NewgenONE

Rule Builder

User Guide

Version: 2024.2

Disclaimer

This document contains information proprietary to Newgen Software Technologies Ltd. User may not disclose or use any proprietary information or use any part of this document without written permission from Newgen Software Technologies Ltd.

Newgen Software Technologies Ltd. makes no representations or warranties regarding any software or to the contents or use of this guide. It also specifically disclaims any express or implied warranties of merchantability, title, or fitness for any particular purpose. Even though Newgen Software Technologies Ltd. has tested the hardware and software and reviewed the documentation, it does not guarantee or imply that this document is error free or accurate regarding any particular specification. As a result, this product is sold as it is and user, the purchaser, is assuming the entire risk as to its quality and performance. Further, Newgen Software Technologies Ltd. reserves the right to revise this publication and make changes in its content without any obligation to notify any person, of such revisions or changes. Newgen Software Technologies Ltd. authorizes no Newgen agent, dealer or employee to make any modification, extension, or addition to the above statements.

Newgen Software Technologies Ltd. has attempted to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Newgen Software, Registered Office, New Delhi
E-44/13
Okhla Phase- II
New Delhi 110020
India
Phone: +91 1146 533 200
info@newgensoft.com

Contents

Preface

This chapter provides information about the purpose of this guide, details on the intended audience, revision history, and related documentation for NewgenONE Rule Builder.

Revision history

Revision date	Description
November 2024	Initial publication

About this guide

This guide explains the creation and management of rules using the NewgenONE Rule Builder. It describes how a business user can deploy business rules to mission-critical processes, ensuring system integrity and smooth operation. It also discusses the creation and management of rule flows and deployment of rules as SOAP or REST API. This guide also provides information on the integration of Rule Builder with NewgenONE Process Designer and the execution of REST services.

To ensure you are referring to the latest and most recent revision of this guide, download it from one of the following locations:

- Newgen Internal Doc Portal, if you are a Newgen employee.
- Newgen Partner Portal, if you are a Newgen partner.

Intended audience

This guide is intended for rule designers and business users who configure simple or complex customized rules using a low-code platform (visual approach) without having to dig into code details. The reader must understand logically driven statements such as IF, ELSE, and ELSE IF, as well as mathematical operators and comparator symbols. The user must also know how to navigate a user interface (UI) based configuration software and specify conditions, actions, and statements for rules and other artifacts. The user must have the necessary rights to access the NewgenONE Rule Builder module.

You must have a working knowledge of the following:

- The fundamentals and standard practices of your business area.
- Business rule management operations
- NewgenONE Automation Studio

Related documents

The following documents are related to NewgenONE Rule Builder:

- NewgenONE Overview Guide
- NewgenONE Automation Studio User Guide
- NewgenONE Data Model Designer User Guide
- NewgenONE Process Designer User Guide
- NewgenONE Deployment Admin User Guide
- NewgenONE Developer Guide
- NewgenONE Rule Builder Microservices Developer Guide

Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, write an email to docs.feedback@newgensoft.com.

To help capture your feedback effectively, share the following information in your email:

- Document name
- Version
- Chapter, topic, or section
- Feedback or suggestions

Introduction

This chapter provides an overview of the NewgenONE Rule Builder. It also provides information on the framework and components of business rules.

The chapter includes the following topics:

- Overview of Rule Builder
- Rule Builder framework
- Components of business rules

Overview of Rule Builder

Business rules are the foundation of every business operation. NewgenONE Rule Builder allows you to author and manage rules that help in automating business decisions. You can define English rules to implement simple logic and Decision rules to implement complex business logic. It also allows you to define the sequential execution of rules using rule flows. Rule Builder enables the deployment of rule packages as REST or web services. The test functionality lets you verify if rule execution is producing desired results. Using Rule Builder, you can easily modify and re-implement rules with respect to changes in business logic and policies.

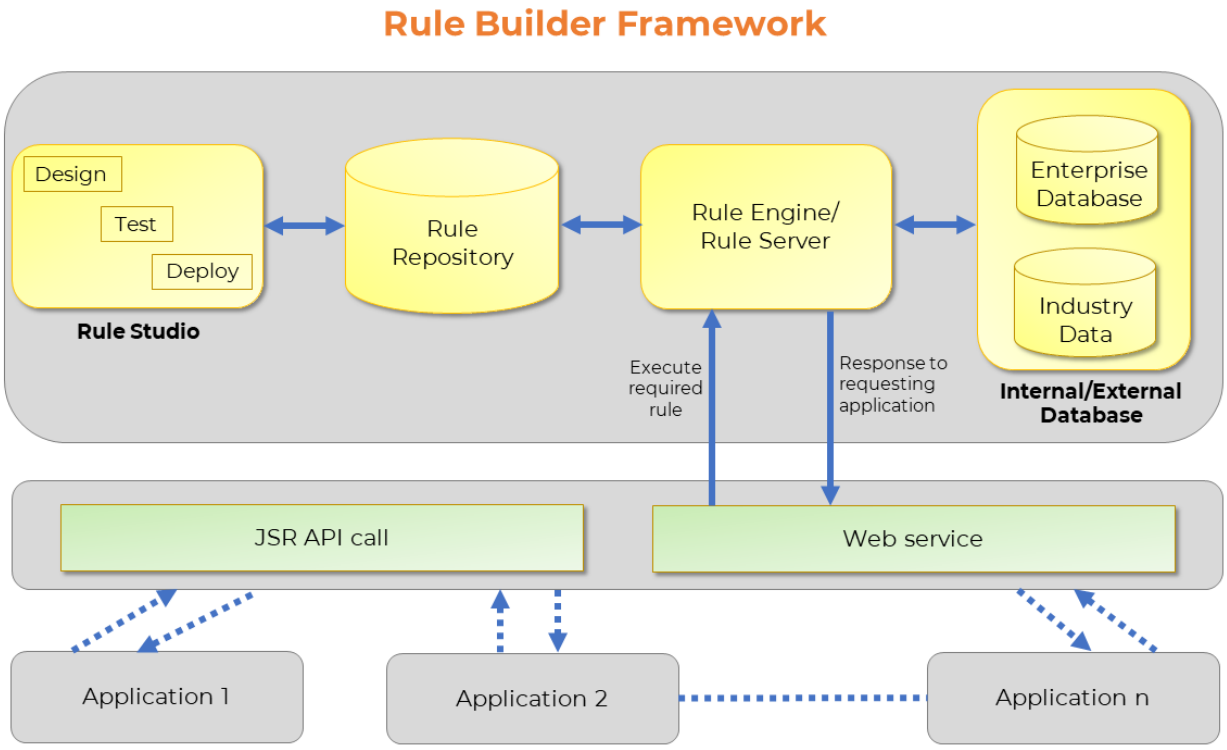
Business rules consist of clearly defined inputs and outputs, which execute associated rule services when invoked. Rule Builder consists of a complete set of tools to create, maintain, and integrate rules, using an intuitive interface.

The Rule Builder offers clear segregation of business logic from the corresponding code. Legacy solutions adopt an approach of hard coding, thereby, making the modification of business logic a specialist-programming job. As the organizational processes evolve and become more complex, making such changes become extremely difficult and a cost- and time-intensive activity. In a dynamic business environment that mandates agility, making such process changes becomes an even more complex task. Rule Builder not only enables the building and administration of business logic easy by using business rules but also goes beyond by enabling business users to manage business policies.

The Rule Builder allow you to create and share entities and rules with each other. Rules once designed can be used in different applications, for example, eligibility rules defined for an organization can be used in different business applications such as credit card processing, loan processing, and many more. You may share and reuse entity definitions imported into the system.

Rule Builder framework

NewgenONE Rule Builder is a complete business rule management system for managing business rules through authoring, design, and deployment. It also provides easy maintenance of rules after deployment.



Applications that require business rules, such as loan applications, credit card applications, insurance applications, and more, use JSR94 APIs, rules, and REST services. The business rule engine or rule server is responsible for receiving calls from multiple clients, maintaining sessions, executing rules, and performing actions according to the business rules.

Business Rules also maintains data in a repository for all the calls made for a rule. This data can be used later for reporting purposes (if required). In addition, a log is maintained for each rule executed and rule management activities, which can be used to investigate the system if necessary.

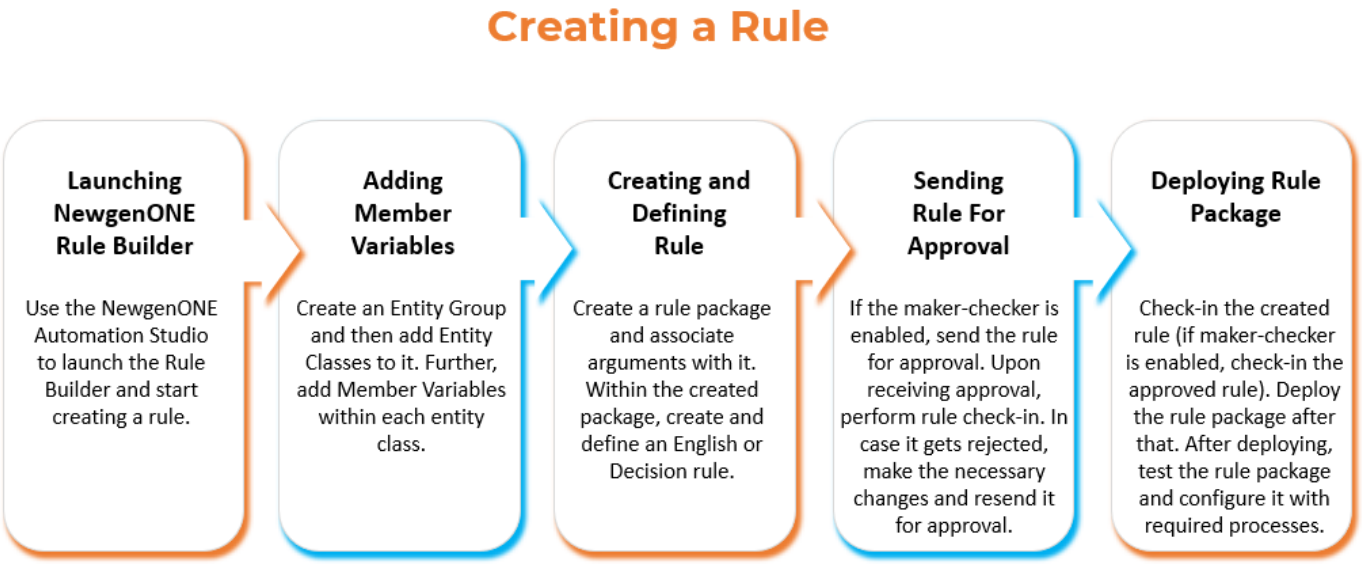
Components of Rule Builder

The following are the various components of Rule Builder:

- **Rule studio** — The rule studio is a web-based environment for designing, defining, testing, managing, and deploying rules. Repositories store rules created with rule studio. Rule Studio allows rights-based rule editing. The non-technical users can also easily work with the rule studio, eliminating the need to consult IT each time a rule logic needs to change.
- **Rule repository** — Business rules designed or created using NewgenONE Rule Builder, get stored in the rule repository. The rule repository is a database schema. Rule Builder supports Microsoft SQL servers. At the time of rule designing, define or use predefined interface objects from the rule repository. Whenever a business application requires some decision-making, a call is made, and the appropriate rule is fetched from the repository and executed by the engine. The rule repository updates every time whenever there is any update in a rule. Using Rule Builder, you can deploy the required rules as web services.
- **Rule engine or rule server** — Every time a business application requires some decision to be made depending upon a rule, the rule engine fetches the appropriate rule from the repository and executes the rule to generate the required output. Rule Builder engine is a J2EE-compliant engine based on EJB 2.0 specifications. The Rule Builder server provides the layer, which listens to the requests of business applications such as application and credit card applications, routes these requests to execution logic (or Web service call), and sends the data back to the calling application. The Rule Builder engine uses Web standard interfaces such as Web Services Definition Language (WSDL), XML Schema Definition (XSD), and Simple Object Access Protocol (SOAP) so that it can communicate with other applications based on open standards. Business Rules can also be integrated with any BPM application using the JSR 94 API interface and other interfaces.

By default, SOAP service configuration is disabled. To enable the SOAP service configuration, refer to the *Automation Studio* section of the *NewgenONE Configuration Guide*.

The basic steps involved in creating a business rule using NewgenONE Rule Builder are:



Getting started

This chapter describes how to access the NewgenONE Rule Builder module and walks you through the UI of its landing page.

The chapter includes the following topics:

Accessing Rule Builder

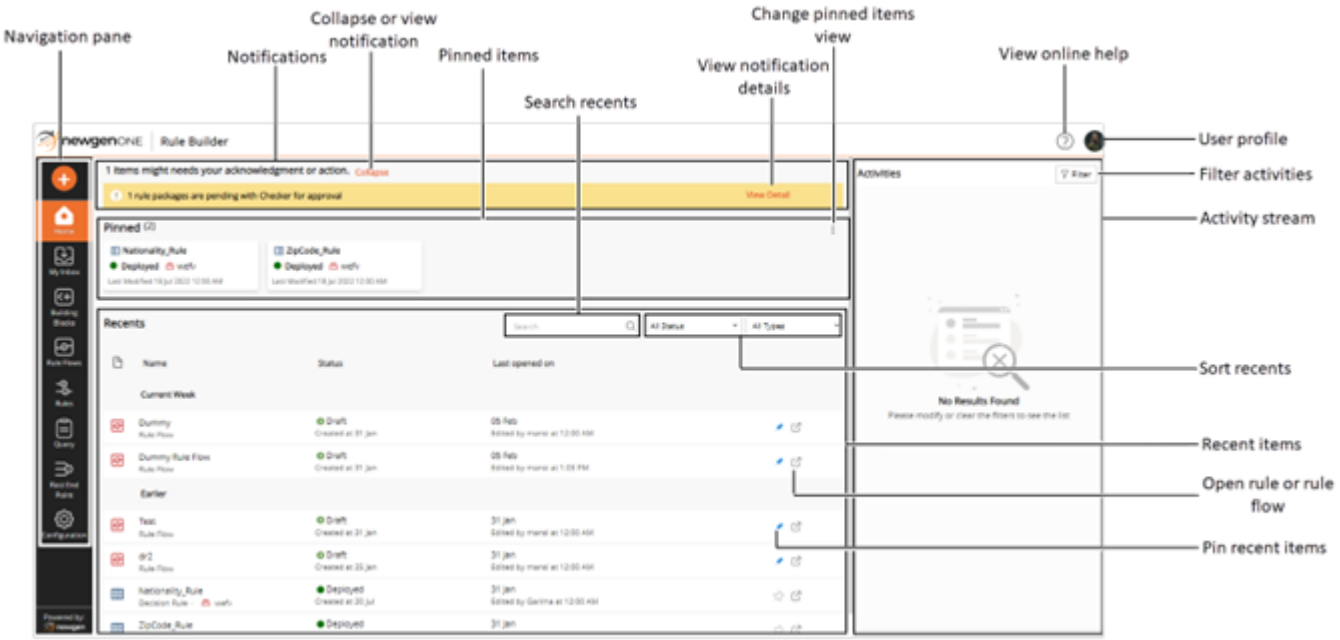
To access the NewgenONE Rule Builder module, follow the below steps:

1. Sign in to the NewgenONE Automation Studio using your valid user credentials. For more details, refer to the *NewgenONE Automation Studio User Guide*. On successful sign in, the Automation Studio home page appears.
2. Click **View All** against **Design Tools** and then select **Rule Builder**. The Rule Builder opens in a separate browser tab.

Exploring the Rule Builder interface

On successful sign-in, the Rule Builder module lands on the Home tab. It displays the list of pinned and recently actioned items. The notifications related to items requiring your acknowledgment appear. Also, a stream of activities performed on the artifacts such as building blocks, rule packages, rules, and rule flows appear.

Furthermore, you can navigate through the elements of the rule builder user interface using the Tab key.



The Home tab of the Rule Builder includes the following elements:

Element	Description
Navigation pane	Allows you to navigate between different tabs of the Rule Builder. The navigation pane includes the following: • My Inbox • Building Blocks • Rule Flows • Rules • Query • Rest End Point • Configuration
Notifications	Displays the items that need your acknowledgment or actions.
Collapse or view notifications	Click **Collapse** to collapse the notification flags. Similarly, click **View** to display and view notification flags.
View notification details	Click **View Details** to view complete details of the notification in My Inbox.
Pinned items	Displays the items pinned to the home tab for easy access. Click the required item to view its definition. Click **View All** to view the complete list of pinned items.
Change pinned items view	Click the ellipsis icon ⋮ to change the view of pinned items list. Select: • **Tile View** — to display pinned items in the form of tiles • **List View** — to display pinned items in the form of a list
Recent items	Displays the list of recently actioned items. Click the required item to view its definition.
Search recent items	Search the recently actioned item by name.

Element	Description
Sort recent items	Sort the list of recent by: • **Status**: ◦ **All Status** — to view all recent items irrespective of status. ◦ **Deployed Asset** — to view recently worked on deployed assets. ◦ **Draft Asset** — to view recently worked on draft assets. • **Types**: ◦ **All Types** — to view all recent items irrespective of type. ◦ **Rule Flow** — to view recently worked on rule flow. ◦ **English Rule** — to view the English rules you have recently worked on. ◦ **Decision Rule** — to view the Decision rules you have recently worked on.
Pin recent items	Click the pin icon to pin a recent item to the home tab.
Open rule or rule flow	Click the expand icon to open and view the rule or rule flow definition.
Activity stream	Lists the recent activities performed on the building blocks, rule flows, and rules managed in the cabinet.
Filter activities	Use the **Filter** option to filter the activities based on the artifact type, artifact name, and executioner. For more information, see the Filtering activities section.
View online help	Redirects to online help of the Rule Builder.
User profile	Click the user profile icon to view the signed-in user details and settings. To sign out from all the active sessions of the NewgenONE Rule Builder, use the Logout option.

Filtering activities

The Activities stream on the right displays the recent activities performed on the processes registered in the cabinet. Here, you can apply filters on the activities based on the artifact type, artifact name, and executioner.

To apply a filter on the latest activities, perform the following steps:

1. Click **Filter** on the Activities pane of the Home page. The Activity Filters dialog appears.
2. Enter the following filter criteria in the dialog:

Filter Criteria	Description
Artifact Type	Select a suitable type for the artifact. It can either be a rule package or rule flow.
Artifact Name	Select the name of the artifact from the dropdown list. This list varies depending on the chosen artifact type.
Action Performed By	Select the name of the executioner from the dropdown list.

3. Click **Apply** to view the results.

Working with building blocks

A building block is an entity variable, picklist, or external function used to define a rule or rule flow. For example, a rule "Age>18, then set Eligibility = True". Here, Age and Eligibility are the entity variables.
Before creating a rule, you must associate variables with the rule package as arguments, and then use them to define the conditions and actions of the rules. Once defined, the building blocks can be reused to define various types of rules.

This chapter includes the following sections:

- Entities
- Picklist
- External functions

Adding and managing entities

An entity is an object that defines a rule. Say in an entity group called "Bank", "Customer" is an entity.

For example:
If A > B, then set C=20. The entities in this rule are A, B, and C. You can reuse entities to create multiple rules.

Only a supervisor user or a member of the supervisor group can create, modify, or delete entities.

The steps to create entities involves creating:

- Entity group
- Entity class
- Member variables

Add and manage entity groups

Entity groups are also referred to as entity packages. They help in grouping logical entities belonging to a business type, for example, a bank, an insurance company, and more.

The list of entity groups appears on the left side of the screen. The name of the entity group along with the number of entity classes defined within it is displayed on the screen.

Use the search box to search the entity group by name.
Use the **Sort By** option to sort the list of entities group using one of the following criteria:

- **Last Modified** — Sorts the list in descending order as per the recent.
- **Name: A-Z** — Sorts the list in alphabetically ascending order.
- **Name: Z-A** — Sorts the list in alphabetically descending order.

Related topic(s)

- Creating an entity group
- Editing an entity group
- Copying an entity group
- Deleting an entity group
- Exporting an entity
- Exporting entities in bulk
- Importing an entity
- Viewing entity group audit trail

Creating an entity group

To create an entity group, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, click the create icon .
   Alternatively, use the create option present in the navigation pane to add or create an entity group. Click the create icon , then select **Add Entity Group**. The Create Entity Group dialog appears.
2. Refer to the below table to specify details in the fields:

Field	Description
Alias Name	Specify the name of the entity group. It is a mandatory field.
Group Name	Based on the alias name, the system generates a non-editable group name. This name is essential for integration purposes.
Description	Enter a description of the entity group.

3. Click **Save**. The entity group gets created and added to the list.

Related topic(s)

- Creating an entity class
- Creating a member variable

Editing an entity group

To edit an entity group, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.
2. Click the ellipsis icon and select **Edit**. The Edit Entity Group dialog appears.
3. Make the required modifications.

   You cannot modify the group name.

4. Click **Save**. The changes made to the entity group get saved.

Copying an entity group

Using the copy entity group option, you can create a new entity group from a copy of an existing one.

To create a new entity group using the copy option or create a copy of an existing entity group, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.
2. Click the ellipsis icon and select **Copy**. The Copy Entity Group dialog appears.
3. Specify **New Alias Name** for the entity group. Based on the alias, **Group Name** gets generated.
4. Under List of Entity Class, select the checkbox present against the entity class name that you want to add to the newly created copy.
   Select the **Select All** checkbox to add all the entity classes to the copy. When you select an entity class, the list of member variables defined within it appears adjacent to it. Select the checkbox present against the member variable name that you want to add to the newly created copy.
   Alternatively, select the **Select All** checkbox to add all the member variables to the copy.
5. Click **Copy**. A new copy of the existing entity member gets created and added to the list. Conflict(s) might appear due to the use of an Alias Name which is already in use. In this case, change the Alias Name and then click **Copy**.

Deleting an entity group

To delete an entity group, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.
2. Click the ellipsis icon and select **Delete**. The Confirm Deletion dialog appears.

   Deleting an entity invalidates all rules that are using the entity. Invalidated rules must be modified to work further. If you delete some entity member or entity classes associated

with a rule package or rule flow, you must remove the associated entity from the argument list.

3. Click **Delete**. The entity group gets deleted from the system.

Exporting an entity

The export feature is used to synchronize the data of two entities on separate servers. The options available to export Entity Class, Entity Member, and Audit Trail. On successful export, the .zip file gets downloaded to the client machine.

To export an entity, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.
2. Click the ellipsis icon and select **Export**. The Export Entity dialog appears. The Entity Class and Entity Members appear selected by default.
3. Select the **Audit Trail** checkbox, if you want to include the logs of the operations in the to-be imported .zip file.
4. Click **Export.** The entities get downloaded in a .zip file in the local memory of the client machine.

Related topic(s)

Importing an entity

Exporting entities in bulk

Rule Builder allows you to export entities in bulk and import entities as well. The bulk export feature allows the synchronization of data of two Entities on separate servers.

To export entities in bulk, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.

2. Click the ellipsis icon present next to the create entity group icon and select **Bulk Export**. The Bulk Export Entities dialog appears.
   The list of entities available within the cabinet appears.
3. Select the checkbox present against the entity name that you want to export.
   You can use the **Select All** option to select all the entities present at once. You can also use the search box to search the entities by name.
4. Click **Export**. The entities get downloaded in a ..zip file in the local memory of the client machine.

Related topic(s)

Importing an entity

Importing an entity

Before importing an entity on another server, search for the .zip file in the client machine. Create an entity with the same name as that of the uploaded .zip file.

To import an entity, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the required entity group.
2. Click the ellipsis icon and select **Import**. The Import Entity dialog appears.
3. Click **Browse** to select the entity from the client machine.
4. Select:
   - **Append Mode** — to append the new members in an already existing entity.
   - **Overwrite Mode** — to overwrite the existing member list of entities with the newly imported one.
5. Select the **Audit Trail** checkbox to include the audit history in the to-be imported .zip file.
6. Click **Import**. The entity gets imported within the selected entity group.

Related topic(s)

- Exporting an entity
- Exporting entities in bulk

Viewing entity group audit trail

The audit trail option enables you to view the history of operations performed on the entity group.

To view the audit trail of an entity group, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, hover over the name of the entity group for which you want to view the audit trail.
2. Click the ellipsis icon and select **Audit Trail**. The Audit Trail for the selected entity group appears.
   The audit trail displays the following details:

Field	Description
Action	This field displays the action performed on the entity group such as updation, deletion, creation, and more.
Action by	This field displays the name of the user who has performed action on the entity group.
Timestamp	This field displays the date and time when the action was performed on the entity group.
Summary	This field displays a short summary of the action performed on the entity group.

Use the information icon to view the details of the actions performed on the entity.

3. Click **Close** to exit the Audit Trail dialog.

Add and manage entity classes

An entity class is a class that helps in defining business processes or objects. For example, loan applications, account openings, and more.

After creating an entity group, you need to add classes within the entity group. Entity classes are used to group entities belonging to a business process, say, a loan application.
Select an entity to view the list of entities defined within it or to create new entities.

The list of entity classes associated with an entity appears next to the list of entity groups. The name of the entity class along with the number of member variables defined within it is displayed on the screen.

Use the search box to search the entity class by name.
Use the **Sort By** option to sort the list of entity classes using one of the following criteria:

- **Last Modified** — Sorts the list in descending order as per the recent.
- **Name: A-Z** — Sorts the list in alphabetically ascending order.
- **Name: Z-A** — Sorts the list in alphabetically descending order.

Related topic(s)

- Creating an entity class
- Editing an entity class
- Copying an entity class
- Deleting an entity class
- Viewing entity class audit trail

Creating an entity class

To create an entity class, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the entity group under which you want to create the entity class.
2. Click **Create Entity Class** button.
   If you have an existing list of entity classes within an entity group, then click create icon  . The Create Entity Class dialog appears.
3. Refer to the below table to specify details in the fields:

Field	Description
Alias Name	Specify the name of the entity class. It is a mandatory field.
Class Name	Based on the alias name, the system generates a non-editable class name. This name is essential for integration purposes.
Array	Select **Yes** to create an array-type entity class that can be used to add forEach column while defining rule flows. Else, select **No** to create a non-array type entity class.
Description	Enter a description of the entity class.

4. Click **Save**. The entity class gets created and added to the list under the selected entity group.

Related topic(s)

- Creating a member variable
- Adding forEach column in rule flow definition

Editing an entity class

To edit an entity class, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Hover over the name of the required entity class.
3. Click the ellipsis icon and select **Edit**. The Edit Entity Class dialog appears.
4. Make the required modifications.

   You cannot modify the class name.
5. Click **Save**. The changes made to the entity class get saved.

Copying an entity class

Using the copy entity class option, you can create a new entity class from a copy of an existing one.

To create a new entity class using the copy option or create a copy of existing entity class, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Hover over the name of the required entity class.
3. Click the ellipsis icon and select **Copy**. The Copy Entity Class dialog appears.
4. Specify **New Alias Name** for the entity class. Based on the alias, **Class Name** gets generated.
5. Under List of Variables, select the checkbox present against the member variables name that you want to add to the newly created copy.
   You can use the **Select All** checkbox to add all the member variables to the copy.
6. Click **Copy**. A new copy of the existing entity member gets created under the same parent entity group.
   Conflict(s) might appear due to use of Alias Name which is already in use. In this case, change the Alias Name and then click **Copy**.

Deleting an entity class

To delete an entity class, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Hover over the name of the required entity class.
3. Click the ellipsis icon and select **Delete**. The Confirm Deletion dialog appears.

   Deleting an entity class invalidates all rules that are using it. Invalidated rules must be modified to work further. If you delete some entity member or entity classes associated with a rule package or rule flow, you must remove the associated entity from the argument list.
4. Click **Delete**. The entity class gets deleted from the system.

Viewing entity class audit trail

The audit trail option enables you to view the history of operations performed on the entity class.

To view the audit trail of an entity class, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Hover over the name of the required entity class.
3. Click the ellipsis icon and select **Audit Trail**. The Audit Trail for the selected entity group appears.

   The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the entity class such as updation, deletion, creation, and more.
Action by	This field displays the name of the user who has performed an action on the entity class.
Timestamp	This field displays the date and time when the action was performed on the entity class.

Field	Description
Summary	This field displays a short summary of the action performed on the entity class.

Use the information icon to view the details of the actions performed on the entity class.

4. Click **Close** to exit the Audit Trail dialog.

Add and manage member variables

An entity member is the lead node of an entity that contains real data for an entity. A collection of all such members represents an entity. This entity can further be used to define rules.
After creating an entity class, you need to add member variables to the entity class.

The list of member variables associated with an entity class appears next to the list of entity classes. The name of the entity class along with the data type and input values is displayed on the screen.

Use the search box to search the member variable by name.
Use the **Sort By** option to sort the list of member variables using one of the following criteria:

- **Last Modified** — Sorts the list in descending order as per the recent.
- **Name: A-Z** — Sorts the list in alphabetically ascending order.
- **Name: Z-A** — Sorts the list in alphabetically descending order.

Use the **Filters** option to filter the list of member variables by data type or array type.

Related topic(s)

- Creating a member variable
- Importing member variables
- Editing a member variable
- Copying a member variable
- Deleting a member variable
- Viewing member variable audit trail

Creating a member variable

To create a member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class under which you want to add member variables.
3. Click **+ Create Variable**.
   If you have an existing list of member variables within an entity class, then click create **+ Member Variable** icon.
   The Create Member Variable dialog appears.
4. Refer to the below table to specify details in the fields:

Field	Description
Alias Name	Specify the name of the entity member variable. It is a mandatory field.
Variable Name	The user-defined variable name appears in this textbox as per the naming convention of the alias.
Report Display Name	Specify a business or user-friendly name for the variable that appears in the doc-type reports downloaded for the rule packages containing this member variable.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Data Type</td><td>

Select the data type as: BigDecimal, Boolean, Date, Float, Integer, Long, or String. It is a mandatory field.
If the selected data type is Date, Integer, Float, Long, or BigDecimal, then you must specify Minimum and Maximum Value.

This Min Value and Max Value is used when you define a rule using this entity member. While defining the rule you are not allowed to enter any value below the Min Value and above the Max Value.

If the selected variable is of BigDecimal type then two additional fields, Rounding Mode and Precision appear. Select:

Rounding Mode — Round off unwanted digits to limit the size of the number in cases when the exact result has mutiple digits.

The following rounding modes are available to select:
<table>
<tr><th>Rounding Mode</th><th>Description</th></tr>
<tr><td>Ceiling</td><td>Rounding mode to round towards positive infinity. Note that this rounding mode never decreases the calculated value.
Example: If the input number is 1.6 then with Ceiling rounding mode the input gets rounded off to one digit, that is, 2. If the input number is 6.2 then it gets rounded off to one digit, that is, 7.</td></tr>
<tr><td>Down</td><td>Rounding mode to round towards zero. Never increments the digit prior to a discarded fraction (that is, truncates). Note that this rounding mode never increases the magnitude of the calculated value.
Example: If the input number is 1.6 then with Down rounding mode the input gets rounded off to one digit, that is, 1. If the input number is -2.5 then it gets rounded off to -2.</td></tr>
</table>
</td></tr>
</table>

Field	Description	
	Floor	Rounding mode to round towards negative infinity. Note that this rounding mode never increases the calculated value. Example: If the input number is 1.6 then with Floor rounding mode the input gets rounded off to one digit, that is, 1. If the input number is -2.5 then it gets rounded off to -3.
	Half Down	Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant (in this case round down). Behaves as Up rounding mode if the discarded fraction is > 0.5; otherwise, behaves as Down rounding mode. Example: If the input number is 1.6 then with Half Down rounding mode the input gets rounded off to one digit, that is, 2. If the input number is -2.5 then it gets rounded off to -2. If the input number is 7.1 then it gets rounded off to 7.
	Half Up	Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant (in this case round up). Behaves as Up rounding mode if the discarded fraction is >= 0.5; otherwise, behaves as Down rounding mode. Example: If the input number is 7.8 then with Half Up rounding mode the input gets rounded off to one digit, that is, 9. If the input number is -4.5 then it gets rounded off to -5. If the input number is 7.2. then it gets rounded off to 7.
	Half Even	Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant (in this case round towards the even neighbor). Behaves as Half Up rounding mode if the digit to the left of the discarded fraction is odd; behaves as Half Down rounding mode if it's even. Example: If the input number is 7.8 then with Half Even rounding mode the input gets rounded off to one digit, that is, 8. If the input number is -4.5 then it gets rounded off to -4. If the input number is -1.1. then it gets rounded off to -1.

Field	Description	
	Up	Rounding mode to round away from zero. Always increments the digit prior to a non-zero discarded fraction. Note that this rounding mode never decreases the magnitude of the calculated value. Example: If the input number is 1.6 then with Up rounding mode the input gets rounded off to one digit, that is, 2. If the input number is -2.5 then it gets rounded off to -3.
	Unnecessary	Rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary. If this rounding mode is specified on an operation that yields an inexact result, an Arithmetic Exception is thrown. Example: If the input number is 7.8 then with Unnecessary rounding mode, an arithmetic expression gets thrown. If the input number is -4.5, an arithmetic expression gets thrown. If the input number is -1.1. then it gets rounded off to -1.
	• **Precision** — Specify a precision value for BigDecimal variable. The value of precision may lie between 0 and 99.	
Array	Select Yes to create an array-type member variable. Else, select No. It is a mandatory field. Creation of array type member variable is not allowed for the Boolean data type.	
Default, Maximum, and Minimum value	Specify the values to manage validations for rule definition and execution. In case the end user does not specify a value for the variable, the system automatically picks the default value. • Depending on the data type you selected, the maximum and minimum values fields appear. Specify the values accordingly. These options are essential to set a range of values allowed in the fields.	
Description	Enter a description of the member variable.	

5. Click:
   - **Add & Exit** — to add the declared member variable and exit the dialog.
   - **Add Another** — to add the declared member variable and clear all the fields in order to declare another variable.
     The newly created member variable gets created.

Related topic(s)

- Associating arguments with a rule package
- Creating and defining a rule
- Associating arguments with a rule flow
- Creating a rule flow

Importing member variables

To import a member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class under which you want to import member variables.
3. Click **+ Import Variable**.
   If you have an existing list of member variables within an entity class, then click create **+ Import** icon.
   The Import Member Variable dialog appears.
4. Select **Import from** as :
   - Process — to import variable from a process defined under NewgenONE Process Designer.
   - Data Model Categories — to import variables from a defined data object under NewgenONE Data Model Designer.
5. Click **Import**. The variable gets imported and added to the list under selected entity class.
   - Import Member Variable Conflict(s) dialog appears when you try to import variables that display conflicts due to the use of a similar Alias Name, this indicates that variables with same alias name are already in use by Rule Packages, Rule Flows, and more.

- In this case, check its usability using the **View usage details** option. Specify an **Alternate Alias Name** and click **Check Availability** to ensure the name is available and not in use already.
- Then, click **Import**. The variables get imported with a new alias name and added to the list.

- Alternatively, use the **Import Non-Conflicting Variables** option to discard the variables that have conflict and import which do not have any conflict.

Importing from Process

To import variables from a process, follow the below steps:

1. Select the state of the process as:
   - **Draft** — to select a process that is in draft state, that is, not deployed yet.
   - **Deployed** — to select a process that is already being deployed.
2. Select the **Process Name** using the dropdown under the selected state of the process. A list of available data objects defined under the process appears.
3. Select the required data object. You can use the search option to search the data object by name. A list of variables added within the data object appears.
4. Select the checkbox present against the variable name to import it. The added variables appear under the selected list along with the number of available slots.

Importing from Data Object

To import variables from a data object, follow the below steps:

1. Select the **Category Name** using the dropdown. A list of available data objects defined under the category appears.
2. Select the required data object. You can use the search option to search the data object by name. A list of variables added within the data object appears.
3. Select the checkbox present against the variable name to import it. The added variables appear under the selected list along with the number of available slots.

Editing a member variable

To edit a member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class. The list of member variables listed within it appears.
3. Hover over the name of the required member variable and click the edit icon  . The Edit Member Variable dialog appears.
4. Make the required modifications.

   - You cannot modify the variable name.
   - While editing a date-type entity member variable, the array checkbox remains disabled.

5. Click **Save**. The changes made to the member variable get saved.

Copying a member variable

Using the copy member variable option, you can create a new variable from a copy of an existing one.

To create a new variable using the copy option or create a copy of an existing member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class. The list of member variables listed within it appears.
3. Hover over the name of the required member variable.
4. Click the ellipsis icon  and select **Copy**. The Copy Member Variable dialog appears.
5. Specify **New Alias** for the member variable. Based on the alias, **New Name** gets generated.

6. Select **Destination** from the dropdown list. This Destination dropdown list contains all the Entity Classes defined under the parent Entity Group.
7. Click **Copy**. A new copy of the existing member variable gets created under the destination entity class.

Deleting a member variable

To delete a member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class. The list of member variables listed within it appears.
3. Hover over the name of the required member variable.
4. Click the ellipsis icon and select **Delete**. The Confirm Deletion dialog appears.

   Deleting an entity member variable invalidates all rules that are using it. Invalidated rules must be modified to work further. If you delete some entity members associated with a rule package or rule flow, you must remove the associated entity from the argument list.
5. Click **Delete**. The entity member variable gets deleted. to confirm deletion.

Viewing member variable audit trail

The audit trail option enables you to view the history of operations performed on the member variable.

To view the audit trail of a member variable, follow the below steps:

1. In the **Building Blocks** > **Entities** tab, select the required entity group. The list of entity classes listed within it appears.
2. Select the required entity class. The list of member variables listed within it appears.
3. Hover over the name of the required member variable.

4. Click the ellipsis icon and select **Audit Trail**. The Audit Trail for the selected entity group appears.
   The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the member variable such as updation, deletion, creation, and more.
Action by	This field displays the name of the user who has performed an action on the member variable.
Timestamp	This field displays the date and time when the action was performed on the member variable.
Summary	This field displays a short summary of the action performed on the member variable.

Use the information icon to view the details of the actions performed on the entity member variable.

4. Click **Close** to exit the Audit Trail dialog.

Adding and managing picklists

A picklist is a set of options from which a user can select an item. With Rule Builder, you can create a picklist and then associate it with a rule package or a rule flow. The values listed in the picklist provide input values against the variables defined in the rule or rule flow definition.

For example, you can create a picklist namely Products containing different product categories of a store inventory. Then, associate the picklist with a rule package AvailableStocks to get a list of available products.

The following are the different types of picklists supported in Rule Builder. Static and dynamic picklists are used in rules definition, while the runtime and runtime array picklists are used in rule execution –

- **Static picklist** — It is a collection of pre-defined values that do not change frequently based on user input and external factors. For example, a list of countries.

- **Dynamic picklist** — It is a collection of dynamic values that change frequently based on user input and external factors. In this type of picklist, the values are fetched directly from the database. For example, you can get a list of country codes stored in the database.
- **Runtime picklist** — It loads the option while executing the query at the time of rule execution. This type of picklist fetches the latest data value from the database based on the defined database query. For example, use a runtime picklist to get real-time inventory information about the available and unavailable products.
- **Runtime array picklist** — It loads the options while executing the query at the time of rule execution. Use this type of picklist to fetch one or more data values in the form of a list from the database based on the defined database query. For example, use a runtime picklist to check and update the inventory levels of products in real time.

The picklists you create are available at the cabinet level. With Rule Builder, you can create picklists from scratch or seamlessly import them.

This section covers the following topics:

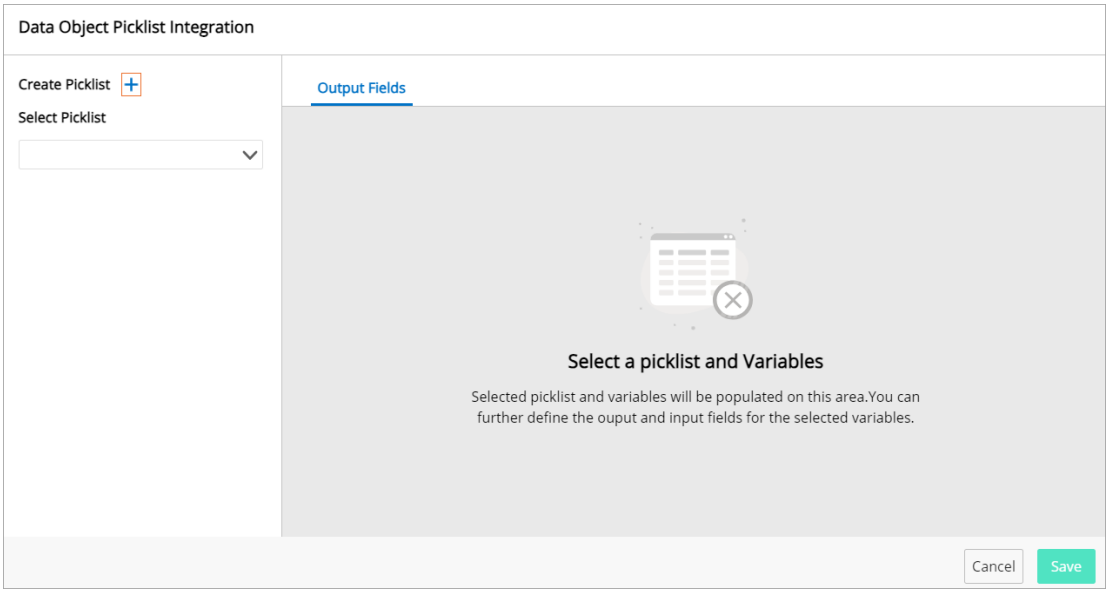
- Creating a static picklist from scratch
- Creating a dynamic picklist from scratch
- Creating a runtime picklist
- Creating a runtime array picklist
- Importing a picklist
- Associating output fields with picklist variables
- Applying a filter on picklist variable
- Managing picklists

Creating a static picklist from scratch

This topic explains how to create a new static picklist from scratch. Creating from scratch means you create a picklist without importing the picklist from an external source. It involves defining the data type of the picklist and adding picklist data.

To create a static picklist, follow the below steps:

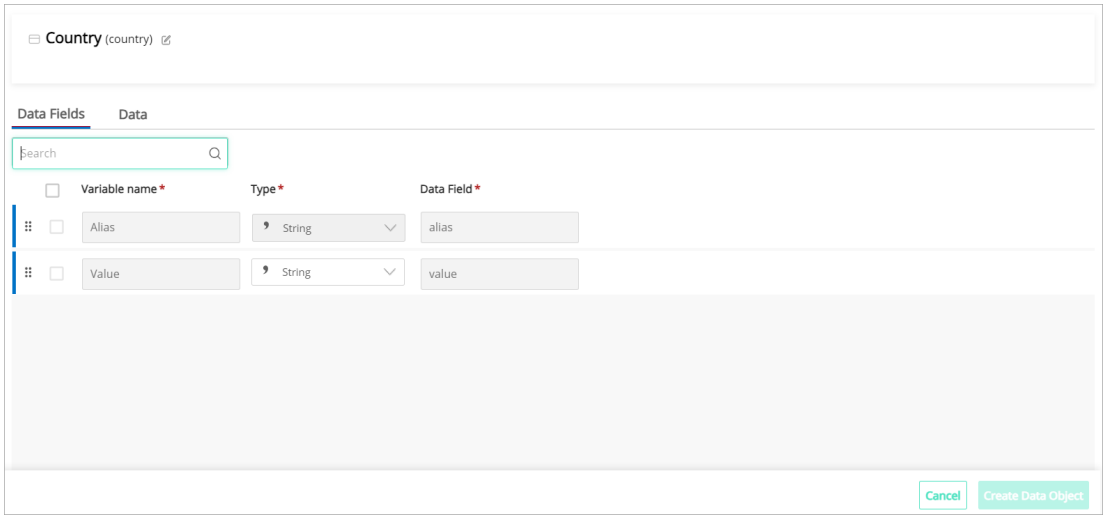
1. Navigate to the **Building Blocks** tab.
2. From the left pane, select **Picklist**. A list of created picklists appears.
3. Click **Create New Picklist** + to add a new one, then select **Static picklist** from the dropdown list. The Data Object Picklist Integration dialog appears.



4. Click **Add Picklist** +. The Create Picklist Data Object dialog appears.
5. Enter the following information in the dialog:

Field	Description
Name	Enter the name of the picklist.
Object ID	It is a system-generated object ID that uniquely identifies the picklist in the database. You can also modify the ID at the time of object creation.
Description	Enter the description of the picklist if any.
Define Alias	Select this checkbox to add an alias for the picklist variable.

6. Click **Create**. A new dialog appears.



Here, you can perform the following operations:

- Defining picklist data type
- Adding picklist data

Defining picklist data type

This topic explains how to define the data type of the picklist. By default, a static picklist contains only two variables of string type – alias and value. These variables store the picklist values.

No alias appears if you disabled the Define Alias option during the picklist creation process. Also, you cannot modify the type of alias. For more information on the creation of a picklist, see Creating a static picklist from scratch.

In the Data Fields tab, you can define the type of data you want to store in the picklist. It can be integer, string, long, float, short date, currency, Boolean, phone number, and email. For example, the CountryName variable of string data type stores string information like the United States, India, Canada, and others.

- Currency data type allows the storage of decimal values.
- Email data type allows the storage of string information.
- Phone number data type allows the storage of integer values.
- Short date data type allows the storage of date values.

Adding picklist data

After creating the picklist and defining its data type, you can add the data to display in the picklist. Picklist data means the values that appear in a dropdown list. This topic explains how to add data manually in the picklist.

For example, a Country picklist contains the names of different countries. This picklist data can further be used to define a rule for blacklisted countries.

To add picklist data, follow the below steps:

1. Go to **Data** tab.
2. Click **New Data** to add picklist data. The Add Data dialog appears.
3. Enter the picklist data next to the alias and value.

   Alias only appears when you enable the **Define Alias** option during the picklist creation process. For more information on the creation of a picklist, see Creating a static picklist from scratch.
4. Click **Add & Exit** to add the data and close the dialog. To add more picklist data, click **Add & Continue**.
   The picklist data gets added. You can then perform the following operations on the added picklist data:
   - Click the corresponding edit icon to modify the added picklist data.
   - Click the corresponding delete icon to remove the added picklist data.
5. Click **Save**. The picklist data gets saved.

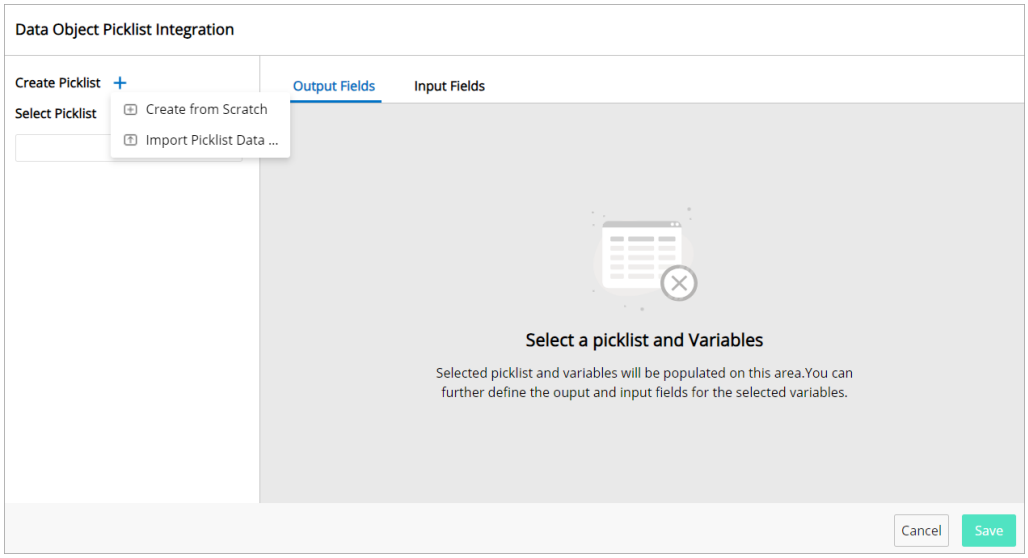
Creating a dynamic picklist from scratch

This topic explains how to create a new dynamic picklist from scratch. Creating from scratch means you create a picklist by importing a picklist from external sources. It involves defining the data type, key field constraint, and data mapping for the picklist.

To create a dynamic picklist from scratch, perform the following steps:

1. Navigate to the **Building Blocks** tab.
2. From the left pane, select **Picklist**. A list of created picklists appears.
3. Click **Create New Picklist** + to add a new one, then select **Dynamic picklist** from the dropdown list. The Data Object Picklist Integration dialog appears.

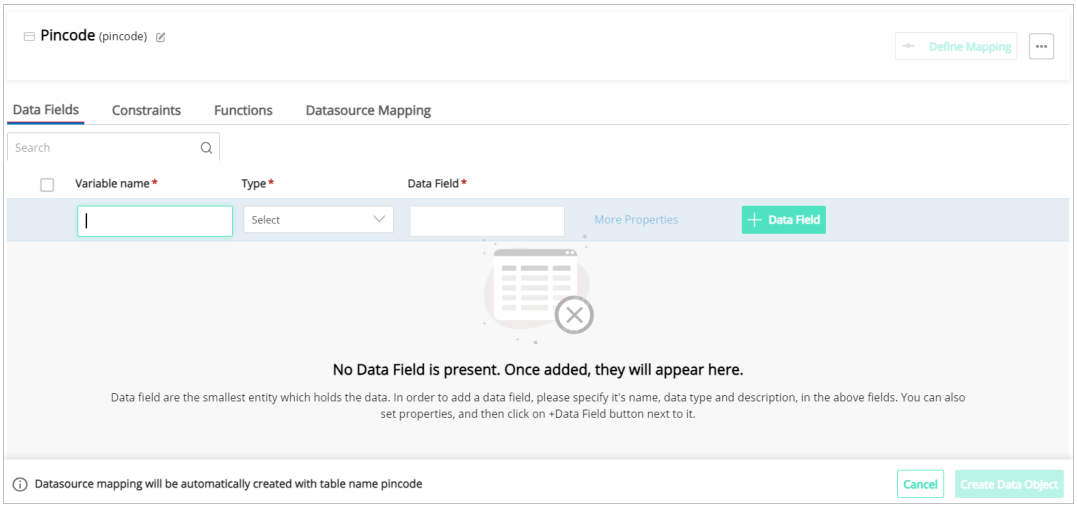
4. Click **Add Picklist** +, and then select **Create from scratch** from the dropdown list. The Create Picklist Data Object dialog appears.



5. Enter the following information in the dialog:

Field	Description
Name	Enter the name of the picklist.
Object ID	It is a system-generated object ID that uniquely identifies the picklist in the database. You can also modify the ID at the time of object creation.
Description	Enter the description of the picklist if any.
Define Alias	Select this checkbox to add an alias for the picklist variable.

6. Click **Create**. A new dialog appears.



Here, you can perform the following operations:

- Defining picklist data type
- Defining constraints
- Mapping picklist data

You can define the mapping for the picklist using the NewgenONE Data Model Designer. For more information, refer to the **Creating and managing picklist data objects** chapter in *NewgenONE Data Model Designer*.

Defining picklist data type

This topic explains how to define the data type of the picklist. A picklist can contain multiple variables to store the picklist values and aliases.

No alias appears if you disabled the Define Alias option during the picklist creation process. Also, you cannot modify the type of alias. For more information on the creation of a picklist, see Creating a dynamic picklist from scratch.

In the Data Fields tab, you can define the type of data you want to store in the picklist. It can be integer, string, long, float, short date, currency, Boolean, phone number, and email. For example, the CountryName variable of string data type stores string information like the United States, India, Canada, and others.

- Currency data type allows the storage of decimal values.
- Email data type allows the storage of string information.
- Phone number data type allows the storage of integer values.
- Short date data type allows the storage of date values.

To define the data type of the picklist, follow the below steps:

1. Go to the **Data Fields** tab.
2. Enter the name of the variable under the Variable name column.
3. Select the data type of the variable. It can be integer, string, long, float, short date, currency, Boolean, phone number, and email.
4. Enter the data field name of the variable. By default, it is prefilled with the variable name value.
5. Click **Data Field**. The variable gets added. You can perform the following operations on the added variable:
    - To remove the added variable, click the **More Options** icon ••• and then select **Delete**.
    - To remove multiple variables, select the checkbox next to the Variable name column and then click **Delete**.
    - To change the sequence of added variables, drag and drop the required variable to the desired location.

    Additionally, you can add more variables by clicking **Data Field**.

Defining constraints

You can define constraints to add rules to the picklist variables. This helps to ensure the reliability and integrity of the data. By applying constraints, you can determine what type of data must be stored within the picklist variable.

These are the different types of constraints you can apply on a picklist variable:

> Currently, Rule Builder supports only key field constraints on picklist variables.

- **Key field** — A key field constraint is applied to a variable or a group of variables that contains only unique values. A key field cannot have any null value. For example, you can set the employee ID variable as the key field. This is because employee ID is unique for each employee within an organization.

  > It is necessary to define a key field constraint when creating a picklist from scratch.

- **Not Null** — A not null constraint is applied to prevent null values within a picklist variable. For example, when entering the delivery details, it is necessary to provide the location, or else the information remains incomplete.
- **Index** — An index constraint is applied to speed up the search performance. For example, fields such as employee name and email ID in an IT complaint form.

- **Unique Key** — A unique key is applied to a picklist variable that contains unique values. For example, the email address field in an event registration form.

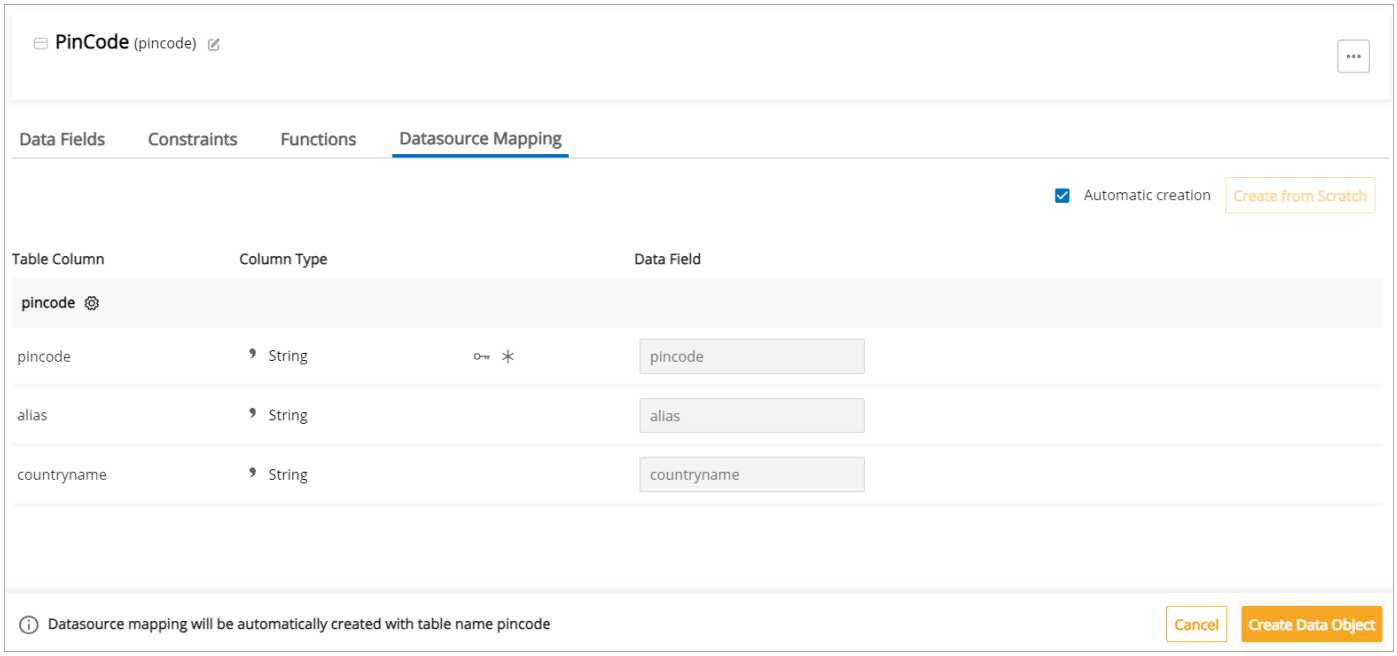
To apply a constraint on the picklist variable, follow the below steps:

1. Go to the **Constraints** tab.
2. Click the key field constraint on the upper-right section of the tab. A dialog box appears displaying a list of picklist variables.
3. From the list, select the required picklist variable. You can also use the search bar to find a particular variable or apply filters to view the picklist variables of specific data types.
4. Click **Create**. The newly created constraint now appears in the constraints list along with its constraint ID. To view constraint details including the variable name, data type, and data field, click **Expand All**.
   Additionally, you can remove the added constraint by clicking its corresponding delete icon .

Mapping picklist data

Mapping picklist data is a process of linking the variables of the current picklist with those in another physical table existing in the signed-in cabinet. This tab displays the basic details about the picklist variables with their data types.

By default, the **Automatic Creation** checkbox is selected. It then displays the variables with their defined data types and their physical location in the database that exists within the currently signed-in cabinet.



However, you can map the picklist variables with the required data objects available in the data sources existing in the current signed-in cabinet by clearing the **Automatic Creation** checkbox and then clicking the **Create from Scratch** button. The Create-Datasource mapping dialog appears containing three tabs — Basic Details, Select Data Field, and Mapping.

Mapping picklist with NewgenONE cabinet data object

To map the picklist with the NewgenONE cabinet data object, follow the below steps:

1. In the Basic Details tab of the Create – Datasource mapping dialog, In the Basic Details tab of the Create – Datasource mapping dialog, set the data source type you want to use for picklist mapping — **NewgenONE Cabinet**.
   In this type of source, you can use the data objects available in the currently signed-in cabinet. A cabinet can contain multiple data objects.
2. Click **Next** to proceed further. You are then navigated to the Select Data Field tab of the dialog that displays the list of available data objects in the cabinet.
   Additionally, you can use the search box to find the picklist by name.
3. Drag and drop the required tables to the empty canvas of the dialog. You can also use the search box to find specific columns in the table or click the delete icon to remove the table.
   By default, all the table columns appear selected.
4. Clear the columns in the table you do not require.

5. Click **Next** to proceed. You are then navigated to the Join Data Objects tab of the dialog. Here, you can join the tables using an SQL expression.

   This tab is only available in case you drop multiple tables.

6. Enter the SQL expression to join the tables. You can also click the Keywords, Operators, and Fields to add them to the SQL expression.
7. Click **Next** to proceed further. You are then navigated to the Mapping tab of the dialog. Here, you can map the columns of the chosen table(s) with the variables present in the current picklist. This tab displays the available column names in the table along with their data types.
   A **Mark as Primary** checkbox appears on the upper-right corner of the page to mark the table as the primary source. This means the table acts as the primary source of data from where the data flows to the secondary tables during data processing. This checkbox is selected by default and is uneditable.
8. Select the data field from the dropdown list next to the required table column.
9. Repeat the above step to map other variables of the picklist.
10. Click **Save**. The mapping gets saved.
11. To change the data source mapping, click **Modify Mapping**.
12. Click **Create Picklist Data Object**. The data object is now created and appears in the picklist list on the left.

Creating a runtime picklist from scratch

This topic explains how to create a new runtime picklist from scratch. Creating from scratch means you create a picklist without importing it from an external source. It involves defining the data type, key field constraint, and data mapping for the picklist.

To create a runtime picklist from scratch, perform the following steps:

1. Navigate to the Building Blocks page.
2. From the left pane, select **Picklist**. A list of created picklist appears.
3. Click **Create New Picklist** + to add a new one, then select **Runtime picklist** from the dropdown list. The Data Object Picklist Integration dialog appears.
4. Click **Add Picklist** +, and then select **Create from scratch** from the dropdown list. The Create Picklist Data Object dialog appears.

5. Enter the following information in the dialog:

Field	Description
Name	Enter the name of the picklist.
Object ID	It is a system-generated object ID that uniquely identifies the picklist in the database. You can also modify the ID at the time of object creation.
Description	Enter the description of the picklist if any.

6. Click **Create**. A new dialog appears. Here, you can perform the following operations:
    - Defining picklist data type
    - Defining constraints
    - Mapping picklist data

> You can define the mapping for the picklist using the NewgenONE Data Model Designer. For more information, refer to the **Creating and managing picklist data objects** chapter in *NewgenONE Data Model Designer*.

Creating a runtime array picklist from scratch

This topic explains how to create a new runtime array picklist from scratch. Creating from scratch means you create a picklist without importing it from an external source. It involves defining the data type, key field constraint, and data mapping for the picklist.

To create a runtime array picklist from scratch, perform the following steps:

1. Navigate to the Building Blocks page.
2. From the left pane, select **Picklist**. A list of created picklist appears.
3. Click **Create New Picklist** + to add a new one, then select **Runtime array picklist** from the dropdown list. The Data Object Picklist Integration dialog appears.
4. Click **Add Picklist** +, and then select **Create from scratch** from the dropdown list. The Create Picklist Data Object dialog appears.
5. Enter the following information in the dialog:

Field	Description
Name	Enter the name of the picklist.
Object ID	It is a system-generated object ID that uniquely identifies the picklist in the database. You can also modify the ID at the time of object creation.
Description	Enter the description of the picklist if any.
Define Alias	Select this checkbox to add an alias for the picklist variable.

6. Click **Create**. A new dialog appears. Here, you can perform the following operations:
    - Defining picklist data type
    - Defining constraints
    - Mapping picklist data

> You can define the mapping for the picklist using the NewgenONE Data Model Designer. For more information, refer to the **Creating and managing picklist data objects** chapter in *NewgenONE Data Model Designer*.

Importing a picklist

NewgenONE Rule Builder enables you to create new picklist by importing them in JSON format from your machine or a database.

This section covers the following topics:

- Importing using a JSON file
- Importing using a database

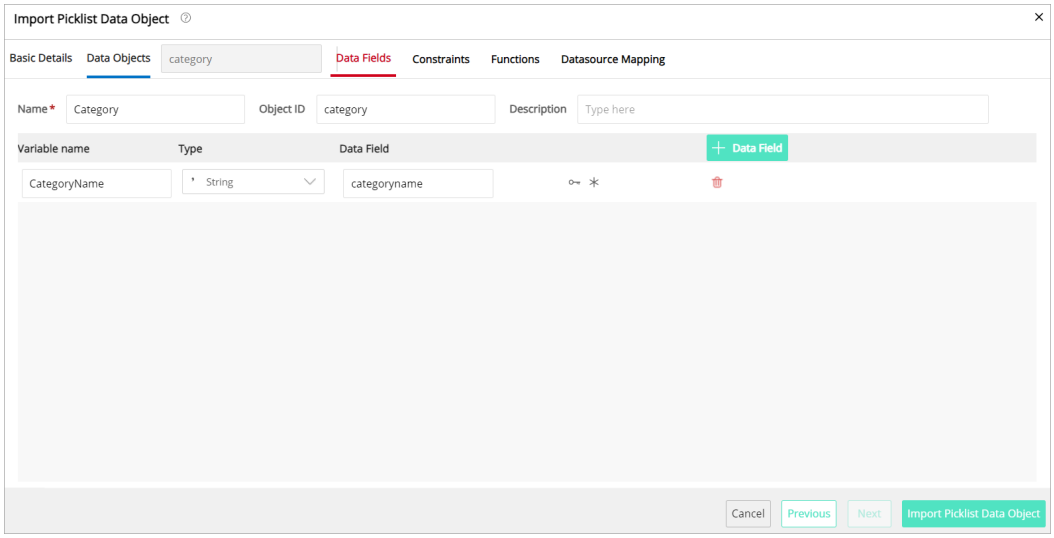
Importing using a JSON file

To import a picklist using a JSON file, follow the below steps:

1. Open the Data Object Picklist Integration dialog.
2. Click **Add Picklist** + and then select **Import Data Object** from the dropdown list. The Import Picklist Data Object dialog appears containing two tabs – Basic Details and Data Objects.
3. In the Basic Details tab, fill in the following information in the respective fields:

Field	Description
Import from	Set the importing method to File. By default, the File option is already selected.
Browse file	Click the **Browse File** button to select the required JSON file from your local machine and then click **Open**. You can also import multiple picklist from a JSON file containing multiple object definitions.
Data objects	It displays the name of the picklist in the uploaded JSON file.
Define Alias	Select this checkbox to add an alias for the picklist variable.

4. Click **Next** to proceed further.
   In case a picklist of the same name is already present in the system, an alert message appears prompting you to provide a new name and ID for the picklist. Click **Next**.
   The Data Object tab appears that contains four subtabs such as Data Fields, Constraints, Data Object Relations, and Data Source Mapping. The textbox next to the Data Objects tab shows the name of the picklist you are importing.
   However, in the case of multiple picklists, you can select the required one from the dropdown list.



5. You can perform the following operations in the Data Objects tab:
   - **Data Fields** — This tab displays the picklist variables with their data types. Here, you can modify the name, object ID, and description of the picklist. You can also add new picklist variables or modify the existing ones by

changing the variable name, data type, or data field. For more information, see Defining picklist data type.

- **Constraint** — This tab displays the constraints applied to the picklist variables. Here, you can mark the required variable as the key field. For more information, see Defining constraints.
- **Mapping picklist data** — This tab displays the mapping details of the table columns and data fields with their data types. Here, you can modify the existing mapping of the picklist. For more information, see Mapping picklist data.

6. Click **Import Picklist Data Object**. The picklist gets imported.

Importing using a database

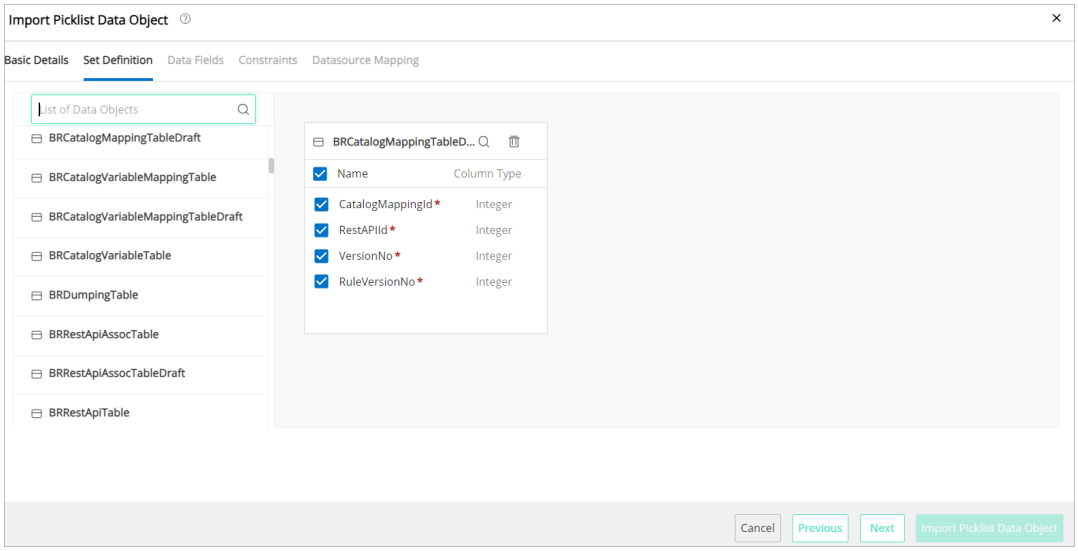
To import a picklist using a database, follow the below steps:

1. Open the Data Object Picklist Integration dialog.
2. Click **Add Picklist** + and then select **Import Data Object** from the dropdown list. The Import Picklist Data Object dialog appears containing two tabs — Basic Details and Data Objects.
3. In the Basic Details tab, enter the following information in the respective fields:

Field	Description
Import from	Set the importing method to Database.
Datasource	Specify the type of data source you want to use. You can set the data source type to **NewgenONE Cabinet**. With this type of data source, you can use the data objects available in the currently signed-in NewgenONE cabinet
Name	Enter the name of the data object.
Description	Enter the description of the picklist, if any.
Object ID	It is a system-generated object ID that uniquely identifies the picklist in the database. You can also modify the ID at the time of object creation.
Define Alias	Select this checkbox to add an alias for the picklist variable.

4. Click **Next** to proceed further. The Set Definition tab appears.

5. Drag and drop the required tables to the empty canvas of the dialog. You can also use the search box to find specific columns in the table or click the delete icon to remove the table.
   By default, all the table columns appear selected.



6. Clear the columns in the table you do not require.
7. Click **Next** to proceed. The Join Data Objects tab appears. Here, you can join the tables using an SQL expression.

   This tab is only available in case you drop multiple tables.

8. Enter the SQL expression to join the tables. You can also click the Keywords, Operators, and Fields to add them to the SQL expression.
9. Click **Next** to proceed further. The Data Fields tab appears. This tab displays the variable name with its corresponding type and data field.
   Here, you can modify the definition of the variable such as its name, type, or data field name. Additionally, you can add new fields or remove the existing ones. For more information, see Defining picklist data type.
10. Click **Next**. The Constraints tab appears. Here, you can view the key constraints applied to the picklist variables. For more information, see Defining constraints.
11. Click **Next**. The Datasource Mapping tab appears. It displays the mapping details of the picklist variables. For more information, see Mapping picklist data.
12. Click **Import Picklist Data Object**. The picklist gets imported.

Associating output fields with picklist variables

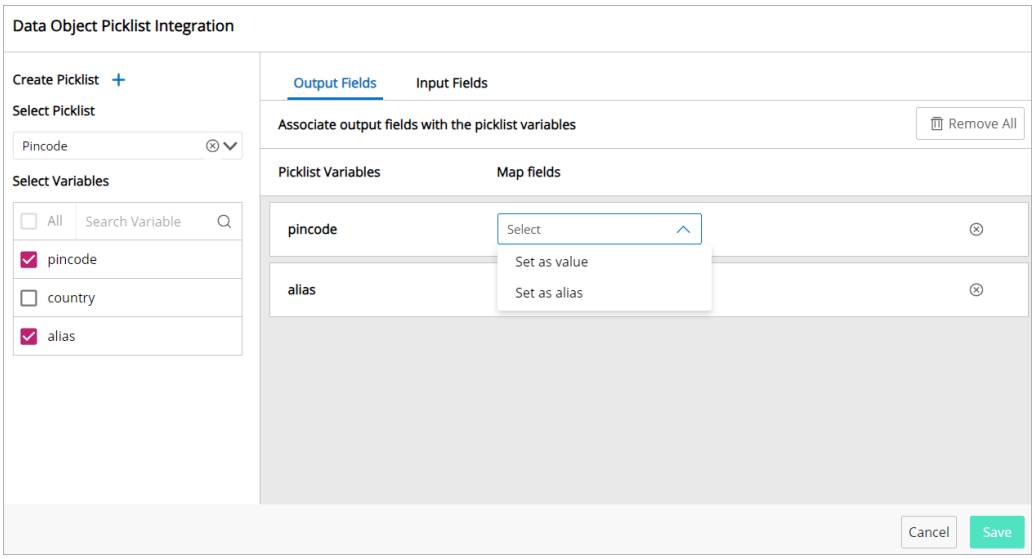
After creating or importing a picklist, you can associate the output fields with the picklist variables. Here, output fields represent the picklist variable type — alias or value.

To associate the output fields with picklist variables, perform the following steps:

1. In the Data Object Picklist Integration dialog, choose the required picklist from the **Select Picklist** dropdown. A list of picklist variables appears under the Select Variables section.

   The list of picklists varies depending on your picklist selection type. For example, in case you are performing variable mapping for dynamic picklist. Then, only dynamic picklists appear in the list. This list also contains picklists created using Data Model Designer and Interface Designer.

2. From the list, select the required picklist variable you want to associate with the output field. However, if you enabled the **Define Alias** option at the time of picklist creation process, you can select up to two picklist variables from the list.
   You can also use the search box to search a picklist variable by its name.
   The chosen picklist variables appear on the right of the dialog.
3. Select the required output field you want to associate with the picklist variable from the **Map Field** dropdown list. It can either be an alias or a value.



The Set as Alias option appears only if you selected the **Define Alias** option at the time of picklist creation process.

4. (Optional) To add conditions on the picklist variables, go to the **Input Fields** tab. For more information, see Applying a filter on picklist variable.
5. Click **Save**.

Related topic(s)

Applying a filter on picklist variable

Applying a filter on picklist variable

After associating the output fields with the picklist variables, Rule Builder allows you to apply filter conditions on the picklist variables. Filters help to display only relevant options in the picklist. For example, in the case of a Language picklist, you can apply a filter condition to limit the options to display based on the user's residential area.

You cannot apply filters on variables in the case of static picklists.

For more information on picklist variable association, see Associating output fields with picklist variables.

To define a filter condition for a picklist variable, follow the below steps:

1. In the Data Object Picklist Integration dialog, navigate to the **Input fields** tab. Here, you can define the filter conditions to apply to the picklist variables.
2. Click **Add Condition**.
3. Select the **(** from the dropdown list.
4. From the **Field** dropdown, select the required data field. It lists all the data fields created for the picklist.
5. Select the operator type for the condition. It can be any of the following =, >, <, <=, >=, <>, and LIKE.

   The operator types vary depending on the picklist variable data type.
6. From the **Value** field, select the comparison value. This comparison value can be a static or runtime value.
   - To use a static value, enter directly in the space provided.
   - To use a dynamic value, select the placeholder from the dropdown list.

   Dynamic picklists support only static values.

7. Select the **)** from the dropdown list.
8. (Optional) From the **AND/OR** dropdown, select the required conditional operator. This is used in case you are adding multiple filter conditions. Then, click **Add Condition** and repeat the above steps to create another filter condition.
   To remove the added condition, click the corresponding delete icon ⊗.
9. Click **Save**.

Related topic(s)

Associating output fields with picklist variables

Managing picklists

After creating a picklist, it appears on the left pane. Select the picklist from the list to view its related information. It includes the picklist name, type, list value, alias, description, array, and database query. The database query is generated based on your data selection during the picklist creation process.

> The database query does not appear for static picklists.

Here, you can perform the following operations:

- Editing a picklist
- Copying a picklist
- Viewing audit trail
- Setting default sorting
- Modifying a picklist
- Importing a picklist
- Exporting a picklist
- Deleting a picklist

Editing a picklist

To edit a picklist, follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, select the picklist that you want to modify. The definition of the selected picklist appears.
2. Click **Edit**. The Data Object Picklist Integration dialog appears. Here, you can modify the mapping of the picklist variables and the applied filter conditions. Additionally, in the case of runtime and runtime array picklists, an **Array** toggle appears.
    - In the case of a runtime picklist, the **Array** toggle appears disabled. You can enable the toggle to make the picklist a runtime array picklist.
    - In the case of a runtime array picklist, the **Array** toggle appears disabled. You can enable the toggle to make the picklist a runtime picklist.
3. Click **Save** to confirm. The picklist definition gets updated.

Copying a picklist

Duplicate a picklist to create a new data object using an existing one.

To duplicate a picklist, follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, select the picklist that you want to copy. The definition of the selected picklist appears.
2. Click the ellipsis icon and select **Copy**. The Copy Picklist dialog appears.
3. Specify a **New Name** for the copied picklist.
4. Click **Copy**. A copy of the existing picklist gets created with a new name and added to the list. This copy consists of all the attributes of original picklist.

Viewing audit trail

The audit trail feature in NewgenONE Rule Builder allows you to view the history of operations performed on the picklist.

To view the audit trail of a picklist, follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, select the picklist for which you want to view audit history. The definition of the selected picklist appears.
2. Click the ellipsis icon and select **Audit Trail**. The Audit Trail dialog specific to the selected picklist appears.
   The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the picklist such as updation, deletion, creation, and more.
Action by	This field displays the name of the user who has performed the action on the picklist.
Timestamp	This field displays the date and time when the action was performed on the picklist.
Summary	This field displays a short summary of the action performed on the picklist.

   Use the information icon to view the details of the actions performed on the entity.
3. Click **Close** to exit the Audit Trail dialog.

Setting default sorting

Rule Builder allows you to specify the sorting order for the variable at the time of picklist creation. This helps to arrange the data in a meaningful order. For example, you can arrange the country names in ascending order to analyze the data effectively.

To set the the default sorting, follow the below steps:

1. On the upper-right section of the dialog, click **More Options** ••• and then select **Default Sorting**. The Default Sorting dialog appears.
2. Select the required picklist variable based on which you want to apply the sorting.
3. Select the required sorting arrow to specify the default sorting.
    - Click the up arrow ↑ to sort the data in ascending order.
    - Click the down arrow ↓ to sort the data in descending order.
4. Click **Save**.

Modifying a picklist

Modify a picklist to change its name or description. You can modify a picklist only during its creation process.

To modify a picklist, follow the below steps:

1. When creating the picklist, click **Edit** next to the picklist name in the dialog. The Modify Picklist Data Object dialog appears.
2. Change the name, description, or both for the picklist.

    Fields such as Object ID and Picklist Type are non-modifiable.
3. Click **Modify**. The picklist is now modified.

Deleting a picklist

To delete a picklist, follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, select the picklist that you want to delete. The definition of the selected picklist appears.
2. Click the ellipses icon and then select **Delete**. The Confirmation Deletion dialog appears.
3. Click **Delete** to confirm.

Once you delete a picklist, the rules using this picklist become invalid.

Importing a picklist

To import a picklist, follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, click the ellipsis icon in the picklist pane. The Import Picklist(s) dialog appears.
2. Click **Browse** to select .zip file to import picklist.
3. Select the picklist .zip file from your local machine and click **Open**.
4. Select the **Mode of Import** as:
   - **Append Mode** — to add the imported picklists to the list of existing picklists.
   - **Overwrite Mode** — to overwrite the existing picklist(s) with the newly imported picklist(s).

   If the Imported .zip file contains more than one picklist, then the list of picklist(s) available to import appears under the **Select Picklist(s) to Import** section. Select the checkbox present against the picklist name that you want to upload.
5. Click **Import**. The picklist gets imported and added to the list.

Related topic(s)

Exporting picklists

Exporting a picklist

The export picklist feature allows synchronization of the data of the picklists on separate servers.

To export picklist(s), follow the below steps:

1. In the **Building Blocks** > **Picklist** tab, click the ellipsis icon in the picklist pane. The Export Picklist(s) dialog appears.
2. A list of available picklists appear. Select the checkbox against the picklist name that you want to export.

   To select all the picklists at once, select the checkbox present against the option **Select Picklist(s) to Export**.
3. Click **Export**. The picklist(s) get exported (downloaded) as a .zip file and saved in the local memory.

Adding and managing external functions

An external function is a user-defined function that executes the defined operation on the entity variable when used to define a rule or rule. Additionally, the system has predefined functions that perform specific functions. For more information, see the Understanding system functions section.

In the **Building Blocks** tab, click **External functions**. The list of available external functions appears.

Related topic(s)

- Creating an external function
- Editing an external function
- Unregistering an external function
- Defining custom external functions

Creating an external function

To create a new external function, follow the below steps:

1. In the **Building Blocks** > **External Functions** tab, click **Create New Function**. Alternatively, use the create option present in the navigation pane to add or create an external function. Click the create icon  , then select **Add External Function**. The External Function - Create a customised function dialog appears.
2. Refer to the below table to specify details in the available fields:

Field	Description
Application Name	Specify a name for the application.
Function Name	Specify a name for the external function.
Function Type	Select the type of external function: • In Condition And Action • In Action Only

Field	Description
Return Type	Select a return type for the external function.

3. Click **Add**. The field to specify function parameter and data type appears. Specify the **Parameter Name** and select the **Data Type** against it.
   Repeat this step to add more parameters to the external function.

To remove a specific parameter, select the checkbox present against the parameter name and click Delete. The selected parameters get deleted.

4. Click **Save**. The external function gets created and added to the list.

Related topic(s)

- Creating and defining a rule
- Creating a rule flow
- Understanding system functions

Editing an external function

To edit an external function, follow the below steps:

1. In the **Building Blocks** > **External Functions** tab, click the ellipsis icon ⋯ present against the required external function and select **Edit**. The External Function – Set/Edit the details of the external function dialog appears.
   The Application Name and Function Name fields remain disabled.
2. Make the necessary changes and click **Save**.

You can modify the input parameters and return type of the function. Modifying a function definition invalidates the rules it is associated with. You must update and check in the affected the rules.

Unregistering an external function

To unregister an external function, follow the below steps:

1. In the **Building Blocks** > **External Functions** tab, click the ellipsis icon ⋯ present against the required external function and select **Unregister**.
   The Unregister function dialog appears.

Unregistering a function invalidates the rules it is associated with. You must update and check in the affected the rules.

2. Click **Ok** to confirm. The function gets unregistered.

Defining custom external functions

For defining customized external functions in Rule Builder, you need to configure the server-side settings, after defining them in the application.

To define a custom external function, perform the following steps:

1. Define the code in a java file that contains the following package - **com.newgen.func**.
   For example, following is the sample function code for the sum of three integers.

```
package com.newgen.func;
import java.io.Serializable;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class External
implements Serializable {
  public static int sum(int paramInt1, int paramInt2, int paramInt3) {
```

```
    return paramInt1 + paramInt2 + paramInt3;
  }

}
```

2. Compile the code and create a class (JAR) file for it.
3. Place the class (JAR) file at the following location - *C:\jboss-eap-7.x\modules\omnidocs_library\main*.
4. Input the entry for this class (JAR) file in the **module.xml** file present at this location - *C:\jboss-eap-7.x\modules\omnidocs_library\main*.

Working with rules

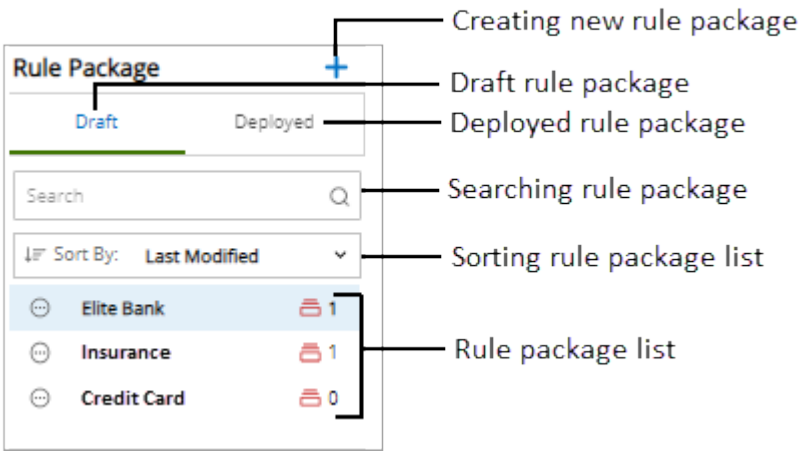
A business rule is a set of clearly defined conditions. When invoked, it executes an associated rule service. Thus, a rule governs the workflow of a system allowing you to decide what actions must execute under particular circumstances.

Using the Rules tab, you can create, deploy, view, and manage rule packages and rules. You can also import rule packages. The separation of rule packages into drafts and deployed simplifies the management of rules and rule packages.

The Rules tab contains the following panes:

- Rule package pane
- Rules pane
- Rule package or rules definition pane

Rule package pane

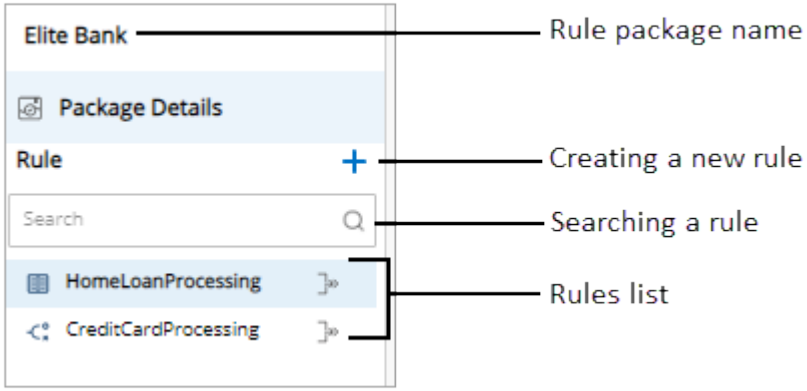


The rule package of the Rules tab contains the following UI elements:

Element	Description
Creating a new rule package	Click the add icon + in the rule package to create a new rule package. This option is available under the Draft tab.
Draft rule package	Click **Draft** in the rule package to view list the rule packages in draft state, that is, sent for approval, checked out, new, and not deployed. This tab appears by default.

Element	Description
Deployed rule package	Click **Deployed** in the rule package to view the list of deployed rule packages.
Searching rule package	Use the search box to search the rule package by name.
Sorting rule package list	Use the **Sort By** dropdown to sort the list of rule packages by one of the following criteria: • **Last Modified** — sort the list of items as per the recents. • **Name (A-Z)** — sorts the list of items in alphabetically ascending order. • **Name (Z-A)** — sorts the list of items in alphabetically descending order.
Rule packages list	Displays the list of rule packages available under the Draft or Deployed tab.

Rules pane

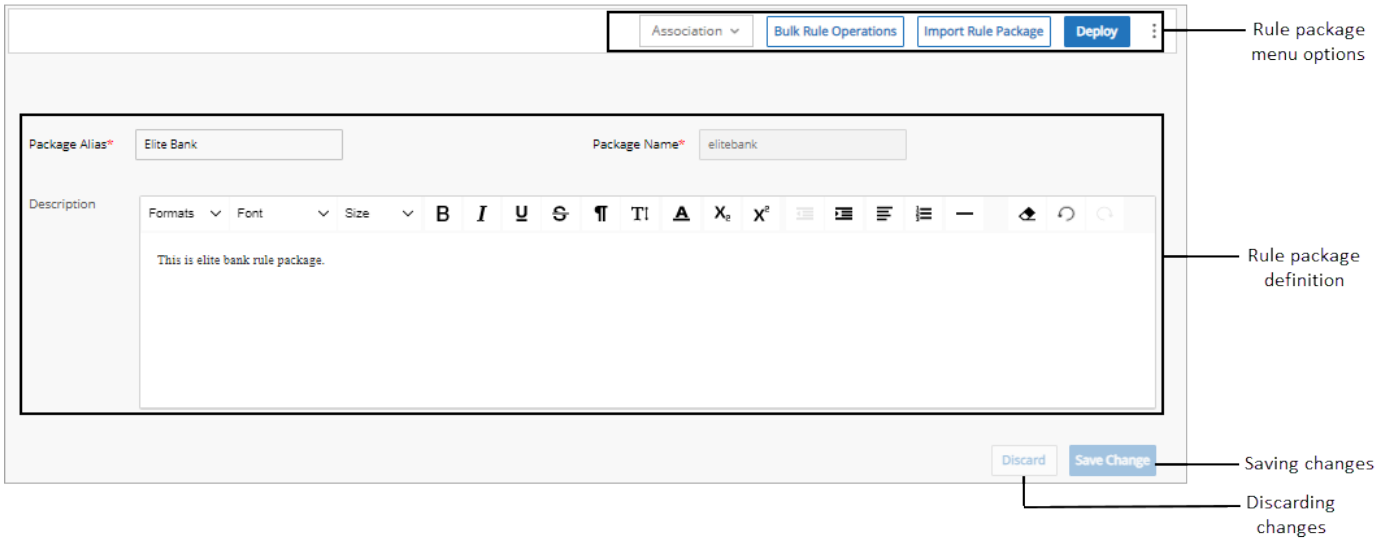


The rules pane contains the following UI elements:

Element	Description
Rule package name	Name of the selected rule package appears on top of the rules pane.
Creating a new rule	Click the add icon + to create a new rule. This option is available under the Draft tab.
Searching a rule	Use the search box to search the rule by name.
Rules list	Displays the list of rules available under the selected rule package.

Click the unpin icon present with the rules pan to collapse the rules and rule package pane. This maximizes the rule package definition pane. Likewise, in order to exit the maximized view, click the expand icon .

Rule package or rules definition pane



The definition of the selected rule package appears by default. To view the definition of a rule, select the required rule from the rules pane.

The rule package definition pane contains the following UI elements:

Element	Description
Rule package menu options	The rule package menu bar appears on the top of the rule package definition pane.
Editing or viewing rule package details	In the case of draft rule packages, you can modify the package alias and description of the package. While in the case of deployed rule packages, you can view the rule package details such as alias and package name and description.
Rule package definition	Displays the name, alias, and description of the rule package. You can modify the Package Name and Description.
Saving changes	Click **Save Change** to save the changes made to the rule package details.
Discarding changes	Click **Discard** to discard the changes made to the package details.

Creating a rule package

A rule package is a collection of multiple rules serving related purposes. For example, a rule package named ScoreCard contains rules such as age score, gender score, and payroll score.

To create a rule package, follow the below steps:

1. Click **Draft** under the Rules tab, then click the add icon + present in the left pane. Alternatively, use the create option present in the navigation pane to add or create a rule package. Click the create icon  , then select **Add Rule Package**. The New Rule Package Creation dialog appears.
2. Specify the following details in the dialog:

Field	Description
Package Alias	Specify the rule package alias. It is an alternative name used for the rule package.
Package Name	The package name generates automatically based on the rule package alias. This cannot be modified.
Description	Specify a description about the rule package. For more information, see the Formatting text section.

3. Click **Save** to create the rule package.

You can also copy or import an existing rule package to create a new rule package.

Related topic(s)

- Associating arguments with a rule package
- Associating picklist with a rule package

Associating arguments with a rule package

The argument association feature allows you to associate entity variables with a rule package and then use these as arguments in defining rules. The Argument Association dialog appears as soon as you create the rule package. Else, you can select the required rule package from the list of Draft rule packages, click the **Association** dropdown present in the upper-right corner of the rule package definition pane, and then select **Argument**.

The Argument Association dialog contains the following fields:

Field	Description
Select Entity Group	Displays the list of entity groups available within the logged in cabinet.
Searching Entity Group	Allows searching of entity group by name.
Select Entity Class	Displays the list of entity classes available within the selected entity group.
Parameters	Displays the list of added parameters.
Type	Displays the added parameters as condition, action, or condition and action type parameters. • Selecting **Condition** allows you to use the selected member variable to provide input only. • Selecting **Action** allows you to use the selected member variable to get the output only. • Selecting **Condition/Action** allows you to use the selected member variable to provide the input and get the output both.
Adding entity +	Allows creation and addition of new entity member variable. For procedural steps, refer to Adding variable.

To associate arguments with the rule package, follow the below steps:

1. Select the required entity group from the **Select Entity Group** section. The list of entity classes present within the selected entity group appears in the **Select an Entity Class** section.
2. Click the add icon + to add the required entity class as a parameter.
3. Using the **Type** dropdown, specify the selected entity variable as **Condition**, **Action**, or **Condition/Action** parameter.

You can add more than one entity class as arguments with the rule package. Member variables belonging to the selected entity classes get used to define the rules in the rule package.

4. Click **Save**. The arguments get associated with the rule package.

When defining rules, you can use the associated arguments only.

Related topic(s)

- Creating and defining a rule
- Viewing arguments associated with a rule package

Removing added parameters

To remove added parameter(s), follow the below steps:

1. In the **Argument Association** dialog, select the checkbox against the required parameter(s). The Delete option enables.
2. Click **Delete** to remove the added parameter from the arguments list.
3. Click **Save** to save the changes made.

Removing the parameters used to define the conditions and actions of existing rules available in the rule package invalidates them.

Associating REST API with a rule package

NewgenONE Rule Builder can consume the REST services that are registered in the Service Catalog of the Process Designer module. You can associate any REST service with a rule package to perform a specific operation.

For example, a bank wants to enhance its customer onboarding process by integrating external REST APIs within a rule. When customers apply for a savings account, the rule engine evaluates their credit score through a credit bureau API that checks for any suspicious activity using a fraud detection API. The API then approves or rejects the application based on predefined criteria, streamlining the account opening process while ensuring compliance and security.

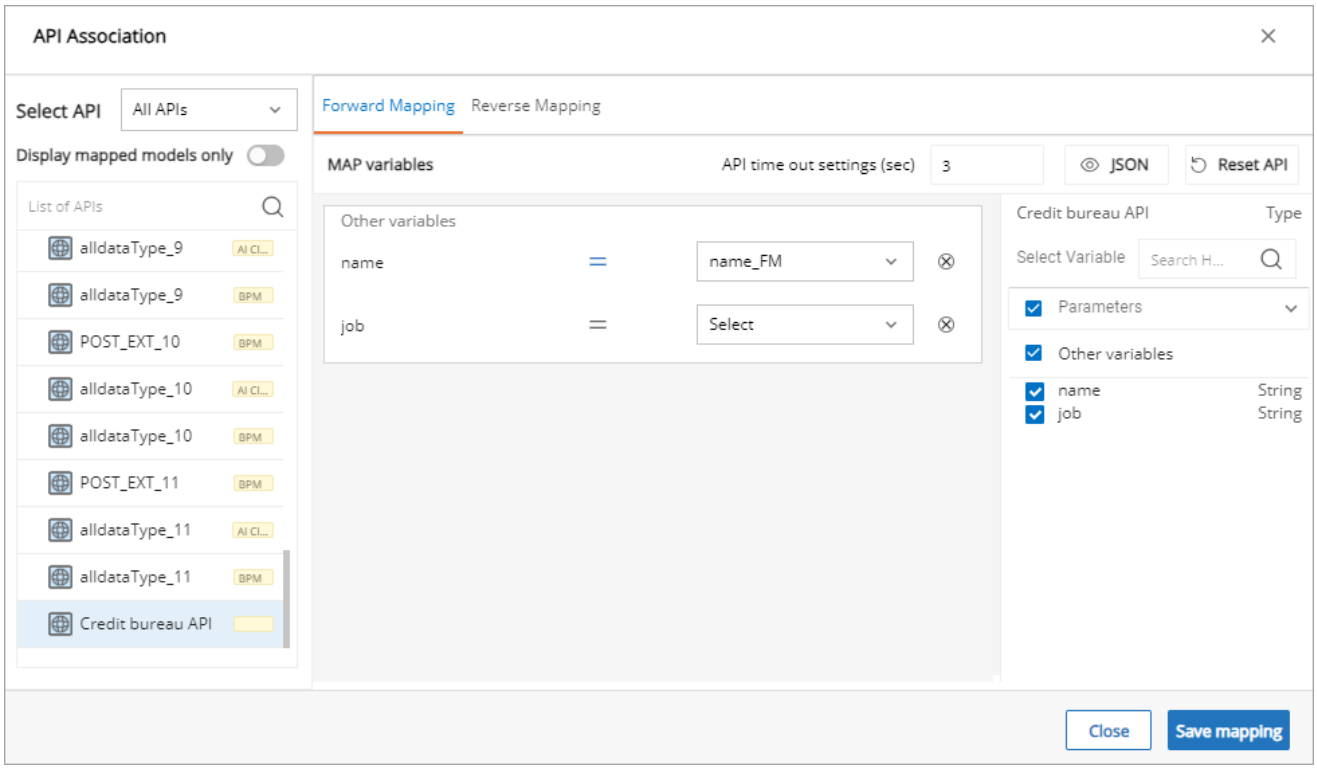
To associate a REST API with a rule package, follow the below steps:

1. In the Rule Package pane, from the Draft tab, select the required rule package.
2. In the right pane, click the **Association** dropdown and select **REST API**. The API Association dialog appears.

   Ensure that you associate the required argument before associating the REST API. This enables you to map the selected API variable with the required entity member variable and then use the mapped API variables while defining rules. For procedural details, see the Associating arguments with a rule package section.
3. (Optional) Turn on the **Display mapped models only** toggle to show the already mapped APIs.
4. Select the required API to map the API variables. In case the API list is long, you can navigate to the required API as follows:
   - Click the **Select API** dropdown and select the required API domain. By default, the All APIs option is selected.
   - Search the required API using the **List of APIs** search box.

   In the right pane, the list of all variables related to the selected API appears.
5. Select the checkbox against the required variables to map with the corresponding entity member variables. You can select all variables at once by selecting the **Other variables** checkbox.
   The selected variable appears in the MAP variables section.

6. In the Forward Mapping tab, from the dropdown, choose the required forward mapping values for the selected variables. The forward mapping defines the request parameters of an API.



7. Click the **Reverse Mapping** tab and define the reverse mapping values as explained in steps 5 and 6. The reverse mapping defines the response parameters of an API.
8. Click **Save mapping**. The API is associated with the selected rule package, and a green dot appears on the left pane displaying a successful association with the mapped API.

The following are the additional options:

- **API time out settings** — This allows you to specify the API response time (in seconds) after which the API call expires.
- **JSON** — This displays a preview of forward and reverse mapping parameters and their associated values in JSON format.
- **Reset API** — This button allows you to clear the selected options and redefine the mapping.

Once the API is associated with the rule package, you can use the API variable while defining the rule. For procedural details, see the Creating and defining a rule section.

Creating and defining a rule

NewgenONE Rule Builder allows you to create and define the following type of rules:

- English rule
- Decision rule

Related topic(s)

- Sending a rule for approval
- Understanding system functions

Defining an English rule

An English rule uses multiple if-else statements to define business logic. For example, an apparel store wants to offer a 3% discount to customers who spend more than INR 7000 while shopping at their store. The developer creates a simple English rule "If Spend is greater than 7000, set Discount to 3%" and implements it at the billing desk of the store. Now, whenever the bill of a customer exceeds INR 7000, the system automatically applies a 3% discount and calculates the final bill accordingly.

To define English rules, follow the below steps:

1. In the **Draft** tab, select the required rule package. The definition of the selected rule package appears.
2. Click the add icon + in the rules pane. The New Rule Creation dialog appears.
3. Specify the following details:
   - **Rule Name** — Specify the name of the rule.
   - **Rule Type** — Select **English Rule** to create and define a rule using if-else statements.
   - **Description** — Specify a description of the rule. For more information, see the Formatting text section.
4. Click **Save.** The rule definition pane appears.
5. Build the rule using the following conditional statements:
   - If
   - Else if

- Else
- Adding break rule

6. Click **Save** in the header of the rule definition pane.

Specifying If statement

To specify an If statement, follow the below steps:

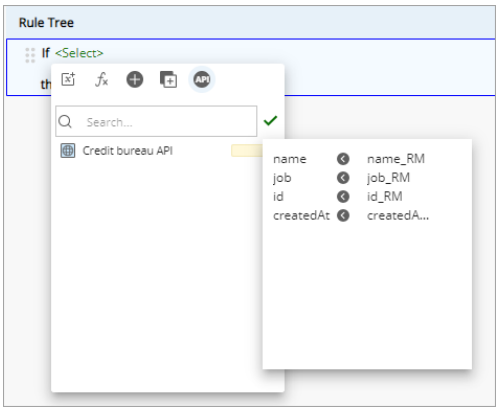
1. In the Rule Tree section, click **<Select>** present after *If*. A pop-up appears.
2. Select the field using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.
System Function List	• Click the function icon to view the list of available system functions. For more information on system functions, see the Understanding system functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as field.

Option	Description
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the selected data type, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete the variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a field.
REST Service	This option allows you to associate a mapped REST API variable with the selected rule. On associating, the API provides the desired result when the rule is executed. For procedural details to map the REST API, see the Associating REST API with a rule package section. To associate REST API variables with a rule condition, click here.

To associate REST API variables with a rule condition, follow the below steps:

a. Click the **REST Service** icon . The list of all the mapped APIs appears.
b. Select the required API. You can search an API using the Search box. The list of mapped API variables appears.
c. Select the required variable.



If you want to modify the forward mapping of an API variable, then follow the below sub-steps, otherwise move to the next step:

i. Hover over the required variable and click the ellipsis icon against it. The API Association dialog appears.
ii. From the dropdown, select the required entity member variable and click **Save And Use**. The forward mapping of the selected variable is modified.

d. Click the check icon .

3. Click **<operator>**. A pop-up appears.
4. Select the required operator. The available operators are:

Operator	Description
==	Checks if the values of two operands are equal or not case-sensitive. If yes, then the condition becomes true.
>	Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.
<	Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.

Operator	Description
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.
!=	Excludes rows that match the criteria followed by == operator.
LIKE	Checks if the values of two operands are equal or not case-insensitive. If yes, then the condition becomes true.
IN	Use this operator to specify multiple values. It helps in reducing the need for multiple OR condition. It returns True if the expression is equal to any of the values in the IN list “IN (value1, value2, …. value_n)”. It works with all data types, except Boolean.
NOT LIKE	Excludes those rows that matching the criteria followed by LIKE operator.
RANGE	Use this operator to specify a range for an input. For example, if the salary is between 10000 and 50000 the person is eligible for a loan. The range includes the boundary values as well.
NOT IN	In essence, it is the opposite of IN operator, where the condition satisfies if the entered value does not appear in the list. Example: NOT IN (12, 15, 20, 40), any value except these satisfies the condition and rule gets executed.

Availability of operators depends on the type of field selected using the first dropdown.

5. Click **<Select>** to select the value against the field selected in the previous steps using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.

Option	Description
System Function List	• Click the function icon to view the list of available system functions. For more information, see the Understanding system functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as value.
Add Constant	Specify a constant value and click the check icon to save it.
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a value.

Option	Description
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a value.
REST Service	See the REST Service option.

6. (Optional) Click **LogicalOp** to specify another rule condition and link it logically to the previous rule condition.
7. Select one of the following:
   - **AND** — to execute the action only when all relational statements combined with the AND operator are true.
   - **OR** — to execute the action if at least one of the relational statements combined with the OR operator is true.

   For a logical operator, repeat steps 1 to 5 to define another if condition.
8. Click **<Select>** present after *then* and select one of the following:
   - **Set** — to specify an action that must be executed upon satisfying the condition(s) specified in the previous steps. For procedural steps, see the Setting Actions section.
   - **ExecuteAPI** — to execute the associated API upon satisfying the defined rule condition(s) and provide the desired result. After selecting the ExecuteAPI option, click **<Select>** and select the required API.

     If you want to modify the default API variables for forward and reverse mappings, then follow the below sub-steps:
     a. Click **<Select>** and hover over the required API.
     b. Click the edit map icon Map against it. The API Association dialog appears.
     c. In the Forward Mapping tab, from the dropdown, select the required entity member variable.
     d. Click the **Reverse Mapping** tab and from the dropdown, select the required entity member variable.
     e. Click **Save And Use**. The forward and reverse mapping of the selected variables are modified.
   - **Exit** — to terminate the rule execution upon satisfaction of condition(s) specified in the previous steps.

9. Click **Save** to save the rule definition.

Setting actions

- Click **<Select>** and select the required field using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a field.

The equal operator appears selected by default.

- Click **<Select>** to select the value against the field selected in the previous steps using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.
System Function List	• Click the function icon to view the list of available system functions. For more information, see the Understanding system functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as value.
Add Constant	Specify a constant value and click the check icon to save it.
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a value.

Option	Description
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a value.

Additional options:

- Click the copy icon to create a copy of the if statement.
- You cannot delete an if statement.
- Click the expand icon to view the complete definition of the if statement. Similarly, click the collapse icon to exit the expanded view.

Specifying Else if statement

Use else if to specify a new condition statement that gets executed when the if condition fails.

To specify an Else if statement, follow the below steps:

1. Click the **+Else if** button present in the bottom-right corner. An additional else if field gets added to the rule definition.
   You can also create an Else if statement by copying an existing if or else if statement.
2. To define an else-if statement, follow all the steps mentioned in the Specifying If statement section.

Additional options:

- Click the copy icon to add another else-if statement by creating a copy of the current one.
- Click the delete icon to delete the else if statement.
- Click the expand icon to view the complete definition of the else- if statement. Similarly, click the collapse icon to exit the expanded view.

Specifying Else statement

Use else to specify a condition statement that gets executed when all the previously mentioned conditions fail.

To specify an Else statement, follow the below steps:

1. Click the **+Else** button present in the bottom-right corner. An additional else field gets added to the rule definition.
2. To define an else statement, follow steps 7 to 9 mentioned in the Specifying If statement section.

Additional options:

- Click the delete icon to delete the else statement.
- You cannot create a copy of an else statement.
- Click the expand icon to view the complete definition of the else statement. Similarly, click the collapse icon to exit the expanded view.

Adding a break rule

The break rule feature allows you to skip the execution of the current rule. It prevents a rule from getting executed when any of the entities used during the rule definition gets a NULL value. A skipped rule redirects control to the next rule in the package.

To mark a rule as a break rule, follow the below steps:

1. Select the **Mark as Break Rule** checkbox to skip the execution of the current rule on receiving a null value.
2. Click **Save** to complete the marking of the break rule.

Additional options:

- To remove the break rule condition, clear the **Mark as Break Rule** checkbox and click **Save**.

- Click the expand all icon present next to the Mark as Break Rule checkbox to view the complete definition of the English rule. Similarly, click the collapse all icon to exit the expanded view.

Defining a Decision rule

A decision rule is a collection of rows and columns containing multiple conditions and actions. Using decision rules, you can implement complex business logic. For example, a bank wants to review the eligibility of an applicant for a home loan based on the Salary and Credit Score range. The developer at the bank defines a Decision rule, for different scenarios as follows:

Salary Range	Credit Score Range	Eligibility
30000 to 50000	600 to 700	True
50001 to 75000	500 to 600	False
75001 to 100000	600 to 800	True

Now, whenever a customer applies for a home loan, the system generates an automatic decision based on the aforementioned logic.

To define a decision rule, follow the below steps:

1. In the **Draft** tab, select the required rule package under which you want to add rules. The definition of the selected rule package appears.
2. Click the add icon + in the rules pane. The New Rule Creation dialog appears.
3. Specify the following details:
   - **Rule Name** — Specify the name of the rule.
   - **Rule Type** — The rule type appears selected as **Decision Rule** by default.
   - **Rows and Columns** —
     - Enter the number of columns required to define the condition logic for the rule in the **Condition Columns** field.
     - Enter the number of columns required to define the action that must take place on the satisfaction of the specified condition(s) in the **Action Columns** field.
     - Enter the number of rows required to specify different conditions and their respective actions in the rule in the **Number of Rows** field.

- **Description** — Specify a description of the rule. For more information, refer to the Formatting text section.

4. Click **Save**. The rule definition pane appears.

Defining conditions

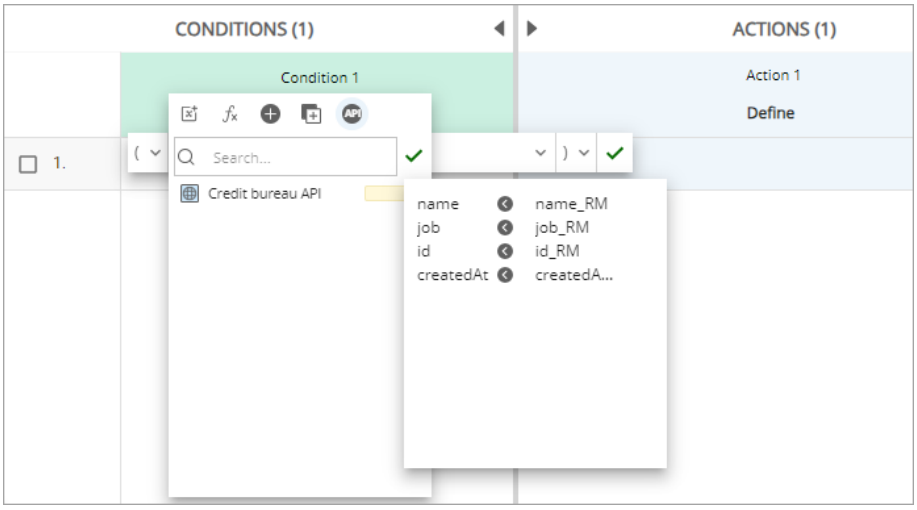
5. Click **Define** in the **CONDITIONS** column. A pop-up appears.
6. Click the first dropdown to select the opening brace. The available options are (and ((.
7. Click the second dropdown to select the field using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.
System Function List	• Click the function icon to view the list of available system functions. For more information, see the Understanding system functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as field.

Option	Description
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a field.
REST Service	This option allows you to associate a mapped REST API variable with the selected rule. On associating, the API provides the desired result when the rule is executed. For procedural details to map the REST API, see the Associating REST API with a rule flow section. To associate REST API variables with a rule condition, click here.

To associate REST API variables with a rule condition, follow the below steps:

a. Click the **REST Service** icon . The list of all the mapped APIs appears.
b. Select the required API. You can search an API using the Search box. The list of mapped API variables appears.



c. Select the required variable.
If you want to modify the forward mapping of an API variable, then follow the below sub-steps, otherwise move to the next step:
   i. Hover over the required variable and click the ellipsis icon against it. The API Association dialog appears.
   ii. From the dropdown, select the required entity member variable and click **Save And Use**. The forward mapping of the selected variable is modified.
d. Click the check icon .

8. Click the third dropdown to select an operator. The available operators are:

Operator	Description
==	Checks if the values of two operands are equal or not case-sensitive. If yes, then the condition becomes true.
>	Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.
<	Checks if the value of the left operand is less than the value of the right operand, if yes then condition becomes true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then condition becomes true.

Operator	Description
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.
!=	Excludes rows that match the criteria followed by == operator.
LIKE	Checks if the values of two operands are equal or not case-insensitive. If yes, then the condition becomes true.
IN	Use this operator to specify multiple values. It helps in reducing the need for multiple OR condition. It returns True if the expression is equal to any of the values in the IN list “IN (value1, value2, ... value_n)”. It works with all data types, except Boolean.
NOT LIKE	Excludes those rows which are matching the criterion followed by LIKE operator.
RANGE	Use this operator to specify a range for an input. For example, if the salary is between 10000 and 50000 the person is eligible for a loan. The range includes the boundary values as well.
NOT IN	In essence, it is the opposite of IN operator, where the condition satisfies if the entered value does not appear in the list. Example: NOT IN (12, 15, 20, 40), any value except these satisfies the condition and the rule gets executed.

9. Click the fourth dropdown to select the value against the field selected in the previous steps using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.
Placeholder	Click the placeholder icon to mark the value field as a placeholder. This enables empty rows below the condition column thus allowing you to enter multiple values directly in the decision table.

Option	Description
System Function List	• Click the function icon to view the list of available system functions. For more information, see the Understanding System Functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as value.
Add Constant	Specify a constant value and click the check icon to save it.
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a value.

Option	Description
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a value.
REST Service	See the REST Service option.

10. Click the fifth dropdown to select the closing brace. The available options are) and)).
11. Click the sixth dropdown to select a logical operator to specify another rule condition and link it logically to the previous rule condition. Select one of the following:
    - **AND** — to execute the action only when all relational statements combined with the AND operator are true.
    - **OR** — to execute the action if at least one of the relational statements combined with the OR operator is true.

    The dropdown to select logical operators appears only when you add more than one condition columns to the decision table.
12. Click the check icon to save the defined condition.
    For a logical operator, repeat steps 7 to 14 to define another if condition.

Defining actions

13. Click **Define** in the **ACTIONS** column. A pop-up appears.
14. Click the first dropdown to select one of the following options:
    - **Set** — to specify an action that must be executed upon satisfaction of condition(s) specified in the previous steps. For procedural details, see Setting actions.
    - **Execute** — to execute rules available in the same rule package independently as actions. To use the execute command refer to the below points:
        - Each rule called using the execute works on the same business objects (entity variables) mapped to the current rule package. The updated values are taken as inputs subsequent execution of other rules of the package.
        - If the updated data values are not available, then the rule is executed using the default values specified for the variables.
        - After execution of the called rule, the control comes back to the calling rule.

- **ExecuteAPI** — to execute the associated API upon satisfying the defined rule condition(s) and provide the desired result. After selecting the ExecuteAPI option, click the rule row and select the required API.
  If you want to modify the default API variables for forward and reverse mappings, then follow the below sub-steps:
  a. Click the rule row and hover over the required API.
  b. Click the edit map icon Map against it. The API Association dialog appears.
  c. In the Forward Mapping tab, from the dropdown, select the required entity member variable.
  d. Click the **Reverse Mapping** tab and from the dropdown, select the required entity member variable.
  e. Click **Save And Use**. The forward and reverse mapping of the selected variables are modified.
- **Exit** — to terminate the rule execution upon satisfaction of condition(s) specified in the previous steps.

> You can add only one column for the exit and execute commands. The set command can have multiple columns.

- Click **Save**. The rule definition gets saved.

Setting actions

- Click the second dropdown to select the field using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.

Option	Description
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a field.

The equal operator appears by default.

- Click the third dropdown to select the value against the field selected in the previous steps using one of the following options:

Option	Description
Variable List	The variable list displays the associated entity members. It appears by default. Select the required entity. Use the search box to search the entity member by name.

Option	Description
Placeholder	Click the placeholder icon to mark the value field as a placeholder. This enables empty rows below the condition column thus allowing you to enter multiple values directly in the decision table.
System Function List	• Click the function icon to view the list of available system functions. For more information, see the Understanding system functions section. • Select the required system function and define the parameters accordingly. • Click the check icon to save the defined function as value.
Add Constant	Specify a constant value and click the check icon to save it.
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a value.

Option	Description
Add Argument	• Click the add argument icon to associate another entity with the rule package and use it for defining the rule. For procedural steps, see the Associating arguments with a rule package section. The associated entity member variables now appear as variables. • Click the variable icon and select the newly associated variable as a value.

- Click the check icon to save the defined action.
  Repeat the above-mentioned steps to define more actions.

Related topic(s)

Performing Rule Builder configurations

Using Decision rule toolbar

The decision rule toolbar appears in the upper-right corner of the rule definition pane and it consists of the following options:

Option	Description
Preview	Use this option to view the rule description of the selected row in a simple if-based format.
Upload	Use this option to create a new decision rule by uploading a rule definition in an XLS document. For more information, see the Import data to define a Decision rule section.
Download	Use this option to download and save the rule definition to your local machine in XLS format.
Add Otherwise	Otherwise Rule is a rule in the decision table that gets executed when none of the other rows of the decision table get executed. It is shown by all the conditions in the table marked as * (asterisk). Use this option to add an otherwise condition in your decision rule.

Option	Description
Copy	Use this option to create a copy of the selected row. On clicking the copy icon, the copied row gets added just below the selected row.
Add Right and Add Left	Select the required column and click the Add Right icon to add a column to the right of the selected one. Similarly, click the Add Left icon to add another column to the left of the selected one.
Add Above and Add Below	Select the required row and click the Add Above icon to add a row above the selected one. Similarly, click the Add Below icon to add another row below the selected one.
Move Right and Move Left	Select the required column and click the Move Right icon to move the selected column to one position right. Similarly, click the Move Left icon to move the selected column to one position left.
Move Up and Move Down	Select the required row and click the Move Up icon to move the selected row one position above. Similarly, click the Move Down icon to move the selected row one position below.
Previous and Next	Click the previous icon to move to the previous batch of rows in the table. Similarly, click the next icon to move to the next batch of rows in the table.
First and and Last	Click the first icon to move to the first batch of rows in the table. Similarly, click the last icon to move to the last batch of rows in the table.

Import data to define a decision rule

This feature allows you to import data in XLS format from your local machine and use it to add data to already-defined rules. This option is available in the Draft mode only. If you want to add data to a deployed rule, you must check out the rule first and then upload the XLS file.

To import data to define a decision rule, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package in the **Draft** subtab.
   A list of rules available within it appears. Select the rule in which you want to import the data using XLS. The rule definition appears.
2. Click the upload icon in the toolbar. The Upload Rule dialog appears.
3. Specify the size of the header in the **Header Size** text box. The value is set to 4 by default.

   Specify 0 if no header is present in the file.
4. Click **Browse** to select the XLS file containing the required data from your local machine.

   For uploading a large file, the xml tag <uploadThresholdSize> must be set to a value greater than the size of the file upload inside brms.war->WEB-INF->web.xml.
   For example, if the size of the file is 5 MB then the value of <uploadThresholdSize> must be greater than 5000k.
5. Click **Save** to complete the XLS upload. The system updates the existing data values with the uploaded ones.

- Ensure uploading data in XLS format.
- Uploaded data must have the same columns as defined in the current rule.
- Do not leave the first row should empty in the current rule as the system reads the data type from the first row only.
- Once the file gets uploaded, the system replaces the previously defined rule rows with uploaded ones.

Sending a rule for approval

Once you create a rule, you can send it for approval to the supervisor user. The details of the rule sent for approval appear in the inbox of the checker as well as the maker. For more information on handling approval requests, see the Managing inbox section.

To send a rule for approval, follow the below steps:

1. Navigate to the **Rules** tab.
2. Select the newly created or modified rule that you want to send for approval under the **Draft** subtab. The definition of the selected rule appears.
3. Review the rule definition before sending it for approval.
4. Click **Send for Approval**. The rule gets added to the checker's queue for approval.

You cannot delete a rule that is in the approval process.

Perform rule check-in

Upon receiving approval, you must check-in the approved rule in the system to make it available for deployment with the rule package.

To check-in a rule, follow the below steps:

1. Navigate to the **Rules** tab.
2. Select the required rule package under the **Draft** subtab. The definition of the selected rule appears.
3. Select the approved rule that you want to check in the system.
4. Click **Check in** to complete the operation. You can now deploy it along with the rule package.

Related topic(s)

Deploying a rule package

Deploying a rule package

Upon receiving approval from the checker for a new or modified rule and performing the check-in operation, you can deploy a rule package to make it available for use in the business processes.

To deploy a rule flow, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package under the **Draft** subtab. The definition of the selected rule package appears.
2. Click **Deploy**. The Deploy Rule Package dialog appears.
3. Select one of the following options:
   - **Increment Rule package major version** — to increment the major version of a rule package. For example, if the current version is numbered 1.1, then the next version gets incremented 2.0.

- **Increment Rule package minor version** — to increment the minor version of a rule package. For example, if the current version is numbered 1.1, then the next version gets incremented as 1.2.
- **Replace Rule package version** — to replace the current version of the rule package.

> When you deploy a newly created rule package, the options to increment a minor version and replace appear disabled.

4. Select Deploy as one of the following options:
   - **None** — to deploy the rule package neither as a web nor REST service.
   - **WebService** — to deploy the rule package as a Webservice.
   - **REST Service** — to deploy the rule package as a REST service.
   - **Both** — to deploy the rule package both as a web and REST service.
5. Select the **Secured** checkbox to secure the deployed rule package and rules listed within it from unauthorized access and add an additional layer of security.
6. Click **Deploy** to complete the deployment of the rule package.
   The deployed rule package appears under the Deployed subtab.

> To deploy a rule package, all rules added within it must be in the approved and checked-in state.

Related topic(s)

- Testing a rule package
- Testing a rule package deployed as a RESTservice
- Viewing rule package REST endpoint data
- Viewing items deployed as REST services

Testing a rule package

Use the Test option to test the rules present in a rule package for their functional correctness. When you run a test on the main rule package, the latest version of that rule gets tested. You can test the latest version as well as the other versions of a rule package can be tested individually. You need to provide the input value, and on testing the output gets generated. If any input field is left blank, the default value is considered for that field. This option is available in the lower-right corner of the Deployed subtab.

> If you have used the break rule in the rule definition, then the default value gets ignored and the execution of the rule gets skipped. Once the rule gets skipped, the control gets redirected to the next rule in the rule package. For more information, see the Adding a break rule section.

You can perform the following type of tests:

- Test of a single request
- Bulk testing

Test a single request

To test a single request, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click **Test**. The Test Rule Package dialog appears.
   The entities appearing in the Test Rule Package dialog have an accordion icon present with it. It allows you to expand and collapse categories of content. To expand entities and view their entity member variables, click the accordion icon  . Click the accordion icon  again to collapse it.
3. Enter the required Input.
4. Click **Test**. The output gets generated based on the input you entered.
   Similarly, select the required version to Test different versions of a deployed rule package.

Use the Version Free Test option to test the deployed rule package irrespective of its version.

Bulk testing

To perform bulk testing, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click **Test**. The Test Rule dialog appears.
3. Click **Bulk Testing**. The Bulk Rule Testing dialog appears.
4. Select the **click here** link to download the spreadsheet with the correct data format.
   The spreadsheet format gets downloaded.
5. Open the downloaded spreadsheet on your local machine and enter the **Input Values** according to the specified data types and the expected **Output Values**. Then save and close the spreadsheet.
6. Click **Choose a File** and upload the spreadsheet of the rule package.
7. Click **Test**. The results of the execution appear.
8. Click **Download** to download and view the test report.

Testing a rule package deployed as a RESTservice

The Test via REST is an execution feature. Use this feature to test the rule package deployed as REST services. When a test is run on the main rule package, the latest version of the rule package gets tested as well. The latest version as well as the other versions of a rule package also get tested individually. This option is available in the lower-right corner.

You can perform the following type of tests:

- Testing a single request
- Bulk testing

Testing a single request

To test the rule package deployed as a REST service, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
   The rules listed with the package are available for testing.
2. Click **Test via Rest**. The Test Rule Via Rest dialog appears.
3. In the **Input** section, enter the required values against the fields.
4. Click **Test**.
   If you have encrypted the rule package, you must provide the key to access it.
   The output gets generated based on the entered input.

Bulk testing

To perform bulk testing, follow the below steps:

1. Select the required rule package under the Deployed subtab.
2. Click **Test via Rest**. The Test Rule Via Rest dialog appears.
3. Click **Bulk Testing**. The Bulk Rule Testing dialog appears.
4. Click the **click here** link to download the spreadsheet with the correct format of data.
   If you have encrypted the rule package, you must provide the key to access it.
   The spreadsheet gets downloaded.

5. Open the downloaded spreadsheet on your local machine and enter the **Input Values** as per the defined data type and the expected **Output Values**. Save and close the spreadsheet.
6. In the **Bulk Rule Testing** dialog, click **Choose a File** and upload the spreadsheet of the rule package.
7. If you have encrypted the rule package, you must provide the key to access it.
8. Click **Test**. The results of the execution appear.
9. Click **Download** to save the test report to your local machine and view it.

Related topic(s)

Viewing rule package execution logs

Performing operations on rule packages

This section explains the additional operations you can perform on the rule packages listed under the Draft and Deployed sub-tabs.

Associating picklist with a rule package

You can associate picklist with a rule package only after associating arguments with it.

To associate a picklist to a rule package, follow the below steps:

1. Select the required rule package under **Drafts** tab.
2. Click the **Association** dropdown, and select **Picklist** from the list. The Picklist Association dialog appears.
3. Select an **Entity Class** from the list. You can use the search box to search entity classes by name.

   The list of entity classes displays only those classes that were selected during argument association.
4. The list of member variables defined under the selected entity class appears adjacent to the list of entity classes. Select the required member variable. You can use the search box to search the variable by name.

5. Click the add icon to add the member variables to the list.
   Use the minus icon if you want to remove any added variable.
6. Select the **Picklist** against each entity member using the dropdown.

> The picklist dropdown contains only those picklists that are of the same data type as that of the selected entity member variable. The Picklist row remains blank if there is no matching picklist. The string type function, however, can take all data types.

7. Select the default value using the **Default** dropdown.

> The default value is the value defined at the time of picklist creation. It can be Static Data, Dynamic Data, or Runtime Data.

8. The WHERE clause option appears when you select a runtime picklist. Click it to view the information about available entities. The Where Clause Association dialog appears. It displays the query used while creating the runtime picklist.
9. Click the field against the variable field to map an entity value against it. Click **Save**. The entities get mapped.

   Repeat the above steps to add and associate more entity member variables and picklists.

> To remove a picklist added to the list, select the checkbox present against the picklist name and click Delete. The selected picklist gets deleted.

8. Click **Save**. The picklist association gets completed.
   Once you associate the arguments and picklists to the rule packages, you can create rules.

Related topic(s)

Viewing picklist associate with a rule package

Importing a rule package

Use the Import Rule Package feature to import a rule package on another server by uploading the rule package details in a .zip file from your local machine. Create a rule package with the same name as that of the uploaded .zip file. Also, you can check out an already deployed rule package having the same name to synchronize it with the imported one.

To import a rule package, follow the below steps:

1. Navigate to the **Rules** tab and select the rule package that you created with the same name as that of which you want to import under the **Draft** subtab.
2. Click the **Import Rule Package** button present in the upper-right corner of the page. The Import Rule Package dialog appears.
3. Click **Browse** in the **File Name** field to select the .zip file of the rule package from your local machine.
4. Select one of the following options to import the rule package:
   - **Append** — to add the new rules available in the imported file with the existing rules in the package.
   - **Overwrite** — to overwrite the rules present in the rule package with the ones available in the imported file.
5. Click **Import** to complete the importing of rule package.

Related topic(s)

- Exporting a rule package
- Exporting rule packages in bulk

Copying a rule package

Rule Packages can also be added by copying the existing rule packages. In addition to the rule package, the selected rules of the source rule package are copied to the new rule package.

To copy a rule package, follow the below steps:

1. Select the required rule package that you want to copy under the **Draft** tab.
2. Click ellipsis icon and select **Copy**. The Copy Rule Package dialog appears.
3. Specify the name of the new rule package in the **Package Alias** text box. System-generated **Package Name** appears based on the Package Alias you entered. This cannot be modified.
4. Select the checkbox present against the rule that you want to add to the new rule package.
   Use the search box to search rule by name.
   Use the **Select All** option to select all the rules available to add.

Clear the checkbox present against the name of the rule that you do not want to add to the new rule package.

5. Click **Copy**. A new rule package with the selected rules gets created and added to the list of draft rule packages.

Performing bulk operations on rule packages

Bulk rule operation option allows performing check in, check out, and send for approval tasks for multiple rules at once.

To perform bulk operations on rule packages, follow the below steps:

1. In the **Rules > Draft** sub tab, click **Bulk Rule Operations** present in the upper-right corner. The Bulk Rule Operations dialog appears.
2. Select:
   - **Send for Approval** — to send rules defined under the selected Rule Package for approval.
     - Select the checkboxes against the name of the rules that you want to send for approval.
   - **Check in** — to check in rules defined under the selected Rule Package.
     - Select the checkboxes against the name of the rules that you want to check-in.

     A rule can be checked in only by the user who has modified the rule or by the supervisor.
   - **Check Out** — to check out rules defined under the selected Rule Package.
     - Select the checkboxes against the name of the rules that you want to check out.
3. Click **Save** to complete the bulk operation.

Deleting a rule package

Use the Delete option to delete the currently opened version of the rule package.

To delete rule package, follow the below steps:

1. Select the required rule package under the **Draft** tab.
2. Click ellipsis icon and select **Delete**. The Confirm Deletion dialog appears.

3. Click **Delete**. The currently selected version of the rule package gets deleted.

Encrypting a rule package

The encryption feature allows you to prevent unauthorized access to rule packages or rules. If you apply encryption then the rule package gets locked using a key. The system asks for the key when anyone tries to access the encrypted rule. The rule definition appears only after entering the correct key.
You (if an authorized user) can work on different rules in the same rule package by entering the key only once. The key, however, must be entered again if you switch to a different rule package, or the session gets timed out.

You can set encryption at rule packages level, that is., only one key is maintained for all the rules defined within the rule package.

Points to remember:

- You must enter the encryption key to open an encrypted rule searched using Query.
- If you export an encrypted rule package and import it on a different cabinet, the rule package gets imported successfully, but to open it, you must enter the encryption key.
- Checker does not require any key to open an encrypted rule from the Checker queue. However, the key must be entered to open a rule from the rule repository.

Encryption key policy:

- Key must be a combination of alphanumeric and special characters.
- Minimum and maximum length of characters for the key is 8 and 25, respectively.
- There must be at least one alphabet, one numeric value, and one special character present in the key.
- Spaces are considered as special characters.

To encrypt a rule package and rules listed within it, follow the below steps:

1. Select the required rule package under the **Draft** tab.
2. Click the ellipsis icon ⋮ and select **Encrypt**. The Securing dialog appears.
3. In the **Enter Key** text box, specify the encryption key as per Encryption Key Policy.
4. In the **Confirm Key** text box, re-enter the key to confirm.
5. Click **Save**. The rule package gets encrypted.

Decrypting a rule package

To decrypt a rule package and rules listed within it, follow the below steps:

1. Select the required rule package that you want to decrypt under the **Draft** tab.
2. Enter the encryption key to open the rule package definition.
   The system asks for the encryption key if you are switching from a different Rule Package.
3. Click ellipsis icon and select **Decrypt**. A dialog to confirm decryption appears.
4. Click **Decrypt**. The rule package gets decrypted.

Defining execution mode of the rules

NewgenONE Rule Builder allows you to specify the mode in which rules within a package get executed.

The order of execution can be:

- **Sequential** — In the sequential mode, rules get executed sequentially. Example, there are five rules defined in a rule package: Rule 1, Rule 2, Rule 3, Rule 4, and Rule 5. In the sequential mode, Rule 1 gets executed first if it qualifies for execution, followed by Rule 2, and more. Now, say, if Rule 3 modifies the value of an entity, the already executed or checked rules, that is, Rule 1 and Rule 2 do not get executed or checked again for the new or modified value of the entity. The rules in sequence after Rule 3 get executed according to the modified value of the entity.
- **Rete** — Consider the same example of five rules defined in a rule package. If the execution mode is specified as Rete, Rule 1 gets executed first if it qualifies for execution, followed by Rule 2, and more. Let us assume that Rule 2 did not qualify and therefore, it does not get executed. Now, if Rule 3 modifies the value of an entity, the already checked rule (Rule 2), which was not executed previously, gets checked again for the modified value. If Rule 2 qualifies now, it gets executed. Already executed rules (Rule 1 in this case) do not get executed again for the modified value. Other rules in sequence follow the same execution procedure.

To specify the rule execution mode, follow the below steps:

1. Select the required rule package under the **Drafts** subtab. Click the ellipsis icon  .
2. Select **Execution Mode**. The Execution Mode dialog appears.
3. Select the order of execution for rules as **Sequential** or **Rete**.
4. Click **Save**. The property of the execution mode gets modified.

Exporting a rule package

Use the Export option to download and save the selected rule package to your local machine.

To export the rule package, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click the ellipsis icon  and select **Export**. The Export Rule Package dialog appears.
3. Click **Export**. The rule package gets downloaded as a .zip file in the local memory space of the client machine.

Related topic(s)

Importing a rule package

Exporting rule packages in bulk

Use the Bulk Export Packages option to export rule packages in bulk. The exported file includes information regarding the business object model, external functions, rules definitions, and audit trail for each selected rule package.

To export rule packages in bulk, follow the below steps:

> You can export the rule packages that are not checked out.

1. Select any rule package under the **Deployed** subtab.
2. Click **Bulk Export Packages**. A dialog with a list of available deployed rule packages appears.
3. Select the checkbox present against the name of the rule package that you want to export.

Use the Search box to search the rule package by name.
Use the **Select All** option to select all the rule packages available to export.

Clear the checkbox present against the name of the rule package that you do not want to export.

4. Click **Export**. The selected rule packages.
   Once the Rule Packages are exported get downloaded as a .zip file in the local memory space of the client machine.

Downloading model of a rule package

Whenever you deploy a rule package whether new or modified, different versions of the same get created, and these are categorized as Major Version and Minor Version. For example, if the current version is numbered 1.1 and the next version is numbered 2.0, then this method of incrementing the version is called the major version. If the current version is numbered 1.1 and the next version is numbered 1.2, then this method of incrementing the version is called the minor version.

The Download Model feature allows you to define or create entities (by importing) with the same properties as that of the downloaded rule package.

To download the model, follow the below steps:

1. Select the required rule package under the **Deployed** section.
2. Click ellipsis icon and select Download Model.
3. Select the version for which you want to download the model.
4. Click **Download**. The model gets downloaded as a .jar file. The downloaded file follows the naming convention as: Rule Package: ruleflowname_version.jar.

Depending on the type of browser you are using, you may be asked to save the file.

Downloading report of a rule package

The Report feature allows you to generate information about the different versions of a rule package. The generated report gets downloaded as a .zip file containing a consolidated Report.html file and a Help_File folder containing html files of rule package.

You can download the report in the following ways:

- Normal — The report downloaded in this mode can be viewed by anyone as it does not require any password to open it.
- Password Protected — This feature prevents unauthorized access of a report. A user-generated key must be provided at the time of downloading and opening the downloaded report.

Download normal report

To download a normal report, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click the ellipsis icon and select **Report**.
3. Select **Normal**.
4. Select the report format for downloading, **Word Doc** or **HTML**.
   The report gets downloaded as a .zip file.

Depending on the type of browser you are using, you may be asked to save the file.

Download password protected report

To download password protected report, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click ellipsis icon and select **Report**.
3. Select **Password Protected**.

4. Select the report type that you want to download, **Word Doc** or **HTML**. The Set Password Key dialog appears.
5. Specify a password Key.

   System asks you to enter the password key when you extract files from the downloaded .zip file.
6. Click **Download**. The report gets downloaded as a .zip file.

   Depending on the type of browser you are using, you may be asked to save the file.

Opening password protected report

To open the downloaded report, follow the below steps:

1. Navigate to the folder where the report is downloaded.
2. Select the method of extracting the files from the downloaded .zip file. The system asks to provide the password.
3. Enter the same password key.
4. Click **OK**. The files get extracted in a folder.

Managing versions of a rule package

Upon opening the definition of a rule package in the Deployed subtab, the version of the rule package appears alongside its name. When you deploy a rule package for the first time, version 1.0 is assigned to it.

When you redeploy a rule package in event of modifications to the rule definition listed within it, multiple major and minor version gets created.
For example, for a rule package named CustomerEligibiltiy the current version is 1.0. The developer makes modifications to the rules logic listed with the selected rule package and redeploys it as a minor or major version. If the rule package gets redeployed as a minor version, the version number changes to 1.1. While if it gets redeployed as a major version, the version number changes to 2.0.

The term Enabled or Disabled represents whether the current version of the rule package is available to external applications for use or not.

Upon redeploying a rule package in event of modifications to the rule definition, multiple major and minor version gets created.
The term **Enabled** or **Disabled** represents whether the current version of the rule package is available to external applications for use or not.

Click the version dropdown to view the following options:

- **Enable All** — to make all the versions of the rule package available to external applications.
    - Click **Enable All**. All the version of the rule package gets enabled.
- **Disable All** — to make all the versions of the rule package unavailable to external applications.
    - Click **Disable All**. The Confirm Disabling dialog appears.
    - Click **Disable**. All the version of the rule package gets disabled.
- Click **Enabled** or **Disabled** to view the following option for a specific version:
    - **Set As Latest** — to tag a lower-order rule package version as the latest in the database table.
    - **Enable or Disable** — to enable or disable the selected version.

Viewing audit trail of a rule package

Use the audit trail option to view the details of operations performed on the rule package such as approved, checked out, creation, modification, and more.

To view audit trail of a rule package, follow the below steps:

1. Select the required rule package under the **Draft** subtab.
2. Click the ellipsis icon and select **Audit Trail**. The Audit Trail for the selected rule package appears. The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the rule package such as modification, deletion, creation, and more.
Action by	This field displays the name of the user who has performed action on the rule package.
Timestamp	This field displays the date and time when the action was performed on the rule package.

Field	Description
Summary	This field displays a short summary of the action performed on the rule package.

Use the information icon  to view the details of the modification operation performed on the rule package.

3. Click **Close** to exit the Audit Trail dialog.

Viewing arguments associated with a rule package

Use the Associated Arguments option to view the details of arguments associated with the selected rule package. This option is available in the upper-right corner.

To view arguments associated with a rule package, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click **Associated Arguments**. A dialog with the argument details appears. The list displays the name and type of the argument parameters.
3. Click **Close** to exit the dialog.

Related topic(s)

Associating arguments with a rule package

Viewing picklist associate with a rule package

Use the associated picklists option to view the details of picklists associated with the selected rule package.

To view associated picklists, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click ellipsis icon and select **Associated Picklist**. A dialog with the picklist details appears.
   The list displays the name and type of the picklist parameters.
3. Click **Close** to exit the dialog.

Related topic(s)

Associating picklist with a rule package

Viewing rule package execution logs

Use the Execution Logs option to view the details of all the instances where a particular rule package was called (executed) previously. You can set specific time ranges to filter the calls (executions) logs as per your requirement (example: 1 day/week/month). Also, the data about the total number of calls, first call, and the last (latest) appears.

To view execution logs of a rule package, follow the below steps:

1. Select the required rule package under the **Deployed** section.
2. Click **Execution Logs**. A dialog with the list of logs appears.

If no execution log data appears, that means no call of the selected rule package is made yet.

The top menu bar of the **Execution Log** dialog provides the following details:

- **Name** of the rule package.
- Under **Executed Within Last**, select one of the following filter options.
  - **1 Day** — to view the execution logs of the last one day.

- **1 Week** — to view the execution logs of the last one week.
- **1 Month** — to view the execution logs of the last one month.
- **Executed Between** — to view the execution logs between a specific range of dates. Select From and To date ranges.

- **Total No. of Calls** — The total number of times the rule package was executed during the selected time period.
- **First Call** — Date and time when the first call was made for the selected time period.
- **Last Call** — Date and time when the last call was made for the selected time period.

The left pane displays a list of all the **Calls** made during the selected time period. The middle pane displays the **Input** values entered for the call selected in the left pane under Calls. The right pane displays the **Output** generated after the rule gets executed.

3. Click **Close** to exit the Execution Log dialog.

Related topic(s)

- Testing a rule package
- Testing a rule package deployed as a RESTservices

Viewing rule package REST endpoint data

The REST Endpoint Data is an execution feature. Use this option to test the rule package deployed as REST Endpoints. When the main rule package gets tested, the latest version of that rule package gets tested as well. The latest version as well as the other versions of a rule package can be tested individually. This option is available in the lower-right corner.
You can view all the rule package that you have deployed as REST services. Each Endpoint is the URL location from which REST services can access the resource that you want to execute. This allows you to download the Endpoint information rule package together.

You can view REST Endpoint Data of the main rule package as well as data of the other versions.

To view REST Endpoint Data of the rule package, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click **Rest End Point Data**. The Rest End Point dialog appears.
   The dialog displays information required to execute REST services, that is, URL, HTTP Method, Content type, and Request JSON.
   Use the copy icon to copy the required information.
3. Click **Close** to exit the Rest End Point dialog.

For information on how to execute the REST services, see the Appendix A: Executing Rule Builder REST services section.

Performing operations on rules

This section explains the additional operations you can perform on the rules listed within the rule packages available under the Draft and Deployed sub tabs.

Configuring rule properties

Once you create a rule, you can define some additional properties such as the effective date, expiry date, and priority. This option is available in the Draft mode only. If you want to configure the properties of a deployed rule, you must perform the checkout operation first.

A separate version of the rule package is not required to reflect changes made to the properties of the rules available within it.

To configure rule properties, follow the below steps:

1. In the **Rules** > **Draft** tab, select the required rule package. The list of rules available within it appears.
2. Select the required rule. The definition of the selected rule appears.
3. Click the ellipsis icon in the rule definition pane and select **Properties**. The Rule Properties dialog appears.
   The dialog displays the name of the rule, its status such as new, defined, approved,

and more, the date on which it was created, the name of the creator, the date on which it was modified, the name of the modifier, priority such as normal, low, maximum, and high, the date from which it is effective, and the date on which it expires (if any).
Here you can configure the effective and expiry date, and the priority of the rule.

4. In the **Effective Date** field, click the date picker icon to specify the date from which the rule comes into effect.
5. In the **Expiry Date** field, click the date picker icon to specify the date on which the rule expires or becomes ineffective.
   In case you do not want to specify an expiry date for the rule, select the **Never Expire** checkbox. The Expiry Date field appears disabled in this case.
6. From the **Priority** dropdown, select the priority of the rule as **Normal**, **Low**, **High**, or **Maximum**. This option allows you to define the priority of its execution during runtime. By default, the priority is set to Normal.
   This property is essential for cases where multiple rules need to be executed. The rule with the maximum priority gets executed first followed by its subsequent lower-value priority rules.
   When two or more rules have the same priority, they get executed in parallel.
7. Click **Save**. The changes made to the rule properties get saved.

   After making changes to the properties, send the rule for approval again. Upon receiving approval from the checker, deploy the rule package to which it belongs again.

Related topic(s)

Viewing rule properties

Analyzing a rule

The analyze rule feature allows you to determine whether or not a decision rule is applicable. The rule is checked for duplicated rows. Also, it lets you perform validation of specified ranges for the cases of impossible matches, overlapping, missing range, and boundary range. For more information, see the Examples of rules analysis section.

- The analyze rule feature cannot detect the logical correctness of a rule. Thus, ensure creating logically sound decision rule.

- The Analyze Rule feature is available in the Draft mode only.
- In case you want to analyze a deployed rule, check out the rule first and then perform its analysis.

To analyze a decision rule, follow the below steps:

1. Navigate to the **Rules** tab, and select the required rule package. A rules pane specific to the selected package appears.
   It displays the list of rules available within the package.
2. Select the required decision rule. Its definition appears on the adjacent canvas.
3. Click the ellipsis icon ⋮ present above the rule definition and select **Analyze Rule**.
   The Analyze Rule dialog appears.
4. Select the **Check Duplicate Rows** checkbox to verify the existence of duplicated rows in the decision table. Further, select one of the following:
   - **Action(s) Only** — to identify duplicate rows in the action(s) column only.
   - **Condition(s) Only** — to identify duplicate rows in the condition(s) column only.
   - **Condition(s) & Action(s)** — to identify duplicate rows in both condition(s) and action(s) columns.
5. Under **Range Validators**, select:
   - **Impossible Match** — to search for impossible condition(s) in the decision table.
   - **Overlapping** — to search for overlapping rows in the decision table.
   - **Missing Range** — to search for the missing range of values in the decision table.
   - **Boundary Range** — to search for rows whose value is outside the scope of minimum and maximum range (as specified while creating entities).
     For more information on the range validator options, see the Examples of rules analysis section.

     You can select one or more types of range validator options at once to identify gaps in specified value ranges.
6. Click **Analyze Rule**. The Analyze Rule-Result dialog appears.
   The result generates based on the options you selected in the Analyze Rule dialog.

   In case of any conflict(s), click **View Detail** to view a detailed description of the conflict(s). This option appears in the case of a conflict only, otherwise, it does not appear.
7. Click **Download** to download the details of the rule analysis to your system.

- The validation can be performed for integer, long, float, and date types of data only.
- The rule analysis result is just a type of warning for duplication and range errors. It does not impact the actual deployment and execution of the rule.
- Range validation works only if the range for the same entity is specified in two or more columns.

- Range validation works for up to three columns within a set of braces. If given without braces then consecutive columns on the same entity are considered for range validation. For more information, see the Examples of rules analysis section.
- In the case of impossible match, overlapping and missing ranges, only && operator is supported.

Examples of rules analysis

This section lists various examples of rules analysis.

- **Impossible match of specified range** — For example, a user wants to analyze a rule to detect the cases of impossible matches for the following age range.

C1 (age>____) &&	C2 (age<____)
0	10
10	20
20	10
30	40

Here, for the third row as per column 1 (C1), the age must be greater than 20, and as per column 2 (C2), it must be less than 10. This is a case of an impossible match as the specified age range is not logically correct.

- **Overlapping ranges** — For example, a user wants to analyze a rule to detect the cases of overlapping ranges for the following age range.

C1 (age>____) &&	C2(age<=____)
0	10
10	20
15	30
30	40

Here, as per the second row, the age must be between 11 and 20. And as per the third row, the age must be between 16 and 30. Therefore, for ages 15 to 20, both these conditions get validated. This is a case of overlapping ranges.

- **Missing ranges** — For example, a user wants to analyze a rule to detect the cases of missing ranges or values for the following age range.

C1 (age>____) &&	C2 (age<=____)
0	10
11	20
20	30
50	60

Here, none of the ranges covers the age value = 11. And, the age ranges 31 to 50 is missing.
This is the case of missing ranges.

- **Boundary range** — In this case, the system identifies rows where the value range is outside the boundary of the minimum and maximum range defined while creating the entity. For example, if the minimum and maximum range for an entity named Age is specified as 40-60, then the boundary range checks rows for values less than 40 and more than 60.
- **Column consideration in the cases of braces**:
  Example 1 — Consider the following age range. Here, the system considers columns, C2 and C3 for validating the range.

C1	C2	C3	C4
Employee ID=_____&&	(age>=_____&&	age<=_____)&&	Salary=_____
1	18	21	2000
2	22	25	4000

Example 2 — Consider the following age range. Here, the system considers columns, C2, C3, and C4 for validating the range.

C1	C2	C3	C4
Employee ID=_____&&	(age>=_____&&	age<=_____&&	age!=_____)
1	18	21	18

Example 3 — Consider the following age range. Here, the system considers columns, C2 and C5 to validate the age range. While columns, C2, C3, and C4 for validating the salary range.

C1	C2	C3	C4	C5
age>=_____	salary>=_____&&	salary<=_____&&	salary!=_____&&	age<=_____
20	5000	6000	7000	22

- **For the cases where all conditions have &&(And) logical operators, columns can follow any sequence of entities.**
  For example, consider the following age range. Here, the system considers columns, C1 and C3 to validate the age range.

C1	C2	C3
Age>=_____&&	Employee ID=_____&&	age<=_____
18	1	21
22	2	25

- **For the cases where conditions have a combination of &&(And) logical and ll(OR) logical operators, then the system considers the columns that have the same entity values and && operator for range validation.**
  For example, consider the following age range. Here, the system considers columns, C1 and C3 to validate the age range.

C1	C2	C3	C4
Age>_____&&	Age<=_____||	Employee ID>_____||	Employee ID<_____
18	21	1	10
22	25	10	20

Perform rule check out

Use the Checkout option to check out a deployed rule package or a just checked-in rule package. Once checked out, you can modify it, save the changes, and redeploy it. While redeploying, you can create a new version or save the changes on the same deployed version. This option is available in the upper-right corner of the Deployed subtab.

- Select the required rule package under the **Deployed** subtab.
- Click **Checkout**. The rule package gets checked out and moves to the draft section.

Copying a rule

Using the copy feature, you can create a copy of existing English and Decision rules within the same rule package. Thus, a new rule with the same attributes as the original one gets created with a new name. This feature is available in the Draft mode only. If you want to create a copy of a deployed rule, you must perform the checkout operation first.

To create a new rule using the copy feature, follow the below steps:

1. In the **Rules** > **Draft** tab, select the required rule package. The list of rules available within it appears.
2. Select the required English or Decision rule. The definition of the selected rule appears.
3. Click the ellipsis icon ⋮ in the rule definition pane and select **Copy**. The Copy Rule dialog appears.
4. In the **New Name** text box, specify the name of the newly created copy of the rule.
5. Click **Copy**. A copy of the selected rule gets created with the new name. This copy contains all the attributes of the original rule.

Creating a copy of a rule in a different rule package is not possible.

Modifying a rule

With changes in business logic, it is essential to modify the business rules. The modify a rule definition operation is available in the Draft mode only.

To modify a rule, follow the below steps:

1. In the **Rules** tab, select the required rule package in the Draft or Deployed subtab. The list of rules available within it appears.
2. Select the required rule.
   If you select a rule in Draft mode, ensure that it is not sent for approval or checked

in. In case it has been checked in, perform the checkout operation first. Similarly, for a Deployed rule, perform the checkout operation first and then modify it in the Draft tab.

3. Make the required changes to the rule definition and click **Save**. The modified rule gets saved.
4. Click **Send for Approval** to share the modified definition for review with the Checker.
   Upon receiving approval, check in the rule again in the system and deploy the rule package. The modified rule is now visible in the Deployed subtab.

Deleting a rule

To delete a rule, follow the below steps:

1. In the **Rules** > **Draft** tab, select the required rule package. The list of rules available within it appears.
2. Select the required rule. The definition of the selected rule appears.
3. Click the ellipsis icon ⋮ in the rule definition pane and select **Delete**. The Confirm Deletion dialog appears.
4. Click **Delete**. The rule gets deleted from the system.
   If you sent the rule for approval previously and it has been approved, when you click Delete, the rule deletion operation is sent for approval again with the Checker. Once the checker approves deletion, the rule gets removed from the system.

Pin or unpin a rule

Use the pin feature to pin the rule on the home tab for easier access. You can pin the rules on which you are currently working to the home tab.

To pin or unpin a rule to the home tab, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package under the **Draft** and **Deployed** subtab. The list of rules available within it appears.
2. Select the required rule. Its definition appears.

3. Click the ellipsis icon and select **Pin**. The rule gets pinned to the home tab. Alternatively, you can also pin the rule directly from the list of rules. Hover over the name of the required rule, the pin icon appears. Click it, the rule gets pinned to the Home tab.

In case the rule is already pinned to the home tab, the option to unpin appears. This Click Unpin to remove it from the home tab.

Viewing audit trail of a rule

Use the audit trail option to view the details of operations performed on a rule such as approved, checked out, creation, modification, and more.

To view the audit trail of a rule, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package under the **Draft** and **Deployed** subtab. The list of rules available within it appears.
2. Select the required rule. Its definition appears.
3. Click the ellipsis icon and select Audit Trail. The Audit Trail dialog specific to the selected rule appears. The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the rule such as modification, deletion, creation, and more.
Action by	This field displays the name of the user who has performed action on the rule.
Timestamp	This field displays the date and time when the action was performed on the rule.
Summary	This field displays a short summary of the action performed on the rule.

Use the information icon to view the details of the modification operation performed on the rule.

4. Click **Close** to exit the Audit Trail dialog.

Viewing rule details

To view the details of the rule, such as name, description, and definition of the rule, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package under the **Deployed** subtab. The list of rules available within the selected package appears.
2. Select the required rule. Its definition appears.
   Here, you can view the rule definition.
3. Click the **View** option present next to the rule name. The Rule Details dialog appears.
   Here, you can view the rule name and description details.
4. Click **Cancel** to close the dialog.

Viewing rule properties

Use the properties option to view details of a deployed rule such as rule name, status, that is enabled or disabled, the date from which the rule is effective, expiry date (if any), creation, and modification date, and the name of the creator and modifier.

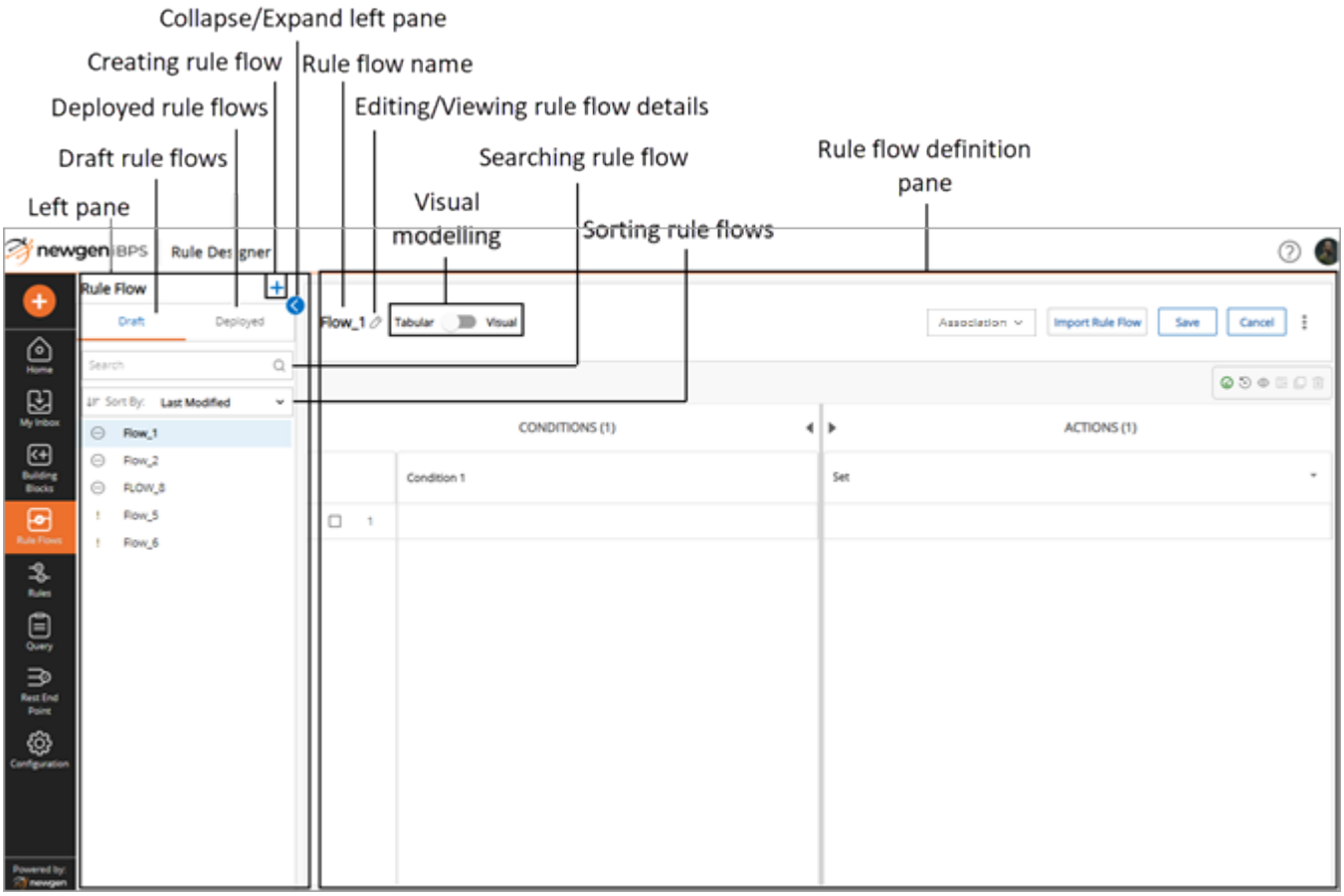
To view the properties of a rule, follow the below steps:

1. Navigate to the **Rules** tab and select the required rule package under the **Deployed** subtab. The list of rules available within the selected package appears.
2. Select the required rule. Its definition appears.
3. Click the ellipsis icon and select **Properties**. The Rule Properties dialog appears.
   Here, you can view the required rule properties.
4. Click **Cancel** to close the dialog.

Working with rule flows

Rule flow refers to the sequence in which the rules get executed. The execution of rules can be conditional or unconditional. Rule flows enable you to define the conditional calling of rules and rule packages present anywhere in the system. It also allows you to call a specific rule or rule package based on the satisfaction of applied conditions.

Using the Rule Flows tab, you can create, deploy, view, and manage rule flows. You can also import rule flows. The separation of rule flows into drafts and deployed simplifies the management of rule flows.



The Rule Flows tab contains the following UI elements:

Element	Description
Creating rule flow	Click the add icon + in the rule flows pane to create a new rule flow. This option appears in the Draft tab only.
Draft rule flows	Click **Draft** in the rule flows pane to view list the rule flows that are in draft state. This tab appears by default.
Deployed rule flows	Click **Deployed** in the left pane to view the list of deployed rule flows.
Searching rule flow	Use the search box to search the rule flow by name.
Sorting rule flow list	Use the **Sort By** dropdown to sort the list of rule flows using one of the following criteria: • **Last Modified** — displays the list of rule flows as per recent. • **Name (A-Z)** — sorts the list of rule flows in alphabetically ascending order. • **Name (Z-A)** — sorts the list of rule flows in alphabetically descending order.
Collapse or expand left pane	Click the unpin icon to collapse the left pane and maximize the rule flow definition pane. Likewise, in order to exit the maximized view, click the expand icon .
Rule flow name and description	Name and description of the rule flow (selected from the left pane) appear on top of the rule flow definition pane.
Editing or viewing rule flow details	In the case of draft rule flows, click the edit icon to modify the rule flow details. While in the case of deployed rule flows, click the View button to view the rule flow details.
Visual modelling	Use the Tabular or Visual toggle to switch between the decision table and flow chart representation of the rule flow definition respectively. Tabular definition appears by default.

Creating a rule flow

To create a rule flow, follow the below steps:

1. In the **Rule Flows** > **Draft** tab, click the add icon +.
   Alternatively, you can use the create option present in the navigation pane of to add or create a rule flow. Click create icon  , then select **Add Rule Flow**. The Rule Flow Creation dialog appears.
2. Specify the following details in the dialog:

Field	Description	
Rule Flow Alias	Specify the rule flow alias. It is an alternative name used for the rule flow.	
Rule Flow Name	Based on the rule flow alias, system a non-editable rule flow name. This is essential for integration purposes.	
Rows and Columns	**Condition Columns**	Specify the number of condition columns required in the rule flow definition.
	Action Columns	Specify the number of action columns required in the rule flow definition.
	Number of Rows	Specify the number of rows required in the rule flow definition.
Description	Specify a description of the rule flow. For more information on how to use the formatting options, see the Formatting text section.	

3. Click **Save**. The Argument Association dialog appears.

Related topic(s)

- Associating arguments with a rule flow
- Associating picklist with a rule flow
- Understanding system functions

Associating arguments with a rule flow

The Argument Association dialog appears as soon as you create the rule flow. Else, you can select the required rule flow from the list of Draft rule flows, click the **Association** dropdown present in the upper-right corner of the rule flow definition pane, and then select **Argument**.

To associate arguments with a rule flow, follow the below steps:

1. In the Select Entity Group section, select the required entity group. The list of entity classes present within the selected entity group appears in the **Select an Entity Class** section.
2. Click the add icon + to add the required entity class as a parameter.
3. Using the **Type** dropdown, specify the selected entity variable as **Condition, Action**, or **Condition/Action** parameter.

   You can add more than one entity class as arguments with the rule flow. Member variables belonging to the selected entity classes get used to define the rule flow.
4. Click **Save**. The arguments get associated with the rule flow.

When defining rule flow, you can only use the associated arguments.

Related topic(s)

- Defining a rule flow
- Viewing arguments associated with a rule flow

Removing added parameters

To remove added parameter(s), follow the below steps:

1. In the **Argument Association** dialog, select the checkbox against the required parameter(s). The Delete option enables.
2. Click **Delete** to remove the added parameter from the arguments list.
3. Click **Save** to save the changes made.

Removing the parameters used to define the conditions and actions of the rule flow invalidates it.

Associating REST API with a rule flow

NewgenONE Rule Builder can consume the REST services that are registered in the Service Catalog of the Process Designer module. You can associate any REST service with a rule flow to perform a specific operation.

For example, a bank utilizes a third-party API with rule flow to automate its customer onboarding process. The API performs real-time identity verification checks and assesses creditworthiness based on predefined rules, streamlining the account opening procedure and ensuring compliance with regulatory requirements.

To associate a REST API with a rule flow, follow the below steps:

1. In the Rule Flow pane, from the Draft tab, select the required rule flow.
2. In the right pane, click the **Association** dropdown and select the **REST API** option. The API Association dialog appears.

   Ensure that you associate the required argument before associating the REST API. This enables you to map the selected API variable with the required entity member variable and then use the mapped API variables while defining the rule flow. For procedural details, see the Associating arguments with a rule flow section.

   For the remaining steps, see step 3 onward in the Associating REST API with a rule package section.

Once the API is associated with the rule flow, you can use the API entity member variable while defining the rule flow. For procedural details, see the Defining a rule flow section.

Defining a rule flow

Rule flow definition includes the following:

- Defining conditions
- Defining actions

To define a rule flow, follow the below steps:

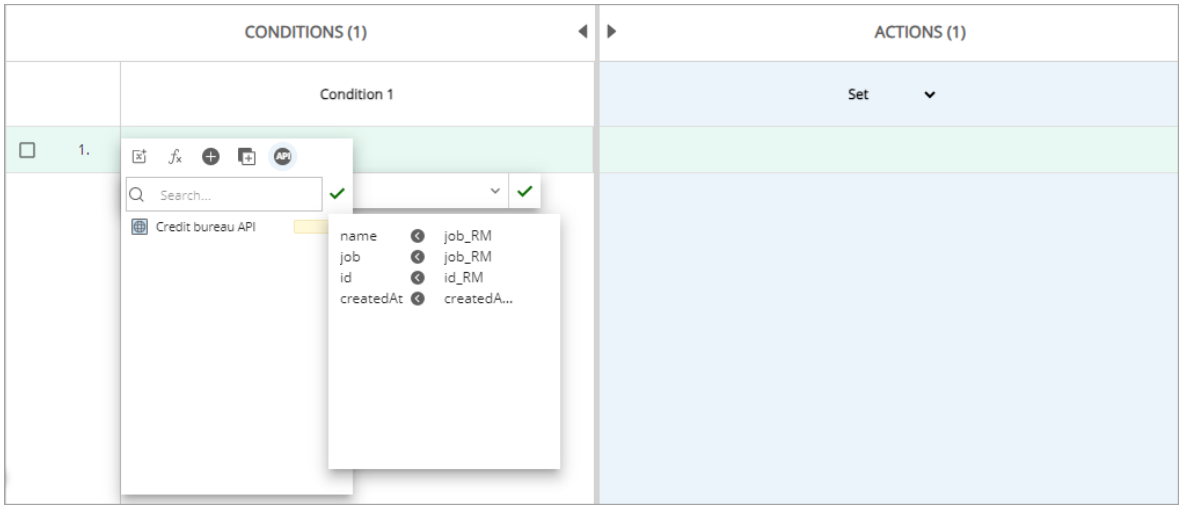
1. **Defining conditions**: Click the required row under the **CONDITIONS** column.
   - **Selecting a Field** — Click the first dropdown to select a field using one of the following options:

Option	Description
Variable List	The list of member variables defined within the entities selected during argument association appears by default. Select the required variable. Use the search box to search the variable by default.
System Function List	• Select the required system function from the available list. For more information on the available list of system functions, see the Understanding system functions section. • Select the parameter and select another system function to define the field. • Then click **Constant +** to further assign a constant value to the parameter. Or you can directly assign a constant value to the initially selected system function parameters. • Click the check icon ✔.
REST Service	This option allows you to associate a mapped REST API variable with the selected rule. On associating, the API provides the desired result when the rule is executed. For procedural details to map the REST API, see the Associating REST API with a rule flow section. To associate REST API variables with a rule condition, click here.

To associate REST API variables with a rule condition, follow the below steps:

a. Click the **REST Service** icon. The list of all the mapped APIs appears.

b. Select the required API. You can search an API using the Search box. The list of mapped API variables appears.



c. Select the required variable.

If you want to modify the forward mapping of an API variable, then follow the below sub-steps, otherwise move to the next step:

i. Hover over the required variable and click the ellipsis icon ⋮ against it. The API Association dialog appears.

ii. From the dropdown, select the required entity member variable and click **Save And Use**. The forward mapping of the selected variable is modified.

d. Click the check icon ✔.

Alternatively, you can either create a new variable (Add Variable) or associate an existing variable (Add Argument) to define the system function expression.

Option	Description
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click to add or associate a new argument and use it for defining the field. For procedural steps, see the Associating arguments with a rule flow section. • Select the variable defined within the associated argument (entities) to define the field. Alternatively, create a new variable using Add Variable option within the newly associated entity.

- **Selecting an Operator** — Click the second dropdown to select an operator. The available operators are:

Option	Description
==	Checks if the values of two operands are equal or not case-sensitive. If yes, then the condition becomes true.
>	Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.
<	Checks if the value of the left operand is less than the value of the right operand. If yes, then condition becomes true.
<=	Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then condition becomes true.
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.
!=	Excludes rows that match the criteria followed by == operator.
LIKE	Checks if the values of two operands are equal or not case-insensitive. If yes, then the condition becomes true.
IN	Use this operator to specify multiple values. It helps in reducing the need for multiple OR condition. It returns True if the expression is equal to any of the values in the IN list “IN (value1, value2, ... value_n)”. It works with all data types, except Boolean.
NOT LIKE	Excludes those rows which are matching the criterion followed by LIKE operator.
RANGE	Use this operator to specify a range for an input. For example, if the salary is between 10000 and 50000 the person is eligible for a loan. The range includes the boundary values as well.
NOT IN	In essence, it is the opposite of IN operator, where the condition satisfies if the entered value does not appear in the list. Example: NOT IN (12, 15, 20, 40), any value except these satisfies the condition and rule flow gets executed.

Availability of operators depends on the type of field selected using the first dropdown.

- **Specifying a Value** — Click the third dropdown to specify a value against the selected field using one of the following options:

Option	Description
Variable List	The list of member variables defined within the entities selected during argument association appears by default. Select the required variable. Use the search box to search the variable by default.
System Function List	• Select the required system function from the available list. For more information on the available list of system functions, see the Understanding system functions section. • Select the parameter and select another system function to define the field. • Then click **Constant +** to further assign a constant value to the parameter. Or you can directly assign a constant value to the initially selected system function parameters. • Click the check icon ✔.
Add Constant	Click to specify a constant value. Enter the required constant.
REST Service	See the REST Service option.

Alternatively, you can either create a new variable (Add Variable) or associate existing variable (Add Argument) to define the system function expression.

Option	Description
Add Variable	• Click the add icon to create a new variable within associated entities. • Select the required entity group using the **Group Alias** dropdown. • Select the required entity class using the **Entity Alias** dropdown. • Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one. • Select the data type using the **Type** dropdown. • Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**. • Specify a default value for the variable in the **Default** field. • Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable. • Specify a description of the variable in the **Description** text box. • Click **Save** to complete variable creation. It now appears as an associated entity. • Click the variable icon and select the newly created variable as a field.
Add Argument	• Click to add or associate a new argument and use it for defining the field. For procedural steps, see the Associating arguments with a rule flow section. • Select the variable defined within the associated argument (entities) to define the field. Or create a new variable using Add Variable option within the newly associated entity.

- **Selecting a Logical Operator*** — The logical operator allows you to link two rule flow conditions logically. This dropdown appears only when you have added multiple conditions columns.
  Select the required logical operator using the fourth dropdown. The available options are:
  - **AND** — It results in action execution only if both conditions are true.
  - **OR** — It results in action execution if either of the two conditions is true.
- Click the check icon ✔. The condition is defined.
- Repeat step 1 to define more conditions.

2. **Defining actions**:
   Click the dropdown under **ACTIONS** and select one of the following options:
   - **Set** — to assign new values to business variables. By default, this option is selected. For procedural steps, refer to Setting actions.
   - **Execute** — to execute rules and rule packages independently as actions. While defining rule flow, you can select to call either a single rule or a whole package using the Execute command in output actions. For procedural steps, refer to Executing rule or rule package.
   - **ExecuteAPI** — to execute the associated API upon satisfying the defined rule condition(s) and provide the desired result. After selecting the ExecuteAPI option, click the rule row and select the required API.
     If you want to modify the default API variables for forward and reverse mappings, then follow the below sub-steps:
     a. Click the rule row and hover over the required API.
     b. Click the edit map icon Map against it. The API Association dialog appears.
     c. In the Forward Mapping tab, from the dropdown, select the required entity member variable.
     d. Click the **Reverse Mapping** tab and from the dropdown, select the required entity member variable.
     e. Click **Save And Use**. The forward and reverse mapping of the selected variables are modified.

Setting actions

- Click the required row under the **Actions** column.
- **Selecting a Field** — Click the first dropdown to select a field using one of the following options:
  - **Variable List** — The list of member variables defined within the entities selected during argument association appears by default. Select the required variable.

    Use the search box to search the variable by default.
  - **Add Variable** — Click to create a new variable to define the field. For procedural steps, refer to Creating and associating new variables during rule definition.
  - **Add Argument** — Click to add or associate a new argument and use it for defining the field. For procedural steps, see the Associating arguments with rule flow section.

Select the variable defined within the associated argument (entities) to define the field. Alternatively, create a new variable using **Add Variable** option within the newly associated entity.
The operator appears selected by default.

- **Specifying a Value** — Click the third dropdown to specify a value against the selected field using one of the following options:
  - **Variable List** — The list of member variables defined within the entities selected during argument association appears by default. Select the required variable.
    You can use the search box to search the variable by default.
  - **System Function List** — Select the system function as per the selected field. For more information on this, see the Understanding system functions.
    Click the parameter and select another system function to define the field, then click **Constant +** to further assign a constant value to the parameter. Alternatively, directly assign a constant value to the initially selected system function parameters.
    Click the check icon.

    Alternatively, you can either create a new variable (Add Variable) or associate existing variable (Add Argument) to define the system function expression.

  - **Add Variable** —
    - Click the add icon to create a new variable within associated entities.
    - Select the required entity group using the **Group Alias** dropdown.
    - Select the required entity class using the **Entity Alias** dropdown.
    - Specify the name of the new member variable in the **Member Alias** text box. Click **Existing List** to view the names of existing variables and avoid conflicts while naming the new one.
    - Select the data type using the **Type** dropdown.
    - Select **Yes** in the **Array** field to create an array-type variable. Else, select **No**.
    - Specify a default value for the variable in the **Default** field.
    - Based on the data type you selected, you might be asked to specify a minimum and maximum value for the variable.
    - Specify a description of the variable in the **Description** text box.
    - Click **Save** to complete variable creation. It now appears as an associated entity.

- Click the variable icon and select the newly created variable as a field.

- **Add Argument** — Click to add or associate a new argument and use it for defining the field. For procedural steps, see the Associating arguments with rule flow section.
- Select the variable defined within the associated argument (entities) to define the field. Alternatively, create a new variable using **Add Variable** option within the newly associated entity.
- **Add Constant** — Click to specify a constant value. Enter the required constant.

- Click the check icon. The condition is defined.

Executing rule or rule package

Execute one of the following actions:

- **Execute rule or rule package** —
  - Click the required row under **Actions** column.
  - Click **Execute**. The list of available rule packages appears.
  - Select the required rule package. Alternatively, click the dropdown against the required rule package and select the rule listed within it.

    - Use the search box to search the rule package by name.
    - Use the View option to preview the rule definition.
- **Execute rule or rule package and exit** —
  - Click the required row under **Actions** column.
  - Click **Execute.** The list of available rule packages appears.
  - Select the required rule package. Or click the dropdown against the required rule package and select the rule listed within it.
  - Again, click the row and click **Exit**.
- **Exit** — Click the required row under **Actions** column and select **Exit**.
- Click **Save**. The rule flow definition gets saved.

Related topic(s)

Sending a rule flow for approval

Using rule flow definition toolbar

The rule flow definition toolbar consists of the following options:

Option	Description
Rule Preview	Use this option to view the definition of the rule flow for the selected row in a simple if-else statement format. For procedural steps, see the Previewing rule flow definition section.
forEach	Use this option to add the forEach column to the rule flow definition table. For procedural steps, see the Adding forEach column in rule flow definition section.
Always	Use this option to add the Always column to the rule flow definition table. For procedural steps, see the Adding Always column in rule flow definition section.
Copy	Use this option to create a copy of the selected row in the rule flow definition table. For procedural steps, see the Copying row in rule flow definition section.
Add Coulmn	Use this option to add columns to the right side of the selected column. For procedural steps, see the Adding columns to rule flow definition section.
Delete	Use this option to delete the selected row or column.
Up and Down	Use these options to change the order of rows, that is, move them up or down in the rule definition table. For procedural steps, see the Changing order of rows in rule flow definition section.
Add Above and Add Below	Use these options to add rows above and below the selected row. For procedural steps, see the Adding rows to rule flow definition section.

Previewing rule flow definition

The Preview option to view the definition of the rule flow for the selected row in a simple if-else statement format.

To preview the definition of the selected row for a rule flow, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under **Draft** or **Deployed** subtab.
   The definition for the selected rule flow appears.
2. Select the row whose definition you want to view.
3. Click the preview icon in the toolbar. The Rule Preview dialog appears.
   You can preview the definition for the selected row here.

Adding forEach column in rule flow definition

Use the forEach column while defining a rule flow to execute an action for all the elements added to the array attribute.
You can select an array attribute, that is, an array-type entity class corresponding to a row. The condition and action defined for the row get executed for all the elements in the array attribute.

To add forEach column in the rule flow definition table, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Select the row to which you want to apply the for each condition.
3. Click the forEach icon in the toolbar. A forEach gets added above the selected row.
4. Click the edit icon in the forEach row. The Add "For Each" Condition dialog appears.
5. Select the required entity group using the dropdown. The list of entity classes available within it appears.
   You can click **View Members** to view the list of member variables along with their type defined within available entity classes.

6. Select the required entity class and click **Add**.
7. Click **Save** to complete the addition of forEach column.

Click the delete icon in the forEach row to remove the forEach condition. The Remove For Each dialog appears. Click **Remove** to confirm.

Adding Always column in rule flow definition

Use the Always column to execute the action defined for the selected row without any condition.

To add Always column in the rule flow definition table, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Select the row for which you want to apply the always condition.
3. Click the Always icon in the toolbar. The condition for the selected row transforms to Always.
   The action present against the Always column gets executed every time you execute the rule flow.
4. Click **Save** to complete the addition of Always column.

To remove the always condition, select the required row with the Always condition and click the Always icon in the toolbar.

Copying row in rule flow definition

To copy a row in the rule flow definition, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab.
   The definition for the selected rule flow appears.
2. Select the row whose definition you copy you want to create.
3. Click the copy icon in the toolbar. A copy of the selected row gets added to the rule flow definition.
4. Click **Save** to save the copied row.

Adding columns to rule flow definition

To add more columns to the rule flow definition, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Select the required condition or action column after which you want to add another column.
3. Click the add right icon in the toolbar. A column gets added to the right of the selected column.
4. Click **Save** to save the changes made.

Changing order of rows in rule flow definition

To change the order of rows and columns in the rule flow definition, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Select the required row that you want to move upward or downward in the table.
3. Click the following in the toolbar to:
   • Up icon — move the selected row upward in the table.
   • Down icon — move the selected row downward in the table.
4. Click **Save** to save the changes made.

Adding rows to rule flow definition

To add more rows to the rule flow definition, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Select the checkbox against the row, above or below which you want to add another row.
3. Click the following in the toolbar to:
   - Add Above icon — add a row above the selected row in the table.
   - Add Below icon — add a row below the selected row in the table.
4. Click **Save** to save the changes made.

Sending a rule flow for approval

Once you create a rule flow, you can send it for approval to the supervisor user. The details of the rule flow sent for approval appear in the inbox of the checker as well as the maker. For more information on handling approval requests, see the Managing inbox section.

To send a rule flow for approval, follow the below steps:

1. Navigate to the **Rule Flows** tab.
2. Select the newly created or modified rule flow that you want to send for approval under the **Draft** subtab. The definition of the selected rule flow appears.
3. Review the rule flow definition before sending it for approval.
4. Click **Send for Approval**. The rule flow gets added to the checker's queue for approval.

You cannot delete a rule flow that is under the approval process.

Related topic(s)

- Deploying a rule flow
- Viewing rule flow REST endpoint data
- Viewing items deployed as REST services

Deploying a rule flow

Upon receiving approval from the checker for a new or modified rule flow, you can deploy a rule flow to make it available for use in the business processes.

To deploy a rule flow, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Draft** subtab. The definition of the selected rule flow appears.
2. Click **Deploy**. The Deploy RuleFlow dialog appears.
3. Select one of the following options:
   - **Increment Rule flow major version** — to increment the major version of a rule flow. For example, if the current version is numbered 1.1, then the next version gets incremented 2.0.
   - **Increment Rule flow minor version** — to increment the minor version of a rule flow. For example, if the current version is numbered 1.1, then the next version gets incremented as 1.2.
   - **Replace Rule flow version** — to replace the current version of the rule flow.

   > When you deploy a newly created rule flow, the options to increment a minor version and replace appear disabled.
4. Select Deploy as one of the following options:
   - **None** — to deploy the rule flow neither as a web nor REST service.
   - **WebService** — to deploy the rule flow as a Webservice.
   - **REST Service** — to deploy the rule flow as a REST service.
   - **Both** — to deploy the rule flow both as a web and REST service.
5. Select the **Secured** checkbox to secure the deployed rule flow from unauthorized access and add an additional layer of security.
6. Click **Deploy** to complete the deployment of the rule flow.
   The deployed rule flow appears under the Deployed subtab.

Related topic(s)

- Testing a rule flow
- Testing a rule flow deployed as a RESTservice

Testing a rule flow

Use the Test option to test the rules flow for their functional correctness. When you run a test on the main rule flow, the latest version of that rule flow gets tested. You can test the latest version as well as the other versions of a rule flow can be tested individually. You need to provide the input value, and on testing the output gets generated. If any input field is left blank, the default value is considered for that field. This option is available in the lower-right corner of the Deployed subtab.

You can perform the following type of tests:

- Testing a single request
- Bulk testing

Testing a single request

To test a single request, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click **Test**. The Test Rule Flow dialog appears.
   The entities appearing in the Test Rule Flow dialog have an accordion icon present with it. It allows you to expand and collapse categories of content. To expand entities and view their entity member variables, click the accordion icon  . Click the accordion icon  again to collapse it.
3. Enter the required Input.
4. Click **Test**. The output gets generated based on the input you entered.
   Similarly, select the required version to Test different versions of a deployed rule flow.

Use the Version Free Test option to test the deployed rule flow irrespective of its version.

Bulk testing

To perform bulk testing, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click **Test**. The Test Rule Flow dialog appears.
3. Click **Bulk Testing**. The Bulk Rule Testing dialog appears.

4. Click the **click here** link to download the spreadsheet with the correct format of data.
   The spreadsheet format gets downloaded.
5. Open the downloaded spreadsheet on your local machine and enter the **Input Values** according to the specified data types and the expected **Output Values**. Save and close the spreadsheet.
6. Click **Choose a File** and upload the spreadsheet of the rule flow.
7. Click **Test**. The results of the execution appear.
8. Click **Download** to download and view the test report.

Related topic(s)

Viewing rule flow execution logs

Testing a rule flow deployed as a REST service

The Test via REST is an execution feature. Use this feature to test the rule flow deployed as REST services. When a test is run on the main rule flow, the latest version of the rule flow gets tested as well. The latest version as well as the other versions of a rule flow also get tested individually. This option is available in the lower-right corner.

You can perform the following type of tests:

- Testing a single request
- Bulk testing

Testing a single request

To test the rule flow deployed as a REST service, follow the below steps:

1. Select the required rule flow under the Deployed subtab.
2. Click **Test via Rest**. The Test Rule Flow Via Rest dialog appears.
3. In the **Input** section, enter the required values against the fields.
4. Click **Test**.
   If you have encrypted the rule flow, you must provide the key to access it.
   The output gets generated based on the entered input.

Bulk testing

To perform bulk testing, follow the below steps:

1. Select the required rule flow under the Deployed subtab.
2. Click **Test via Rest**. The Test Rule Flow Via Rest dialog appears.
3. Click **Bulk Testing**. The Bulk Rule Testing dialog appears.
4. Click the **click here** link to download the spreadsheet with the correct format of data.
   If you have encrypted the rule flow, you must provide the key to access it.
   The spreadsheet gets downloaded.
5. Open the downloaded spreadsheet on your local machine and enter the **Input Values** according to the specified data type and the expected **Output Values**. Save and close the spreadsheet.
6. In the **Bulk Rule Flow Testing** dialog, click **Choose a File** and upload the spreadsheet of the rule flow.
7. If you have encrypted the rule flow, you must provide the key to access it.
8. Click **Test**. The results of the execution appear.
9. Click **Download** to save the test report to your local machine and view it.

Related topic(s)

Viewing rule flow execution logs

Performing operations on rule flows

This section explains the additional operations you can perform on the rule flows listed under the Draft and Deployed sub-tabs.

Associating picklist with a rule flow

You can associate picklist with a rule flow only after associating arguments with it.

To associate a picklist to a rule flow, follow the below steps:

1. Select the required rule flow under **Drafts** tab.

2. Click the **Association** dropdown, and select **Picklist** from the list. The Picklist Association dialog appears.
3. Select an **Entity Class** from the list.
   You can use the search box to search entity class by name.

   The list of entity classes displays only those classes that were selected during argument association.
4. The list of member variables defined under the selected entity class appear adjacent to the list of entity class. Select the member variable.
   You can use the search box to search the variable by name.
5. Click the add icon to add the member variables to the list.
   Use the minus icon if you want to remove any added variable .
6. Select the **Picklist** against each entity member using the dropdown.

   The picklist dropdown contains only those picklists that are of the same data type as that of the selected entity member variable. The Picklist row remains blank if there is no matching picklist. The string type function, however, can take all data types.
7. Select the default value using the **Default** dropdown.

   The default value is the value defined at the time of picklist creation. It can be Static Data, Dynamic Data or Runtime Data.

   Repeat the above steps to add and associate more entity member variables and picklists.

   To remove a picklist added to the list, select the checkbox present against the picklist name and click Delete. The selected picklist gets deleted.
8. Click **Save**. The picklist association gets completed.

Related topic(s)

Viewing picklist associated with a rule flow

Configuring rule flow properties

Once you create rule flows, you can define some additional properties such as the effective date and expiry date. This option is available in the Draft mode only. If you

want to configure the properties of a deployed rule flow, you must perform the checkout operation first.

A separate version of the rule flow is not required to reflect changes made to its properties.

To configure rule flow properties, follow the below steps:

1. In the **Rule Flows** > **Draft** tab, select the required rule flow. The rule flow definition appears.
2. Click the ellipsis icon in the upper-right corner and select **Properties**. The Rule Flow Properties dialog appears.
   The dialog displays the name of the rule flow, its status such as new, defined, approved, and more, the date on which it was created, the name of the creator, the date on which it was modified, the name of the modifier, the date from which it is effective, and the date on which it expires (if any).
   Here you can configure the effective and expiry date of the rule flow.
3. In the **Effective Date** field, click the date picker icon to specify the date from which the rule flow comes into effect.
4. In the **Expiry Date** field, click the date picker icon to specify the date on which the rule flow expires or becomes ineffective.
   In case you do not want to specify an expiry date for the rule flow, select the **Never Expire** checkbox. The Expiry Date field appears disabled in this case.
5. Click **Save**. The changes made to the rule flow properties get saved.

After making changes to the properties, send the rule flow for approval again. Upon receiving approval from the checker, deploy the rule flow again.

Related topic(s)

Viewing rule flow properties

Importing a rule flow

Using the import feature, you can create a rule flow by importing a previously defined rule flow from your system. Import the rule flow in the form of a .zip file.
The Import operation cannot be performed on those Rule Flows which are already sent for approval. Also, a Rule Flow can be imported in Design Mode only.

To create a rule flow by importing a previously defined rule flow, follow the below steps:

1. Select the required rule flow under the **Draft** tab.
2. Click **Import Rule Flow** button. The Import Rule Flow dialog appears.
3. Click **Browse** to select the rule flow .zip file from your system.
4. Select **Deploy Associated Rule Package As** one of the following:
   - **None** (neither as a WebService nor REST Service)
   - **WebService**
   - **REST Service**
   - **Both** (both as a WebService and REST Service)
5. Click **Import**. The rule flow gets imported.

When importing a rule flow, if any conflicting alias names or data types are detected, the Import Rule Flow Conflict(s) dialog will appear. It displays the total number and list of the data type warnings and alias name conflicts.

Related topic(s)

Exporting a rule flow

Copying a rule flow

Using the copy feature, you can create a new rule flow by copying an existing one. The feature is available both for drafts and deployed rule flows.

To create a new rule flow using the copy feature, follow the below steps:

1. For a draft rule flow:
   - Select the required rule flow under **Draft** subtab.
   - Click the ellipsis icon present in the rule definition pane and select **Copy**.

   For a deployed rule flow:
   - Select the required rule flow under **Deployed** subtab.
   - Click the Copy button in the rule definition tab.

   The Copy RuleFlow dialog appears.
2. Specify the **Rule Flow Alias**.

   The **Rule Flow Name** generates based on the specified rule flow alias.

   - The newly created rule flow includes all the attributes of the original rule flow.
   - The rule flow created using a deployed rule flow appears under Draft tab.
3. Click **Copy**. A new copy of the rule flow gets created.

Modify a rule flow

The modify a rule flow definition operation is available in the Draft mode only.

To modify a rule flow, follow the below steps:

1. In the **Rule Flows** tab, select the required rule flow in the Draft or Deployed subtab. The rule flow definition appears.
   If you select a rule flow in Draft mode, ensure that it is not sent for approval.
   For a Deployed rule flow, perform the checkout operation first and then modify it in the Draft tab.
2. Make the required changes to the rule flow definition and click **Save**. The modified rule flow gets saved.
3. Click **Send for Approval** to share the modified definition for review with the Checker.
   Upon receiving approval, deploy the rule flow. The modified rule flow is now visible in the Deployed subtab.

Deleting a rule flow

Use the Delete option to delete the currently opened version of the rule flow.

To delete rule flow, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click ellipsis icon and select **Delete**. The Confirm Deletion dialog appears.
3. Click **Delete**. The currently selected version of the rule flow gets deleted.

Encrypting a rule flow

Encryption feature allows the user to prevent unauthorized access to rule flows. If you apply encryption then the rule flow gets locked using a key. The system asks for the key when anyone tries to access the encrypted rule flow. The rule flow definition appears only after entering the correct key.

Points to remember:

- You must enter the encryption key to open an encrypted rule flow searched using Query.
- If you export an encrypted rule flow and import it on a different cabinet, it gets imported successfully, but to open rule flow, you must enter the encryption key.
- Checker does not require any key to open an encrypted rule flow from the Checker queue. However, the key must be entered to open a rule flow from the rule repository.

Encryption key policy:

- The key must be a combination of alphanumeric and special characters.
- The minimum and maximum length of characters for the key is 8 and 25, respectively.
- There must be at least one alphabet, one numeric value, and one special character present in the key.
- Spaces are considered as special characters.

To encrypt a rule flow listed within it, follow the below steps:

1. Select the required rule flow under the **Draft** tab.
2. Click the ellipsis icon and select **Encrypt**. The Securing dialog appears.
3. In the **Enter Key** text box, specify the encryption key as per Encryption Key Policy.
4. In the **Confirm Key** text box, re-enter the key to confirm.
5. Click **Save**. The rule flow gets encrypted.

Decrypting a rule package

To decrypt a rule flow, follow the below steps:

1. Select the required rule flow that you want to decrypt under the **Draft** subtab.
2. Enter the encryption key to open the rule flow definition.
3. Click ellipsis icon and select **Decrypt**. A dialog to confirm decryption appears.
4. Click **Decrypt**. The rule flow gets decrypted.

Exporting a rule flow

Use the Export option to download and save the selected rule flow to your machine.

To export rule flow, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click the ellipsis icon and select **Export**. The Export Ruleflow dialog appears.
3. Click **Export**. The rule flow gets downloaded as a .zip file in the local memory space of the client machine.

Related topic(s)

Importing a rule flow

Downloading model of a rule flow

The Download Model feature allows you to define or create entities (by importing) with the same properties as that of the downloaded rule flow.

To download the model, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click ellipsis icon and select **Download Model.**
3. Select the version for which you want to download the model.
4. Click **Download**. The Model gets downloaded as a .jar file. The downloaded file follows the naming convention as: Rule Flow: ruleflowname_version.jar.

Downloading report of a rule flow

Whenever you deploy a rule flow whether new or modified, different versions of the same get created. The Report feature allows you to generate information about the different versions of a rule flow. The generated report gets downloaded as a .zip file containing a consolidated Report.html file and a Help_File folder containing HTML files of rule flow.

You can download the report in the following ways:

- Normal — The report downloaded in this mode can be viewed by anyone as it does not require any password to open it.
- Password Protected — This feature prevents unauthorized access of a report. A user-generated key must be provided at the time of downloading and opening the downloaded report.

Download normal report

To download a normal report, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click the ellipsis icon and select **Report**.
3. Select **Normal**.
4. Select the report format for downloading, **Word Doc** or **HTML**.
   The report gets downloaded as a .zip file.

Depending on the type of browser you are using, you may be asked to save the file.

Download password protected report

To download password protected report, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.
2. Click ellipsis icon and select **Report**.
3. Select **Password Protected**.

4. Select the report type that you want to download, **Word Doc** or **HTML**. The Set Password Key dialog appears.
5. Specify a password Key.

   System asks you to enter the password key when you extract files from the downloaded .zip file.
6. Click **Download**. The report gets downloaded as a .zip file.

   Depending on the type of browser you are using, you may be asked to save the file.

Opening password protected report

To open the downloaded report, follow the below steps:

1. Navigate to the folder where the report is downloaded.
2. Select the method of extracting the files from the downloaded .zip file. The system asks to provide the password.
3. Enter the same password key.
4. Click **OK**. The files get extracted in a folder.

Managing versions of a rule flow

Upon opening the definition of a rule flow in the Deployed subtab, the version of the rule flow appears alongside its name. When you deploy a rule flow for the first time, version 1.0 is assigned to it.

When you redeploy a rule flow in event of modifications to the rule flow definition, multiple major and minor version gets created.
For example, for a rule flow named HomeLoanEligibiltiy the current version is 1.0. When a developer makes modifications to the rule flow logic, he or she can redeploy the rule flow as a minor or major version. If the rule flow gets redeployed as a minor version, the version number changes to 1.1. While if it gets redeployed as a major version, the version number changes to 2.0.

The term Enabled or Disabled represents whether the current version of the rule flow is available to external applications for use or not.

Click the version dropdown to view the following options:

- **Enable All** — to make all the versions of the rule flow available to external applications.
    - Click **Enable All**. All the version of the rule flow gets enabled.
- **Disable All** — to make all the versions of the rule flow unavailable to external applications.
    - Click **Disable All**. The Confirm Disabling dialog appears.
    - Click **Disable**. All the version of the rule flow gets disabled.
- Click **Enabled** or **Disabled** to view the following option for a specific version:
    - **Set As Latest** — to tag a lower-order rule flow version as the latest in the database table.
    - **Enable or Disable** — to enable or disable the selected version.

Pin or unpin a rule flow

The pin feature makes the rule flow appear on the Home tab upfront making access easier.

To pin or unpin a rule, follow the below steps:

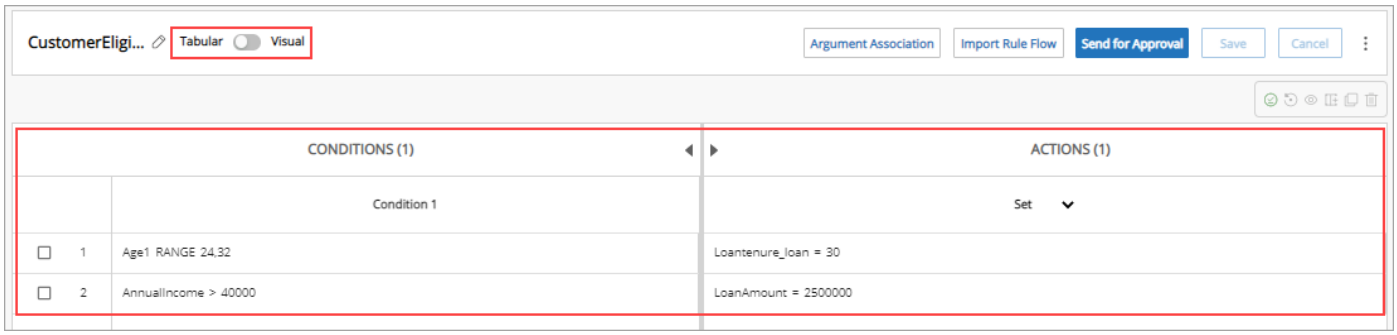
1. Select the required rule under the **Deployed** subtab.
2. Click ellipsis icon present in the upper-right corner.
3. Click:
    - **Pin** — Pins the rule to the Home tab. This option appears when the rule flow is not already pinned.
    - **Unpin** — Unpins the rule from the Home tab. This option appears when the rule flow is already pinned.

You can also pin the rule flow from the list of the rule flow. Hover over the name of the rule flow, the pin icon appears. Click it, the rule flow gets pinned to the Home tab.

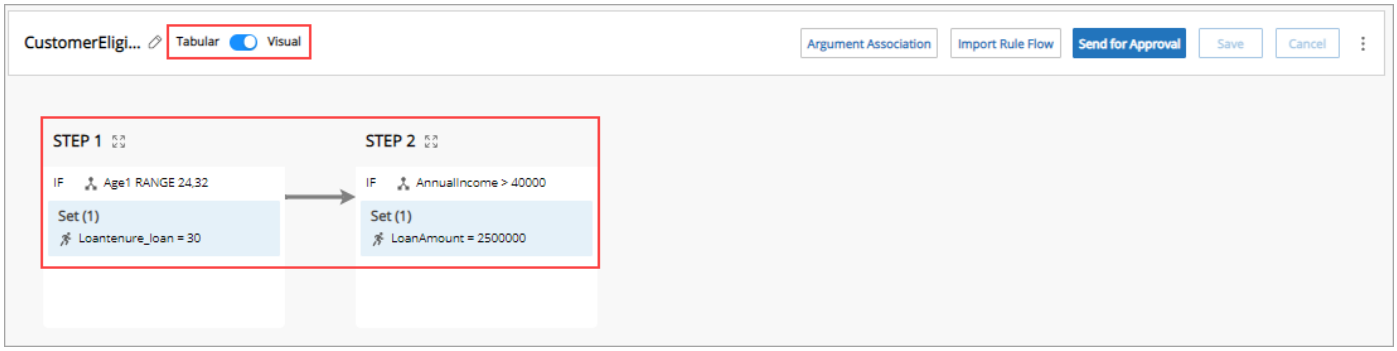
Viewing visual model of a rule flow definition

With the Tabular or Visual toggle, you can view the visual representation of a rule flow model. It makes it easier for you to understand the configured rules. In comparison to textual or tabular representations, visual representations offer easier cognitive understanding. A flow chart (or diagram) allows you to track the order of execution of rules and highlight any dependencies.

Tabular representation



Visual model



Viewing audit trail of a rule flow

Use the audit trail option to view the details of operations performed on the rule flow such as approved, checked out, creation, modification, and more.

To view audit trail of a rule flow, follow the below steps:

1. Select the required rule flow under the **Draft** or **Deployed** subtab.
2. Click the ellipsis icon and select **Audit Trail**. The Audit Trail for the selected rule flow appears. The audit trail dialog displays the following details:

Field	Description
Action	This field displays the action performed on the rule flow such as modification, deletion, creation, and more.
Action by	This field displays the name of the user who has performed action on the rule flow.
Timestamp	This field displays the date and time when the action was performed on the rule flow.
Summary	This field displays a short summary of the action performed on the rule flow.

Use the information icon to view the details of the modification operation performed on the rule flow.

3. Click **Close** to exit the Audit Trail dialog.

Viewing arguments associated with a rule package

Use the Associated Arguments option to view the details of arguments associated with the selected rule package. This option is available in the upper-right corner.

To view arguments associated with a rule package, follow the below steps:

1. Select the required rule package under the **Deployed** subtab.

2. Click **Associated Arguments**. A dialog with the argument details appears.
   The list displays the name and type of the argument parameters.
3. Click **Close** to exit the dialog.

Related topic(s)

Associating arguments with a rule package

Viewing picklists associated with a rule flow

Use the associated picklists option to view the details of picklists associated with the selected rule flow.

To view associated picklists, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click ellipsis icon and select **Associated Picklist**. A dialog with the picklist details appears.
   The list displays the name and type of the picklist parameters.
3. Click **Close** to exit the dialog.

Related topic(s)

Associating picklist with a rule flow

Viewing rule flow properties

Use the properties option to view details of a deployed rule flow such as rule flow name, status, that is enabled or disabled, the date from which the rule flow is effective, expiry date (if any), creation, and modification date, and the name of the creator and modifier.

To view the properties of a rule flow, follow the below steps:

1. Navigate to the **Rule Flows** tab and select the required rule flow under the **Deployed** subtab.

2. Click the ellipsis icon and select **Properties**. The Rule Flow Properties dialog appears.
   Here, you can view the required rule flow properties.
3. Click **Cancel** to close the dialog.

Related topic(s)

Configuring rule flow properties

Viewing rule flow execution logs

Use the Execution Logs option to view the details of all the instances where a particular rule flow was called (executed) previously. You can set specific time ranges to filter the calls (executions) logs as per your requirement (example: 1 day/week/month). Also, the data about the total no. of calls, first call, and the last (latest) appears. This option is available in the lower-right corner.

To view execution logs of a rule flow, follow the below steps:

1. Select the required rule flow under the **Deployed** section.
2. Click **Execution Logs**. A dialog with the list of logs appears.

   If no execution log data appears, that means no call of the selected rule flow is made yet.

   The top menu bar of the **Execution Log** dialog provides the following details:

   - **Name** of the rule flow.
   - Under **Executed Within Last**, select one of the following filter options.
     - **1 Day** — to view the execution logs of the last one day.
     - **1 Week** — to view the execution logs of the last one week.
     - **1 Month** — to view the execution logs of the last one month.
     - **Executed Between** — to view the execution logs between a specific range of dates. Select From and To date ranges.
   - **Total No. of Calls** — The total number of times the rule flow was executed during the selected time period.
   - **First Call** — Date and time when the first call was made for the selected time period.
   - **Last Call** — Date and time when the last call was made for the selected time period.

The left pane displays a list of all the **Calls** made during the selected time period. The middle pane displays the **Input** values entered for the call selected in the left pane under Calls. The right pane displays the **Output** generated after the rule gets executed.

3. Click **Close** to exit the Execution Log dialog.

Viewing rule flow REST endpoint data

The REST Endpoint Data is an execution feature. Use this option to test the rule flow deployed as REST Endpoints. When the main rule flow gets tested, the latest version of that rule flow gets tested as well. The latest version as well as the other versions of a rule flow can be tested individually. This option is available in the lower-right corner. You can view all the rule flow that you have deployed as REST services. Each Endpoint is the URL location from which REST services can access the resource that you want to execute. This allows you to download the Endpoint information rule flow together. You can view REST Endpoint Data of the main rule flow as well as data of the other versions.

To view REST Endpoint Data of the rule flow, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click **Rest End Point Data**. The Rest End Point dialog appears.
   The dialog displays information required to execute REST services, that is, URL, HTTP Method, Content type, and Request JSON.
   Use the copy icon  to copy the required information.
3. Click **Close** to exit the Rest End Point dialog.

For more information on how to execute the REST services, see Appendix A: Executing Rule Builder REST services section.

Managing inbox

This chapter provides information on the My Inbox tab. After you create a rule or rule flow, you can send it for approval. Rules and rule flow sent for approval can be reviewed for their usability and then approved or rejected accordingly. A reviewing authority (supervisor) must sign in to Rule Builder to approve or reject a rule or rule flow. When you create or modify a rule or rule flow definition, it must be approved by another user before it gets accepted and saved in the system.

The tab allows you to view the following:

- Pending for my action — The tab lists the approval requests you have received from other users.
- To be actioned by others — The tab lists the approval requests you initiated when creating or modifying a rule or rule flow.

The approval requests visible to you depends on the rights assigned to you by the supervisor.

Reviewing items pending for my action

The **Pending for my action** tab appears by default. It lists the items waiting for your review. You can reject or approve the item directly from here. The tab segregates the available items under the following:

- **Decision/English Rule** — This subtab appears by default. It displays the list of rules, either newly created or modified, waiting for your review.
  The list provides the following information about the rule:
  - Name of the rule — Click it to preview the rule definition.
  - Name of the rule package to which it belongs
  - Type of action performed on the rule, that is, created or modified
  - When was the rule last modified
  - Maker's name
  - Audit Trail — Click the audit icon to view the logs of actions performed on the rule.

- **Rule Flow** — Click **Rule Flow** to view the list of rules flows either newly created or modified, waiting for your review.
  - Name of the rule flow — Click it to preview the rule flow definition.
  - Type of action performed on the rule flow, that is, created or modified
  - When was the rule flow last modified
  - Maker's name
  - Audit Trail — Click the audit icon to view the logs of actions performed on the rule flow.

Use the **Sort By** option to sort the list of rules or rule flows using one of the following options:

- **Last Modified** — Sorts the list in descending order of modification dates.
- **Name: A-Z** — Sorts the list in alphabetically ascending order.
- **Name: Z-A** — Sorts the list in alphabetically descending order.

Approving or rejecting an item

To approve or reject an item, follow the below steps:

1. Navigate to **My Inbox** and select the name of the required rule or rule flow under the **Pending for my actions** tab.
2. Click the name of the rule to preview the rule or rule flow definition.
3. Review the rule or rule flow.
4. Click one of the following options:
   - **Approve** — to approve the rule or rule flow. The Comments dialog appears.
     - Specify **Comments** for approval and click **Save**.
   - **Reject** — to reject the rule or rule flow. The Comments dialog appears.
     - Specify **Comments** for rejection and click **Save**.

   You can also approve or reject a rule or rule flow using the respective options present upfront with the item description in the list.

Viewing items pending for action by others

Click **To be actioned by others** to open the related tab. It lists the items you have sent for approval. You can keep a track of the items and dismiss approval requests for the items if required. The tab segregates the available items under the following:

- **Decision/English Rule** — This subtab appears by default and displays the list of newly created or modified rules that you have sent for approval.
  The list provides the following information about the rule:
  - Name of the rule — Click it to preview the rule definition.
  - Name of the rule package to which it belongs.
  - Type of action performed on the rule, that is, created or modified.
  - When was the rule last modified.
  - Maker's name
  - Audit Trail — Click the audit icon to view the logs of actions performed on the rule.
- **Rule Flow** — Click **Rule Flow** to view the list of rules flows newly created and modified rules that you have sent for approval.
  - Name of the rule flow — Click it to preview the rule flow definition.
  - Type of action performed on the rule flow, that is, created or modified
  - When was the rule flow last modified
  - Maker's name
  - Audit Trail — Click the audit icon to view the logs of actions performed on the rule flow.

Dismissing approval requests

To dismiss an item and remove it from the checker's queue, follow the below steps:

1. Navigate to **My Inbox** and click **To be actioned by others**.
2. Select the name of the required rule or rule flow.
3. Click the name of the rule to preview the rule or rule flow definition.
4. Click **Dismiss** to remove the approval request from the checker's queue. The Comments dialog appears.

5. Specify your comments for dismissing the request and click **Save**.

You can dismiss multiple requests at once by selecting the checkboxes against the required items and then clicking the **Dismiss** option appearing in the upper-right corner.

Running search query

Business organizations need their policymakers to work directly on the business processes to implement policies, change or delete policies, or add new policies. It is often required that the policymakers have a minimum dependency on technical support (implementation engineer) to find out the effective rule, change the rule, and redeploy it.

NewgenONE Rule Builder provides policymakers with the ease of implementing policy changes on their own.
The query tab allows you to search rules, rule flows, and entities based on some criteria such as rule package name, rule name, rule flow name, and more.

To search rules or rule flows based on some search criteria, follow the below steps:

1. Select **Search by** as:

Option	Description
Name	Allows you to perform a search by name of the entity, rule package, rule, or rule flow. For more information, refer to Searching by Name.
Date	Allows you to perform a search by date of creation, expiry, and more. For more information, refer to Searching by Date.

Searching by Name

- Select **Search in** as:

Option	Description
All	Allows you to perform a search both on rule packages and rule flows. For more information, refer to Searching in All.
Rule Package	Allows you to perform a search on rule packages. For more information, refer to Searching in Rule Package.
Rule Flow	Allows you to perform a search on rule flows.. For more information, refer to Searching in Rule Flow.

- Select **Search in** as:

Option	Description
Deployed	Allows you to perform a search on deployed rule packages, rule flows, or both.
Draft	Allows you to perform a search on draft rule packages, rule flows, or both. This option appears selected by default.

Use this option to make your search more specific.

Searching in All

- Perform the search using one of the following options or a combination of options:

Option	Description
Contains Entity	Use this option to perform a search rule or rule flow by an entity defined within it. 1. Click the ellipsis icon. The Select Entity Member dialog appears. 2. Select the required entity group using the dropdown. 3. Select the required entity class defined within the selected entity group. 4. Select the required entity member from the available list defined within the selected entity class. 5. Click **Done** to confirm the selections.
Created By	Use this option to perform a search by the name of the creator of the rule or rule flow. Use the search box to search the creator by name.
Status	Use this option to perform a search based on the status of a rule or rule flow, such as, defined, approved, rejected, and more.

- Click **Search**. The search result appears on the canvas below the menu bar.

Searching in Rule Package

- Perform the search using one of the following options or a combination of options:

Option	Description
Rule Package	Select the required rule package using the dropdown. Use the search box to search the rule package by name.

Option	Description
Rule Name	Select the required rule defined within the selected rule package using the dropdown. Use the search box to search the rule by name.
Contains Entity	Use this option to perform a search rule or rule flow by an entity defined within it. 1. Click the ellipsis icon  . The Select Entity Member dialog appears. 2. Select the required entity group using the dropdown. 3. Select the required entity class defined within the selected entity group. 4. Select the required entity member from the available list defined within the selected entity class. 5. Click **Done** to confirm the selections.
Created By	Use this option to perform a search by the name of the creator of the rule. Use the search box to search the creator by name.
Status	Use this option to perform a search based on the status of rules such as, defined, approved, rejected, and more.

- Click **Search**. The search result appears on the canvas below the menu bar.

Searching in Rule Flow

- Perform the search using one of the following options or a combination of options:

Option	Description
Rule Package	Select the required rule flow using the dropdown. Use the search box to search the rule flow by name.
Contains Entity	Use this option to perform a search rule or rule flow by an entity defined within it. 1. Click the ellipsis icon  . The Select Entity Member dialog appears. 2. Select the required entity group using the dropdown. 3. Select the required entity class defined within the selected entity group. 4. Select the required entity member from the available list defined within the selected entity class. 5. Click **Done** to confirm the selections.
Created By	Use this option to perform a search by the name of the creator of the rule flow. Use the Search box to search the creator by name.
Status	Use this option to perform a search by the status of the rule flow, such as, defined, approved, rejected, and more.

- Click **Search**. The search result appears.

Searching by Date

- Perform the search using one of the following options or a combination of options:

Option	Description	
Date Range Type	Created between	Use this option to search rules or rule flows created between a certain time range or period.
	Expiry between	Use this option to search rules or rule flows expired between a certain time range or period.
	Effective between	Use this option to search rules or rule flows that are effective between a certain time range or period.
	Last Edited between	Use this option to search rules or rule flows edited or modified between a certain time range or period.
Date Range Type	Last 7	Use this option to search rules or rule flows created, expired, effective, or edited in the last 7 days.
	Last 15	Use this option to search rules or rule flows created, expired, effective, or edited in the last 15 days.
	Last 30	Use this option to search rules or rule flows created, expired, effective, or edited in the last 30 days.
	Custom	Use this option to search rules or rule flows created, expired, effective, or edited in the selected range of dates. Select **From Date** to **To Date** to perform the search.

- Select **Search in** as:

Option	Description
Deployed	Use this option to perform a search among deployed rule packages, rule flows, or both.
Draft	Use this option to perform a search among draft rule packages, rule flows, or both. This option appears selected by default.

Use this option to make your search more specific.

2. Click **Search**. The search result appears. Click the rule or rule flow to open its preview.
   Hover over the name of the rule or rule flow, the option to Checkout the rule

appears. Use this option to check out a rule or rule flow, such that, you can modify it and send for approval again.

Viewing items deployed as RESTservices

The REST endpoint tab allows you to view all the rule flows and rules packages deployed as REST services. Each endpoint is the URL location from which REST services can access the resource that you want to execute. This tab allows you to download the endpoint information for rule flows and rule packages together. The downloaded information can then be used to integrate the business logic externally (on client side) and execute associated services.

To download REST endpoints, follow the below steps:

1. Select:

Option	Description
Rule Package	Use this option to view all the rule packages deployed as REST services.
Rule Flow	Use this option to view all the rule flows deployed as REST services.

   The list of rule packages or rule flows deployed as REST services appears.

2. Select the required rule package or rule flow.
   Use the **Rule Package Name** or **Rule Flow Name** checkbox to select all the rule packages and rule flows respectively, available in the list at once.
3. Click **Version number** appearing against the rule package or rule flow name. The End Point Data dialog appears. It displays the list of versions of the particular rule package or rule flow.
   - Click **Version number** again to view REST endpoint details such as URL, HTTP Method, Content type, and Request JSON. You can copy these details using the Copy icon  . Click **Close** to exit the dialog.
   - Click the **Download** icon  to save the REST service endpoint .zip file to your system.
4. Click the details icon  to view and copy REST endpoint details as described in the previous step.

5. Click:
   - **Download All** — to download all the REST service endpoint information. This option gets disabled when you select one or more rule packages or rule flows.
   - **Download Selected** —to download the selected REST service endpoint information. This option appears disabled when you have not selected any rule package or rule flow.
     The API_Endpoint_Report.zip gets downloaded containing all the selected rulesets.

Performing Rule Builder configurations

The Configurations tab allows you to configure the following type of settings:

- General
- Decision Table

General —

Refer to the below table to configure general settings:

Key	Description
HttpOnlyCookie	Turn on the **HttpOnlyCookie** toggle to prevent scripts from accessing the protected cookie or data. This helps in ensuring data security.
Send Request In SOAP Response	Turn on the **Send Request In SOAP Response** toggle to allow sending of the specified payload (content from XML) as response to the sender of the request. By default, SOAP service configuration is disabled. To enable the SOAP service configuration, refer to the *Automation Studio* section of the *NewgenONE Configuration Guide*.
Auto Catalog Registration	Turn on the **Auto Catalog Registration** toggle to allow automatic registration of web service.
Wait time for axis2 restart	Specify the time (in minutes) after which the Axis2 restarts on the server.
Delimiter for array element	Specify delimiter for array-type member variables to hold multiple values.

Decision Table Settings —

Refer to the below table to configure decision table settings:

Key	Description
Function Search	Turn on the **Function Search** toggle to enable function search in Decision Table.
Batch Size of Decision rule	Specify the number of rows a single page of the decision table must contain. The minimum value can be 1 and the maximum value can be 100.

Click **Save** to save the changes made to the configurations.

Use the Set to default to restore the default values and options for the fields.

Related topic(s)

Defining a Decision rule

Understanding system functions

This chapter describes the system functions used to define condition and action expressions for rules and rule flows in the NewgenONE Rule Builder.

The chapter includes the following topics:

Array functions

The following is the list of array functions:

- **get (intArray1, int2)** — This function returns an integer value of the integer array present at the specified input integer2 index. getarray function returns a -1 value if the index number is given out of size in an array.
- **get (longArray, int1)** — This function returns a long value of the long array present at the specified input integer1 index. getarray function returns a -1 value if the index number is given out of size in an array.
- **get (floatArray, int2)** — This function returns a float value of the float array present at the specified input integer2 index. getarray function returns a -1 value if the index number is given out of size in an array.
- **get (DateArray, int1)** — This function returns a date value of the date array present at the specified input integer1 index. getarray function returns a -1 value if the index number is given out of size in an array.
- **get (StringArray, int2)** — This function returns a string value of the string array present at the specified input integer2 index. getarray function returns a -1 value if the index number is given out of size in an array.
- **indexOf(intArray, int)** — This function returns an integer index at which the first input integer argument is present in the integer array. If the input integer is not found in the integer array, this function returns -1.
- **indexOf(long, longArray)** — This function returns an integer index at which the first input long argument is present in the long array. If the input long is not found in the long array, this function returns -1.

- **indexOf(floatArray, float)** — This function returns an integer index at which the first input float argument is present in the float array. If the input float is not found in the float array, this function returns -1.
- **indexOf(Date, DateArray)** — This function returns an integer index at which the first input date argument is present in the date array. If the input date is not found in the date array, this function returns -1.
- **indexOf(StringArray, String)** — This function returns an integer index at which the first input string argument is present in the string array. If the input string is not found in the string array, this function returns -1.
- **isEmpty(intArray)** — This function returns the Boolean value as 1 if input integer array is empty ,else returns 0.
- **isEmpty(longArray)** — This function returns the Boolean value as 1 if input long array is empty ,else returns 0.
- **isEmpty(floatArray)** — This function returns the Boolean value as 1 if input float array is empty ,else returns 0.
- **isEmpty(DateArray)** — This function returns the Boolean value as 1 if input date array is empty ,else returns 0.
- **isEmpty(StringArray)** — This function returns the Boolean value as 1 if input string array is empty ,else returns 0.

  During execution of isEmpty function, the Soap UI requires no input tags for the array[ ] parameter, that is, <floatarr></floatarr> must not be present in the input XML if isempty is being checked on floatarr.

- **lastIndexOf(int, intArray)** — This function returns an integer index at which the last input integer argument is found in the integer array. If the input integer argument is not found in the integer array, this function returns -1.
- **lastIndexOf(longArray, long)** — This function returns an integer index at which the last input long argument is found in the long array. If the input long argument is not found in the long array, this function returns -1.
- **lastIndexOf(float, floatArray)** — This function returns an integer index at which the last input float argument is found in the float array. If the input float argument is not found in the float array, this function returns -1.
- **lastIndexOf(DateArray, Date)** — This function returns an integer index at which the last input date argument is found in the date array. If the input date argument is not found in the date array, this function returns -1.
- **lastIndexOf(String, StringArray)** — This function returns an integer index at which the last input string argument is found in the string array. If the input string argument is not found in the string array, this function returns -1.

- **set(int1, int2, intArray)** — This function returns the Boolean value by setting the input integer2 at the input integer1 index in the input integer array. If the function successfully sets the required integer value at the specified index in the integer array it returns true otherwise false.
- **set(longArray, long, int)** — This function returns the Boolean value by setting the input long argument at the input integer index in the input long array. If the function successfully sets the required long value at the specified index in the long array, it returns true otherwise false.
- **set(int, float, floatArray)** — This function returns the Boolean value by setting the input float argument at the input integer index in the input float array. If the function successfully sets the required float value at the specified index in the float array, it returns true otherwise false.
- **set(DateArray, Date, int)** — This function returns the Boolean value by setting the input date at the input integer index in the input date array. If the function successfully sets the required date value at the specified index in the date array, it returns true otherwise false.
- **set(int, String, StringArray)** — This function returns the Boolean value by setting the input string at the input integer index in the input string array. If the function successfully sets the required string value at the specified index in the string array, it returns true otherwise false.
- **size(intArray)** — This function returns the size of the input integer array in the output integer argument.
- **size(longArray)** — This function returns the size of the input long array in the output integer argument.
- **size(floatArray)** — This function returns the size of the input float array in the output integer argument.
- **size(DateArray)** — This function returns the size of the input date array in the output integer argument.
- **size(StringArray)** — This function returns the size of the input string array in the output integer argument.
- **lookUpArrayIntValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple integer type values.
- **lookUpArrayLongValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple long type values.

- **lookUpArrayFloatValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple float type values.
- **lookUpArrayDateValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple date type values.
- **lookUpArrayBigDecimalValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple big decimal type values.
- **lookUpArrayStringValuesFromDB** — Based on the lookup parameters provided in the runtime query, this function returns the output as an array of multiple string type values.

Additional array functions

The following additional functions are available in the NewgenONE Rule Builder:

- **Average of Array Elements**: Determines the average of all the values in an array object.
  - Syntax — Array.average(input_array)
  - Syntax — Array.average(input_array, startindex, endindex)
  - Result — Returns the average of all values in the array
  - Applicable Data Type — Integer, Long, Float, BigDecimal
- **Concatenate Array Elements**: The entire set of elements inside the array are concatenated into a single string.
  - Syntax — output array = Concatenate (array1, array2, separator)
  - Syntax — output array = Concatenate (array1, array2, separator, startindex, endindex)
  - Separator is optional. If omitted, the elements will get concatenated without space.
  - Result — Concatenate the elements of thee array into a single value/string
  - Data Type — Text
- **Contains()**: Determines whether the specified object exists as an element in an array object.
  - Syntax — Array.contains(input_array, item)
  - Result — True if the specified object exists as an element in the array; otherwise false
  - Applicable Data Types — Text, Date, Integer, Long, Float, BigDecimal
  - Syntax — Array.contains (input_array, item, startIndex, endIndex)

- **Max and Min Functions for Array Elements**: Determines the maximum and minimum values found in an array.
    - Syntax — Max(input_array) & Min(input_array)
    - Syntax — Max(input_array, startindex, endindex) & Min(input_array, startindex, endindex)
    - Result — Return the highest/lowest value
    - Data Type — Integer, Long, Float, BigDecimal, Date
- **Sum of Array Elements**: Determines the sum of all the values in an array object.
    - Syntax — Array.sum (input_array)
    - Syntax — Array.sum (input_array, startindex, endindex)
    - Result — Returns the sum of all values in the array
    - Applicable Data Type — Integer, Long, Float, BigDecimal

System functions

The following is the list of system functions:

- **absolute(int)** — This function returns an integer value for the absolute value of the input integer argument.
- **absolute(float)** — This function returns afloat value for the absolute value of the input float argument.
- **absolute(long)** — This function returns a long value for the absolute value of the input long argument.
- **booleanValue(String)** — This function returns a Boolean value of the input string argument.
- **booleanValue(int)** — This function returns a Boolean value of the input integer argument.
- **caseContains(String1, String2)** — This function is used for matching values to be searched for a single word.
- **ceiling(float)** — This function returns the smallest integer value that is greater than or equal to the input float argument.
- **concat(String1, String2)** — This function returns a string value by adding input string 2 after input string 1.
- **contains(String, String)** — This function returns a Boolean value by checking whether the first string contains the second string. This function works with case-sensitive strings only.

- **contains(String, String, boolean)** — This function returns a Boolean value by checking whether the first string contains the second string. If the Boolean value is true, the function works with case-sensitive strings.
- **dateAfter(Date, Date)** — This function returns a Boolean value by checking whether input first date comes after input second date or not.
- **dateBefore(Date, Date)** — This function returns a boolean value by checking whether input first date comes before input second or not.
- **dateDiff(Date1, Date2)** — This function returns time as along value by calculating difference by subtracting input Date 2 from input Date 1.
- **dateEqual(Date, Date)** — This function returns a Boolean value by checking whether input first date is equal to input second date or not.

  The date provided at the execution time must be in the formats defined below:
  Add dateType Variable format in rest execution through postman like “DD-MM-YY” while testing execution from soap UI is like "YYYY-MM-DD". Example for DD-MM-YY Thh:mm:ssZ: "30-01-2015T09:30:10Z" represents date 30 January 2015 and time 9:30:10 in 12-hour format. Example for YYYY-MM-DDT "2015-01-30" represents date 30 January 2015.

- **division(float1, float2)** — This function returns a float value by dividing float input 1 by float input 2.
- **divide(BigDecimal, BigDecimal)** — This function returns a BigDecimal value by dividing BigDecimal input 1 by BigDecimal input 2.
- **floor(float)** — This function returns the largest integer value that is less than or equal to the input float argument.
- **getCurrentDateTime()** — This function returns the current date.
- **getDurationInDays(Date)** — This function returns the number of days elapsed between the provided date and the current date.
- **getDurationInMonths(Date)** — This function returns the number of months that elapsed between the provided date and the current date.
- **getDurationInYears(Date)** — This function returns the number of years that elapsed between the provided date and the current date.
- **getLogValue(float)** — This function returns a log value at base 10 of the input float argument.
- **getLogValue(int)** — This function returns a log value at base 10 of the input integer argument.
- **getLogValue(long)** — This function returns a log value at base 10 of the input Long argument.
- **max(int, int)** — This function returns an integer value for the maximum of two input integer arguments.

- **max(float, float)** — This function returns a float value for the maximum of two input float arguments.
- **max(long, long)** — This function returns a long value for the maximum of two input long arguments.
- **max(BigDecimal, BigDecimal)** — This function returns a BigDecimal value for the maximum of two input BigDecimal arguments.
- **min(int, int)** — This function returns an integer value for the minimum of two input integer arguments.
- **min(float, float)** — This function returns a float value for the minimum of two input float arguments.
- **min(BigDecimal, BigDecimal)** — This function returns BigDecimal value for the minimum of two input BigDecimal arguments.
- **min(long, long)** — This function returns a long value for the minimum of two input long arguments.
- **mod(int, int)** — This function returns an integer value for the mod of two input integer arguments.
- **mod(float, float)** — This function returns a float value for the mod of two input float arguments.
- **mod(long, long)** — This function returns a long value for the mod of two input long arguments.
- **mod(BigDecimal, BigDecimal)** — This function returns BigDecimal value for the mod of two input BigDecimal arguments.
- **multiply(int1, int2)** — This function returns integer value by multiplying input integer 1 with input integer 2.
- **multiply (float1, float2)** — This function returns float value by multiplying input float 1 with input float 2.
- **multiply (long1, long2)** — This function returns a long value by multiplying input long 1 with input long 2.
- **multiply(BigDecimal, BigDecimal)** —This function returns BigDecimal value by multiplying input BigDecimal 1 with input BigDecimal 2.
- **negate(int)** — This function returns the negative value of the input integer argument.
- **negate(float)** — This function returns the negative value of the input float argument.
- **negate(long)** — This function returns the negative value of the input long argument.
- **negate(BigDecimal)** — This function returns the negative value of the input BigDecimal argument.

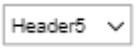
- **normalizeSpace(String)** — The function returns the argument string with whitespace normalized by stripping leading and trailing whitespace and replacing sequences of whitespace characters with a single space.
- **numberValue(String)** — This function returns the float value for the input string argument.
- **numberValue(float)** — This function returns the float value for the input float argument.
- **numberValue(long)** — This function returns the float value for the input long argument.
- **numberValue(int)** — This function returns the float value for the input integer argument.
- **numberValue(boolean)** — This function returns the float value for the input Boolean argument.
- **pickBigDecimalValuesFromDB(String)** — This function takes one string parameter which takes a database select query for decimal values as input and returns the database values in a comma separated string.
- **pickDateValuesFromDB(String)** — This function takes one string parameter which takes a database select query for date values as input and returns the database values in a comma separated string.
- **pickFloatValuesFromDB(String)** — This function takes one string parameter which takes a database select query for float values as input and returns the database values in a comma separated string.
- **pickIntValuesFromDB(String)** — This function takes one string parameter which takes a database select query for integer values as input and returns the database values in a comma separated string.
- **pickLongValuesFromDB(String)** — This function takes one string parameter which takes a database select query for long values as input and returns the database values in a comma separated string.
- **pickStringValuesFromDB(String)** — This function takes one string parameter which takes a database select query for string values as input and returns the database values in a comma separated string.
- **round(float)** — This function returns the nearest long value for the input float argument by rounding off.
- **roundToInt(float)** — This function returns the nearest integer value for the input float argument by rounding off.
- **startsWith(String, String)** — This function returns a Boolean value by checking whether the first string starts with characters of the second string for the two given strings.

- **stringLength(String)** — This function returns the integer value for number of characters in the input string argument.
- **stringValue(String)** — This function returns the string value of the input string argument.
- **stringValue(Date)** — This function returns the string value of the input Date argument.
- **stringValue(float)** — This function returns the string value of the input float argument.
- **stringValue(long)** — This function returns the string value of the input long argument.
- **stringValue(int)** — This function returns the string value of the input integer argument.
- **stringValue(boolean)** — This function returns the string value of the input Boolean argument.
- **substring(String, int, int)** — This function returns the substring of the string starting at the position specified in start argument with length specified in length argument.
- **subStringAfter(String, String)** — This function returns the substring of the first argument string that follows the first occurrence of the second argument string in the first argument string.
- **subStringBefore(String, String)** — This function returns the substring of the first argument string that precedes the first occurrence of the second argument string in the first argument string.
- **subtract(int1, int2)** — This function returns an integer value by subtracting input integer 2 from input integer 1.
- **subtract(float1, float2)** — This function returns a float value by subtracting input float 2 from input float 1.
- **subtract(long1, long2)** — This function returns a long value by subtracting input long 2 from input long 1.
- **subtract(BigDecimal, BigDecimal)** — This function returns a BigDecimal value by subtracting input BigDecimal 2 from input BigDecimal 1.
- **sum(int1, int2)** — This function returns an integer value by adding input integer 1 with input integer 2.
- **sum(float1, float2)** — This function returns a float value by adding input float 1 with input float 2.
- **sum(long1, long2)** — This function returns a long value by adding input long 1 with input long 2.

- **sum(BigDecimal, BigDecimal)** — This function returns a long value by adding input BigDecimal 1 with input BigDecimal 2.
- **translate(String1, String2, String3)** — This function returns a string value by searching input string 2 in input string1 and replacing the matching string with input string3 in input string1.
- **wildCardAst(String1, String2)** — This function returns a Boolean value by searching input String 2 in input String 1.It allows asterisk searching.
- **lookUpIntValuesFromDB** — This function is used to find integer-type values in the database.
- **lookUpLongValuesFromDB** — This function is used to find long-type values in the database.
- **lookUpFloatValuesFromDB** — This function is used to find float-type values in the database.
- **lookUpDateValuesFromDB** — This function is used to find date-type values in the database.
- **lookUpBigDecimalValuesFromDB** —This function is used to find big decimal-type values in the database.

Formatting text

Refer the below table to use the text formatting options:

Option	Description
Formats 	Use this option to change the style format of the selected text.
Font 	Use this option to change the font of the selected text.
Size 	Use this option to change the size of the selected text.
Bold 	Use this option to bold the selected text.
Italic 	Use this option to italicize the selected text.
Underline 	Use this option to underline the selected text.
Strike 	Use this option to strike-through the selected text.
Paragraph style 	Use this option to change the paragraph style.
Line height 	Use this option to change the space between the lines of the selected text or the paragraphs.
Font color 	Use this option to change the color of the selected text.
Subscript 	Use this option to make the selected text subscript of the normal text.
Superscript 	Use this option to make the selected text superscript of the normal text.
Outdent 	Use this option to change the outdent of the selected text, that is, shift the text towards the left margin.
Indent 	Use this option to change the indent of the selected text, that is, shift the text away from the left margin or towards the right margin.

Option	Description
Align	Use this option to change the alignment of the selected text. • **Align left** — Changes the alignment of the text to left. • **Align center** — Changes the alignment of the text to the center. • **Align right** — Changes the alignment of the text to right. • **Align justify** — Changes the alignment of the text to justify making the left and right edges of the text straight.
List	Use this option to create a numbered or bulleted list in the description.
Quote	Use this option to create a quote in the description.
Horizontal line	Use this option to add different types of horizontal lines to the description.
Code view	Use this option to create a code snippet in the description.
Remove format	Use this option to clear the formatting applied to the selected text.
Undo	Use this option to undo the changes made to the selected text.
Redo	Use this option to redo the changes that were undone just now.

Appendix A: Executing Rule Builder REST services

REST endpoint data is an execution feature that allows you to test the rule packages and rule flows deployed as REST endpoints. By default, the latest version of the main rule package or rule flow gets tested. However, you can also test other versions of a rule package or rule flow individually.
You can view and download all the rule flows and rules packages deployed as REST services. Each endpoint is the URL location from which REST services can access the resource that you want to execute.
You can view REST endpoint data of the main rule package and rule flow as well as of the other versions.

Viewing REST Endpoint Data of the Rule Package or Rule Flow

To access the REST Endpoint Data, follow the below steps:

1. Select the required rule package or rule flow under the Deployed tab. mode. The definition of the selected Rule Package or Rule Flow appears.
2. Click **REST End Point Data** button present in the lower-right corner. The REST End Point dialog appears. The following information required to execute the REST services appears:
   • URL
   • HTTP Method
   • Content-Type
   • Request JSON

3. Click the copy icon to copy the URL.
4. Paste the URL in Postman.
5. Again, copy and paste the HTTP Method as per the REST Endpoint Data. Modify the HTTP Method in Postman accordingly.
6. Copy the JSON code and paste the copied JSON code in the body section of the Postman with the following settings:
   • Select raw radio button.
   • Select JSON (Application/JSON) using the dropdown.
7. Change the values in place of "?" For example, "value": "answer",
8. In case of **array-type** entities, create an array format and then add the values accordingly. For example, "value": [1,2],
9. *(Optional)* Go to the **Headers** tab and search for "Authorization" in order to add the corresponding value which is Password (Token) if you have set the password for the rule execution while deploying. Enter the password in the Token field that you have set while deploying the Rule Flow. The token must be non-encrypted.
10. Click **Send** to execute the JSON script. A success message appears after the successful execution of the JSON script. In case of wrong input or authentication failure, an error appears.
11. After entering complete values of input and output, click the **Download API Data** to download the .zip folder that includes data of rules. It is the same data that appears in the REST End Point dialog.

For a date-type member variable, the input appears in the "dd-MM-yyyy" format, and in JSON, the output appears in the "Wed Feb 03 00:00:00 IST 2021" format.

Generating encrypted password

To generate an encrypted password, follow the below steps:

1. Go to the following path to download the *BRMSPasswordEncryptionUtility.zip* file. *https://ftp.newgen.co.in/~OmniRules/iBPSCommon/BRMSPasswordEncryptionUtility.zip*
2. Extract and open the *BRMSPasswordEncryptionUtility.zip* file.
3. Open the *Run_Encryption.bat* file in the edit mode.
4. Replace the *Java Path* mentioned in the file with the *Java Path* of your environment in the *Run_Encryption.bat* file.
5. Run the *Run_Encryption.bat* batch file. The command prompt appears.
6. Enter the drive location where you want to generate and store the encrypted password.
7. Enter the password. An *AuthCode.txt* file gets generated at the specified location with the content: Encrypted Password=<encrypted_key>
8. Use the value of Encrypted Password as the key mentioned in the newly generated file, with the “Authorization” header when running the password-protected rest service from code, Postman, or any other similar tool to execute rest services.

Appendix B: Integrating Rules with NewgenONE Process Designer

Refer to the section *Integration points* in the *NewgenONE Process Designer User Guide* for information on integrating rules and rule flows using Business Rule and WebService worksteps.

Appendix C: Executing Rule Builder secured services

With NewgenONE Rule Builder, you can create both secured and unsecured REST and SOAP services. However, by default, Rule Builder enables users to create only secured services for protecting sensitive information. You can control this service behavior by setting the *SecuredButtonVisibleOverDeployScreen* key in the Rule Builder INI to "N" to support the creation of unsecured services. You get an option to either create a secured or unsecured Rule Builder service while deploying a rule package.

For more information on Rule Builder INI, refer to the *NewgenONE Configuration Guide*.

By default, SOAP service configuration is disabled. To enable the SOAP service configuration, refer to the *Automation Studio* section of the *NewgenONE Configuration Guide*.

This topic explains how to execute a secured Rule Builder SOAP or REST service with third-party vendors including SoapUI, Postman, and others. It includes the following:

- Executing rule with a secured SOAP service
- Executing rule with a secured REST service

Executing rule with a secured SOAP service

In this case, NewgenONE provides the following types of authorization to execute a secured Rule Builder SOAP service using SoapUI:

- The signed-in user must have valid user credentials including userID and token, or userDBId:
  - **UserID** — The name of the user.
  - **Token** — The encrypted password of the user. You encrypt the password using the **WAS_Password_Encryption** utility available at the following FTP location — */OmniRules/public_html/iBPSCommon/WAS_Password_Encryption*
  - **userDBId** — The session ID of the user. You get the session ID from OmniDocs. It is encoded and encrypted.

- The users must have sufficient rights on the rule flow or rule package they want to execute. These rights include Rulepackage Execution Rights and Ruleflow Execution Rights.

Executing rule with a secured REST service

In this case, NewgenONE provides the following types of authorization to execute a secured Rule Builder REST service using Postman:

- The signed-in user must have valid user credentials including userID and token, or userDBId:
  - **UserID** — The name of the user. Add the username to the HTTP header with the **userName** key.
  - **Token** — The encrypted password of the user. You encrypt the password using the **WAS_Password_Encryption** utility available at the following FTP location – */OmniRules/public_html/iBPSCommon/WAS_Password_Encryption* Add the encrypted password to the HTTP header with the **tokenId** key.
  - **userDBId** — The session ID of the user. You get the session ID from OmniDocs. It is encoded and encrypted. Add the session ID to the HTTP header with the **Authorization** key.
- The users must have sufficient rights on the rule flow or rule package they want to execute. These rights include Rulepackage Execution Rights and Ruleflow Execution Rights.

Download normal report

To download a normal report, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click the ellipsis icon and select **Report**.
3. Select **Normal**.
4. Select the report type that you want to download, **Word Doc** or **HTML**.
   The report gets downloaded as a .zip file.

Depending on the type of browser you are using, you may be asked to save the file.

Viewing arguments associated with a rule flow

Use the Associated Arguments option to view the details of arguments associated with the selected rule flow. This option is available in the upper-right corner.

To view arguments associated with a rule flow, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click **Associated Arguments**. A dialog with the argument details appears.
   The list displays the name and type of the argument parameters.
3. Click **Close** to exit the dialog.

Related topic(s)

Associating arguments with a rule flow

Download password protected report

To download password protected report, follow the below steps:

1. Select the required rule flow under the **Deployed** subtab.
2. Click ellipsis icon and select **Report**.
3. Select **Password Protected**.

4. Select the report type that you want to download, **Word Doc** or **HTML**. The Set Password Key dialog appears.
5. Specify a password Key.

   System asks you to enter the password key when you extract files from the downloaded .zip file.

6. Click **Download**. The report gets downloaded as a .zip file.

   Depending on the type of browser you are using, you may be asked to save the file.