advertising platforms.

Note: This schedule applies across all issues and groups in your business hierarchy, and uses the top level customer context. If you want to define a schedule specific to a different context, or a particular issue or group, see step 7.

1. In the **Channels** tab of the Next-Best-Action Designer, in the **Primary schedule > Recurrence** section, set up a recurring schedule, start date, and end date for outbound Next-Best-Action.
2. To select the starting population, in the **Outbound schedule Starting population** section, click **Configure**.
Note: The starting population for Next-Best-Action is based on the top level customer context. For more information about configuring segments, see [Defining customer audiences with customer segments](#). In the current version of Pega Customer Decision Hub, prospect segments cannot be used as starting populations for Next-Best-Action.
3. To apply constraints to actions and avoid overexposure, specify the number of times that an action can be offered to a

group of customers, in the **Volume constraint** section, click **Configure**. For more information about volume constraints, see [Avoiding overexposure of actions with volume constraints](#).

4. Click **Save**.

Note: To review and adjust the outbound schedule, click **Actions Open outbound schedule**. This opens the underlying multi-channel campaign strategy. This action is restricted to administrator-type operators.

Result: The outbound schedule configuration is created in Draft mode.

5. To launch the outbound schedule, click **Actions Run**.

The Run action is only available after you configure the schedule.

6. To suspend an active outbound schedule, click **Actions Suspend**.

7. **Optional:** In addition to the primary schedule defined for your main customer context, you can define secondary schedules to run your marketing operations for different customer groups or contexts.

- a. In the **Channels** tab of the Next-Best-Action Designer, click **Actions Add schedule**.

- b. In the new **Additional schedule** section, in the **Issue / Group** list, select the issue and group or groups to which the schedule applies.

Note: As a best practice, you should launch all outbound actions at the top level of your Next-Best-Action Designer hierarchy. However, if required, you can select specific issues and groups to define additional outbound schedules that run your marketing operations at different times for different levels of your business hierarchy.

- c. In the **Context** list, select the customer context that you want to target. **For example:** For example, when targeting the customers at the account level, you might use a context like Household or Account, depending on the settings that you configured in the Context Dictionary.

Note: Choosing context other than top level customer context will result a warning, as using contexts below primary will impact engagement policies and might result in action failures.

- d. Select the starting population and optionally also the volume constraint, as described in steps 2 and 3.

Result: After an outbound run is launched, you can click the **Run details** link to view details about the campaign data flow. For more information about run details and scheduled runs, see [Viewing Run Details](#) and [Run schedule](#).

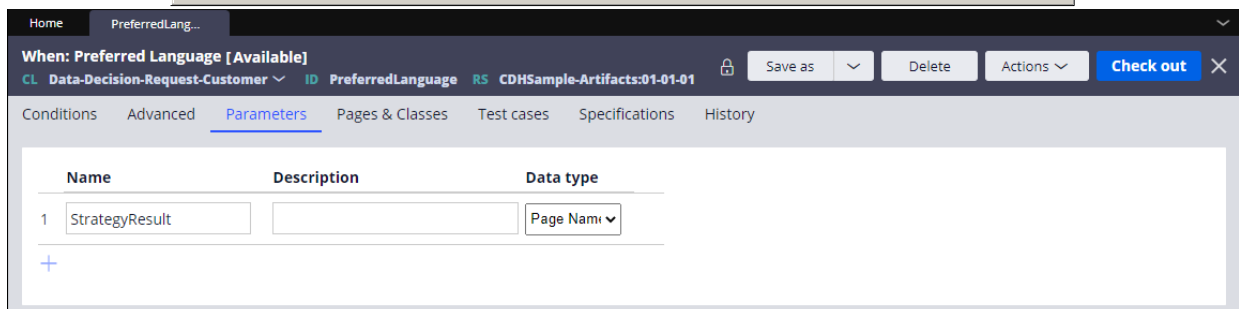
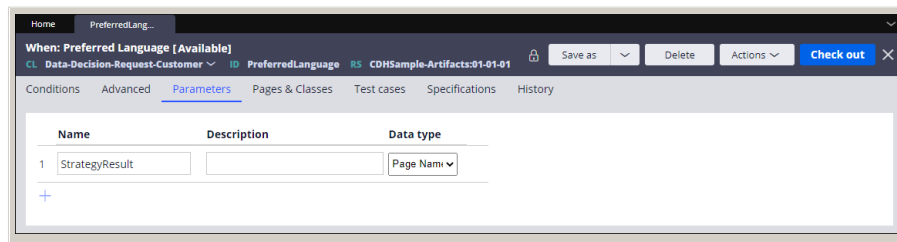
Creating treatment eligibility criteria

To enable treatment eligibility criteria in Pega Customer Decision Hub, create an associated When rule and mark it as a relevant record.

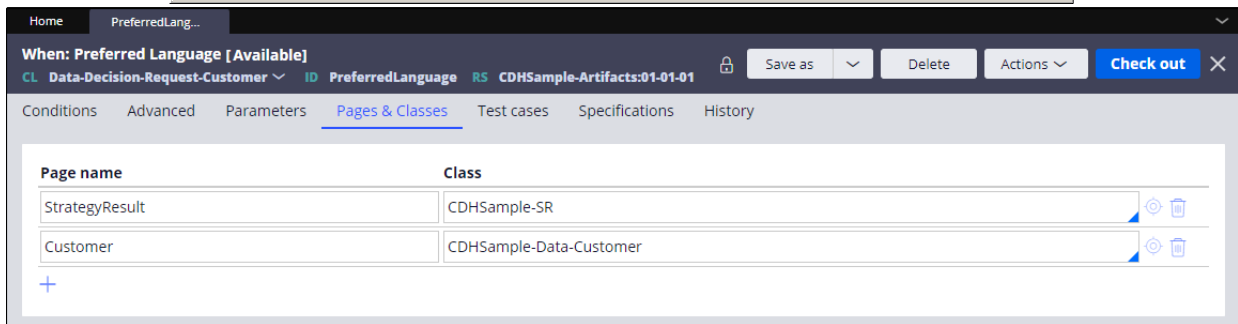
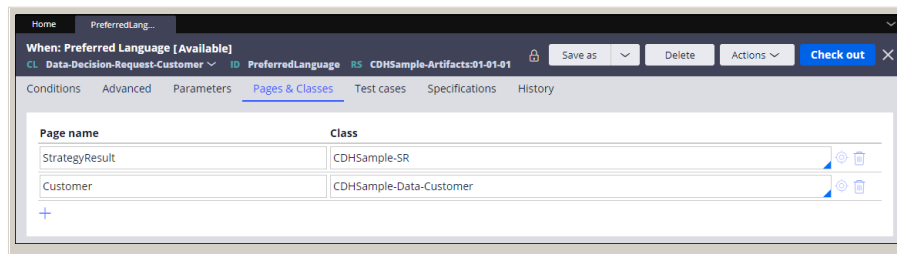
When defining your eligibility criteria, you will need to access customer-level properties, as well as properties from your strategy result (SR) class. In order to access these properties within eligibility conditions, you must add a page for each.

Tip: Do not use treatment eligibility rules for targeting treatments to customers. Treatment targeting is better handled by the Arbitration process. Treatment eligibility is intended for eligibility rules such as Treatment is in a customer's preferred language, customer has opted into this channel, customer has a downloaded the mobile app, and so on.

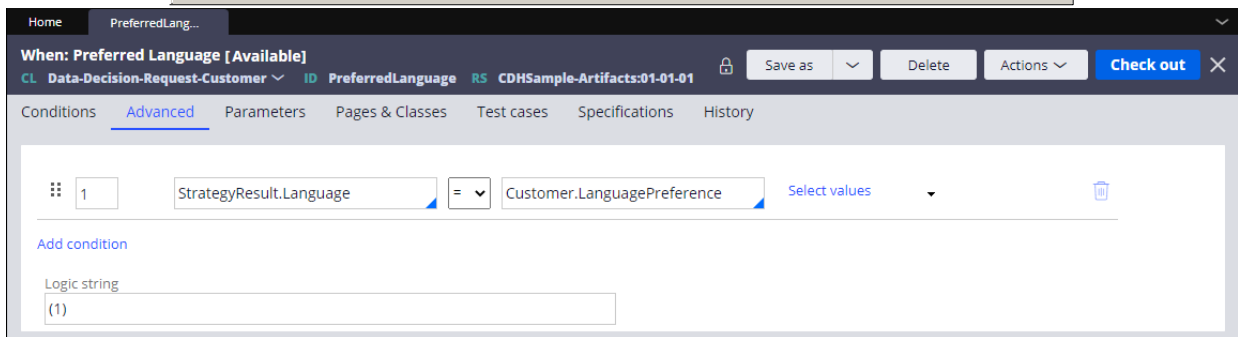
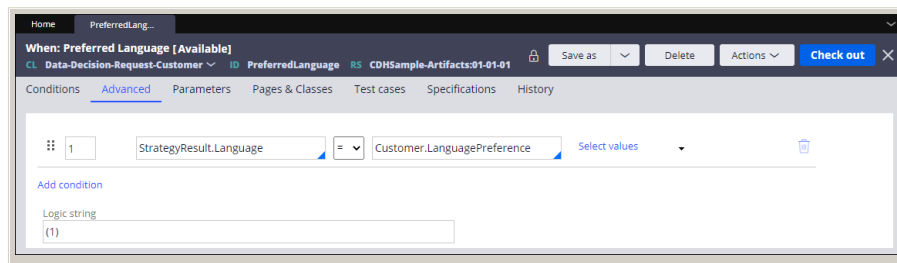
1. In Dev Studio, create a When rule in the *Data-Decision-Request-Customer* class.
2. In the **Parameters** tab, add a parameter named StrategyResult, with **Page Name** as the data type.



3. In the **Pages & Classes** tab, add a page named StrategyResult on the top-level SR class.
Note: Strategy results (SR) represent the next-best-action recommended actions and treatments.
4. In the **Pages & Classes** tab, add an entry for each context entity that you [defined in the Context Dictionary](#).
Note: For the names, use the value specified in the **Singular name** field in the Context Dictionary.



5. In the **Advanced** tab, define the conditions for your rule. **For example:** To match the customer's language preference with the language of the treatment, enter StrategyResult.Language = Customer.LanguagePreference.



6. Mark the When rule relevant by clicking **Actions Mark as relevant record**.
7. Open the **Relevant Records** landing page.
Tip: After marking the When rule relevant, click the **View** link in the confirmation message shown in the header.
8. In the **Relevant Records** landing page, enter Data-Decision-Request-Customer as the class name.
9. Find the When rule that you marked as relevant and click the Menu icon to the right of the When rule row.
10. Click **Assign categories Treatment Eligibility**.

Result: Your treatment eligibility When rule now appears in the **Treatment eligibility** dropdown list.

Optional: Configuring repository storage for marketing images

You can configure Pega Customer Decision Hub to use repository storage for your Image Library.

The Image Library is used to store and maintain marketing digital assets. By default, the assets are stored in a traditional rule-based Image Library. You can instead configure repository storage for your images. Using repository storage for your Image Library gives you the integrated ability to manage and distribute marketing images, improves usability, and simplifies the launch of email campaigns at scale.

1. [Configuring custom repository storage](#)
2. [Configuring default repository storage for Pega Cloud deployments](#)

Configuring custom repository storage

For all cloud or on premise implementations you can enable repository storage with your existing S3 repositories (S3, Azure,

or local file storage).

Before you begin: Make sure that the IP of the Content Delivery Network you want to use for repository storage is allowed to access the Pega Web Service, so that the Content Delivery Network can serve images from the Web Service.

1. Configure the repository connection.
For more information, see [Creating a repository connection](#) or [Creating custom repository connections](#).
2. In Dev Studio, configure the following dynamic system settings:
 - **MarketingImageAssets/IsRepositoryStorageEnabled** - Set to true.
 - **MarketingImageAssets/CustomRepositoryName** - Enter the name of the repository rule which points to your repository. This setting overwrites the value of the setting **MarketingImageAssets/DefaultRepositoryName**.
 - Optional: **MarketingImageAssets/Folder** - Enter the name of the folder where Pega Customer Decision Hub will store the image assets. This folder will be located inside the root folder defined in the repository rule. If you do not define a folder for the images, they will be stored directly in the repository root folder.
 - Optional: **MarketingImageAssets/CustomBaseURL** - Enter the URL of the Content Delivery Network which you configured, and update the **MKTDefaultSecured** content security policy rule to include **CustomBaseURL** under the **Image-Source** category in the **Allowed Websites** column. If you leave this setting blank, the images will be rendered directly from the Pega Web Service. This setting overwrites the value of the setting **MarketingImageAssets/BaseURL**.
3. Optional: To modify the maximum file size allowed for images, edit the **MarketingImageAssets/AllowedFileSize** Dynamic System Setting.
4. Optional: To allow image types other than .png, .jpg, .jpeg, .tiff, and .heif, save the data transform **AllowedImageTypes** to your application ruleset and add new types as required.
5. Run the **D_RepositorySettings** data page with flush mode on, or clear the cache.
Flushing the DSS settings is required to apply the changes made to the settings.

What to do next: If you replace an image in the repository with a new version, the cached image on the Content Delivery Network must be invalidated and replaced with the new version of the image. Cache invalidation is automatic in the Pega Cloud repository, but when using your own S3 repository, you must configure the invalidation logic, which should be applied on a DELETE event to clear the image cache. For more information, refer to the documentation of your S3 repository.

Configuring default repository storage for Pega Cloud deployments

For Pega Cloud deployments, you can enable and configure the default repository storage.

1. In Dev Studio, configure the following dynamic system settings:
 - **MarketingImageAssets/IsRepositoryStorageEnabled** - Set to true.
 - Optional: **MarketingImageAssets/Folder** - Enter the name of the folder where Pega Customer Decision Hub will store the image assets. This folder will be located inside the root folder defined in the repository rule. If you do not define a folder for the images, they will be stored directly in the repository root folder.
 - Optional: **MarketingImageAssets/BaseURL** - Enter the URL of the Content Delivery Network which you configured, and update the **MKTDefaultSecured** content security policy rule to include **BaseURL** under the **Image-Source** category in the **Allowed Websites** column. If you leave this setting blank, the images will be rendered directly from the Pega Web Service.
2. Optional: To modify the maximum file size allowed for images, edit the **MarketingImageAssets/AllowedFileSize** Dynamic System Setting.
3. Optional: To allow image types other than .png, .jpg, .jpeg, .tiff, and .heif, save the data transform **AllowedImageTypes** to your application ruleset and add new types as required.
4. Run the **D_RepositorySettings** data page with flush mode on, or clear the cache.
Flushing the DSS settings is required to apply the changes made to the settings.

What to do next: If you replace an image in the repository with a new version, the cached image on the Content Delivery Network is invalidated and replaced with the new version of the image. Cache invalidation is automatic in the Pega Cloud repository. The request to invalidate the cache is made immediately, but may take some minutes to fully refresh the image cache.

Optional: Integrating Pega Customer Decision Hub with Adobe Experience Manager

Adobe Experience Manager (AEM) is a content management solution which provides users with an interface to build, deliver, and maintain digital experiences. Use the Pega AEM Connector to connect your Pega actions to Adobe Experience Manager assets, so that Adobe Experience Manager assets can be used for customer engagement.

You can navigate the Adobe Experience Manager folder structure and browse Adobe Experience Manager assets from inside Pega Customer Decision Hub. The Pega AEM Connector uses the Query Builder API provided by Adobe. For more information about the API, see the Adobe API documentation.

1. Download the Pega AEM Connector from Pega Exchange.
2. To open the Application rule form, in Dev Studio, click the down arrow by the name of your application, then click **Definition**.
3. On the Definition tab, in the Enabled components section, click **Manage components**.
4. Click **Install new**, select the file that you downloaded from Pega Exchange, and click **Open**.
5. Select the **Enabled** checkbox to enable this component for your application, and click **OK**.

6. In the list of enabled components, verify that the Pega AEM Connector is selected, and save your changes to the rule form.
7. In Dev Studio, click **Records SysAdmin Repository**.
8. Click **Create**.
9. Enter a description and a name for your repository.
10. Click **Create and open**.
11. In the **Definition** tab, click **Select**.
12. Select **Adobe Experience Manager** as the repository type.
13. **Optional:** By default, Pega Customer Decision Hub can create and delete files and folders in Adobe Experience Manager. If you want to disable this functionality, select **Read-Only**.
14. In the **Repository configuration** section, in the **Publish host name** field, enter the host name of your Adobe Experience Manager repository.
Pega Customer Decision Hub will use this host name to access Adobe Experience Manager.
15. **Optional:** If you want to automatically publish content to Adobe Experience Manager, enter the Author host name of your Adobe Experience Manager repository in the **Author host name** field, and select **Auto Publish**. The **Author host name** field is only available if you did not set the Adobe Experience Manager repository to Read Only in step 13.
16. In the **Root path** field, enter the name of the folder where Adobe Experience Manager assets are stored. The Pega AEM connector will allow you to browse the assets in this folder from inside Pega Customer Decision Hub.
Note: The Root folder path must not start with the / character.
17. In the **Authentication** section, in the **Authentication profile** field, enter the name of a new authentication profile and click the **Open** icon to configure the profile.
The authentication profile stores the credentials that Pega Customer Decision Hub needs to authenticate with the Adobe Experience Manager API.
18. In the **Type** field, select **Basic**.
19. Enter a name and description for your authentication profile.
20. Click **Create and open**.
21. Enter the user name, password, realm and host name required to authenticate with Adobe Experience Manager. For more information, refer to the Adobe API documentation.
22. Select **Preemptive authentication**.
23. Click **Save**.
24. On the **Definition** tab of your repository, click **Test connectivity**. If there are any errors, ensure that the credentials in the authentication profile are correct, and that Pega Customer Decision Hub can access the URLs you entered in the **Publish host name** and **Author host name** fields.
25. Click **Save**.
26. Click **Records SysAdmin Dynamic System Settings**.
27. Edit the following Dynamic System Settings:
 - *MarketingImageAssets/IsRepositoryStorageEnabled* – Set the value to True
 - *MarketingImageAssets/CustomRepositoryName* – Enter the name of the repository you created
 - *MarketingImageAssets/Folder* – Enter a valid sub-folder of the Root folder. If you leave this field blank, the Root folder is used as the destination folder for your images.
28. Flush the *D_RepositorySettings* data page to make sure the configuration changes are applied.
 - a. Click **Records Data Model Data Page**.
 - b. Open the *D_RepositorySettings* data page.
 - c. On the **Load Management** tab, click **Clear data page**.
 - d. Select **Flush All**.
 - e. Click **Submit**.

What to do next: Enable access to images from AEM, by updating the content security policy. For more information, see [Allowing Pega Customer Decision Hub to access images from Adobe Experience Manager](#).

Allowing Pega Customer Decision Hub to access images from Adobe Experience Manager

Images stored in Adobe Experience Manager are treated as images from an unknown source by Pega Customer Decision Hub. To allow your application to access an image from Adobe Experience Manager, add the Adobe Experience Manager host in the Content security policy.

1. In Dev Studio, click the down arrow by the name of your application, then click **Definition**.
2. Click **Security**.
3. In the Content security section, open the content security policy.
4. Expand the **Image-Source** section.
5. In the Allowed Websites table, click **Add a row**.
6. Enter the URL of your Adobe Experience Manager host.
7. Save the content policy.

Optional: Implementing Paid Media Manager

Paid Media Manager extends Next-Best-Action to modern digital advertising platforms, such as Facebook Ads and Google Ads. Each individual's Next-Best-Action is communicated through audiences on ad platforms. To implement Paid Media Manager in your organization, you must start by configuring the connection to the paid destination.

1. [Configuring CRM-based connections to paid destinations](#)

2. [Configuring web-based and DMP connections to paid destinations](#)
3. [Generating Next-Best-Action web audiences with Paid Media Manager](#)
4. [Enabling AI-driven advertising with paid click capture on Facebook Ads, Google Ads, and other platforms](#)
5. [Enabling the offline conversion job scheduler for Paid Media Manager](#)
6. [Enabling learning from own-channel models for the Paid channel](#)
7. [Enabling learning from Paid-specific models for the Paid channel](#)
8. [Configuring the PaidMediaSync queue processor to optimize performance](#)
9. [Defining triggers for paid media](#)

Configuring CRM-based connections to paid destinations

Paid Media Manager uses the CRM-based matching APIs for Facebook Ads, Google Ads, and LinkedIn Ads to synchronize paid audiences. To set up a connection, you must configure both Paid Media Manager and the paid destination to which you want to connect.

- [Mapping individuals to paid destinations with CRM attributes](#)
- [Configuring Facebook Ads destinations](#)
- [Configuring Google Ads destinations](#)
- [Configuring LinkedIn Ads destinations](#)

Mapping individuals to paid destinations with CRM attributes

Facebook Ads, Google Ads and LinkedIn Ads destinations enable you to target individuals based on their personally identifiable information (PII). To do so, map destination API properties to Pega Customer Decision Hub properties. For example, you can map **Email (EMAIL)** to .pyEmail1.

While creating a paid destination, in the **Field mapping** section, map the API parameters to Pega Customer Decision Hub properties. At least one mapping is required. You can map email, phone number, and device ID API parameters to multiple Pega Customer Decision Hub properties. In this way, you can specify multiple emails or phone numbers for each individual. Mapping multiple emails or phone numbers makes it possible to generate higher match rates and reach more individuals.

You can use the following parameters to target individuals:

Parameter	Encoding for Facebook Ads	Encoding for Google Ads	Additional notes	Example
Email address	- Leading and trailing spaces removed - Converted to lower case - Hashed with the SHA256 algorithm	- Leading and trailing spaces removed - Converted to lower case - Hashed with the SHA256 algorithm	Must be in lower case.	- example@email.com - test@email.com
Phone number	- If the <i>Country</i> property is filled out, the phone number is converted to the E164 format. If the <i>Country</i> property is empty, it is assumed that the country code is included in the phone number - All characters except for +, alphabetical characters, and numbers removed - Leading zeroes and non-numeric characters removed - Leading and trailing spaces	- If the <i>Country</i> property is filled out, the phone number is converted to the E164 format. If the <i>Country</i> property is empty, it is assumed that the country code is included in the phone number - All characters except for +, alphabetical characters, and numbers removed + added as a prefix if not already present - Leading and trailing spaces removed	Must be in E164 format.	+12345678910 +01987654321

Parameter	- Encoding removed for Facebook Ads - Converted to lower case	Hashed with the SHA256 algorithm	Additional notes	Example
	Hashed with the SHA256 algorithm			
First name	- Converted to lower case - Numbers and punctuation removed - Special characters converted to UTF-8 - Leading and trailing spaces removed - Hashed with the SHA256 algorithm	- Converted to lower case - All characters other than alphabetical characters and accents removed - Hashed with the SHA256 algorithm	Must not include prefixes, such as Dr.	- john - zoë
Last name	- Converted to lower case - Numbers and punctuation removed - Special characters converted to UTF-8 - Leading and trailing spaces removed - Hashed with the SHA256 algorithm	- Leading and trailing spaces removed - Converted to lower case - All characters other than alphabetical characters and accents removed - Hashed with the SHA256 algorithm	Must not include suffixes, such as Jr.	- smith - smith-jones - pérez
Country	- Converted to upper case - If the input is a valid country name, it is converted to a two-letter country code in ISO 3166-1 alpha-2 format. If the input is already a two-letter ISO 3166-1 alpha-2 code, it is left unchanged. If it is not a valid country name, it is also left unchanged - Leading and trailing spaces removed - Converted to lower case - Hashed with the SHA256 algorithm	- Converted to upper case - If the input is a valid country name, it is converted to a two-letter country code in ISO 3166-1 alpha-2 format. If the input is already a two-letter ISO 3166-1 alpha-2 code, it is left unchanged. If it is not a valid country name, it is also left unchanged - Leading and trailing spaces removed	Must be a valid full country name or two-letter country code in ISO 3166-1 alpha-2 format, for example, United Kingdom OR GB	- us for United States - sg for Singapore
	- White spaces removed - If the Country property is set to United States, the first five	Leading and	Must be a valid and	

Parameter	Encoding for Facebook Ads	Encoding for Google Ads	Additional notes	Example
ZIP code	characters of the code are stored, and the others removed. For other countries, the full code is stored - Converted to lower case - Leading and trailing spaces removed - Hashed with the SHA256 algorithm	trailing spaces removed - All characters other than alphabetical characters and spaces removed - Converted to upper case	verified postal code. For United States, use five digits followed by a four-digit extension, for example, 94303-5300. For the United Kingdom, use the Area/District/Sector format.	94303 94303-5300 - WC2H 8LG
Mobile device id	Not applicable	No formatting applied	Not applicable	75d147d6-xx5e-11eb-8529-0242ac130003
Gender	- If the input is female or F, it is converted to f - If the input is male or M, it is converted to m - Leading and trailing spaces removed - Converted to lower case - Hashed with the SHA256 algorithm	Not applicable	Not applicable	- male - female - m - f
Date of birth	- Parsed using the format dd/MM/yyyy, then split by the / character to extract day, month, and year - Leading and trailing spaces removed - Converted to lower case - Hashed with the SHA256 algorithm - Hashed day, month, and year sent to Facebook	Not applicable	Must have the DDMMYYYY (for example, 20201231) or DDMMYYYYTHHmmSS (for example, 20201231T101010) format	16/09/1980 16-09-1980 16091980
	- If the field is filled out: - Leading and trailing spaces removed - Converted to lower case - Numbers and punctuation removed - Special characters converted to UTF-8 format - Hashed with the SHA256			

Parameter	Encoding for Facebook	Encoding for Google	Additional notes	Example
First name initial	• If the field is blank: ◦ The First name property is used instead ◦ Converted to lower case ◦ Numbers and punctuation removed ◦ First character of the name taken and converted into lower case ◦ Hashed with the SHA256 algorithm	Not applicable	Not applicable	• Mr. • Example • Dr.
State	• For United States, the two-character ANSI code is used instead of the state name. For other countries, the full name is stored • Converted to lower case • Non-alphabetical characters removed • Leading and trailing spaces removed • Hashed with the SHA256 algorithm	• Leading and trailing spaces removed • Converted to lower case • Non-alphabetical characters removed	Not applicable	• California
City	• Converted to lower case • Non-alphabetical characters removed • Leading and trailing spaces removed • Hashed with the SHA256 algorithm	• Leading and trailing spaces removed • Converted to lower case • All characters except for alphabetical characters, spaces, and accents removed	Not applicable	• New York • Chicago
Mobile advertiser ID (MADID)	• Leading and trailing spaces removed • Converted to lower case	Not applicable	Not applicable	• 75d147d6-xx5e-11eb-8529-0242ac130003

After the connection is established, a one-way hashed PII is sent directly from Pega Customer Decision Hub to the ad platforms through a secure connection. If the one-way hashed PII matches what the ad platform knows about the individual, the information about the individual is updated, for example, by adding the individual to a new audience, or communicating a conversion outcome. The hashed PII values are then discarded. The one-way hash cannot be reversed to expose PII values to the paid destination. In this way, the individuals' PII is not exposed and the one-way hash is only used to update the audience membership on the paid destination.

After an anonymous link is established, Paid Media Manager can add and remove individuals based on the actions that have

been assigned to them by Next Best Action in near real-time. It can also use the same mechanism to communicate data about customer conversions to the destination platform. Because people-based updates do not rely on website cookies, they can be triggered by scheduled runs, real-time container calls, or action responses.

Configuring Facebook Ads destinations

Facebook Ads destinations enable ads to be targeted on Facebook, Instagram, Messenger, and third party apps and websites. Paid Media Manager can leverage information on known individuals (for example, their email, phone number, device ID, and so on) to target these ads based on Next-Best-Action, and also segment eligibility. Paid Media Manager integrates directly with Facebook Ads and the Facebook Custom Audience API.

1. Create a Facebook App to access the Facebook Marketing API. For more information, see [Creating a Facebook Ads app and credentials](#) and [Submitting a Facebook app for review](#).
 2. Configure the connection to a Facebook Ads destination in Pega Customer Decision Hub. For more information, see [Configuring Facebook Ads destinations in Paid Media Manager](#).
- [Creating a Facebook Ads app and credentials](#)
 - [Submitting a Facebook app for review](#)
 - [Configuring Facebook Ads destinations in Paid Media Manager](#)

Creating a Facebook Ads app and credentials

As part of the process of connecting Paid Media Manager to a Facebook Ads destination, you must have an account in Facebook Ads and use the credentials for that account to configure a destination in Paid Media Manager.

1. If you do not already have a [Facebook Business Manager](#) account, create one.
2. To add your Facebook page to your Facebook Business Manager account, in Facebook Business Manager, click **Business Settings Accounts Pages** , and then add an existing page or create a new one.
For more information, see [Adding a page to Business Manager](#).
3. To add an ad account, in Facebook Business Manager, click **Business Settings Accounts Ad accounts** , and then select an existing page or create a new one.
For more information, see [Adding ad accounts in Business Manager](#).
4. To enable API integration for the Facebook Ads account, create a Facebook app.
 - a. Log in to [Facebook for developers](#).
 - b. Click **My Apps Add New App** .
 - c. Fill in the necessary details and create a Facebook app.
 - d. Log in to [Facebook Business Manager](#).
 - e. Click **Business Settings Accounts Apps** .
 - f. Click **Add**, and then add the app that you created.
5. To use the API integration to create artifacts in the ad account, link the Facebook app to an ad account.
 - a. Log in to [Facebook for developers](#).
 - b. Click **My Apps <App Name>** .
 - c. Click **Settings Advanced** .
 - d. Select the ad account that you linked with your Facebook Business Manager account in step 3.
 - e. Click **Save changes**.
6. Generate the access token which Paid Media Manager requires to connect to a Facebook Ads destination.
 - a. Log in to [Facebook for developers](#).
 - b. Click **My Apps <App Name>** .
 - c. Expand the **Products** header.
 - d. In the **Add a product** section, select **Marketing API**.
 - e. Click **Marketing API Tools** .
 - f. In the **Select Token Permissions** section, select the required permissions, and then click **Get Token**.
 - g. Copy the access token.

You need the token to continue setting up the Paid Media Manager connection to the Facebook Ads destination.
For more information, see [Configuring Facebook Ads destinations in Paid Media Manager](#).

Submitting a Facebook app for review

Facebook requires that third-party apps undergo a review process to ensure the safety of Facebook users' data. The following article describes how to generate the user access token by selecting relevant permissions and get the appropriate access levels for these permissions. For general information about the process, see [App Review](#).

Note: Facebook periodically changes the review process. Contact your Facebook account representative to help expedite the process.

- [Facebook Ads access levels and permissions](#)
- [Applying for advanced access permissions for the Lead Ads functionality](#)

Facebook Ads access levels and permissions

The app used for integration with Paid Media Manager should have the **App Type** set to **Business**. Business apps do not have [app modes](#) and instead use only [access levels](#).

Facebook Ads apps have two possible access levels: Standard and Advanced. When creating the app for Paid Media Manager integration, you can use the Standard access levels to generate the access token and test the app. By default, on the creation of the Business type app, all the permissions and features are provided with a Standard access level. If you are using the Lead Ads functionality, after you complete the testing, you must apply for the Advanced access level before going live with the integration.

Mandatory permissions

To integrate Paid Media Manager with Facebook Ads, you must obtain the following permissions:

- [ads_management](#) - Required to manage the ad accounts, audiences, campaigns, ad sets, and insights.
- [business_management](#) - To generate custom audiences, the ad account must be linked to the Business Manager account. This permission is required in order to validate this linking.

Mandatory feature

- [Ads Management Standard Access](#) - This feature is required to lower the rate limiting. Using a large customer base can result in a high volume of API calls, which causes Facebook to apply rate limiting if your app does not have access to the Ads Management Standard Access feature. **Note:** Rate limiting is not affected by the access level of permissions.

Optional permissions for the Lead Ads functionality

In order to use the Lead Ads functionality, your Facebook app must have the following permissions:

- [leads_retrieval](#) - Allows Paid Media Manager to retrieve the leads generated in Facebook.
- [page_manage_ads](#) - As per Facebook permission guidelines, the permission is used in parallel to the *leads_retrieval* permission. It is not explicitly used anywhere in Paid Media Manager.
- [pages_show_list](#) - To use lead ads, your Facebook app must be subscribed to the Facebook page. This permission is required for the subscription.
- [pages_read_engagement](#) - Required to subscribe the Facebook app to the Facebook page, and to retrieve the leads generated in Facebook.
- [pages_manage_metadata](#) - Required to subscribe Facebook app to the page, and for Facebook to send the leads to the Paid Media Manager service.

Applying for advanced access permissions for the Lead Ads functionality

To use the Lead Ads functionality, you must apply for advanced access permissions for your Facebook app before going live with the integration. To get the advanced access level for any permissions, you must complete the Facebook app review process.

1. Log in to your [App Dashboard](#).
2. Select your Paid Media app.
3. In the left navigation panel, click **App review Permissions and features**.
4. Find the following permissions:
 - *leads_retrieval*
 - *pages_show_list*
 - *pages_manage_ads*
5. If the **Ads API Access level** for a permission is set to **No access**, click **Get Standard Access** and confirm your identity.
6. If you already have standard access for a permission, click **Request Advance Access**.
7. Click **Continue the Request**.
8. In the **Complete App Verification** section of the **Request for App Review** window, click the **Provide verification details** icon.
9. In the **App Verification Details** window, provide the following information:

This is not a customer-facing app. This is a server-to-server integration and this system can only be accessed by internal employees for the purposes of advertising to Pega customers.

This Pega Customer Decision Hub-based application generates Facebook Ads artifacts to target customers with offers selected by the Pega Customer Decision Hub-based system. A customer may be selected to receive an offer for a new Credit Card A. We then create a custom audience and ad set for Credit Card A and associate them. Whenever a customer is selected for the Credit Card A offer they will be placed in that custom audience. If they are later selected for a Credit Card B offer the individual will be removed from the Credit Card A custom audience and placed in the Credit Card B custom audience. If an audience and ad set does not yet exist for that offer, it will be generated.

In another use case, the Pega Customer Decision Hub-based application generates tactic recommendations by pulling the ad performance data and analyzing it. Advertisers might have a campaign and ad set for a credit card with generic targeting for which the Pega Customer Decision Hub-based application can provide tactic recommendations to include the eligible customers list into the ad set targeting to improve the ad spend.

10. Click **Save**.
11. In the **How will your app use the Advanced Access leads_retrieval permission?** section, fill out the following details:

The application exchanges data directly with Facebook APIs only. We use this permission to retrieve the leads that were generated from Facebook. Sample API

CallGet Leads by IDsThis API is used to retrieve the Lead information for a Lead Gen ID.APINodeList<Lead> leads = Lead.fetchByIds(Lists.newArrayList(pageldLeadGenIdMap.values()), REQUEST_FIELDS, context);The prerequisite for retrieving leads from Facebook is configuring the webhook. We provide the necessary endpoint and the verify token in the Paid Media Manager channels configuration for Facebook.

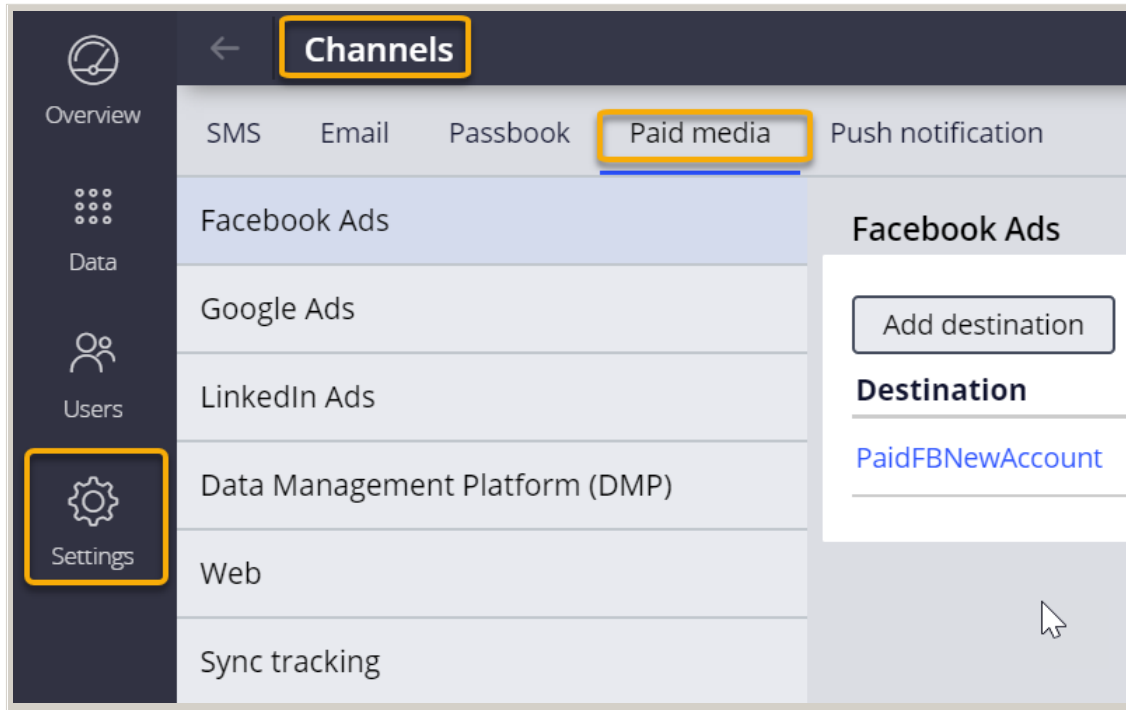
1. A user fills the details through the Facebook lead ad displayed.
 2. Upon filling, a lead is generated and will be retrieved by our REST endpoint.
 3. We fetch the lead id and use it to fetch the details filled in the lead ad.
 4. We store the details as a prospect.
12. In the **How will your app use the Advanced Access pages_show_list permission?** section, fill out the following details:
The application exchanges data directly with Facebook APIs only. This permission is a dependent permission for "pages_manage_ads" which in turn dependent on "leads_retrieval" permission. As per the Facebook permission guidelines, the permission is used in parallel to the "leads_retrieval" permission. We are not explicitly using this permission anywhere in our application. We require this permission to enable the leads and retrieve the lead ad data that is submitted by the user.
13. In the **How will your app use the Advanced Access pages_manage_ads permission?** section, fill out the following details:
The application exchanges data directly with Facebook APIs only. This permission is a dependent permission for "leads_retrieval" permission. As per the Facebook permission guidelines, the permission is used in parallel to the "leads_retrieval" permission. We are not explicitly using this permission anywhere in our application. We require this permission to enable the leads and retrieve the lead ad data that is submitted by the user.
14. Click **Submit for Review**.

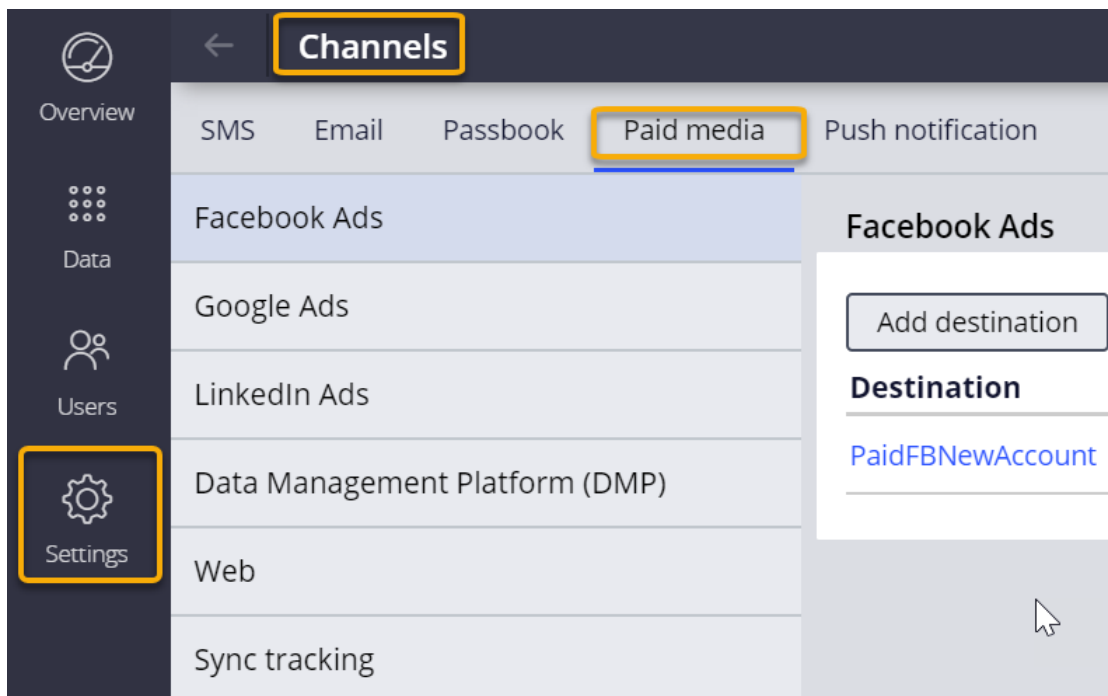
Configuring Facebook Ads destinations in Paid Media Manager

After you create a Facebook App to access the Facebook Marketing API, configure the connection to a Facebook Ads destination in Paid Media Manager.

1. Log in to Pega Customer Decision Hub as an operator with access to App Studio.
2. In App Studio, click **Settings Channels Paid Media Facebook Ads** , as in the following figure:

Accessing the Paid Media landing page





3. Click **Add destination**.
4. Enter the credentials that you created in Facebook Ads.
For more information, see [Creating a Facebook Ads app and credentials](#).
5. In the **Field mapping** section, map destination API properties to Pega Customer Decision Hub properties.
 - a. Map the **Facebook Ads Email (EMAIL)** parameter to a property.
 - b. **Optional:** Map additional PII properties, such as device ID, address, phone number or date of birth.
Note: Mapping multiple properties makes it possible to generate higher match rates and reach more individuals.
For more information about PII mapping, see [Mapping individuals to paid destinations with CRM attributes](#).
6. In the **Use External Id (EXTERN_ID)** field, specify a unique ID that can be used in place of hashed data for subsequent updates for a given individual.
This feature is enabled and set to use the class key from your Customer class by default. Keep in mind that disabling this setting will result in slower synchronization with Facebook.**Note:** The property used as external ID is not hashed during synchronization with Facebook. If you want to change it, select a property that can identify an individual uniquely without exposing their internal data.
7. Configure the additional settings in the **Facebook Ads assets generation** section:
 - **Enable campaign generation** - For more information, see [Streamlining Campaign Generation for Next-Best-Action Paid Audiences](#).
 - **Enable offline conversion push** - For more information, see [Enabling offline conversions](#).
 - **Lookalike audience country** and **Lookalike audience ratio** - For more information, see [Prospecting with Next-Best-Action paid audiences](#).
8. Click **Apply**.

Configuring Google Ads destinations

Google Ads Destinations enable ads to be targeted on Google Search, YouTube, Gmail, and other Google-owned media properties. Paid Media Manager integrates directly with Google Ads and the Google Customer Match API.

1. Create a Google Ads account and credentials. For more information, see [Creating a Google Ads account and credentials](#).
2. Configure the connection to a Google Ads destination. For more information, see [Configuring Google Ads destinations in Paid Media Manager](#).
 - [Creating a Google Ads account and credentials](#)
 - [Configuring Google Ads destinations in Paid Media Manager](#)
 - [Enabling device ID targeting for Google Ads](#)
 - [Migrating from Google AdWords API to Google Ads API](#)

Creating a Google Ads account and credentials

As part of the process of connecting Paid Media Manager to a Google Ads destination, you must have a manager account in Google Ads and use the credentials for that account to configure a destination in Paid Media Manager.

Creating a Google Manager Account to call the Google Ads APIs for production accounts

If you do not already have a Google Manager Account, create it, and then apply for access to the AdWords API.

1. Sign in to Google with a Google account that is not already linked to a Google Ads account.
2. Go to [Google Ads Manager Accounts](#).
3. Click **Create a Manager Account** and fill out the registration form.
4. In Google Ads, click **Tools & Settings Setup API Center**.

Google Ads API Center access

The image displays two screenshots of the Google Ads API Center access form. The top screenshot shows the form with the 'API Center' option highlighted in the 'TOOLS & SETTINGS' menu. The bottom screenshot shows the same form with the 'API Center' option highlighted in the 'TOOLS & SETTINGS' menu.

Google Ads API Center Access Form Fields:

- API contact email:** Enter an email that you regularly check
- Company name:** Enter the name of your company
- Company URL:** Enter your company URL

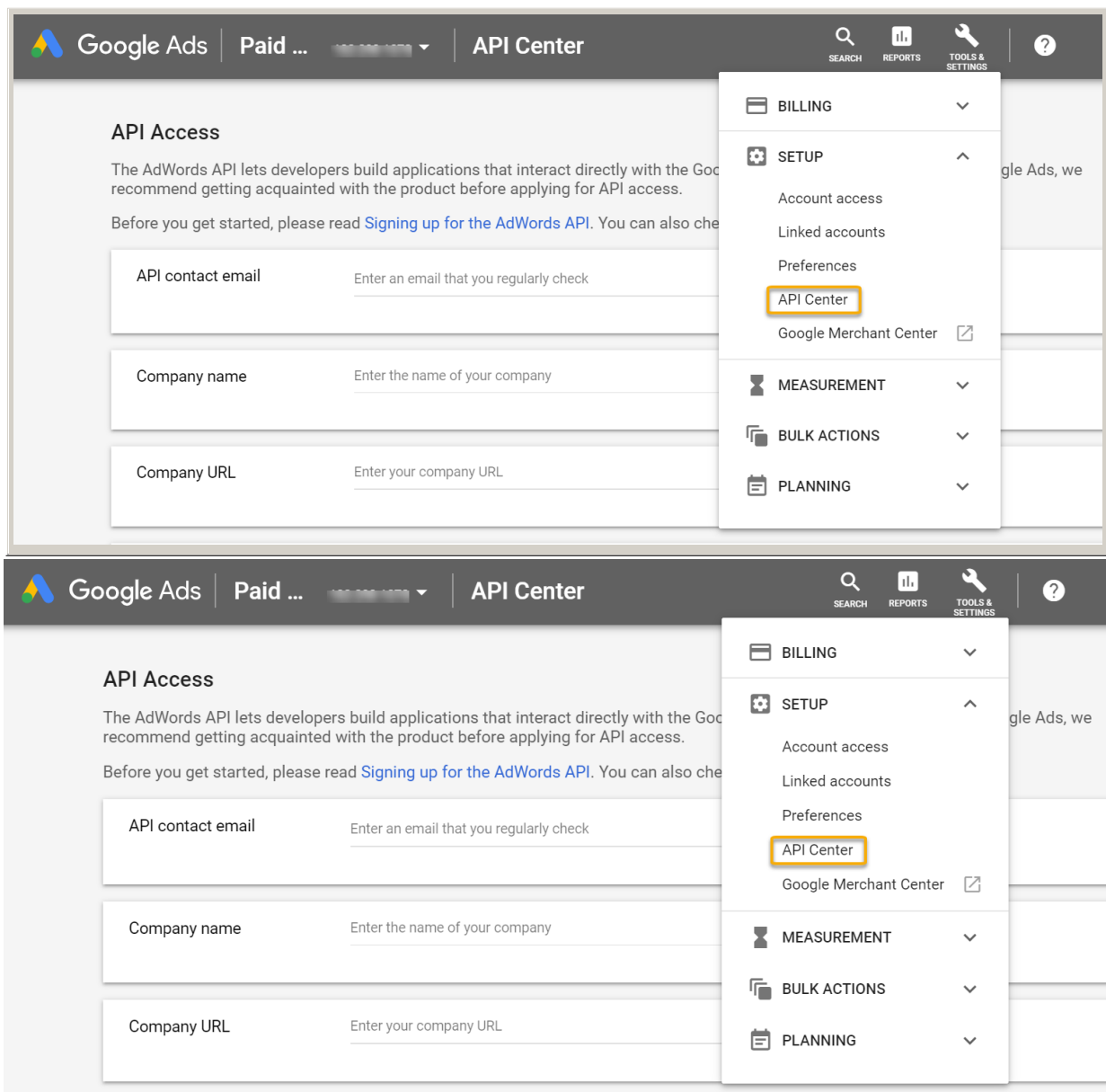
Google Ads API Center Access Form Instructions:

The AdWords API lets developers build applications that interact directly with the Google Ads API. We recommend getting acquainted with the product before applying for API access.

Before you get started, please read [Signing up for the AdWords API](#). You can also check out the [AdWords API documentation](#).

5. In the API Center, fill out the access form.
Note: In the **Company type** section, select **Advertiser**.
6. Click **Create Token**.
Make a note of the token. You will need it to configure the connection between Paid Media Manager and Google Ads later. For more information, see [Configuring Google Ads destinations in Paid Media Manager](#).
7. Go to the Google Ads overview, and then click **Accounts**.
8. Click **+** to create a new account or link an existing account.

Google Ads API Center access



You can use either a Google Ads account or an AdWords Express account, depending on your requirements. For more information, see [Compare Smart campaigns with other Google Ads campaign types](#).

9. Create the OAuth credentials. For more information, see [Creating a Google Ads account and credentials](#).

What to do next: Before you can use the Google Ads Manager Account in production, your API token must first be approved by Google. To start testing immediately without waiting for the token to be approved, [create a test manager account](#).

Creating a test manager account in Google Ads

Create a test manager account to test new API implementations or account configurations before implementing the changes in your production environment. Test accounts do not require an approved developer token, so you can start working with the API immediately.

Note: Test accounts cannot be used in production, since they do not actually serve ads to customers.

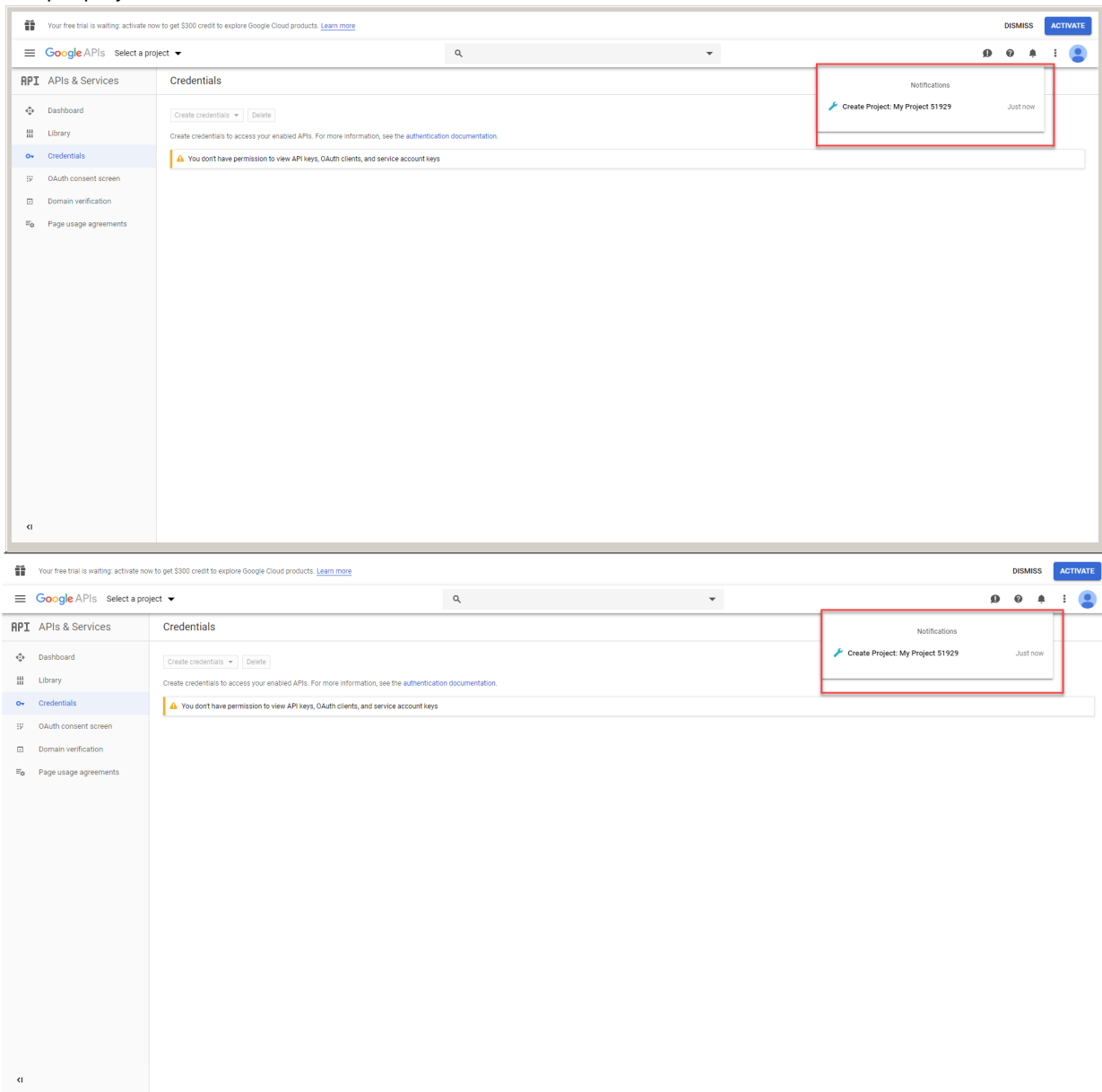
1. Sign in to Google with a Google account that is not already linked to a Google Ads account.
2. Go to [Create a Google Ads Test Account](#).
3. Fill out the registration form and click **Save and continue**.
4. Create the OAuth credentials for use with the test account.
For more information, see [Creating a Google Ads account and credentials](#).

Creating the OAuth credentials in Google Ads

Create the OAuth credentials to obtain the client ID and client secret required to configure the connection to a Google Ads destination in Paid Media Manager. You can create these credentials for both the production and test manager accounts.

1. Log in to Pega Customer Decision Hub.
Note: Make sure that the link you use to log in includes the `prweb` root domain and the Pega Customer Decision Hub instance domain, that is, `https://host.domain:port/prweb`. If you log in using a link without either the root domain or the instance domain, the redirect URL for OAuth will not be valid and authentication will not succeed.
2. Log in to the [Google API Interface console](#).
3. Create a new project by clicking **Create**.
4. Select the project which you created.

Sample project



5. Click **OAuth consent screen** and fill out the required information. Details of some of the critical fields are provided below.
 - Under **User type**, select **External**.
 - On the first screen, under **Authorized domains**, enter the root domain of your Pega Customer Decision Hub instance.
 - On the second screen, click **Add users** and add an email ID that will be used for the OAuth process. This is required while the app has the Testing status. You can move the app to Production status when you start to use it for production systems.
- Note:** Domain verification is an optional step for Paid Media Manager integration.
6. Click **Save**.
7. Click **Credentials Create credentials OAuth client ID**.
8. Under **Application type**, select **Web application**.
9. Under **Authorized redirect URLs**, click **Add** and enter the redirect URL for your instance, that is, `https://host.domain:port/prweb/PRRestService/oauth2/v1/redirect`.
10. Click **Create** and note the client ID and client secret that are displayed.

What to do next: Important: Your Google Ads account must be able to access the following APIs:

- *CrmBasedUserList* - Mandatory. Paid Media Manager uses this API to manage the audiences in the Google Ads network.
- *OfflineDataUploadService* - Optional. This API is required if you want to use Paid Media Manager to upload data about historical offline conversions into Google Ads.

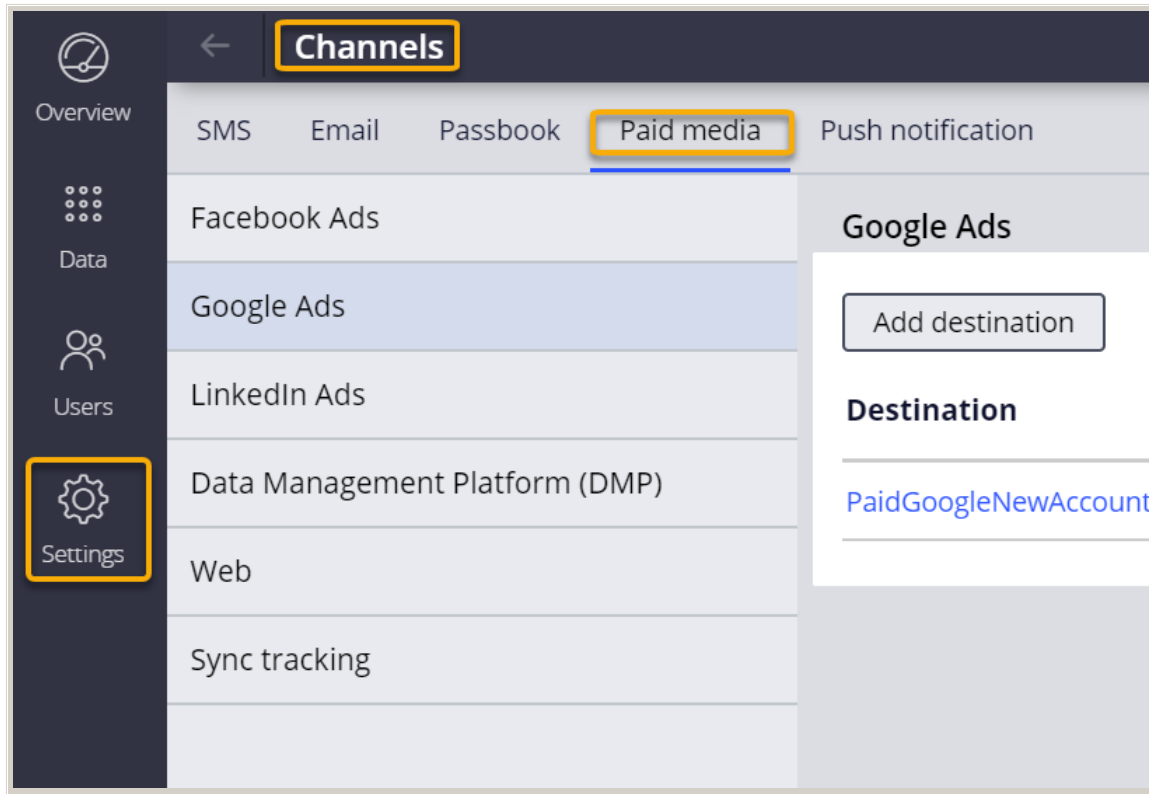
Contact your Google Ads account representative for information about enabling the APIs for your account. To continue setting up the Paid Media Manager connection to the Google Ads destination, see [Configuring Google Ads destinations in Paid Media Manager](#).

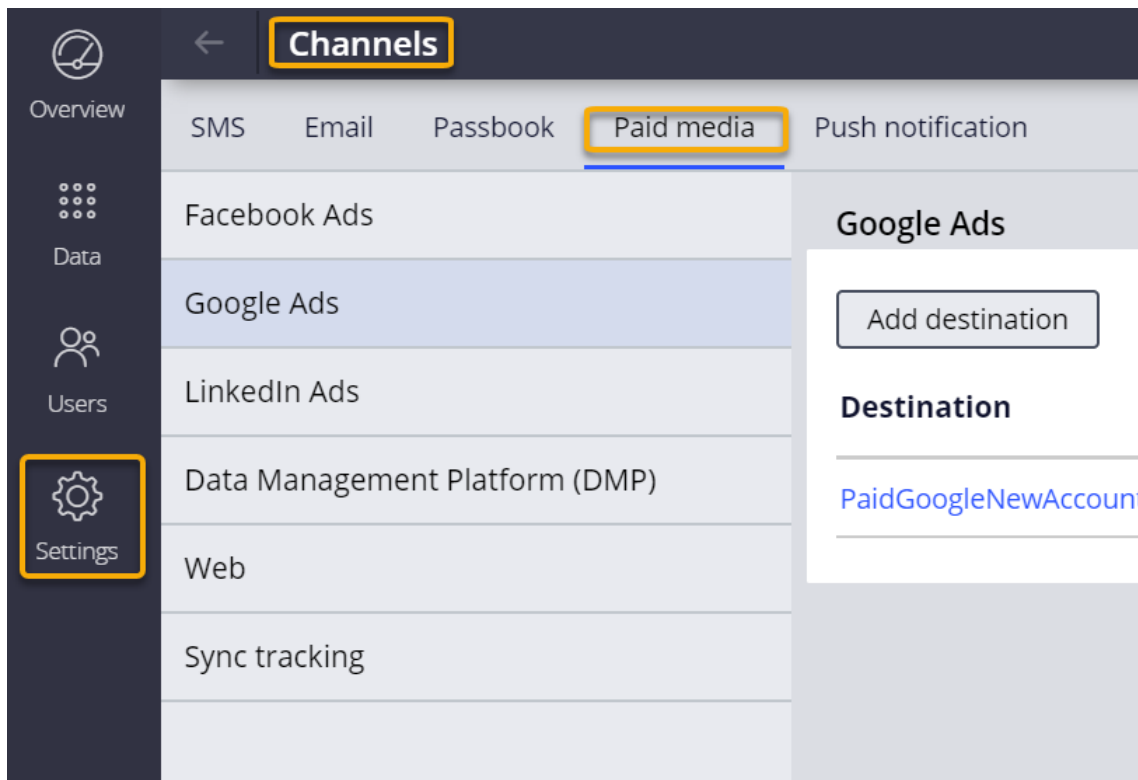
Configuring Google Ads destinations in Paid Media Manager

After you [configure the Google Ads account and credentials](#), use them to configure the connection to a Google Ads destination in Paid Media Manager.

1. Log in to Pega Customer Decision Hub as an operator with access to App Studio.
2. In App Studio, click **Settings Channels Paid Media Google Ads**, as in the following figure:

Accessing the Paid Media landing page





3. Click **Add destination**.
4. Fill out the required fields, such as the destination name, as well as the developer token and OAuth credentials that you created in Google Ads.
For more information about the credentials and developer token, see [Creating a Google Ads account and credentials](#).
5. Click Grant Access and authenticate in the pop-up window. **Result:** This generates the authentication token for OAuth. **Note:** If an error message shows during the authentication process, close the pop-up window and authenticate again. To provide a more detailed error message for users who might encounter the issue, open the OAUTH2 service package in Dev Studio and change the **Service access group** value to PegaNBAM:Agents.
6. In the **Field mapping section**, map destination API properties to Pega Customer Decision Hub properties.
 - a. Map the **Google Ads Email (EMAIL)** parameter to a property.
 - b. **Optional:** Map the **Google Ads Phone (PHONE)** property to a property.
Note: Mapping multiple properties makes it possible to generate higher match rates and reach more individuals.
 - c. To target individuals based on their device ID, configure the additional settings in the **Device configuration** section.
For more information, see [Enabling device ID targeting for Google Ads](#).
- For more information about PII mapping, see [Mapping individuals to paid destinations with CRM attributes](#).
7. In the **Google Ads assets generation** section enable automatic campaign generation for Google Ads.
For more information, see [Streamlining Campaign Generation for Next-Best-Action Paid Audiences](#).
8. Click **Apply**.

What to do next: **Note:** If you get a USER_PERMISSION_DENIED error after completing all the steps, go to your Google Ads account and click **TOOLS & SETTINGS Access and security**. In the **USERS** section, verify that the email ID used for granting access is listed. If not, add the email, and then click **Settings Revoke access Grant access Apply again**.

Enabling device ID targeting for Google Ads

For Google Ads paid destinations, you can obtain individuals' device IDs in an application that you own, and then use that device ID to retarget the individual with Next-Best-Action-based actions relevant to the individual in third-party Android or iOS apps.

For example, when an individual named Troy Murphy uses the banking app owned by his bank, the UPlus Bank, the bank can collect Troy's mobile advertiser ID (that is, AdID on Android and IDFA for Apple), and use that ID to target their ads at Troy in third-party apps. When Troy next opens the Youtube app, he is recognized by his device ID, and may see an ad for UPlus Bank. Paid Media Manager integrates with this mechanism by ensuring that the ad targeted at Troy is his most relevant Next-Best-Action.

To configure device ID targeting for a Google Ads destination:

1. Configure Google Ads to use a mobile device ID.
For more information, see [About Mobile Device IDs for Remarketing](#).
2. Log in to Pega Customer Decision Hub as an operator with access to App Studio.
3. In App Studio, click **Settings Channels Paid Media Google Ads**.
4. Edit or create a destination.

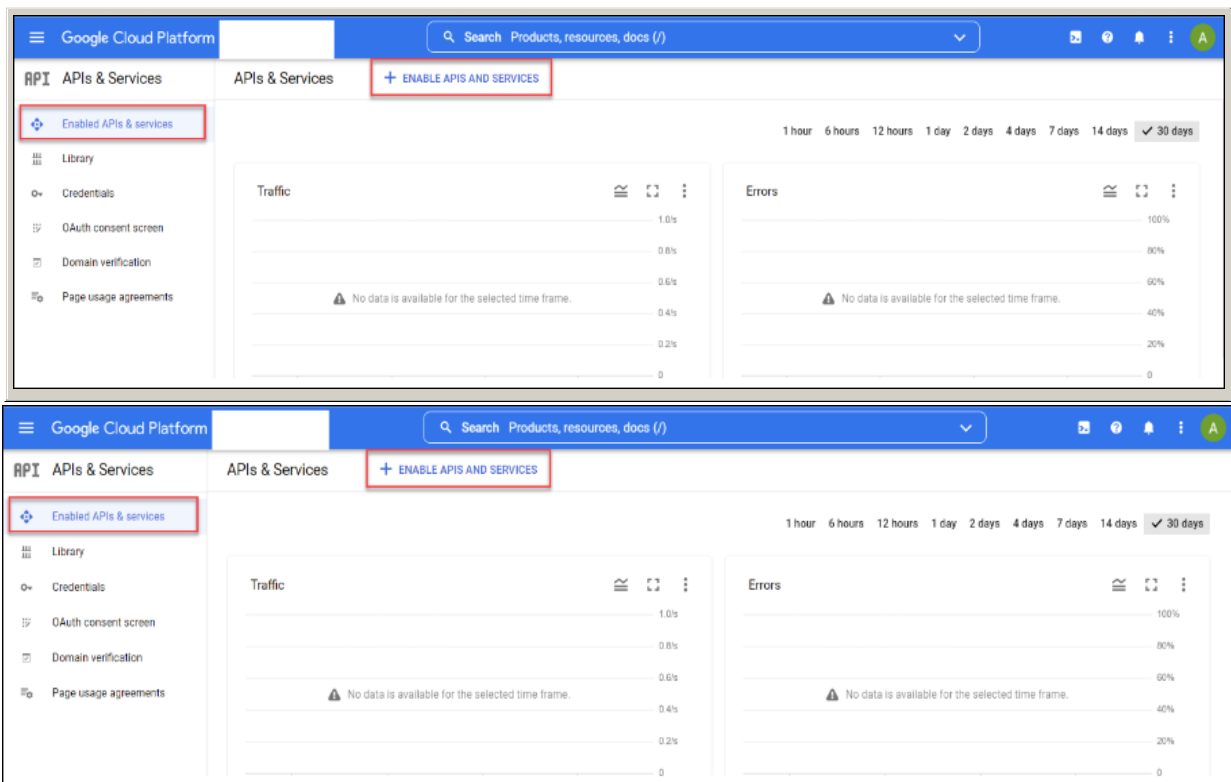
5. In the **Device configuration** section, configure the following settings:
 - **App ID** - Enter the ID of an Android or iOS app that you own. You can specify multiple apps. **Note:** For Android apps, the ID is the application package name from Google Play store. For iOS apps, the ID is a unique number provided by Apple. For more information about app IDs, refer to Android or iOS documentation.
 - **Device** - Enter the Pega Customer Decision Hub property used to identify the type of mobile device that the individual owns. To add more devices, click **Add device**. You can specify multiple devices for each of your apps.
6. Click **Apply**.

Migrating from Google AdWords API to Google Ads API

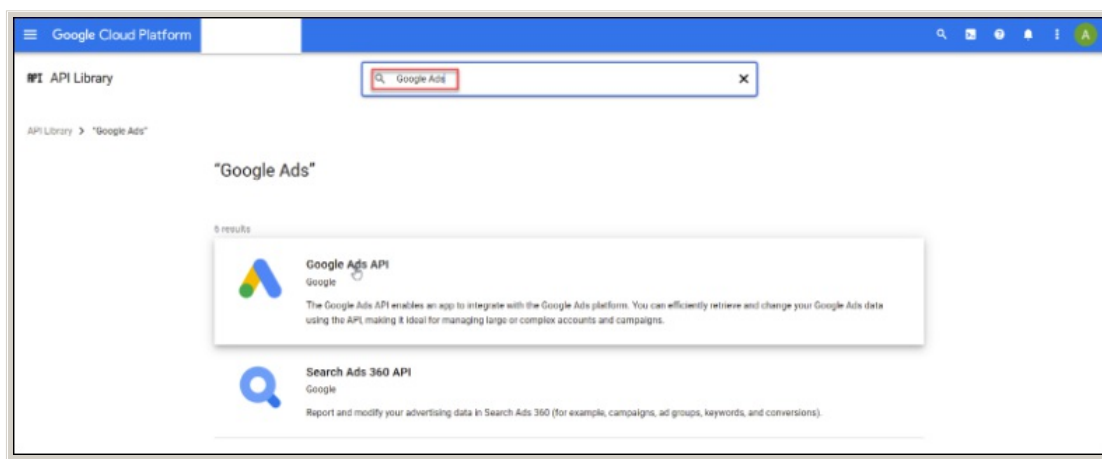
Migrate your Paid Media Manager to the new Google Ads API. Google ceased development on Google AdWords in April 2022, if you still have AdWords in your system, you must update to the new Google Ads API.

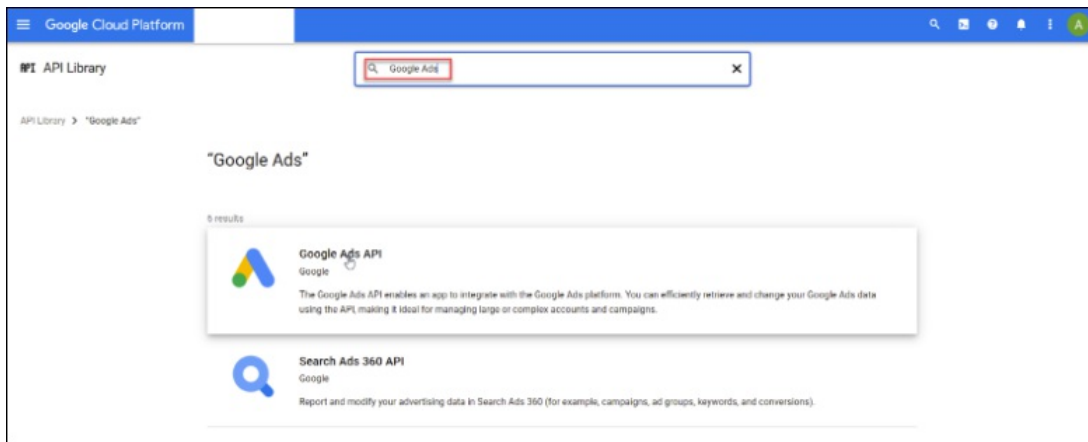
Enabling Google Ads APIs

1. Log in to the [Google API Interface](#) console.
2. Navigate to **Enabled APIs & services** tab and click on **+ Enable APIs and services**.

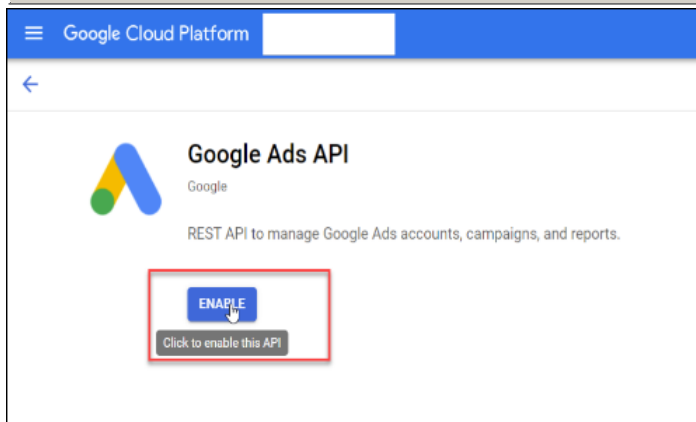
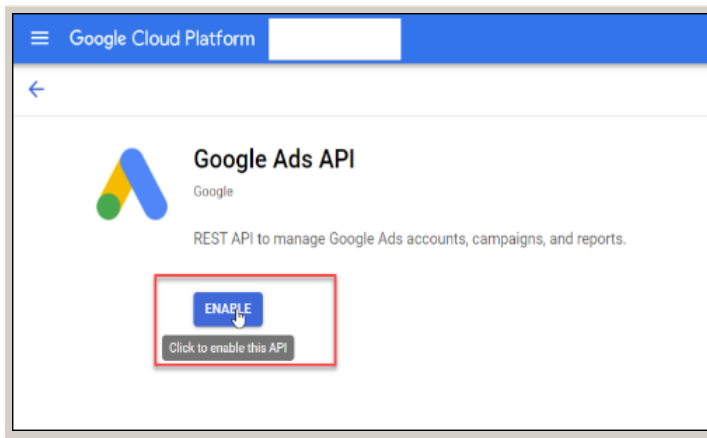


3. In the header of Google Cloud Platform console, in the search field enter: Google Ads API.

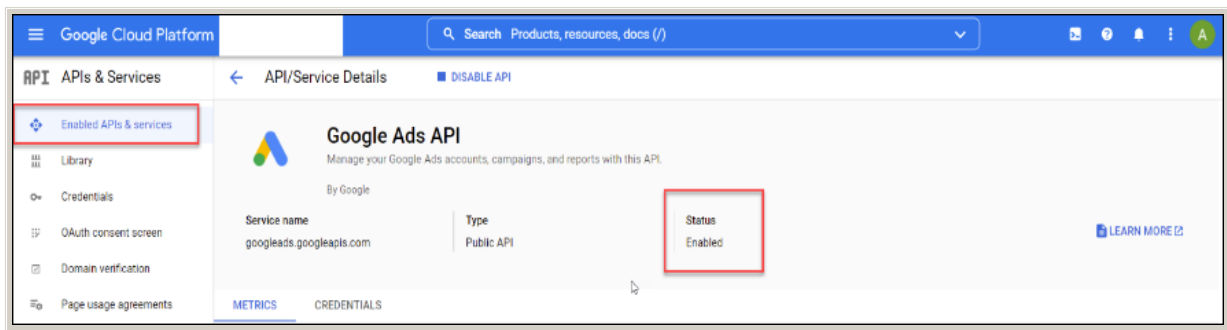


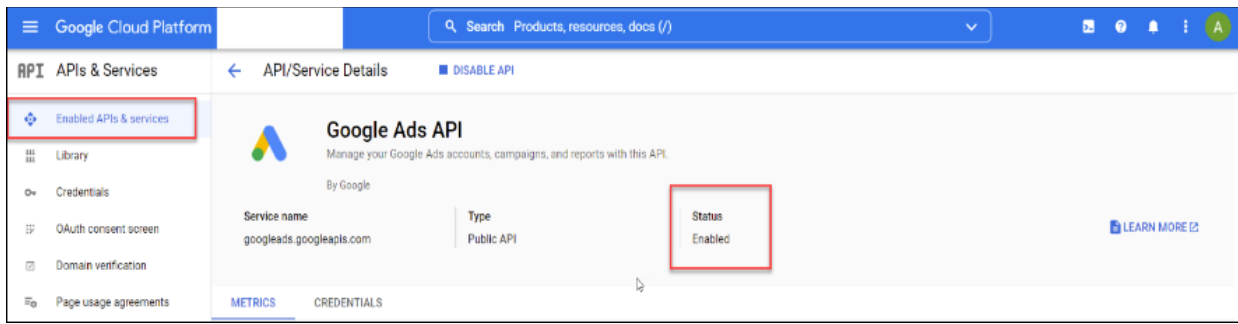


4. Click the **Enable** button, to enable the Google Ads API.



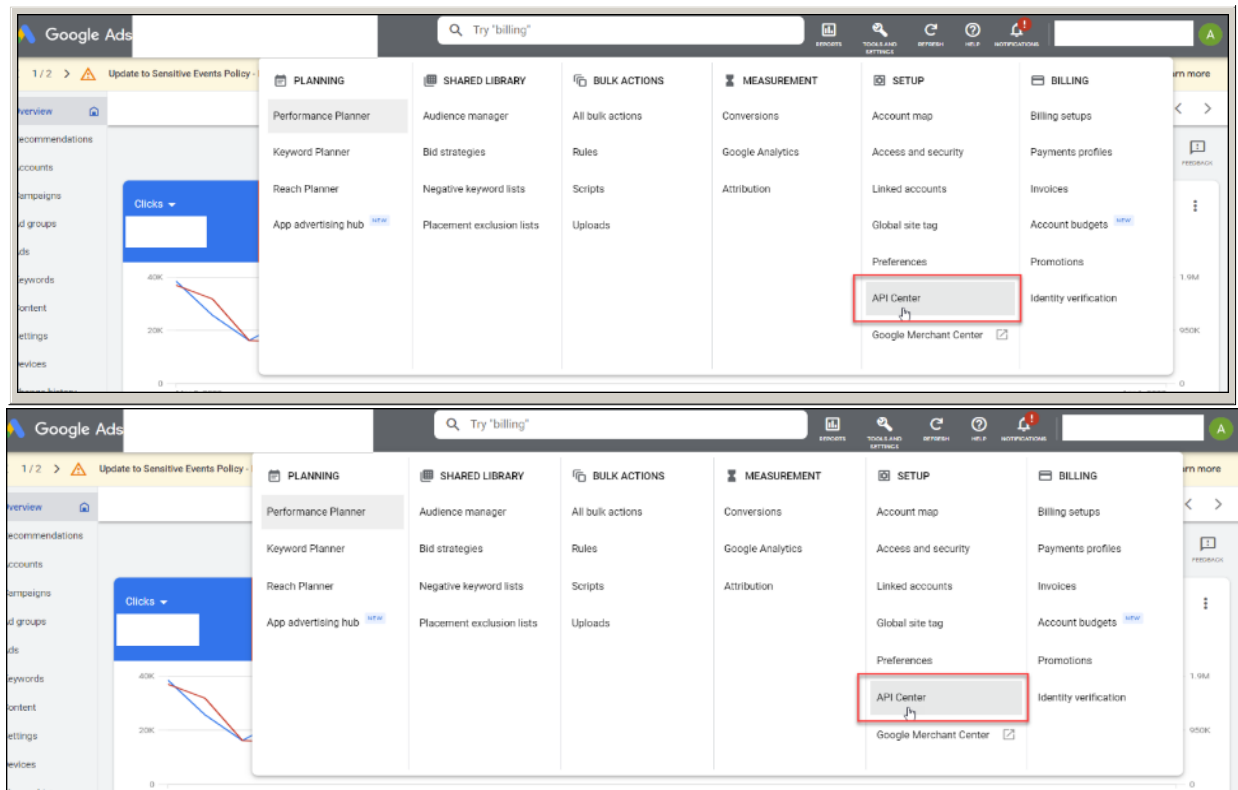
Note: The Google Ads API is now visible in the **Enabled APIs & services** tab and the status is changed to *Enabled*.





Accepting Terms of Service for exiting developer token to use Google Ads API

1. Log in to your [Google Ads Account](#).
2. Click **Tools & Settings Setup API Center**



3. Accept the terms of service for using Google Ads API. **Note:** If you do not accept the terms, a call to Google API will result in MISSING_TOS error with a message: Developer must sign the terms of service. They can be found here: ads.google.com/aw/apicenter.

Enabling the job scheduler

Enable the *GetPaidAsyncJobStatus* job scheduler for integration with Google Ads API. This job scheduler polls the status of Google sync jobs from the Google Ads Network. By default, this scheduler runs once every hour.

You should enable it after you configure any Google paid destinations. For more information, see [Optional: Implementing Paid Media Manager](#).

Dynamic System Settings and new sync tracking status

For [monitoring paid synchronization jobs](#), new *Awaiting response* status has been introduced to track the asynchronous Google Ads API.

Two new [Pega Customer Decision Hub Dynamic System Settings](#) have been added for Google Audience and Google Conversion sync jobs.

Configuring LinkedIn Ads destinations

LinkedIn Ads Destinations enable ads to be targeted on LinkedIn apps and websites. Paid Media Manager integrates directly with LinkedIn Ads and the LinkedIn Matched Audiences API.

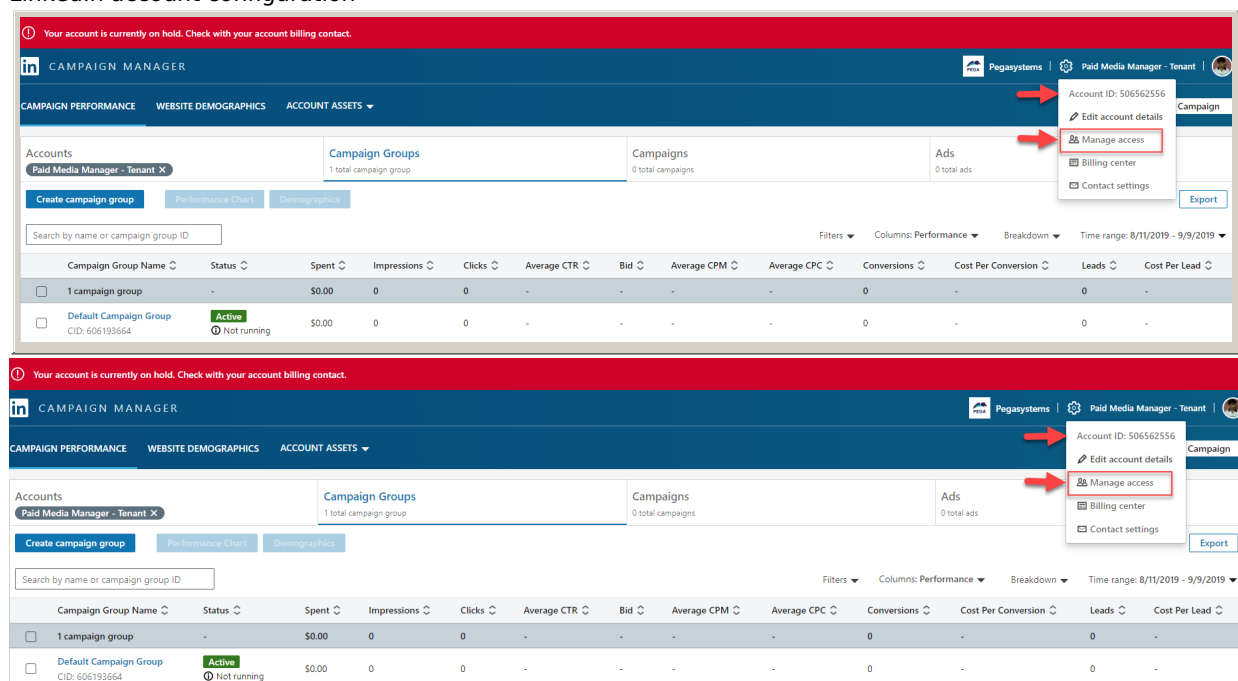
1. Create and configure a LinkedIn account. For more information, see [Creating a LinkedIn account and an authorization profile](#).
 2. Configure the connection to a LinkedIn Ads destination. For more information, see [Configuring LinkedIn Ads destinations in Paid Media Manager](#).
- [Creating a LinkedIn account and an authorization profile](#)
 - [Configuring LinkedIn Ads destinations in Paid Media Manager](#)

Creating a LinkedIn account and an authorization profile

As part of the process of connecting Paid Media Manager to a LinkedIn Ads destination, you must have an account in LinkedIn Ads and use the credentials for that account to configure a destination in Paid Media Manager.

1. Log in to [LinkedIn Campaign Manager](#).
2. Click **Create account**.
3. Enter the account name and currency, and click **Save**.
4. In the account settings, note the account ID.
5. To add users to the account, click **Manage users**, as in the following figure:

LinkedIn account configuration



6. Contact Pega Support to authorize the connector. Include the Redirect URL in the Service Request. This is the URL that LinkedIn uses to access your Pega Marketing application. It has the format `http://<hostname>:<port>/prweb/PRRestService/oauth2/v1/redirect`. You can check the redirect URL for your application in the destination settings in Paid Media Manager.

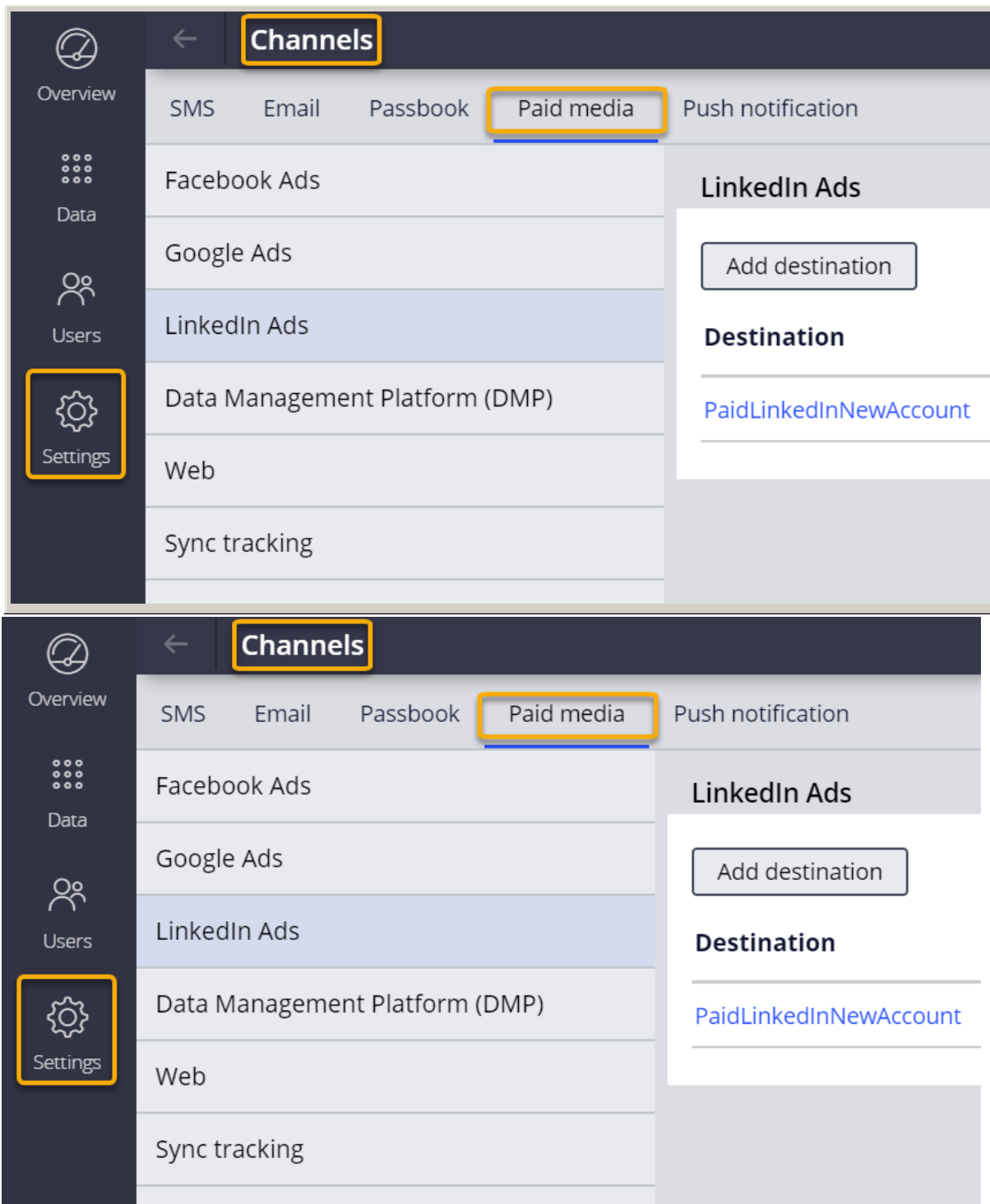
What to do next: To continue setting up the Paid Media Manager connection to the LinkedIn Ads destination, see [Configuring LinkedIn Ads destinations in Paid Media Manager](#).

Configuring LinkedIn Ads destinations in Paid Media Manager

After you create a [LinkedIn account and authorization profile](#), configure the connection to a LinkedIn Ads destination in Paid Media Manager.

1. Log in to Pega Customer Decision Hub as an operator with access to App Studio.
2. In App Studio, click **Settings Channels Paid Media LinkedIn Ads**, as in the following figure:

Accessing the Paid Media landing page



3. Click **Add destination**.
4. Enter the credentials that you created in LinkedIn Ads.
For more information, see [Creating a LinkedIn account and an authorization profile](#).
5. Take note of the URL in the **Redirect URL** section. LinkedIn uses this URL to authorize connections.
For more information, see [Creating a LinkedIn account and an authorization profile](#).
6. Fill out the following fields:
 - **Destination name**
 - **Account ID**
 - **Source platform** - enter PEGA
 - **Client ID** - Enter the Client ID. [Pega Support provides the required Client ID after authorizing the Paid Media connector for LinkedIn](#).
7. **Optional:** Configure the additional settings in the **LinkedIn Ads assets generation** section.
8. Click **Apply**.

Configuring web-based and DMP connections to paid destinations

In addition to modern, CRM-based platforms, Paid Media Manager also supports pixel-based APIs provided by advertising technology platforms, including demand-side platforms (DSPs), data management platforms (DMPs), and advertising networks. Instead of CRM-based APIs, these platforms generally rely on third-party cookies to build anonymous profiles and target individuals with ads. Paid Media Manager integrates with these mechanisms to deliver people-based Paid Audiences and Next-Best-Action decisions.

Typically pixel-based mechanisms target individuals with advertisements based on only the current content of the website which the individual is viewing. For example, if Troy goes to uplusbank.com and views information on a Platinum credit card, the pixel would then activate and target Troy with ads for the credit card. Paid Media Manager instead bases the ad targeting decision on Next-Best-Action. This takes into account the history of Troy's interactions and the learnings derived from them in order to show him an ad for a product which he has a high propensity to buy.

Unlike with CRM-based APIs where audience can be updated via S2S API calls at any time, pixel-based audiences can only be updated when an individual is in a live web session and on an advertiser owned website. Many browsers also prevent access by third party cookies, which may further limit the ability to update audiences through this mechanism.

- [Understanding the PaidAudiencesContainer REST service](#)
- [Enabling Next-Best-Action targeting on DMP-based destinations](#)

Understanding the PaidAudiencesContainer REST service

Paid Media Manager allows you to include the power of Next-Best-Action decisioning into the pixel calls. You can extend the pixel call with a call to the *PaidAudiencesContainer* web service to determine ad targeting based on the enterprise decisioning and intelligence embedded in Next-Best-Action.

PaidAudiencesContainer is a REST service that allows callers to retrieve the list of paid media audiences for a customer in one or more paid media destinations.

The service supports the GET HTTP method.

The URL pattern for this service is:

`http://<host>:<port>/prweb/PRRestService/PegaMKTContainer/V1/PaidAudiencesContainer`

For the GET request, the service expects a JSON Object with the following attributes:

CustomerID

Used to identify individuals based on information captured during their interactions with the destination. For individuals yet to be identified, a unique id is stored in a browser cookie. Once an individual has been identified as a customer, their encrypted customer id is stored. For more information, see [Identity management for the Container REST service](#).

DestinationKey

Destination key of the paid media destination. For multiple destinations, comma-separated lists of paid media destination keys are allowed. To check the destination key for a specific destination, in App Studio, click **Settings Channels Paid Media Web**. Take note of the value in the **Destination key** column.

Refresh

If set to `true`, this parameter triggers the Next-Best-Action strategy at the top level for the customer. The results of the strategy execution are sent back to the caller. This process also takes into account any paid strategy that is configured at the top-level Paid Media configuration in the **Channels** tab of Next-Best-Action Designer. The response will contain all the base audiences returned by the Next-Best-Action strategy and paid strategy (if configured).

If set to `false`, or not included in the request, the response will contain all the audiences in the paid media master records which include base audiences, negative audiences, lookalike audiences, device-based audiences, and all customer next-best-action audiences.

Note: The paid web service is designed for low latency and does not execute Next-Best-Action for the site visitor which would update the individual's NBA Paid Audiences. However, the real-time container can be called separately to do so.

Sample invocation request and response

Request:{"CustomerID":"101","DestinationKey":"DKa93e1234"}

Response:{"Status":"OK","CustomerID":"101","AudienceData":{"DKa93e1234":{"PaidAudiences":[{"AudienceID":"6e43cf87"}, {"AudienceID":"f7fb59b1"}, {"AudienceID":"d9d172bb"}], "pxSubscript":"DKa93e1234"}}

The *AudienceID* in the responses is generated by Paid Media Manager and associated with an audience generated in the paid destination. You can see the audience ID for each audience in the **AUDIENCE KEY** field on the Paid Media Audiences landing page.

You can use the audience ID to fire Facebook Ads pixels or Google Ads events for each audience. Because the audience name is the same as the name of a corresponding action in Pega Customer Decision Hub, you may want to avoid exposing the name at the client-side. Because of that, you can use the audience ID instead of the audience name in a browser cookie to keep track of pixels and events which have already been fired for a particular individual. In the network, while creating audiences out of the pixel, get the relevant audience name for the audience ID from the Paid Media landing page and use the same.

Enabling Next-Best-Action targeting on DMP-based destinations

Configure one or more paid destinations for Adobe Audience Manager and other Data Management Platforms (DMPs) under Paid Media settings in App Studio.

DMPs provide a mechanism to update cookie-based audiences outside of a client web session. After a DMP establishes a cookie for an individual who visits an advertiser's website, subsequent updates can be made through S2S updates to the DMP. These updates can then be delivered to a separate ad platform responsible for selecting and bidding ad inventory, such as a Demand Side Platform (DSP). Each of these media buying platforms should have a separate destination to account for differences in paid action priority for an individual across different destinations.

Paid Media Manager provides out-of-the-box support for the Adobe Audience Manager DMP. Paid Media Manager generates the appropriate traits and audiences within Adobe Audience Manager through API. After a scheduled run completes, a file is also generated with the audience updates produced by Next-Best-Action Designer. This file is delivered via S3 bucket or FTP to Adobe. Adobe Audience Manager can take up to 48 hours to propagate these updates to downstream DSPs and other third-party cookie-based platforms. In addition to Adobe Audience Manager, you can also configure paid destinations for a generic DMP.

For DMPs, updates are only made in response to scheduled runs and segments.

1. Log in to App Studio.
2. In the navigation pane of App Studio, click **Settings Channels Paid Media Data Management Platform (DMP)**.
3. Click **Add destination**.
4. Fill out the destination name and description.
5. Select the destination type.
 - **Generic DMP**
 - **Adobe Audience Manager**
6. For Adobe Audience Manager, fill out the following required fields:
 - **Destination name**
 - **DP ID**
 - **Data source ID**
 - **API user name**
 - **API password**
 - **API client ID**
 - **API secret key**
7. In the **Sync Configuration** section, map the Adobe Audience Manager **Sync ID** property to a Pega Customer Decision Hub property that holds the value of the universally unique identifier (UUID) that identifies the individual. The UUID property is implemented as part of the Adobe ID synchronization and works as a synchronization key, allowing the integration to match known individuals to their accounts and target them with the right ads on the right platform.
8. **Optional:** Configure the additional settings in the **Sync Configuration** section.
 - To send all Paid audience information to DMP destinations every time that the synchronization runs, enable **Full synchronization**.
Enable this setting if you want to send all Paid audience information to DMP destinations every time that the synchronization runs. Enabling full synchronization results in larger files being uploaded into the DMP. This can slow the synchronization. This setting is not enabled by default. When full synchronization is not enabled, the synchronization only sends information which has been updated in Pega Customer Decision Hub since the last synchronization date.
9. In the **Delivery Configuration** section, configure the location where Paid Media Manager sends files with synchronization data to be uploaded into the DMP.
This location may be an FTP location or an Amazon S3 bucket for the files.
10. Click **Apply**.

Generating Next-Best-Action web audiences with Paid Media Manager

Use Paid Media Manager to target individuals on paid channels based on your enterprise data and decisions.

With the traditional approach to web marketing, if an individual views information on a product (for example, a Samsung Galaxy mobile handset), they will likely be targeted with advertisements for that product for days, weeks, or even months after. The company selling the product will leverage pixel-based APIs from various ad platforms to retarget the individual based solely on this one interaction.

With Paid Media Manager, you can improve the targeting by leveraging the full fidelity of enterprise data and intelligence to determine which ads to target the individual with, and at what priority. Instead of relying just on the last web interaction, Paid Media Manager communicates the individual's next best action, considering that interaction as well as data about previous interactions to determine how to target individuals with ads.

The Paid Web Service can be used with any ad platform pixels, including those from Twitter Ads, Amazon Advertising, and The Trade Desk.

Considerations for using pixel-based APIs

When implementing pixel-based APIs, keep in mind the following considerations:

- Unlike people-based APIs that rely on hashed Personally Identifiable Information (PII), such as an email address or phone number, pixel-based APIs rely on a web cookie. As a result, this targeting mechanism requires an individual to be in a live web session to call a pixel. Because the ability to call cookies from a third-party server (a.k.a third party cookies) is increasingly restricted in modern web browsers, the third-party server may sometimes be unable to identify the individual.
- Pixel-based audiences cannot be updated. This means that once an individual is added to a pixel-based audience, they cannot be removed from it and may still see the same ad even after they accept or reject the action, until the individual's audience membership expires. Make sure to set an appropriate expiration these web audiences based on Next-Best-Action.
- As a best practice, use people-based APIs in conjunction with pixel-based APIs to maximize the opportunities to target individuals with ads based on their Next-Best-Action.

Below are specific instructions on how you can implement the Paid Web Service to generate Next-Best-Action audiences on Facebook Ads and Google Ads:

Sample Javascript to send Next-Best-Actions to the advertising pixel

Use Paid Media Manager to target individuals on paid channels based on your enterprise data and decisions.

As part of implementing pixel-based APIs on your website, you must configure an advertising pixel which will trigger when an individual does specific actions on your website. To help you configure the pixel for Next-Best-Actions, you can refer to the following sample Javascript: [Sample Paid Web Service Pixel Call for Facebook Ads and Google Ads](#). The sample contains examples of Facebook- and Google-specific code that you can reference in your own implementation.

The sample code uses the Pega Marketing and Pega Customer Decision Hub identity management feature. For more information, see [Identity management for the Container REST service](#).

Using a Facebook Pixel to create action-based audiences in Facebook Ads

This tutorial describes how to customize a Facebook Pixel to include the information required by the Paid Web Service, and how to convert a Facebook event into a people-based audience.

For more information about the Facebook Pixel, see [Creating and installing a Facebook Pixel](#).

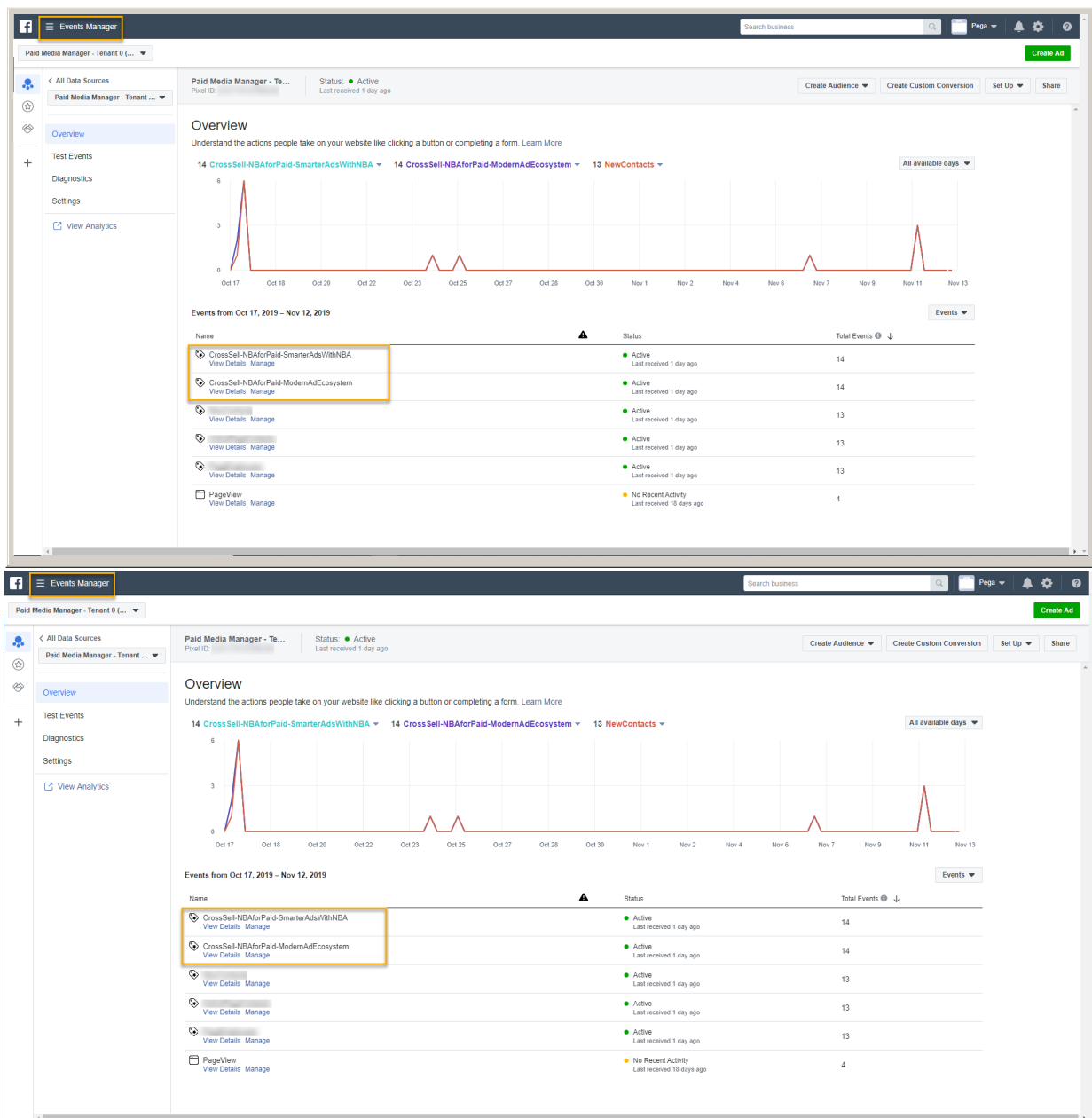
1. Create a Facebook Pixel in Facebook Ads.
To help you configure the pixel, see [Sample Javascript to send Next-Best-Actions to the advertising pixel](#).

Make sure to include the following line in the `<script>` section of the pixel code: `fbq('trackCustom', 'AudienceName');`

The value of the *AudienceName* parameter is provided by the *PaidAudiencesContainer* REST service. This service returns a list of all paid audiences to which an individual belongs, that is, the latest actions selected for the individual by the Next-Best-Action strategy. For more information about the *PaidAudiencesContainer* REST service and its parameters, see [Understanding the PaidAudiencesContainer REST service](#).

2. Install the Facebook pixel on your website.
3. To verify that the pixel is correctly configured, generate some website traffic, and then log in to Facebook Events Manager, as in the following figure:

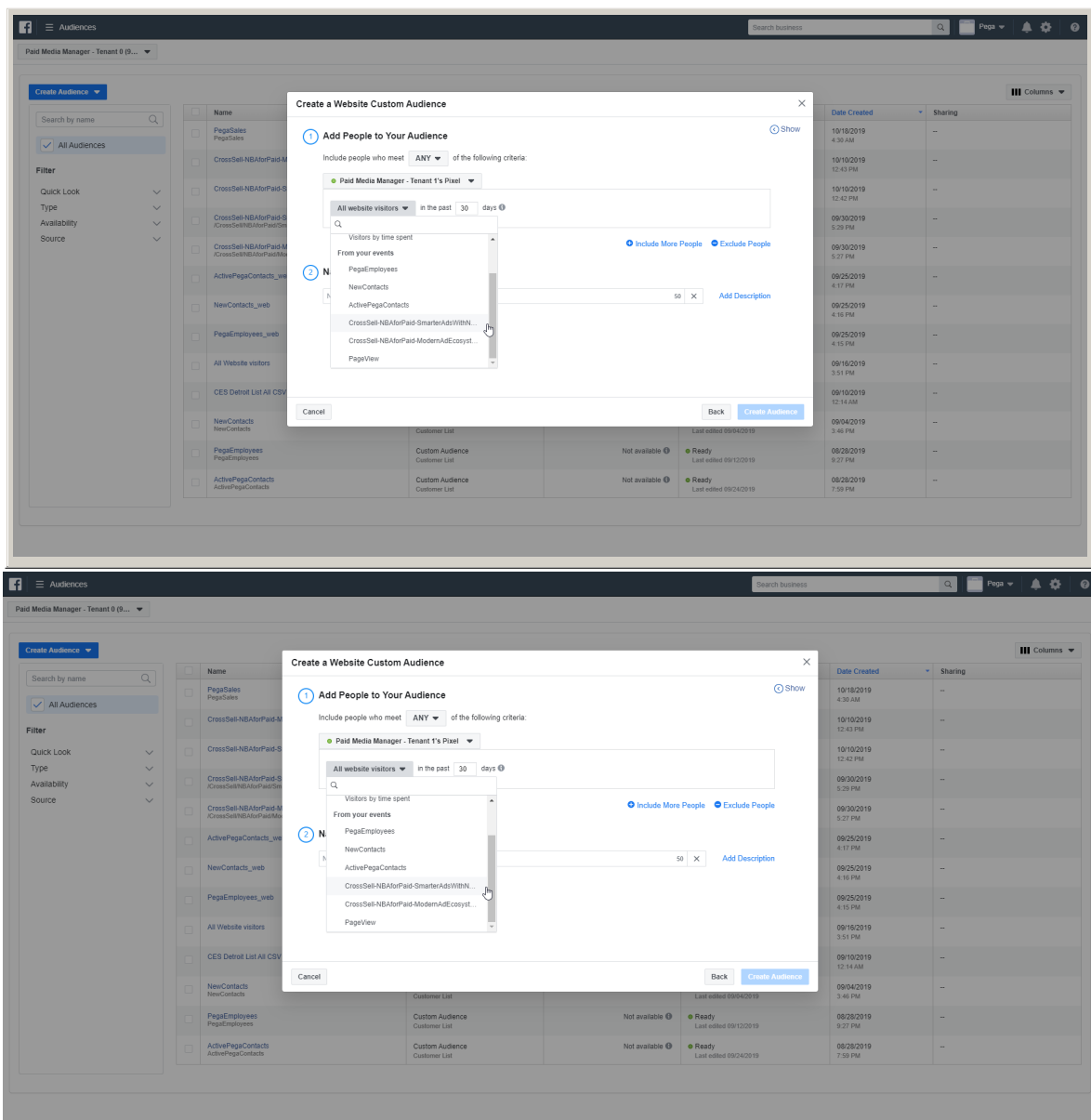
Sample events created with data provided by Next-Best-Action



There might be a delay of several minutes before the events are created. For more information, see [Navigate Events Manager for Facebook Pixel](#).

4. To use the event data in your targeting, convert the events to Facebook audiences:
 - a. Log in to [Facebook Business Manager](#).
 - b. Click **Audiences Create Audience Custom Audience**.
 - c. In the list of sources, select **Website traffic**.
 - d. In the **Add people to your audience** section, select your custom event, as in the following figure:

Creating an audience from event data provided by Next-Best-Action



e. Save the audience and use it in your Facebook campaigns.

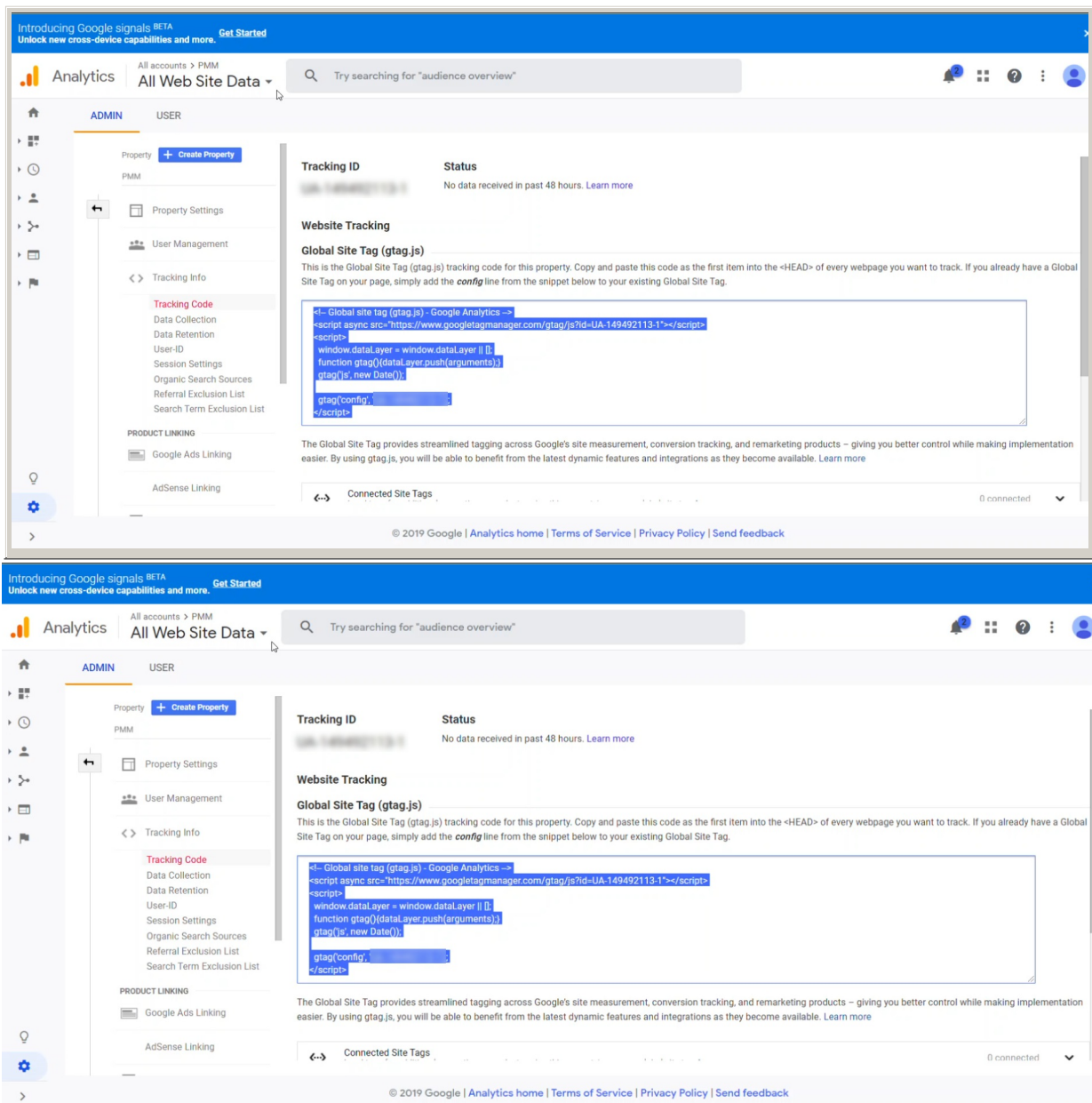
Result: The Facebook Pixel is now installed on your website and ready to gather information about visitor activity on your website, enriched with data about Next-Best-Actions selected for them by Pega Customer Decision Hub. If you want to implement a Google Ads pixel, see [Generating Next-Best-Action web audiences with Paid Media Manager](#).

Using Event Tracking to create action-based audiences in Google Ads

This tutorial describes how to customize a Google Analytics Event to include the information required by the Paid Web Service, and how to convert an Event into a people-based audience.

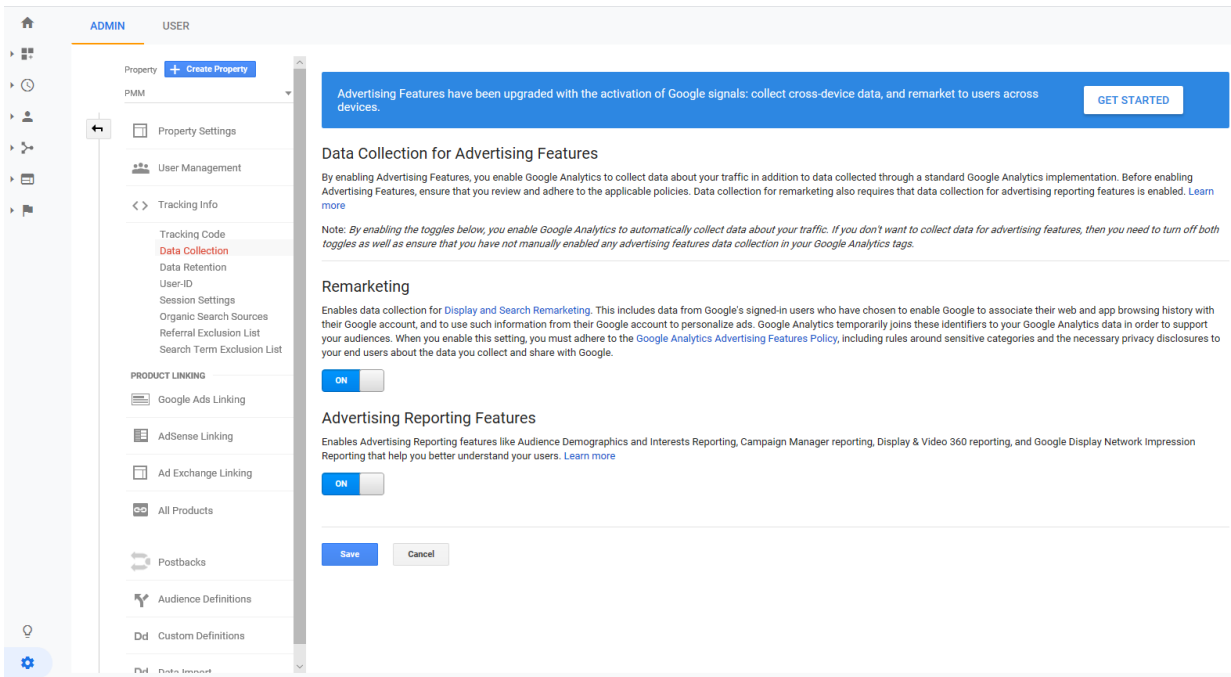
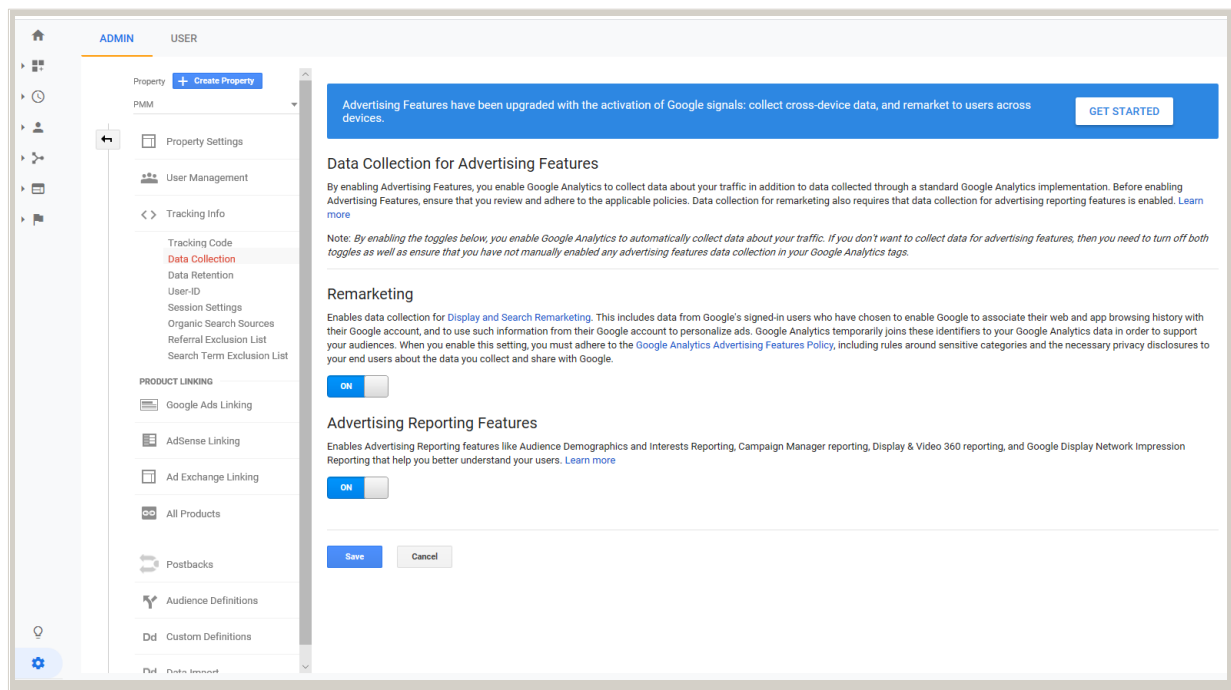
1. Log in to [Google Analytics](#).
If required, sign up for a new Google Analytics account.
2. In the **Admin** panel, click **Tracking Info** **Tracking Code** > and take note of the ID in the **Tracking ID** field.
3. In the **Global Site Tag** field, copy the **Global Site Tag** tracking code.

Global Side Tag and Tracking ID



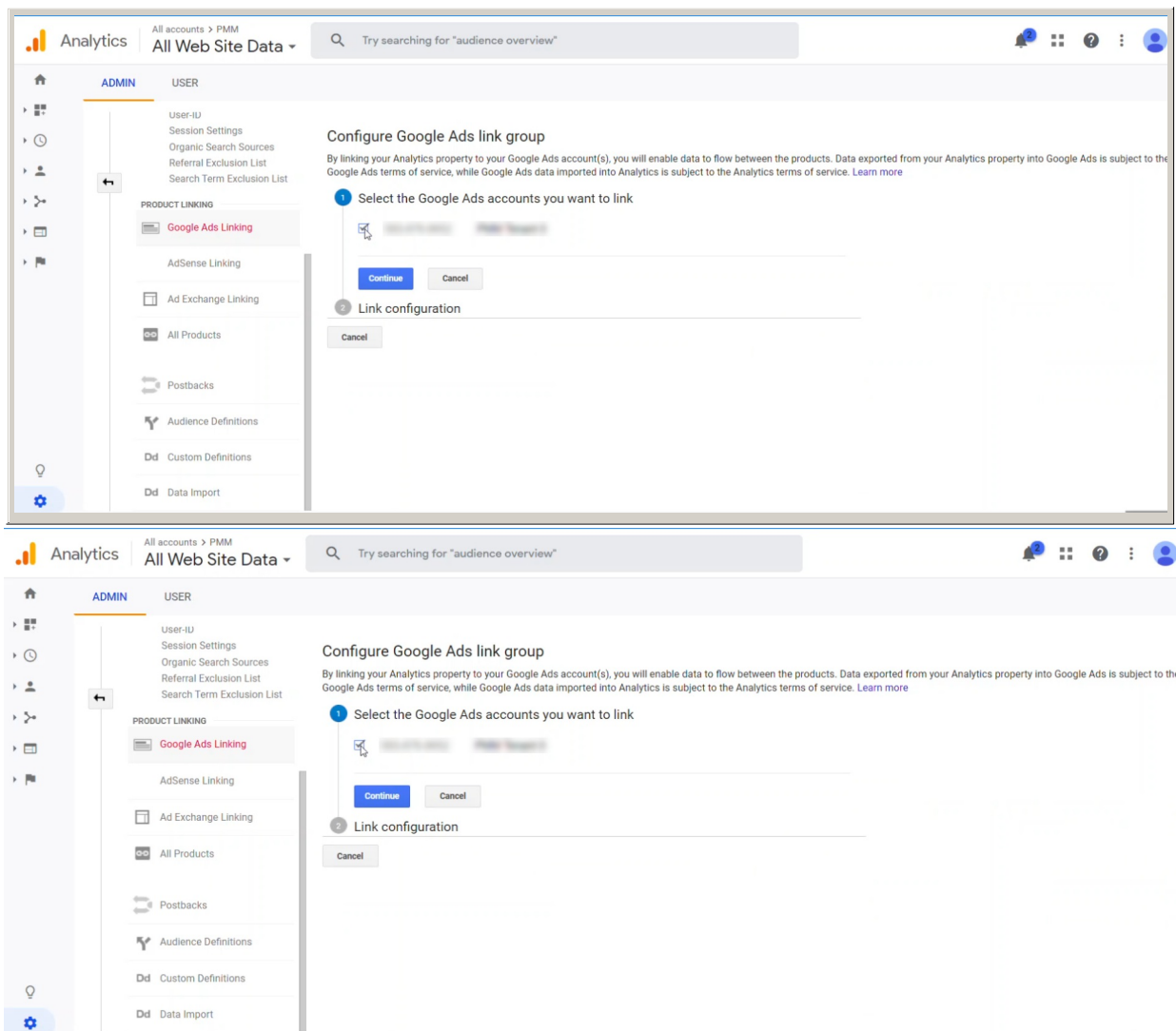
4. Work with your web team to add the Global Site Tag tracking code to the <HEAD> element of every website where you want to track visitor activity.
After implementing this tag, you can monitor site visitors on the Home page of Google Analytics.
5. In Google Analytics, in the **Admin** panel, click **Tracking Info Data Collection**.
6. Set the **Remarketing** and **Advertising Reporting features** toggles to **ON**.

Data Collection page in Google Analytics



7. In the **Admin** panel, click **Product Linking Google Ads Linking** .
8. Select your Google Ads account or create one with the same email as the one that you used for the Google Analytics account.

Selecting the account



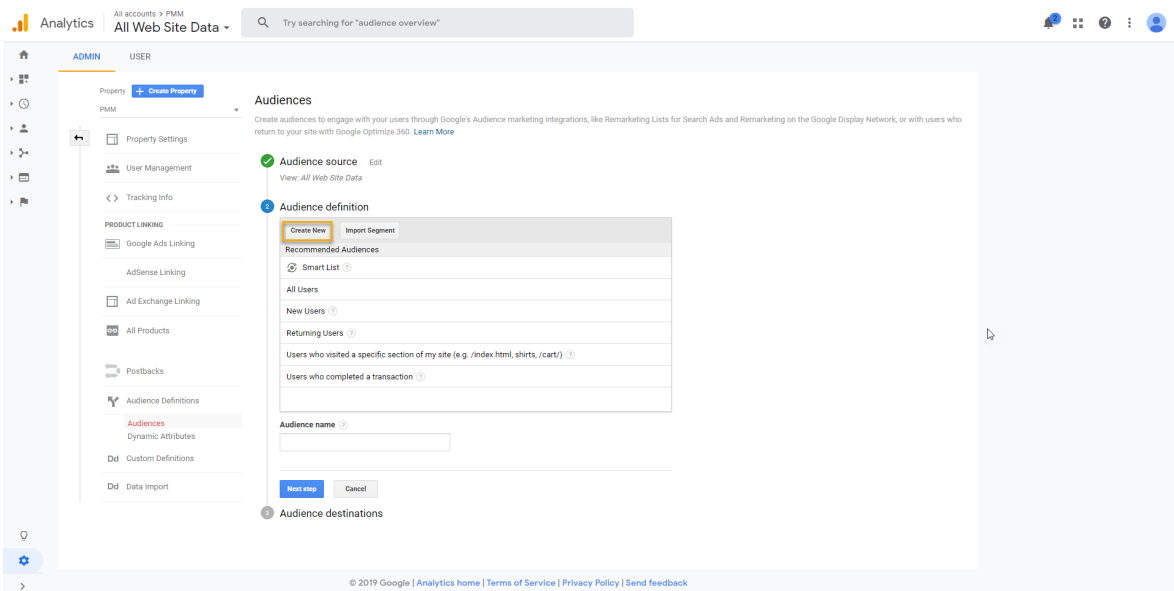
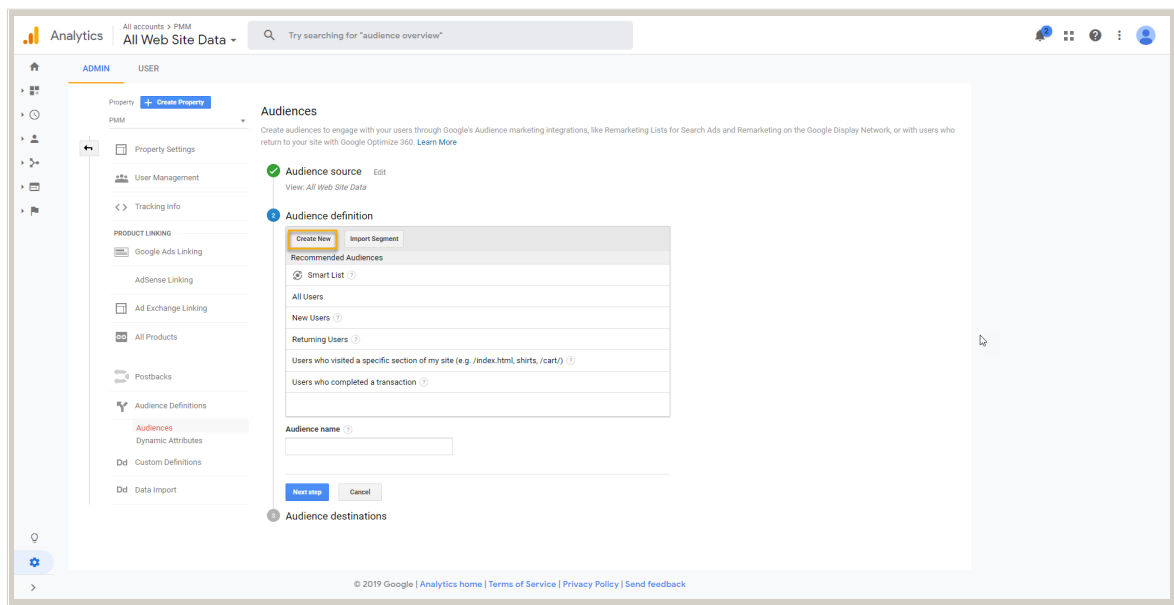
9. Click **Continue**.
10. Set the **All Website Data** toggle to **ON**.
11. Click **Link account**.
12. Work with your web team to add the pixel to your website.
To help you configure the pixel, see [Sample Javascript to send Next-Best-Actions to the advertising pixel](#).

Make sure to include the following line in the `<script>` section of the pixel code: `<script>gtag('js', new Date());gtag('config', '<your global site tag tracking code>');// In the Admin panel, click Tracking Info > Tracking Code and take note of the ID in the Tracking ID field.gtag('event', '<your event action>', { // event action could be Page View, click, download etc. 'send_to': '<your global site tag tracking code>','event_category': '<event category>','event_label': '<event label>'});</script>`

The value of the `<Event Label>` parameter is provided by the `PaidAudiencesContainer` REST service. This service returns a list of all paid audiences to which an individual belongs, that is, the latest actions selected for the individual by the Next-Best-Action strategy. For more information about the `PaidAudiencesContainer` REST service and its parameters, see [Understanding the PaidAudiencesContainer REST service](#).

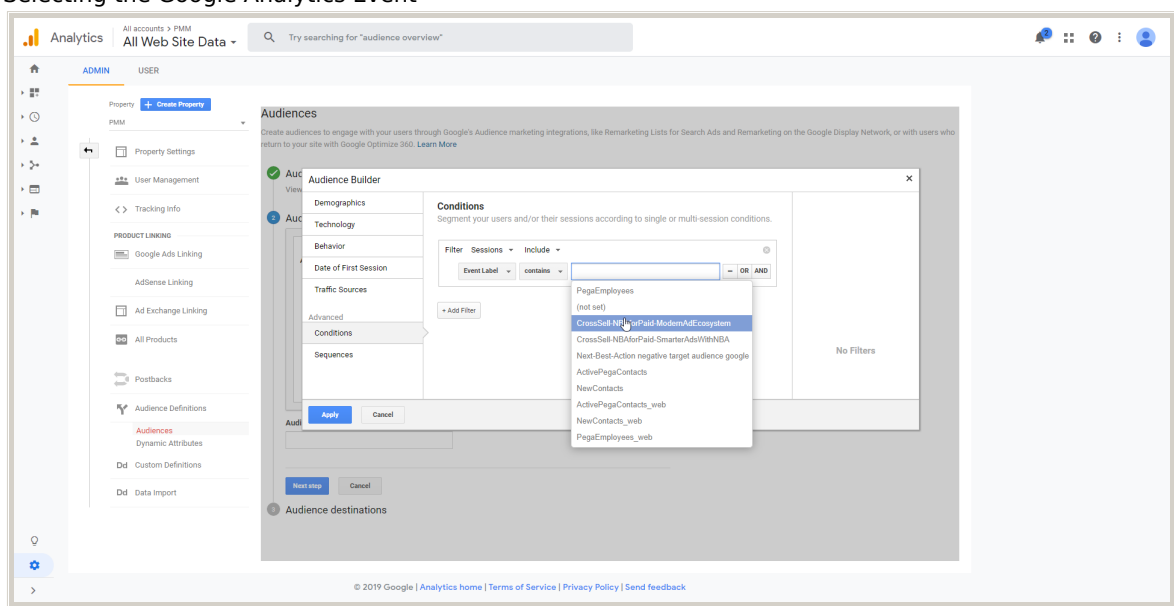
13. To verify that the pixel is correctly configured, generate some website traffic, and then log in to Google Analytics and click **RealtimeEvents** .
Note: There might be a delay of several minutes before the events are created.
14. To use the Event data in your targeting, convert the events to Google Ads audiences:
 - a. In the **Admin** panel of Google Analytics, click **Audience Definitions Audiences** .
 - b. Click **New Audience Create New** .

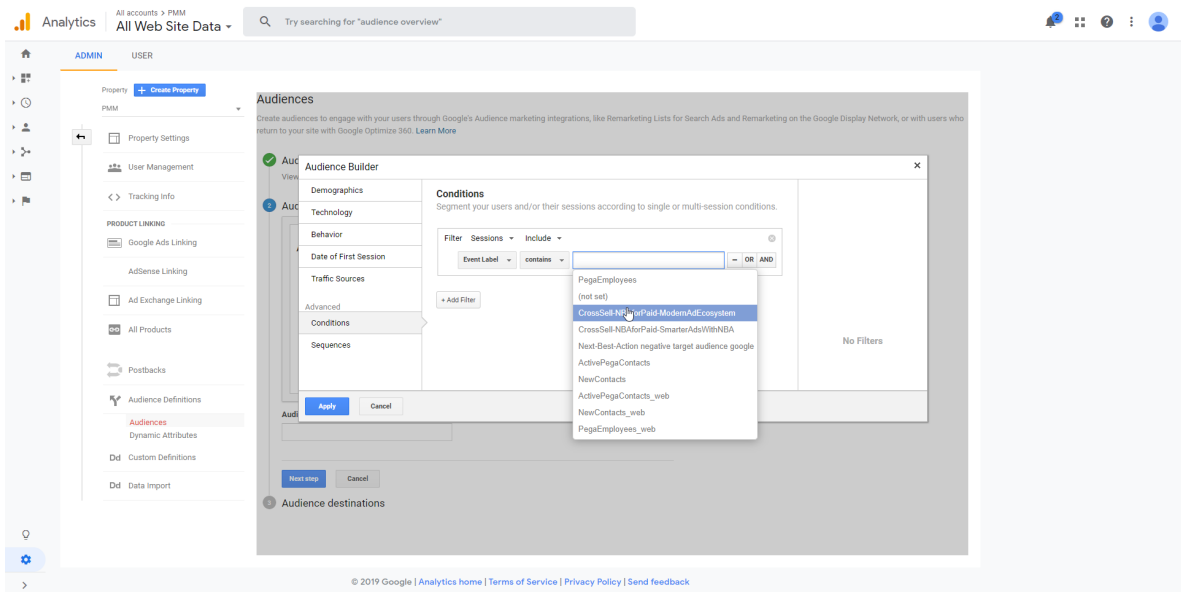
Create a new audience in Google Analytics



- c. Click **Conditions** and select your custom Google Analytics Event by its Event Label, that is, the name of the paid audience to which an individual belongs in Paid Media Manager.

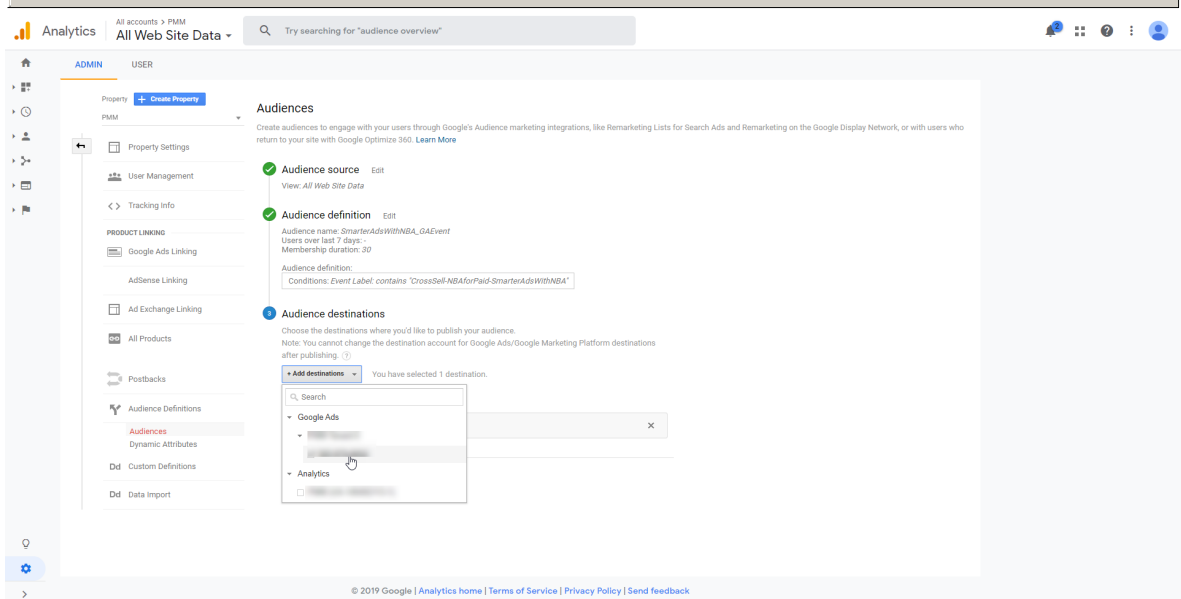
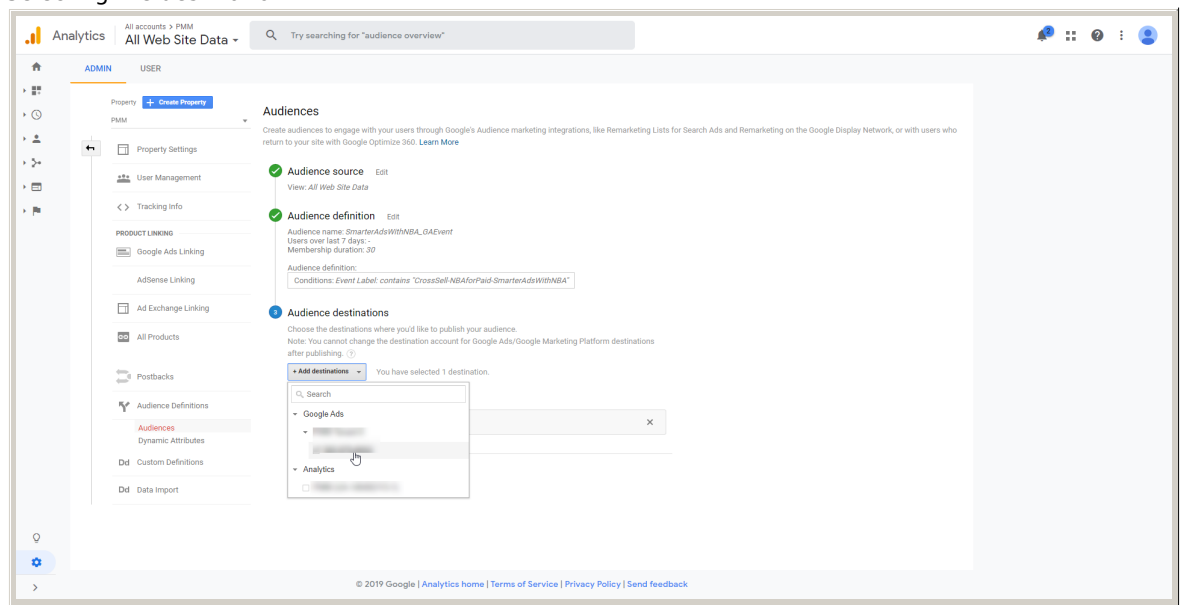
Selecting the Google Analytics Event





d. Add any other required conditions and select the Google Ads account where the audience should be created.

Selecting the destination



e. Click **Publish**.

Result: You have implemented pixel-based APIs for two paid destinations. You can now leverage the full fidelity of enterprise data and intelligence to communicate the individual's Next-Best-Action to paid platforms, considering not only their last interaction on a website, but also data about known individuals' interactions with your ads. In this way, you can maximize your opportunities to target individuals with ads based on their Next-Best-Action.

Enabling AI-driven advertising with paid click capture on Facebook Ads, Google Ads, and other platforms

Pega Customer Decision Hub campaigns that target paid audiences use Next-Best-Action to select the most relevant action for each individual. Paid Media Manager communicates that decision to ad platforms, and targets the individual with relevant ads. Regardless of whether the targeted individual responds positively (by clicking the ad) or negatively (by ignoring it), their behavior should be communicated back to Pega Customer Decision Hub, so that the AI models that govern targeting can learn from the interaction. To do this, you can configure both the paid destination and Paid Media Manager to capture paid clicks.

Sample use case

A marketing analyst configures an action promoting the Open Platinum 50K Bonus Credit Card to known individuals, including customers and prospects. Based on the next-best-action decision, Pega Customer Decision Hub creates a paid audience for the action, for example, a Facebook Custom Audience. The willingness to pay for an ad for a given action to be shown to an individual is designated by the audience and aligned with the priority for that action for that individual. The priority decision can be based on things such as eligibility, propensity, or predicted value for the action per individual. For more information about the mechanism behind paid audience updates, see [Extending next-best-action to digital advertising platforms with Paid Media Manager](#).

An individual named Troy Murphy is eligible, has a high propensity, and high predicted value for the Platinum Credit Card, so he is selected for a 50K bonus for that credit card. Troy's response to an ad for this action can help inform AI models.

Delayed learning

Individuals' responses to actions can determine the propensity for the action across channels and also the effectiveness of the ad to drive the individual to complete the action. However, the response needs to be tied to the specific decision that chose this action for the individual. This done by using an Interaction ID.

For that, Paid Media Manager leverages delayed learning. Delayed learning is a method of capturing response data for previous decisions, for example, the decision that results in Next-Best-Action selecting the Open Platinum 50K Bonus Credit Card for Troy. Delayed learning stores the latest Interaction ID for each action for each individual. This enables Paid Media Manager to retrieve the Interaction ID and close the loop when there is a response to an ad, for example, a click. If the individual does not respond within a week, the model records a negative response. This can be overridden with the *paidmedia/ResponseWaitingTime* dynamic system setting. For more information, see [Delayed learning of adaptive models](#).

Obtaining the Interaction ID

To obtain the Interaction ID, Paid Media Manager must have the customer ID of the individual and the ID of the action for which the interaction was recorded. Depending on the destination type, this is handled in one of the following ways:

- For Facebook Ads and Google Ads, the click URL of an ad must contain the ad set or ad group ID. When the ID is returned after an individual clicks on an ad, Paid Media Manager checks the ad platform to determine which audience was used to target the ad.
- For other ad platforms, Pega Customer Decision Hub generates a unique identifier for each Next-Best-Action audience. This identifier is called the external audience key. When the ID is returned after an individual clicks on an ad, it tells Paid Media Manager which action the paid click is for.

All next-best-action audiences are created for a specific action, so identifying the audience is the same as identifying the original action. The advertiser's own identity graph can determine the customer ID of the individual based on their past actions, such as logging in, clicking on an email link, or completing a form.**Note:** If you do not have an existing identity matching solution, Pega Customer Decision Hub provides an identity management functionality that you can implement. For more information, see [Identity management for the Container REST service](#).

- [Enabling your website to capture paid clicks](#)
- [Capturing paid interactions](#)

Enabling your website to capture paid clicks

To capture information about the way in which individuals interact with your ads, do the following steps:

1. Use your first-party identity management solution to make the customer ID available for the *PaidCaptureResponse* service when a known individual visits your website.**Note:** If you do not have an existing identity matching solution, Pega Customer Decision Hub provides an identity management functionality that you can implement. For more information, see [Identity management for the Container REST service](#).
2. To determine the interaction ID that drove the ad click, implement the `webwb/RealTimeContainerScript.js` JavaScript file on your

website to call the `capturePaidClickResponseExt` function, which communicates the customer ID and action ID to Paid Media Manager. For more information, see [Configuring a container invocation script](#).

Note: Specific ad platforms have different requirements for configuring the click URL. Refer to the destination-specific sections of this article for more information.

For an example of a service call, see [Enabling your website to capture paid clicks](#).

Available parameters

- **CustomerID** - Required. Used to identify individuals based on information captured during their interactions with the destination.
- **Adset_id** - Required for Facebook Ads and Google Ads. Used to identify which next-best-action paid audience was used to target the individual with the ad, resulting in the click. The parameter corresponds to the ad set on Facebook Ads, and ad group on Google Ads.
- **ExternalAudienceId** - Required for other ad platforms, for example, LinkedIn Ads. Used to identify which next-best-action paid audience was used to target the individual with the ad, resulting in the click. For more information, see [Retrieving the external audience ID](#).
- **DestinationType** - Optional. Used to identify the destination type. You can specify it if you are sure about the source of the click (that is, if you can identify the source platform either as Facebook Ads or Google Ads). If the **ClickThroughURL** parameter contains the **adset_id** (Facebook) or **adgroup_id** (Google), Paid Media Manager can also automatically recognize the destination type as Facebook or Google.
- **ReferrerUrl** - Optional. The HTTP referrer of the requested website. You can obtain by using the **document.referrer** web API.

The service call must include the callback function. The function is required to process the response coming from the Paid Web Service. It inserts data into the following Interaction History tables:

- The **referrerurl**, **utm_medium**, **externalaudienceid** columns of the *pr_data_ih_fact* table
- The **pydirection**, **pychannel**, **channelgroup**, **channelsubgroup** columns of the *pr_data_ih_dim_channel* table
- The **pyoutcome** and **pybehaviour** columns of the *pr_data_ih_dim_outcome* table. The value of **pyoutcome** is always set to PaidClick and the value of **pybehaviour** is set to Positive.

After the paid click outcomes are recorded in Interaction History and the Adaptive Analytics data set, you can use the information in your reporting, modeling, and Next-Best-Action.

Retrieving the external audience ID

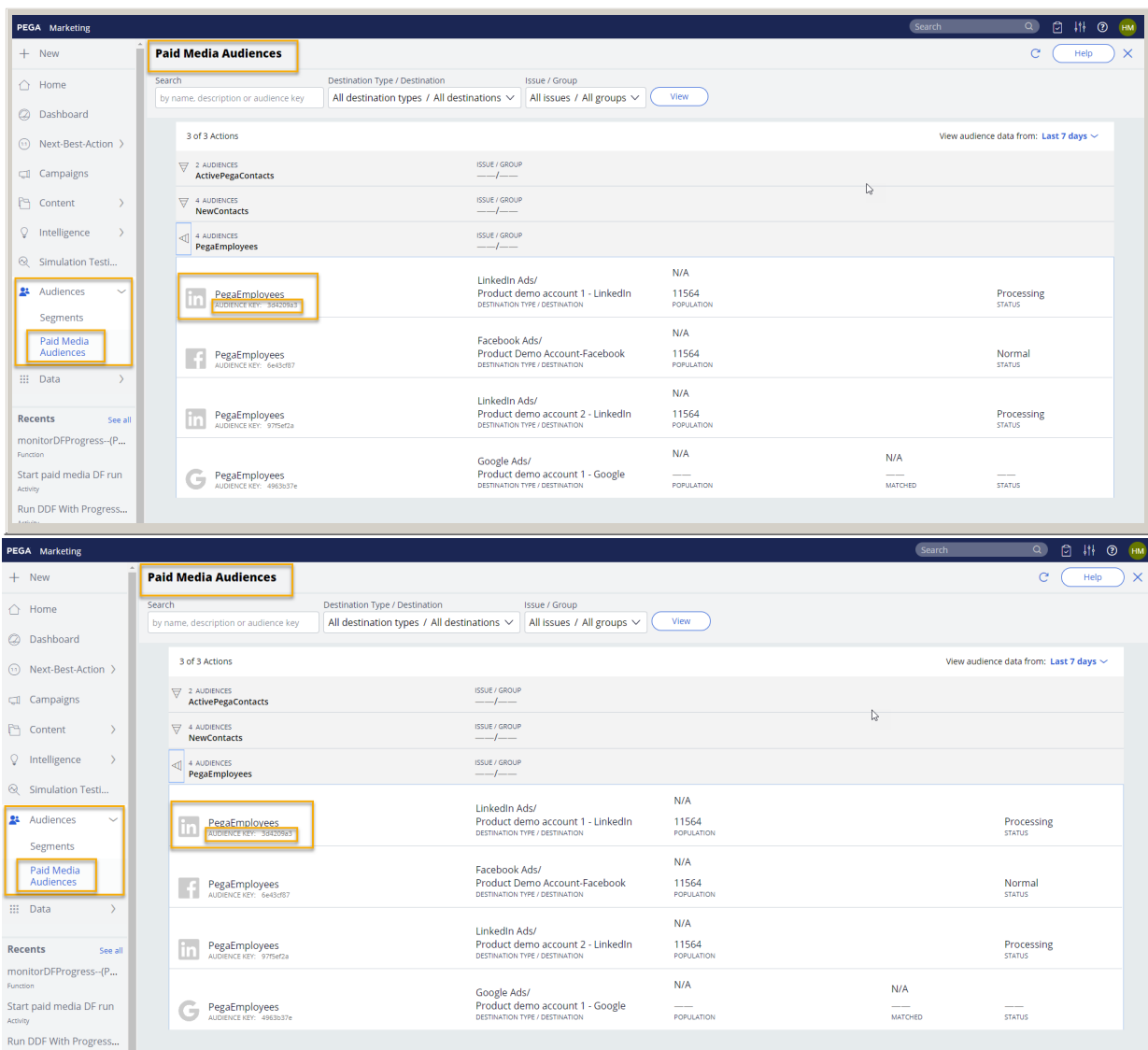
For platforms other than Facebook Ads and Google Ads, the external audience ID is required on the destination click URL for to determine which audience resulted in the click. It acts as a unique identifier, enabling you to determine the action for which the audience was created.

For Facebook Ads and Google Ads, you can instead use the ad set/group ID, which you can configure within Facebook or Google Ads to be dynamically added to the destination click URL. Paid Media Manager can determine the NBA paid audience that drove the click based on this parameter, so for these ad platforms you do not need to manually modify the destination click URL for each ad.

To retrieve the external audience ID, do the following steps:

1. Use Paid Media Manager to create paid audiences that are based on an existing customer segment, outbound runs, or real-time container updates.
For more information, see [Extending next-best-action to digital advertising platforms with Paid Media Manager](#).
2. After the run is finished, in the Pega Customer Decision Hub portal, click **Audiences Paid Media Audiences**.
3. Find the audience that was created in the ad platform and note the ID in the **Audience key** field, as in the following figure:

External audience ID on the Paid Media Audiences page



4. In the ad platform, for example, LinkedIn Ads, edit or create an ad and add the external audience ID to the destination URL, for example, https://cdh.uplusbank.com?Site=PMM&ext_aud_id=3d4209a3&ref_url=linkedin.com.

Result: If an individual clicks on an ad resulting from a Next-Best-Action decision, the response is available in Interaction History and the adaptive analytics data set.

Sample service call for the CapturePaidResponse REST service

CapturePaidResponse is a REST service that captures customer responses to actions in the individuals' Interaction History. The responses are posted to adaptive analytics in order to allow adaptive models to learn. The service supports the POST HTTP method.

The following is an example of JSON data for a POST request for a LinkedIn Ads paid click: {"CustomerID": "1234", "ExternalAudienceId": "a12sd", "ReferrerUrl": "linkedin.com", "Utm_medium": "LinkedIn", "AdSetId": "", "DestinationType": "LinkedIn", "Outcome": "PaidClick"}

Capturing paid interactions

The following are instructions to capture paid interactions from Facebook Ads, Google Ads, LinkedIn Ads, and other paid destinations.

Capturing paid interactions from Facebook Ads

To capture paid interactions, configure the Facebook Ads website URL to redirect to your website or microsite.

1. Log in to [Facebook Ads Manager](#).
2. In the ad details, click **Build a URL Parameter** and specify the following URL using dynamic parameters provided by Facebook: `adset_id={{adset.id}}&utm_medium={{placement}}&ref_url={{site_source_name}}`
For more information about dynamic URL parameters, see [About URL parameters](#).
3. Test that the paid interactions are captured in Interaction History. For more information, see [Capturing paid](#)

[interactions.](#)

Capturing paid interactions from Google Ads

To capture paid interactions, configure the Google Ads final URL to redirect to your website or microsite.

1. Log in to [Google Ads](#).
2. In the ad details, configure the destination click URL for the ad:
 - a. In the **Final URL** field, enter the URL of the website or microsite to which the individual should be redirected on clicking the ad.
 - b. In the **Ad URL options** section, in the **Tracking template** field, specify the URL using valuetrack parameters provided by Google Ads: `{lpurl}?adgroup_id={adgroupid}&ref_url={_refUrl}&utm_medium={_utmMedium}`
The **{adgroupid}** parameter is the ID of the ad group to which the ad belongs. Paid Media Manager uses it to determine the action for which the click was recorded. When an individual clicks the ad, the parameter is replaced by the actual value and passed in the request. You can include other valuetrack parameters as necessary. For more information about valuetrack parameters, see [Set up tracking with ValueTrack parameters](#).
 - c. Click **Save and create extensions**.
3. Test that the paid interactions are captured in Interaction History. For more information, see [Capturing paid interactions](#).

Capturing paid interactions from LinkedIn Ads and other platforms

To capture paid interactions, configure the LinkedIn Ads final URL to redirect to your website or microsite.

The instructions below are specific to LinkedIn, but you can use the same URL parameters to configure the destination click URL for other ad platforms.

1. Log in to your LinkedIn account.
2. In the ad details, configure the destination click URL for the ad.
The proper format of the URL is `<microsite URL>?ext_aud_id=<enter_external_audience_id>&ref_url=linkedin.com`. For more information about the external audience ID, see [Enabling your website to capture paid clicks](#).
3. Test that the paid interactions are captured in Interaction History. For more information, see [Capturing paid interactions](#).

Verifying that paid clicks are recorded in Interaction History

After configuring the *CapturePaidResponse* service for your website, test that the interactions are captured in Interaction History.

1. Identify as a known individual, so that your customer ID is available to the *CapturePaidResponse* service.
2. Enter a sample ad URL to simulate a paid click, for example, `https://cdh.uplusbank.com?Site=PMM&ext_aud_id=3d4209a3&ref_url=youtube.com`
3. Verify that the interaction was captured in Interaction History by doing one of the following steps:
 - In Dev Studio, click **Configure Decisioning Monitoring Interaction History Recent Interactions**, and then verify that the paid click shows in the list
 - Run the following queries in your Pega Customer Decision Hub database client to check that the click is available in Interaction History:
`select referrurl, utm_medium, externalaudienceid from pegadata.pr_data_if_fact;select pydirection, pychannel, channelgroup, channelsubgroup from pegadata.pr_data_ih_dim_channel;select pyoutcome, pybehaviour from pegadata.pr_data_ih_dim_outcome;`

Result: Once you have enabled paid clicks to come into Pega Customer Decision Hub, you can use those paid interactions to enable AI-driven paid strategies in Paid Media Manager.

For example, you can use the interactions recorded by the *PaidCaptureResponse* service to predict whether clicking on an ad would increase an individual's propensity to accept an offer. With that information available, you can reduce wasted ad spend by aligning the willingness to pay with the predicted value of a click. For more information about this and other use cases, see [Reducing wasted ad spend based on the predicted value of a click](#).

Enabling the offline conversion job scheduler for Paid Media Manager

To enable Paid Media Manager to send data about offline conversions to CRM-based destination platforms, enable the OfflineConversionRTTrigger job scheduler.

The OfflineConversionRTTrigger job scheduler runs at specific intervals (by default, once every hour) and triggers the synchronization RealTimeOfflineConversionTriggerSync activity that sends [conversion event](#) data to a destination platform. The job scheduler is disabled by default. You must enable it to synchronize your offline conversion data.

1. Log in as an operator with access to Admin Studio.
2. In Admin Studio, click **Resources Jobs**.
3. Find and open the OfflineConversionRTTrigger job scheduler.
4. Set the scheduler to **Enabled**.
5. Save your changes.

Enabling learning from own-channel models for the Paid channel

You can enable surrogate adaptive models for paid media while the Paid-specific models mature.

Channel-specific adaptive models in Pega Customer Decision Hub learn from the outcomes of recorded customer interactions, to help the models make predictions about customer behavior and guide the next-best-action strategy to present the most relevant actions to each customer on each channel. To ensure that the models have enough data to make accurate predictions, adaptive models for individual channels are not considered mature until they have recorded a specific number of responses. Immature models are not used to drive decisions – instead, the action selection is randomized for strategies that use an immature model.

As with any new channel, when you implement Paid Media Manager for the first time, the Paid-specific adaptive models are not yet mature enough to drive next best actions. If you already have mature adaptive models for other engagement channels, for example, Web, you can use them to inform the paid channels while the paid models mature. **Note:** In the current version of Pega Customer Decision Hub version 8.6, the paid-specific models are disabled by default because of performance impact. For more information, see [Known issues in Pega Customer Decision Hub 8.6](#) and [Enabling learning from Paid-specific models for the Paid channel](#).

To enable the surrogate adaptive model for the Paid channel, perform the following steps:

1. Log in to Pega Customer Decision Hub as an operator with access to Dev Studio.
2. In Dev Studio, in the left navigation pane, click **Records Decision Data**.
3. Find and open the *BusinessControlSettings* DDR that is in your implementation ruleset.
4. Click **Check out**.
5. Click **New**.
6. Enter the following information:
 - **Name** - Paid Media Surrogate Channel
 - **Model Name** - NBA
 - **SettingName** - PaidMediaLearningChannel
 - **SettingValue** - *<Direction>/<Channel>*, for example, Inbound|Web

Sample configuration with Web as the surrogate channel

Home BusinessContr... Decision Data Manage propos...

Create proposition Create Cancel ?

BUSINESS ISSUE GROUP

Name ★ Description

Paid Media Surrogate Channel

Model Name

NBA

SettingName

PaidMediaLearningChannel

SettingValue

Inbound|Web

7. Click **Create**.
8. Save and check in the changes to the *BusinessControlSettings* DDR.

What to do next: After the Paid model has received enough responses to be considered mature, you can disable the surrogate channel by editing the *BusinessControlSettings* DDR and removing the **Paid Media Surrogate Channel** proposition.

Enabling learning from Paid-specific models for the Paid channel

If required, you can enable channel-specific adaptive models for paid media to help optimize your advertising based on responses from the Paid channel.

Unlike [surrogate channel models](#), a Paid-specific model also enables the strategy to take the type of paid destination into account as one of the parameters, so that the same action can have a different priority for YouTube, Instagram, and LinkedIn. This accounts for the difference in how each individual is influenced by each destination for each action.

To enable Paid-specific models, perform the following steps:

1. In Dev Studio, create a dynamic system setting with the following values:
 - **Purpose** - dataflow/paidmedia/paidstrategy/enableStoreDecisionResults
 - **Ruleset** - PegaMKT-Engine
 - **Value** - true
2. If your Next-Best-Action strategy uses real-time containers or real-time event triggers, refresh the Next-Best-Action Designer configuration by performing the following steps:
 - a. In the Pega Customer Decision Hub portal, click **Next-Best-Action Designer Channels Edit**.
 - b. Click **Save**.

Note: You do not need to refresh the Next-Best-Action Designer configuration if you use only outbound runs.
3. **Optional:** To disable delayed learning again, change the value of the *enableStoreDecisionResults* dynamic system setting to false, and then edit and save the **Channels** tab in Next-Best-Action Designer again.

What to do next: For more information about configuring and running paid strategies, see [Extending next-best-action to digital advertising platforms with Paid Media Manager](#).

Configuring the PaidMediaSync queue processor to optimize performance

The PaidMediaSync queue processor manages the synchronization queue for Paid Media Manager. To help optimize performance, configure the queue processor as described below.

- If you use or plan to use Paid Media Manager to extend your next-best-action decisioning to paid ad platforms such as Facebook Ads or Google Ads, verify that the number of nodes per thread is optimized for the expected number of requests. For more information, see [Recommended number of threads per node for PaidMediaSync](#).
- If you do not plan to use Paid Media Manager, disable the PaidMediaSync queue processor. For more information, see [Disabling the PaidMediaSync queue processor](#).
- [Recommended number of threads per node for PaidMediaSync](#)
- [Disabling the PaidMediaSync queue processor](#)

Recommended number of threads per node for PaidMediaSync

By default, the queue processor has two threads per node. You can modify the number of threads based on the specifications of your environment.

Environment description	Recommended number of threads
• Production environment • Large customer segments • Real-time containers with millions of requests • Multiple paid destinations	5 Note: For best performance results, most production environments should use this option.
• Production environment • Small customer segments • Real-time containers with thousands of requests • Few paid destinations	2
Testing environment	Any number between 1 and 5, depending on testing loads and observed performance

For general information about increasing the performance of queue processors, see [How do I increase the performance of my queue processors?](#)

- [Changing the number of threads per node for PaidMediaSync](#)
- [Enabling background processing](#)

Changing the number of threads per node for PaidMediaSync

1. Log in to Pega Customer Decision Hub as an operator with access to Admin Studio.
2. In Admin Studio, click **Resources Queue processors**.
3. Click **PaidMediaSync**.
4. Verify that the number of threads per node matches your requirements.
5. If needed, save a copy of the *PaidMediaSync* rule into your application ruleset, and then change the number of threads to the required value.
6. Check in your changes.
7. If the queue processor displays any errors related to the system run time context, add your application to the system run time context, as described in [Enabling background processing](#).

Enabling background processing

Include your application in background processing to enable several functionalities of Pega Customer Decision Hub, such as file and database templates, the processing of event-triggered actions, resilience checks for customer segments, and others.

1. Log in to Dev Studio as an administrator.
2. In the navigation pane of Dev Studio, click **Records**.
3. Expand the **Application Definition** category, and then click **Application**.
4. On the **Application** tab, open the application that is associated with the *PegaNBAM:Agents* access group.
5. On the **Definition** tab, in the **Advanced** section, select the **Include in background processing** check box if it is not already selected.
6. After changing the setting, save your changes and restart the *pzStandardProcessor* queue processor.
Note: You do not need to restart the queue processor if **Include in background processing** was already selected for your application.
 - a. In Admin Studio, click **Resources Queue Processors**.
 - b. Find and restart the *pzStandardProcessor* queue processor.
7. **Optional:** To ensure that you added the application to the system runtime context, perform the following steps:
 - a. In the header of Dev Studio, click **Configure System General System Runtime Context**.
 - b. Verify that your application appears in the **Applications** list.

Disabling the PaidMediaSync queue processor

If you do not use Paid Media Manager, you should disable the PaidMediaSync queue processor to help optimize performance.

1. Log in to Pega Customer Decision Hub as an operator with access to Admin Studio.
2. In Admin Studio, click **Resources Queue processors**.
3. Click **PaidMediaSync**.
4. Disable the *PaidMediaSync* rule by doing one of the following steps:

- If the *PaidMediaSync* rule opens from your application ruleset, clear the **Enable Queue Processor** check box.
 - If the *PaidMediaSync* rule opens from the PegaMKT-Engine ruleset, save a copy of the rule into your application ruleset, and then clear the **Enable Queue Processor** check box.
5. In Admin Studio, verify that the status of the PaidMediaSync queue processor is Disabled.

Defining triggers for paid media

For consistent results across owned and paid channels, you can use Next-Best-Action to drive customer engagement on external media platforms such as Facebook Ads and Google Ads. Using Next-Best-Action with paid media ensures that each individual in an audience is targeted with the optimal offers at the right priority and maximum bid amount.

For more information, see [Triggering paid updates through Next-Best-Action](#).

Implementing Pega Next-Best-Action Advisor

Pega Next-Best-Action Advisor provides a configurable, ready-to-use user interface that delivers next best actions for inbound channels. You can control and configure Pega Next-Best-Action Advisor from through Next-Best-Action Designer in Pega Customer Decision Hub.

Set up Pega Next-Best-Action Advisor to simplify your customer acquisition and retention journeys, create personalized recommendation that utilize AI-driven historical and real-time data, and empower your business with a configurable and adaptable way to present and negotiate recommendations.

For more information, see the *Pega Next-Best-Action Advisor Implementation Guide* on the [Pega Next-Best-Action Advisor product page](#).

Locking the release

After all merges are completed, the release engineer locks the applications and rulesets to be released. They are also responsible for creating the new, higher-level ruleset versions and higher-level application rules for the next release.

To lock the release, do the following steps:

1. In Dev Studio, click **Application Structure Ruleset Stack**.
2. Click **Lock & Roll**.
Note: As a best practice, lock your built-on applications first, and then work your way up the stack to your top-level application. This way, as each higher version application rule is created, you can open that rule and update the version of the built-on application.
3. For each ruleset:
 - a. Click **Lock** and provide a password.
 - b. Click **Roll**.
4. Click **Create a new version of my Application**.
5. Click **Run**.

Result: The application rules and ruleset versions for the current release are locked and require passwords to make changes. Also, you will have created the higher-level ruleset versions and application rules that will be used for the next release.

- [Implementing Pega Customer Decision Hub](#)
- [Deployment architecture for Pega Customer Decision Hub](#)

Configuring higher environments

Package the entire Pega Customer Decision Hub application that you deployed in the Development environment and migrate it to higher environments: Staging, Business Operations Environment (BOE), and Production.

1. [Getting ready for the initial deployment](#)
2. [Getting ready for enterprise changes](#)
3. [Getting ready for business changes](#)
4. [Creating operators in the Business Operations Environment](#)

Getting ready for the initial deployment

When you first deploy your applications from Development to higher environments, you must create a Product rule that includes the overlay application, the Pega 1:1 Operations Manager extension application, and the Pega Customer Decision Hub implementation application.

1. Create the Product rule for initial deployment.
Keep the following considerations in mind:
 - Your Customer Decision Hub implementation application that is being migrated should be built on the Deployment Manager application.
 - Operator records and relevant records associated to your implementation rulesets will be included as data

instances in the product.

- Do not force-include work objects. This includes campaign instances, the next-best-action schedule, change requests, simulation work objects, and others.
- If the Product rule includes open ruleset to get migrated, select **Allow unlocked ruleset versions** when creating the Product rule.
- Preview the Product rule before creating the RAP file. Verify there are no test rules included.
- Include the Next-Best-Action Designer rules in the Overlay application so that these rules can be managed from the BOE instance.

For example: In the following figure, the *UPlusBank* application is the implementation application of Customer Decision Hub.

RuleSets to include						
	Name	Minimum version	Maximum version	Exclude non-versioned rules	Include associated data	App context
1	UPlusBank-Artifacts	01-01		☐	☒	UPlusBank 01.01.01
2	UPlusBank-NBADTests	01-01		☐	☒	UPlusBank 01.01.01
3	UPlusBank-Rules	01-01		☐	☒	UPlusBank 01.01.01
4	UPlusBank	01-01		☐	☒	UPlusBank 01.01.01
5	UPlusBankint	01-01		☐	☒	UPlusBank 01.01.01
6	UBan	01-01		☐	☒	
7	UBanint	01-01		☐	☒	

RuleSets to include						
	Name	Minimum version	Maximum version	Exclude non-versioned rules	Include associated data	App context
1	UPlusBank-Artifacts	01-01		☐	☒	UPlusBank 01.01.01
2	UPlusBank-NBADTests	01-01		☐	☒	UPlusBank 01.01.01
3	UPlusBank-Rules	01-01		☐	☒	UPlusBank 01.01.01
4	UPlusBank	01-01		☐	☒	UPlusBank 01.01.01
5	UPlusBankint	01-01		☐	☒	UPlusBank 01.01.01
6	UBan	01-01		☐	☒	
7	UBanint	01-01		☐	☒	

BizChanges is the overlay application, *UplusOps* is the implementation application of 1:1 Operations Manager, and *FinancialServicesxCARDataModel* is the data model included in the Customer Decision Hub implementation application.

Edit Product: Initial Migration Package [Available]								
ID: InitialMigrationPackage - 01-01-01 RS: CDHDepl01-01-01								
Contents Dependencies Deployment History								
Applications and Components to include								
	Name	Version	Include associated data	Custom/Production rulesets	Delta mode	Include rule history	Include data types	Update dependent applications
1	UPlusOps	01.01.01	☒	☐	☐	☒	☒	☐
2	BizChanges	01.01.01	☒	☐	☐	☐	☒	☐
3	FinancialServicesxCARDataModel	21.04	☒	☐	☐	☐	☒	☐

Edit Product: Initial Migration Package [Available]								
ID: InitialMigrationPackage - 01-01-01 RS: CDHDepl01-01-01								
Contents Dependencies Deployment History								
Applications and Components to include								
	Name	Version	Include associated data	Custom/Production rulesets	Delta mode	Include rule history	Include data types	Update dependent applications
1	UPlusOps	01.01.01	☒	☐	☐	☒	☒	☐
2	BizChanges	01.01.01	☒	☐	☐	☐	☒	☐
3	FinancialServicesxCARDataModel	21.04	☒	☐	☐	☐	☒	☐

2. Ensure that all rules are checked in.
3. Ensure that your Product rule contains all the required rulesets and any data instances.

4. In the Development environment, do the following steps:
 - a. Ensure that the Customer Decision Hub implementation application and the 1:1 Operations Manager extension application have the *PegaDevOpsFoundation* version 5 or higher as the built-on application.
 - a. Ensure that *DMreleaseadmin_OAuth2* contains the latest generated secret key and access/revoke tokens. For more information, see [Setting up candidate environments](#).
5. In the Sandbox, BOE, and Production environments, do the following steps:
 - a. After setting up the instances with the latest Pega Platform and Customer Decision Hub build, ensure that the repository is configured.
 - b. Ensure that the Authentication profiles have the latest generated secret keys and access/revoke tokens. For more information, see [Setting up candidate environments](#).
6. Set up an orchestrator to manage the deployment of changes through the change pipelines.

Note: In Pega Cloud Services deployments, the default orchestrator is Deployment Manager. For more information, see [Setting up the orchestrator](#). Keep the following considerations in mind:

 - Create the Deployment pipeline with the overlay application, and set the access group to the Revision Manager access group for the overlay application.
 - Configure the pipeline with the Development, Staging and Production environments.
 - Use the Product file created in the Development environment for first-time deployment.
 - After you click **Start deployment**, override any aged updates and continue.
 - After successful deployment, disable the first-time deployment pipeline.

For more information, see [Setting up repositories](#) and [Quick-start guide for Deployment Manager on Pega Cloud Services](#).

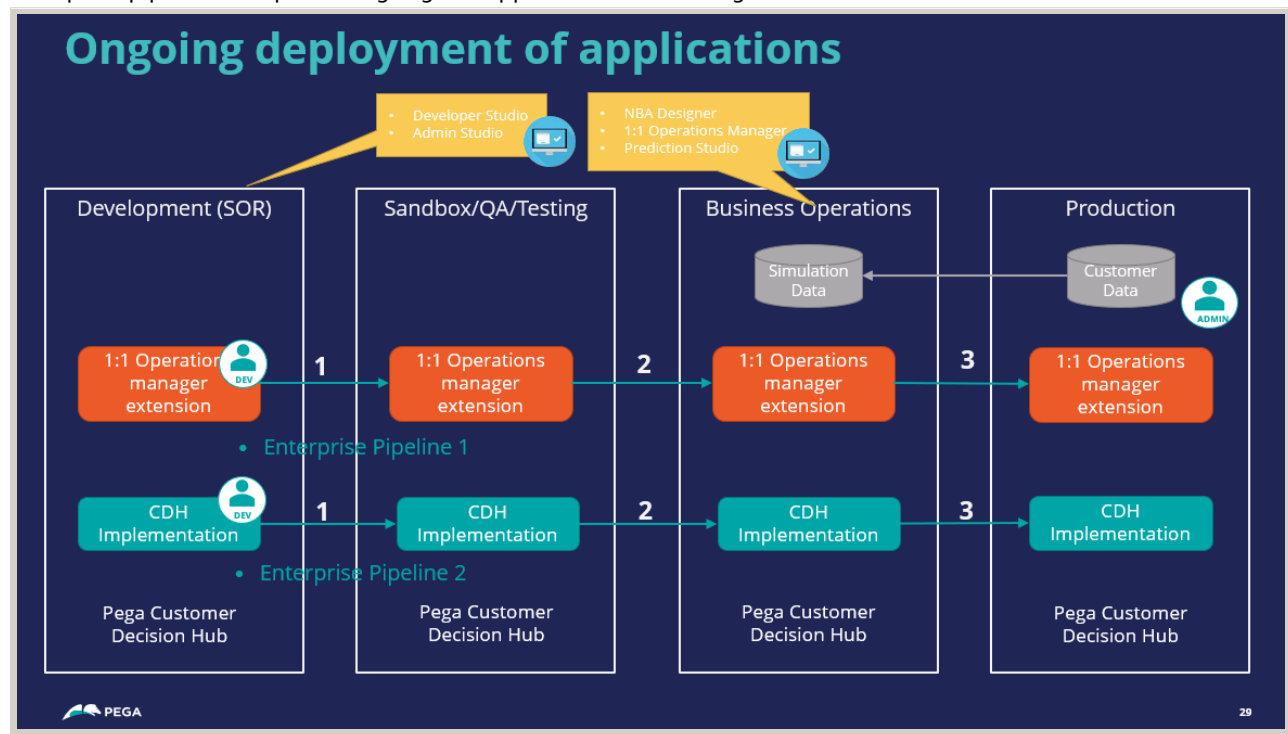
Getting ready for enterprise changes

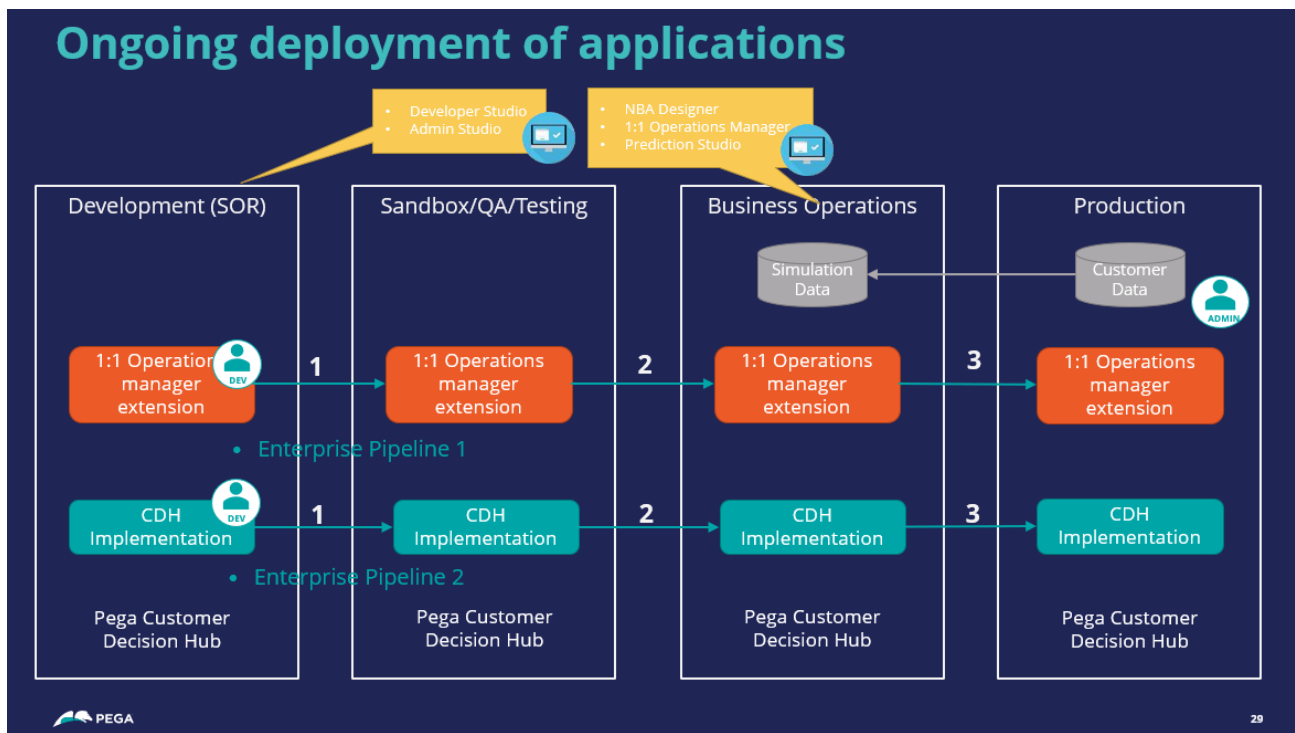
Support ongoing enhancements of your applications by creating Product rules for the Pega 1:1 Operations Manager extension application and the Pega Customer Decision Hub implementation application.

Customer Decision Hub implementation application

To support the ongoing enhancements of your Customer Decision Hub application, create a product file that only includes the rulesets for the Customer Decision Hub implementation application. You can then use the product file in the Enterprise pipeline of Deployment Manager, as in the following figure:

Enterprise pipelines for promoting regular application-level changes





You must create a product rule file in the Development (SOR) environment which generates the artifacts to the repository, and is used to promote the changes to other environments. The product rule needs to include all relevant rulesets and data instances except the application itself.

1:1 Operations Manager extension application

To support the ongoing enhancements of 1:1 Operations Manager, create a product file that only includes the rulesets for the 1:1 Operations Manager extension application. You can then use the product file in the Enterprise pipeline of Deployment Manager.

Note: You do not need to migrate the 1:1 Operations Manager overlay application, because no development should ever be carried out in the overlay application.

You must create a product rule file in the Development (SOR) environment which generates the artifacts to the repository, and is used to promote the changes to other environments. The product rule needs to include all relevant rulesets and data instances except the application itself.

Getting ready for business changes

Once your system is under change management, you must ensure that all next-best-action artifact changes are carried through a change request process. This means that changes to actions, treatments, next-best-action configurations, strategies and other artifacts should be done by using either 1:1 Operations Manager change requests, or standard Pega Platform change requests.

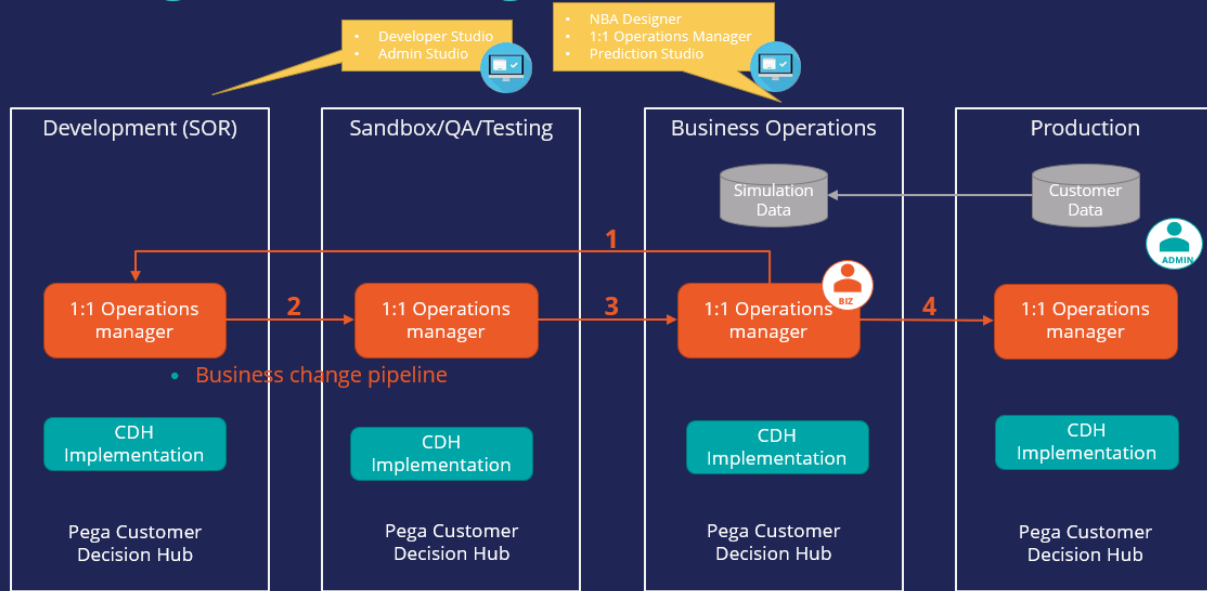
Once deployment is triggered from the Business Operations Environment, the change request branch is published to the repository and is queued for a merge. The orchestrator assigns the task of merging to Development (SOR) environment, and creates a new enterprise application version based on the changes.

You must create a product rule file in the Development (SOR) environment that generates the artifacts to the repository, and is used to promote the changes to other environments.

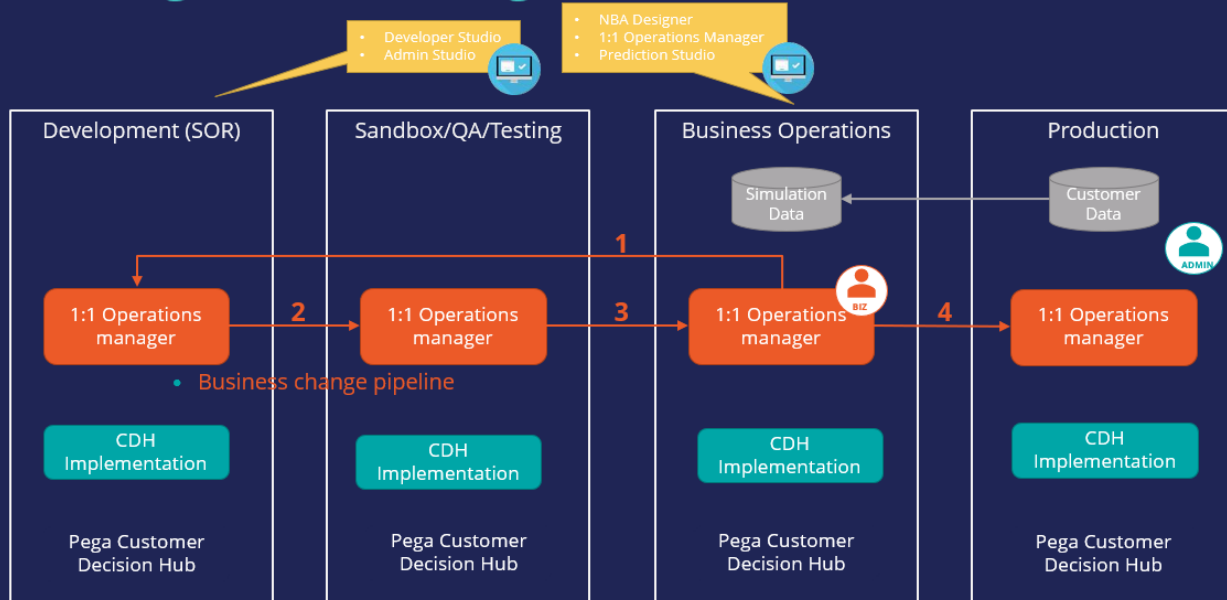
In the Production environment, you can publish the new application version to a group of test users for verification, and then activate or discard the changes for all users. After activating the changes in the Production environment, the application is updated with the latest version.

Business pipeline for promoting regular application-level changes

Moving business changes



Moving business changes



Creating operators in the Business Operations Environment

Create operators in the Business Operations Environment (BOE) to make non-disruptive, low-level business changes to your application, such as adding new offers or messages, modifying engagement policies, or enabling integrated channels.

1. Create the required operators:
 - Operations team operators to log in to Pega 1:1 Operations Manager.
 - Revision Manager and Strategy Designer operators to log in to Customer Decision Hub.

For more information, see [Creating an operator ID](#).

- [Business operations use cases](#)
- [Deployment architecture for Pega Customer Decision Hub](#)

Next steps

After a system architect completes the implementation of Pega Customer Decision Hub, the next steps can be to configure the Deployment Manager and test the implementation of environments and pipelines. Also, the business team can configure the necessary artifacts that they need to solve real business problems.

- Configure the Deployment Manager and pipelines, so that you can automatically export simulation data from a production system and import it into a Business Operations environment. **Note:** Deployment Manager is used by default on Pega Cloud. For non-Pega Cloud implementations, you can use other tools, such as Jenkins.

For more information, see [Configuring Deployment Manager for data migration pipelines](#).

- Test your implementation. Verify whether your environments and pipelines work as designed by running a change management test case and deploying the associated revision to production.

For more information, see [Testing your implementation](#).

- Configure business issues and groups, channel limits and constraints, actions, treatments, and decision strategies by using the Next-Best-Action Designer.

For more information, see [Understanding Next-Best-Action Designer basics](#).