Search documentation

Appian

Rules

Appian 7.6

A rule is an expression you save to the system with a user-defined name, description, definition, and central location. You can reuse a rule throughout the system. To use it with an application object, it needs to be added to the object's application.

Toggle Us

Rules do not belong to a specific process model, report, or channel. Any expression created through the Expression Editor can be saved as a rule. Al rules are stored within the Rules and Constants folder of the Application Designer or a user-defined sub-folder.

There are three types of rules:

- **Interfaces**: A stored expression that returns one or more SAIL components that can be displayed as a SAIL interface.
- **Expression Rules**: A stored expression that works like an Appian Function, except you create your own rule inputs to use as parameters.
- **Query Rules**: A stored expression that allows you to retrieve data from relational databases.

See also: Expressions

Tu

Re

Ot

Rules Tab

The **Rules** tab in the Application Designer displays the rules and rule folders you have **Viewer** rights to.

Each folder displays a rule's name, description, creator, last modified date/time, and version number.

- Select a column header to sort by that column.
- Select a checkbox for a rule to view actions you can take on it.

Available Actions

Depending on your rights within a rules folder, the following actions are available from the toolbar to take on a rule or rule folder:

Click	To ...
Up One Level	Go up to the folder above your current folder.
New Folder	Create a new subfolder under your current folder.
Create an Interface	Open the Interface Designer.
Create an Expression Rule	Open the Create an Expression Rule page.
Create a Query Rule	Open the Create a Query Rule page.
Create a Constant	Open the Create New Constant page.

Rename	Give an existing folder a new name.
Move	Move the selected rule or folder from one location to another. • Rules folders must remain within the parent Rules and Constants folder.
Delete	Delete the selected rule.
Security	Update or define access levels for the selected rule or folder.

Quick Links

From the Rules tab, the following quick links always appear in the left navigation bar:

- **Create an Interface**- Opens the Interface Designer. See also: Interface Designer
- **Create an Expression Rule**- Opens the Create an Expression Rule page.
- **Create a Query Rule** - Opens the Create a Query Rule page.
- **Create New Constant** - Opens the Create New Constant page. See also: Constants
- **Import Rules and Constants** - (Deprecated) Rules can only be imported or exported as part of an application. See also: Importing an Application
- **Root Folder Security** - Opens the Configure Security for Rules and Constants window to modify security for all rules.
- **Starred Rules and Constants** - Displays the rules, constants, and folders you starred as favorites.

Searching

From the Rules tab, there are two ways you can locate rules:

1. Using the **Search** field.
    - All rules and constants you have at least Viewer rights to that match the text you enter display in a drop-down menu.
    - Select an item from the drop-down to open it.
2. **Browsing** through the Rules and Constants folder structure.
    - This can be done through the folder tree in the left navigation bar or the list view.
    - Select an item from the tree or list view to open it.

Starring

You can star rules and rule folders to mark for quick access. These objects appear when you select Starred Rules and Constants in the left-navigation o the Rules tab.

To mark a rule or folder as starred, click the star icon next to its name.

- The icon changes from clear to gold.

To remove an item from the starred list, click the gold star so it displays as clear again.

Executing

Rules execute similar to Appian functions. There are two ways you can add rules to an expression field value:

1. From the **Rules & Constants** tab in the Expression Editor, select a rule and add it to the Expression Editor Canvas.
    - Only rules and rule folders you have at least Viewer rights to display in the Expression Editor.
    - Rest your pointer on a rule to display a description of the rule and the rule inputs.
2. Directly reference the rule on the Expression Editor Canvas or in an expression field. To avoid potential naming conflicts with other process variables or constants, indicate the domain.
    - For example, a rule named myRule, which takes in two inputs as parameters can be referenced in the expression using the following synta `rule!myRule(Input1, Input2)`

When passing arguments by position to a rule, all inputs are required. When passing by keyword, all inputs are optional.

See also: Passing Arguments

Rule inputs can be indirectly evaluated in an expression.

See also: Indirectly Evaluating Arguments

The user executing a query rule must have viewer rights for the data store referenced in the rule.

See also: Assigning Data Store User Rights

NOTE: Users that do not have Viewer rights to a rule are still able to execute it by directly referencing it in an expression field or Expression Editor Canv if they are aware of the rule name and rule inputs.

Query Rule Paging Parameter

You can limit the number of results returned by your query rule by passing an optional paging parameter of type PagingInfo

- This parameter is optional for all query rules.
- It does not show up as a rule input when the query rule in the **Rules** tab or Expression Editor.

Use the topaginginfo() function to construct the paging parameter.

- You can access topaginginfo() through the Expression Editor.
- See also: topaginginfo()

When a paging parameter is specified, a value of type DataSubset with the following fields is returned:

- **startIndex**: the value of the startIndex field in the PagingInfo record passed to the query rule.
- **batchSize**: the value of the batchSize field in the PagingInfo record passed to the query rule.
- **sort**: the value of the sort field in the PagingInfo record passed to the query rule, if any.
- **totalCount**: the total number of results that match the query conditions defined in the query rule.
- **data**: the requested list of items; the type matches the type of the entity selected in the Entity to query section.
- **identifiers**: the value of the primary key for each item in the results list, if the type of the result has a primary key defined.

This DataSubset value can then be used with the todatasubset() function to configure a paging grid form component.

See also: todatasubset() and Paging Grid Component

For example:

If we have the following query rule with one rule input and no paging parameter, the type of the results returned match the type of the entity selected the Entity to query section.

```
getAllData(pv!somePV)
```

To limit the number of results to the first 25 items only, add a paging parameter to create the following rule:

```
getAllData(pv!somePV, topaginginfo(1, 25))
```

To access the list of results returned by the query rule, use dot notation.

For example:

```
getAllData(pv!somePV, topaginginfo(1, 25)).data
```

To return just the total number of results that match your query conditions, use a batchSize of zero and .totalCount.

For example:

```
getAllData(pv!somePV, topaginginfo(1, 0)).totalCount
```

To retrieve a DataSubset with all results that match your query conditions (this is similar to not passing a PagingInfo parameter which returns all the results directly), use a batchSize of -1 .

For example:

```
getAllData(pv!somePV, topaginginfo(1, -1)).totalCount
```

Advanced Sorting Configurations

The value of type PagingInfo passed as an argument to a query rule may optionally reference a value of type SortInfo to order the results by one or more fields to create a multi-column sort on query rules.

To populate the values of a SortInfo value in a process model, use a node input. Outside of a process model, list the CDT values. You cannot use the topaginginfo() expression function to configure sort parameters.

For example, the type of your entity has the following data structure:

```
Person (NewComplexDataType)

    |- FirstName (Text)

    |- MiddleName (Text)

    |- LastName (Text)

    |- DateOfBirth (Date)
```

To return the first 100 persons ordered by their date of birth in descending order, then by last name in ascending order, and finally by first name in ascending order, you configure your query rule as follows:

```
getAllPersons({startIndex:1, batchSize:100, sort:{{field:"DateOfBirth", ascending:false()}, {field:"LastName", ascending:true()}, {field:"FirstName", ascending:
rue()})
```

The sort field that you specify must be a single or array primitive type.

For example, you have a complex data type called Case which references a list of Notes, and each note has a CreatedOn date and time field as follows:

```
Case

  |- Summary (Text)

  |- Notes (List of Notes)

     |- Summary (Text)

     |- CreatedOn (Date and Time)
```

If you want to return the first 25 cases which have the most recent notes, your sort field would be `Notes.CreatedOn`.

If your query rule definition specifies a sort configuration, the sort configuration is applied after all the sort fields specified in the PagingInfo parameter, if any.

The system automatically applies a last sort configuration by the primary key of the query rule entity to ensure a deterministic order of results every tim the query rule is executed.

Creating

The process of creating a rule is different for expression rules and query rules.

You can only create a rule in the rule folders you have **Editor** or **Administrator** rights to.

NOTE: By default, rules created for events or Portal reports can only have up to 20 nested rules or recursively call itself 20 times. A nested rule is a rule called by another rule. This limit is determined by the `RULE_DEPTH_LIMIT` configuration in the `custom.properties` file.

See also: Evaluation Order and Rule Depth Limit

Interfaces

Interfaces are created and tested using the Interface Designer. Additionally, existing expression rules that return SAIL components can be converted to interfaces.

See also: Creating a New Interface and Converting an Existing Expression Rule

Expression Rules

Creating an expression rule involves creating an expression and applying unique identifiers to it.

To create an expression rule, complete the following:

1. From the **Rules** tab, click the **Create an Expression Rule** quick link.
   - The **Create an Expression Rule** page displays.
2. Enter a unique name and description for the rule in the associated fields.
   - This name must be unique among expression rules, query rules, and constants.
   - The description is displayed in the Description Pane for the rule in the Expression Editor.
3. In the **Save In** field, enter in or select a folder to save the rule into.
4. Add the rule inputs by clicking New Input and entering a name and type for each input.
   - Check the Multiple checkbox to allow users to enter multiple values for that rule input.
5. Enter the rule definition in the form of an expression in the Definition field.
   - Line breaks are supported.
   - See also: Expressions
6. Click the **Test Rule** button.
   - The **Test Rule** dialog box displays.
7. For each input parameter, specify a value.
8. Click **Test**.
   - The rule is evaluated and the result is displayed.
9. To test the rule with a different set of input parameters, change the inputs and click Test Again.
10. Click **Close** when done testing.
    - You can make further changes to rule and test again until the expected results return.
11. To automatically indent the expression for improved readability, such as when defining SAIL interfaces, click the **Format Rule** button.
    - The rule definition is indented.
12. Click **Save Rule** to save the new rule.

You can also create an expression rule from within the Expression Editor by clicking **Save as an expression rule**. A window similar to the Create an Expression Rule page displays with the expression from the Expression Editor as the rule definition.

NOTE: Rule definitions that reference a process variable, a property specific to a process model, a process, or a task cannot be tested and become difficult to maintain. Appian recommends passing any data needed for the rule as a rule input.

Query Rules

The query rules you define are limited in how long they wait to return results (10 seconds). This setting can be configured by a system administrator.

There is also a limit to the amount of memory that can be returned by a single query rule. This is set by the `conf.data.query.memory.limit` value. The system will display an error with code `APNX-1-4164-024` if this error is reached by a query rule. Use the paging parameter to return less data (or return data in batches) to avoid the limit.

See also: Configuring Query Rule Limits

Query rules must be included in an expression rule to be tested within the Rules View. See above: **Creating - Expression Rules**

To create a query rule, complete the following:

1. From the **Rules** tab, click the **Create a Query Rule** quick link.
    - A primary data source must be defined for the Create a Query Rule link to appear.
    - The Create an Expression Rule page displays.
2. In the **Enter query rule name here** field, type the name of your query rule.
    - This name must be unique among expression rules, query rules, and constants.
3. (Optional) In the **Enter query rule description here** field, enter the purpose of the query rule.
    - This information is displayed in the Description Pane for the rule in the Expression Editor.
    - When your query rule requires rule inputs, describe the values that a process developer should enter.
4. Click the **Rule Folder** field or the Browse button next to the field.
    - The Choose a Folder dialog box is displayed.
5. Select a rule folder or click the **Create a new folder** button to add the desired folder name.
6. (Optional) Add rule inputs to your query rule.
    - See below: **Defining Query Rule Inputs**
7. In the entity to query field, select the text field or the Browse button.
    - The **Choose an Entity** dialog box is displayed listing published data stores that you have been assigned the right to view.
    - Each query rule is associated with a single entity in a data store.
    - When the query rule is executed, the type of the results matches the type of the entity selected.
8. Select a data store, then select an entity, and click **OK**.
9. (Optional) Add query conditions.
    - See below: **Defining Query Conditions**
10. (Optional) Configure the query rule results to be ordered according to a specific field by completing the following.
    - In the **Result Sorting** section, select the Sort query results by [...] checkbox.
    - Select the text field to view the data store entity.
    - Select the entity element that you want to use for sorting.
    - Select In ascending order or In descending order from the options list.
11. When complete, click **Create Rule** to save it.

Defining Query Rule Inputs

A rule input requires that an additional value be provided when the rule is executed. - You must create a rule input if you want to use conditions in your query rule.

Rule inputs for query rules differ from those used by expression rules in that they are not prefaced with ri!. They also support fewer data types.

To define a query rule input, complete the following:

1. When creating or editing your query rule, click **Add Rule Input**.
    - The **New Rule Input** group box is displayed.
2. Enter a Name for the rule input in the **Name** field.
    - Names cannot contain spaces or special characters, and must contain at least one letter.
3. (Optional) Enter the purpose of the rule input in the **Description** field.
4. Select the data type of the input from the **Types** list.
    - The following data types are available for query rule inputs:
        - Boolean
        - Date
        - Date and Time
        - Number (Decimal)
        - Number (Integer) - default
        - Text
        - Time
5. (Optional) Select the **Allow multiple values** checkbox to accept more than one value in the input.

NOTE: Null values are not allowed as the value of a rule input when the query rule is evaluated.

Defining Query Conditions

By default, a query rule retrieves all rows from a data store entity.

You can add conditions to mine the data store for information that meets various criteria.

1. Multiple conditions can be defined.
2. All conditions must be met for any results to be returned.
3. Conditions can be combined using AND logic.

NOTE: Case-sensitivity or insensitivity for text comparisons is determined by your RDBMS settings - not a setting configured in Appian.

To define a query condition, complete the following:

1. When creating a query rule, click **Add Condition**.
    - A group box with three fields is displayed.
2. In the first field, select a node in the Data Store entity where you want to conditionally retrieve data.
    - The nodes displayed are the same as the nodes in the custom data type used to create the entity.

- You cannot browse through individual records in the node.
- Only the lowest-level child of a node can be selected, and only when the node holds single values.
- No multiple-value entities are displayed.

3. In the second field, select an operator to use in filtering the data.
   - The operators in the table below are available.
   - The operators that display vary according to the data type of your rule input, as listed in the right column of following table.
4. In the third field, select a rule input from the ones you have defined for this rule.
   - This is the comparison value that is checked against values in the data store.
5. Click **Create Rule**.

Operator	Description	Supported Data Types
=	The case-insensitive condition returns True if the field value exactly matches the compared value. • If your rule does not require case-sensitivity, consider using the exact() function to improve performance.	Boolean, Date, Date and Time, Time, Numbers
<	The condition returns True if the field value is less than the compared value.	Boolean, Date, Date and Time, Time, Numbers, Text
<=	The condition returns True if the field value is less than or equal to the compared value.	Boolean, Date, Date and Time, Time, Numbers, Text
>	The condition returns True if the field value is greater than the compared value.	Boolean, Date, Date and Time, Time, Numbers, Text
>=	The condition returns True if the field value field value is greater than or equal to the compared value.	Boolean, Date, Date and Time, Time, Numbers, Text
<>	The condition returns True if the field value does not match the compared value.	Boolean, Date, Date and Time, Time, Numbers, Text
starts with	The condition returns True if the field value begins with the text string listed as your comparison value. • The match must be performed based on the literal string provided. Special characters do not have special meaning as wildcards. • For example, [starts with: "apple*"] searches for the word apple and the asterisk character (*) rather than any character. If the field value is null, the item will not be returned regardless of the rule input value.	Text
ends with	The condition returns True if the field value ends with the text string specified as your comparison value. If the field value is null, the item will not be returned regardless of the rule input value.	Text
includes	The condition returns True if the field value contains the string specified as the comparison value. If the field value is null, the item will not be returned regardless of the rule input value.	Text
in	The condition returns True if the field value exactly matches any one of the values in the compared list of values.	Boolean and list of Boolean, Date and list of Date, Date and Time and list of Date and Time, Time and list of Time, Numbers and list of Numbers, Text and list of Text
not in	The condition returns True if the field value does not match any one of the values in the compared list of values.	Boolean and list of Boolean, Date and list of Date, Date and Time and list of Date and Time, Time and list of Time, Numbers and list of Numbers, Text and list of Text

Editing

You can edit any existing rule in the system that you have **Editor** or **Administrator** rights to.

To edit a rule, search for and open it from the Rules tab. If you have **Editor** or **Administrator** rights to, the detail pages displays in Edit mode After modifying the rule properties, click **Save New Version** to save it.

- When a new version is saved, all subsequent expressions and process models that utilize the rule will use the new rule definition.
- The name field cannot be edited.
- Changing the rule inputs to a rule may cause it to work improperly if it is already being used in the system.

The following read-only fields are shown in addition to the general properties for a rule:

- **Created By**: Specifies then name of the user who created the rule and the date and time at which the rule was created.
- **Last Modified By**: Specifies the name of the user who last modified the rule and the data and time at which the rule was last modified.
- **Version**: When a new rule is created, it will be assigned a version number of 1. All rule versions will have an integer designation (e.g., 1.0, 2.0 3.0 etc., etc.,). Any subsequent saving of the rule will have the standard versioning integer notation (i.e., it will be assigned the next whole number).

See below: **Version Control**

Deleting

Deleting a rule deletes it from the rule repository and prevents users from further viewing or editing it. It is still available at process execution time.

- This allows all process models and the processes instances started from those process models to continue successfully executing the rule.
- If you want existing processes that use the rule to fail, edit the rule to result in an error instead of deleting the rule.
- See also: error() function.

Only users with **Administrator** rights to a rule can delete it.

To delete a rule:

1. Open the folder from the Rules tab that contains the rule.
2. Select the checkbox next to the rule.
    - If you have Administrator rights, a Delete button appears on the toolbar.
3. Click the **Delete** button.

Moving

All rules must be stored within the **Rules and Constants** root folder or a user-defined sub-folder.

Users with admin rights to a rule or folder can move them to other folders within the root folder.

To move a rule or rules folder:

1. Open the folder from the **Rules** tab that contains the rule or rules folder.
2. Select the checkbox next to the rule or folder.
    - You can select more than one.
3. Click the **Move** button on the toolbar.
    - The **Choose Folder** dialog box is displayed listing all top-level folders directly beneath the root folder (Rules and Constants).
4. Locate the target folder and click the **Select** link.
    - You can also create a new folder if you have **Administrator** rights to its parent folder and select it.
5. Click **OK** to complete the move.

NOTE: All items assume the security rights of the target folder.

Importing

You must add rules to an application in order to export and import them.

To add rules to your application:

1. Open your application for editing.
2. Select the **Rules** Tab.
    - The list of rules, constants, and rules folders associated with the application is displayed.
3. (Optional) Click **Add - Rules Folder** and select the rules folder to associate with your application.
    - You cannot import an application rule without its associated rule folder.
    - See also: Application Deployment Guidelines
4. Click **Add - Rule or Constant** and select the rule to add.
5. (Optional) Click **Close** to exit the application definition assistant.
    - Documents and associated objects are saved when they are added to the list.
    - Closing the assistant is not required to save your selections.

See also: Importing an Application

Version Control

Each time you modify and save a rule, Appian updates the rule in all fields that use it and saves a copy of the previous version.

You can revert back to a previous version at any time. All versions are accessible by users with View rights to a rule.

The most recent version is the current version. Previous versions are numbered incrementally with the very first starting at number 1.

To access and revert back to a previous version:

1. From the **Rules** tab, open the rule's folder.
2. Click the rule's **Version** number to view a list of all versions.
    - This link only works if there are multiple versions of a rule.
3. Select a version's rule name to open it for editing.
4. To revert back to this version, click the **Save New Version** button.
    - It saves as the Current version.
    - You can also modify a previous version and save it as the new version without affecting an old version.

For example, you have two versions of a rule: the current version and version 1. If you open version 1, modify the rule, and click **Save New Version**, the system saves the modified version 1 as the current version, the previous current version becomes version 2, and version 1 retains its original definition.

Security

All rules must be stored within the **Rules and Constants** root folder or a user-defined sub-folder. Sub-folders and rules can inherit security from the ro folder, or security can be explicitly set for each object.

Rule security defines the right to create, update, view, and delete a rule or rule folder.

- At run-time, security restrictions do not prevent users from executing these objects.

Security parameters can be defined separately for a folder and its rules. To access a rule, a user must have permission to access both the rule and the folder that contains the rule.

Root Folder

By default, only system administrators can configure the rules root folder security.

To modify the security settings for the root folder, complete the following:

1. From the **Rules** tab, select Root Folder Security from the left=side navigation.
    - The **Configure Security for Rule** dialog box displays.
2. To modify the default security setting, select a different option from the drop-down for the initial statement.
    - This setting applies to all users expect those associated with a specific security setting.
3. To add a security setting specific for a user or group, click **Add Users and Groups**.
    - The **Choose Users and Groups** dialog box displays.
4. Select one or more users or groups and click **OK**.
5. Select the type of rights to give to the selected users and groups.
6. Click **Save Changes**.

Sub-Folders

Security settings for sub-folders can only be modified by users with **Administrator** rights to the folder.

To modify the security settings of a sub-folder, complete the following:

1. From the **Rules** tab, select the parent folder for the sub-folder.
2. Select the checkbox next to the sub-folder.
3. Click **Security** from the toolbar.
    - The **Configure Security for ... Folder** dialog box displays.
    - This dialog box lists the default security settings inherited from the (linked) parent folder.
4. Clear the **Inherit security from parent** checkbox. The security controls are enabled.
5. Select Administrator, Editor, Viewer, or no from the default privileges list (which applies to all users). — or —
6. Click **Add User and Groups** to select the people you want to add to the role map for the folder. The **Choose Users and Groups** dialog box is displayed, allowing you to look up individual users or groups that exist on the system.
    - Select the checkbox next to the user or group you want to update.
    - Select one of the following security settings and click **Save Changes**.

Administrator

- Create sub-folders.
- View the folder in the hierarchy.
- View rules and constants that reside in the folder.
- Delete rules and constants that reside in the folder.
- Create new rules and constants.
- Update rules and constants that reside in the folder provided you have Administrator or Editor rights for the rule or constant.
- Grant or change the access levels for the folder.
- Delete rules and constants.
- Import rules or constants.

Editor

- View the folder in the hierarchy.
- View rules and constants that reside in the folder.
- Create new rules and constants.
- Update existing rules and constants that reside in the folder provided you have Administrator or Editor rights for the rule or constant.
- Delete rules and constants that you have created.

Viewer

- View the folder in the hierarchy.
- View rules and constants that reside in the folder. Users must have Administrator or Editor rights for the rule or constant.

Deny

If a user or group is been assigned the Deny role, access to this folder is removed. They also cannot see the folder within the hierarchy.

Specific Rule

Security settings for a rule can only be modified by users with **Administrator** rights to the rule.

To create a security setting for a rule, complete the following:

1. In the **Rules** view, select the folder where the rule is stored.
2. Select the checkbox next to the rule.
3. Click **Security**.
    - The **Configure Security for ...** dialog box displays.
4. Clear the **Inherit security from parent** checkbox.
5. Select Administrator, Editor, Viewer, or no from the default privileges list (which applies to all users). — or —
6. Click **Add User and Groups**.
    - The Choose Users and Groups dialog box displays.
7. Select users and groups you want to add to the role map for the rule.
8. Select one of the following security settings and click **Save Changes**.

Administrator

- View the rule or constant.
- Delete the rule or constant.
- Update the rules or constant.
- Grant or change the access levels for the rule or constant.

Editor

- View the rule or constant.
- Update the rule or constant.
- Delete the rule or constant.

Viewer

- View the rule or constant.

Deny

If a user or group is been assigned the Deny role, access to the rule or constant is removed.

Remove all Security Settings for a Folder

- Within a **Configure Security for ...** dialog box, click **Clear**.
    - Removed users or groups are given the default folder rights.

Restrict Users from Creating Any Rules

The right to create and edit rules is managed on the **Rules Creation** page in the System Administration console.

By default, all registered users can create and edit rules and constants provided that they have necessary permissions to create a rule or constant withi a folder.

To restrict access to a list of users and groups who can create or edit a rule or constant, complete the following:

1. Open the **Rules Creation** page in the System Administration console
2. Click the **Only the users & groups selected below will be able to create rules and constants** option.
3. Click the button from the toolbar to select a specific set of users and groups.
    - The Choose Users and Groups window is displayed.
4. Select the users and groups you want to provide access.
5. Click **OK**.
6. Click **Save**.

All selected users and groups can be removed by clicking the button on the toolbar. Similarly, a user or group can also be deleted by selecting the radio button that corresponds to the user or group and clicking the Remove button on the toolbar.

After updating the security settings for creating a rule, you can choose to revert back to the default setting by selecting the radio button that corresponds to Allow all users to create rules and constants.

Best Practices

It is important that you tailor your query rules to provide only desired information, especially when substantial growth in the data-sets queried is

expected.

When designing rules, keep the following considerations in mind:

Give Unique Names to Each Rule

When a rule is given the same name as a process variable, the rule takes precedence over the process variable. In other words, the expression utilizes tl rule or constant instead of the process variable during execution. To avoid the potential for naming conflicts, we recommend keeping all rule, constant, and process variable names unique.

Constrain Your Results

By default, query rules do not return a constrained subset of matching data records, unless you configure query conditions and filters and call the quer rule using the PagingInfo parameter.

- In a typical SELECT query against an RDBMS, a constrained query might be built using the LIMIT clause. In Appian, this is achieved by calling the query rule with a PagingInfo parameter.
- Using a query rule, all records from an entity that match the query conditions are returned.
- The likely size of the data-set returned needs to be considered when building query rules.
- A larger number of records returned may affect performance of the application.
- Instead of writing a rule that returns all records or many records, add query conditions and filters that limit the number of results to only those needed.

Use Additional Criteria

- Instead of selecting all records less than or greater than a given record, look for additional attributes that you can use to filter the results.

Use Multiple Conditions

- Use more than one query condition to reduce the number of records in the result set.

Problematic Design Example

The following improper design example illustrates how not to effectively implement query rules.

Given a form that also displays the value of a previous item using a query rule, the following configuration might be used:

- A query rule named **GetPrior(currentId)** might be used to identify the next numerically-lower item according to the ID value.
  - In this example, **currentId** is a rule input that holds the id value of the current item.
- If the query rule used a query condition of **id** < **currentId**, this query rule would effectively return all matching id values that are numerically low than the currentId.
- In this example, only the highest-valued record out of the query rule's result set is desired.
  - This approach returns much more data than is needed, resulting in a much larger system load than is necessary.

Best Practice Example

It is possible to implement a query rule with an additional rule input to function as a boundary for the attribute that you're filtering on.

For example, instead of **id** < **currentId**, you could implement two query conditions:

```
Id < currentId
```

— and —

```
Id > minimumBoundary
```

The query rule would then use two rule inputs GetPrior(currentId, minimumBoundary). This allows you to nest the query rule in one or more if statements within your expression.

For example:

```
if(length(GetPrior(currentId, currentId-1)) > 0, GetPrior(currentId, currentId-1), GetPrior(currentId, currentId-10))
```