BREeze Premium - User Guide

Contents

Introduction

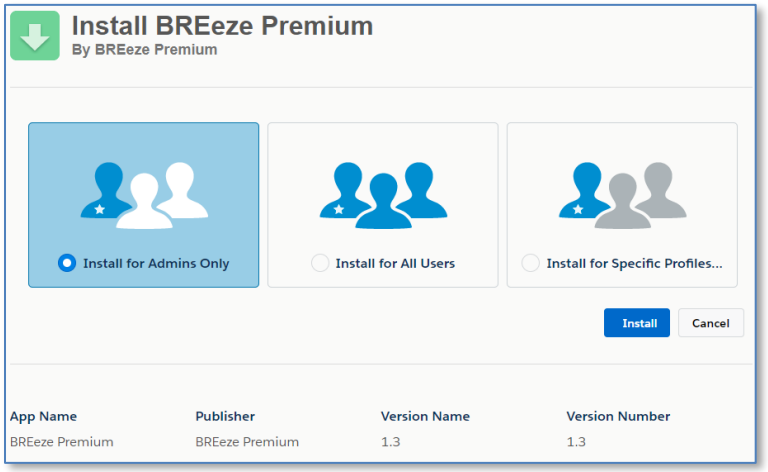
BREeze (**B**usiness **R**ules **E**ngine) is a dynamic rules engine that allows you to develop a rules set that can assign any field (or fields) on any object at any time and/or automate the creation of child records.

BREeze takes the concept of the standard Lead and Case Assignment rules and extends it across all of Salesforce. Whether you are looking to assign the owner of objects other than Leads or Cases, or if you're out of workflow rules due to trying to build a field update for every scenario, BREeze can simplify your complex data rules in an easy to use interface with no developer required to maintain. With BREeze you can assign any field on any object at any time.

Installation

To install the BREeze Premium application in any Salesforce org, follow these steps:

- Ensure that you have the BREeze base application installed: https://appexchange.salesforce.com/appxListingDetail?listingId=a0N3A00000ErAwrUAF
- Send an email to gearsdesignsupport@rafter.one requesting the installation details for the Premium package
- When you begin the installation process with the URL that you receive, you will first select the necessary access to the application
    - We recommend that in most instances, only the System Administrator should have access to BREeze so you can leave the default selection



- Once the appropriate access level is selected, click the Install button
- Depending on the number of customizations made to your org, the installation of BREeze Premium can take anywhere between a minute to ten minutes to complete
- If any installation errors occur, please contact gearsdesignsupport@rafter.one for troubleshooting assistance

Upgrades

Upon availability, notifications of all BREeze Premium version upgrades will be sent to the key stakeholders for the instance in which BREeze is installed. It is then up to the organization to decide whether to upgrade to the newest version or remain on the current version. If an upgrade is requested, GearsDesign can assist with the upgrade to the most recent version. There is no additional cost to upgrade to the most recent version of BREeze Premium.

Prior versions will continue to be supported; however, we recommend all customers upgrade to take full advantage of new BREeze features and functionality.

Frequency of Execution

BREeze is designed so that it can execute in almost any scenario. The most common deployments run on the creation of a new record or the updating of an existing record. However, BREeze can be fine-tuned to execute on a specific set of criteria if need be as well. With the release of BREeze 3.3, in combination with Process Builder, you can easily define how/when your BREeze rules will fire with clicks and not code.

Field Level Security

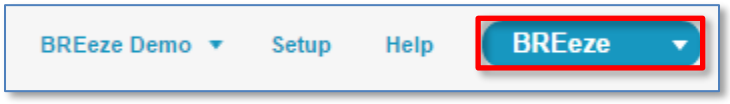
Please Note: BREeze adheres to all Salesforce object & field level security (FLS). For example, if a BREeze rule is configured to update a field and the running user that triggered the BREeze rule to be fired does not have access to that field via FLS, then BREeze will not update that field.

If your use case requires that FLS or object security not be observed, please contact us at gearsdesignsupport@rafter.one to discuss options and customizations for your business requirements.

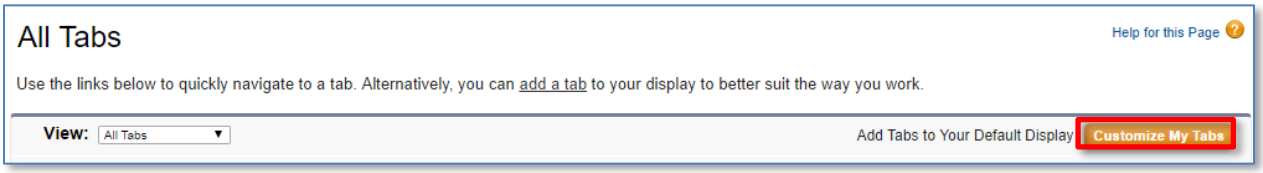
General Setup

Creating a New Rule

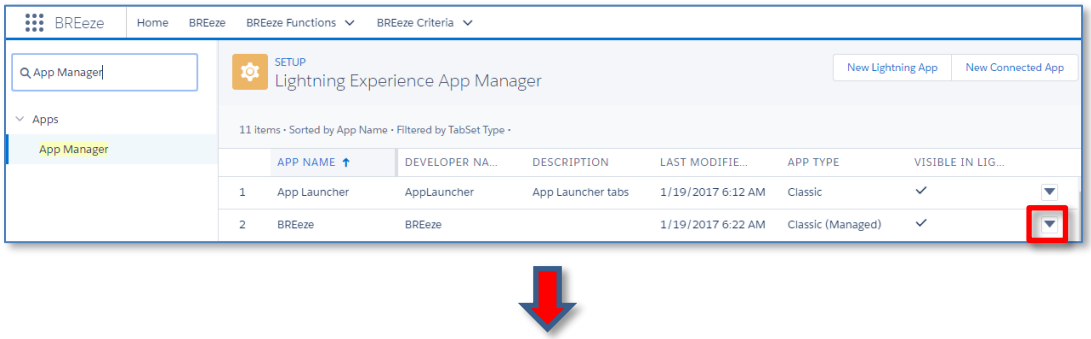
- From your Salesforce screen, navigate to the BREeze app

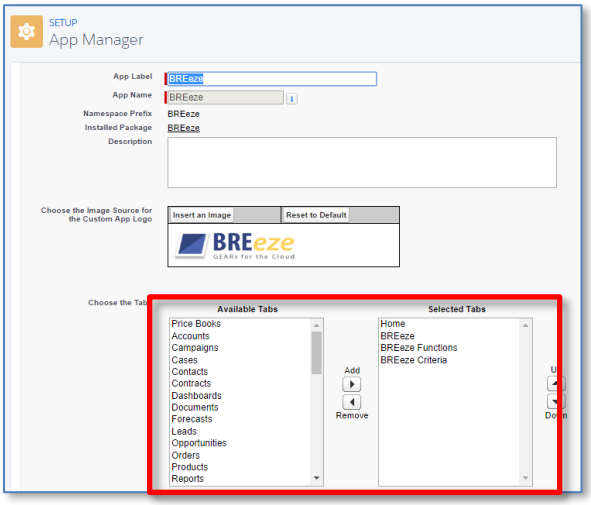


  - If the BREeze app is not available or is not visible by default:
    - *Classic* - You may need to add it to your default displays

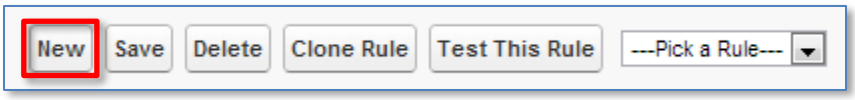


    - *Lightning Experience* – You may need to adjust the available tabs from the App Manager screen in the Setup menu

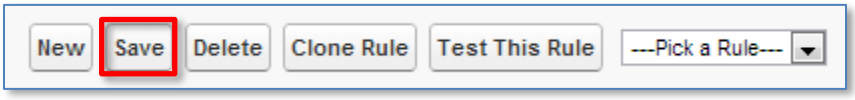




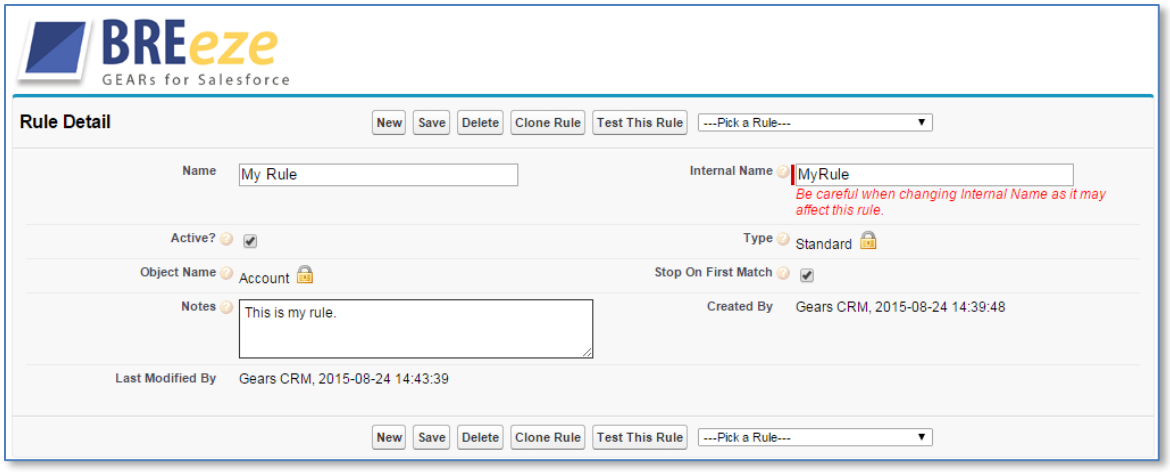
- Click on the ***New*** button at the top of the screen



- From the resulting screen, enter the following:
  - **Name**
    - Enter a descriptive name that describes the rule which you are building
  - **Active**
    - While building and testing the rule, this can remain unchecked however when you are looking to fully execute the rule against your data, you must check this box
  - **Type**
    - Select 'Standard' for a BREeze rule where you will be evaluating a single object and updating that object when it meets the defined criteria
    - Select 'Create Object' for when you are generating a related object (task, event, etc.) tied to a parent record that meets defined criteria which is determined in a 'Standard' rule type
  - **Object Name**
    - Select the core object which you are building the rule for
  - **Internal Name**
    - This will default, however set this to an intuitive name as this is the name that will be referenced in either the trigger that calls BREeze or in Process Builder
  - **Stop On First Match**
    - This indicates whether the rules engine will stop processing when it hits the first entry where all criteria evaluate to true, or whether it continues through all rules and evaluates/executes on all rules that evaluate to true
  - **Notes**
    - Enter a long description of the purpose of the rule or any other pertinent details
- Finally click the ***Save*** button at the top or bottom of the Rule Detail section

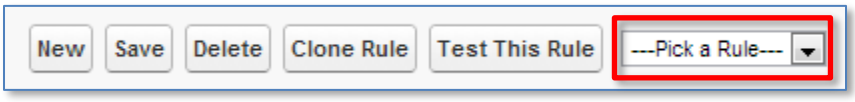


- The result should look like something like this with your rule name & associated object otherwise populated:

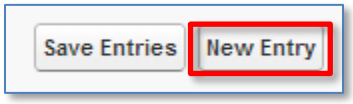


Creating New Rule Entries

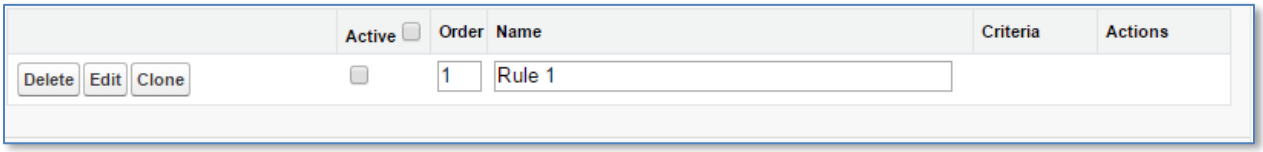
- Navigate to the BREeze tab within Salesforce
- Select the rule which you would like to add a rule entry for by selecting it from the dropdown menu



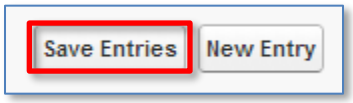
- Under the Rule Entries section, click on the ***New Entry*** button



- A new rule entry will appear within the Rules Entries section

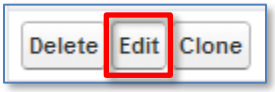


- By default, the entry will be named "Rule 1". In this field, enter a name that describes the criteria and/or action to be evaluated in that entry and then click on the ***Save Entries*** button

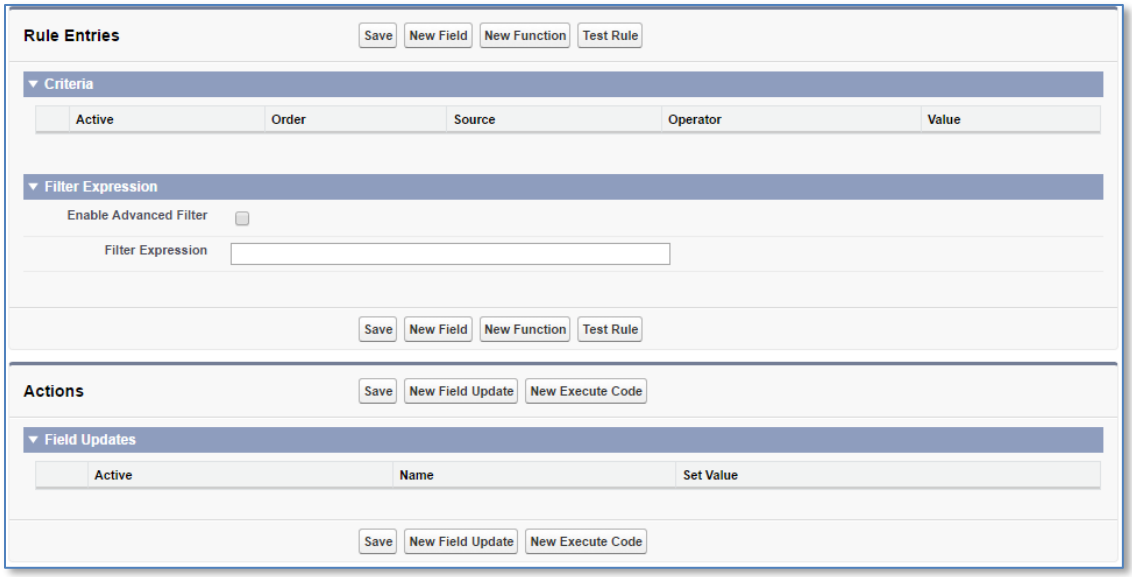


Defining Standard Rule Entries

- Click the ***Edit*** button on the left side of the rule entry



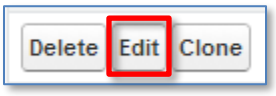
- The resulting screen allows you to define the rule criteria & actions:



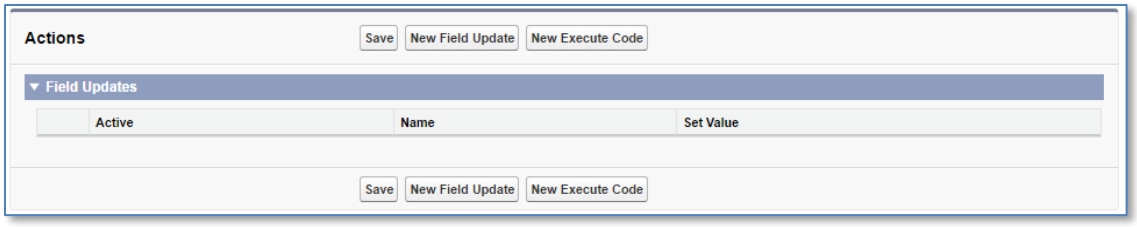
- Starting with the top half of the screen under the Rule Entries header, this is where you'll begin to define the criteria for your rule entry:
  - **Criteria**
    - From this section, identify the field and/or function criteria which should be evaluated
  - **Filter Expression**
    - For complex criteria, use this section to build out an expression leveraging AND/OR groupings
      - BREeze assumes that all criteria are joined via "and". Meaning if there are two criteria entries, it would be interpreted by default as: 'Criteria 1' AND 'Criteria 2'. If the criteria should be interpreted otherwise (for example, 'Criteria 1' OR 'Criteria 2'), use this section to define that logic.
- The bottom half of the screen is where you'll define the actions to take place on the record should the criteria be met for the specific rule entry
  - **Field Updates**
    - Define the actions (either field update or execute code action) that will take place if the record meets the criteria

Defining Create Object Entries

- Click the ***Edit*** button on the left side of the rule entry

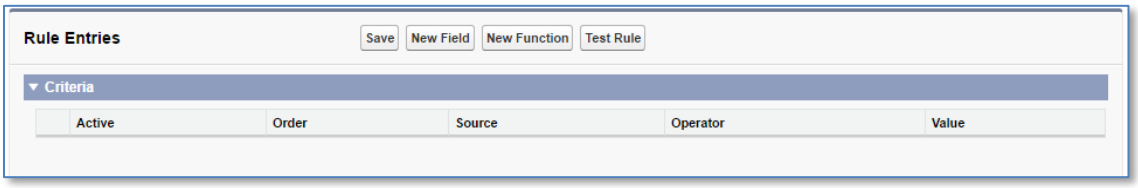


- The resulting screen allows you to define the rule:



- **Actions**
  - Define the fields that will be set upon object creation

Defining Criteria

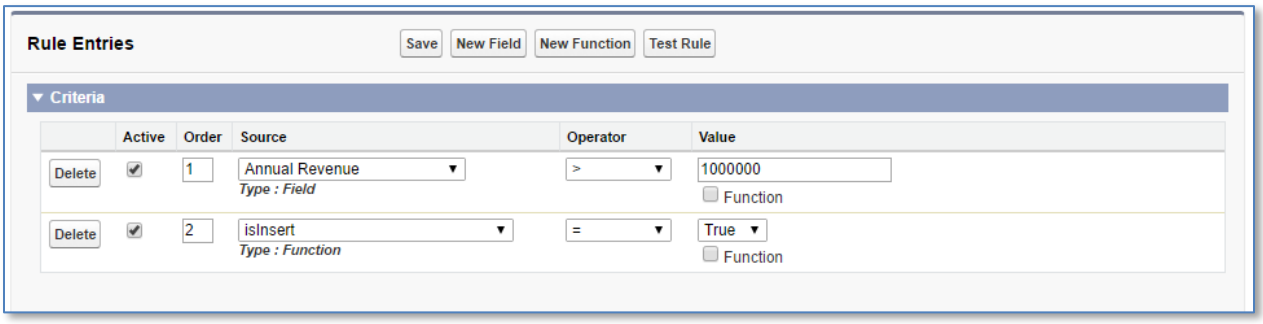


- Click on either the ***New Field*** or the ***New Function*** button
  - **New Field**
    - Allows you to select any field from the object which your initial rule was set up against and evaluate that against a single value, multiple values or evaluate that against a function.
      - Example: Annual Revenue > 10000000
  - **New Function**
    - Allows you to evaluate a function against a set single value, multiple values or evaluate that against another function.
      - Example: isInsert = True
    - See ***Functions*** section of this guide for more information
- The standard operators are as follows:

Operator	Definition
=	Equals
!=	Does Not Equal
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To
isChanged	Determines if field has been updated and returns True or False
contains	Determines if the value exists within the selected field
not contains	Determines if the value does not exist within the selected field
in	Allows for multiple values to be entered without using OR and compares the values to the selected field

not in	Allows for multiple values to be entered without using OR and compares the values to the selected field
in (Case Sensitive)	Allows for multiple values to be entered without using OR and compares the values to the selected field and also enforces case sensitivity
isNull	Determines if there is no value in the selected field
StartsWith	Evaluates whether the field begins with the value

- Enter all appropriate fields/functions that should be used in the first evaluation and click the ***Save*** button

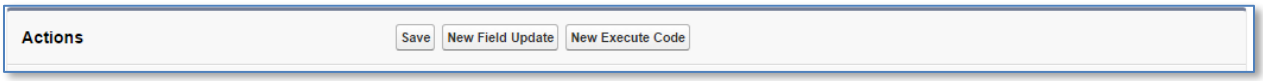


Using Filter Expressions

- Next determine whether your criteria requires an advanced filter expression
  - If all your criteria should be joined with AND's, then there is no need to use a filter expression as it is assumed
    - Example: 1 AND 2 AND 3
  - If your criteria requires grouping and combinations of AND's & OR's, enter your criteria into the Filter Expression text box and then check the Enable Advanced Filter checkbox
    - Examples: "1 OR 2", "(1 AND 2) OR (3 AND 4)", "(1 OR 2) AND 3"
    - **Important Note:** Once you have enabled a filter expression, you will not be able to add or delete rule criteria. If you find that you need to add or delete criteria after enabling the advanced filter, simply uncheck the Enable Advanced Filter checkbox and you will be able to add or delete as appropriate. Before re-enabling the advanced filter, you will need to update your Filter Expression to reflect the additions/removals.

Defining Field Actions

Finally, you will need to select the field updates to take place or functions to call should a record match the criteria defined above. There are two distinct types of field actions that can take place:

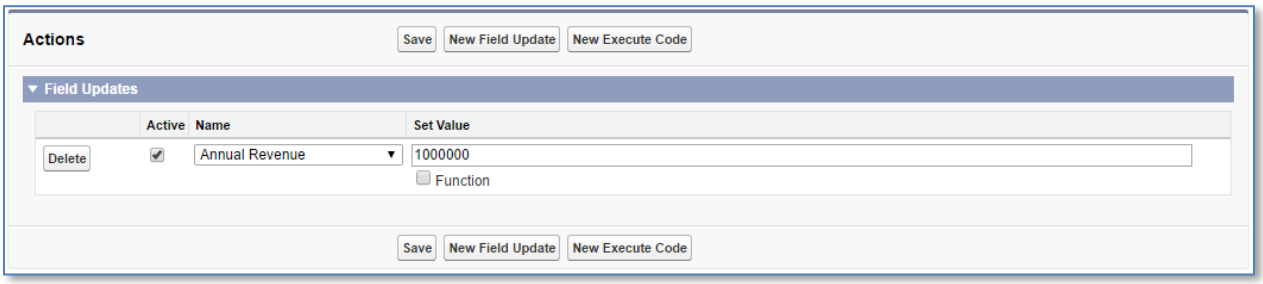


New Field Update

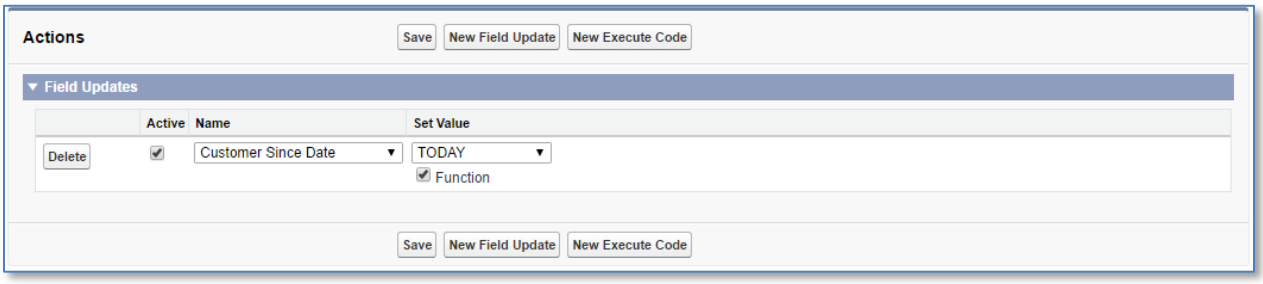
New Field Update should be used in the case that you want to set a specific field on the object with a defined value or have a field set via a function (either standard out of the box or custom)

- Under the Actions section, click the ***New Field Update*** button
  - In the Name field, select the field which should be updated
  - In the Set Value field, depending on whether you selected a text/number, picklist (single or multi), enter a static value(s) which the field should be updated to
    - In the case of a lookup type field (Owner, Record Type, etc.), click the magnifying glass icon to the right of the value field and select the appropriate value via the pop-up window
    - There is also the ability to set the value of a field to a function by simply checking the Function checkbox under the Set Value field. As an example, by using the "Stamp Rule Text" function, it allows you to set the value of a field to the name of the rule detail where the criteria was met

Standard Field



Function

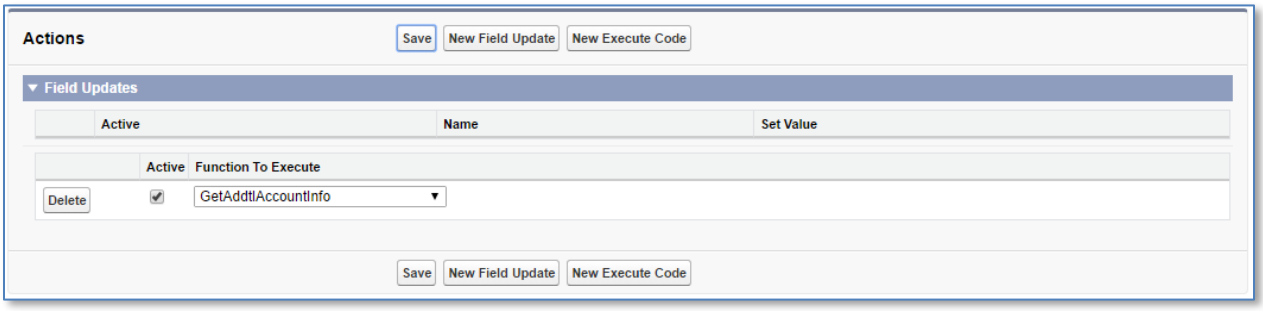


- If you have multiple fields that should be updated, simply click the ***New Field Update*** button again to add as many additional fields as necessary

New Execute Code

Execute Code should be used in the case that you want to have a function called if the criteria is met, but instead of setting a field on the record, instead the Apex code for that function should just be executed.

- Under the Actions section, click the ***New Execute Code*** button
  - In the Function to Execute field, select the function that should be executed and click *Save*
  - If the function that has been selected requires extensions, click the wrench icon to the right of the function that was selected and enter the appropriate values into the function extension pop-up window and click *Save*



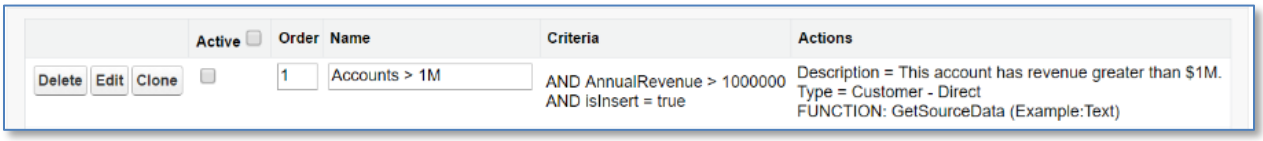
- If you have multiple functions that should be called, simply click the ***New Execute Code*** button again to add as many additional function calls as necessary

Important Note: You can enter any combination of both field updates & execute code entries to a single Actions section. If both field updates & execute code entries are present within the Actions section, should a record meet the criteria for this rule entry, all actions will take place.

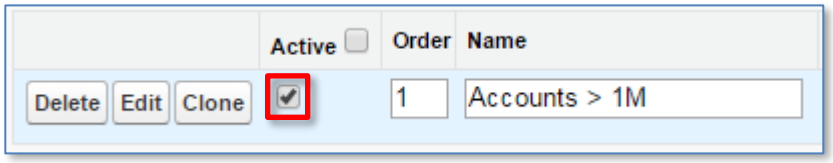
Finalizing Your Field Actions

- Once you have entered all your field updates and/or execute code entries, be sure to click the ***Save*** button at the top or bottom of the Actions section
- Next, click on the ***Back to Rule Maintenance*** link at the top of the screen

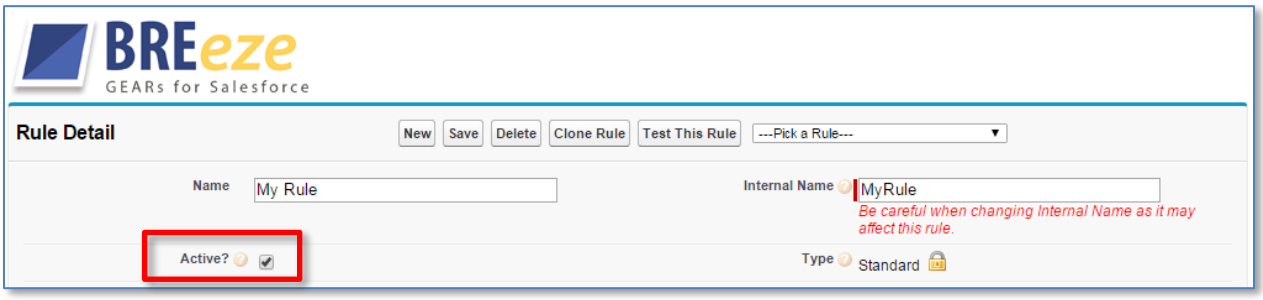
- This will bring you back to your original rule set-up screen however now you should see the rule entry that you just defined with both the criteria and the field actions visible



- From here, you will want to continue to define all your rule entries like the one that was just created
    - If you have a rule that is like another, simply use the ***Clone*** button to copy the rule entry definition. From there, click the ***Edit*** button on the cloned entry and adjust as appropriate.
- Once all your rule entries are complete, be sure to check the Active checkbox within the rule entry itself or the checkbox to the right of the Active checkbox which selects all. Finally, be sure to click the ***Save Entries*** button after setting the rule entries to active.

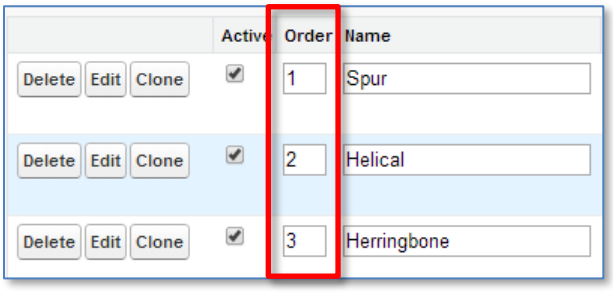


- When you are ready to have records begin flowing through your rule, ensure that the Active checkbox is checked for the overall Rule

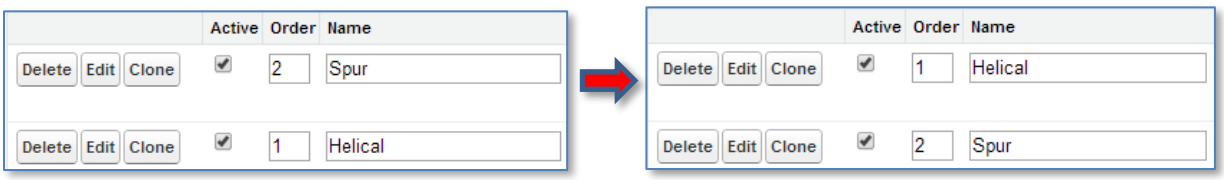


Ordering

BREeze evaluates the rule entries in the order which they are noted from the Rule Detail screen. This order can be altered at any point if you decide that a specific rule or rules should evaluate before others.



To adjust the ordering, simply adjust the number(s) to the desired evaluation sequence and click the ***Save Entries*** button at the top or bottom of the Rules Entries section of the screen

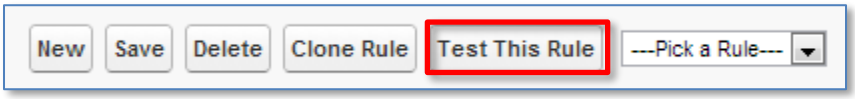


*This can be done one entry at a time or several records simultaneously.

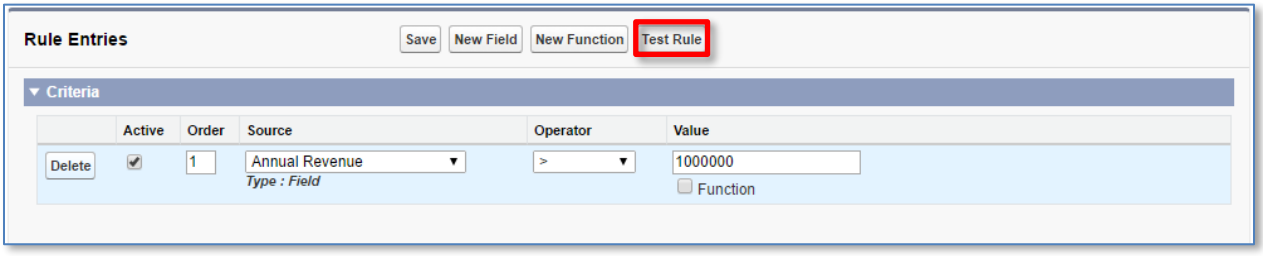
Rule Testing

Before deploying the rules you created in BREeze to a "live data mode", you'll want to test them to ensure they are updating not only the field(s) that you expect but also the populating value(s) you are expecting.

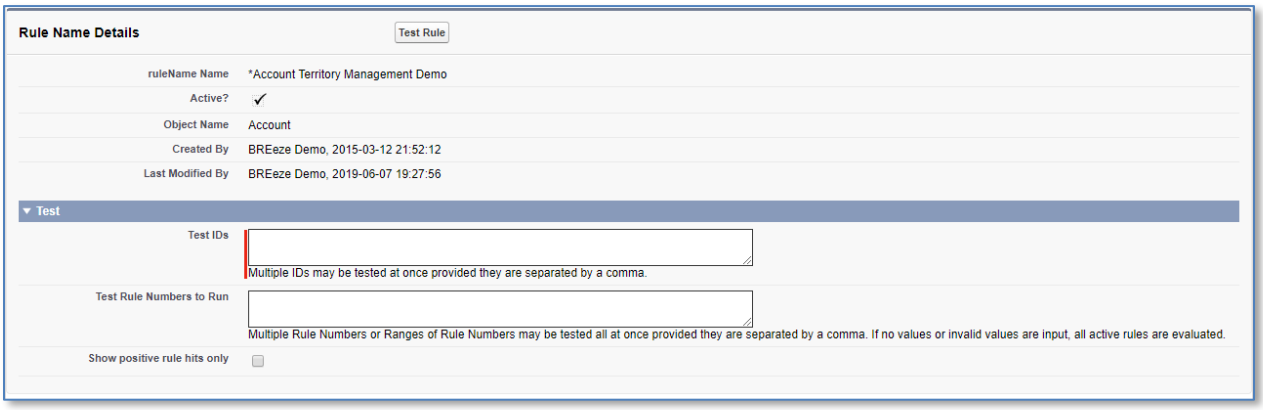
There is the ability to test your rule(s) in two places within BREeze. Each test screen will result in the same test executing so it's a matter of convenience where you execute the test from. The first is found on the main Rule Detail section of BREeze:



And the second is located on the Rule Entries screen above where you define the criteria for your rule :



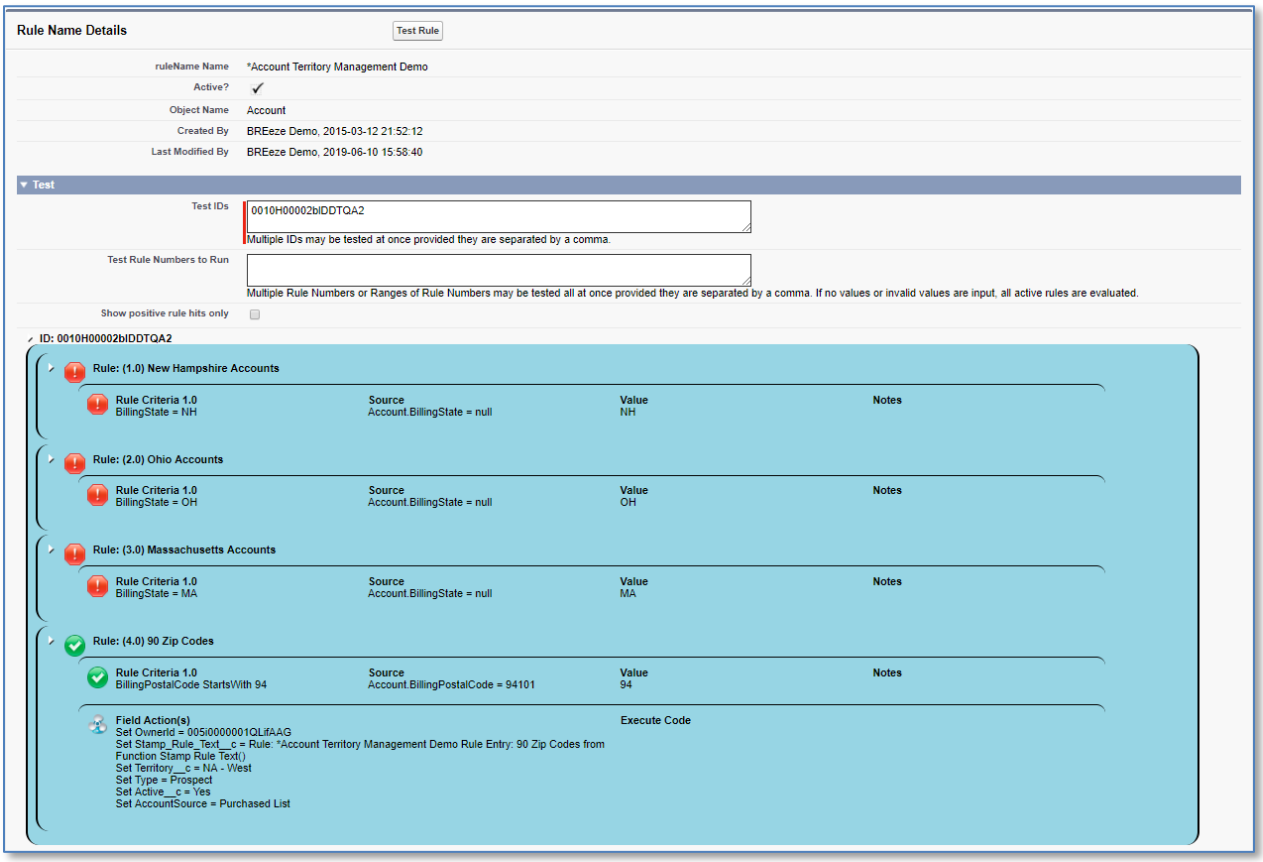
After clicking on either button, you'll be brought to the BREeze Test Rule page



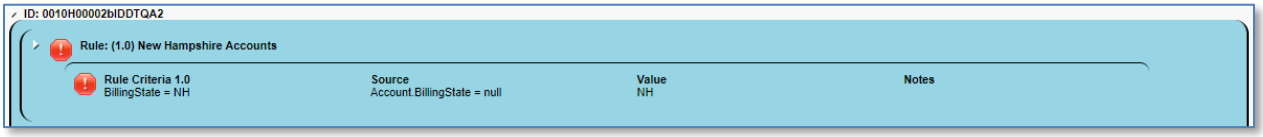
- ***Test IDs***
  - Enter a single Salesforce ID (15 or 18 characters) or multiple Salesforce ID's (separated by commas) for an existing record(s) of the object which you've created the rule for
- ***Test Rule Numbers to Run***
  - If you'd like to evaluate the only a subset of rules for a given record(s), enter the Rule Detail number (Ex. 4), Rule Detail Numbers (comma separated – Ex. 3, 5, 6) or range of Rule Detail numbers (Ex. 4-7)
- ***Show positive rule hits only***

- If enabled, the results from the Test this Rule will only show where the record(s) evaluated to true and all instances where rules evaluated to false will be removed from view

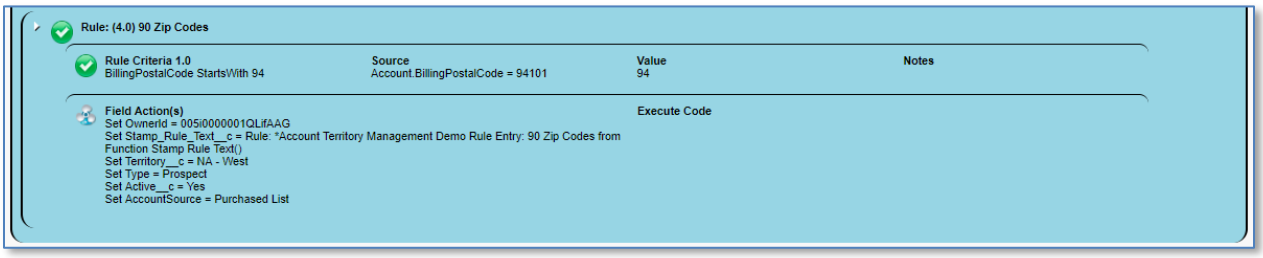
Depending on the number of rule entries, rule criteria, rule actions & when the criteria are first met; the results of your tests will vary. For example, below is a rule which had 6 rule entry records – each having 1 line of criteria and several actions.



From the test results, you can see the individual rules which were evaluated as either true or false depending on whether your test record met the criteria. For example, Rule 1 below called 'New Hampshire Accounts' ran first and evaluated whether the Billing State was equal to "NH". The actual data in the Billing State field was "CA" so the rule ultimately resulted in a false evaluation (indicated by the red exclamation mark icon).

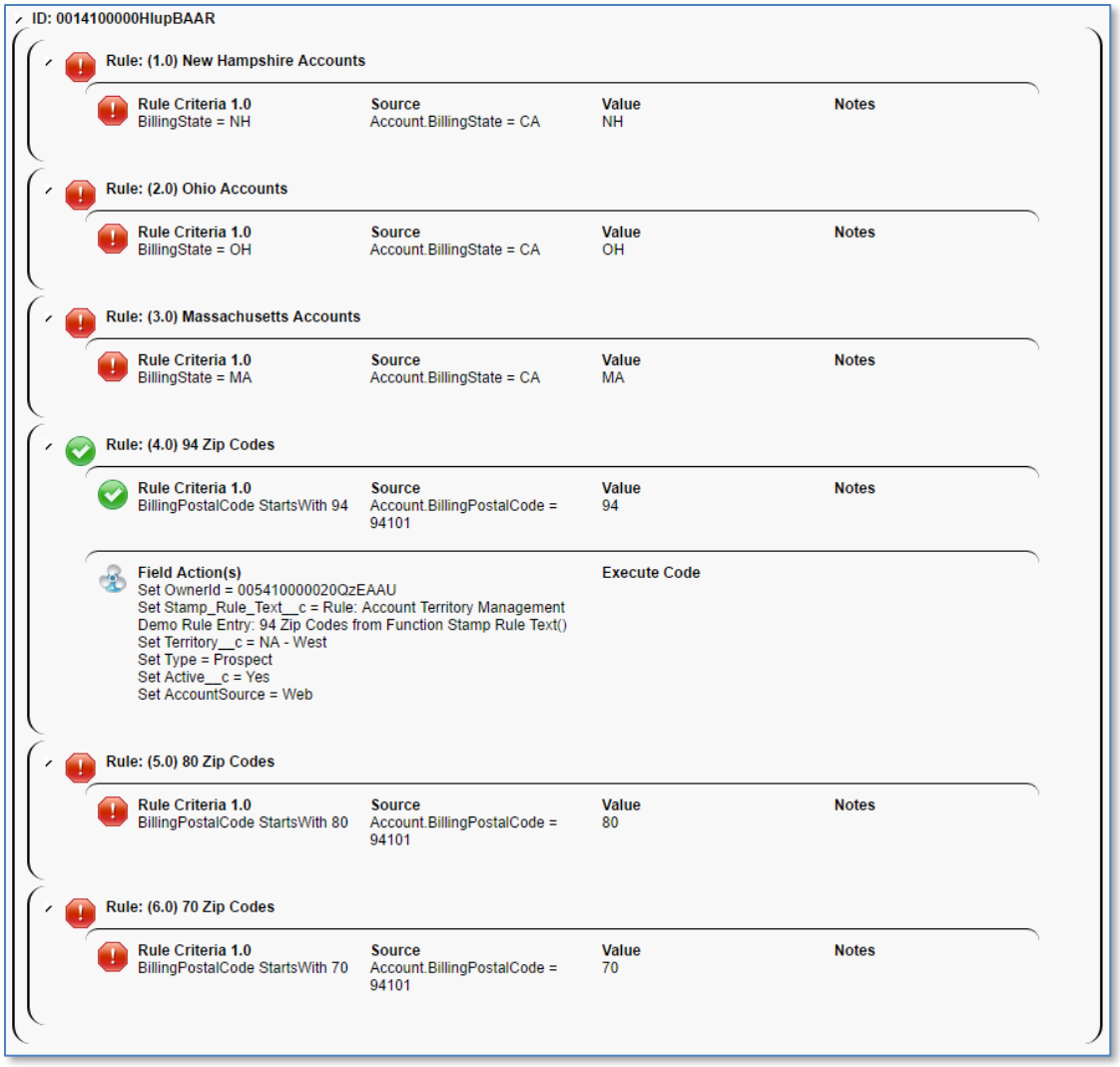


When a rule evaluates to false, BREeze moves to the next rule evaluates that in the same manner until BREeze ultimately hits a rule which evaluates to true (indicated by the green check mark icon).



If the '**Stop On First Match**' field is set to "true" (i.e. checked) on the main Rule screen, BREeze stops evaluating and executes on the actions that are associated to that rule detail entry. So even though there were 6 rule entries in this example, because the 4th rule's criteria was met, BREeze stopped evaluating any further. In this case, it sets a custom field called Territory, the Type, the Source, Active flag, the Owner and finally stamps a text field which the name of the rule entry which ultimately the record met the criteria for. More information on that functionality can be found in the ***Functions*** section of this document.

If the '**Stop On First Match**' field is set to "false" on the main Rule itself, BREeze would continue to run through the remainder of the rules and evaluate/assign should any of the other rule criteria evaluate to true. As you can see from the example below, Rule 4 evaluated to true and then BREeze continued to evaluate rules 5 & 6.



Important Notes:

1.) The information and screenshots above are provided via an add-on package that can be installed on top of the core BREeze package. To download this add-on package, go to the following URL and follow the same installation steps that you took for the core application:

- https://login.salesforce.com/packaging/installPackage.apexp?p0=04t41000002Zgoi

2.) This is only a test so the record(s) you ran through the test will not actually have the updated information that is outlined in the Actions section. This will not occur until your rule & rule entry have been activated and the record itself met the criteria to be evaluated by BREeze.

3.) If a record passes through the BREeze rules and does not evaluate to true for any of them, no action will be taken. In some cases, the recommendation is to put a "catch all" rule at the end of a given rule s set to easily identify the records that have run through the rules without matching any of the criteria.

BREeze Functions

Below is a list of the current functions which are available in BREeze for both the field criteria & update/assignment sections of the Rule Detail screen.

Criteria

Function	Definition

AddDaysToToday	Returns a date value based on the current date and the extensions which are defined:
	• *Include weekends (true/false)*
	  ○ Determines whether weekends should be included in the date calculation.
	• *End on a weekend (true/false)*
	  ○ Determines whether the date can fall on a weekend or whether the date should automatically be advanced to the following Monday.
	• *Return type (Date/DateTime)*
	  ○ Determines whether the field type to return is a Date or Date/Time. For this use case, it should always be "Date" as this function is only available for Date type fields.
	• *#days (integer)*
	  ○ Number of days (+/-) from the current date which should be evaluated.

AddDaysToTodayDT	Returns a date/time value based on the current date/time and the extensions which are defined:
	• *Include weekends (true/false)*
	  ○ Determines whether weekends should be included in the date calculation.
	• *End on a weekend (true/false)*
	  ○ Determines whether the date can fall on a weekend or whether the date should automatically be advanced to the following Monday.
	• *Return type (Date/DateTime)*
	  ○ Determines whether the field type to return is a Date or Date/Time. For this use case, it should always be "DateTime" as this function is only available for DateTime type fields.
	• *#days (integer)*
	  ○ Number of days (+/-) from the current date which should be evaluated.

NOW	Returns the current date/time

TODAY	Returns the current date
TOMORROW	Returns the current date plus 1 day
YESTERDAY	Returns the current date minus 1 day
getWithinCaseBusinessHours	Evaluates the defined case's business hours
getWithinOrgDefaultBusinessHours	Evaluates the org's default business hours
isInsert	Identifies whether the record was inserted (returns true or false). Can be used to differentiate actions based on whether a record is being inserted or updated.
Contact:getDomain	Returns the domain value from an email address. **Note**: This is only available for 'Contact' object rules.
Lead:getDomain	Returns the domain value from an email address. **Note**: This is only available for 'Lead' object rules.

Actions/Assignment

Function	Definition
Stamp Rule Id	Identifies the ID of the rule which fired to update the current record and places that ID in the field specified. **Note**: Can only be used when the field name selected is of type Id.
Stamp Rule Text	Identifies the name of the rule which fired to update the current record and places that text in the field specified. **Note**: Can only be used when the field name selected is a text field.
TODAY	Sets the current date. **Note**: Can only be used when the field name selected is a date field.
TOMORROW	Sets tomorrow's date. **Note**: Can only be used when the field name selected is a date field.
YESTERDAY	Sets yesterday's date. **Note**: Can only be used when the field name selected is a date field.
NOW	Sets the current date/time. **Note**: Can only be used when the field name selected is a date/time field.
Lead:getDomain	Sets the domain from the email address on the record. **Note**: This can only be used on rules for the Lead object.
Contact:getDomain	Sets the domain from the email address on the record. **Note**: This can only be used on rules for the Contact object.
getWithinOrgDefaultBusinessHours	Sets the instances default business hours.
getWithinCaseBusinessHours	Sets the business hours for the case. **Note**: This can only be used on rules for the Case object.
AddDaysToToday	Sets a date value based on the current date and the extensions which are defined: • *Include weekends (true/false)* ○ Determines whether weekends should be included in the date calculation. • *End on a weekend (true/false)*

	○ Determines whether the value can land on a weekend or whether the date should automatically be advanced to the following Monday. • *Return type (Date/DateTime)* ○ Determines whether the field type to return is a Date or Date/Time. For this use case, it should always be "Date" as this function is only available for Date type fields. • *#days (integer)* ○ Number of days (+/-) from the current date which should be evaluated.
AddDaysToTodayDT	Sets a date/time value based on the current date/time and the extensions which are defined: • *Include weekends (true/false)* ○ Determines whether weekends should be included in the date calculation. • *End on a weekend (true/false)* ○ Determines whether the date/time can land on a weekend or whether the date should automatically be advanced to the following Monday. • *Return type (Date/DateTime)* ○ Determines whether the field type to return is a Date or Date/Time. For this use case, it should always be "DateTime" as this function is only available for Date Time type fields. • *#days (integer)* ○ Number of days (+/-) from the current date which should be evaluated.

Custom Functions & Function Extensions

Custom functions can be created within BREeze to further extend the out-of-box BREeze functionality. Some examples of this may include:

- Retrieving a field value from an object which is related to the core object being processed in the rule
- Adding or removing a public group from a user record
- Matching the Company value from a Lead record to an existing Account record

Some of these functions have been created in the past by GearsDesign however BREeze allows you to create any function that may suit your business needs.

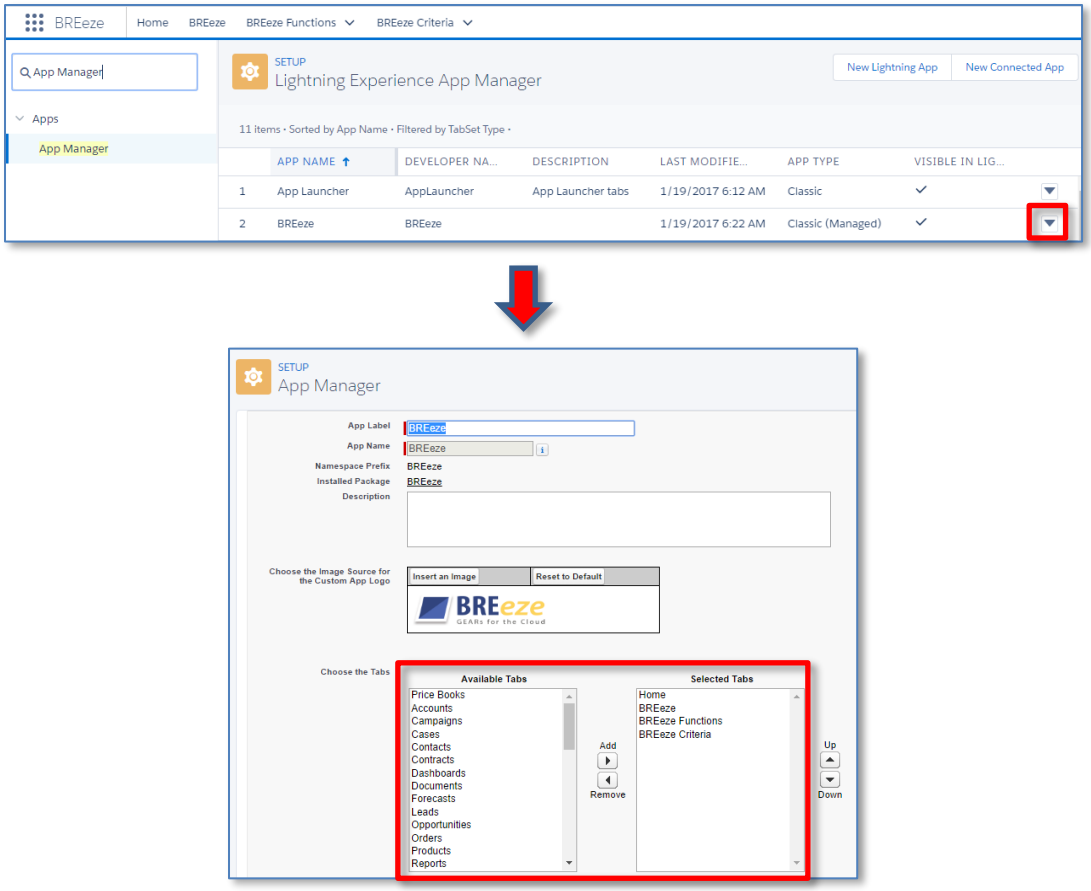
Important Note: Development will be required for all custom functions as you'll need that function to call an existing Apex Class.

Creating Custom Functions

To create a custom function, follow these steps:

- Navigate to the BREeze Functions tab
  - *Classic*:
    - If not already available within the BREeze app, it will be available for selection within the All Tabs screen

  - *Lightning Experience*:
    - If not already available within the BREeze App, it will be available for addition within the App Manager setup accessed via the main Setup screen



- From the BREeze Functions tab, click the ***New*** button
- From the resulting screen, at a minimum enter the following:
  - **Function Name**
    - Enter a descriptive name for the function which you are building
  - **isActive**
    - Indicates that the function is active and can be used in a rule
  - **Return Type**
    - Indicates the field type which is returned by the function (double, string, date, etc.)
  - **Sort Order**
    - Indicates a numeric ordering sequence which controls where the function will appear in a dropdown list
  - **Class Name**
    - Reference to the Apex Class which executes the function logic
  - **Available in Criteria**
    - Indicates that the function will be available for use under the Criteria section of the Rule Details screen
      - This applies to only 'Standard' rules as criteria is not defined for 'Create Object' rules
  - **Available in Assignment**

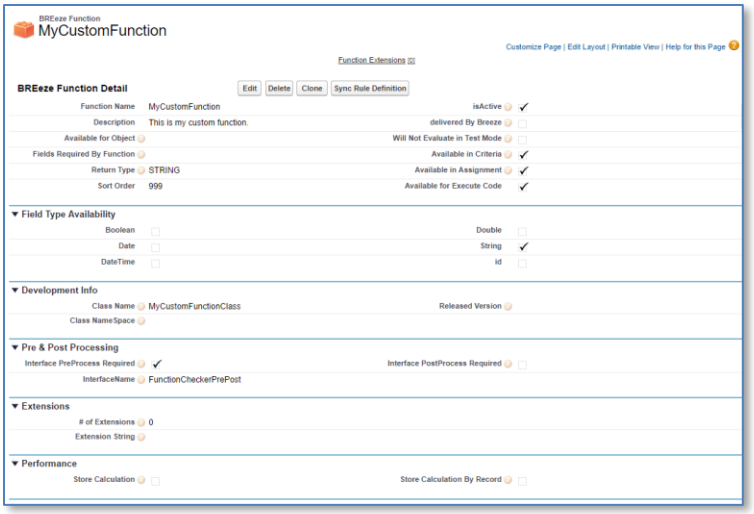
        - Indicates that the function will be available for use under the Actions (Assignment) section of the Rule Details screen
            - This applies to both 'Standard' & 'Create Object' rules
    - **Available for Execute Code**
        - Indicates that the function will be available for use under the Actions (Assignment) section of the Rule Details screen when clicking on the *Execute Code* button
    - **Field Type Availability Section**
        - Indicates which field types the function will be available for:
            - **Boolean** → Checkbox field type
            - **Date** → Date field type
            - **DateTime** → Date/Time field type
            - **Double** → Number field type
            - **String** → Text field type
            - **Id** → Lookup/Id field type
- Optionally, you will likely want to consider populating the following fields:
    - **Available for Object**
        - Indicates which object(s) the function is available for use on
            - **Note**: This can simply be left null if the function can span across all or more than 1 Salesforce objects
    - **Interface PreProcess Required**
        - This should be checked in the case that the function needs to execute a pre-processing step prior to evaluating the criteria.
            - For example, a custom function which does an account name match from a lead record to an account record. To evaluate whether the function returns a true or false value, first the value from the lead record being processed in BREeze must be compared against all existing account records.
        - If this field is selected, you must also select "FunctionCheckerPrePost" in the 'InterfaceName' field.
    - **Interface PostProcess Required**
        - This should be checked in the case that a function needs to execute a post-processing step after the criteria has been met
            - For example, a custom function which adds or removes a public group from a user record. To execute that add/remove of the public group, you use a post-process function to accomplish this.
    - **InterfaceName**
        - Indicates whether the standard or modified Function class template will be used. If 'Interface PreProcess Required' is "True" then, this field must be set to "FunctionCheckerPrePost"
            - FunctionChecker – Uses the standard Function class template
            - FunctionCheckerPrePost – Extends the FunctionChecker class to allow for a pre- and post- process in the template
    - **Store Calculation**

- If the function is not data specific, performance may be improved by executing the function once, storing the result, and referencing the results on each additional evaluation.

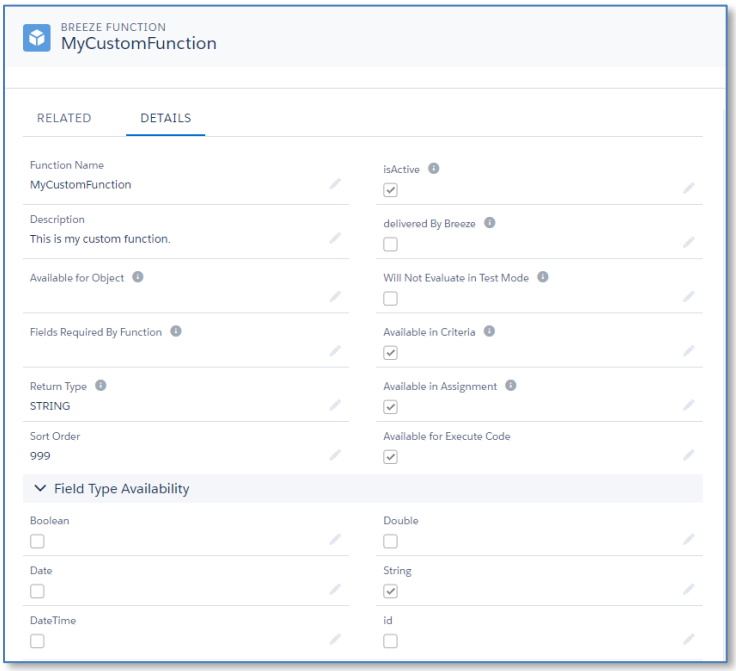
- **Store Calculation By Record**
  - If the function is data specific, performance may be improved by executing the function once per record, storing the result, and referencing the results on each additional evaluation.

- Finally, click the ***Save*** button
- The result should look like something like this:

Classic:



Lightning Experience:



Important Note: If you are upgrading your version of BREeze from a prior release, some of these fields may not be available by default and therefore will need to be added manually to the layouts.

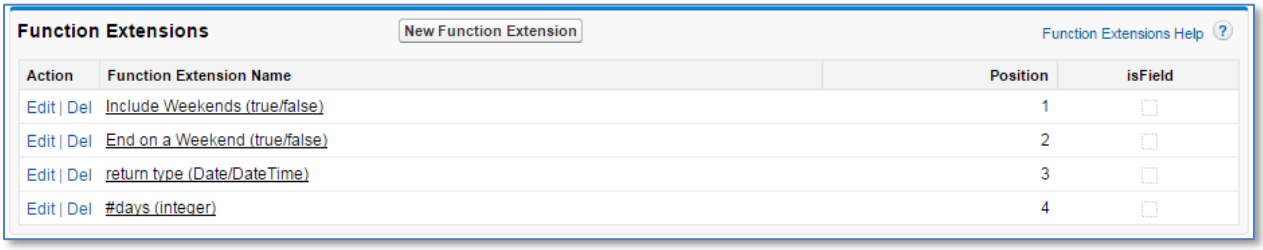
Extending Functions

In some cases, you may want to further extend functions by passing variables into them to either evaluate or set a specific value. This allows for greater flexibility when building functions and eliminating the need to create multiple functions for the same purpose where only the variable being passed into the function varies. With BREeze 3.3, the ability to add ***Function Extensions*** was introduced, which allows you to not only streamline the functions being created, but also allows for greater flexibility when your business requirements change.

An example of this, the out-of-box 'AddDaysToToday' & 'AddDaysToTodayTD' functions leverage extensions by taking the standard TODAY function and manipulating that value based on the variables that are entered.

When looking at the 'AddDaysToToday' function, you will see a related list which displays the four extensions being used in this function:

Classic:



Lightning Experience:

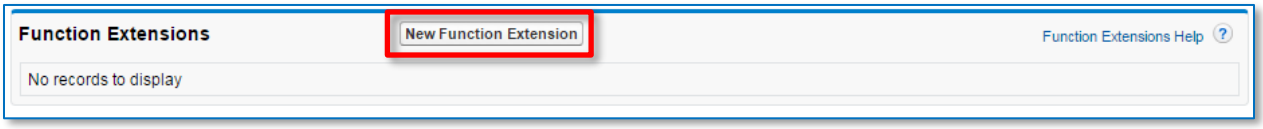
Based on the inputs entered within the BREeze rule itself, you can determine the value that is returned and used in either criteria or assignments.

Creating Function Extensions

If you have created a function which has the need for additional function extensions to be added, follow these steps:

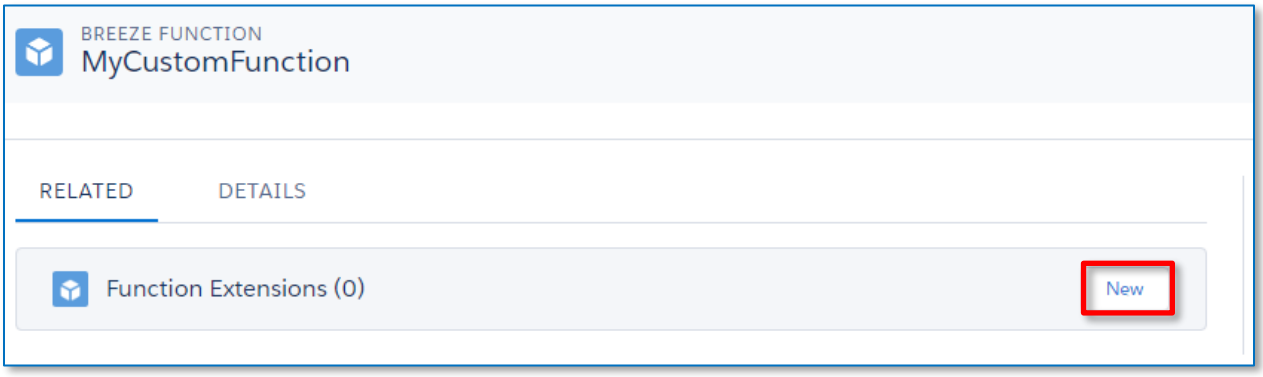
Classic:

- Navigate to the custom function which you would like to add extensions to
- Scroll down to the bottom of the function screen to 'Function Extensions' related list
- Click the *New Function Extension* button



Lightning Experience:

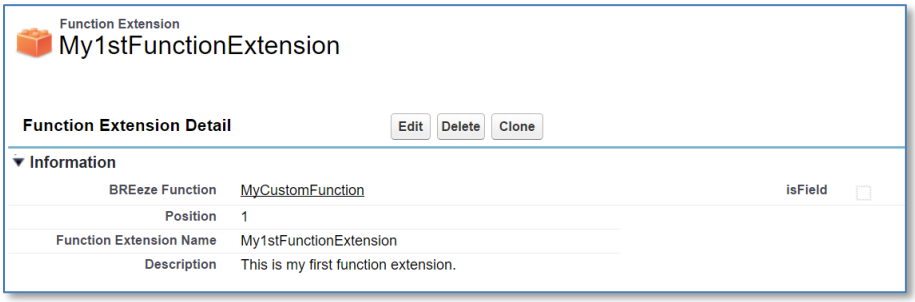
- Navigate to the custom function which you would like to add extensions to
- Click on the Related tab
- Click on the *New* button to the right of Function Extensions



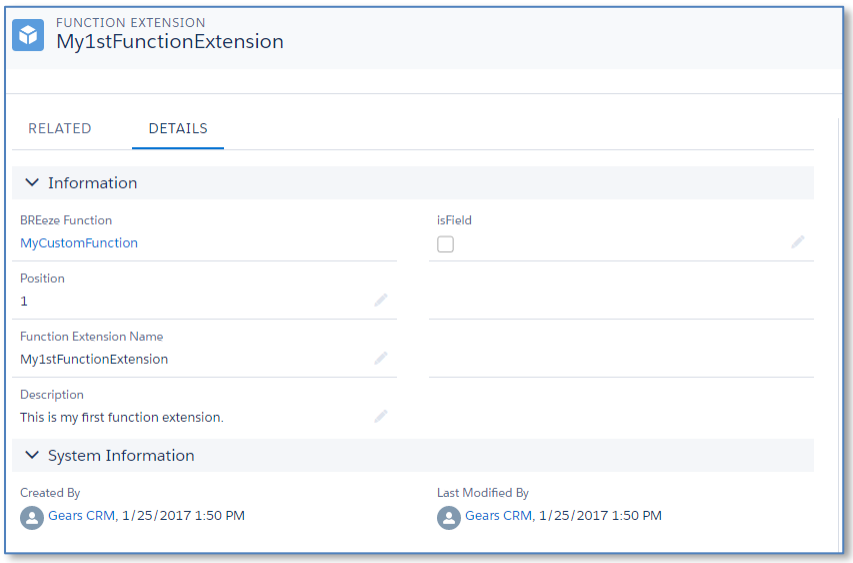
- From the resulting screen, enter the following:
  - **Position**
    - This is the position which this extension should be passed into the custom function logic. Additionally, this defines the order in which the extensions will appear when defining the inputs on the rule itself.
  - **Function Extension Name**
    - This is the name of the function extension itself. This value will appear on the function extension entry screen, so you will want to enter something that is easy to understand when setting up your rules.
    - **Note**: It is also helpful to add information in the name as to the type of data that is expected in this extension. For example:
      - "…(true/false)"
      - "…(Date/DateTime)"

    - "…(integer)"
    - Etc.
  - **Description**
    - This field is optional however it may be helpful to enter information here as to why this extension is being used.
  - **isField**
    - By default, this is unchecked however in the case that you want to reference a specific field instead of a value, set this to true
  - **BREeze Function**
    - This should be pre-populated based on the function which you clicked the ***New Function Extension*** button from
- Finally, click the ***Save*** button or ***Save & New*** button (if you are creating additional extensions)
  - If creating multiple extensions, simply repeat the above process
- The result should look like something similar to this:

Classic

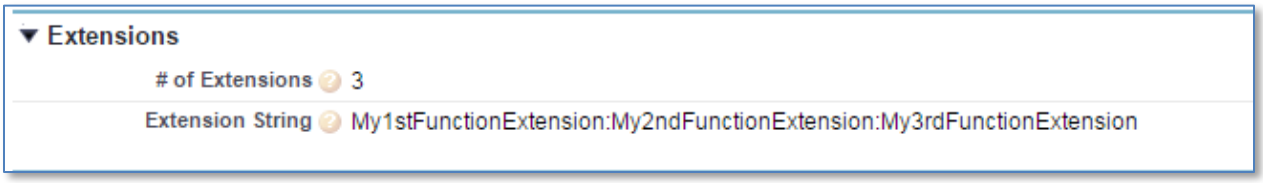


Lightning Experience

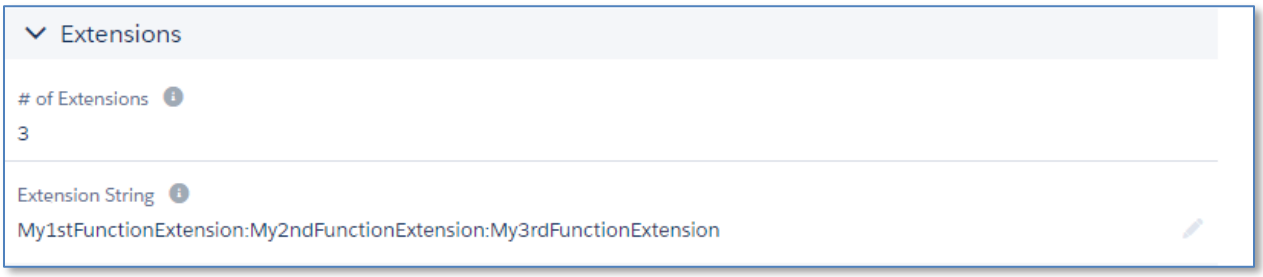


- Once all extensions have been created for the function, navigate back to the function itself
- Within the 'Extensions' section of the function itself, the information has been updated to reflect the extensions which were just added

Classic

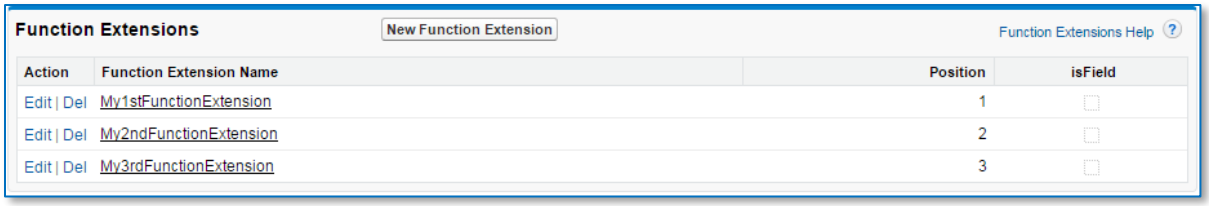


Lightning Experience

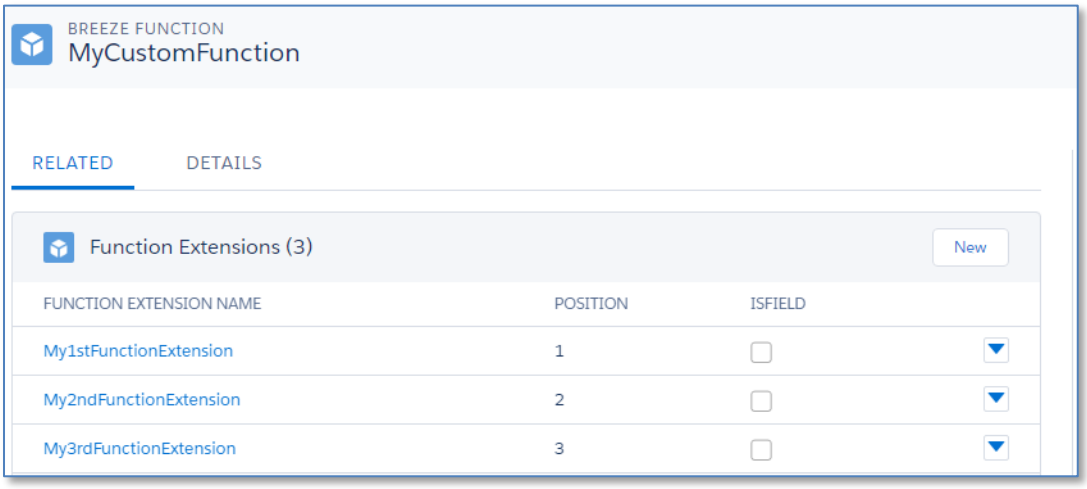


- Additionally, the related list will display all the extensions that are tied to this specific function:

Classic

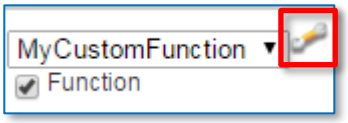


Lightning Experience

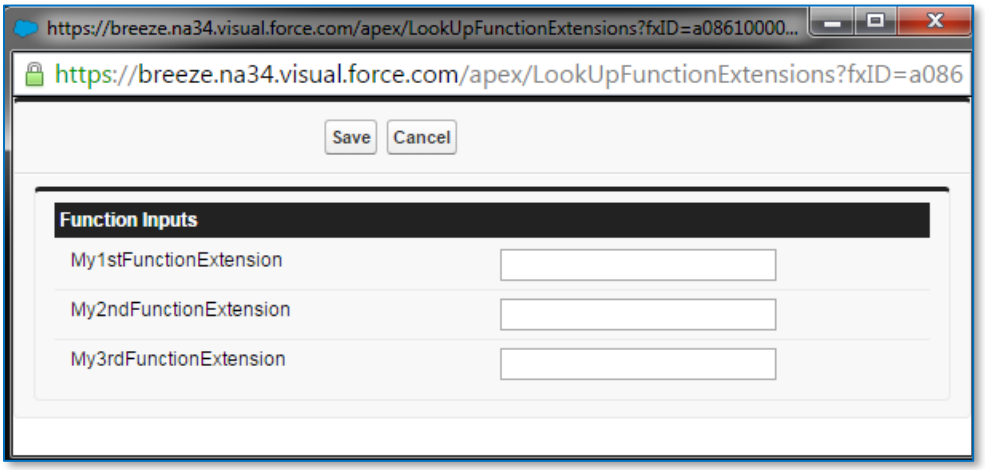


Setting Function Extension Values in Rule Criteria or Assignment

From the Rule Detail screen itself, when you select a function which has extensions available for it, a wrench icon will appear to the right of the value.



Upon clicking that wrench, a subsequent pop-up window will be shown. Within this pop-up window, it will display the function extensions available for that specific function:



By entering values into each of the extensions, these variables will be passed into the function and the rule will either be evaluated using these values or set a specific field based on these values.

BREeze Field List Displays

By default, the BREeze user interface when setting up Rule Entries will display the Field Label. However, there are some circumstances when it is helpful to display both the Field Label & API Field Name to configure your rule sets more easily. For example, when you have two fields on the same object with the same Field Label but different API Field Names. To alleviate these scenarios, BREeze comes with the ability to adjust the Rule Entries user interface to show both the Field Label & API Field Name.

> *** **Important Note** ***
>
> We highly recommend that if you enable this functionality within your BREeze installation, you should not later disable it. In the scenario where multiple fields exist with the same name, if this setting is toggled on and then off, it can potentially lead to the other field with the same name being set thereby incorrectly defining the criteria or action for the rule entry.

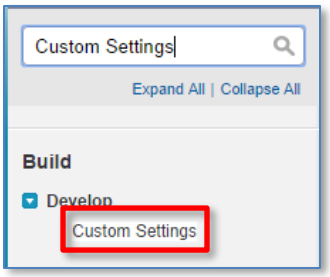
Modifying Field List Displays

To modify the BREeze field list displays on Rule Entries, follow these steps:

Classic:

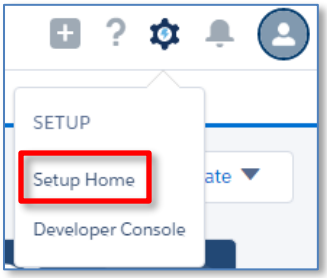
- From anywhere within Salesforce, click on Setup from the top right-hand corner

- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Build → Develop → Custom Settings)

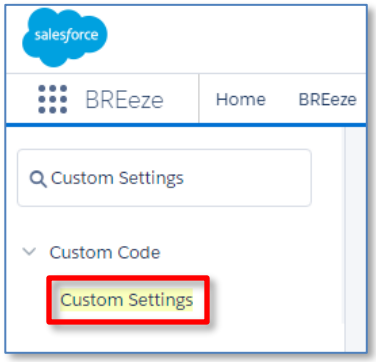


Lightning Experience:

- From anywhere within Salesforce, click on the gear icon and then 'Setup Home' from the top right-hand corner

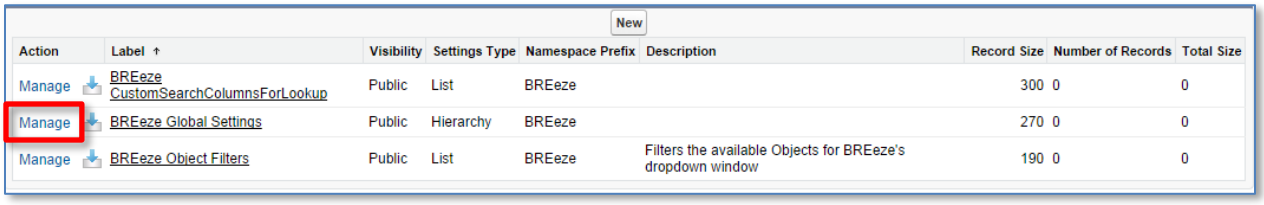


- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Custom Code → Custom Settings)

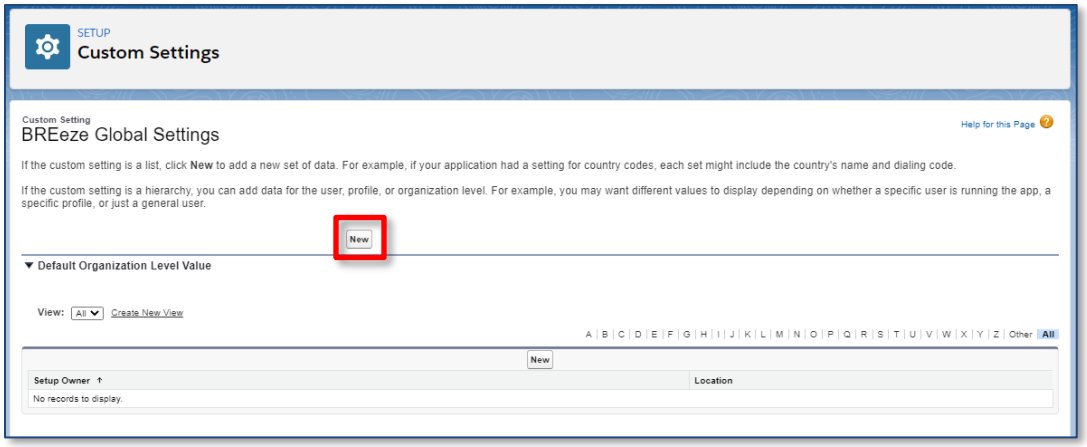


Classic & Lightning Experience:

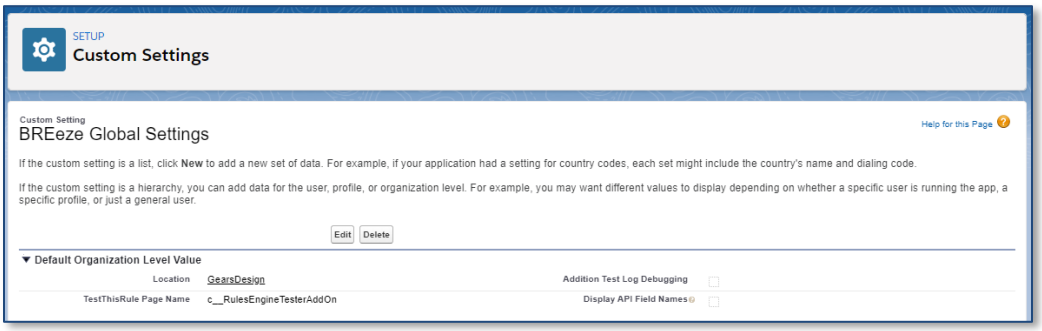
- From the resulting screen, click on the 'Manage' link to the left of the BREeze Global Settings custom setting



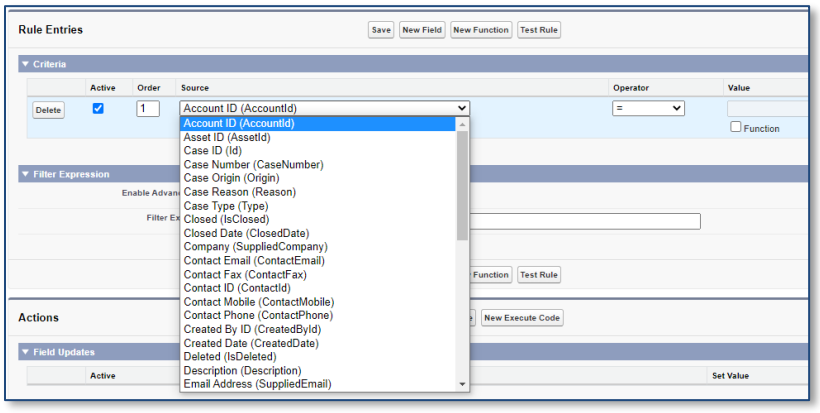
- The following screen will show you the default custom setting record
  - If there is no default record, click on the New button found above the text "Default Organization Level Value". Do not click the New button found above the related list



- By default, the 'Display API Field Names' value will be false (unchecked)

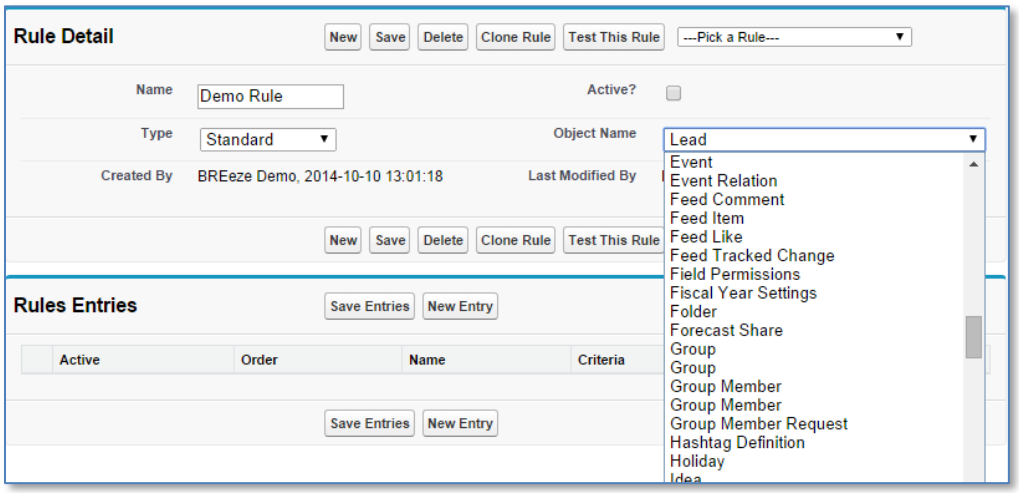


- To change the display of the rule entry screen so that it shows both the Field Label & API Field Name, update the value to true (checked)
- After updating the value, navigate back to any BREeze Rule Detail screen and the results of the dropdown in either the Criteria or Actions section will now display the label & API name



BREeze Object Filtering

Object filtering allows you to restrict the objects which appear in the Object Name dropdown menu from the main BREeze page.



There are two main use cases for this functionality: 1.) your instance has a large number of objects available and you'd like to simplify the values appearing in the dropdown or 2.) Your instance has so many objects available that it exceeds the number of dropdown values allowed by Salesforce.

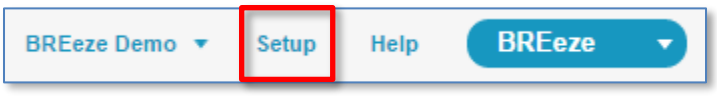
By default, BREeze restricts some of the standard objects which would not normally be used in a BREeze rule set. Should you need to refine this further, you can modify this via the *BREeze Object Filters* custom setting.

Modifying BREeze Object Filters

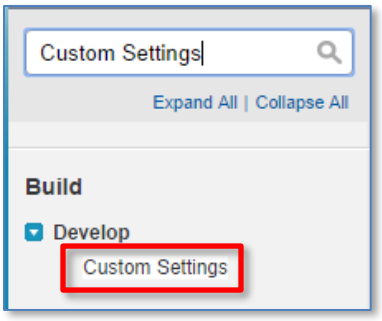
To modify the BREeze Object Filters, follow these steps:

Classic:

- From anywhere within Salesforce, click on Setup from the top right-hand corner

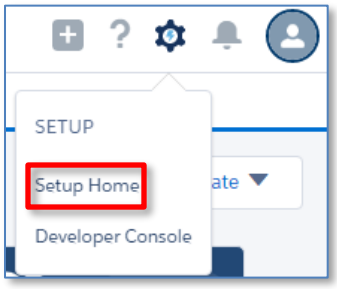


- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Build → Develop → Custom Settings)

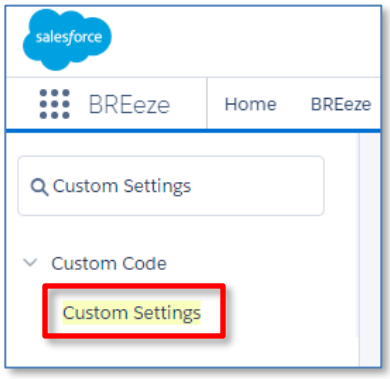


Lightning Experience:

- From anywhere within Salesforce, click on the gear icon and then 'Setup Home' from the top right-hand corner

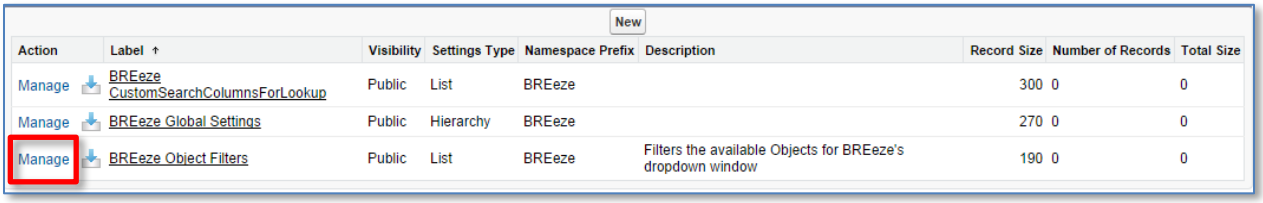


- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Custom Code → Custom Settings)

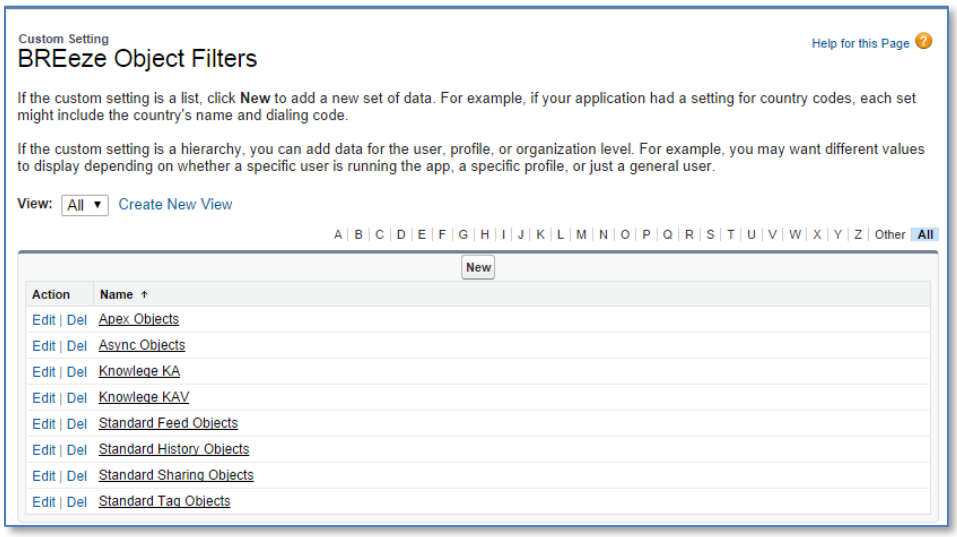


Classic & Lightning Experience:

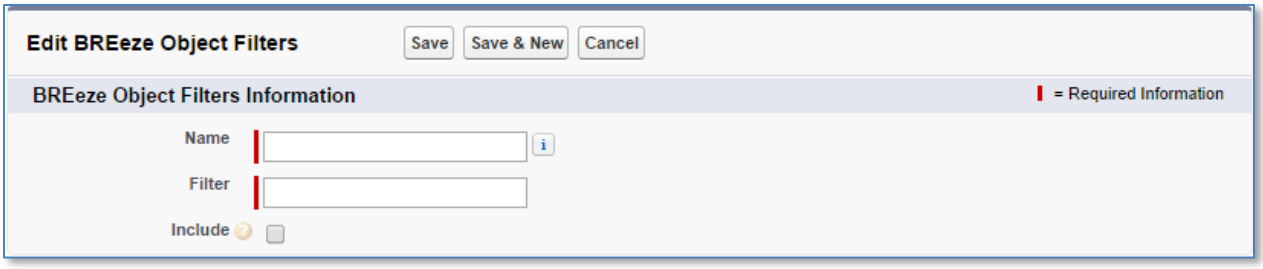
- From the resulting screen, click on the 'Manage' link to the left of the BREeze Object Filters custom setting



- The following screen will show you a list of the current object filters which are in place. By default, the following object filters are in place



- To add an additional filter, click on the ***New*** button
- From the resulting screen, enter values into the following fields:
  - **Name**
    - Enter an intuitive name for the filter
  - **Filter**
    - Enter the filter for which you'd like to apply to the dropdown
      - **Note**: The filter supports regular expressions, so you can use those to exclude larger sets of objects. For example, "(?i).*history", would filter out any objects ending with "history"
  - **Include (optional)**
    - This should be used when you want to include an object or set of objects vs. filtering them out.
      - **Note**: The include checkbox supersedes an exclusion. Meaning, if you exclude a larger set of objects in one filter and then make an inclusion filter for a specific object. That object will surface in the dropdown.



- Once you have entered your filter information, click the ***Save*** or ***Save & New*** button
- Your filters are immediately applied to the BREeze Object Name dropdown

Customizing BREeze Search Columns

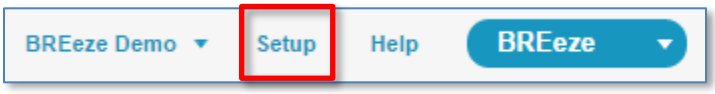
In the case that your instance has several values where the name is similar however they have other attributes which differentiate them (ex. Users, Record Types), there's the ability to adjust the search columns which are returned for any of lookup fields.

Adding & Modifying Search Columns

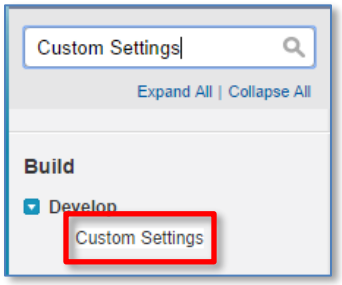
To add/modify the BREeze Search Columns, follow these steps:

Classic:

- From anywhere within Salesforce, click on Setup from the top right-hand corner

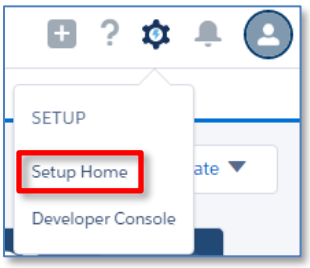


- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Build → Develop → Custom Settings)

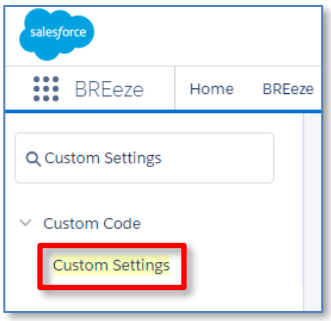


Lightning Experience:

- From anywhere within Salesforce, click on the gear icon and then 'Setup Home' from the top right-hand corner

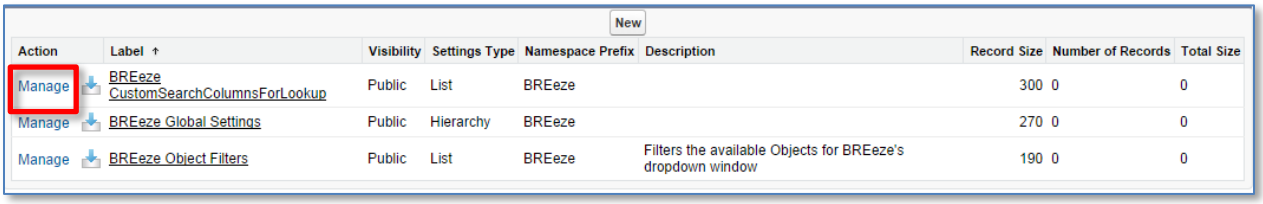


- In the left-hand navigation pane, either search for 'Custom Setting' or navigate to that menu item (Custom Code → Custom Settings)

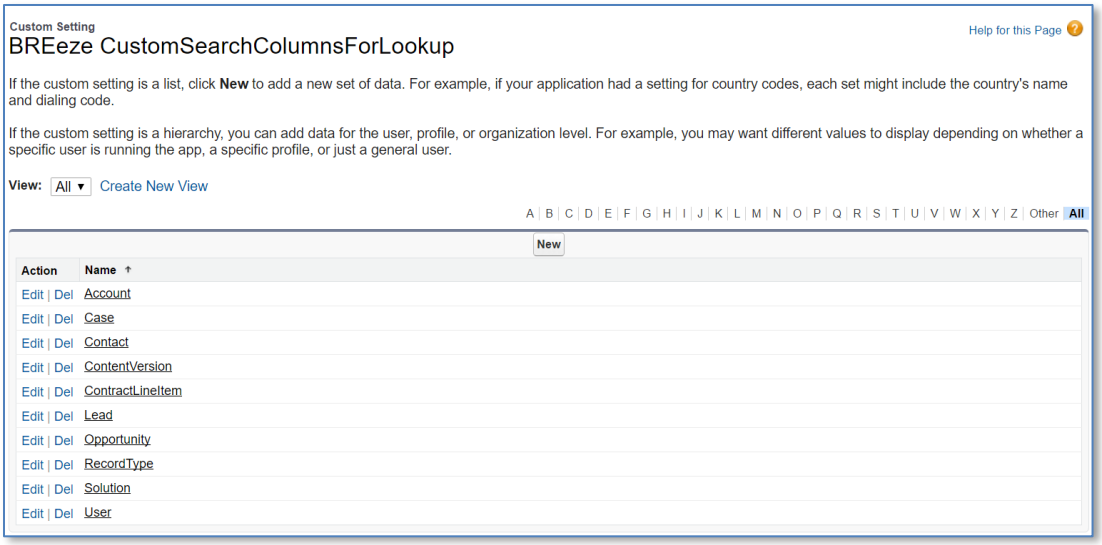


Classic & Lightning Experience:

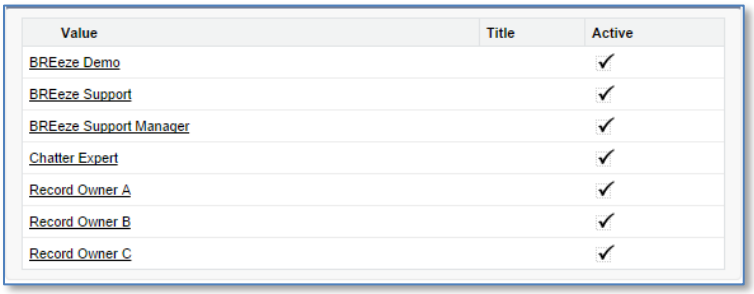
- From the resulting screen, click on the 'Manage' link to the left of the BREeze CustomSearchColumnsForLookup custom setting



- By default, there are entries for Account, Case, Contact, ContentVersion, ContractLineItem, Lead, Opportunity, RecordType, Solution and User search columns available in the custom setting
    - **Note**: These all can be updated to fit the needs of your implementation by simply editing the existing entry



- To create a new custom search column layout, click the ***New*** button
- Enter the following information:
    - **Name**
        - Enter the object you are creating the filter for. In the case of custom objects, be sure to reference the API name and not the name exposed in the main user interface
    - **ColumnsToSelect**
        - Enter the API name for the fields which you would like to appear in the pop-up window separated by commas. For example, "Name,Title,IsActive".
- Once you have entered that information, click ***Save***
- Now when you click a lookup in either the BREeze criteria or actions section, you will see the additional columns noted in the custom setting



Important Note:

For any objects that are either being used as the main rule object or being leveraged in a lookup field within the rule (criteria or assignment) **and** that object does not have a standard "Name" field on that object, you will need to create an entry in this custom setting for that object. BREeze leverages the name field as a standard to return in those lookup fields. If that value does not exist, there is the potential for an error to be returned.

Recommended Setup

As part of a BREeze installation, the recommendation is that the following entries are created within the BREeze CustomSearchColumnsForLookup custom setting. Some of the following entries are simply common use cases however others are due to the error mentioned above whereby some objects do not have a standard 'Name' field available. Again, these can be modified to your business' needs and these are simply a baseline:

- Common Use Cases
  - Account
    - ColumnsToSelect: Name, Type, OwnerId
  - Case
    - ColumnsToSelect: CaseNumber, Id, Subject
  - Contact
    - ColumnsToSelect: Name, Id, Title
  - Lead
    - ColumnsToSelect: Name, Id
  - Opportunity
    - ColumnsToSelect: Name, Id
  - RecordType
    - ColumnsToSelect: Name, Id
  - User
    - ColumnsToSelect: Name, Email, IsActive, Profile.Id, Profile.Name
- Entries for Objects without 'Name' field
  - ContentVersion
    - Description, OwnerId
  - ContractLineItem
    - LineItemNumber, Description
  - Solution
    - SolutionName, Id

Using Process Builder to Call BREeze

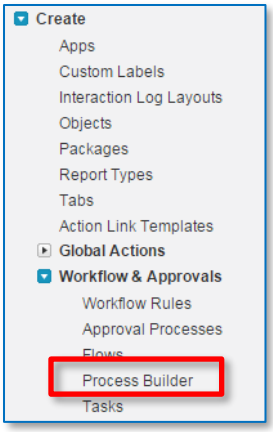
With BREeze 3.3, you now have the ability to trigger BREeze to run by using Process Builder. Previously it was necessary to have a developer create a trigger which would basically tell BREeze to run based on the rule name and the condition of when it should run. As a result, this provides better flexibility for admins as they can now control exactly how & when BREeze should run.

The set-up within Process Builder can be straight-forward or complex depending on your business requirements for BREeze. Below is a very simple example of how to use Process Builder to call BREeze to execute.

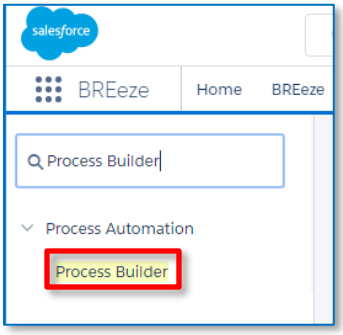
Creating Process

- First navigate to Process Builder from the Salesforce Setup menu

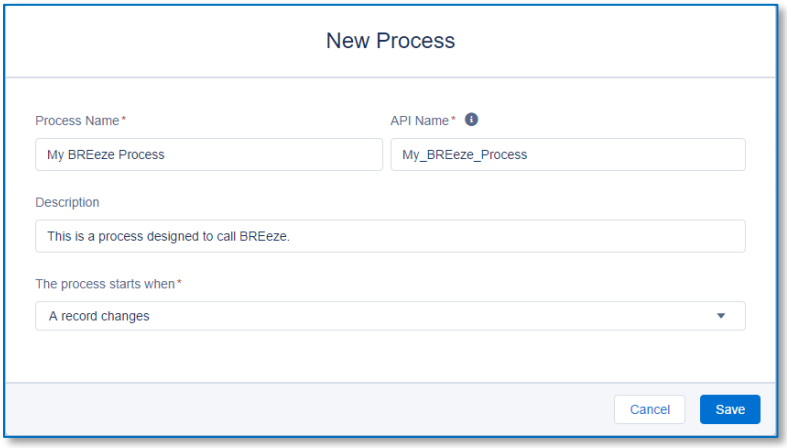
Classic:



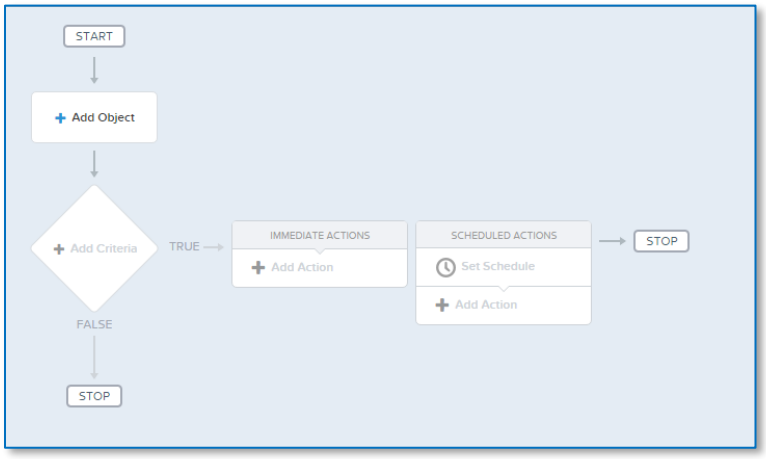
Lightning Experience:



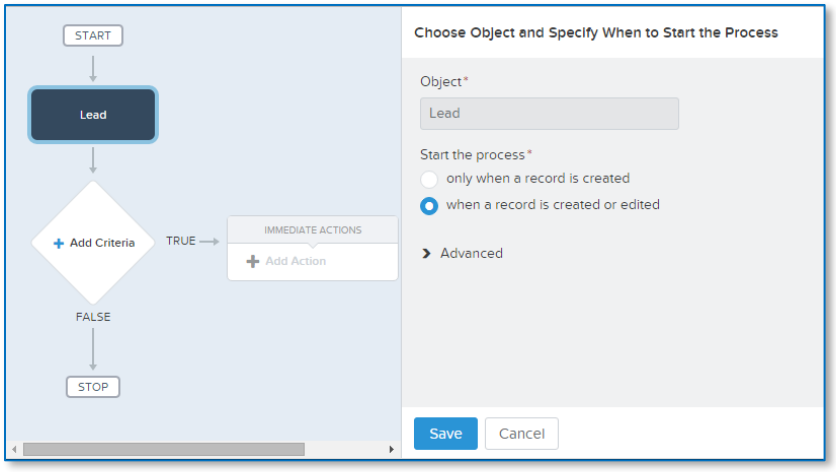
- From the resulting screen, click on the ***New*** button in the top right-hand corner
- You will be prompted with a pop-up window asking for a Process Name, API Name, Description and when the process should start
  - Enter an intuitive Process Name value and Description value
  - API Name will be auto populated based on the Process Name that is entered
  - Assuming this process is not being referenced by another process, select 'A record changes' from the "The process starts when" field



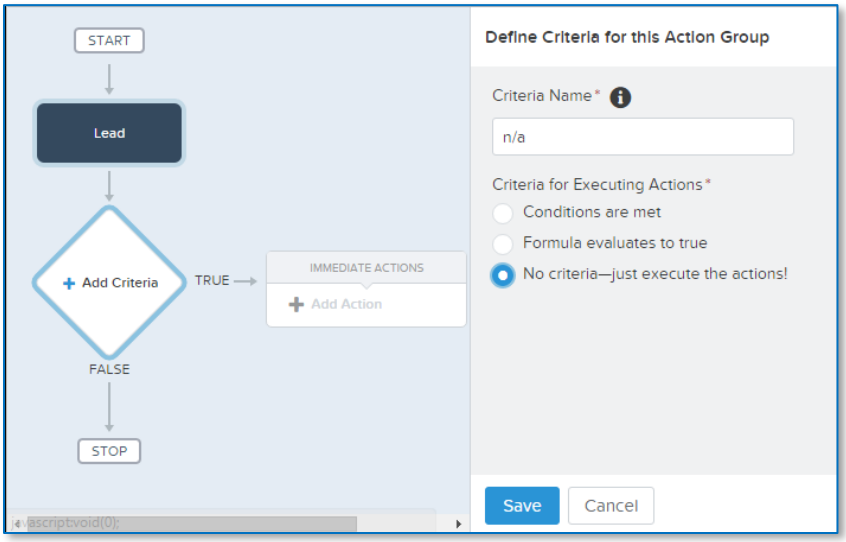
- Click the Save button once all information has been populated and you will then be brought to the main process builder screen



- First, click on the '+ Add Object' shape
- From the right-hand side panel, select the object which your rule has been built for and then also define when you want this process to run
    - If your BREeze rule should only run on insert, leave the "only when a record is created" value selected. However, if your rule should run on insert and update, select the "when a record is created or edited". Furthermore, if your BREeze rule should only run on updates, choose the second selection and you can build exceptions into your process later to only run BREeze on updates.

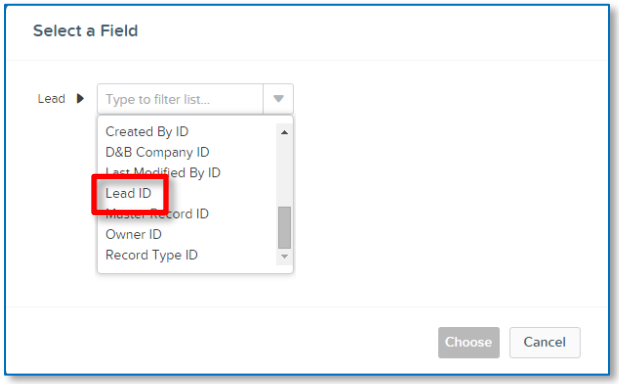


- Once your object & frequency has been defined, click on the ***Save*** button at the bottom of the screen
- Next, click on the '+ Add Criteria' shape
- From the right-hand panel, enter a name for this criteria shape and define what the criteria are for when BREeze should execute. For this example, all records are being passed to BREeze, so no criteria is defined

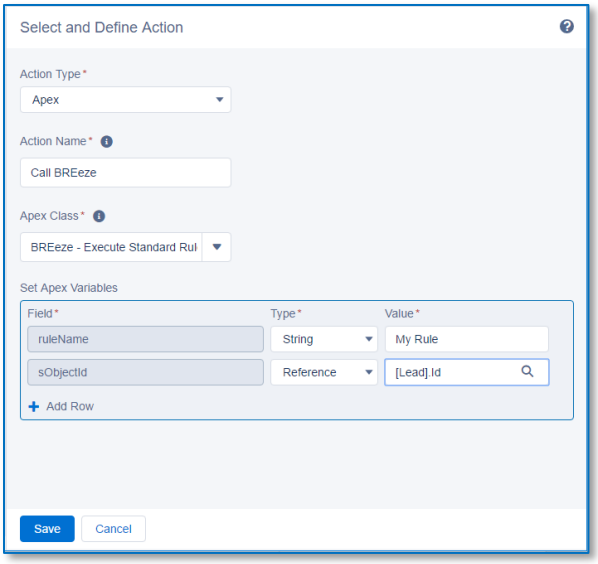


- Once your criteria has been defined, click on the ***Save*** button at the bottom of the screen
- Next, within the 'Immediate Actions' shape, click the '+ Add Action' shape
- From the right-hand panel, in the Action Type field, select "Apex"
- After making your previous selection, the Action Name & Apex Class fields should now be available
    - **Action Name**
        - Enter an intuitive name for the action (Ex. Call BREeze)
    - **Apex Class**
        - Select "BREeze – Execute Standard Rule"
- Finally, within the 'Set Apex Variables' section, enter the following:

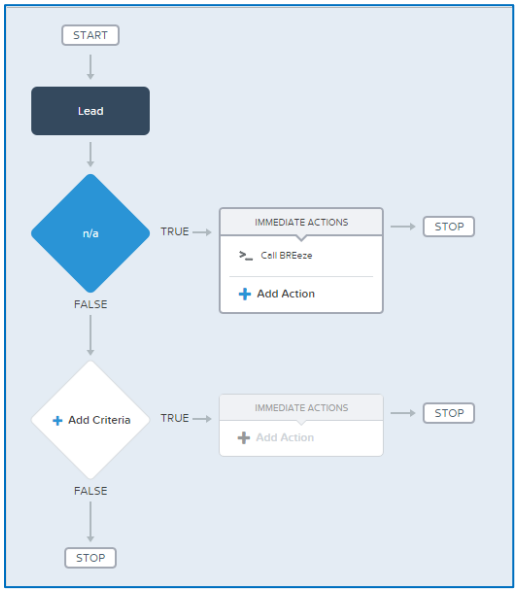
- ruleName
  - **Type**
    - For the purpose of calling BREeze from Process Builder, leave the default value of "String"
  - **Value**
    - The expected value for this field is the 'Internal Name' value from your main Rule screen. Be sure not to use the standard 'Name' field as the rule will not execute correctly.
- sObjectId
  - **Type**
    - To execute BREeze, the ID for the record must be passed into the application. As a result, update this value to "Reference"
  - **Value**
    - After selecting "Reference" above, click on the magnifying glass icon within this field. From the resulting pop-up, select the internal Salesforce ID field for the specific object your rule is built for and click the ***Choose*** button



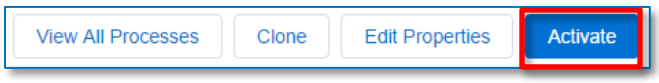
- The result should look like this:



- Once the action has been defined, click the **Save** button
- All required information has now been entered into Process Builder so the process itself should look something like this:



- The final step is to activate the process itself. Simply click the Activate button in the top right-hand corner of the process builder screen



- Verify your process setup by attempting to insert/update a record that will run through your BREeze rule logic

Importing Data into BREeze

Since the rule data that is defined within your rule sets are stored in objects within Salesforce versus in metadata, it is possible to import or migrate your rule sets between Salesforce instances. There are four main objects that make up the core data set for the BREeze platform:

- RuleName (BREeze__RuleName__c)
- Rule (BREeze__Rule__c)
- Rule Detail (BREeze__Rule_Detail__c)
- Rule Assignment (BREeze__RuleAssignment__c)

Additionally, if your rule sets include custom functions that you have defined, you will also need to include the two following objects:

- Function (BREeze__Function__c)
- Function Extension (BREeze__Function_Extension__c)

There is a specific process to import the rule data correctly into BREeze. For more information, reference the BREeze Import Guide.

BREeze Localized Debugging

As a standard, managed packages do not have the ability for debug logging. Meaning if the code within the managed package has a debug statement, it does not appear in the org's debug logs. To get around this issue, BREeze has a feature available for localized debugging.

At a high level, local debug classes can be implemented to return one of four levels of debug information back:

- Debug
  - Provides basic debugging and error trapping
- Information
  - Provides additional processing information regarding the rules
- Transaction
  - Provides data about each record passed into BREeze
- Performance
  - Provides some key performance metrics

Once the local classes have been implemented, BREeze comes packaged with a Custom Metadata Type (CMDT) called 'Information Log Default' where different levels of debugging can be enabled & disabled.

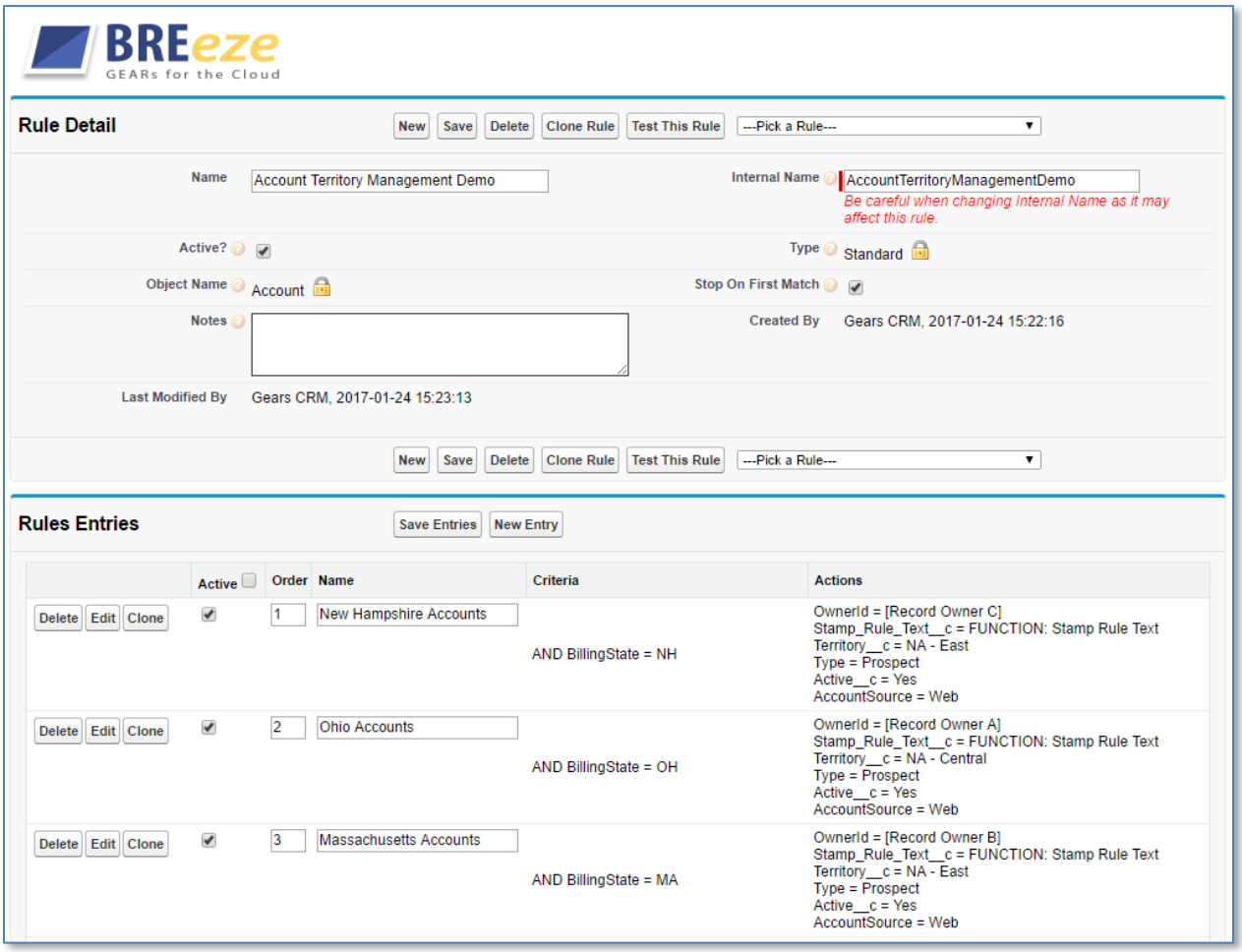
Additional Resources:

- For more information on this topic, reference the Localized Debugging documentation found on the GearsDesign website.

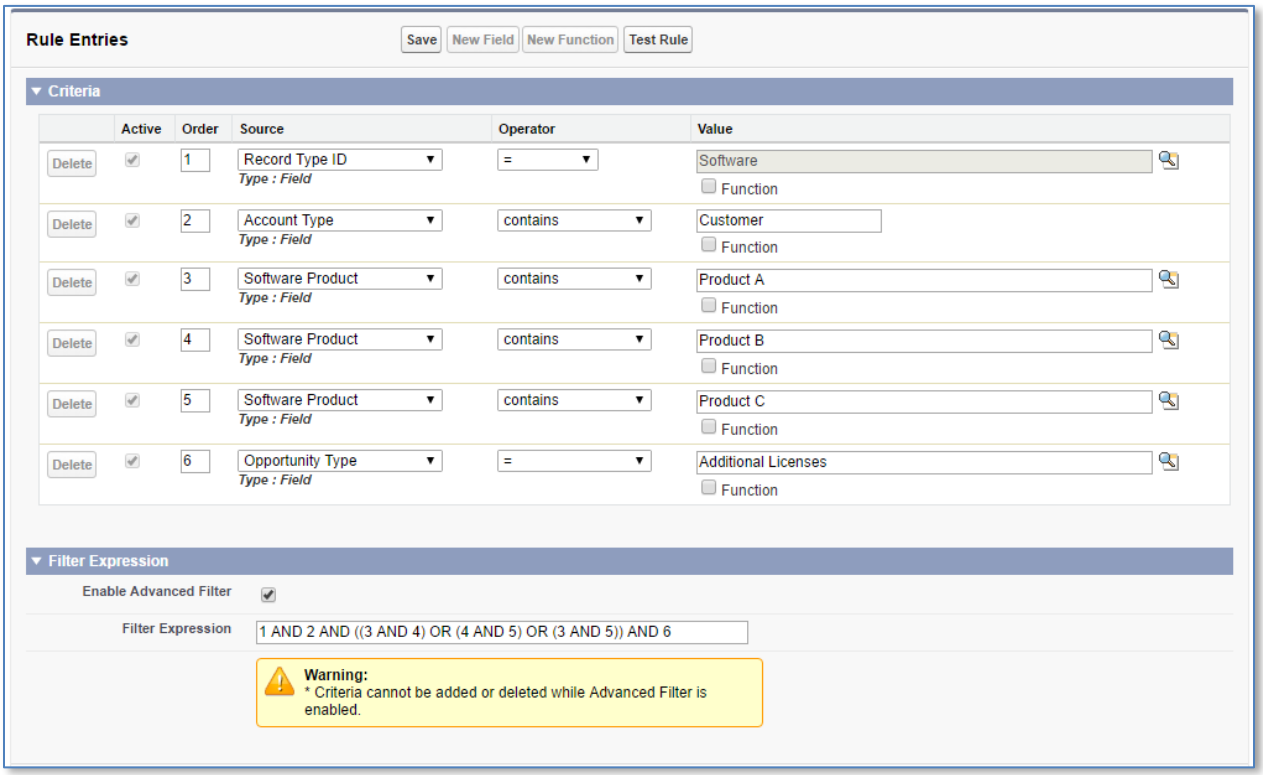
Examples/Use Cases

Below are a few examples of how BREeze can be leveraged within a Salesforce instance.

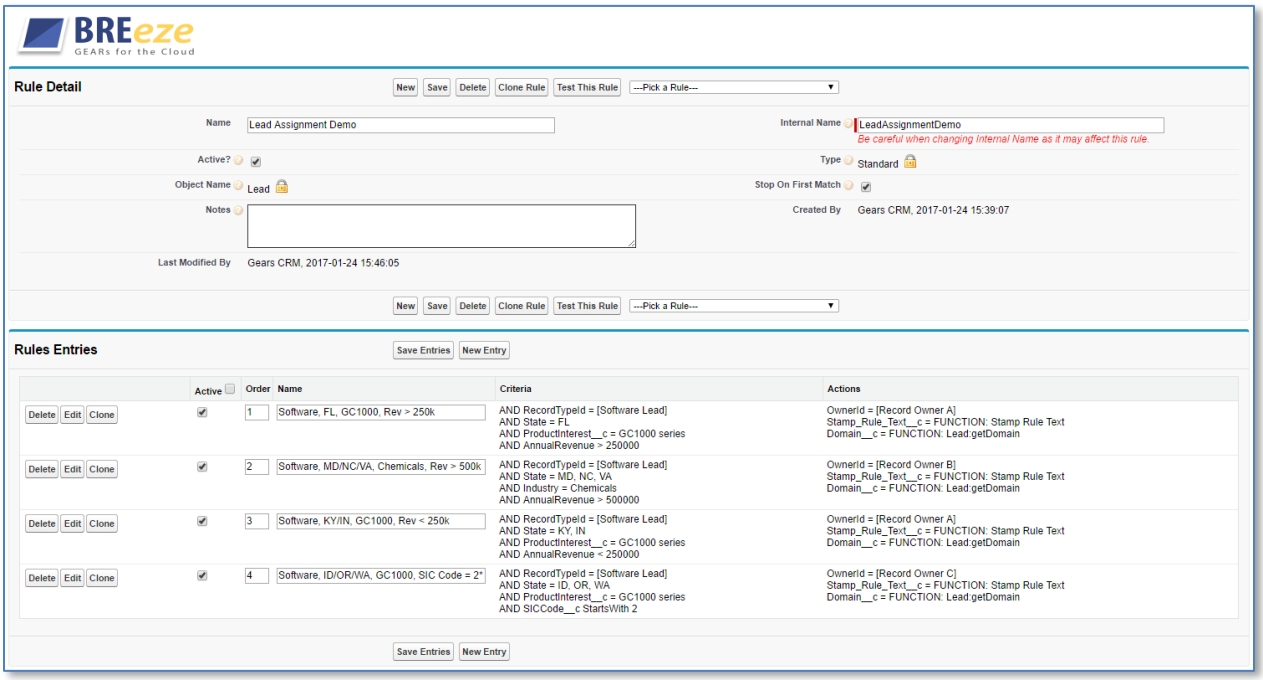
Account Territory Management



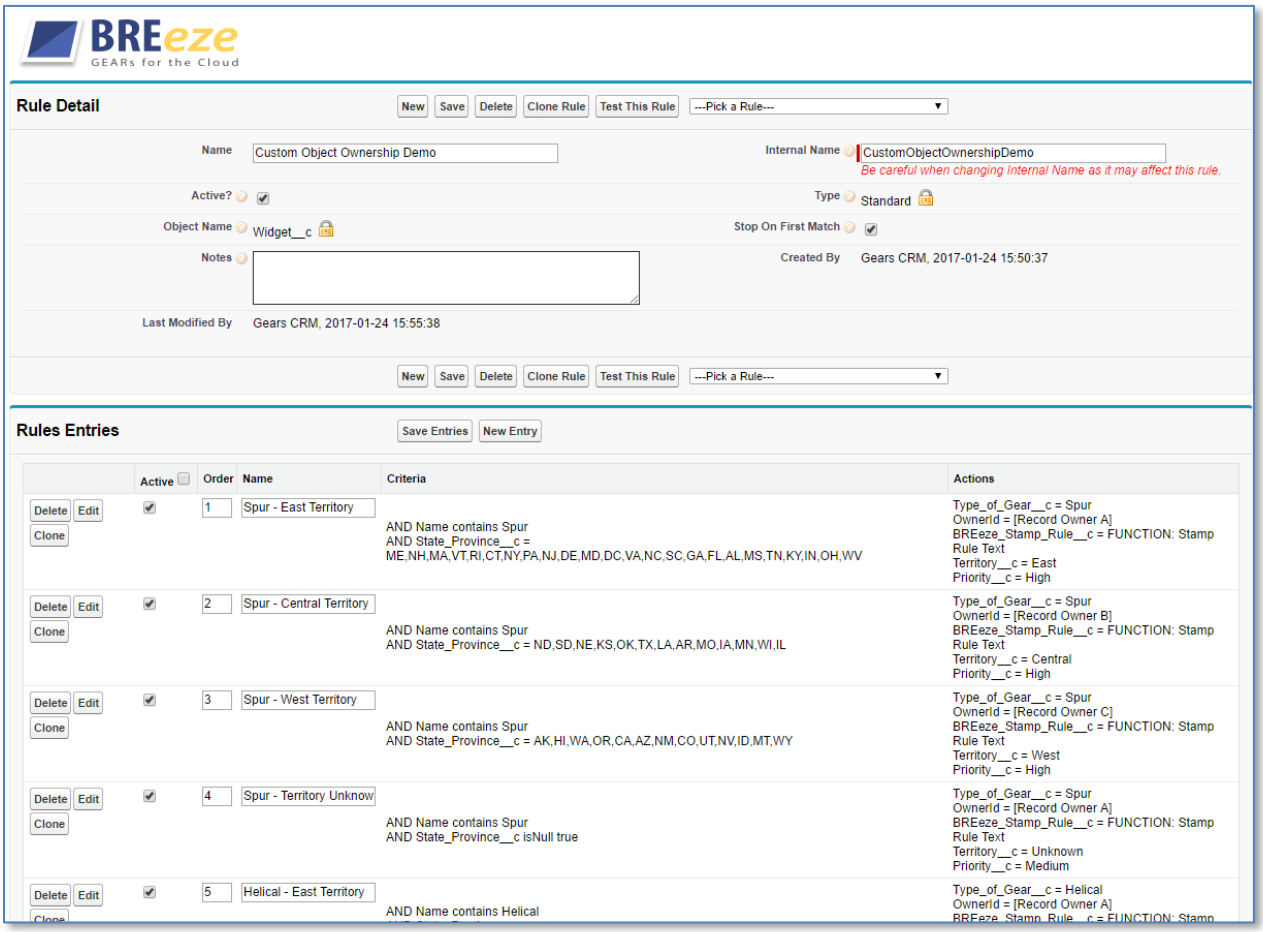
Advanced Filter Criteria



Multiple Field Updates per Rule



Owner Assignment on Custom Objects



BREeze
GEARs for the Cloud

Rule Detail

New | Save | Delete | Clone Rule | Test This Rule | ---Pick a Rule---

Name	Custom Object Ownership Demo	Internal Name	CustomObjectOwnershipDemo *Be careful when changing Internal Name as it may affect this rule.*
Active?	☑	Type	Standard
Object Name	Widget__c	Stop On First Match	☑
Notes		Created By	Gears CRM, 2017-01-24 15:50:37
Last Modified By	Gears CRM, 2017-01-24 15:55:38		

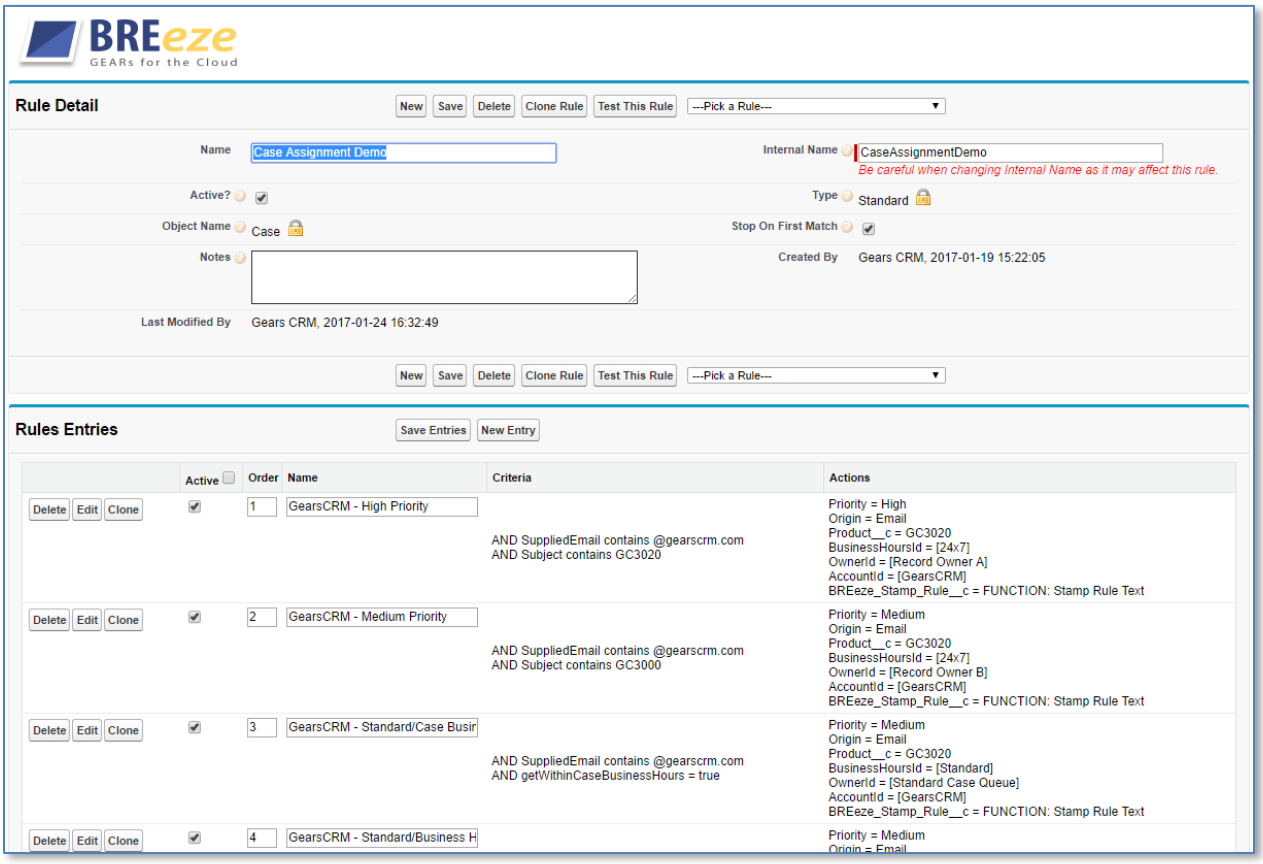
New | Save | Delete | Clone Rule | Test This Rule | ---Pick a Rule---

Rules Entries

Save Entries | New Entry

	Active	Order	Name	Criteria	Actions
Delete Edit Clone	☑	1	Spur - East Territory	AND Name contains Spur AND State_Province__c = ME,NH,MA,VT,RI,CT,NY,PA,NJ,DE,MD,DC,VA,NC,SC,GA,FL,AL,MS,TN,KY,IN,OH,WV	Type_of_Gear__c = Spur OwnerId = [Record Owner A] BREeze_Stamp_Rule__c = FUNCTION: Stamp Rule Text Territory__c = East Priority__c = High
Delete Edit Clone	☑	2	Spur - Central Territory	AND Name contains Spur AND State_Province__c = ND,SD,NE,KS,OK,TX,LA,AR,MO,IA,MN,WI,IL	Type_of_Gear__c = Spur OwnerId = [Record Owner B] BREeze_Stamp_Rule__c = FUNCTION: Stamp Rule Text Territory__c = Central Priority__c = High
Delete Edit Clone	☑	3	Spur - West Territory	AND Name contains Spur AND State_Province__c = AK,HI,WA,OR,CA,AZ,NM,CO,UT,NV,ID,MT,WY	Type_of_Gear__c = Spur OwnerId = [Record Owner C] BREeze_Stamp_Rule__c = FUNCTION: Stamp Rule Text Territory__c = West Priority__c = High
Delete Edit Clone	☑	4	Spur - Territory Unknow	AND Name contains Spur AND State_Province__c isNull true	Type_of_Gear__c = Spur OwnerId = [Record Owner A] BREeze_Stamp_Rule__c = FUNCTION: Stamp Rule Text Territory__c = Unknown Priority__c = Medium
Delete Edit Clone	☑	5	Helical - East Territory	AND Name contains Helical	Type_of_Gear__c = Helical OwnerId = [Record Owner A] BREeze_Stamp_Rule__c = FUNCTION: Stamp

Advanced Case Assignment



Troubleshooting Sandboxes

The data that makes up the BREeze rules & BREeze functions are all stored in data versus metadata. This makes it easy to quickly extract your rules or even push in updates to your rules however it can result in some unintended consequences with regards to sandboxes.

Salesforce offers the following sandbox types:

- Developer
  - Includes only metadata
- Developer Pro
  - Includes only metadata
- Partial
  - Includes metadata & sample data
- Full
  - Includes metadata & all data

Reference: https://help.salesforce.com/articleView?id=data_sandbox_environments.htm&type=0

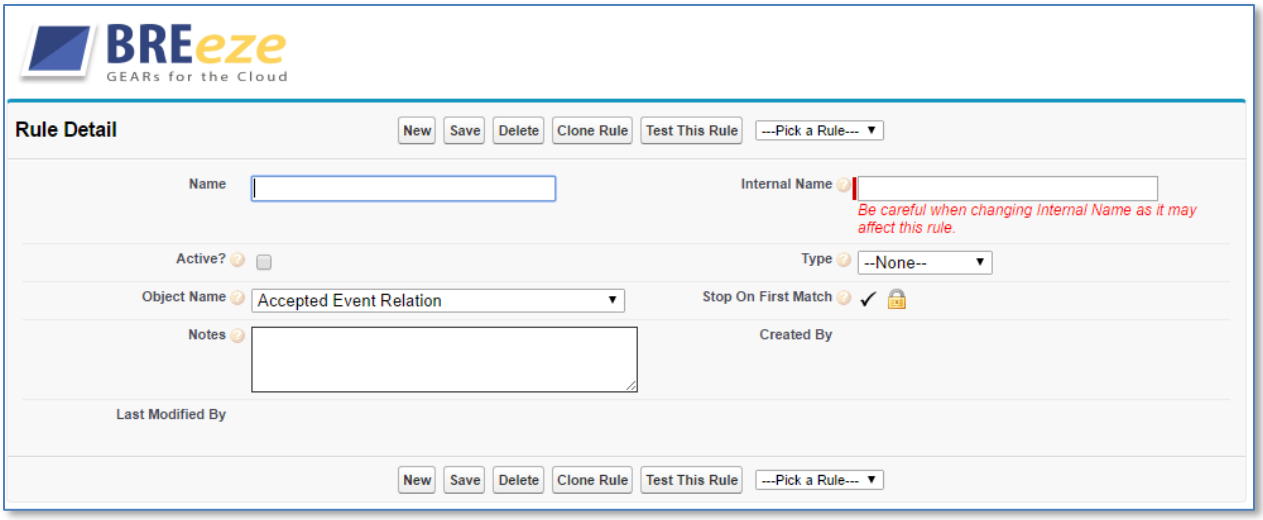
As noted above, for both Developer & Developer Pro sandboxes, no data will be copied to your sandbox and as a result, your BREeze rule and function data will not be copied over. For Partial sandboxes, this data can be brought over assuming it is included in your sandbox template for inclusion. In the case of full sandboxes, there are no issues as all data will be brought over so as a result, your BREeze rules & functions will be intact.

There is a method in which to bring your rule & function data from your production environment into your sandbox which is outlined in the Importing Data into BREeze section of this document.

In some cases, the data not being brought over would be intentional if you are looking to build your BREeze rules from the ground up. If that is the case, there are a few steps to take to prepare your BREeze instance.

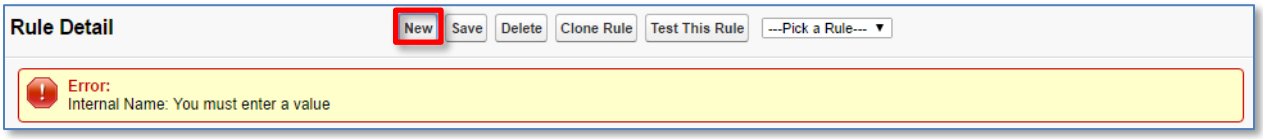
Default Rule

The Visualforce component relies on the existence of a BREeze RuleName record to render correctly. In the case of a sandbox where no data is brought into the instance, this can present an issue. As a result, the default main screen in BREeze will look like this:

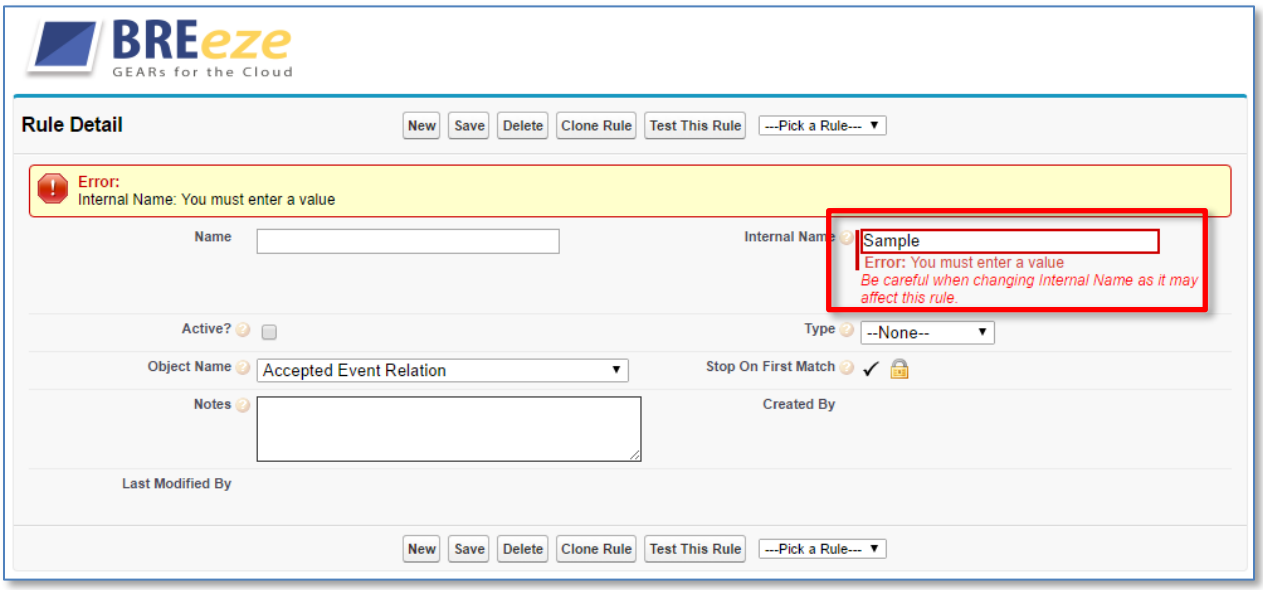


When you attempt to create a new record and save, it loops back to this same screen and will not commit your save. To resolve this issue, take the following steps:

- From the main BREeze screen, first click on the ***New*** button
  - The following error should be shown:



- Next, type any value into the 'Internal Name' field:



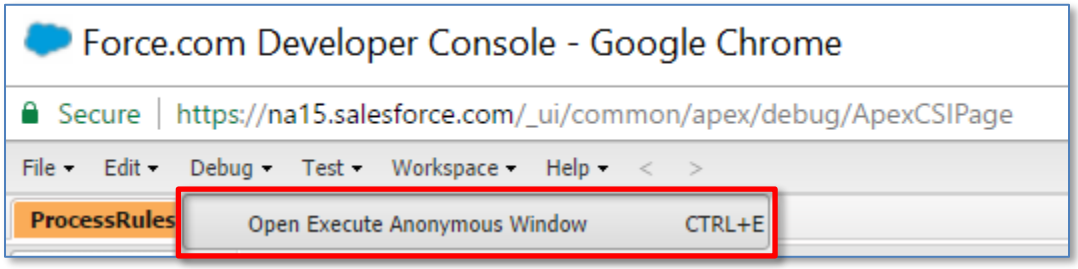
- Finally, click on the ***New*** button again

- After clicking New, a default rule should be inserted and you can now use that rule to create your rule set or create a brand new rule

Default Criteria & Functions

Similarly, to the rule data, the criteria & functions are also data and therefore a developer sandbox or a partial sandbox where these were not included would potentially be missing the records needed to build out rules. To restore the criteria & functions delivered by the BREeze package, simply follow these steps to restore them:

- Open the Developer Console within your org you are looking to restore functions for
- From the Menu Bar, select Debug and then select 'Open Execute Anonymous Window'



- From the resulting window, copy and paste the following:

```
List <breeze__Function__c> f = [select id from breeze__Function__c];
delete f;
List <breeze__Criteria__c> c = [select id from breeze__Criteria__c];
delete c;

breeze.ManualPostSCript.purgeFunctionCustomSetting();
breeze.ManualPostSCript.purgeCriteriaCustomSetting();
breeze.ManualPostSCript.createStandardCriteriaData();
breeze.ManualPostSCript.createStandardFunctionData();
breeze.ManualPostSCript.createStandardFunctionData_3_0();
```

- Finally click the Execute button from the window
- After that has completed processing, when navigating back to the BREeze Criteria & BREeze Functions tabs, all data should be restored

Additional Resources

Data Dictionary

Custom Function Overview

About GearsDesign

At GearsDesign, we build products that help companies achieve breakthrough results with their Salesforce implementations. We know first-hand the challenges you face as a Salesforce administrator. Your implementation must meet your organization's unique requirements. It also needs to stay tightly aligned with Sales and other operations. But your instance also must be able to turn on a dime when business conditions shift or your company's needs change. Sometimes, the changes you need to make go beyond what you're able to do with Salesforce's feature set. Your choices are either a risky and time-consuming customization project, or to just say 'no' and take the heat. We give you better options. GearsDesign solutions make it easy for you to extend and customize Salesforce's capabilities to achieve all your CRM goals. With a GearsDesign solution, any time you're confronted with a tough request, you'll be able to say with confidence, "I can make Salesforce do that."

For more information about GearsDesign or more information on any of our additional applications for Salesforce, visit http://gearsdesign.com/ or contact us at gearsdesignsupport@rafter.one.