NORTH52

Turbocharge Dynamics 365

North52 LaunchPad

The fun and free way to learn North52!

North52 Decision Suite - Decision Tables Overview

Learn more about North52 products at our online knowledge base (support.north52.com)

BUSINESS PROCESS ACTIVITIES

Launch BPA Knowledge Base

TEST SHIELD Assemble. Arrange. Act. Assert.

Launch TestShield Knowledge Base

DATA PACKAGER

Launch DP Knowledge Base

Table of Contents

DT - How to - 01 - Decision Tables Overview

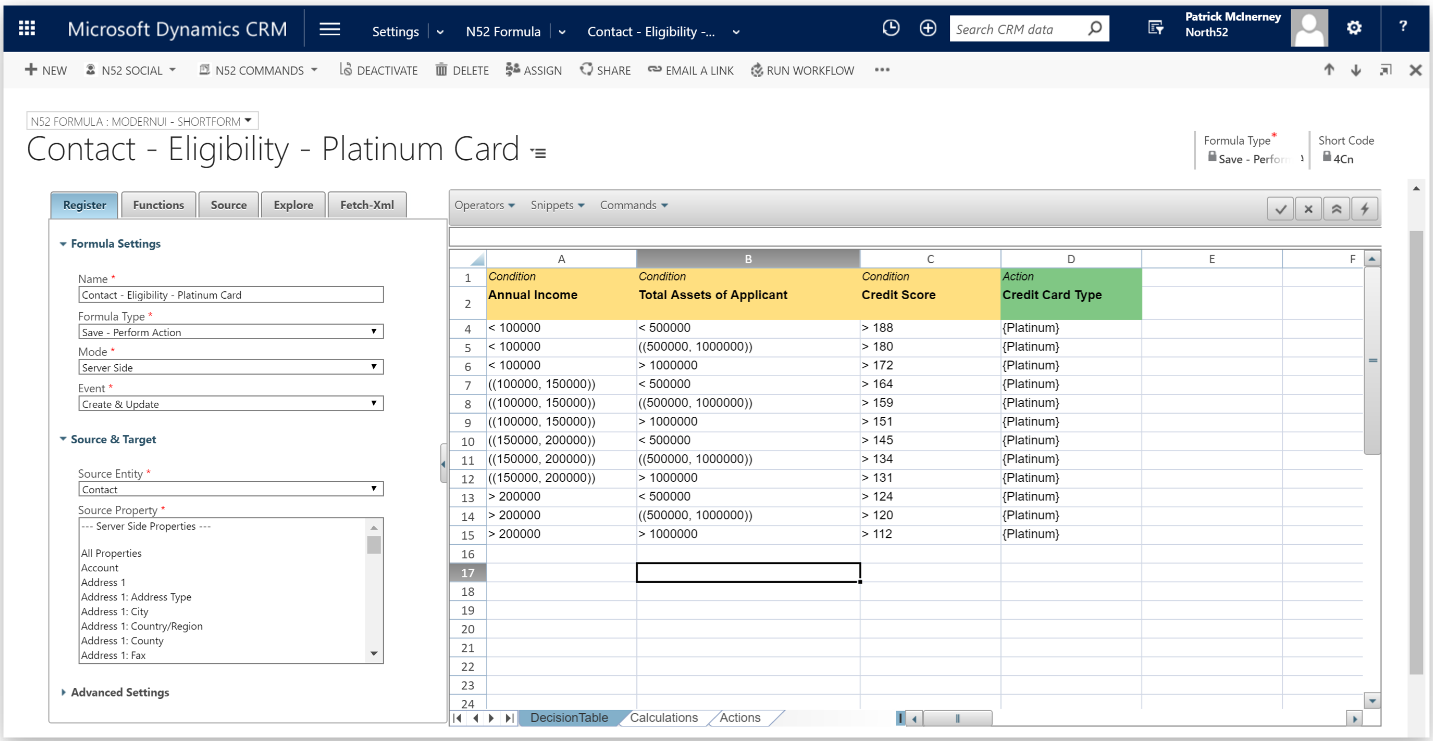
[TOC]

Overview

A Decision Table represents a collection of related business rules with condition rows, rules, and actions presented in a tabular form that is easy to understand. Business users can review cells and their values at a glance and can use Decision Table rule building features by clicking icons and selecting values in the Decisions Table builder.

Decision Tables allow us to create and use business rules in an easy to understand format that provides an alternative to the IF/THEN rule format. The spreadsheet nature of a Decision Table ensures that business analysts, functional consultants, etc. who are familiar with spreadsheets find them easy to construct, read and modify.

The compressed nature of business rules presented in the form of decision table make them fast to understand, clean and readable. The simple visualization of a compact and structured table format aims to mimic real world business problems with many tables breaking down complex problems into simple steps.



Contact - Eligibility - Platinum Card

Formula Type: Save - Perform Action
Short Code: 4Cn

Formula Settings

Name: Contact - Eligibility - Platinum Card
Formula Type: Save - Perform Action
Mode: Server Side
Event: Create & Update

Source & Target

Source Entity: Contact
Source Property: --- Server Side Properties --- All Properties, Account, Address 1, Address 1: Address Type, Address 1: City, Address 1: Country/Region, Address 1: County, Address 1: Fax

Advanced Settings

	A	B	C	D
1	*Condition*	*Condition*	*Condition*	*Action*
2	**Annual Income**	**Total Assets of Applicant**	**Credit Score**	**Credit Card Type**
4	< 100000	< 500000	> 188	{Platinum}
5	< 100000	((500000, 1000000))	> 180	{Platinum}
6	< 100000	> 1000000	> 172	{Platinum}
7	((100000, 150000))	< 500000	> 164	{Platinum}
8	((100000, 150000))	((500000, 1000000))	> 159	{Platinum}
9	((100000, 150000))	> 1000000	> 151	{Platinum}
10	((150000, 200000))	< 500000	> 145	{Platinum}
11	((150000, 200000))	((500000, 1000000))	> 134	{Platinum}
12	((150000, 200000))	> 1000000	> 131	{Platinum}
13	> 200000	< 500000	> 124	{Platinum}
14	> 200000	((500000, 1000000))	> 120	{Platinum}
15	> 200000	> 1000000	> 112	{Platinum}

Example of a Simple Decision Table to Calculate a Contacts eligibility for a Credit Card

Why use Decision Tables?

Automating business processes using software such as Dynamics CRM has become a key return on investment (ROI) factor when conducting benefit analysis for new software deployments.

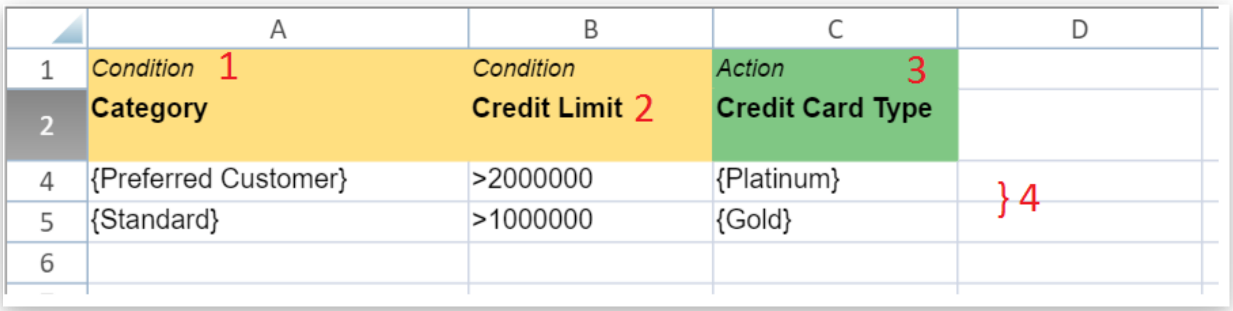
Businesses today are trying to make use of all their resources so this means not relying solely of software developers for implementing business rules. When software can empower business analysts/functional consultants to work together with software developers this leads to CRM projects being delivered on time, to specification and within budget. We believe that Decision Tables enables a synergy between business analysts, functional consultants and software developers when delivering projects.

In addition, many organisations are experiencing an increasing burden to make software systems more responsive to business changes, so allowing trained business users to change business rules is becoming increasingly important.

What makes up a Decision Table?

Decision tables are rules composed of rows and columns. They are used to display in table form all possible situations that a business decision might encounter, and to specify which action to take in each of these situations. A decision table expresses sets of related conditions and actions in a spreadsheet like view. Each row is in effect a single rule in the table made up of many rules. Each column is the definition of a Condition or Action.

Decision tables are composed of rows and columns. As shown in the following table, each row corresponds to a single rule [4], with the columns defining the conditions [1] and actions [3] of the rules. The second row [2] of the table contains the column header, where we have the column names.



	A	B	C	D
1	*Condition* 1	*Condition*	*Action* 3	
2	**Category**	**Credit Limit** 2	**Credit Card Type**	
4	{Preferred Customer}	>2000000	{Platinum}	} 4
5	{Standard}	>1000000	{Gold}	
6				

A condition is a test that must evaluate to true for the associated action to be executed. If a decision table rule uses multiple conditions, all conditions for a row must evaluate to true for the action to execute. Condition tests might be looking to match the exact value of a fields value, or might be comparing values against a greater than or less than value. In more complicated situations functions may be used for this comparison test so for example using a function 'StartsWith' to check does the first 4 characters of an input value match a certain string.

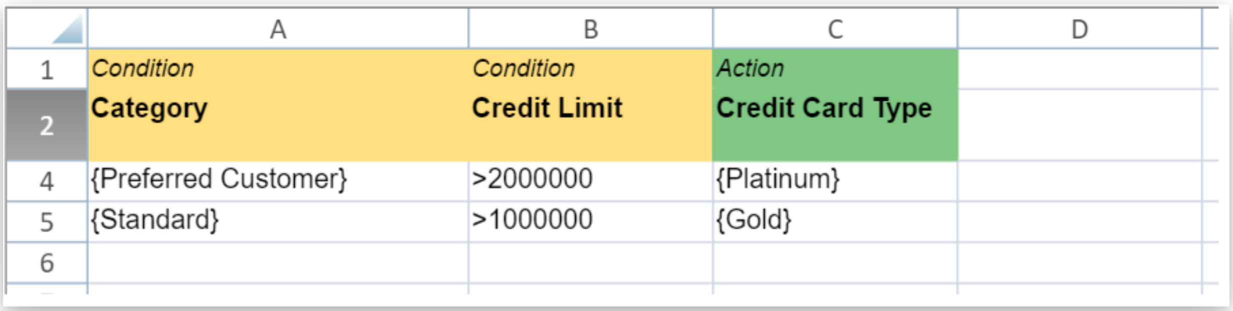
To help understand the make-up of a Decision Table, consider a set of IF/THEN rules that determine if a Contact is eligible for a Gold or Platinum credit card, and an equivalent Decision Table.

The IF/THEN rules follow:

```
if Contact.Category = {Preferred Customer} and Contact.Credit Limit > 2,000,000 then Contact.Credit Card Type = {Platinum}
```

```
if Contact.Category = {Standard Customer} and Contact.Credit Limit > 1,000,000 then Contact.Credit Card Type = {Gold}
```

Corresponding Decision Table



	A	B	C	D
1	*Condition*	*Condition*	*Action*	
2	**Category**	**Credit Limit**	**Credit Card Type**	
4	{Preferred Customer}	>2000000	{Platinum}	
5	{Standard}	>1000000	{Gold}	
6				

What happens when a Decision Table Executes?

When a Decision Table executes each row of the Decision Table is processed one by one. The Conditions are evaluated column by column from left-to-right in an 'AND' or 'OR' fashion depending on the configuration. Then any Action commands associated with a row (i.e. whose conditions evaluate to true) are executed.

Taking the decision table above,

For each row in the decision table, the Contact.Category column value is evaluated first followed by Contact.Credit Limit in an AND manner. If the conditions both equate to True then Contact.Credit Card Type is set to Gold or Platinum.

DT - How to - 02 - Create your first Decision Table

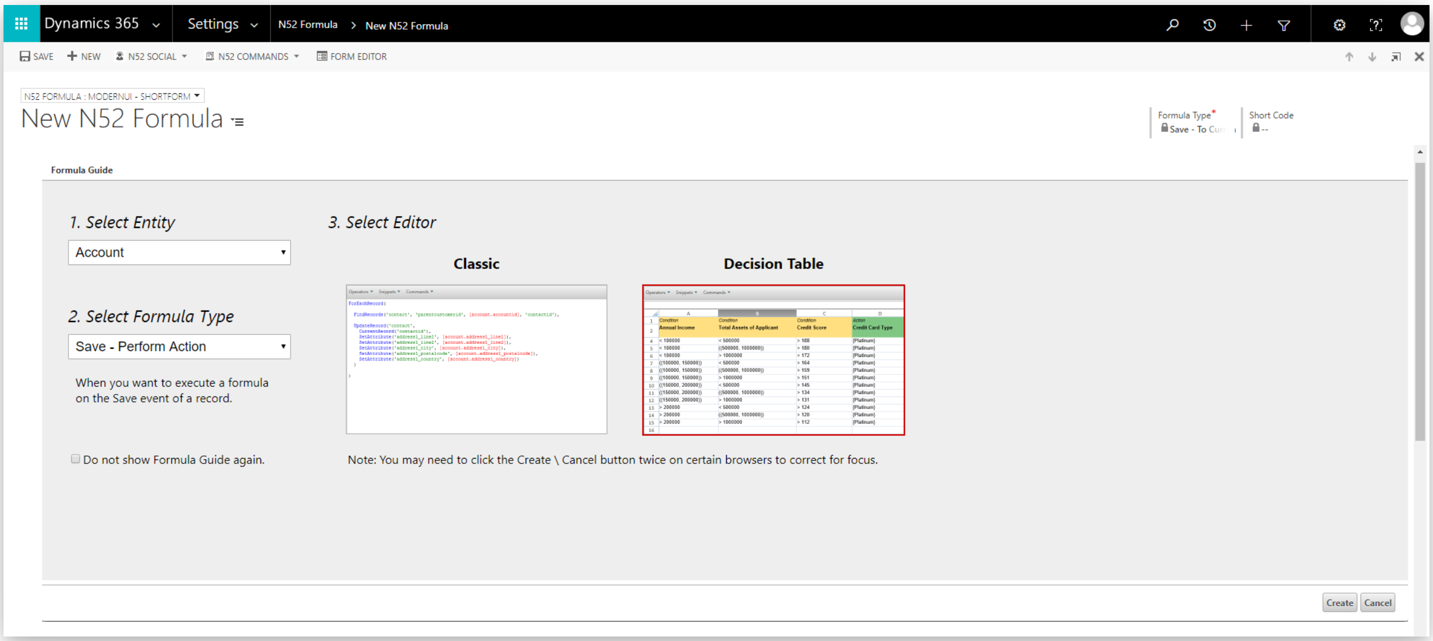
[TOC]

Overview

In this article we will demonstrate the basic steps involved in setting up a North52 Decision Table.

Set up the North52 Formula

- Create a new North52 Formula record, to open the Formula Guide
- Set **Select Entity** to be *Account*
- Set **Select Formula Type** to be *Save - Perform Action*
- Set **Select Editor** to be *Decision Table*
- Click the *Create* button



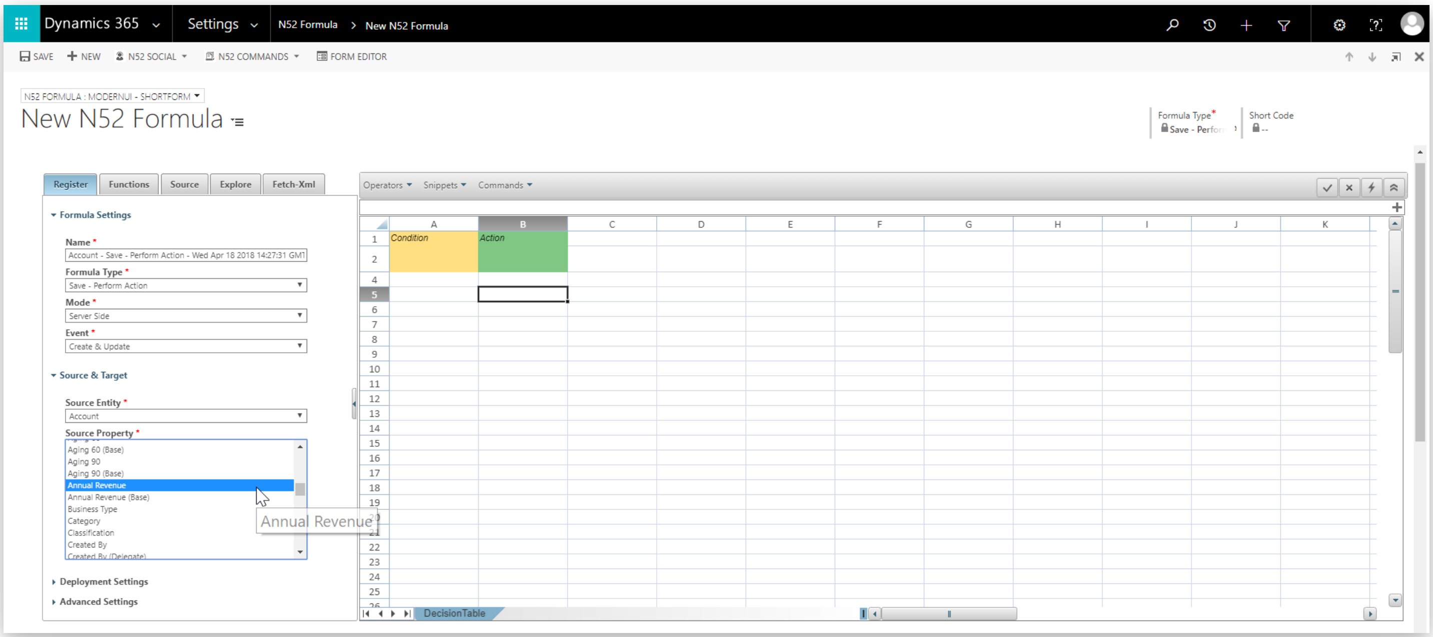
When you want to execute a formula on the Save event of a record.

Do not show Formula Guide again.

Note: You may need to click the Create \ Cancel button twice on certain browsers to correct for focus.

This will open up the Decision Table editor canvas

- From the **Source** tab, expand the **Source & Target** section
- The **Source Entity** should be *Account*
- Click on *Annual Revenue* within the **Source Property** field select it
- Save the Formula



Formula Settings

Name *: Account - Save - Perform Action - Wed Apr 18 2018 14:27:31 GMT
Formula Type *: Save - Perform Action
Mode *: Server Side
Event *: Create & Update

Source & Target

Source Entity *: Account
Source Property *: Annual Revenue

Deployment Settings
Advanced Settings

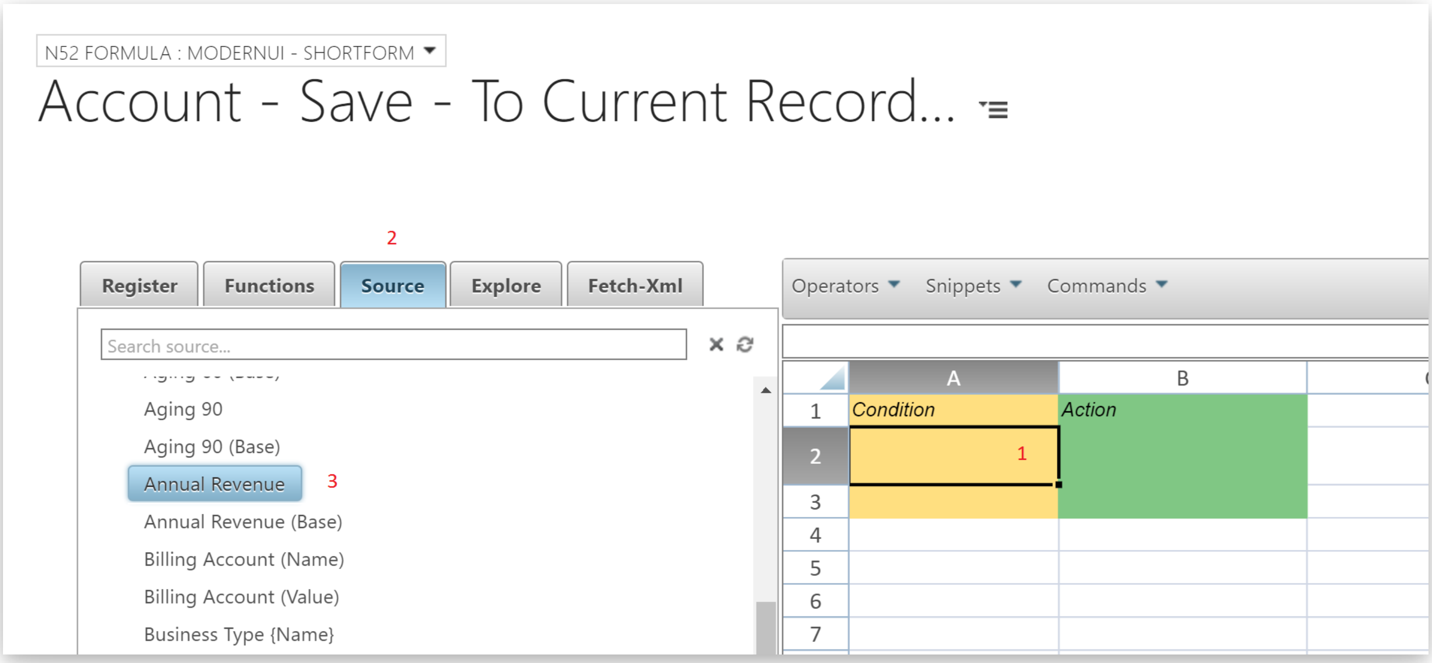
What are Conditions?

Conditions are prerequisites for Actions.

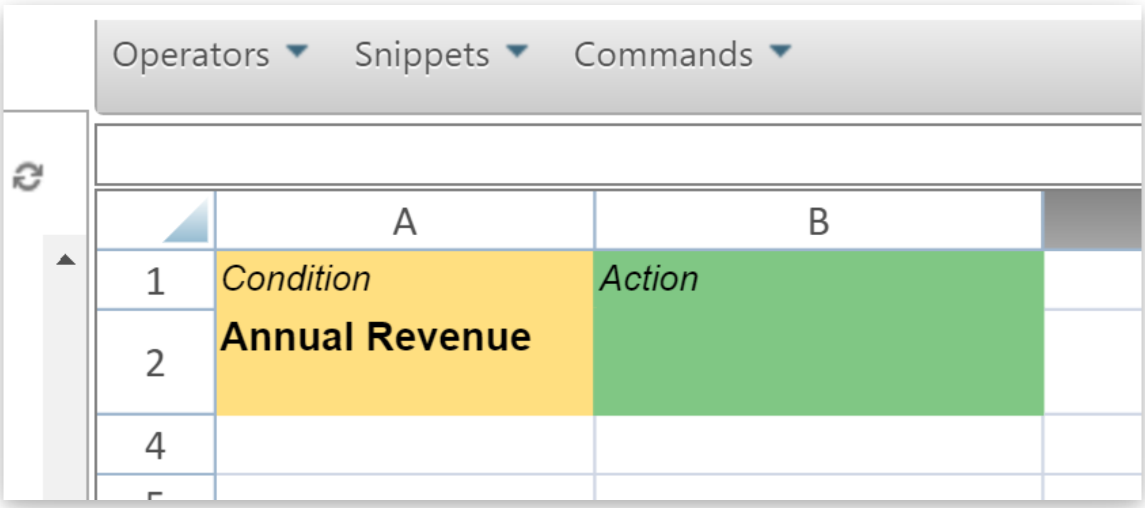
What does that actually mean? When certain conditions are met, then the action will be triggered. For example, a condition might be that when the Annual Revenue of an account is greater than $1,000,000 then the Credit Limit on that account should be set to $75,000.

To set a **Condition** you click on the *blank yellow field* (1) under *Condition* and then click on the *Source* (2) tab on the left-hand configuration pane and then click on the field you want to set as your condition.

In this instance we are going to use **Annual Revenue** (3)

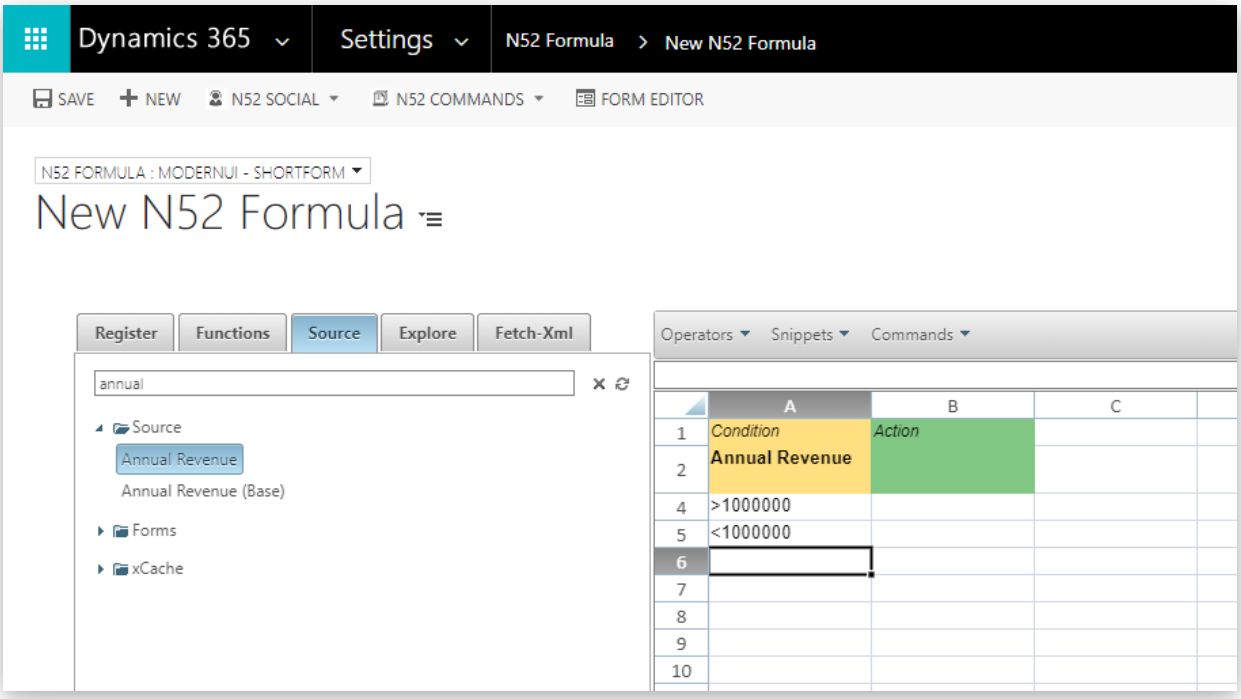


Annual Revenue will now appear in bold text under *Condition.*



Now that we have selected Annual Revenue to be our condition, we need to specify what the condition will be.

In **cell A4** type *>1000000* and in **cell A5** type *<1000000*

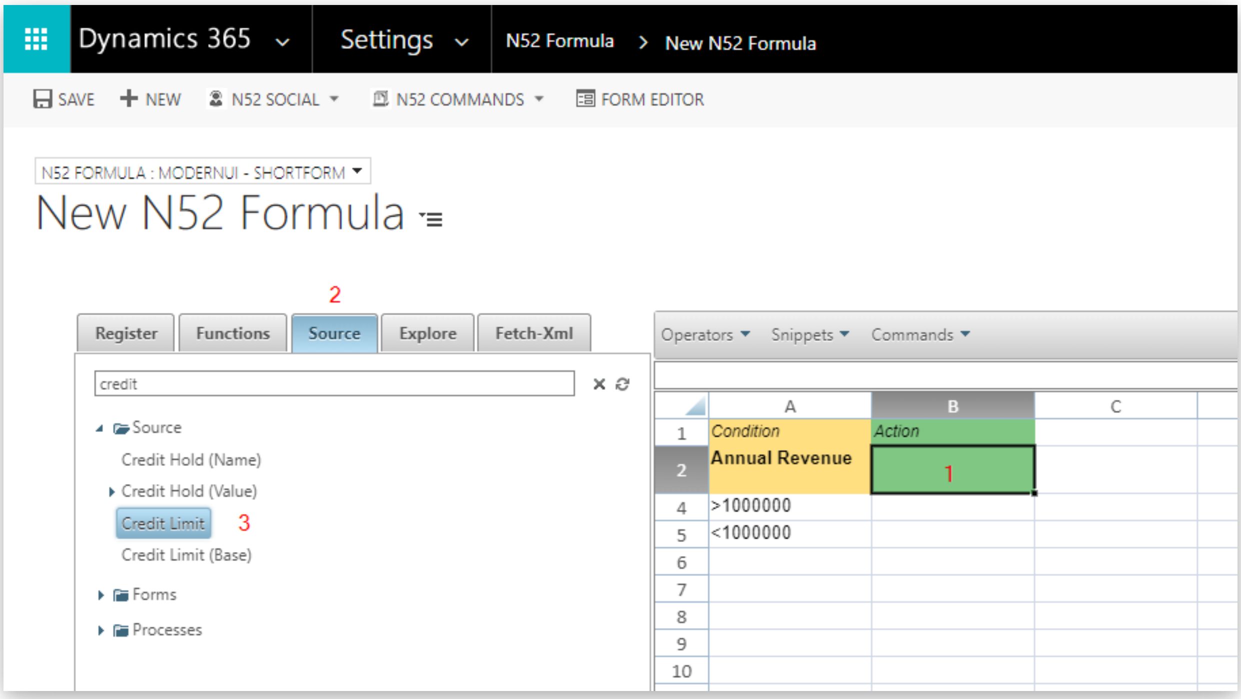


This tells the Decision Table that we want to do 'Something' when the Annual Revenue is greater than $1,000,000 and to do something else when it it less than $1,000,000.

What are Actions?

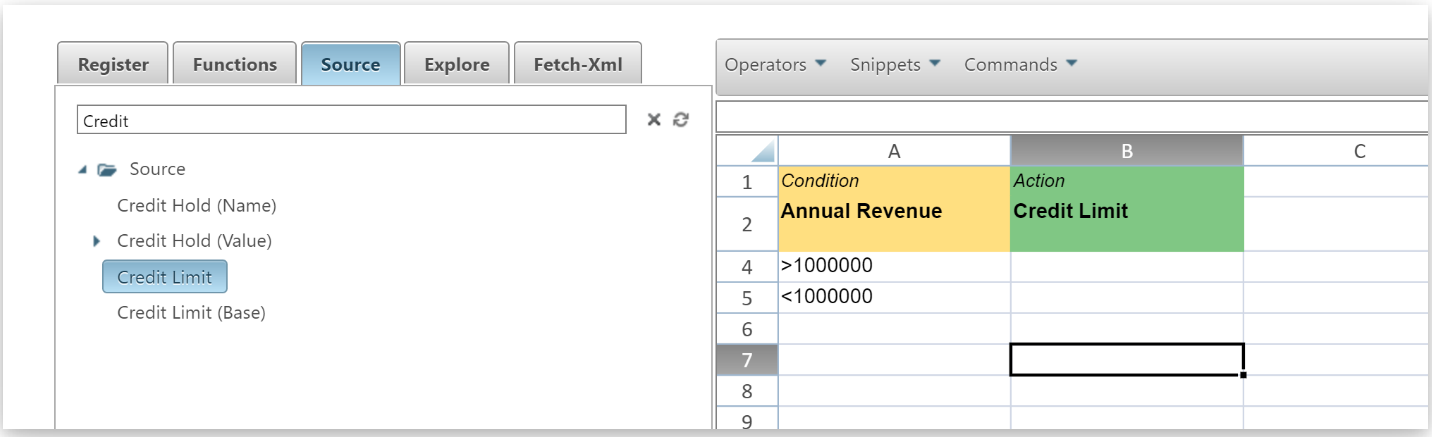
Actions are the desired outcomes you want. In this example the *Action* is that we want to set the value of the **Credit Limit** field on the ***Account***.

To do this, we click on the *blank green cell* (1) under *Action* and then click on the *Source* (2) tab on the configuration pane and then click *Credit Limit* (3) in the list of source fields.



	A	B	C
1	*Condition*	*Action*	
2	**Annual Revenue**		
4	>1000000		
5	<1000000		

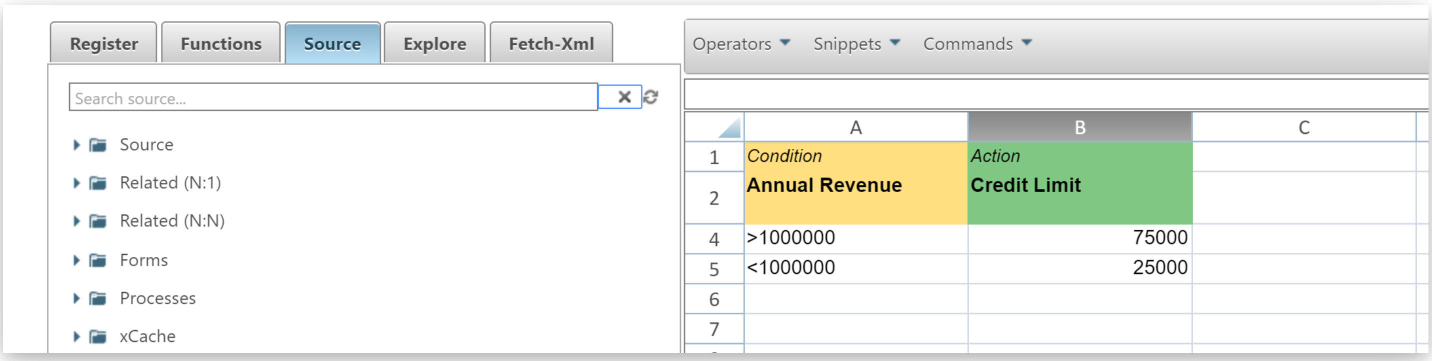
Credit Limit should now appear in the *Action* green cell (B2).



	A	B	C
1	*Condition*	*Action*	
2	**Annual Revenue**	**Credit Limit**	
4	>1000000		
5	<1000000		

For example, if the Annual Revenue is greater than $1,000,000 then the Credit Limit should be set to $75,000. However, if the Annual Revenue is less than $1,000,000 then the Credit Limit should be set to $25,000.

Type *75000* in **cell B4** and *25000* into **cell B5**.

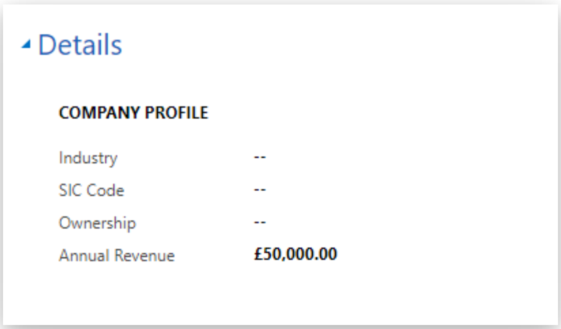


	A	B	C
1	Condition	Action	
2	Annual Revenue	Credit Limit	
4	>1000000	75000	
5	<1000000	25000	
6			
7			

Finally save the formula and you have finished setting up your first Decision Table.

Testing the Decision Table

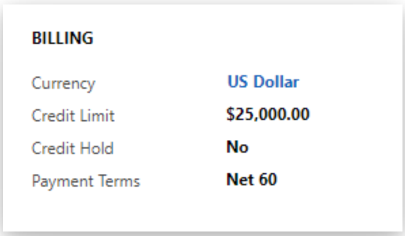
Open an ***Account*** record and set the **Annual Revenue** field to *$500,000*.



COMPANY PROFILE

Industry	--
SIC Code	--
Ownership	--
Annual Revenue	£50,000.00

When you save the account record, the Decision Table will execute and update the **Credit Limit** field to $25,000.



BILLING

Currency	US Dollar
Credit Limit	$25,000.00
Credit Hold	No
Payment Terms	Net 60

Finally, change the **Annual Revenue** to *$1,500,000*, click *Save* and the **Credit Limit** will update to $75,000.

DT - How to - 03 - Learn everything you need to know about Decision Table Conditions

[TOC]

Overview

This article will detail everything you need to know about how to set **Conditions** in N52 Decision Tables.

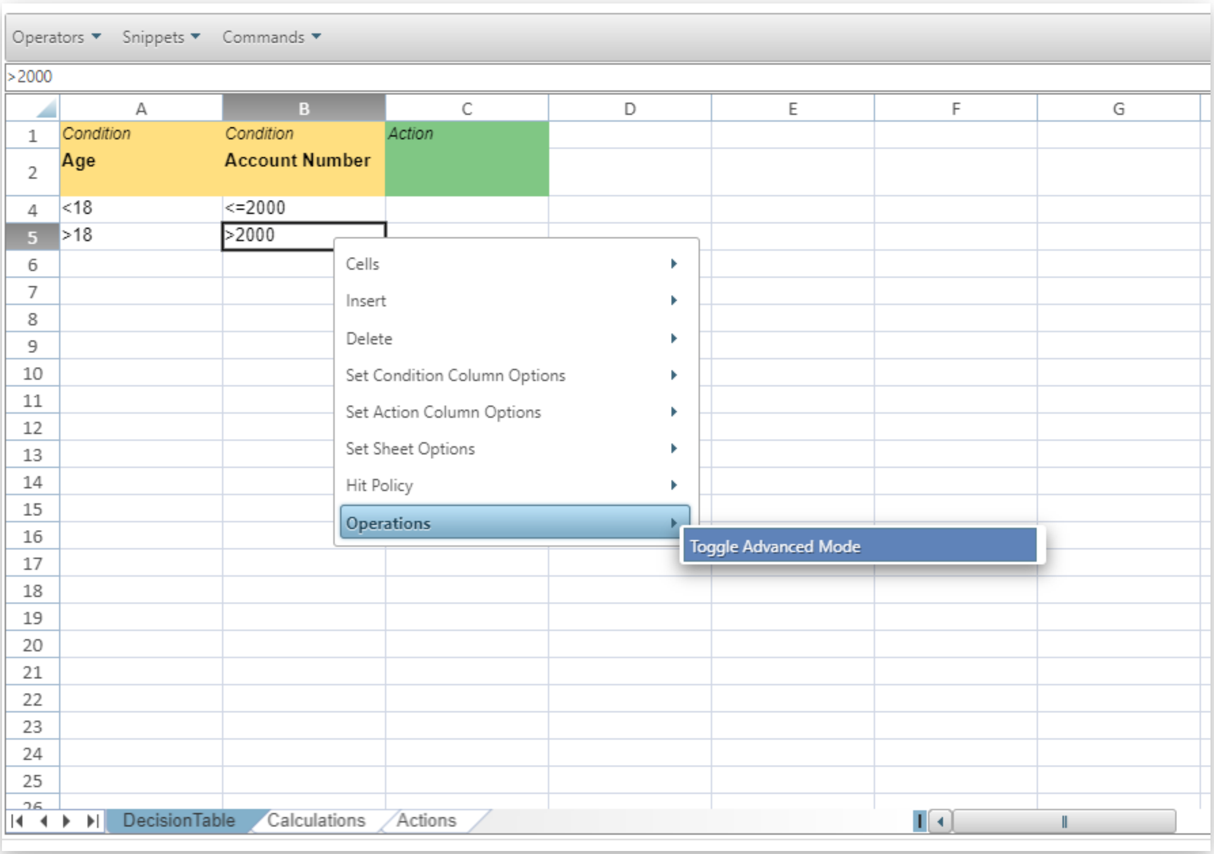
It is assumed you already have some familiarity with Decision Tables or have at least read 'How to - Create your first Decision Table'

What are Conditions?

A condition is what triggers an action, it can be as simple as checking the value of a field or as complex as a multi-field, multiple function lookup. Decision Tables give you the **flexibility** and the **power** to allow you to adapt your conditions to whatever you are trying to achieve.

By default when you open a Decision Table it shows you 1 Condition.

You can add additional conditions by *right-clicking* on the Decision Table editor and selecting *Insert > Insert Condition* from the contextual pop-up menu.



>2000

	A	B	C
1	*Condition*	*Condition*	*Action*
2	**Age**	**Account Number**	
4	<18	<=2000	
5	>18	>2000	

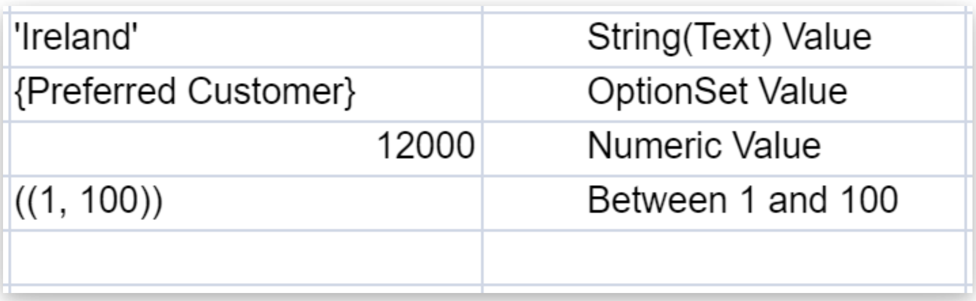
After adding your required Condition columns you will need to populate them.

Condition Values

Before we continue - this is a quick overview of what information you can put in a Decision Table cell:

- If the value is in **single quotes** then it is treated as a **String value**
- If the value is in **curly brackets {}** then it is checked and treated as an **OptionSet value**
- If the value is in **curly brackets {}** and not an **OptionSetValue** then it is treated as a parameter for an In() function
- If the value is a **number** then it is treated as a **Number**
- If 2 number values are **inside parentheses separated by a comma** then they are treated as a **between**

Note: Numerical comparison operators like >, <, >=, <= are all acceptable



'Ireland'	String(Text) Value
{Preferred Customer}	OptionSet Value
12000	Numeric Value
((1, 100))	Between 1 and 100

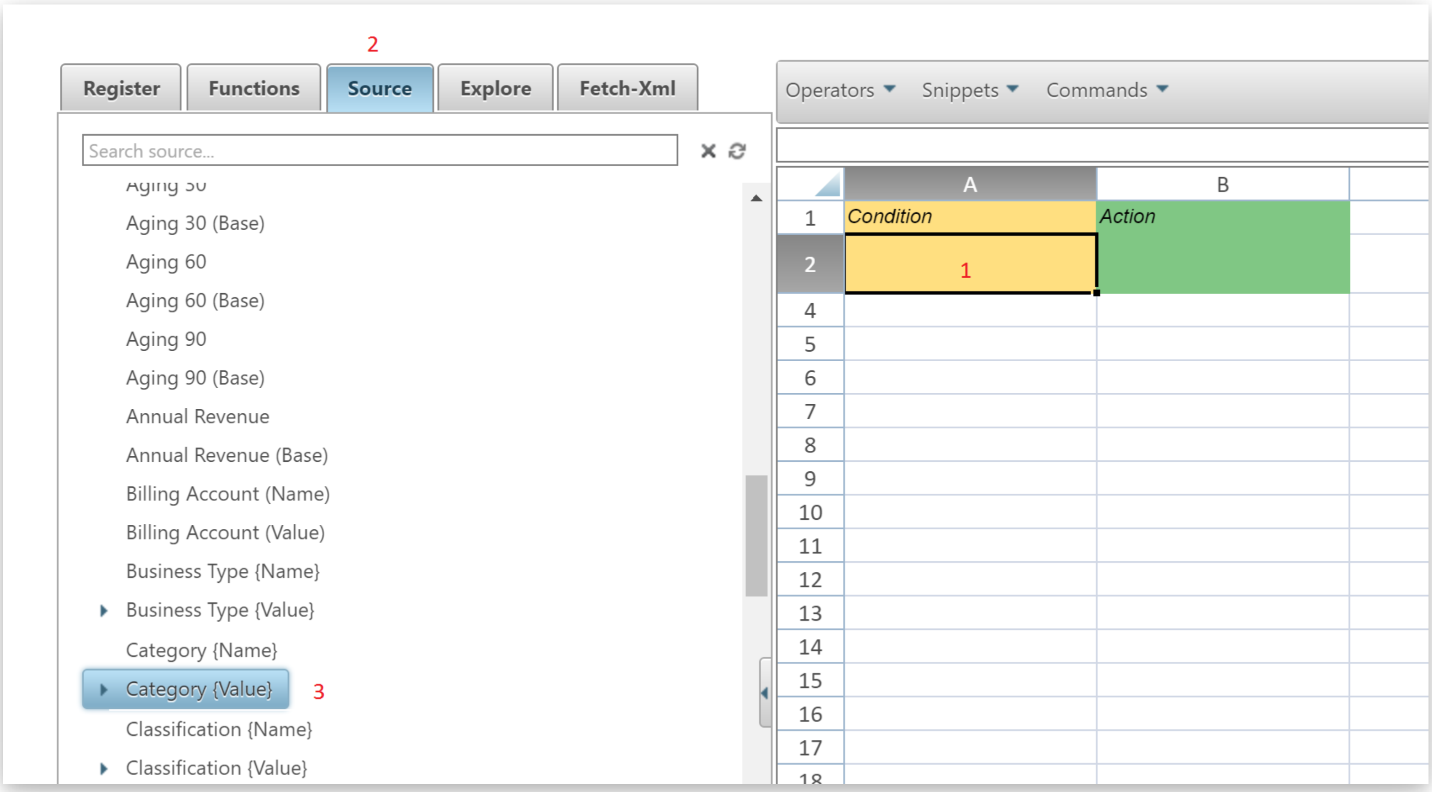
When comparing between number ranges it is possible to use **((** 100, 200 **))** for example. This will be true if the number is >= 100 and <= 200. You can use any combination of **([** and **])** if you want to exclude the outer values.
([100, 200 **))** would be >100 and <= 200 etc.

Setting a simple Condition on a Source Entity field

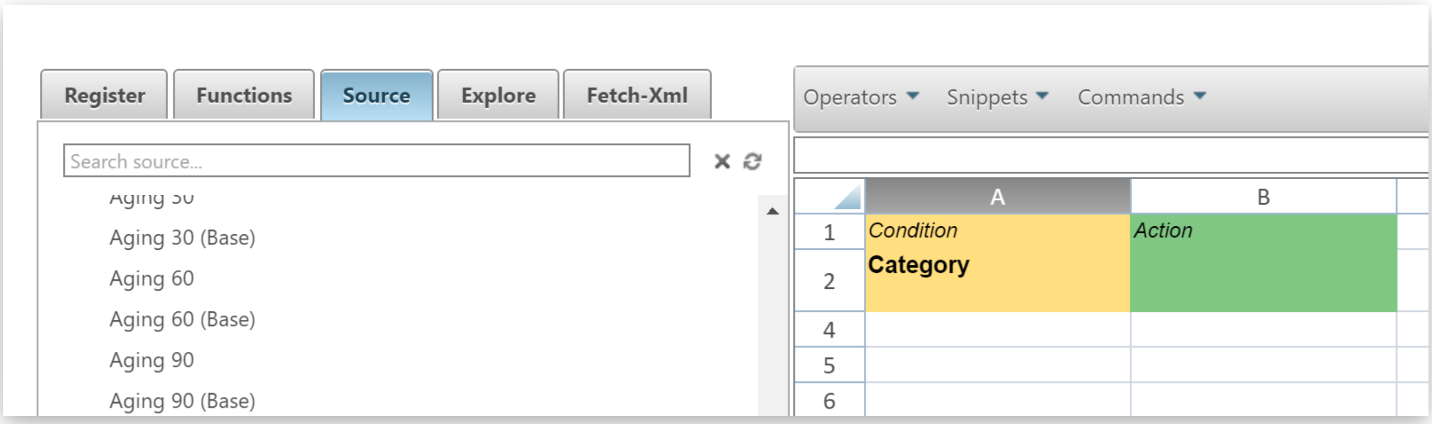
You need to associate a field to the *Condition* column so that the Decision Table can use the value of that field to 'make decisions' according to the rules you specify:

1. Click on the *yellow cell* under *Condition* on the Decision Table
2. Click on the *Source* tab on the configuration pane
3. Click on the *name of the field* you want to set the Condition on
4. The name of that field will appear in **bold text** in the **yellow cell** under *Condition*

For example, if we wanted to set **Category** as a condition on a Decision Table for the ***Account*** entity:

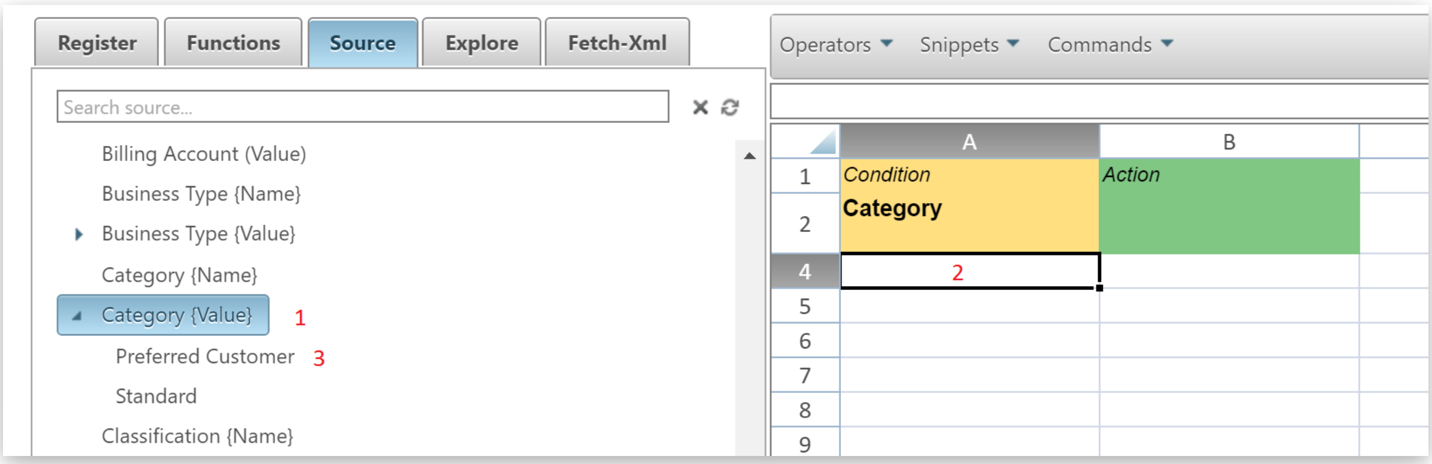


Category would then appear in bold text in cell **A2**.

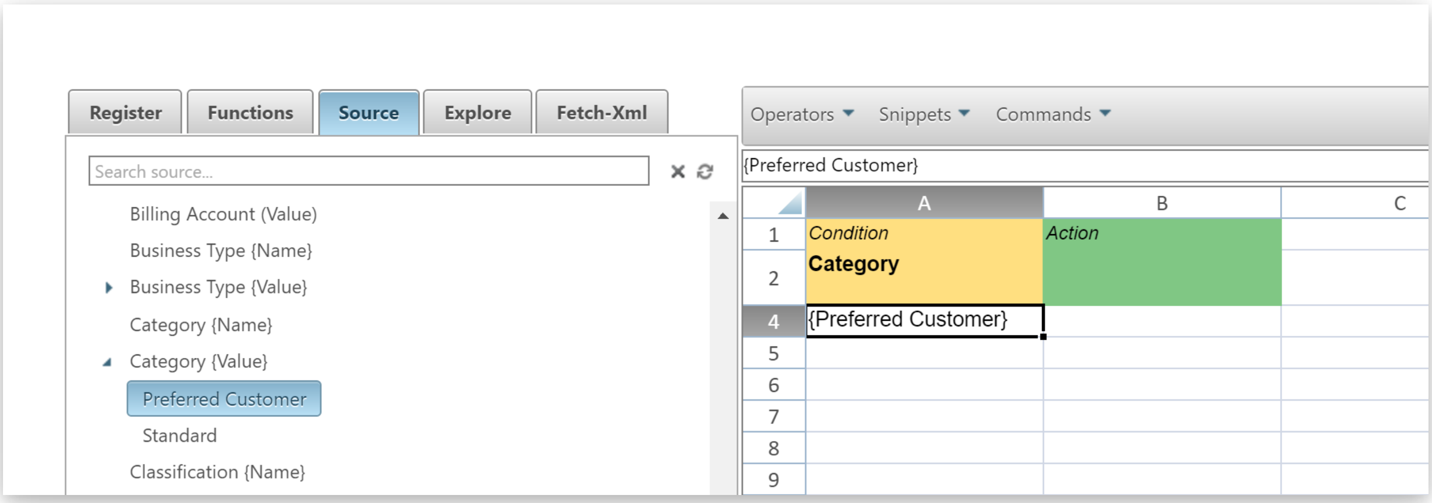


	A	B
1	*Condition*	*Action*
2	**Category**	
4		
5		
6		

Now that we have selected Category as our Condition we need to set the values for Category that we want the Decision Table to act on. *Expand* the list of values available for **Category** and then click on *cell A4* and then click on *Preferred Customer* from the **Category** values.



	A	B
1	*Condition*	*Action*
2	**Category**	
4	2	
5		
6		
7		
8		
9		



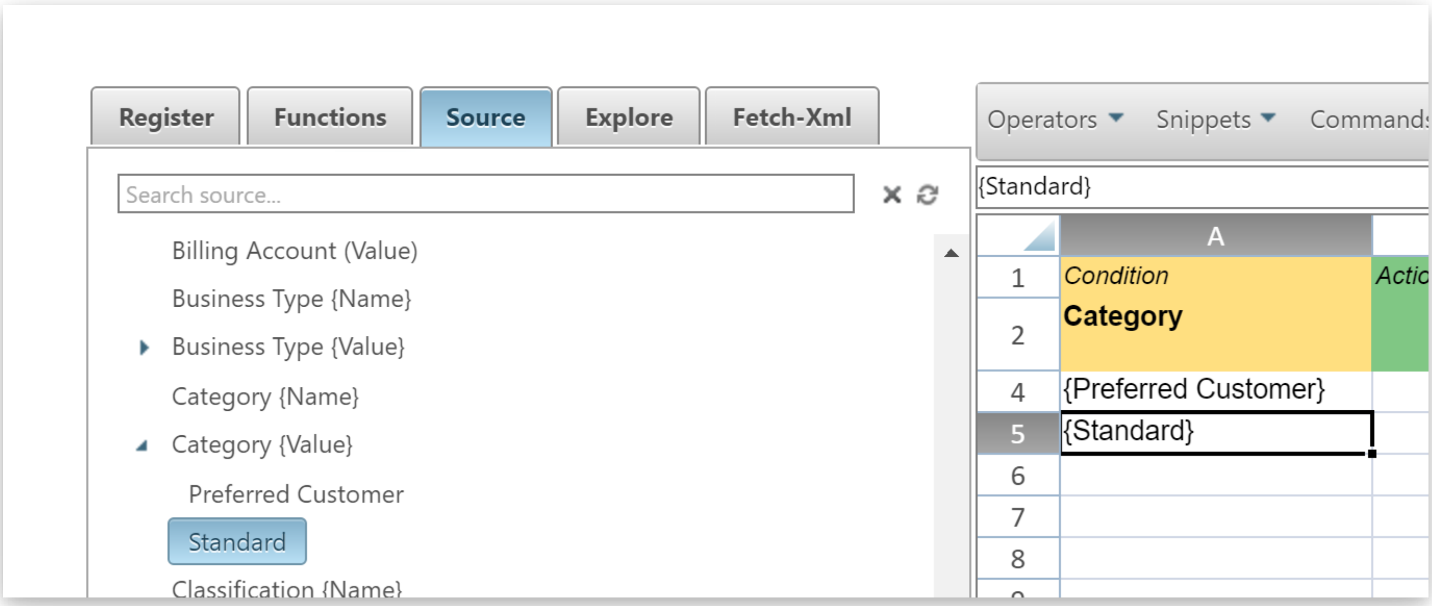
{Preferred Customer}

	A	B	C
1	*Condition*	*Action*	
2	**Category**		
4	{Preferred Customer}		
5			
6			
7			
8			
9			

Preferred Customer appears in **cell A4** surrounded by curly brackets.

Note: The curly brackets mean that this is an OptionSet value.

If you repeat the procedure for **cell A5** and click on *Standard* then this will populate that cell with **{Standard}**:

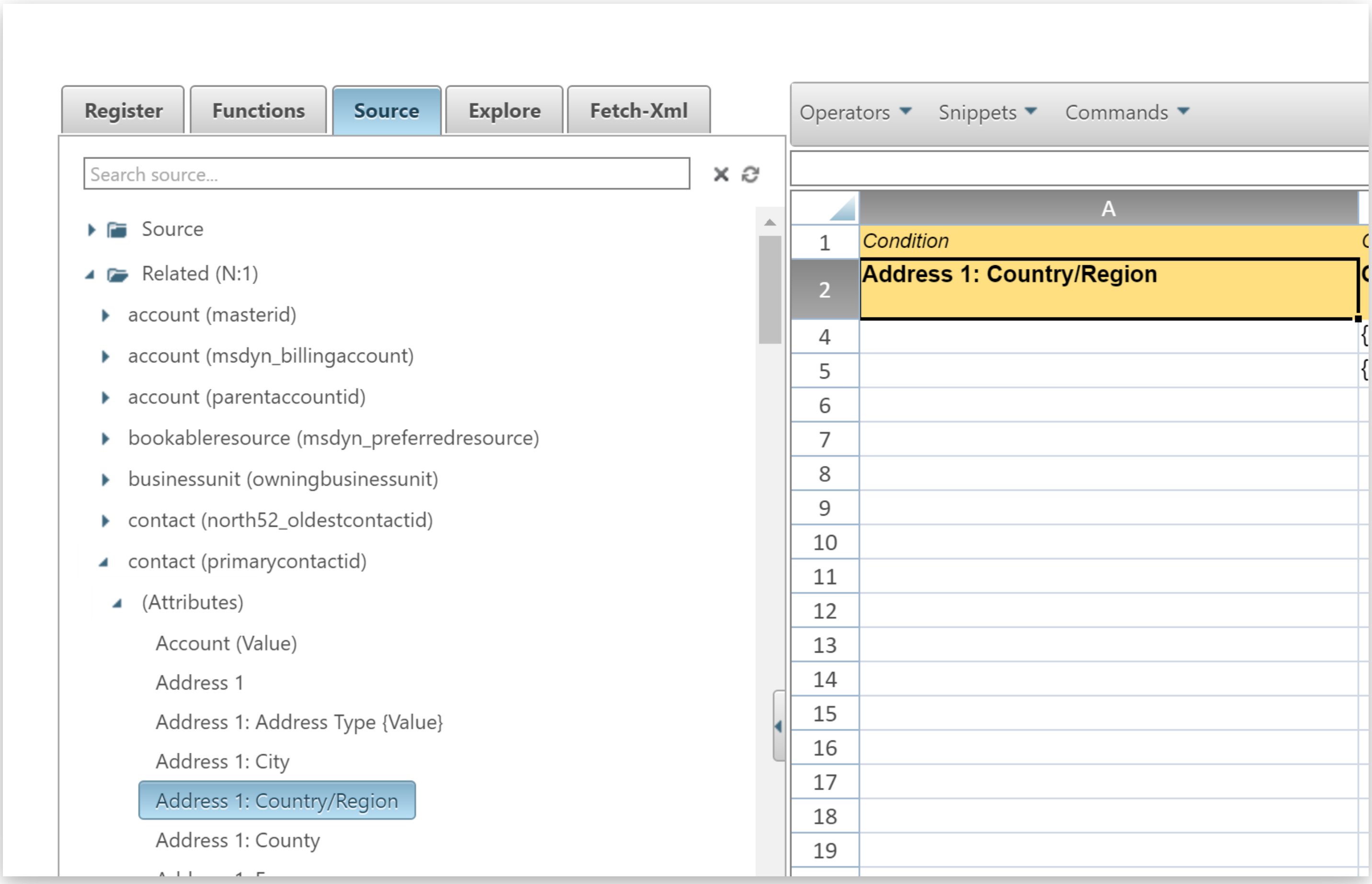


Setting a Condition on a related entity field

If you need to set a Condition based on a field in a related entity then you can do the following:

- Add a *Condition* column to the Decision Table if needed
- Click on the *yellow cell* under *Condition* (in row 2)
- Click on the *Source* tab on the configuration pane
- Expand the *Related (N:1)* tree node and click on the field you want to select

In this example we are adding a condition to the Decision Table that looks up the **Country/Region** field of the Primary *Contact* record connected to the *Account* record the Decision Table is executing on.



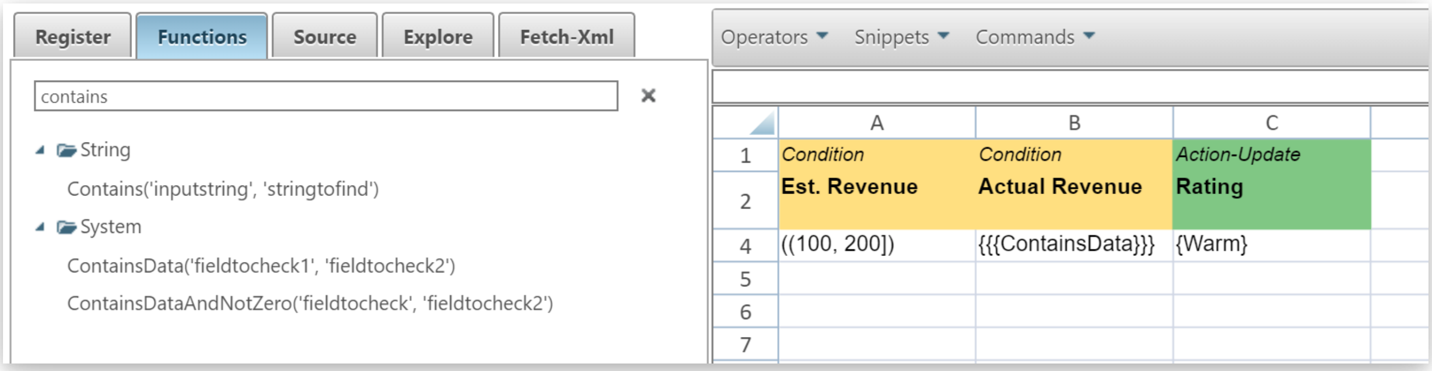
Source
Related (N:1)
account (masterid)
account (msdyn_billingaccount)
account (parentaccountid)
bookableresource (msdyn_preferredresource)
businessunit (owningbusinessunit)
contact (north52_oldestcontactid)
contact (primarycontactid)
(Attributes)
Account (Value)
Address 1
Address 1: Address Type (Value)
Address 1: City
Address 1: Country/Region
Address 1: County

	A
1	Condition
2	Address 1: Country/Region

Adding a Check Condition to a Field

In certain scenarios, you need to check if a field contains data, or does not contain data etc. With Decision Tables, this is easy to do. Instead of writing a separate **Calculation** you add the field as a **condition column** as normal, then find the function you need in the **functions tab.** Hold down the **CTRL** key and click on the function - it will be populated into the **Decision Table** inside {{{ }}} notation.

In the screenshot below, we are checking if the **Estimated Revenue** is greater than or equal to 100 and less than 200, and the **Actual Revenue** field contains data.



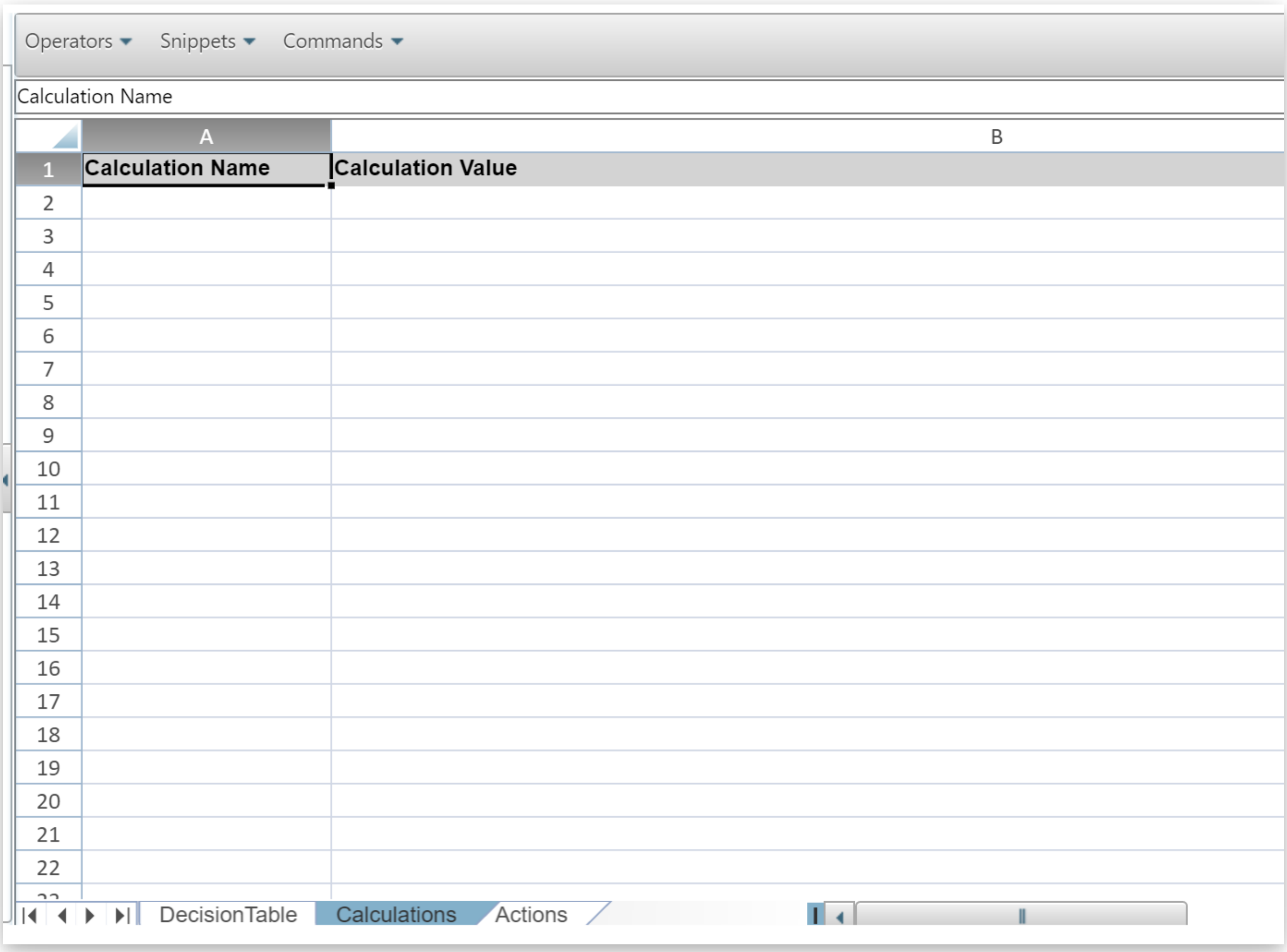
String
Contains('inputstring', 'stringtofind')
System
ContainsData('fieldtocheck1', 'fieldtocheck2')
ContainsDataAndNotZero('fieldtocheck', 'fieldtocheck2')

	A	B	C
1	Condition	Condition	Action-Update
2	Est. Revenue	Actual Revenue	Rating
4	((100, 200])	{{{ContainsData}}}	{Warm}
5			
6			
7			

Adding more Complex Conditions

Decision Tables allow you to include more complex Conditions than just selecting from the Source tab.

You can write formulas that can return values to be checked against. These are done in the **Global Calculations** tab of the Decision Table (select the tab from the bottom of the Decision Table editor - this tab is hidden by default and can be shown by pressing **F4** or right-clicking and selecting **Operations > Toggle Advanced Mode**).



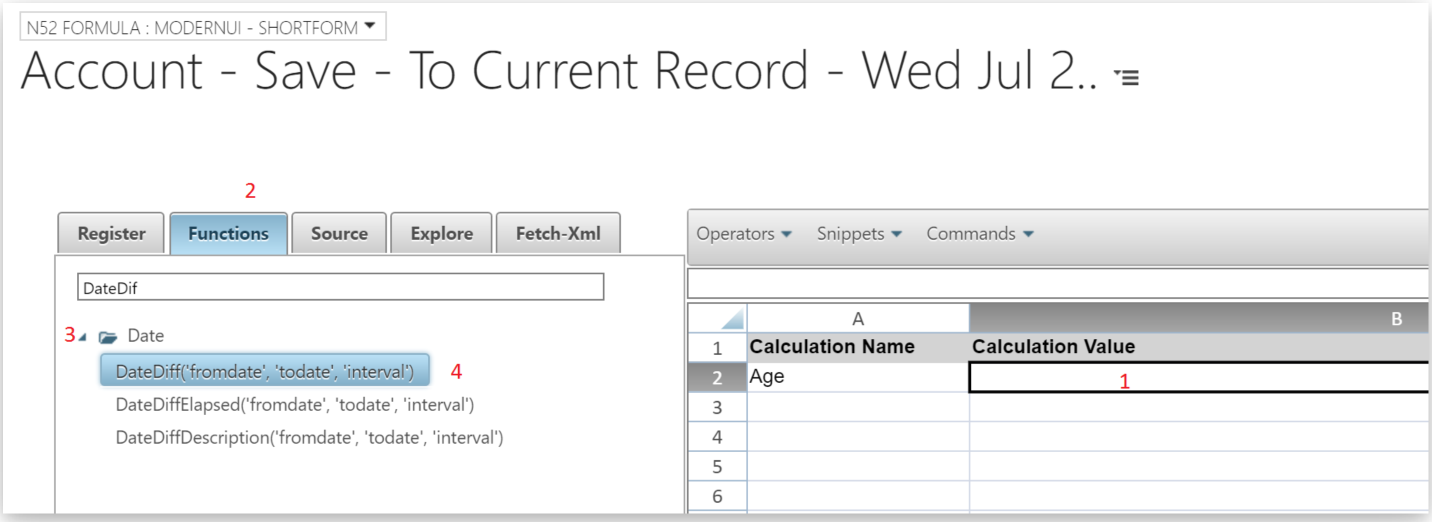
	A	B
1	Calculation Name	Calculation Value

An example of when you would use this functionality is if you needed to check the age of a Contact as part of the Condition.

To accomplish this you would set the **Calculation Name** as *Age* (you can call this whatever you like, but make sure it is meaningful for someone reading the Decision Table). The calculation formula is entered as you would any standard

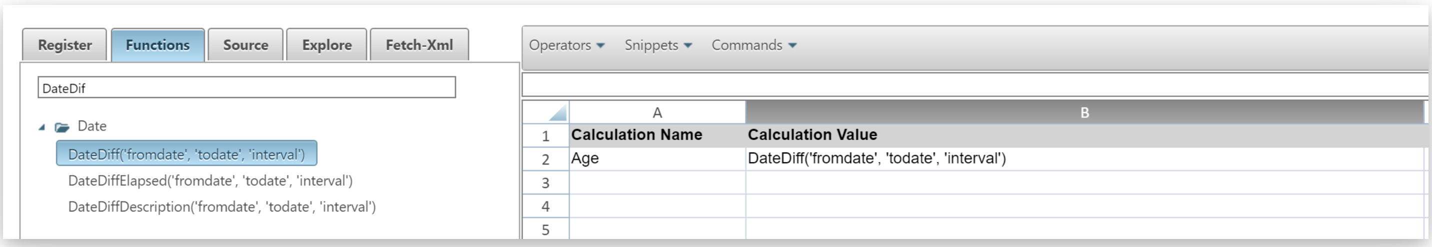
North52 Formula:

Click on *cell B2*, then click on the *Functions* tab and expand the **Date** node and click on the *DateDiff()* function.


N52 FORMULA : MODERNUI - SHORTFORM
Account - Save - To Current Record - Wed Jul 2..
2
Register
Functions
Source
Explore
Fetch-Xml
DateDif
3
Date
DateDiff('fromdate', 'todate', 'interval')
4
DateDiffElapsed('fromdate', 'todate', 'interval')
DateDiffDescription('fromdate', 'todate', 'interval')
Operators
Snippets
Commands
1

	A	B
1	Calculation Name	Calculation Value
2	Age	

This will put the **DateDiff** function template into **cell B2**.

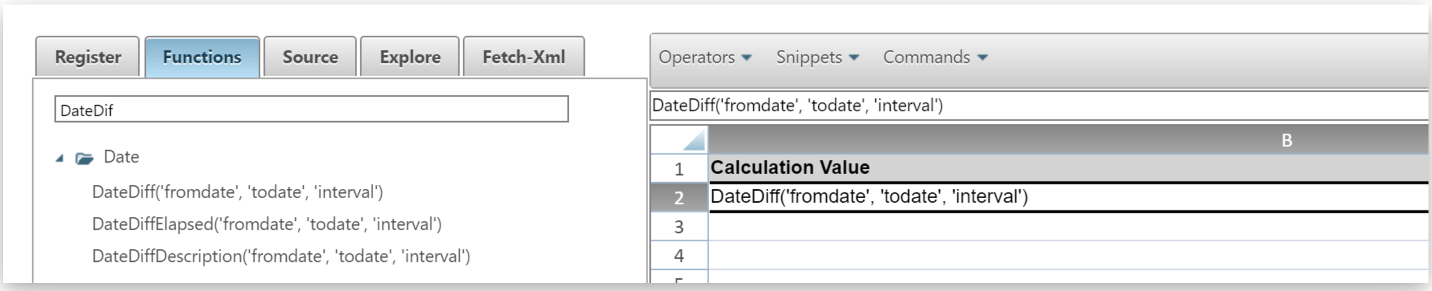


DateDif

Date
- DateDiff('fromdate', 'todate', 'interval')
- DateDiffElapsed('fromdate', 'todate', 'interval')
- DateDiffDescription('fromdate', 'todate', 'interval')

	A	B
1	Calculation Name	Calculation Value
2	Age	DateDiff('fromdate', 'todate', 'interval')
3		
4		
5		

Click on **B2** again and the formula will appear in the **command editing bar**.



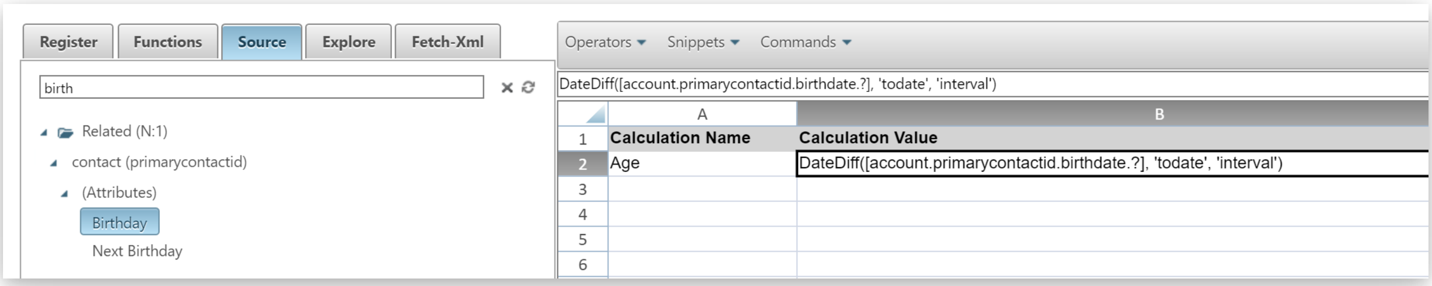
DateDif

DateDiff('fromdate', 'todate', 'interval')

Date
- DateDiff('fromdate', 'todate', 'interval')
- DateDiffElapsed('fromdate', 'todate', 'interval')
- DateDiffDescription('fromdate', 'todate', 'interval')

	B
1	Calculation Value
2	DateDiff('fromdate', 'todate', 'interval')
3	
4	

Now delete *'fromdate'* from the formula until you are left with **DateDiff(, 'todate', 'interval')**
Place the cursor before the the first comma and expand the Source tab looking for the **Birthday** field. In this example we are getting the **Birthday** field of the primary ***Contact*** record on the ***Account***.



birth

DateDiff([account.primarycontactid.birthdate.?], 'todate', 'interval')

Related (N:1)
- contact (primarycontactid)
  - (Attributes)
    - Birthday
    - Next Birthday

	A	B
1	Calculation Name	Calculation Value
2	Age	DateDiff([account.primarycontactid.birthdate.?], 'todate', 'interval')
3		
4		
5		
6		

Now repeat the procedure for the *'todate'* and replace it with the function *Utcdate()* and finally replace *'interval'* with *'y'*

This formula will return the age in years of the primary Contact on the Account record.

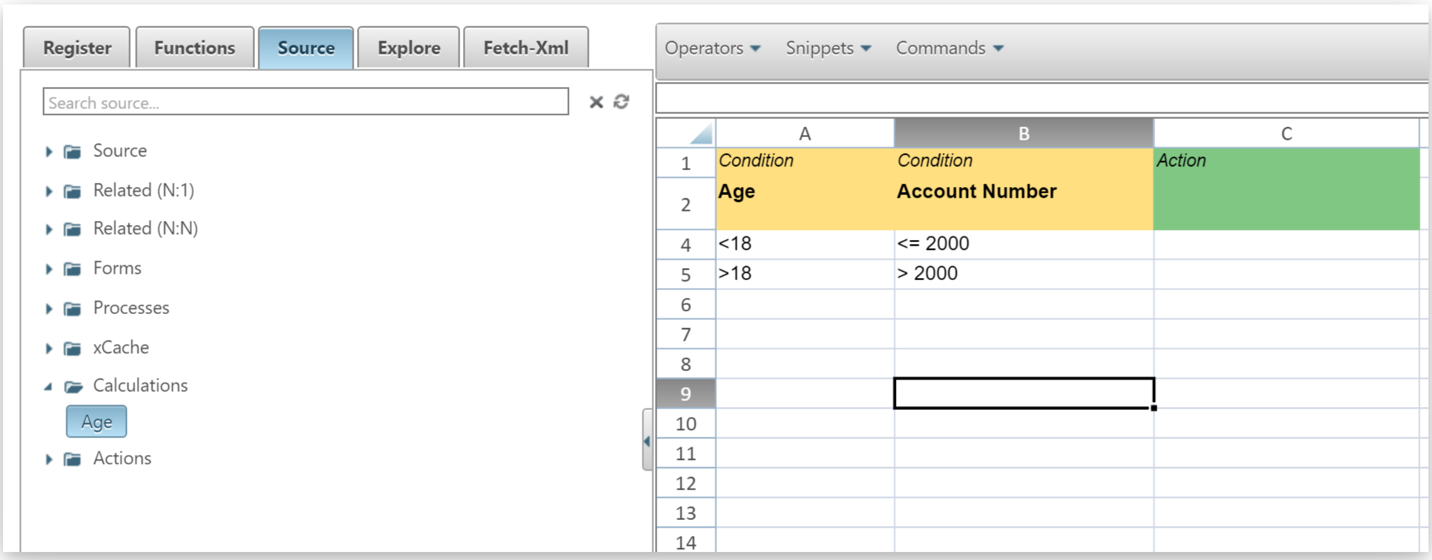
To add this as a Condition on the Decision Table:

- Click back to the Decision Table, by clicking the *Decision Table tab* at the bottom of the editor
- Add a new **Condition** column if needed, by right-clicking on the editor canvas and selecting *Insert > Insert Condition*
- Click on the *yellow cell* under ***Condition*** (in row 2)
- Click on the *Source* tab on the configuration pane
- Expand *Calculations* and **Age** should appear there (or whatever you named your calculation)
- Click *Age*

Age should now appear in bold text for the ***Condition***.

In this example the conditions are now:

- if age is less than 18 and the account number is less than or equal to 2000
- if age is greater than 18 and the account number is greater than 2000

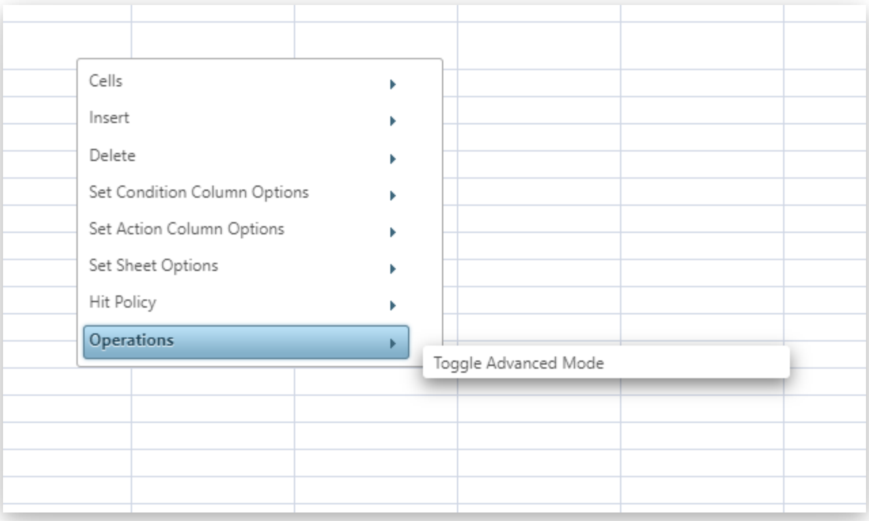


	A	B	C
1	Condition	Condition	Action
2	**Age**	**Account Number**	
4	<18	<= 2000	
5	>18	> 2000	
6			
7			
8			
9			
10			
11			
12			
13			
14			

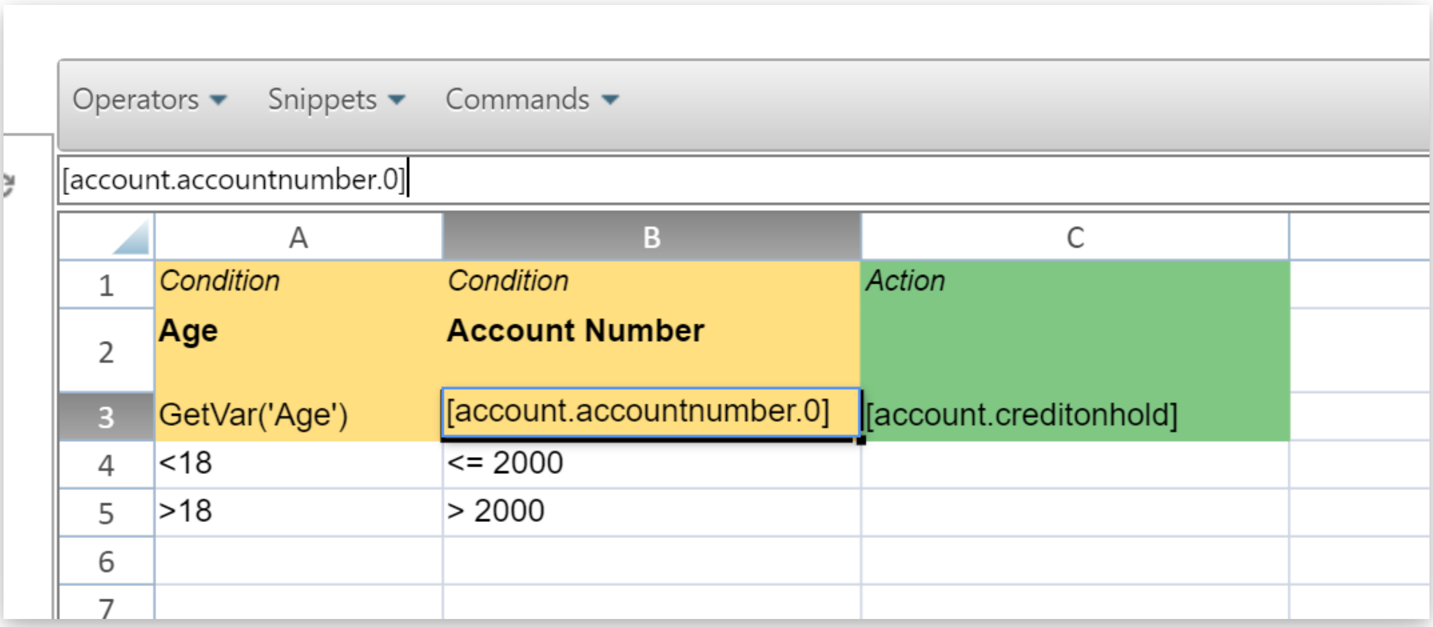
Setting a Default value for a Condition

To learn more about default values - please review this article https://support.north52.com/knowledgebase/articles/473859-how-to-handle-null-or-empty-values-with-north52

You can set a default value for any of your conditions by **right-clicking** on the Decision Table editor canvas and clicking on *Operations > Toggle Advanced Mode* from the contextual menu. This will display another row (row 3, which is normally hidden) under your *Conditions.*



By adding a **dot** and a **0** to the Account Number condition it means that if a null value is found in that field, the Decision Table will use a zero instead thereby protecting your formula from breaking.



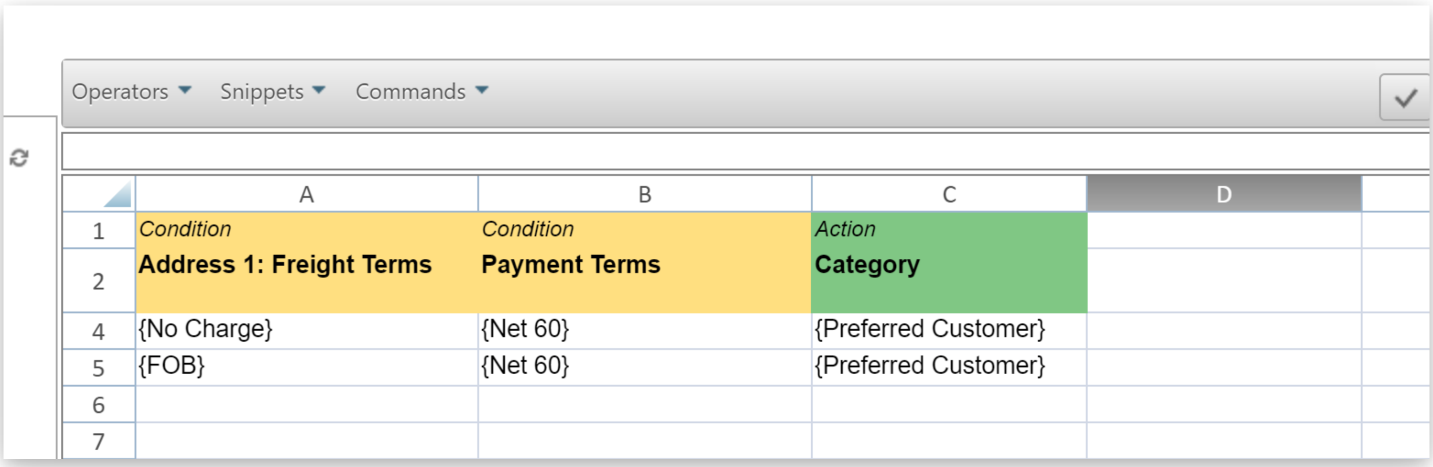
[account.accountnumber.0]

	A	B	C
1	*Condition*	*Condition*	*Action*
2	**Age**	**Account Number**	
3	GetVar('Age')	[account.accountnumber.0]	[account.creditonhold]
4	<18	<= 2000	
5	>18	> 2000	
6			
7			

Setting OR Conditions on an OptionSet Value

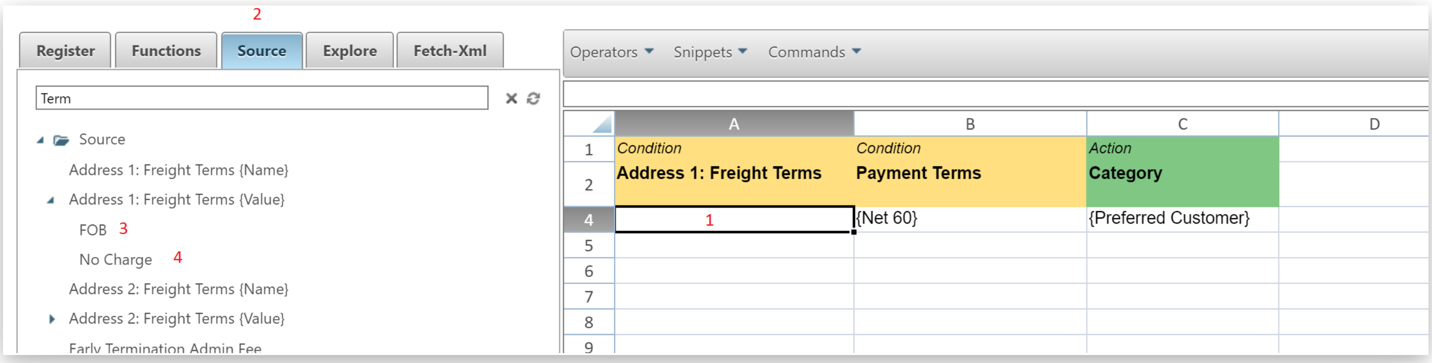
In this example we want to update the account **Category** field to *Preferred Customer* if the **Payment Terms** are *Net 60*and the **Freight Terms** are *No Charge* or *FOB*.

Without using an an OR we could simply do the below.



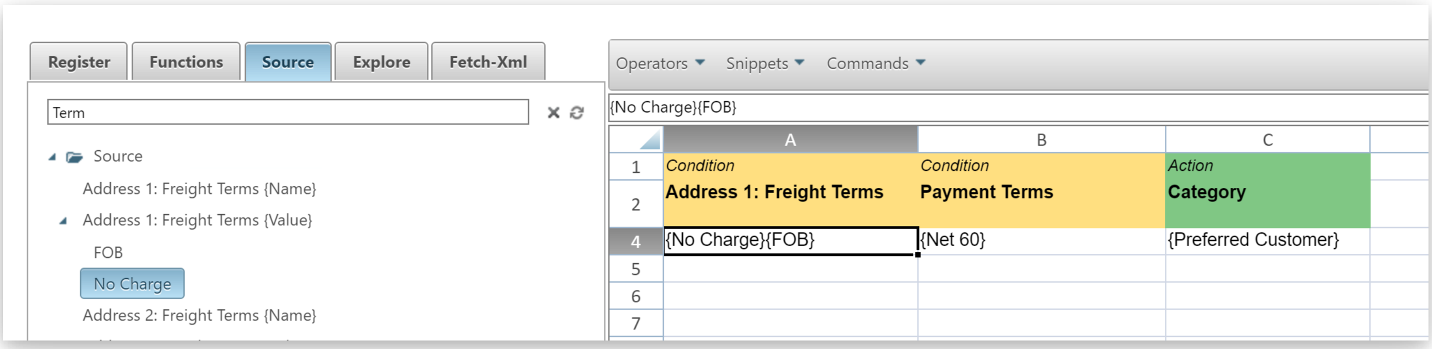
	A	B	C	D
1	*Condition*	*Condition*	*Action*	
2	**Address 1: Freight Terms**	**Payment Terms**	**Category**	
4	{No Charge}	{Net 60}	{Preferred Customer}	
5	{FOB}	{Net 60}	{Preferred Customer}	
6				
7				

However there is an easier way to achieve this by putting **both OptionSet Values into the same field**. After clicking on *No Charge,* immediately click on *FOB* as well.


2
Register
Functions
Source
Explore
Fetch-Xml
Term
Source
Address 1: Freight Terms {Name}
Address 1: Freight Terms {Value}
FOB 3
No Charge 4
Address 2: Freight Terms {Name}
Address 2: Freight Terms {Value}
Early Termination Admin Fee
Operators
Snippets
Commands

	A	B	C	D
1	*Condition*	*Condition*	*Action*	
2	**Address 1: Freight Terms**	**Payment Terms**	**Category**	
4	1	{Net 60}	{Preferred Customer}	
5				
6				
7				
8				
9				

Now both OptionSet Values will appear in **Address 1: Freight Terms** - meaning that *No Charge* or *FOB* and *Net 60* are selected then the **Category** will be updated to *Preferred Customer*.


Register
Functions
Source
Explore
Fetch-Xml
Term
Source
Address 1: Freight Terms {Name}
Address 1: Freight Terms {Value}
FOB
No Charge
Address 2: Freight Terms {Name}
Operators
Snippets
Commands

{No Charge}{FOB}

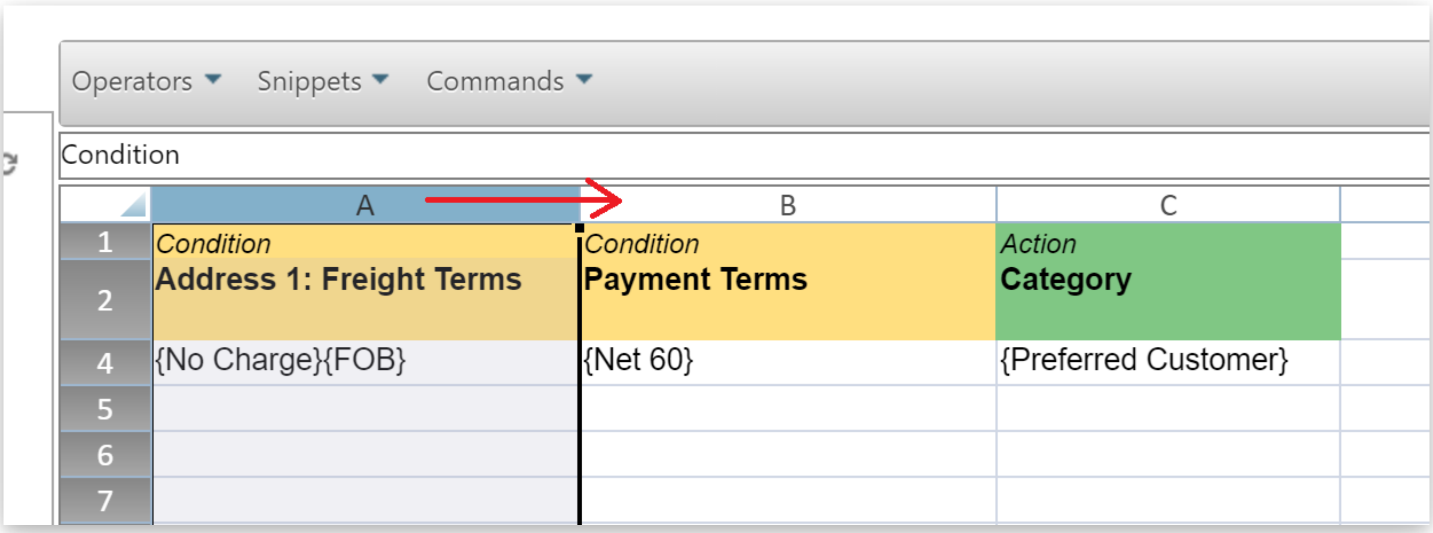
	A	B	C
1	*Condition*	*Condition*	*Action*
2	**Address 1: Freight Terms**	**Payment Terms**	**Category**
4	{No Charge}{FOB}	{Net 60}	{Preferred Customer}
5			
6			
7			

Setting OR conditions on different fields

In this example in order to be a *Preferred Customer* the **Freight Terms** can be either *No Charge* or *FOB* or the **Payment Terms** can be marked as *Net 60.*

It's very easy to change the above example to accommodate this requirement:
Click on the **letter** of the first column you want to add to the OR statement - this will highlight the entire column.

Keep the left mouse button pressed and drag your mouse over to the column you want to add into the OR statement.

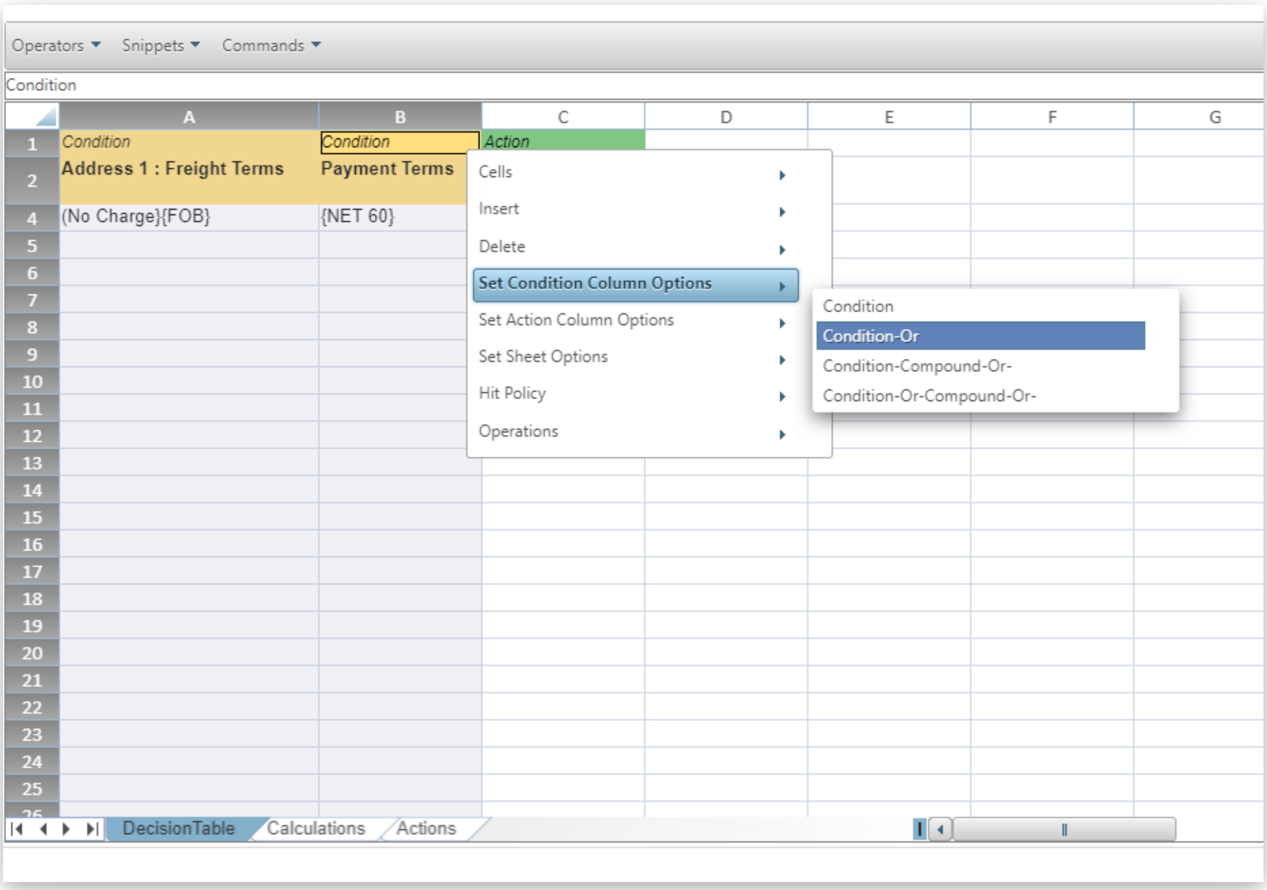


Condition

	A	B	C
1	*Condition*	*Condition*	*Action*
2	**Address 1: Freight Terms**	**Payment Terms**	**Category**
4	{No Charge}{FOB}	{Net 60}	{Preferred Customer}
5			
6			
7			

Note: *You will have to position the columns you want to put into the OR statement next to each other.*

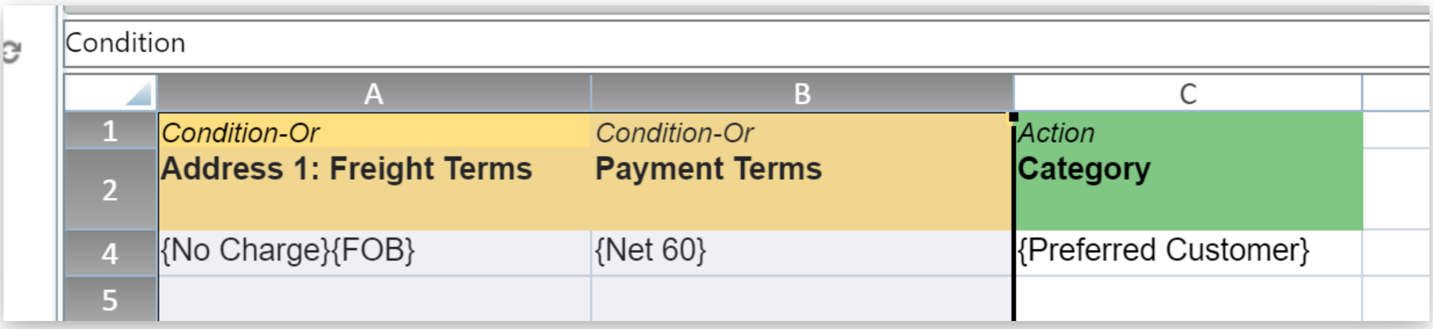
In the below example you can now see that **both column A and column B are highlighted**. Next *right-click* on the editor canvas and from the pop-up menu click *Set Condition Column Options > Condition-Or*



Condition

	A	B	C
1	Condition	Condition	Action
2	Address 1 : Freight Terms	Payment Terms	
4	{No Charge}{FOB}	{NET 60}	

Column A and Column B are no longer showing as **Condition** but as **Condition-OR**.



Condition

	A	B	C
1	Condition-Or	Condition-Or	Action
2	Address 1: Freight Terms	Payment Terms	Category
4	{No Charge}{FOB}	{Net 60}	{Preferred Customer}
5			

Both of these columns are now successfully OR'd together.

Multiple Sets of OR Conditions

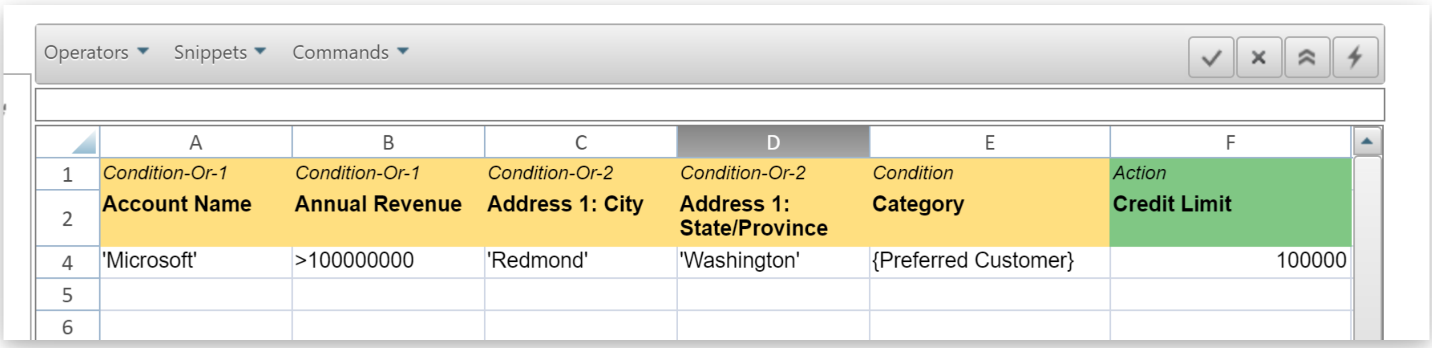
It is possible to chain columns of OR conditions together.

Scenario: If the following Conditions are met:

- Account **Name** is *Microsoft* **OR** the **Annual Revenue** is *greater than $100,000,000*
- AND
- **Address 1 City** is *Redmond* **OR** Address 1 State/Province is Washington
- AND
- **Category** set to *Preferred Customer*

Then sent the Credit Limit on the Account to $100,000.

To do this, you set your Conditions up together and group them as an Or Condition as demonstrated above.
Then edit the cells that contain **Condition-Or** and *add a hyphen and a number* to them, this number then groups the OR statements together.



	A	B	C	D	E	F
1	*Condition-Or-1*	*Condition-Or-1*	*Condition-Or-2*	*Condition-Or-2*	*Condition*	*Action*
2	**Account Name**	**Annual Revenue**	**Address 1: City**	**Address 1: State/Province**	**Category**	**Credit Limit**
4	'Microsoft'	>100000000	'Redmond'	'Washington'	{Preferred Customer}	100000
5						
6						

In this example **Account Name** and **Annual Revenue** are the first *Condition-Or-1*. And then **Address 1 City** and **Address 1 State/Province** are the second *Condition-Or-2*.

DT - How to - 04 - Set up your Actions

[TOC]

Overview

This article will detail everything you need to know about **Actions** in North52 Decision Tables.

What is an Action?

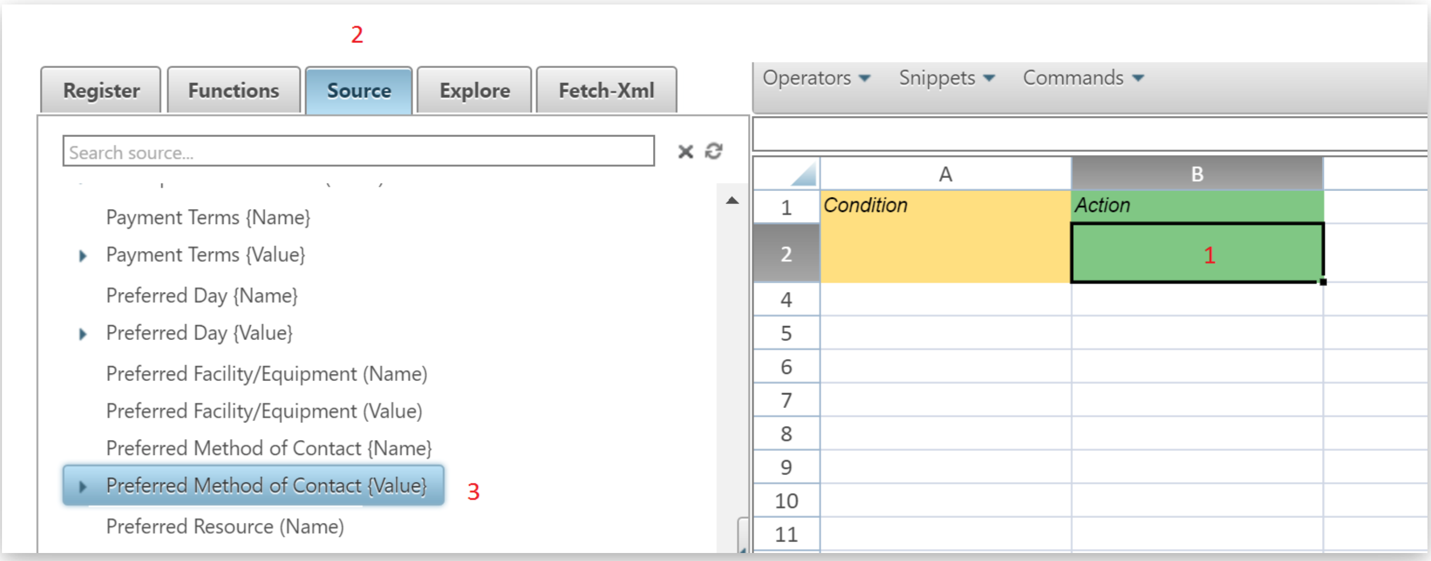
The Action is what you want the Decision Table to do when the Conditions have been met.
It could be anything from setting a value in a field, creating a new entity, hiding a section on a form to executing a workflow.

How to set a Simple Action

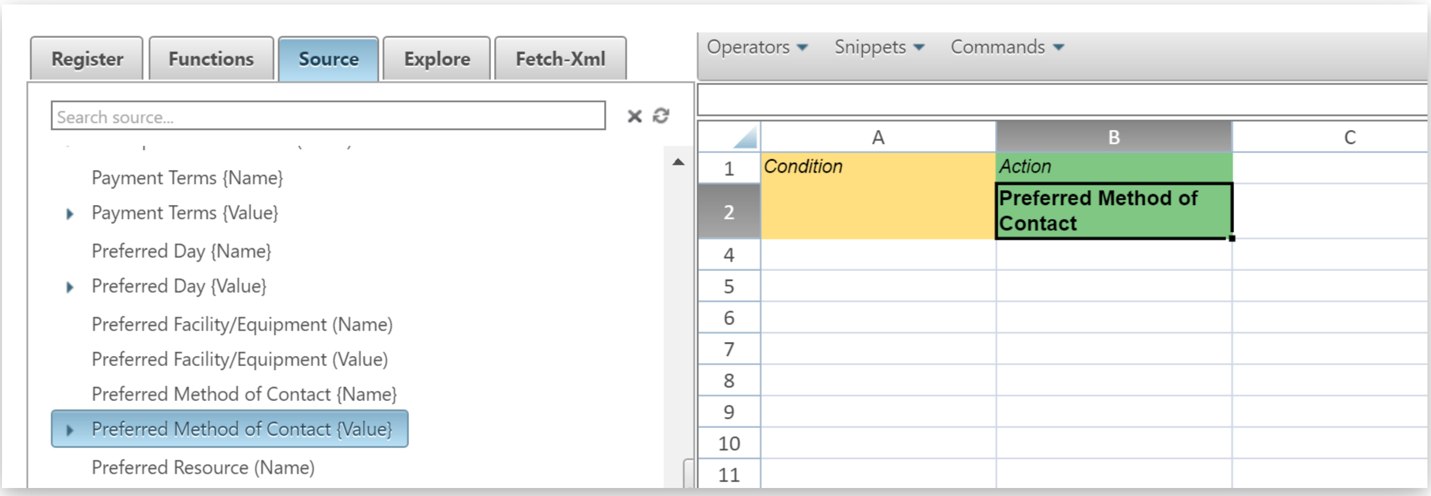
To set an action:

1. Click on the *green cell* under *Action* on the Decision Table
2. Click on the *Source* tab and click on the *name of the field* you want to update

For example to set an Action that updates the **Preferred Method of Contact** on an *Account*:



- Click on the *green cell* (row 2) under the **Action** heading
- Click on the *Source* tab on the configuration pane
- Click on the *Preferred Method of Contact* field

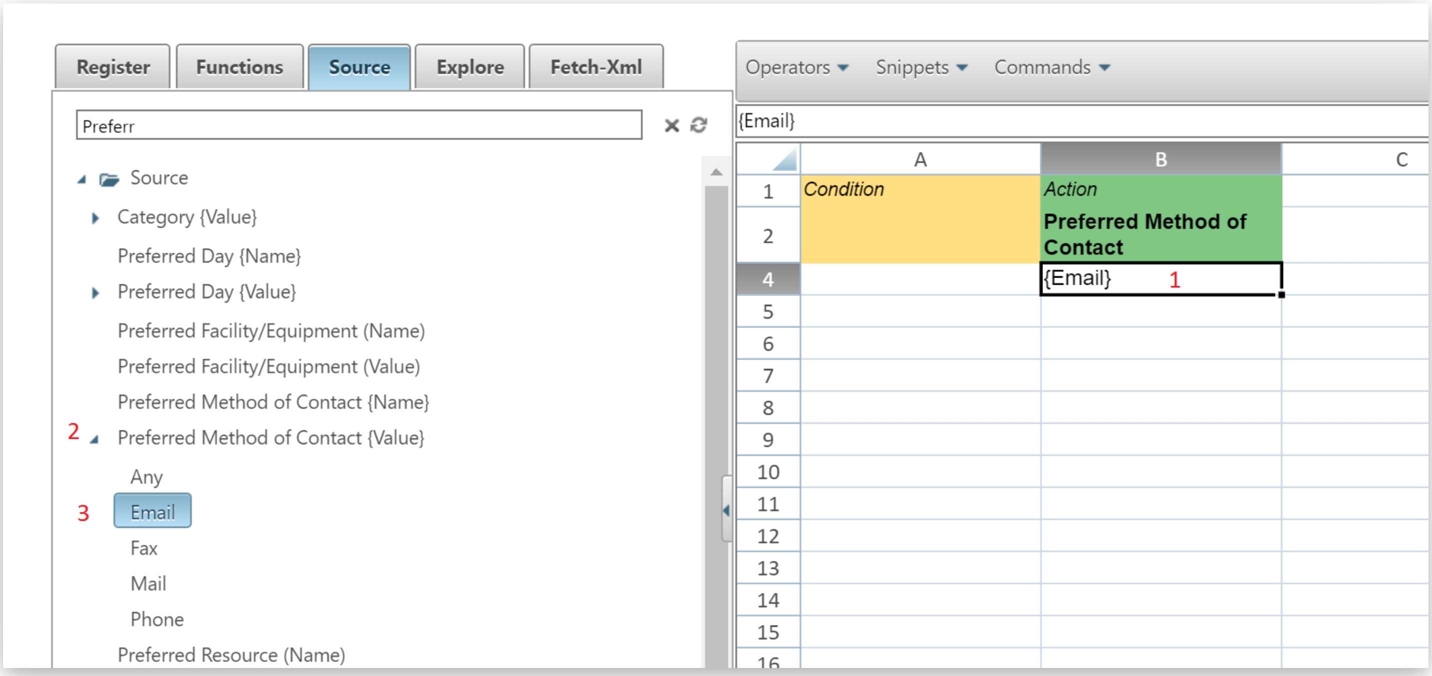

Register
Functions
Source
Explore
Fetch-Xml
Operators
Snippets
Commands
Search source...
Payment Terms {Name}
Payment Terms {Value}
Preferred Day {Name}
Preferred Day {Value}
Preferred Facility/Equipment (Name)
Preferred Facility/Equipment (Value)
Preferred Method of Contact {Name}
Preferred Method of Contact {Value}
Preferred Resource (Name)
A
B
C
Condition
Action
Preferred Method of Contact

Preferred Method of Contact will now appear in the green cell in the **Action** column.

This tells the Decision Table that if the *Conditions* are met then the *Action* to be taken will affect this field.

To make a specific update to the **Preferred Method of Contact** Action we click an *empty cell* in that column and then *expand* the **Preferred Method of Contact** node in the **Source** tab of the configuration pane to select the desired result.

In the below screenshot *Email* has been selected and **{Email}** has appeared in the **Cell B4**.



Register
Functions
Source
Explore
Fetch-Xml
Preferr
Source
Category {Value}
Preferred Day {Name}
Preferred Day {Value}
Preferred Facility/Equipment (Name)
Preferred Facility/Equipment (Value)
Preferred Method of Contact {Name}
2
Preferred Method of Contact {Value}
Any
3
Email
Fax
Mail
Phone
Preferred Resource (Name)
Operators
Snippets
Commands
{Email}
A
B
C
Condition
Action
Preferred Method of Contact
{Email}
1

Action Cell Values

You can type the following values into an *Action* cell:

- If the value is in **single quotes** then it is treated as a **String** value
- If the value is in **curly brackets {}** and the Column **has a Name** then it is treated as an **OptionSet** value
- If the value is a **number** then it is treated as a **Number**
- If the value is in **curly brackets {}** and the Column **has no Name** then it is treated as a ***Complex Action***

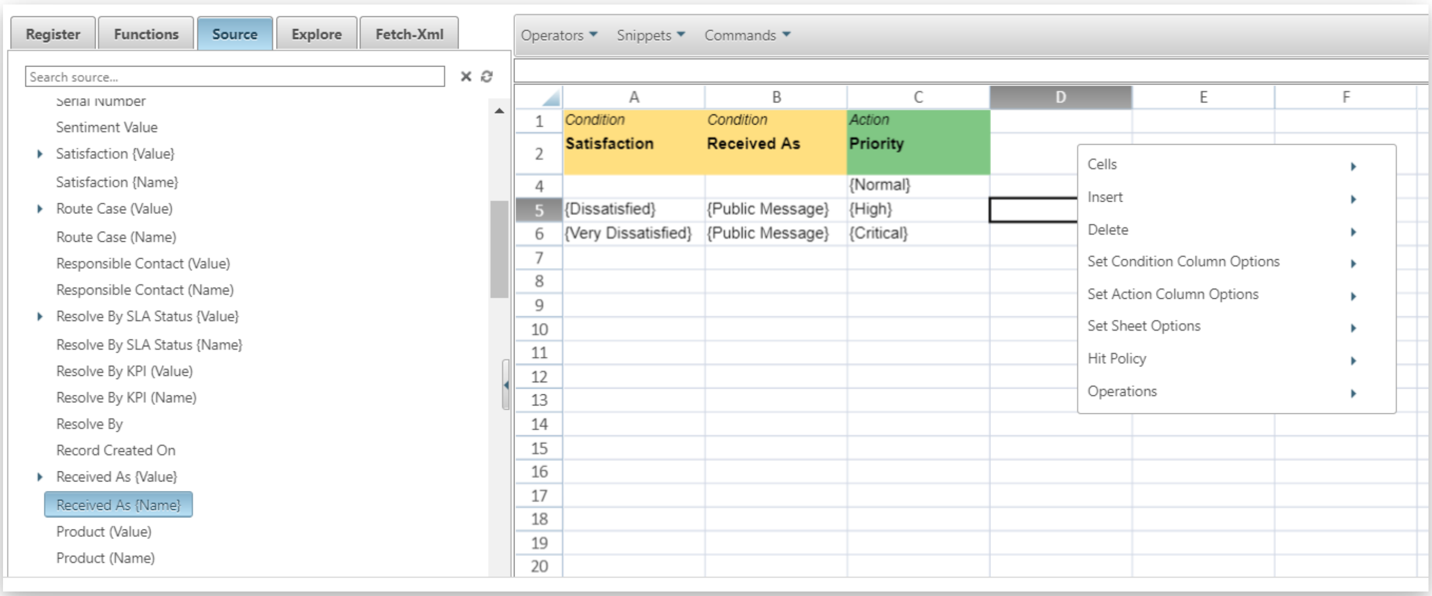
More details on *Complex Actions* can be found further down in this article.

Default Actions

If you want an Action to be completed anytime the Decision Table is triggered then you can set an Action with no Conditions associated to it.

In the example below we want to set the default priority on **all** Case entities to **Normal** when the case is created, however:

- If the Case was **Received As** a *Public message* with a *Dissatisfied* **Satisfaction** rating then the *Case* **Priority** should be *High*
- If the Case was **Received As** a *Public Message* with a *Very Dissatisfied* **Satisfaction** rating then the *Case* **Priority** should be *Critical*



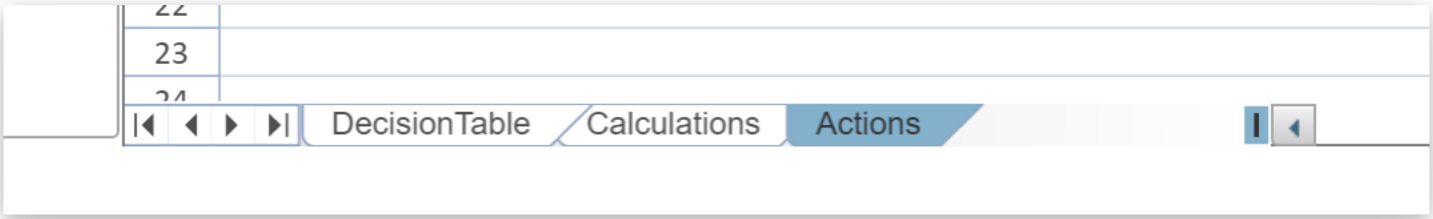
	A	B	C
1	*Condition*	*Condition*	*Action*
2	**Satisfaction**	**Received As**	**Priority**
4			{Normal}
5	{Dissatisfied}	{Public Message}	{High}
6	{Very Dissatisfied}	{Public Message}	{Critical}

Note: Exit this Decision Table on First Match is unchecked (found under Hit Policy on the right-click contextual menu) to set all cases to *Normal* **Priority** first, and then the additional conditions will be checked to determine if additional *Actions* should be carried out as well.

Complex Actions

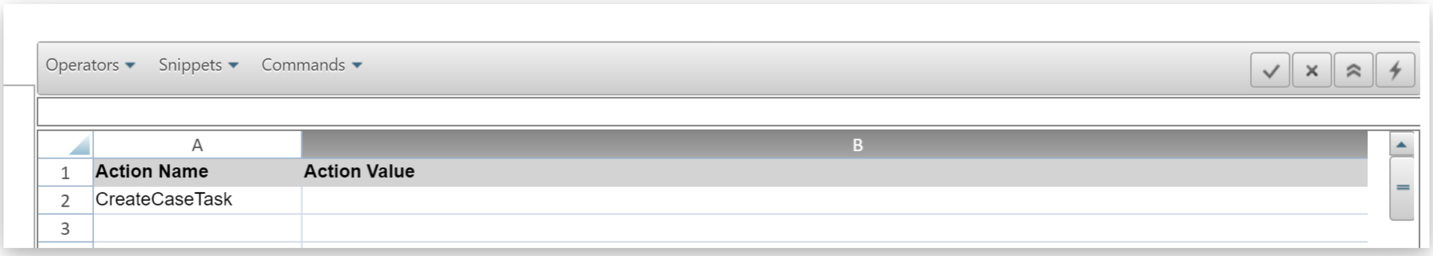
Decision Table Actions can do much more than simply update fields.

At the bottom of the Decision Table you can change to the **Actions Table** by clicking the *Actions* tab:



For this example we are going to extend the above functionality by adding an additional Action to create a Task for the owner of the case record if the *Case* **Priority** is set to *Critical*.

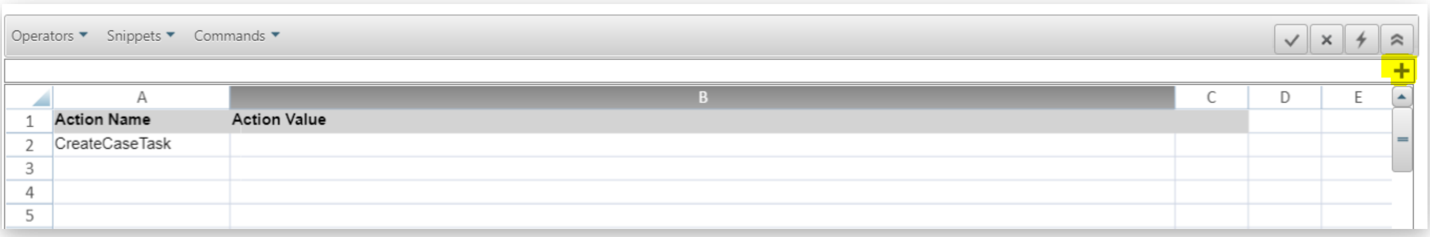
First enter the **Action Name,** this will be used by the Decision Table to identify the Action. Type in *CreateCaseTask* (you can call the action anything you like, choose something meaningful that makes it obvious what the Action will do)



	A	B
1	**Action Name**	**Action Value**
2	CreateCaseTask	
3		

Next click on the *cell* next to the Action Name (B2) in the **Action Value** column.

As we are entering a formula click the **+** icon at the right of the command bar to expand the formula editor.

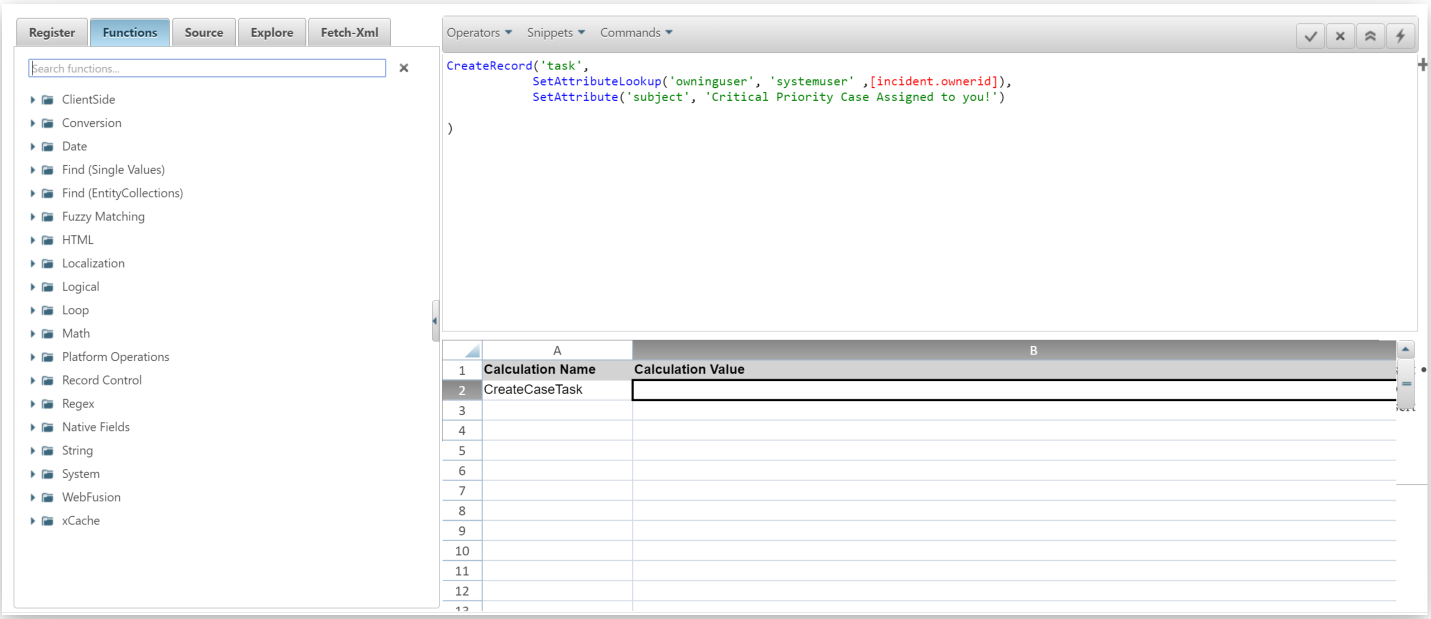


	A	B	C	D	E
1	Action Name	Action Value			
2	CreateCaseTask				
3					
4					
5					

Create the following formula in the formula editor.

Formula

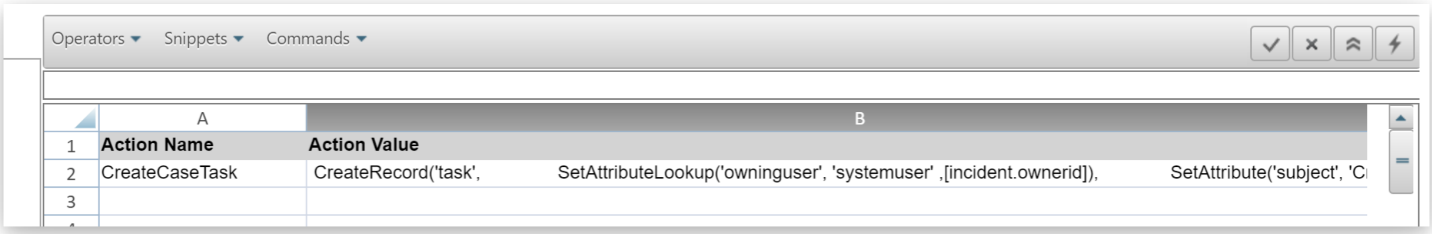
```
CreateRecord('task',
          SetAttributeLookup('owninguser', 'systemuser' ,[incident.ownerid]),
          SetAttribute('subject', 'Critical Priority Case Assigned to you!')

)
```



```
CreateRecord('task',
          SetAttributeLookup('owninguser', 'systemuser' ,[incident.ownerid]),
          SetAttribute('subject', 'Critical Priority Case Assigned to you!')

)
```

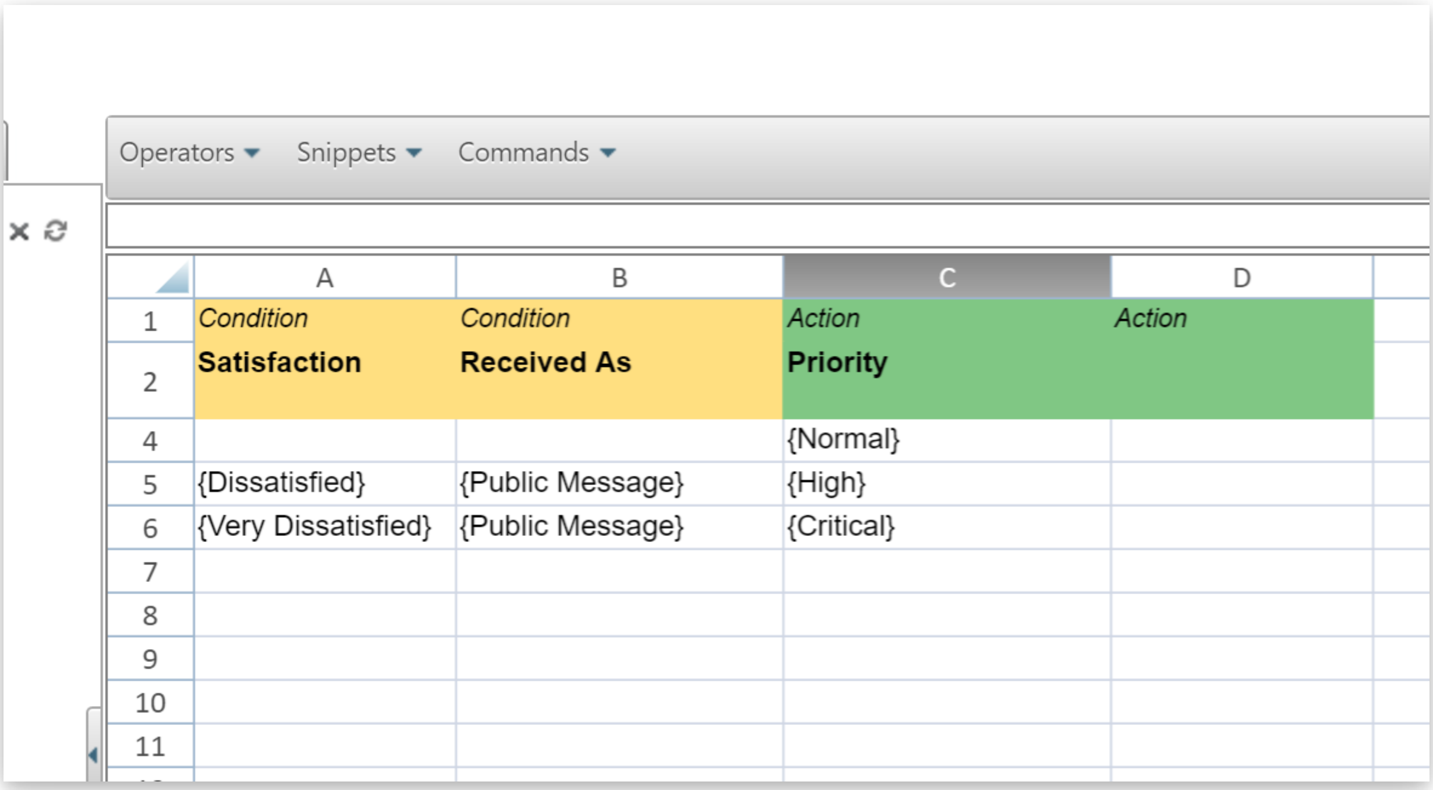
	A	B
1	Calculation Name	Calculation Value
2	CreateCaseTask	

Click on the **X** icon to close the formula editor - this inserts the formula into the **Action Value** cell:



	A	B
1	Action Name	Action Value
2	CreateCaseTask	CreateRecord('task', SetAttributeLookup('owninguser', 'systemuser' ,[incident.ownerid]), SetAttribute('subject', 'C
3		

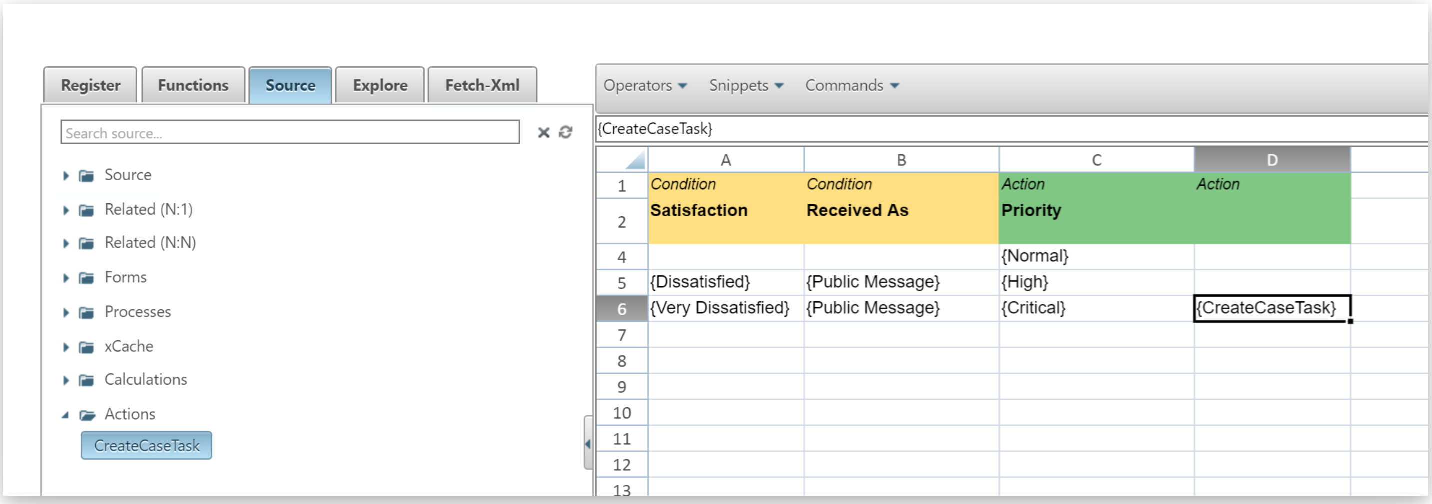
We have now finished defining the Action. Return to the Decision Table sheet and add a new **Action** column by *right-clicking* and selecting *Insert > Insert Action* from the contextual menu.
Note: *We don't name the column this time as we are not updating a specific field*



	A	B	C	D
1	*Condition*	*Condition*	*Action*	*Action*
2	**Satisfaction**	**Received As**	**Priority**	
4			{Normal}	
5	{Dissatisfied}	{Public Message}	{High}	
6	{Very Dissatisfied}	{Public Message}	{Critical}	
7				
8				
9				
10				
11				

Click on the *Source* tab and *expand* the **Actions** node. The **CreateCaseTask** Action is now listed here.

Select *cell D6* next to **Priority** and then click on *CreateCaseTask* from the **Source** tree to add it to the Decision Table editor.



{CreateCaseTask}

	A	B	C	D
1	*Condition*	*Condition*	*Action*	*Action*
2	**Satisfaction**	**Received As**	**Priority**	
4			{Normal}	
5	{Dissatisfied}	{Public Message}	{High}	
6	{Very Dissatisfied}	{Public Message}	{Critical}	{CreateCaseTask}
7				
8				
9				
10				
11				
12				

The Decision Table will now create a *Task* for the **Owner** of the *Case* when it is setting the **Priority** of the *Case* to *Critical*.

You can set as many Complex Actions and Simple Actions against a Condition as you wish, simply add more Action columns.

Variables

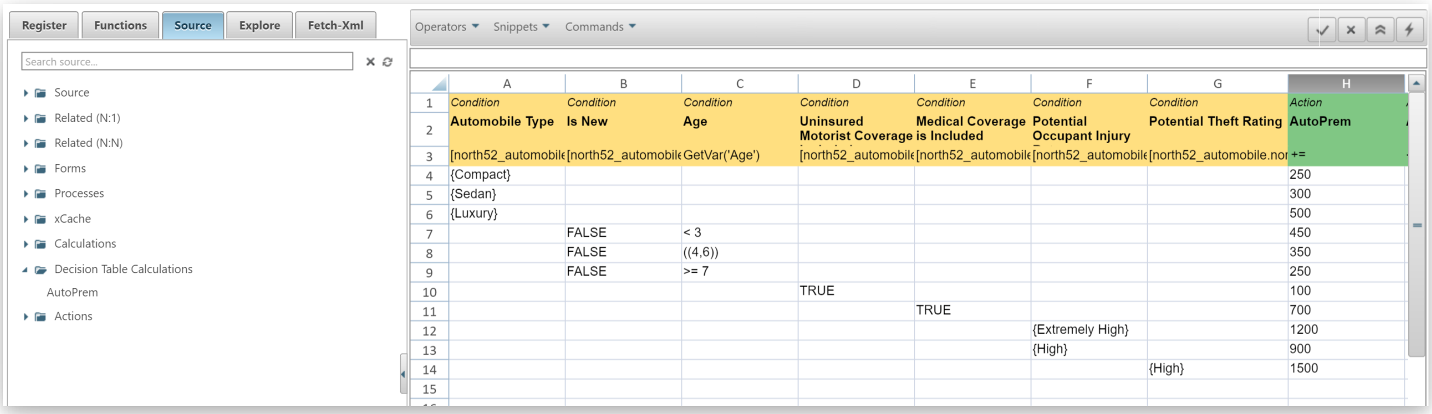
Sometimes in Decision Tables you will need to Sum up a value based on the **Condtions**.
This sort of **Sum up** functionality can only be used in Decision Tables with the **Exit this Decision Table on First Match** feature turned off.

To perform a **Sum up Calculation** like this:

- Name your **Action**
- *Right-click* on the Decision Table and select *Toggle Advanced Mode* from *Operations*
- **Row 3**, which is hidden by default, will appear on the Decision Table
- Under your **Action Name** enter +=

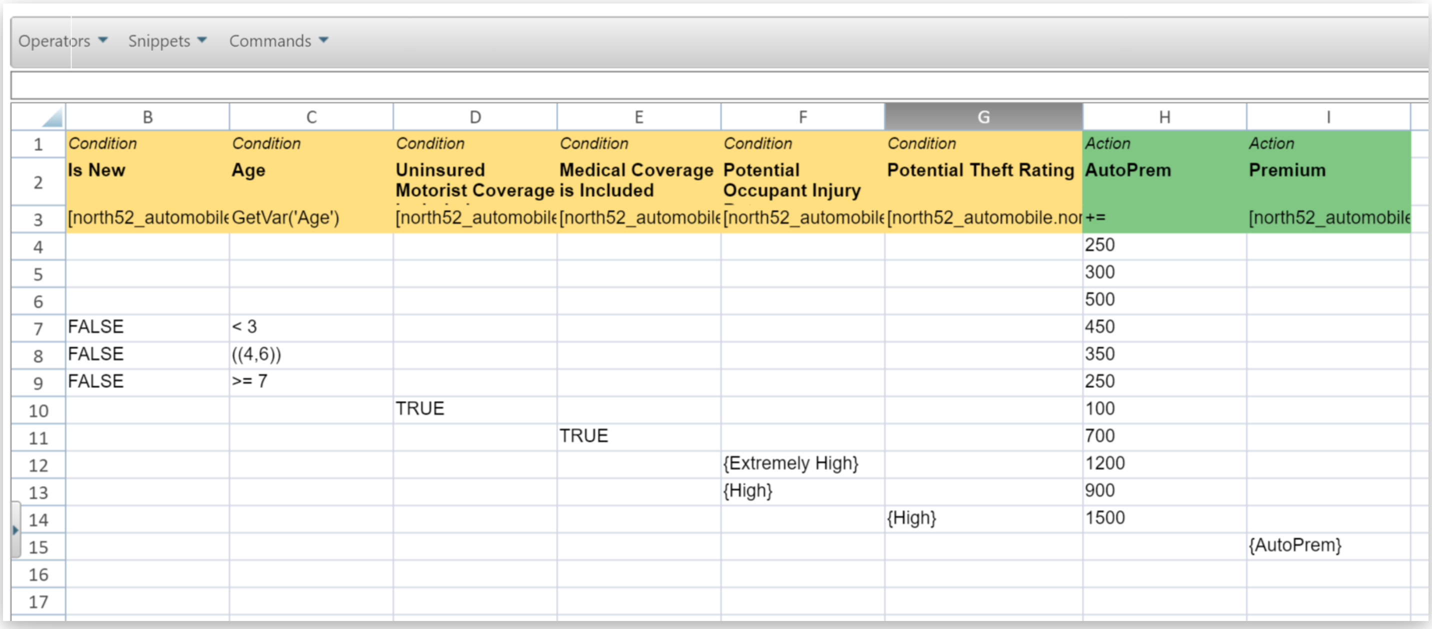
- This will tell the Decision Table that the values in this row should be used in a **Sum** calculation within that Decision Table
- The name of the Action column will then appear under the **Decision Table Calculations** node on the **Source** tabof the configuration pane

In the below example we use this functionality to calculate an **Auto Insurance Premium** based on a variety of **Conditions**



	A	B	C	D	E	F	G	H
1	*Condition*	*Condition*	*Condition*	*Condition*	*Condition*	*Condition*	*Condition*	*Action*
2	**Automobile Type**	**Is New**	**Age**	**Uninsured Motorist Coverage**	**Medical Coverage is Included**	**Potential Occupant Injury**	**Potential Theft Rating**	**AutoPrem**
3	[north52_automobile	[north52_automobile	GetVar('Age')	[north52_automobile	[north52_automobile	[north52_automobile	[north52_automobile.nor	+=
4	{Compact}							250
5	{Sedan}							300
6	{Luxury}							500
7		FALSE	< 3					450
8		FALSE	((4,6))					350
9		FALSE	>= 7					250
10				TRUE				100
11					TRUE			700
12						{Extremely High}		1200
13						{High}		900
14							{High}	1500
15								

We can then use that value by adding in an **Action** to update the **Premium** and referencing the value by clicking on *AutoPrem* from the **Decision Table Calculations** node.



	B	C	D	E	F	G	H	I
1	*Condition*	*Condition*	*Condition*	*Condition*	*Condition*	*Condition*	*Action*	*Action*
2	**Is New**	**Age**	**Uninsured Motorist Coverage**	**Medical Coverage is Included**	**Potential Occupant Injury**	**Potential Theft Rating**	**AutoPrem**	**Premium**
3	[north52_automobile	GetVar('Age')	[north52_automobile	[north52_automobile	[north52_automobile	[north52_automobile.nor	+=	[north52_automobile
4							250	
5							300	
6							500	
7	FALSE	< 3					450	
8	FALSE	((4,6))					350	
9	FALSE	>= 7					250	
10			TRUE				100	
11				TRUE			700	
12					{Extremely High}		1200	
13					{High}		900	
14						{High}	1500	
15								{AutoPrem}
16								
17								

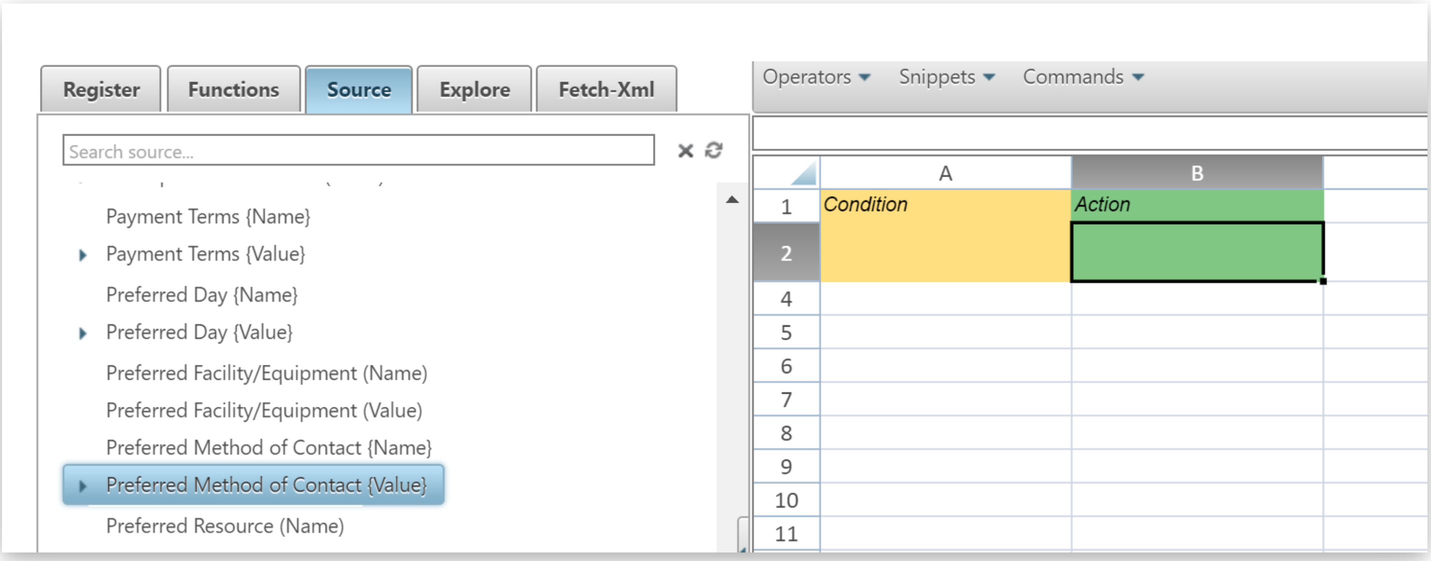
DT - How to - 05 - Use Name or Value fields in Decision Tables

[TOC]

Overview

This article will explain when you would use certain Name and Value fields within a **Decision Table.** There are two variations on this Name and Value combination.

- **Option Sets:** {Name} or {Value}
- **Lookups:** (Name) or (Value)



Payment Terms {Name}
Payment Terms {Value}
Preferred Day {Name}
Preferred Day {Value}
Preferred Facility/Equipment (Name)
Preferred Facility/Equipment (Value)
Preferred Method of Contact {Name}
Preferred Method of Contact {Value}
Preferred Resource (Name)

When used in Conditions

Option Sets

In an **Option Set** the value is the **Integer** value that is actually stored behind the scenes e.g. 100000000 while the name is the string value that appears on the **Option Set** for the end-user. The name shown to the user can be different depending on the language of the user. A user with the language setting of 'English' will see the name in English but a user with a Spanish language setting will see the Spanish name. So we need to be very careful when using the name field when designing business rules as normally we would want the business rule to execute the same independent of language.

When you are checking the value of an **Option Set** in a **Condition**, you would always use the **value** field as this is the actual value stored in the field. This is also the default that North52 BPA uses when you select option set items from the left-hand **Source** menu. It also means that the logic will work the same way independent of language.

*e.g. Account entity **'Industry'** field*

'industrycode'	Stores the integer	**{Value}**
'industrycodename'	Stores the string	**{Name}**

Lookups

With a Lookup field the ***value*** is the Guid of the record being looked up while the name is the ***name*** of that record.

When you are checking against the value of a Guid you would use the **value** field, however you could use the **name** field if you want to do a string comparison. The string comparison is useful because one of the issues with CRM is that Guids change between CRM systems (e.g. Dev/Test/Production) for lookups so using the name keeps everything consistent.

*e.g. Account entity **'Primary Contact ID'** field*

'primarycontactid'	Stores the guid	**(Value)**
'primarycontactidname'	Stores the string	**(Name)**

When used in Actions

With **Actions**, the **value** field should always be used when bringing a field into row 2 of a Decision Table.

This would apply to any **Action** including setting a value into the field or applying a **ClientSide** function like hiding, disabling or showing the field.

Rule of Thumb

Value should be used in almost all cases unless you are attempting a string compare (e.g. checking if the name of a **Contact** lookup = 'Patrick McInerney') in a **Condition.**

Performance

The **Name** and **Value** fields when used correctly can really speed up performance on a CRM system as they can save you performing additional API queries to the CRM system for metadata in the case of option sets and entity queries in the case of lookups.

DT - How to - 06 - Use the Decision Table Context (Popup) Menu

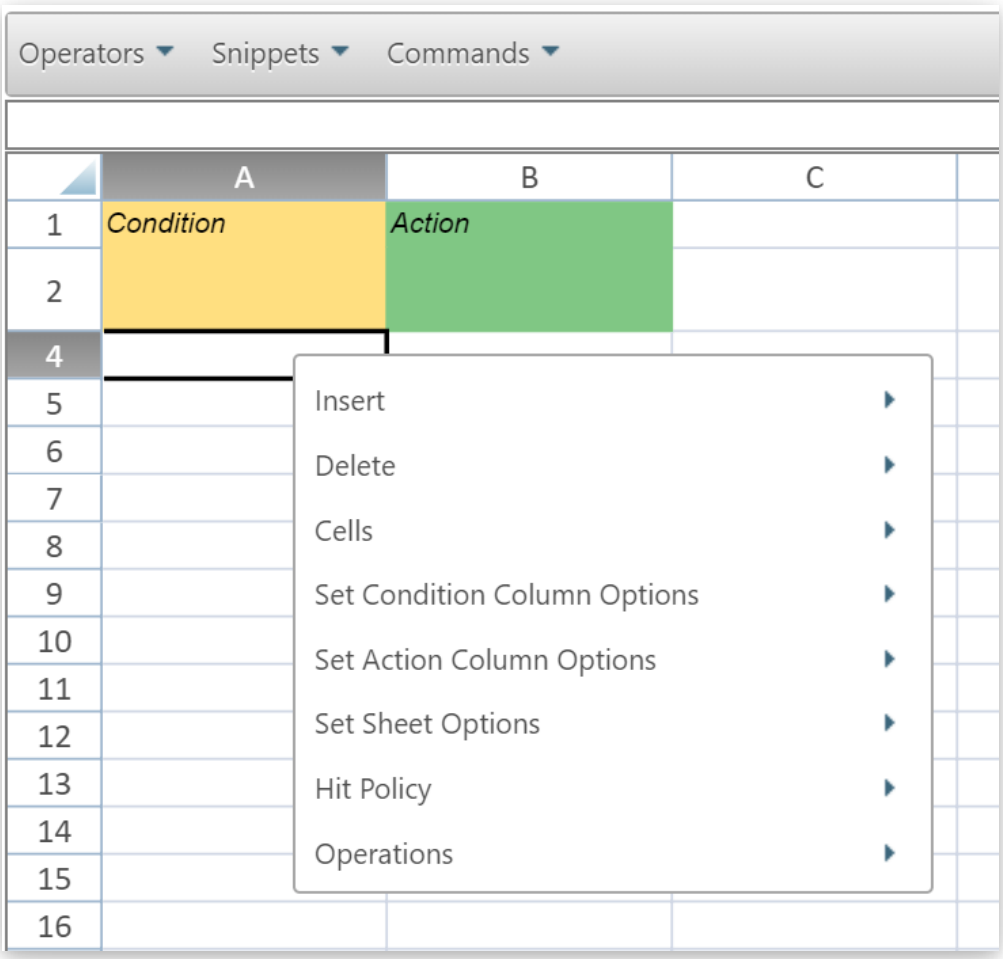
[TOC]

Overview

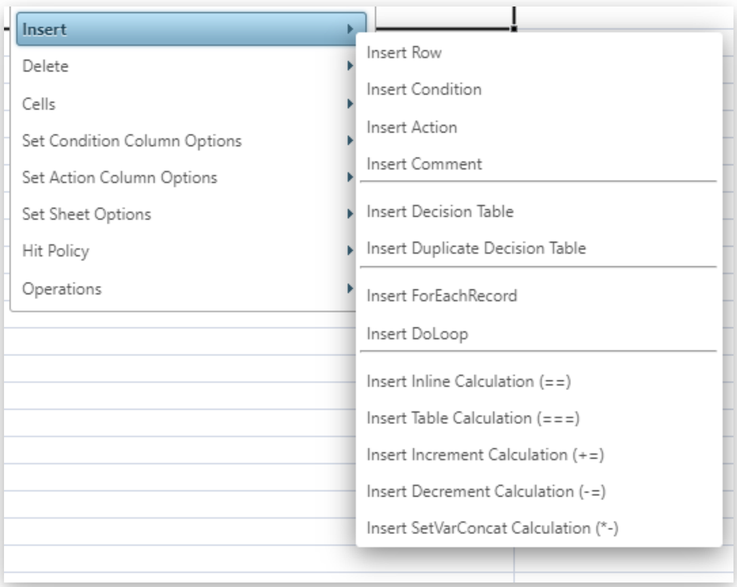
This article will detail all the options available in **Context** or Pop-up menu for a **Decision Table.**

How to Access the Menu

If you right click within a **Decision Table** then the menu will appear:



Insert


Insert
Delete
Cells
Set Condition Column Options
Set Action Column Options
Set Sheet Options
Hit Policy
Operations
Insert Row
Insert Condition
Insert Action
Insert Comment
Insert Decision Table
Insert Duplicate Decision Table
Insert ForEachRecord
Insert DoLoop
Insert Inline Calculation (==)
Insert Table Calculation (===)
Insert Increment Calculation (+=)
Insert Decrement Calculation (-=)
Insert SetVarConcat Calculation (*-)

Insert Row

A decision table comes with 35 rows by default, if you need any additional rows you can add them using this command. Just highlight as many additional rows as you want and then click on the **Insert Row**. If 10 rows are highlighted then 10 rows will be added.

Insert Condition

Adds an additional **Conditions** column.

Insert Action

Adds an additional **Action** column.

Insert Decision Table

Add an additional Decision Tables sheet to your Decision Table - making it a **Multi-Sheet Decision Table.**

Insert Duplicate Decision Table

Creates a copy of the current Decision Table sheet.

Insert ForEachRecord

Adds a **ForEachRecord** column into the Decision Table.

Insert DoLoop

Adds a **DoLoop** column into the Decision Table.

Insert Inline Calculation (==)

Adds a calculation column, whose final value is output as a variable that can be used in subsequent parts of a Decision Table formula.

Insert Table Calculation (===)

Used only when there is a **ForEachRecord** or **DoLoop** column in the Decision Table. Adds a calculation column that is calculated outside of the loop and whose final value is output as a variable that can be used in subsequent parts of a Decision Table formula.

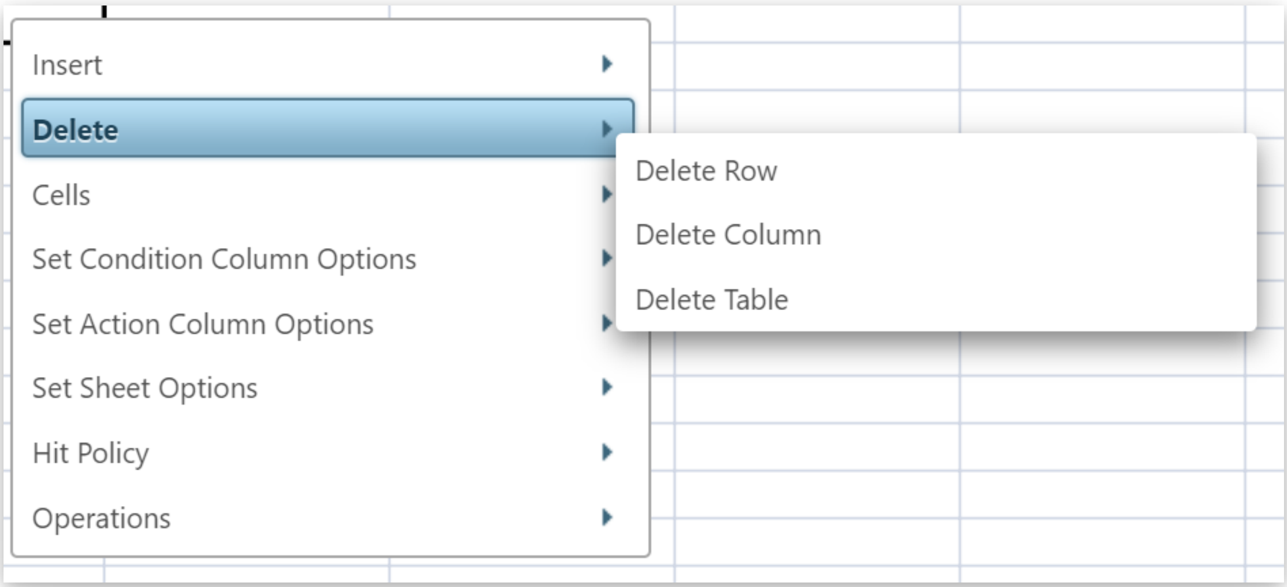
Insert Increment Calculation (+=)

Adds a calculation column, which will increment in value for each row that matches the conditions and whose final value is output as a variable that can be used in subsequent parts of a Decision Table formula. Will only work when the sheet **Hit Policy** has the **Exit this Decision Table on First Match** unchecked.

Insert Decrement Calculation (-=)

Adds a calculation column, which will decrement in value for each row that matches the conditions and whose final value is output as a variable that can be used in subsequent parts of a Decision Table formula. Will only work when the sheet **Hit Policy** has the **Exit this Decision Table on First Match** unchecked.

Delete



Delete Row

Deletes the selected row of data.

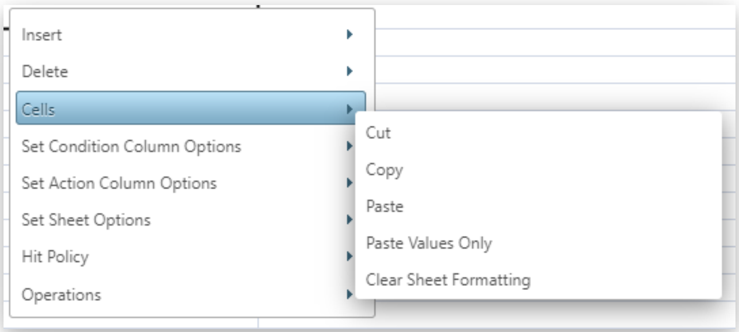
Delete Column

Deletes the selected column.

Delete Sheet

Deletes the **Decision Sheet** you are working on.

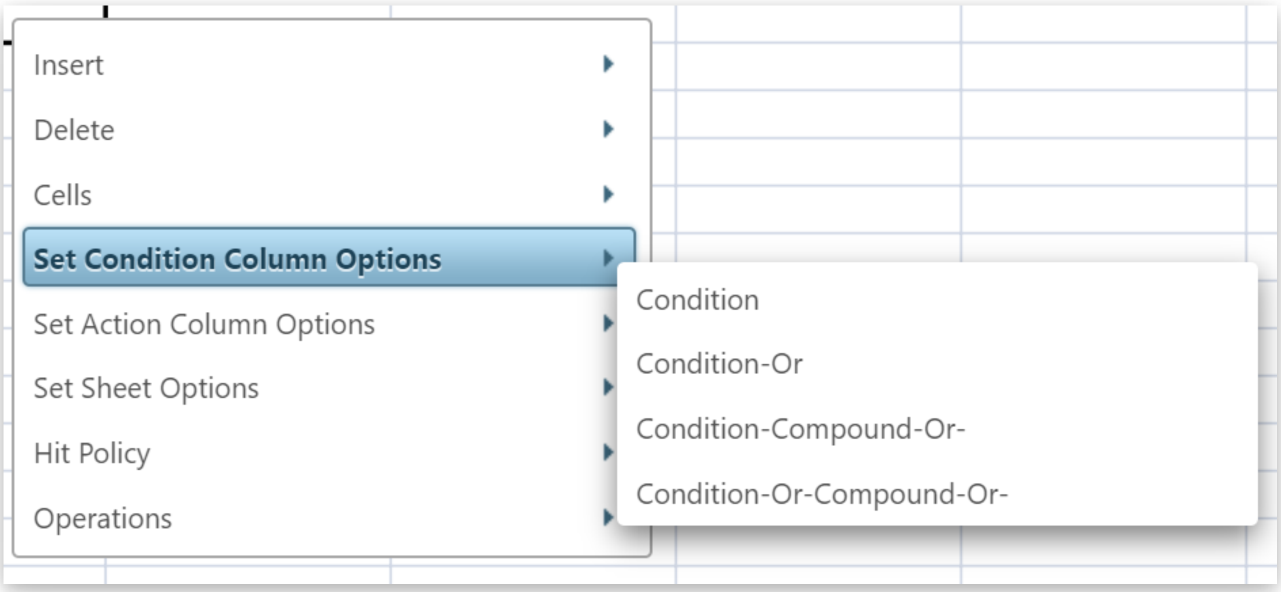
Cells



Cut, Copy & Paste

These work exactly like the standard Windows Excel style Cut, Copy and Paste functionality.

Set Condition Column Options



Condition

Reverts back or change columns into a standalone condition.

Condition A and Condition B

Condition-Or

Combines Conditions Columns together into an OR condition.

Condition A OR Condition B

Condition-Compound-Or

Encloses two or more columns together in brackets. These Columns are then separated by AND conditions.

(*Condition A and Condition B)*

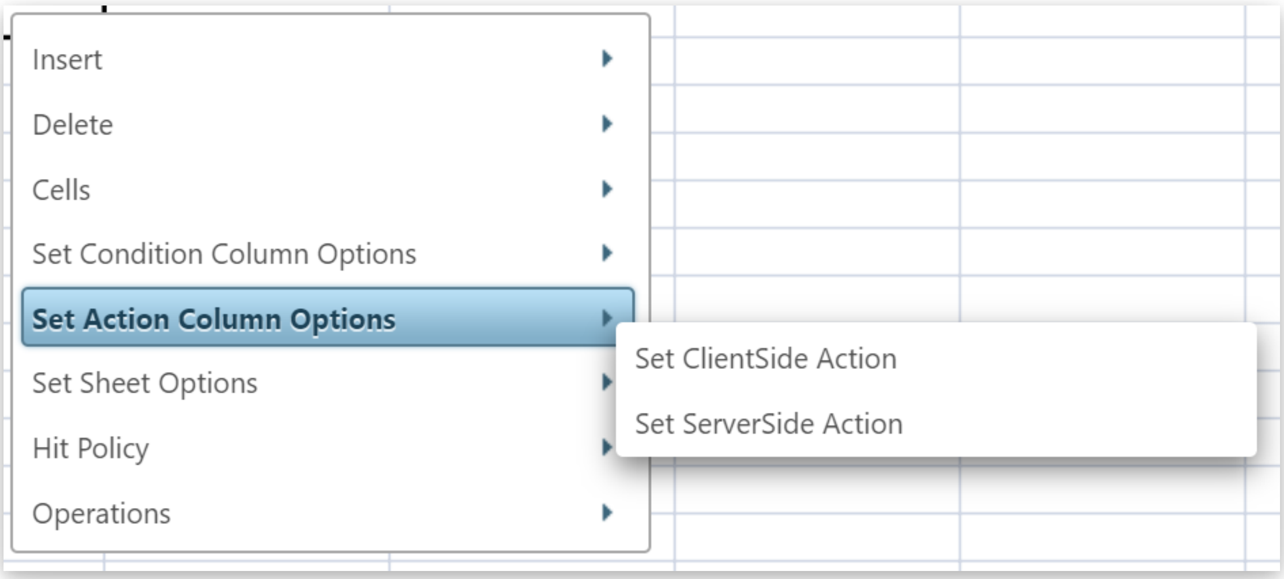
Condition-Or-Compound-Or

Enlcose two or more columns together in brackets. These Columns are then separated by OR conditions.

(Condition A OR Condition B)

*Please review this article for more information on how to configure **OR Conditions**.*

Set Action Column Options



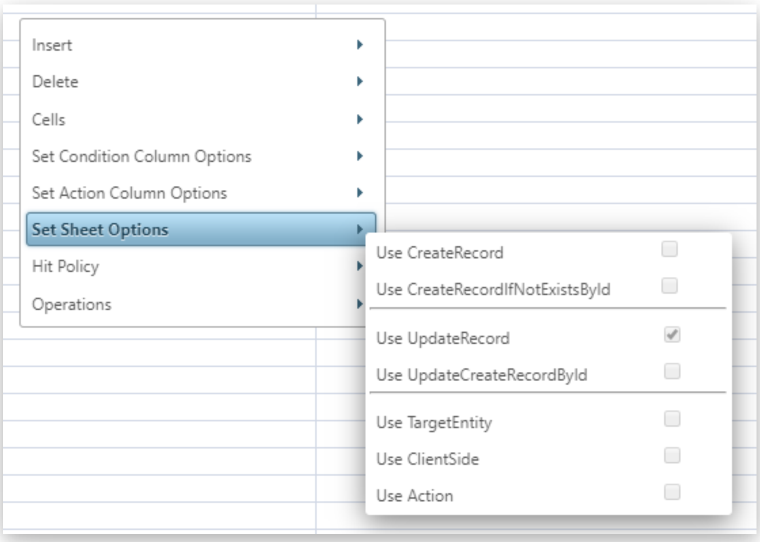
Set ClientSide Action

Allows you to carry out ClientSide actions, only relevant for when Mode is set to Clientside.

Set ServerSide Acton

Allows you to carry out ServerSide actions, only relevant for when Mode is set to Clientside.

Set Sheet Options



Use CreateRecord

Change the sheet behavior to allow for Creation of records.

Use UpdateRecord

Change the sheet behavior to allow for Updating of records (this is the default setting).

Use TargetEntity

Change the sheet behavior to allow for updating of data **before** it is committed to the database.

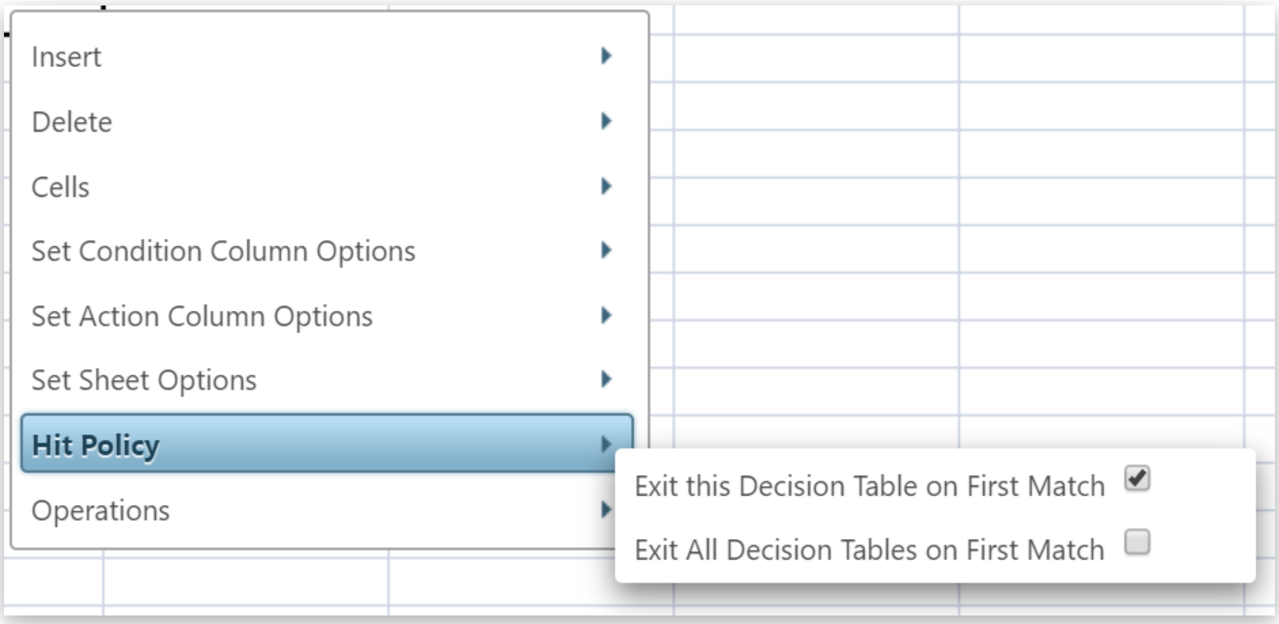
Use ClientSide

Change the sheet behavior to use the **SetClientSideField**() function rather than the default **UpdateRecord**() function. Use ClientSide will be selected by default for formulas with a Type of **ClientSide Perform Action**.

Use Action

Change the sheet behavior to use the **SetActionOutput**() function for any field references.

Hit Policy



Exit this Decision Table on First Match ✓
Exit All Decision Tables on First Match ☐

Exit this Decision Table on First Match

This command tells the Decision Table to stop checking conditions once it finds it first success.

By default this is turned on - however you may not always want this feature turned on.

For example if you have several independent updates executing at the same time. Instead of creating several different formulas you could instead create a single decision table and put all the logic into it.

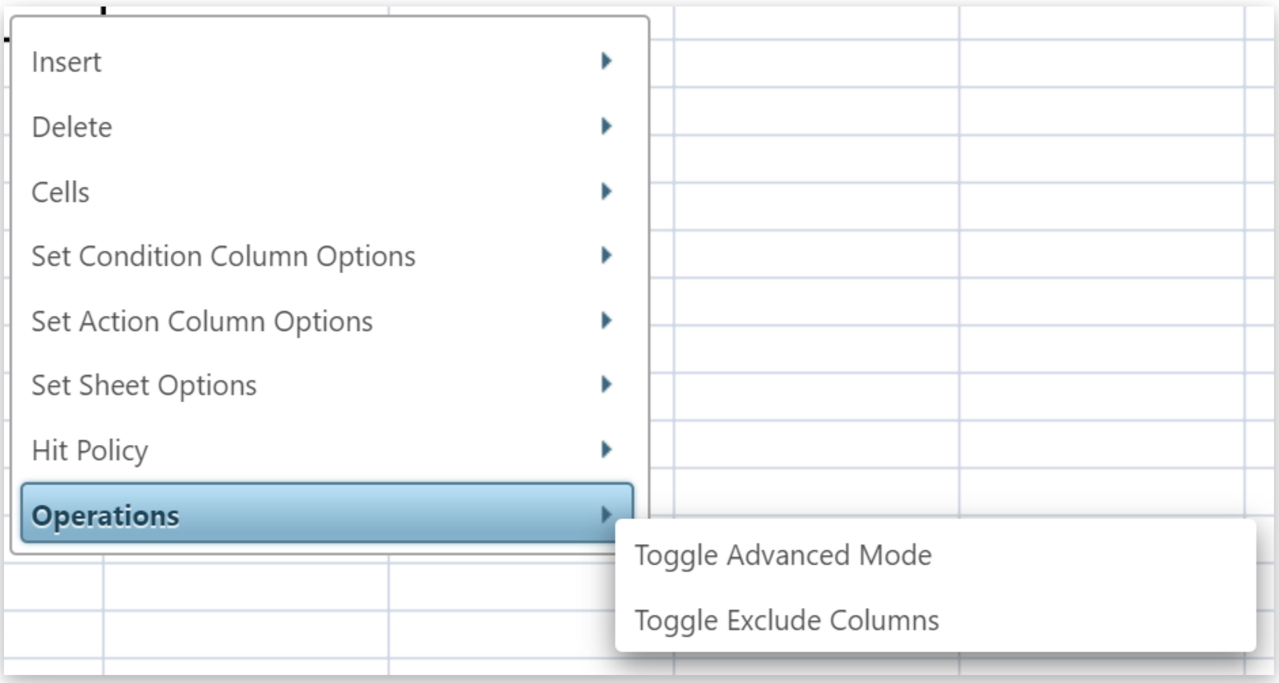
As long as the *Exit on First Match* is turned off the Decision Table will execute the actions for all conditions that are fulfilled.

Exit All Decision Tables on First Match

This command is used in **Multi-Sheet Decision Tables**. If it is turned **OFF** then when the **Decision Table** has finished processing the current **Decision Sheet**, it will move to the next **Decision Sheet** in the **Decision Table**.

However if it is turned **ON** - then the **Formula** will fully exit on this **Decision Sheet** and not execute any additional **Decision Sheets** afterwards.

Operations



Toggle Advanced Mode

This command shows/hides values stored in **row 3**. You will see the full names of any fields being used in the **Decision Table** and be able to set default values for them. Please click here for more information on setting default values. It also shows/hides the **Global Calculations** and **Global Actions** sheet tabs.

Toggle Exclude Columns

If you would like to exclude a column from executing you can click into the column and select this option. Select it again to include the column again. This option is useful when you temporarily need to exclude logic for testing purposes.

DT - How to - 07 - Advanced OR Conditions in Decision Tables

[TOC]

Overview

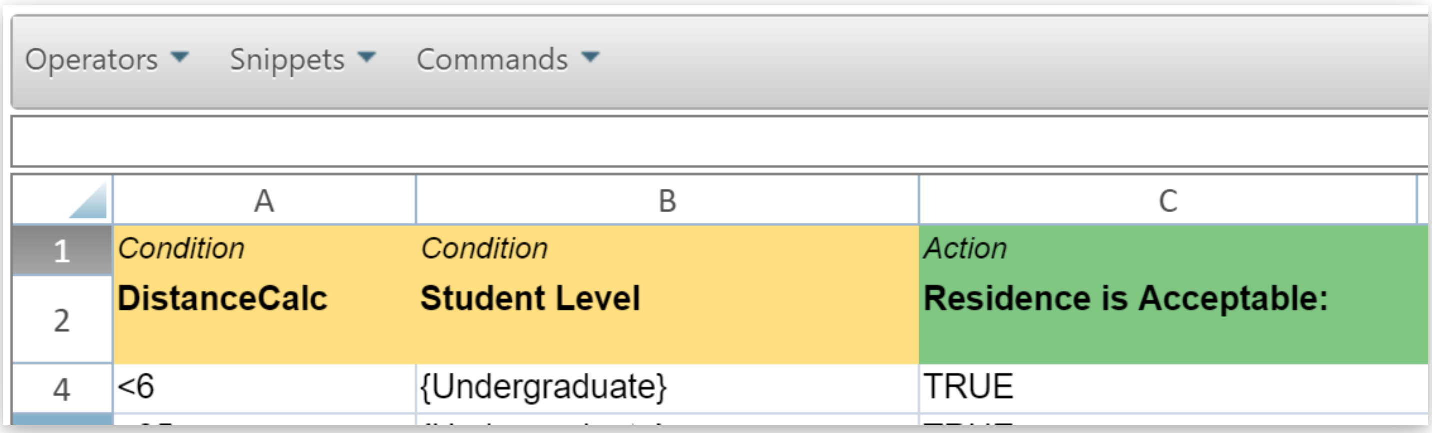
This article will detail how to set up **Complex OR Conditions** in North52 **Decision Tables**.

OR Conditions allow you to keep your Business Logic on the same line instead of creating multiple rows producing the same result.

Basic Condition

When you add a **condition** to your **Decision Table** it is by default an **AND** type **Condition**.
This means that it must be true for your row to execute its **action**.

In the below example: the **Student Level** must be *Undergraduate* **AND** the result of **DistanceCalc** must be *less than 6*for the row to be evaluated as true and execute its **action.**

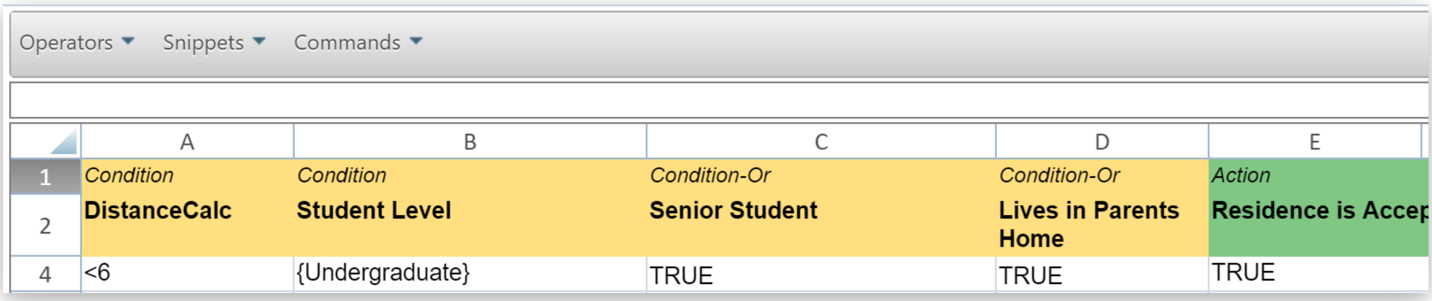


	A	B	C
1	*Condition*	*Condition*	*Action*
2	**DistanceCalc**	**Student Level**	**Residence is Acceptable:**
4	<6	{Undergraduate}	TRUE

Basic OR Condition

OR conditions must be set side by side in a **Decision Table.** To use OR conditions you highlight the columns you want to set together as **OR conditions** and right click on the **Decision Table**. This will bring up the Context Menu. From this you can click on *Toggle Column OR* to set the columns as **OR Conditions**. You can see step by step instructions here.

Below is an example of a basic **OR** Condition.



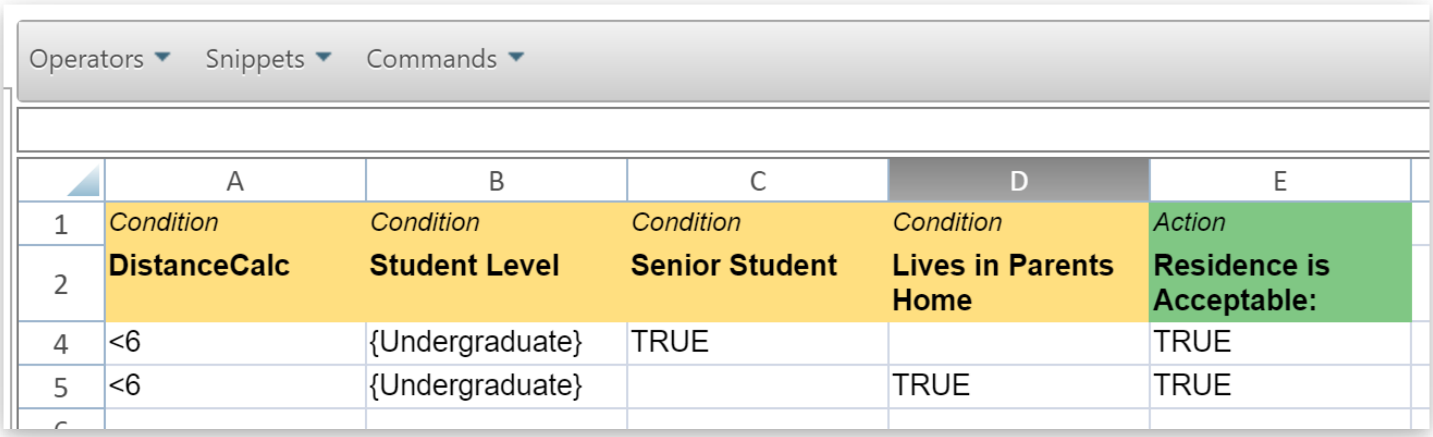
	A	B	C	D	E
1	*Condition*	*Condition*	*Condition-Or*	*Condition-Or*	*Action*
2	**DistanceCalc**	**Student Level**	**Senior Student**	**Lives in Parents Home**	**Residence is Accep**
4	<6	{Undergraduate}	TRUE	TRUE	TRUE

Here the **Student Level** must be *Undergraduate* **AND** the result of **DistanceCalc** must be *less than 6* and they must be a **Senior Student**

OR

the **Student Level** must be *Undergraduate* **AND** the result of **DistanceCalc** must be *less than 6* and they must be **Living in their Parents Home.**

This could also be represented in the Decision Table as the below:



	A	B	C	D	E
1	*Condition*	*Condition*	*Condition*	*Condition*	*Action*
2	**DistanceCalc**	**Student Level**	**Senior Student**	**Lives in Parents Home**	**Residence is Acceptable:**
4	<6	{Undergraduate}	TRUE		TRUE
5	<6	{Undergraduate}		TRUE	TRUE

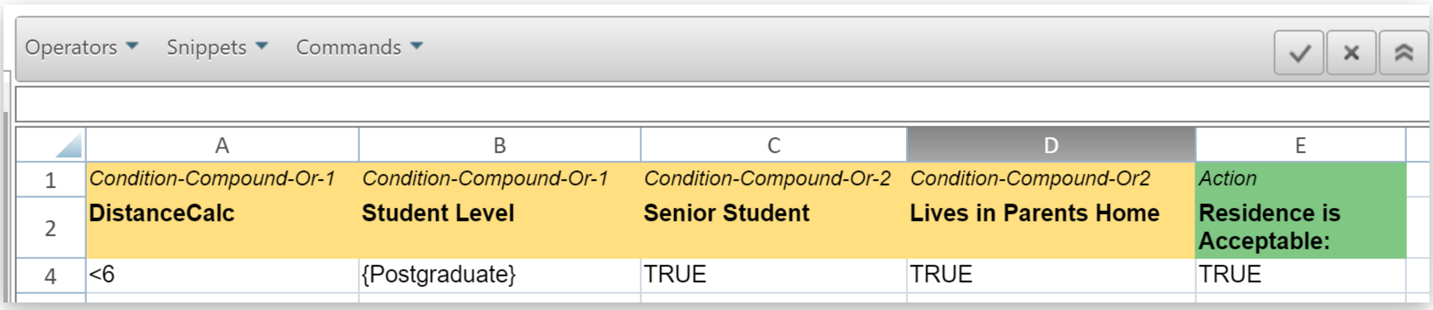
Compound OR Conditions

Compound OR conditions are used for more complex logic. For example where the OR Condition contains internal AND conditions.

To change columns to Compound OR's, highlight the columns and right click to bring up the context menu. Click **Toggle Column OR** and the columns will change to *Condition-OR.* Bring up the Context Menu again and click **Toggle Column OR** again and the columns will become *Condition-Compound-OR* columns.

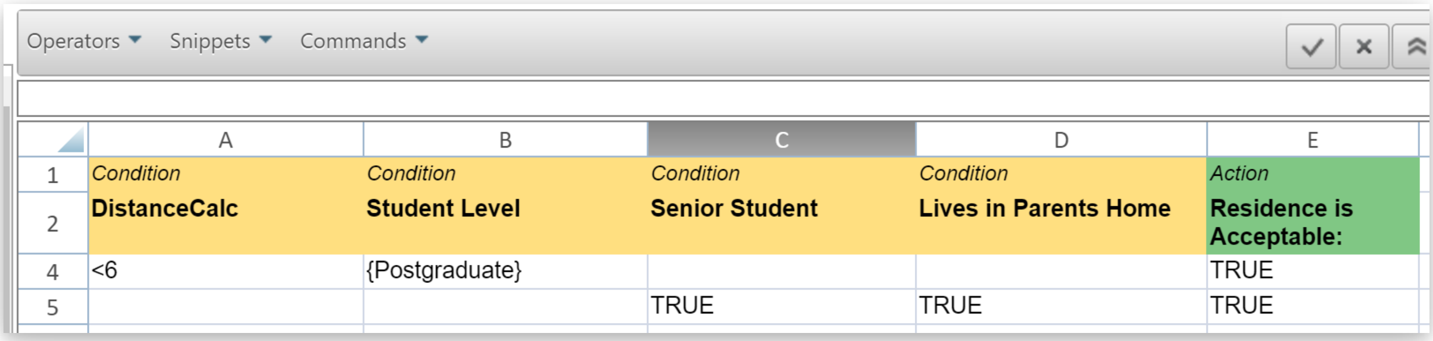
To group **Condition-Compound-OR** columns together; edit the **Row 1** cells and add numbers to them manually. In the below example **Columns A** and **B** are part of the same Condition-Compound-OR statement because their **Row 1** cell's are *Condition-Compound-OR-1.*

Below is an example of a **Condition-Compound-OR**. Here you must be a **Senior Student AND** living at home, **OR** a postgraduate **AND** living within 6 miles.



	A	B	C	D	E
1	*Condition-Compound-Or-1*	*Condition-Compound-Or-1*	*Condition-Compound-Or-2*	*Condition-Compound-Or2*	*Action*
2	**DistanceCalc**	**Student Level**	**Senior Student**	**Lives in Parents Home**	**Residence is Acceptable:**
4	<6	{Postgraduate}	TRUE	TRUE	TRUE

This could also be written in a Decision Table as:



	A	B	C	D	E
1	*Condition*	*Condition*	*Condition*	*Condition*	*Action*
2	**DistanceCalc**	**Student Level**	**Senior Student**	**Lives in Parents Home**	**Residence is Acceptable:**
4	<6	{Postgraduate}			TRUE
5			TRUE	TRUE	TRUE

DT - How to - 08 - Use ClientSide Decision Tables - General Information

[TOC]

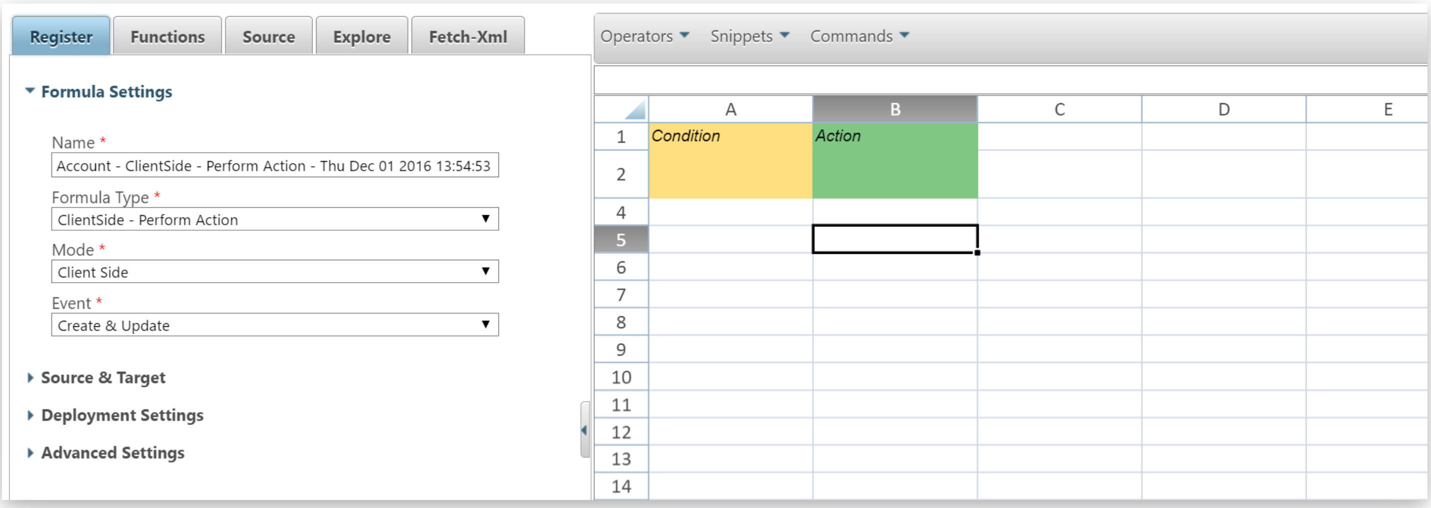
Overview

As of release version 510, North52 Decision Tables support multiple **ClientSide** actions directly from the **Action** columns on your **Decision Table.**

With this functionality you can set **ClientSide** functions against specific fields, make changes to the form and even execute **ServerSide** functionality concurrently.

Formula Type

If you want to use a **ClientSide Decision Table** then you must set the formula type to **ClientSide - Perform Action**.
*Note : Its useful to check your **Decision Sheet** is set to by right clicking and selecting **Set Sheet Options***



Formula Settings

Name *
Account - ClientSide - Perform Action - Thu Dec 01 2016 13:54:53

Formula Type *
ClientSide - Perform Action

Mode *
Client Side

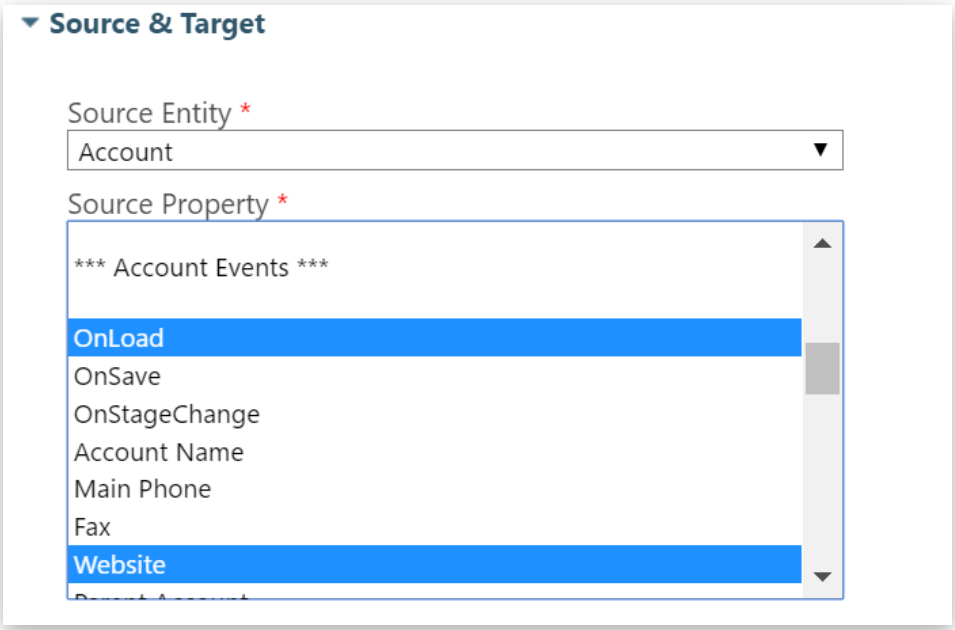
Event *
Create & Update

	A	B	C	D	E
1	Condition	Action			
2					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					

Note: With all **ClientSide** formulas you must click on **N52 Commands -> Publish Formula** after saving to publish the formula to the relevant entity forms.

Source Settings

As with North52 Classic client side Formulas you can register the Decision Table against the various forms on your *entity* executing as **OnLoad OnSave** etc. as well as **OnChange** of various fields. In the example below you can see the **Decision Table** triggers have been set for **OnLoad** of the main Account Form (*** Account Events ***) and will also fire when the **website** field is changed.


Source & Target

Source Entity *

Account

Source Property *

*** Account Events ***

OnLoad

OnSave

OnStageChange

Account Name

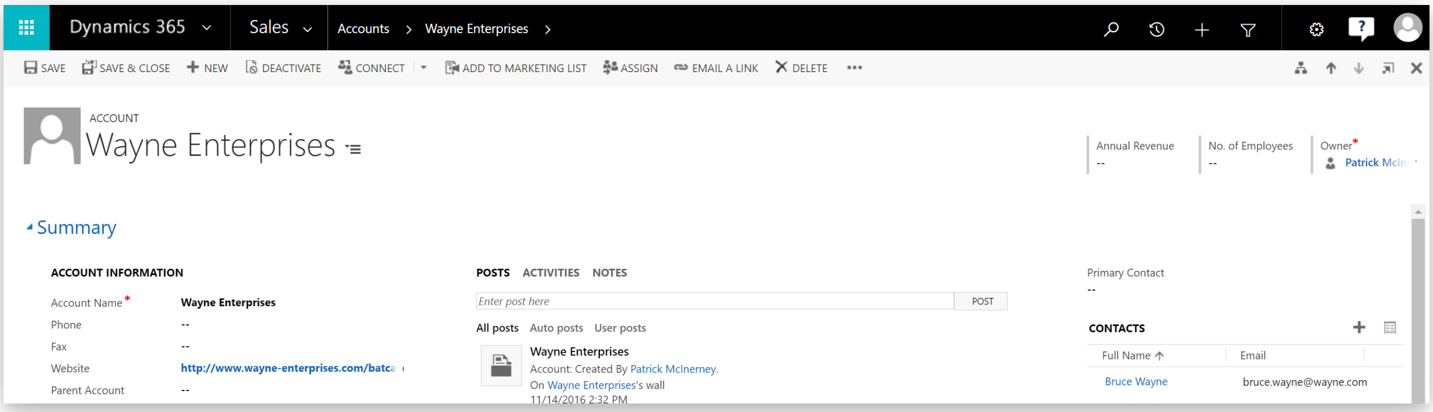
Main Phone

Fax

Website

ClientSide Example/Exercise - Fields

In this simple example we will hide the **Fax** field on the main *account* form if the **Phone** does not contain data:



ACCOUNT

Wayne Enterprises

Annual Revenue	No. of Employees	Owner*
--	--	Patrick McIn

Summary

ACCOUNT INFORMATION

Account Name*	Wayne Enterprises
Phone	--
Fax	--
Website	http://www.wayne-enterprises.com/batca
Parent Account	--

Enter post here

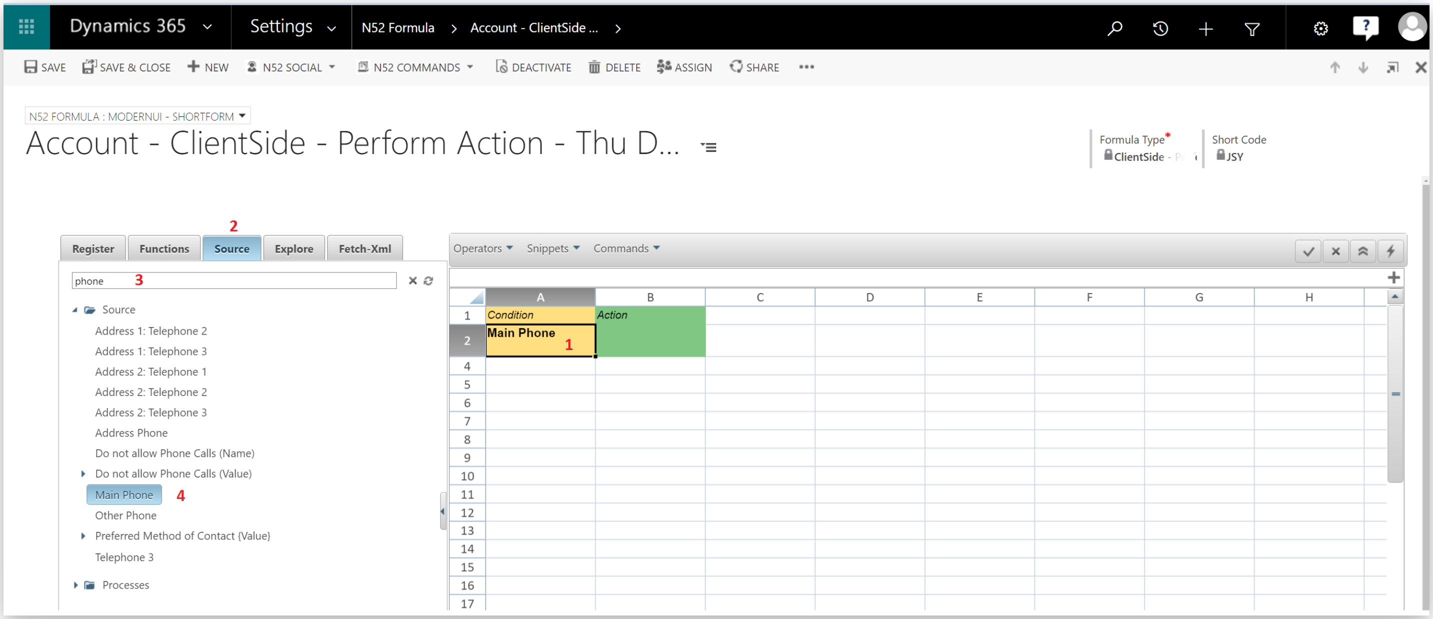
Wayne Enterprises
Account: Created By Patrick McInerney.
On Wayne Enterprises's wall
11/14/2016 2:32 PM

Primary Contact
--

CONTACTS

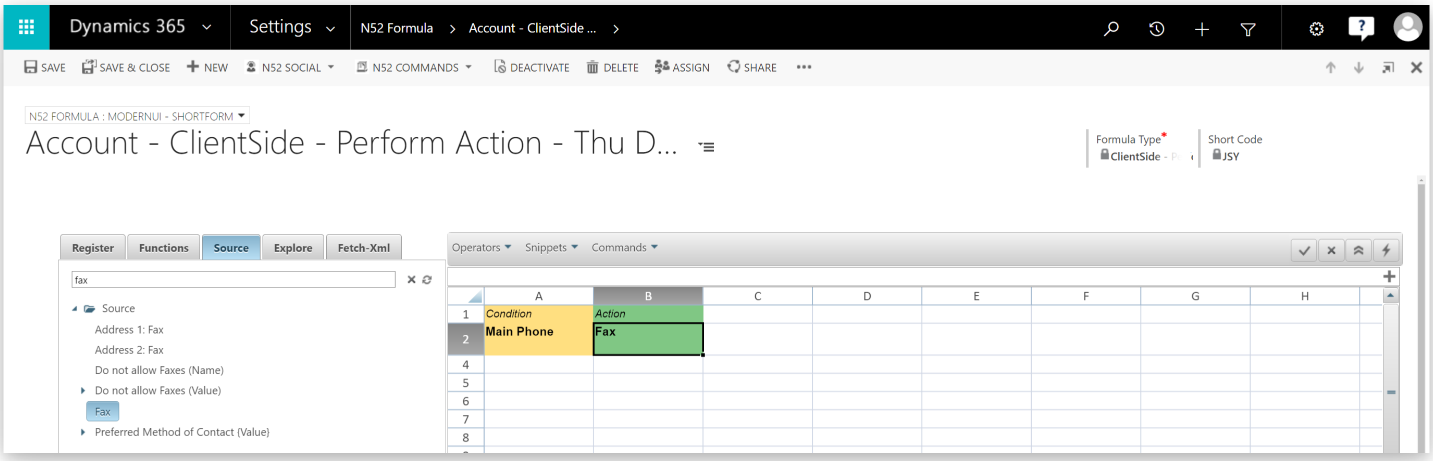
Full Name	Email
Bruce Wayne	bruce.wayne@wayne.com

Click on **Cell A2** (1) in the **Decision Table** and then click the **Source Tab** (2) in the configuration pane. In the search field type *phone* and press *enter* (3). Then click on **Main Phone** from under the Source node (4):


Dynamics 365
Settings
N52 Formula > Account - ClientSide ...
SAVE | SAVE & CLOSE | NEW | N52 SOCIAL | N52 COMMANDS | DEACTIVATE | DELETE | ASSIGN | SHARE
N52 FORMULA : MODERNUI - SHORTFORM
Account - ClientSide - Perform Action - Thu D...
Formula Type* ClientSide
Short Code JSY
Register | Functions | Source | Explore | Fetch-Xml
2
phone 3
Source
Address 1: Telephone 2
Address 1: Telephone 3
Address 2: Telephone 1
Address 2: Telephone 2
Address 2: Telephone 3
Address Phone
Do not allow Phone Calls (Name)
Do not allow Phone Calls (Value)
Main Phone 4
Other Phone
Preferred Method of Contact (Value)
Telephone 3
Processes
Operators | Snippets | Commands
1

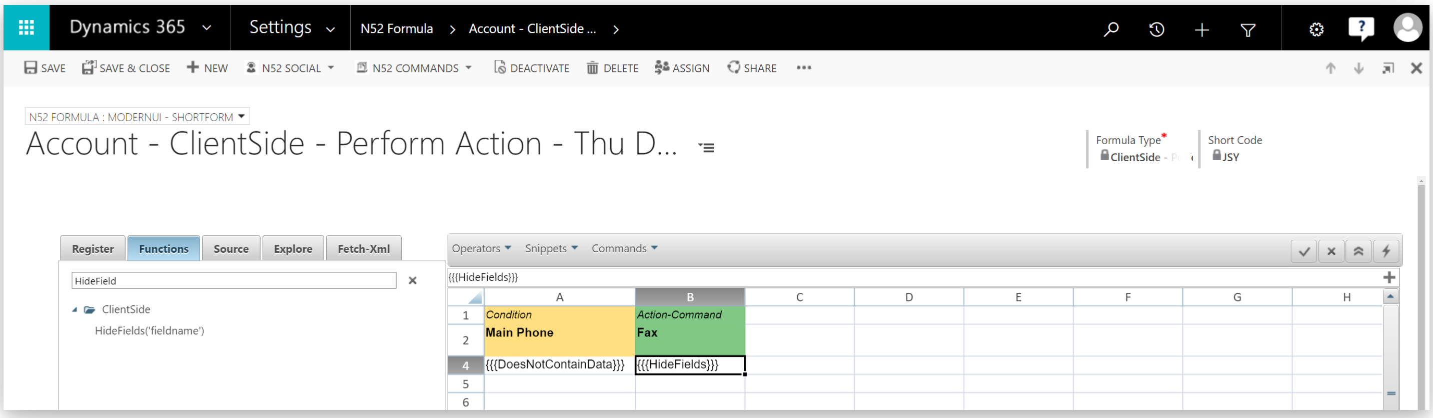
	A	B
1	*Condition*	*Action*
2	**Main Phone**	

Next click on **Cell B3** and add **Fax** here using the same steps as above. When you are finished your **Decision Table** should look like the following:


Dynamics 365
Settings
N52 Formula > Account - ClientSide ...
SAVE | SAVE & CLOSE | NEW | N52 SOCIAL | N52 COMMANDS | DEACTIVATE | DELETE | ASSIGN | SHARE
N52 FORMULA : MODERNUI - SHORTFORM
Account - ClientSide - Perform Action - Thu D...
Formula Type* ClientSide
Short Code JSY
Register | Functions | Source | Explore | Fetch-Xml
fax
Source
Address 1: Fax
Address 2: Fax
Do not allow Faxes (Name)
Do not allow Faxes (Value)
Fax
Preferred Method of Contact (Value)
Operators | Snippets | Commands

	A	B
1	*Condition*	*Action*
2	**Main Phone**	**Fax**

Click on **Cell A4** and select the **Functions Tab**. Enter '*doesnot*' into the search field and press enter. Hold down the **CTRL** key and click on the function name. **{{{DoesNotContainData}}}** will be inserted in **Cell A3**. Click on **Cell B4** and **control-click** on the HideField function, **{{{HideFields}}}** will be inserted in **Cell B4**.


Dynamics 365
Settings
N52 Formula > Account - ClientSide ...
SAVE | SAVE & CLOSE | NEW | N52 SOCIAL | N52 COMMANDS | DEACTIVATE | DELETE | ASSIGN | SHARE
N52 FORMULA : MODERNUI - SHORTFORM
Account - ClientSide - Perform Action - Thu D...
Formula Type* ClientSide
Short Code JSY
Register | Functions | Source | Explore | Fetch-Xml
HideField
ClientSide
HideFields('fieldname')
Operators | Snippets | Commands
{{{HideFields}}}

	A	B
1	*Condition*	*Action-Command*
2	**Main Phone**	**Fax**
4	{{{DoesNotContainData}}}	{{{HideFields}}}

When the field **Main Phone** does not contain data then the **Fax** field will be hidden.

ClientSide Example/Exercise - Form Actions

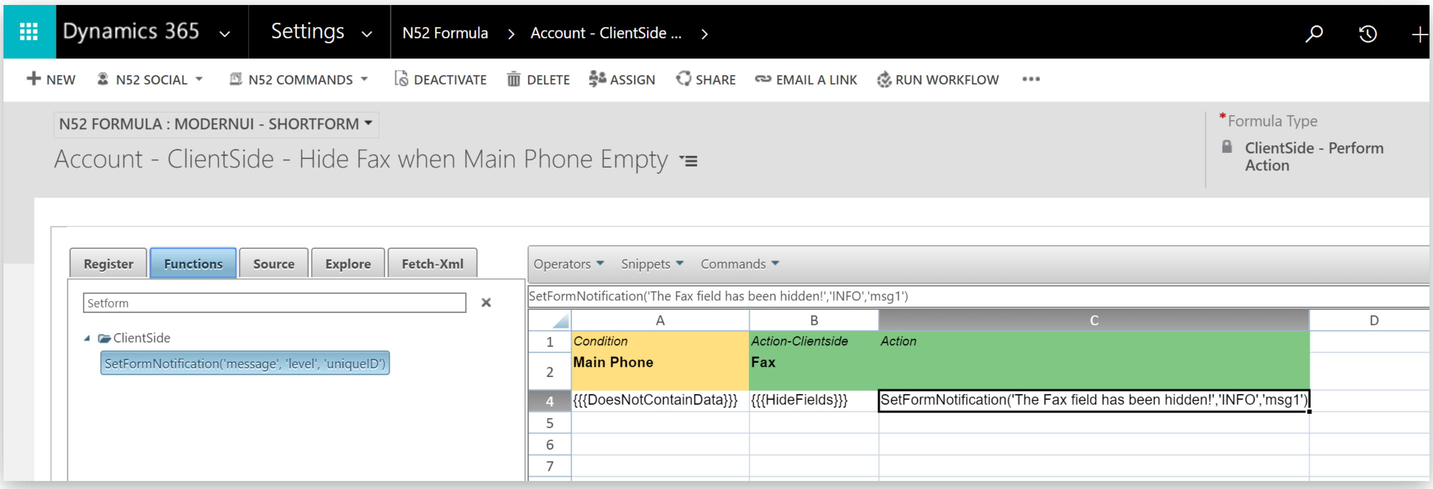
We are going to extend the above example slightly to add a form notification that the **Fax** field has been hidden.

Add another **Action** column to the **Decision Table**

Click in cell **C4**, select the **Functions** tab, enter *SetFor* in the search field and press enter:

- Shift-click the **SetFormNotification** function to open the function wizard
- Leave the **Friendly Name** field blank as we are going to insert the function directly into the cell for this example. If we wanted to show a more friendly name and move the function logic to the Global Actions sheet we would enter a name here and select Action from the Type drop down
- In the **Message** field type *The Fax field has been hidden!*
- Select *INFO* for the **Level**
- Enter *msg1* in the **UniqueID** field
- Click **Generate**

Click **Save**



Account - ClientSide - Hide Fax when Main Phone Empty

*Formula Type: ClientSide - Perform Action

Setform

ClientSide
SetFormNotification('message', 'level', 'uniqueID')

SetFormNotification('The Fax field has been hidden!','INFO','msg1')

	A	B	C	D
1	*Condition*	*Action-Clientside*	*Action*	
2	**Main Phone**	**Fax**		
4	{{{DoesNotContainData}}}	{{{HideFields}}}	SetFormNotification('The Fax field has been hidden!','INFO','msg1')	
5				
6				
7				

Test the formula and you should see that when the **Main Phone** field does not contain data a message will be displayed on the form and the **Fax** field will be hidden. You can add as many **ClientSide** functions together like this as you wish.

Server Side Functionality

It is possible to create **ServerSide** actions that will execute along side the **ClientSide** functionality. You can see an example of this here: xRM-Formula #164 Audit log of Users who access Invoices

DT - How to - 09 - Use ClientSide Decision Tables - Overview

[TOC]

Overview

Multi-Sheet Decision Tables allow you to take complex business requirements and break them down into constituent sets of business rules. It does this by allowing you to have multiple decision sheets with the formula. This can useful for both making your **Decision Table** logic **more readable** as well as separation of different **entitlement criteria.**

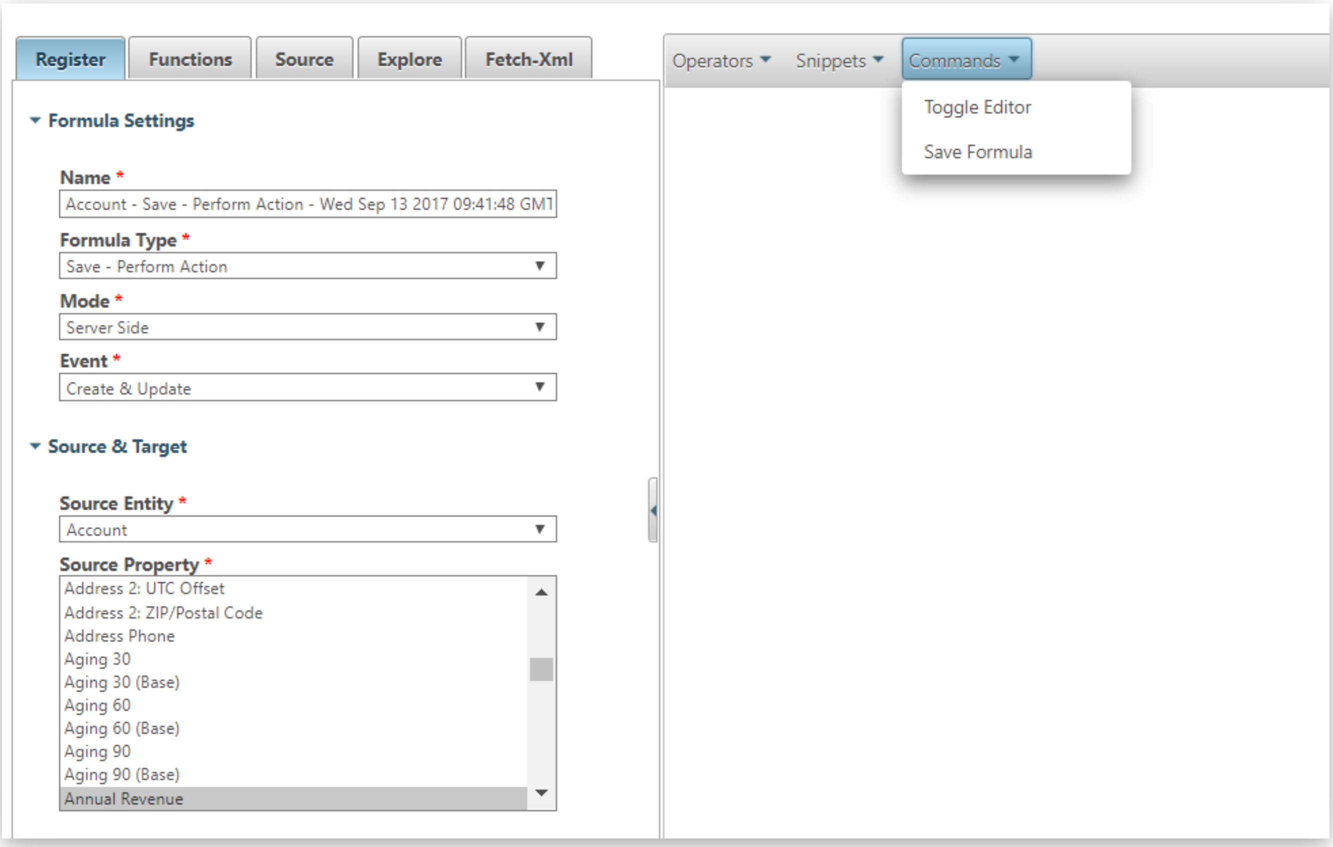
For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: We will not detail step-by-step instructions here on how to set up **Conditions** or **Actions,** please read the above articles if you need detailed configuration steps.

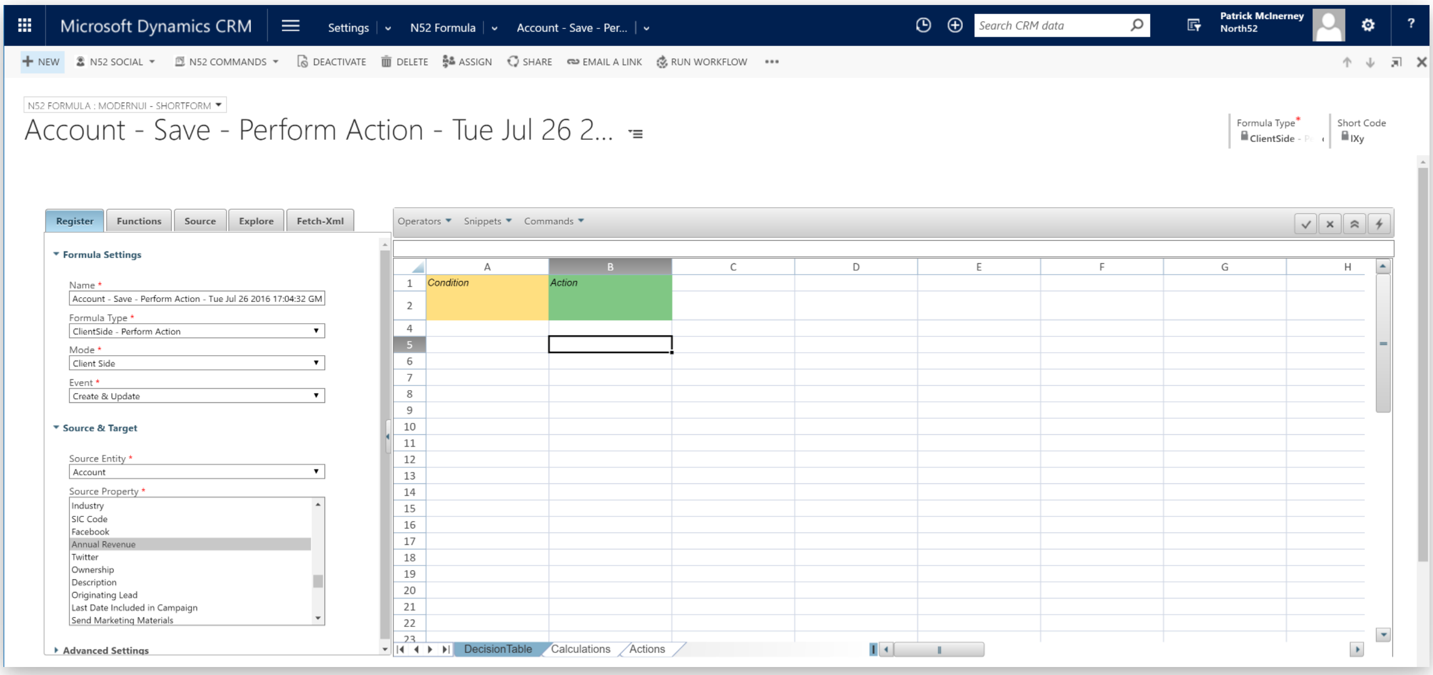
Setting up a Multi-Sheet Decision Table

To create a **Multi-Sheet Decision Table** you click on *Commands* and then on *Toggle Editor* as you would for a standard Decision Table.



Name *: Account - Save - Perform Action - Wed Sep 13 2017 09:41:48 GM1
Formula Type *: Save - Perform Action
Mode *: Server Side
Event *: Create & Update
Source Entity *: Account
Source Property *:
Address 2: UTC Offset
Address 2: ZIP/Postal Code
Address Phone
Aging 30
Aging 30 (Base)
Aging 60
Aging 60 (Base)
Aging 90
Aging 90 (Base)
Annual Revenue

This will open up the standard Decision Table canvas.



Account - Save - Perform Action - Tue Jul 26 2...

Formula Type: ClientSide
Short Code: IXy

Formula Settings

Name: Account - Save - Perform Action - Tue Jul 26 2016 17:04:32 GM
Formula Type: ClientSide - Perform Action
Mode: Client Side
Event: Create & Update

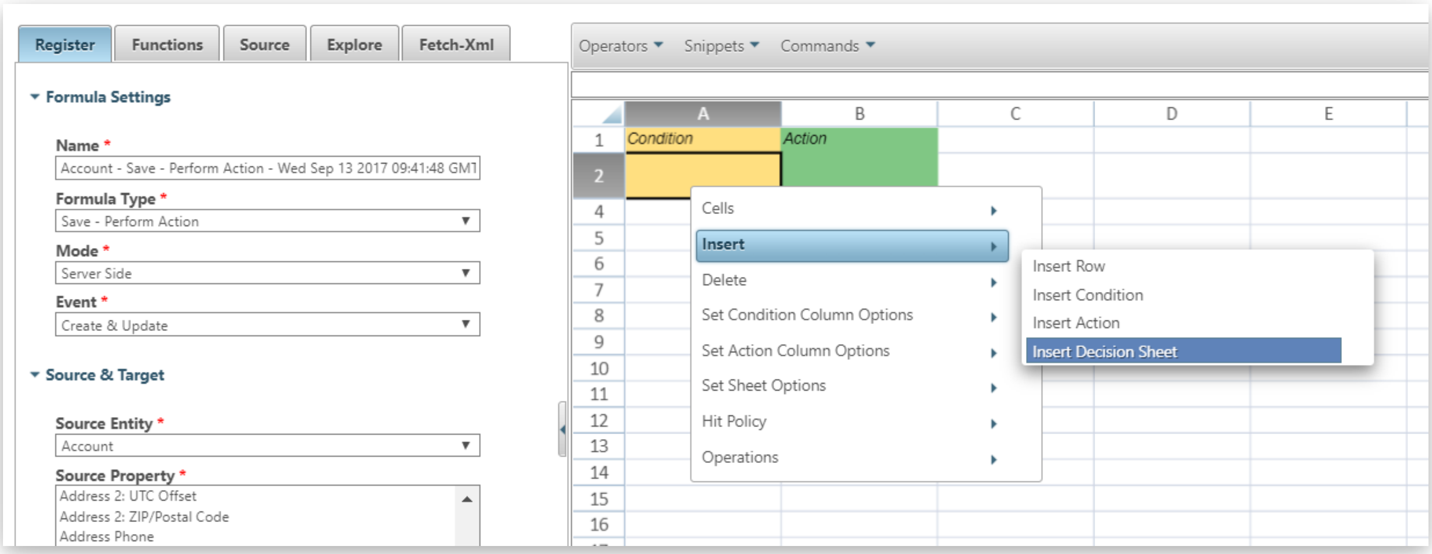
Source & Target

Source Entity: Account
Source Property: Industry, SIC Code, Facebook, Annual Revenue, Twitter, Ownership, Description, Originating Lead, Last Date Included in Campaign, Send Marketing Materials

As you can see from the above screenshot you now have **3 sheets** on the **Decision Table**.

- Decision Table
- Calculations
- Actions

Right-click on the Decision Table and the **Context-Menu** will popup, from this select *Insert > Insert Decision Table*



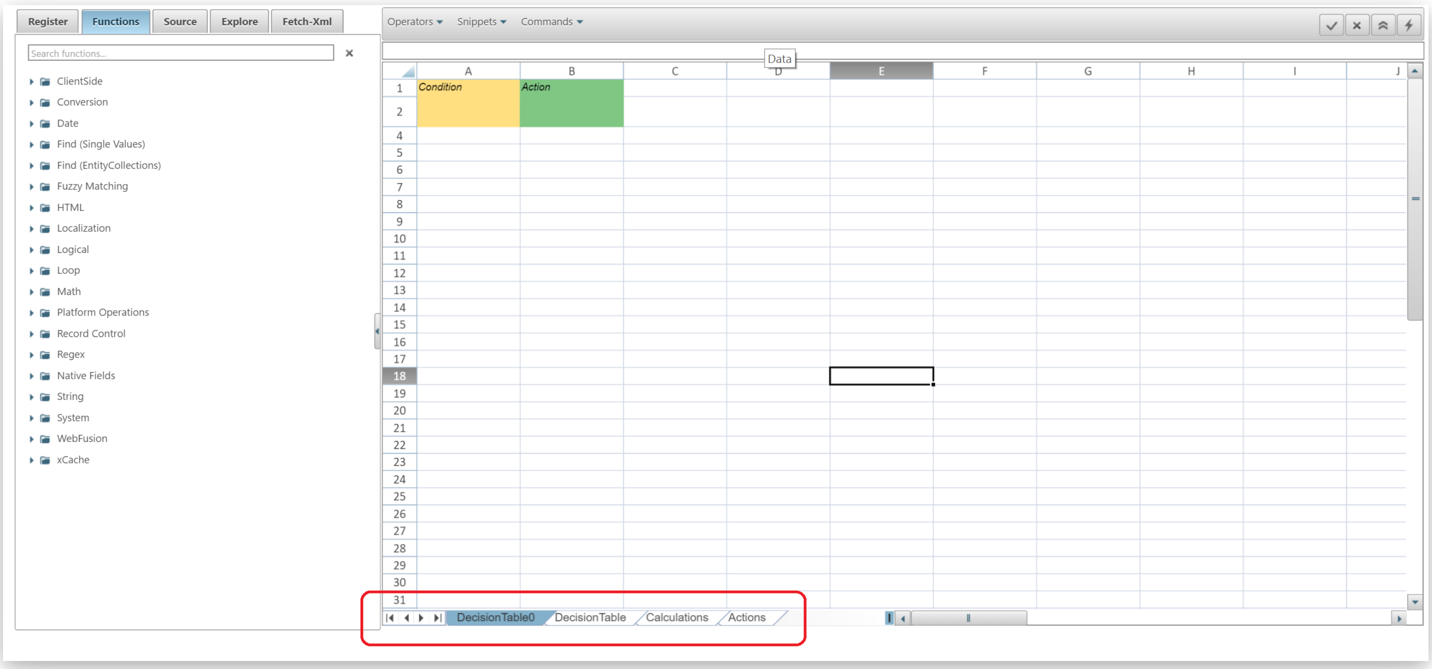
Formula Settings

Name: Account - Save - Perform Action - Wed Sep 13 2017 09:41:48 GM1
Formula Type: Save - Perform Action
Mode: Server Side
Event: Create & Update

Source & Target

Source Entity: Account
Source Property: Address 2: UTC Offset, Address 2: ZIP/Postal Code, Address Phone

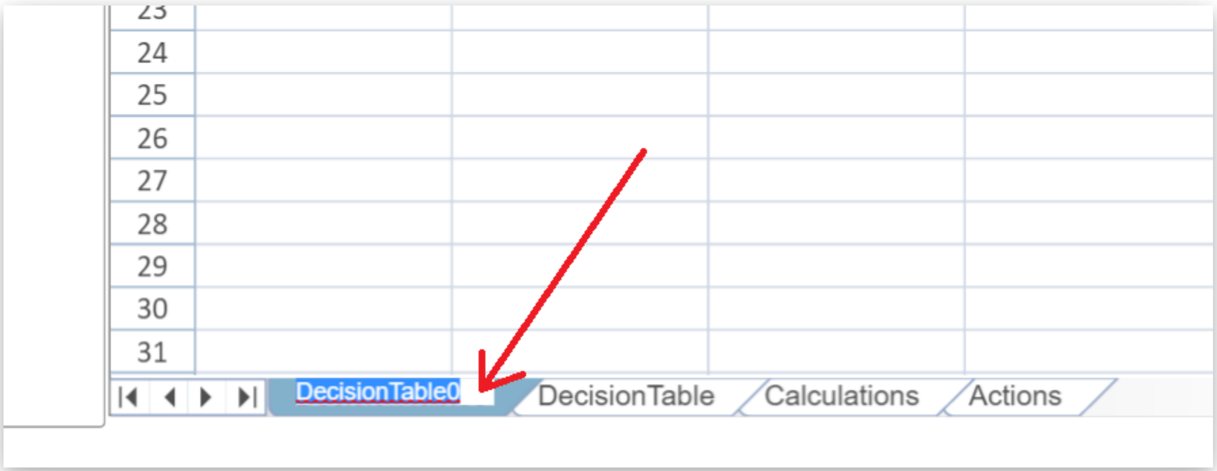
This will add a second **Decision Sheet.**



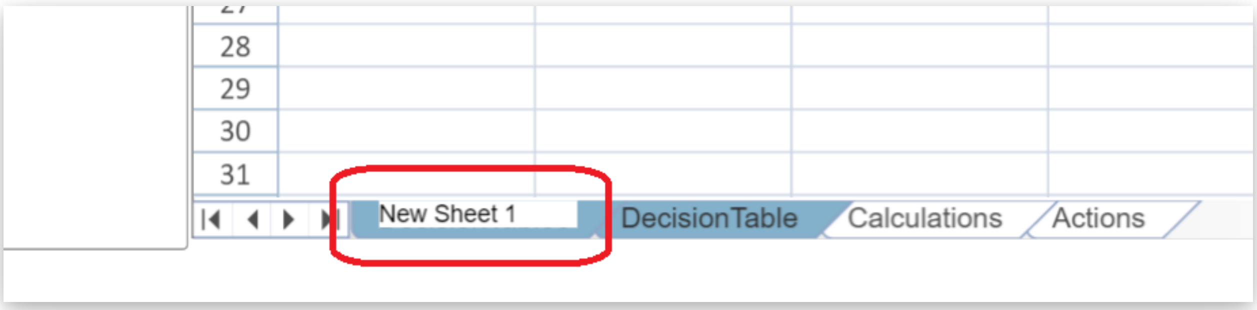
This is now a **Multiple-Sheet Decision Table.**

Renaming Sheets

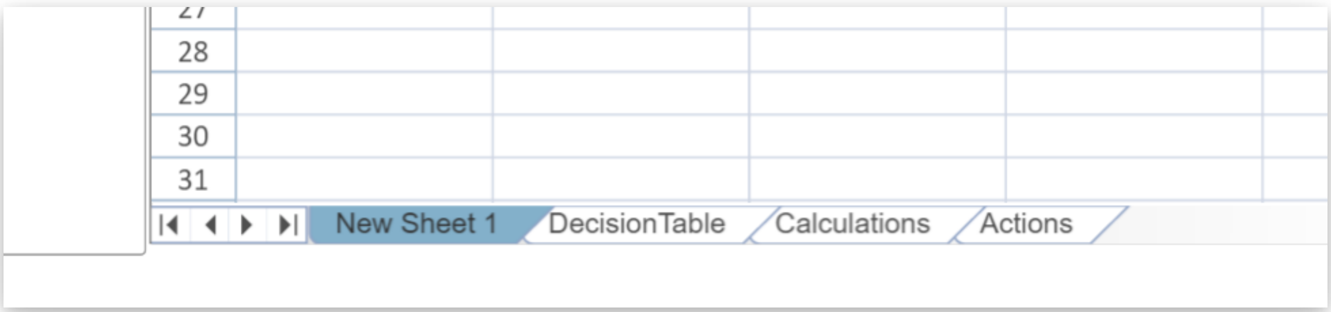
To rename a **Decision Sheet** *double click* on the sheet name.



The text will become editable and you can enter the new name for the **sheet.**



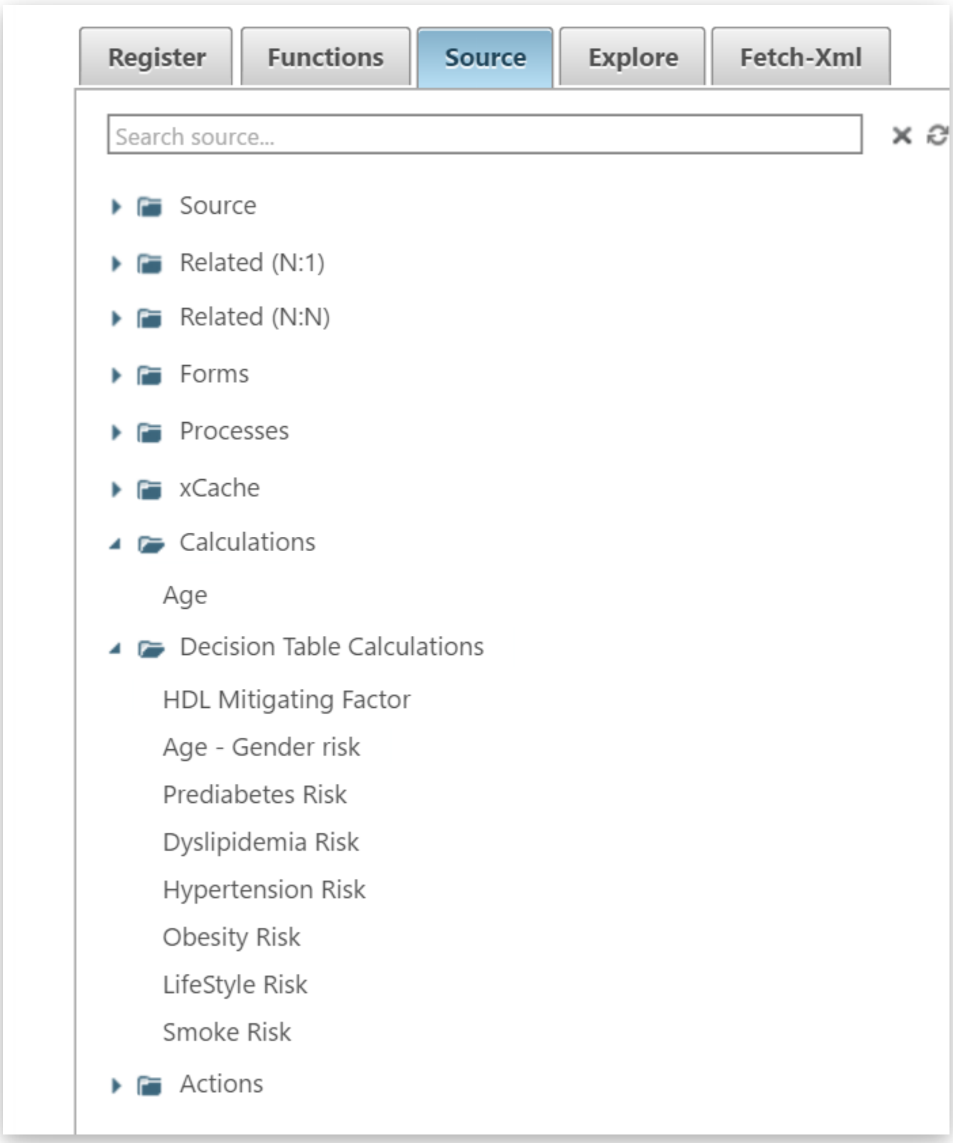
Once the sheet is renamed just click off the field and the **sheet** name will be changed to the new value.



Decision Table Calculations

Each **Sheet** in the **Multi-Sheet Decision Table** can generate results and these may need to be accessed from other **Sheets**.
These will show up under the **Decision Table Calculations node** on the **Source** tab of the configuration pane.

These are separate and distinct from regular **Calculations**.

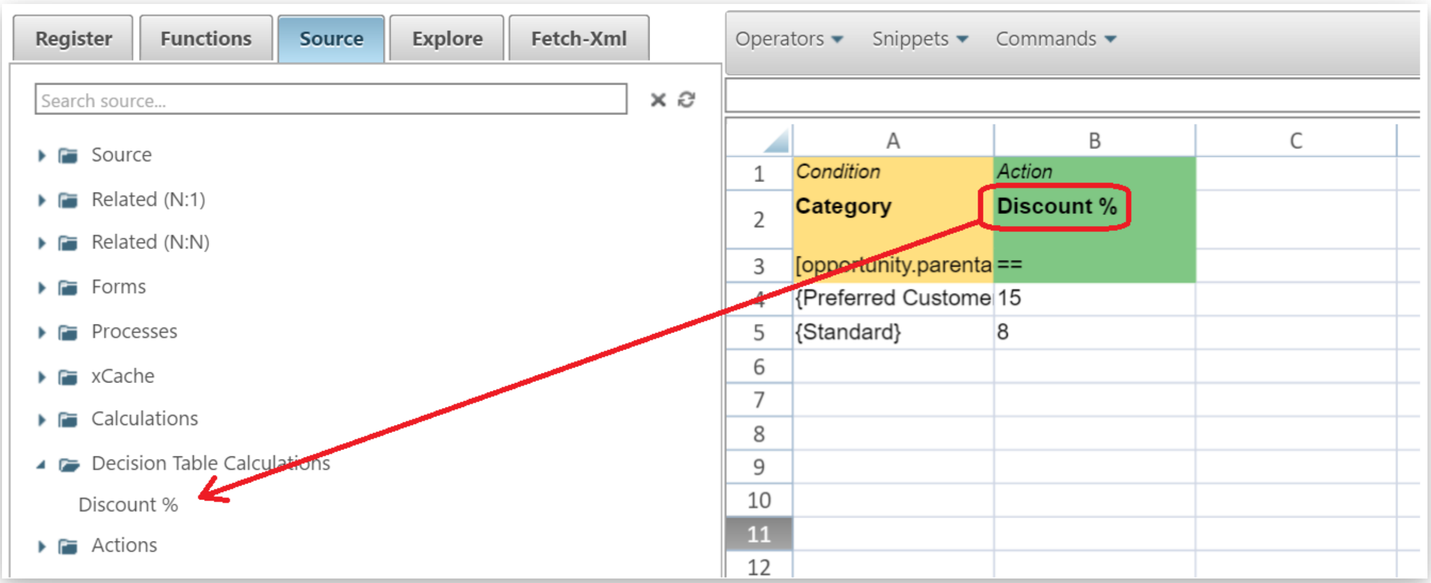


How to set up Decision Table Calculations

- Name your **Action** something meaningful
- *Right-click* on the Decision Table and select *Operations > Toggle Advanced Mode*
- **Row 3** will appear on the **Decision Table Sheet**
- In the **cell** below the **Action Name** type in "=="

- This will mark this **Action** as a **Decision Table Calculation**
- This **Action Name** will now appear in the **Decision Table Calculations** node of the **Source** tab

In the example below we are marking **Discount %** as a **Decision Table Calculation**



	A	B	C
1	Condition	Action	
2	Category	Discount %	
3	[opportunity.parenta	==	
4	{Preferred Custome	15	
5	{Standard}	8	
6			
7			
8			
9			
10			
11			
12			

This Decision Table Calculation is now available to be used in other **Decision Table Sheets**.

Copying Entire Decision Table Sheets

It is possible to copy an entire **Decision Table Sheet** within the same **Multi-Sheet Decision Table.**

To do this:

- *Right-Click* on the **Decision Table Sheet** to bring up the **context-menu**
- Hold down the *CTRL* key on your keyboard and click on *Cells* > *Copy* from the **context-menu**
- The entire **Decision Table Sheet** will be duplicated

DT - How to - 10 - Introduction to Multi-Sheet Decision Tables

[TOC]

Overview

Multi-Sheet Decision Tables allow you to take complex business requirements and break them down into constituent sets of business rules. It does this by allowing you to have multiple decision sheets with the formula. This can useful for both making your **Decision Table** logic **more readable** as well as separation of different **entitlement criteria.**

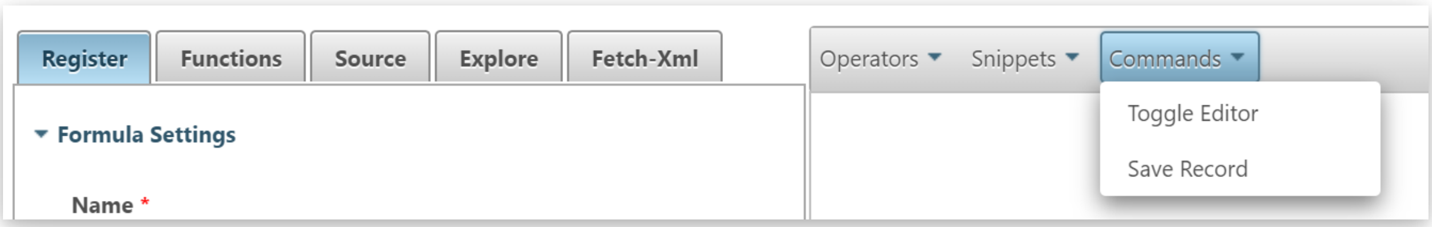
For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

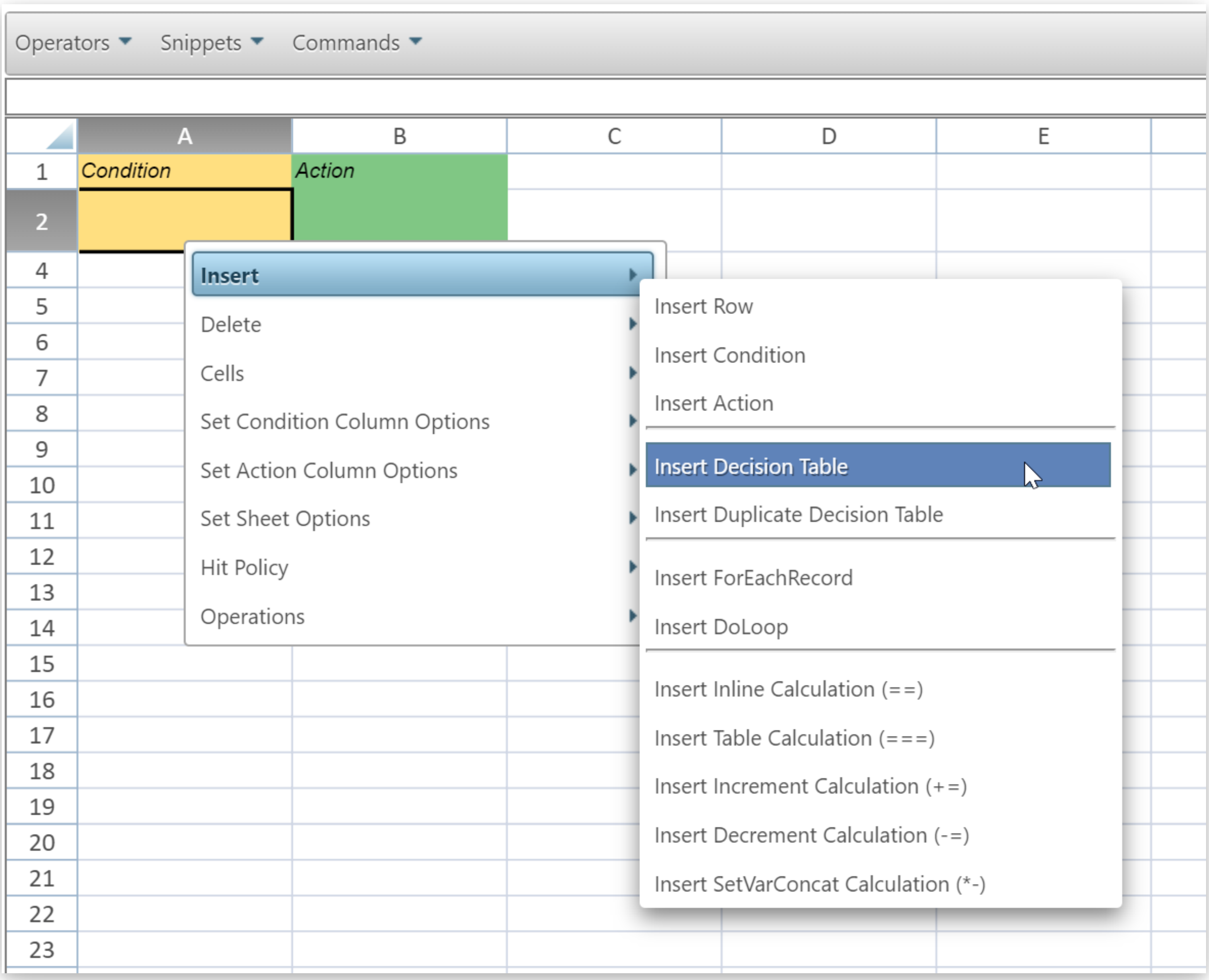
Note: We will not detail step-by-step instructions here on how to set up **Conditions** or **Actions,** please read the above articles if you need detailed configuration steps.

Setting up a Multi-Sheet Decision Table

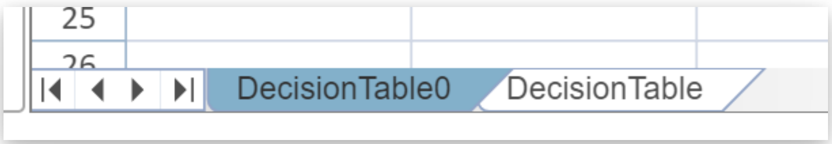
If you are in the Classic editor mode click on *Commands* and then on *Toggle Editor* to change to the Decision Table editor.



Right-click on the Decision Table and the **Context-Menu** will show, from this select *Insert > Insert Decision Table*


Operators
Snippets
Commands
Condition
Action
Insert
Delete
Cells
Set Condition Column Options
Set Action Column Options
Set Sheet Options
Hit Policy
Operations
Insert Row
Insert Condition
Insert Action
Insert Decision Table
Insert Duplicate Decision Table
Insert ForEachRecord
Insert DoLoop
Insert Inline Calculation (==)
Insert Table Calculation (===)
Insert Increment Calculation (+=)
Insert Decrement Calculation (-=)
Insert SetVarConcat Calculation (*-)

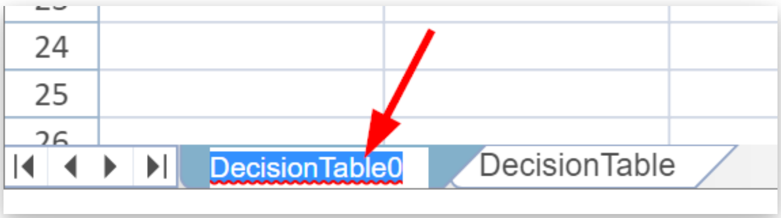
This will add a second **Decision Sheet:**



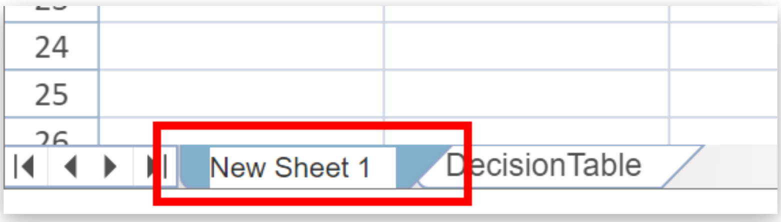
This is now a **Multiple-Sheet Decision Table.**

Renaming Sheets

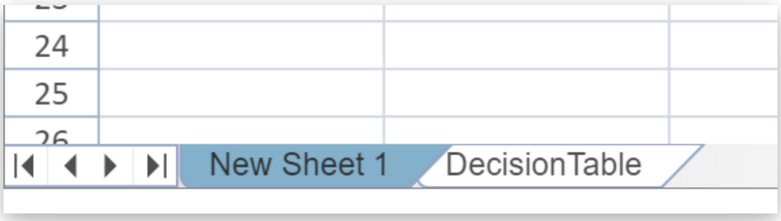
To rename a **Decision Sheet** *double click* on the sheet name.



The text will become editable and you can enter the new name for the **sheet.**



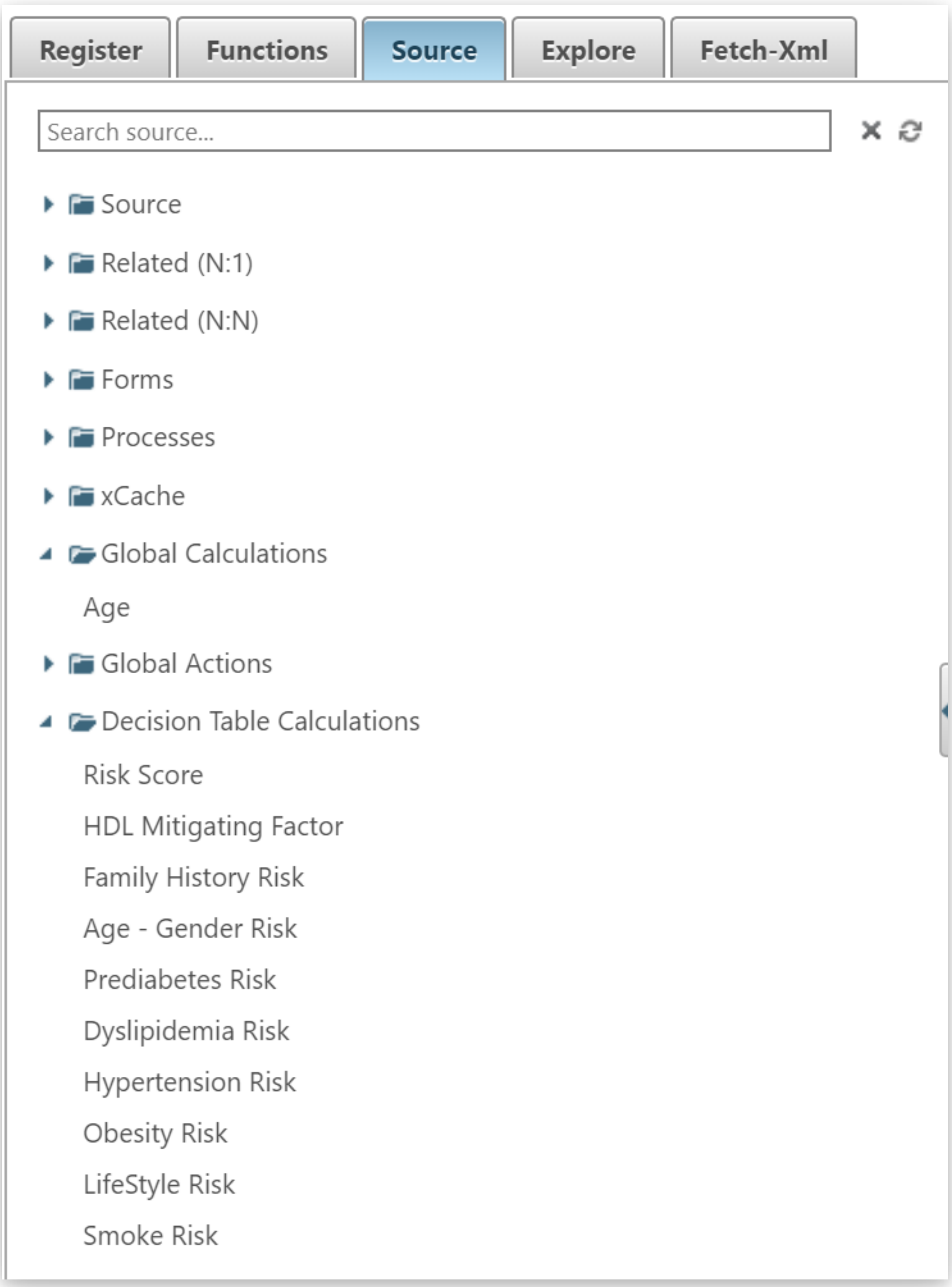
Once the sheet is renamed just click off the field and the **sheet** name will be changed to the new value.



Decision Table Calculations

Each **Sheet** in the **Multi-Sheet Decision Table** can generate results and these may need to be accessed from other **Sheets**. These will show up under the **Decision Table Calculations node** on the **Source** tab of the configuration pane.

These are separate and distinct from **Global Calculations.**

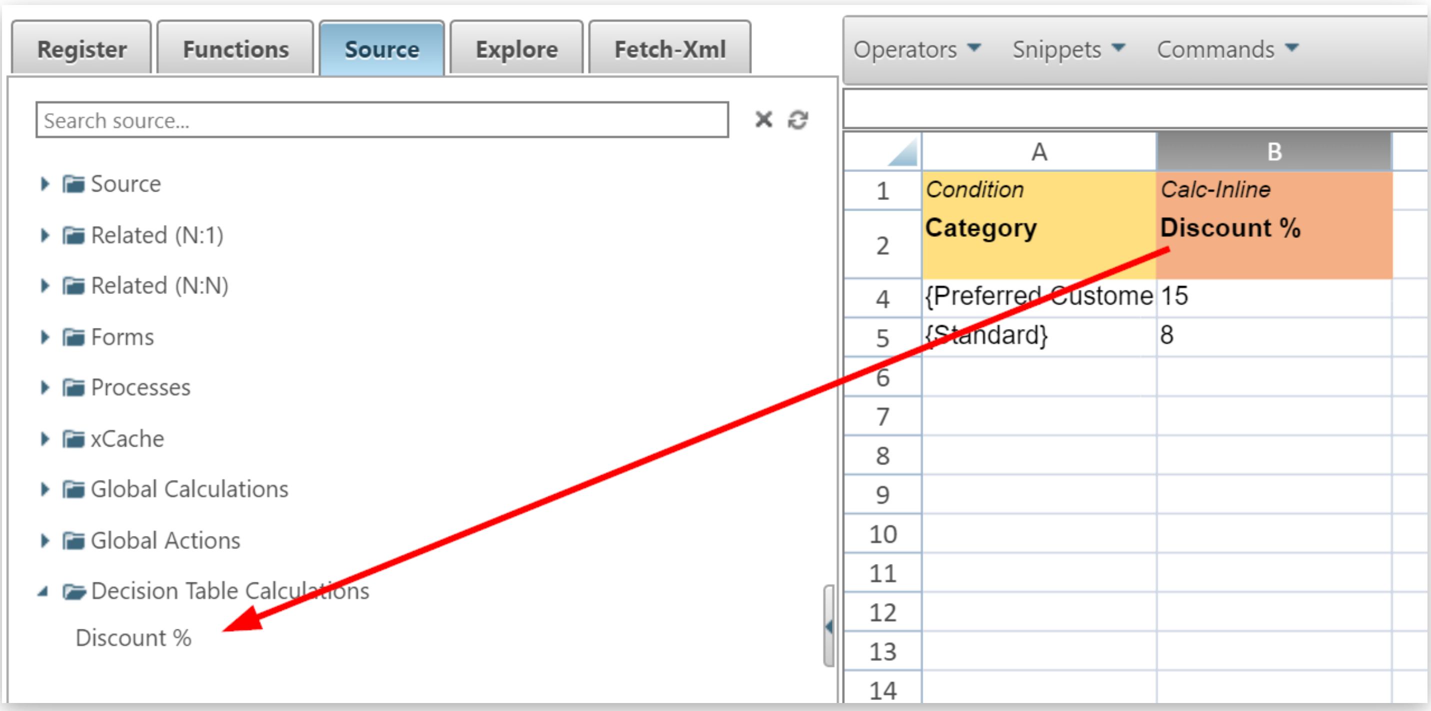


- Source
- Related (N:1)
- Related (N:N)
- Forms
- Processes
- xCache
- Global Calculations
  - Age
- Global Actions
- Decision Table Calculations
  - Risk Score
  - HDL Mitigating Factor
  - Family History Risk
  - Age - Gender Risk
  - Prediabetes Risk
  - Dyslipidemia Risk
  - Hypertension Risk
  - Obesity Risk
  - LifeStyle Risk
  - Smoke Risk

How to set up Decision Table Calculations

- *Right-click* on the Decision Table and select *Insert > Insert Inline Calculation (==)*
- Replace the **Set Name** placeholder text with something relevant to your needs
- This **Calculation Name** will now appear in the **Decision Table Calculations** node of the **Source** tab

In the example below we are marking **Discount %** as a **Decision Table Calculation**



	A	B
1	*Condition*	*Calc-Inline*
2	**Category**	**Discount %**
4	{Preferred Custome	15
5	{Standard}	8
6		
7		
8		
9		
10		
11		
12		
13		
14		

This Decision Table Calculation is now available to be used in other **Decision Table Sheets**.

Copying Entire Decision Table Sheets

It is possible to copy an entire **Decision Table Sheet** within the same **Multi-Sheet Decision Table**.

To do this:

- *Right-click* on the Decision Table and select *Insert > Insert Duplicate Decision Table*
- The entire **Decision Table Sheet** will be duplicated
- Rename the copied sheet to something meaningful to your needs

DT - How to - 11 - Use Find and Replace in a Decision Table

[TOC]

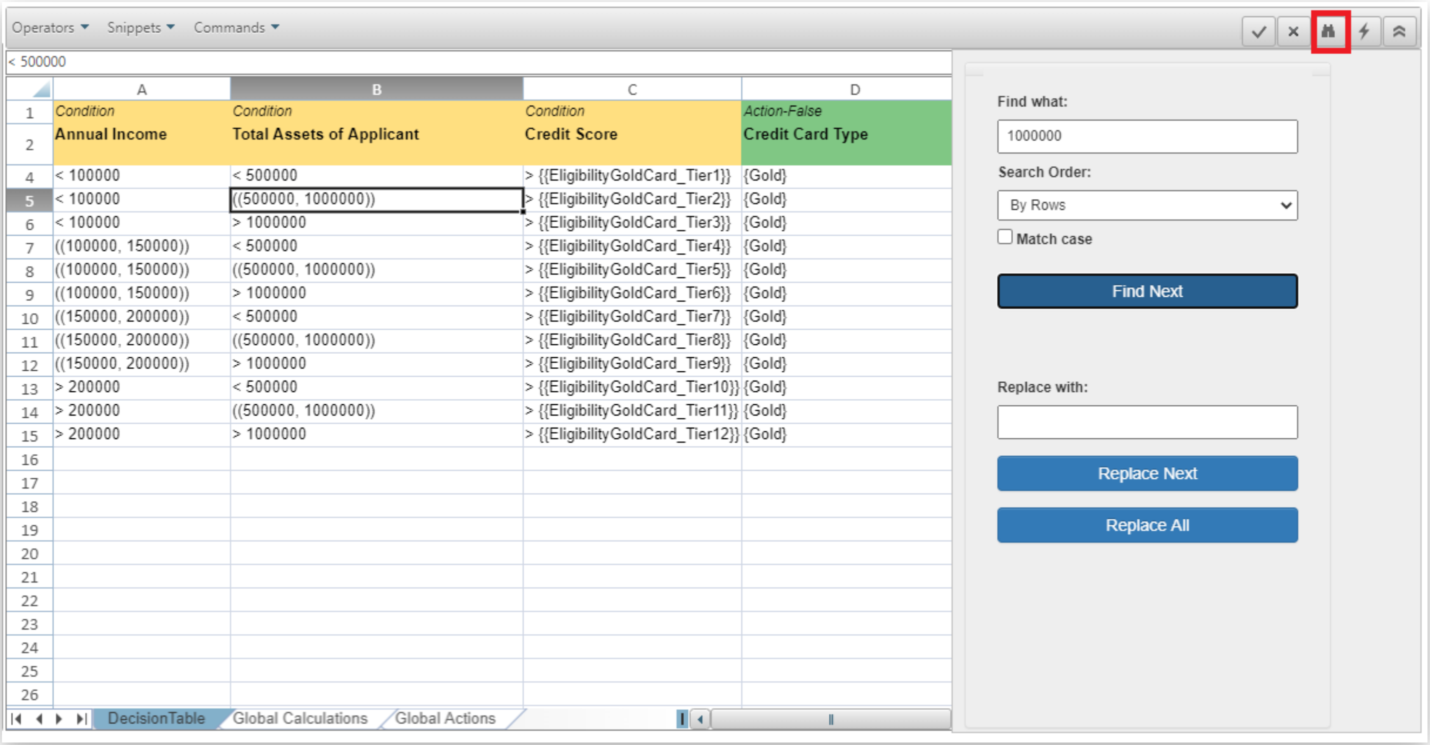
Overview

The **Find** and **Replace** functionality can quickly and accurately find all (and potentially change) references to fields, functions and strings. It works well in conjunction with **N52 Commands** > **Clone Formula** when you have a similar formula to replicate.

For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: *We will not detail step-by-step instructions here on how to set up **Conditions** or **Actions**, please read the above articles if you need detailed configuration steps.*



< 500000

	A	B	C	D
1	*Condition*	*Condition*	*Condition*	*Action-False*
2	**Annual Income**	**Total Assets of Applicant**	**Credit Score**	**Credit Card Type**
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

Find what: 1000000

Search Order: By Rows

Match case

Find Next

Replace with:

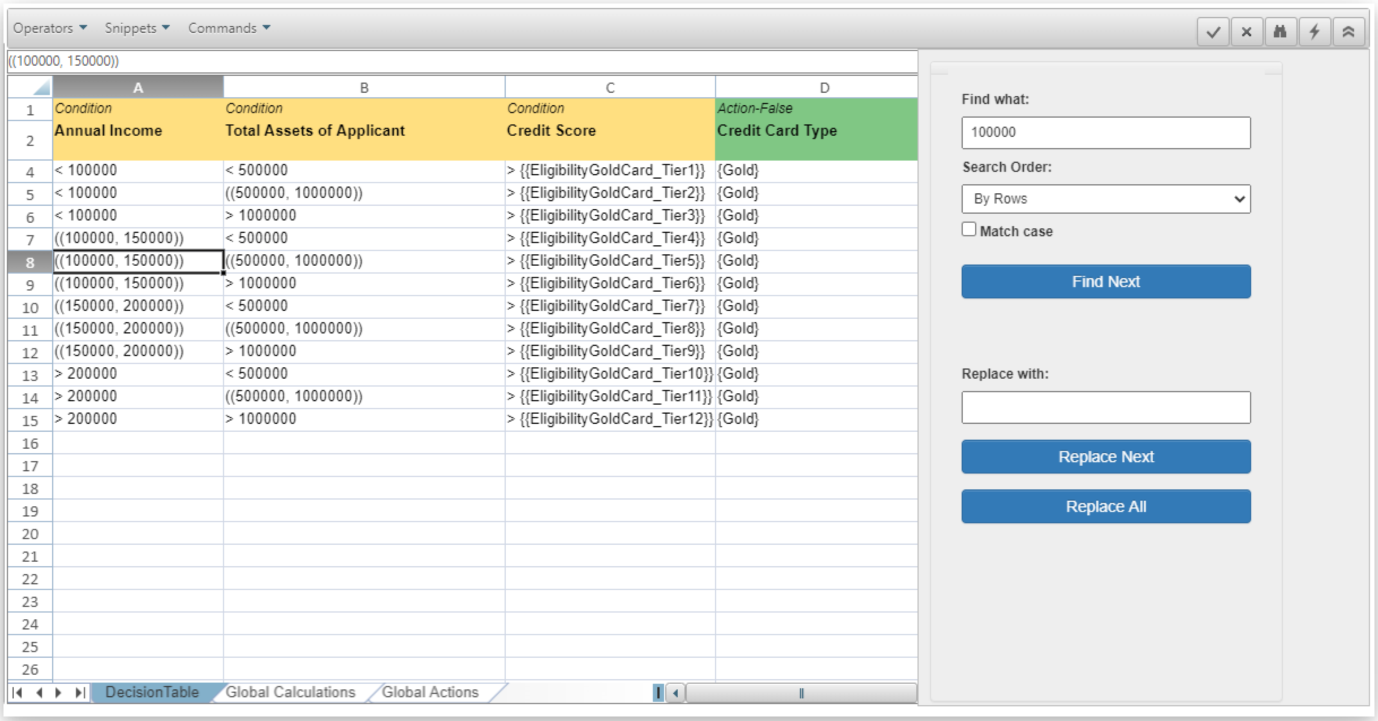
Replace Next

Replace All

Find

You can quickly search all Decision Sheets for any text, function or reference inside all Decision Table sheets.

You can search row by row:


Operators ▾ Snippets ▾ Commands ▾

((100000, 150000))

	A	B	C	D
1	*Condition*	*Condition*	*Condition*	*Action-False*
2	**Annual Income**	**Total Assets of Applicant**	**Credit Score**	**Credit Card Type**
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

DecisionTable | Global Calculations | Global Actions

Find what: 100000

Search Order: By Rows

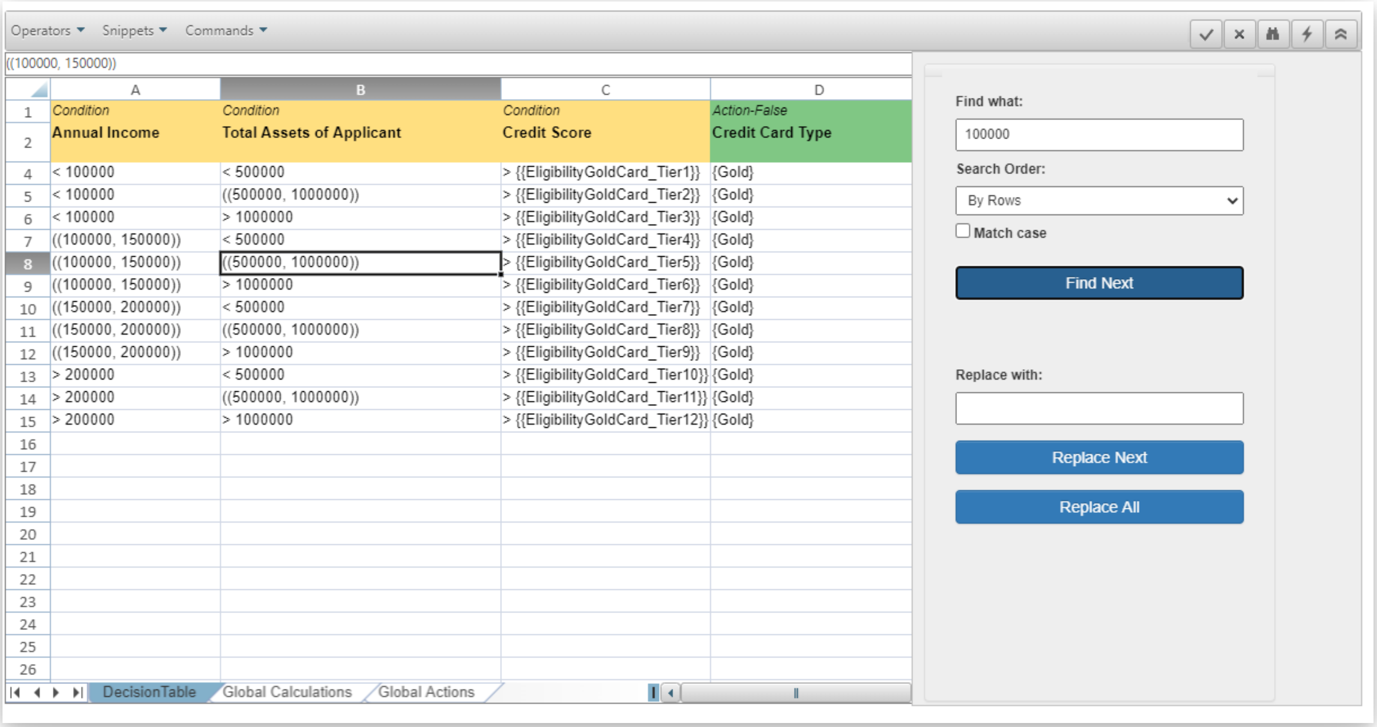
☐ Match case

Find Next

Replace with:

Replace Next

Replace All


Operators ▾ Snippets ▾ Commands ▾

((100000, 150000))

	A	B	C	D
1	*Condition*	*Condition*	*Condition*	*Action-False*
2	**Annual Income**	**Total Assets of Applicant**	**Credit Score**	**Credit Card Type**
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

DecisionTable | Global Calculations | Global Actions

Find what: 100000

Search Order: By Rows

☐ Match case

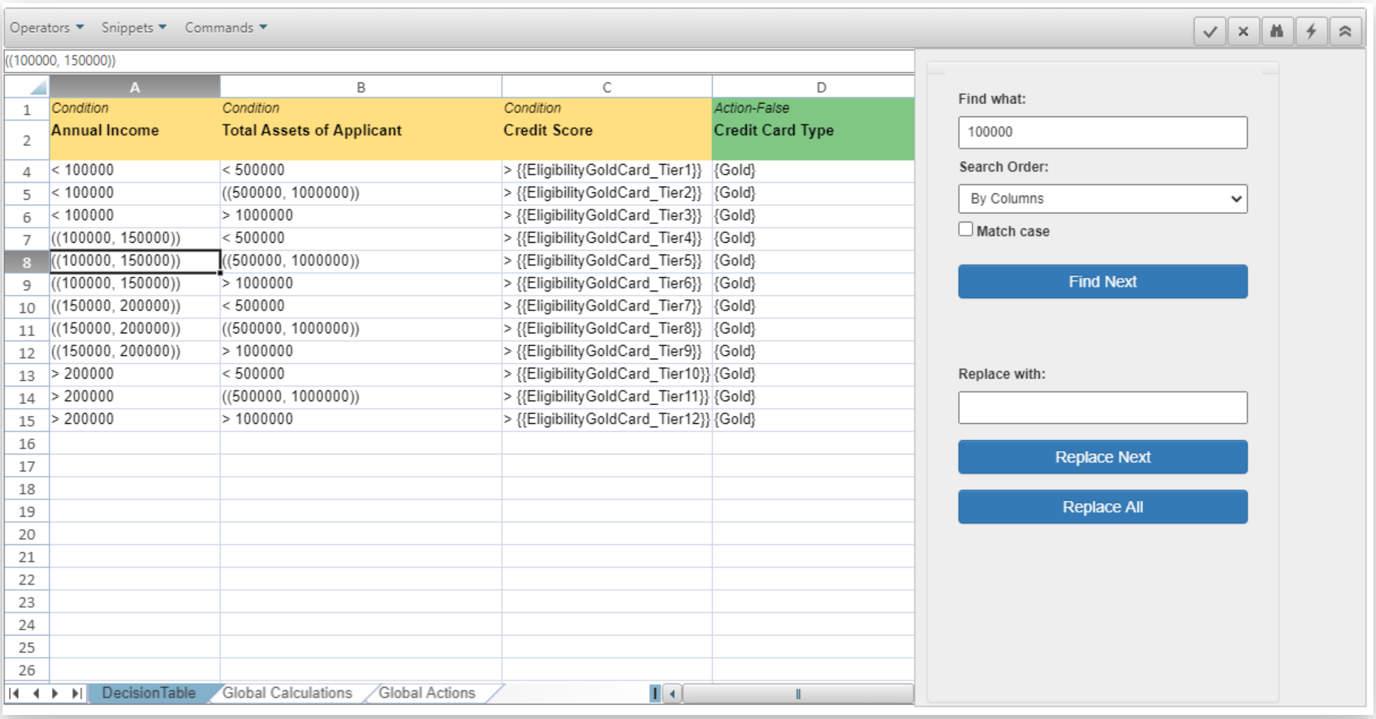
Find Next

Replace with:

Replace Next

Replace All

or column by column



((100000, 150000))

	A	B	C	D
1	Condition	Condition	Condition	Action-False
2	Annual Income	Total Assets of Applicant	Credit Score	Credit Card Type
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

Find what: 100000

Search Order: By Columns

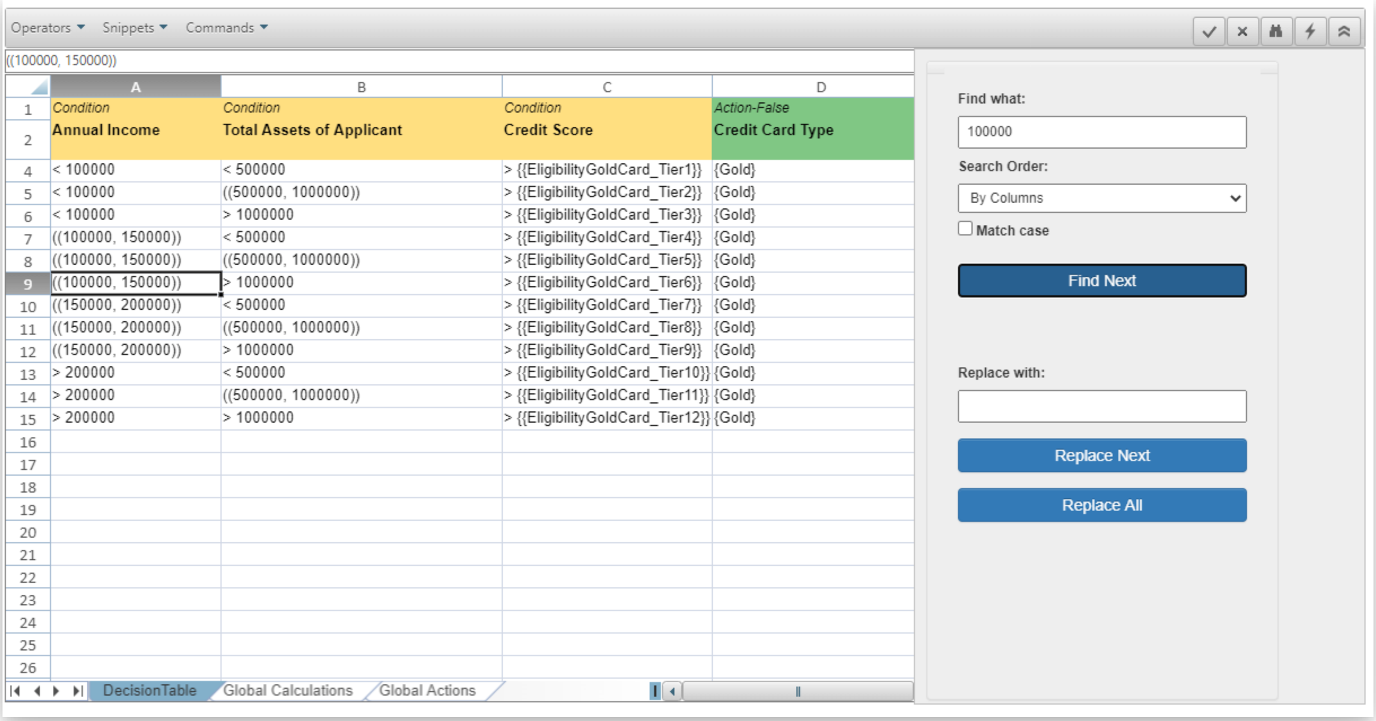
☐ Match case

Find Next

Replace with:

Replace Next

Replace All



((100000, 150000))

	A	B	C	D
1	Condition	Condition	Condition	Action-False
2	Annual Income	Total Assets of Applicant	Credit Score	Credit Card Type
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

Find what: 100000

Search Order: By Columns

☐ Match case

Find Next

Replace with:

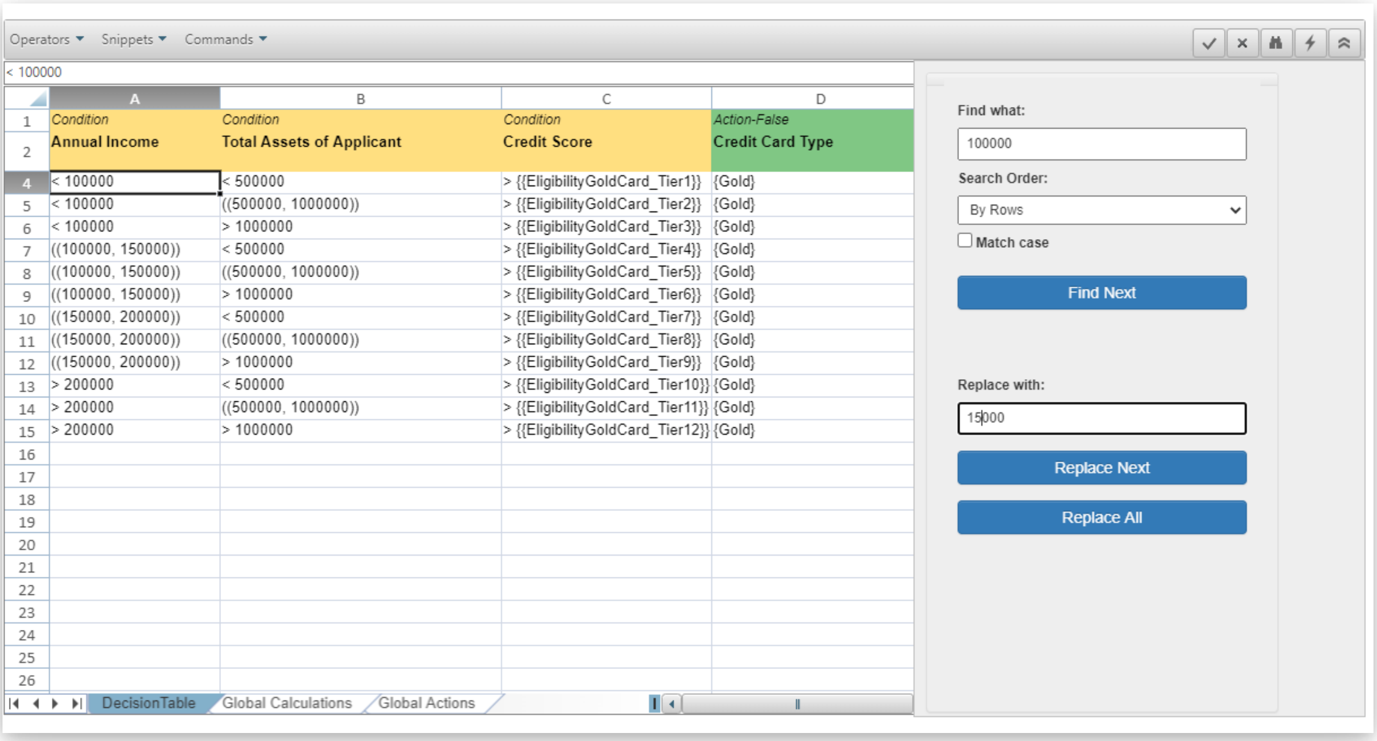
Replace Next

Replace All

Replace

Anything you find can be also replaced. This allows for quick renaming or adjustments to your Decision Table.

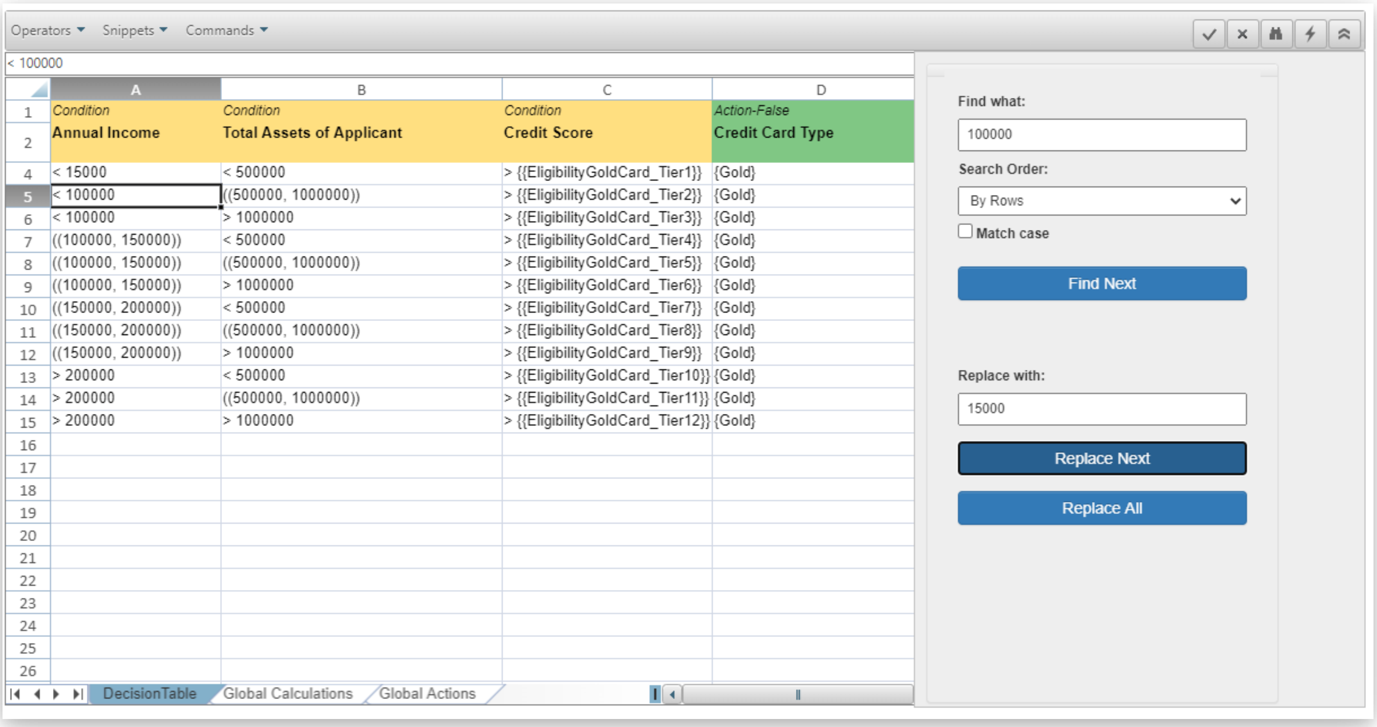
Replace values one cell at a time:



< 100000

	A	B	C	D
1	Condition	Condition	Condition	Action-False
2	Annual Income	Total Assets of Applicant	Credit Score	Credit Card Type
4	< 100000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}

Find what: 100000
Search Order: By Rows
Match case
Replace with: 15000

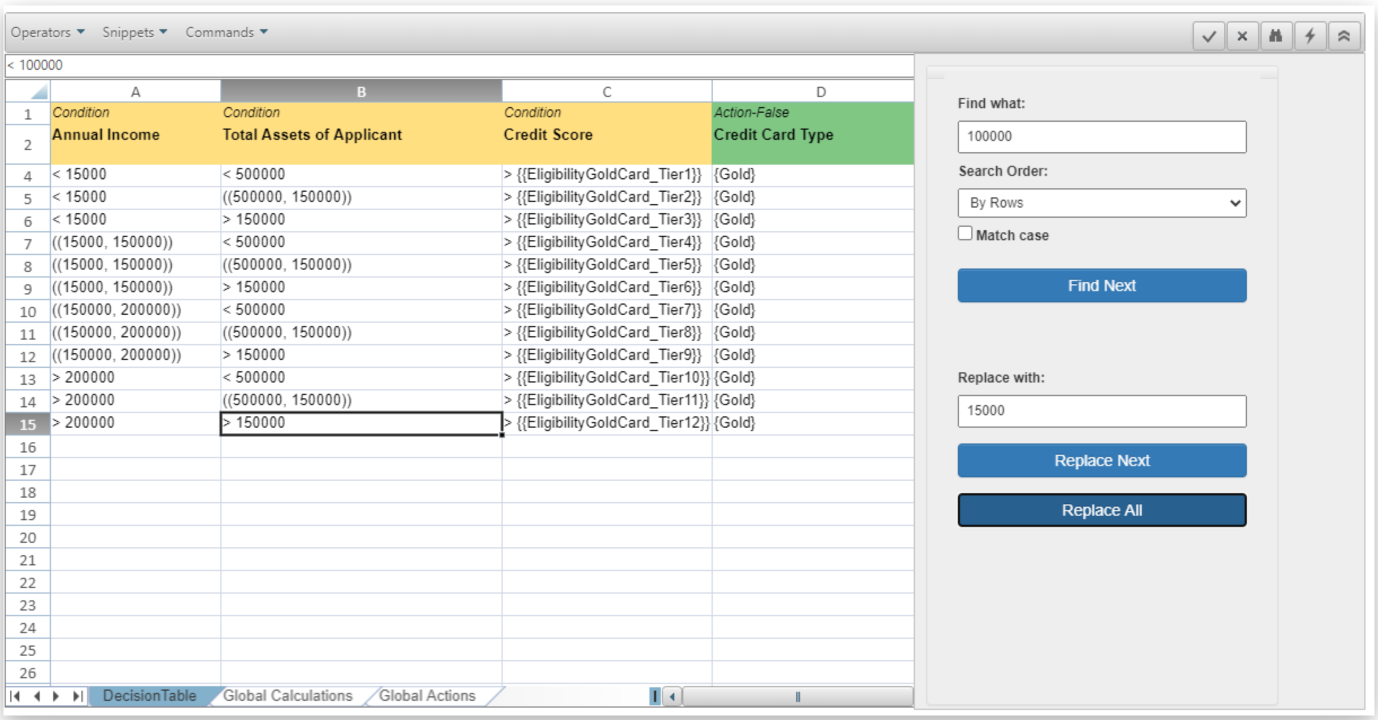


< 100000

	A	B	C	D
1	Condition	Condition	Condition	Action-False
2	Annual Income	Total Assets of Applicant	Credit Score	Credit Card Type
4	< 15000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 100000	((500000, 1000000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 100000	> 1000000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((100000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((100000, 150000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((100000, 150000))	> 1000000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 1000000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 1000000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 1000000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 1000000	> {{EligibilityGoldCard_Tier12}}	{Gold}

Find what: 100000
Search Order: By Rows
Match case
Replace with: 15000

Or all at once:



< 100000

	A	B	C	D
1	*Condition*	*Condition*	*Condition*	*Action-False*
2	**Annual Income**	**Total Assets of Applicant**	**Credit Score**	**Credit Card Type**
4	< 15000	< 500000	> {{EligibilityGoldCard_Tier1}}	{Gold}
5	< 15000	((500000, 150000))	> {{EligibilityGoldCard_Tier2}}	{Gold}
6	< 15000	> 150000	> {{EligibilityGoldCard_Tier3}}	{Gold}
7	((15000, 150000))	< 500000	> {{EligibilityGoldCard_Tier4}}	{Gold}
8	((15000, 150000))	((500000, 150000))	> {{EligibilityGoldCard_Tier5}}	{Gold}
9	((15000, 150000))	> 150000	> {{EligibilityGoldCard_Tier6}}	{Gold}
10	((150000, 200000))	< 500000	> {{EligibilityGoldCard_Tier7}}	{Gold}
11	((150000, 200000))	((500000, 150000))	> {{EligibilityGoldCard_Tier8}}	{Gold}
12	((150000, 200000))	> 150000	> {{EligibilityGoldCard_Tier9}}	{Gold}
13	> 200000	< 500000	> {{EligibilityGoldCard_Tier10}}	{Gold}
14	> 200000	((500000, 150000))	> {{EligibilityGoldCard_Tier11}}	{Gold}
15	> 200000	> 150000	> {{EligibilityGoldCard_Tier12}}	{Gold}
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				

Find what: 100000

Search Order: By Rows

Match case: [ ]

Replace with: 15000

DT - How to - 12 - Use Global Fetch XML Sheet

[TOC]

Overview

The **Global FetchXml** sheet is displayed when using the Decision Tables Advanced Mode. It allows the user to quickly add and edit Fetch XML queries to be referenced by specific functions (e.g. FindValueFD) with the Decision Table sheets.

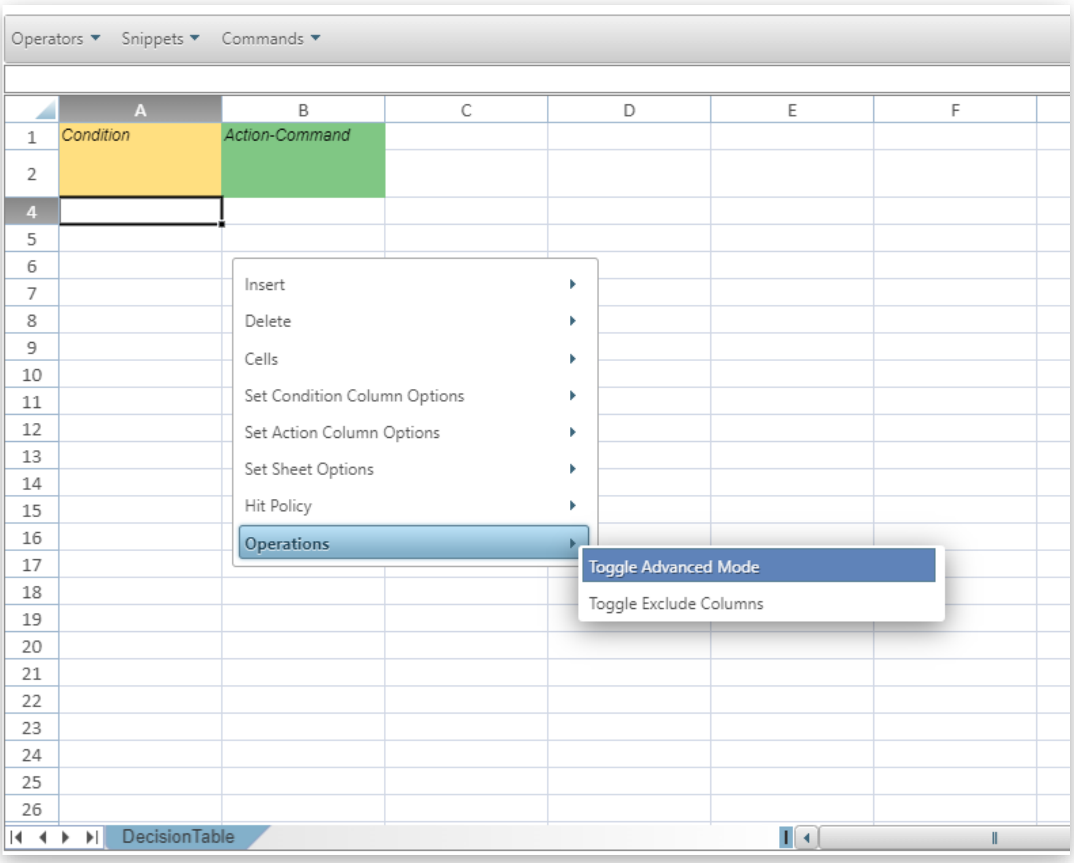
For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

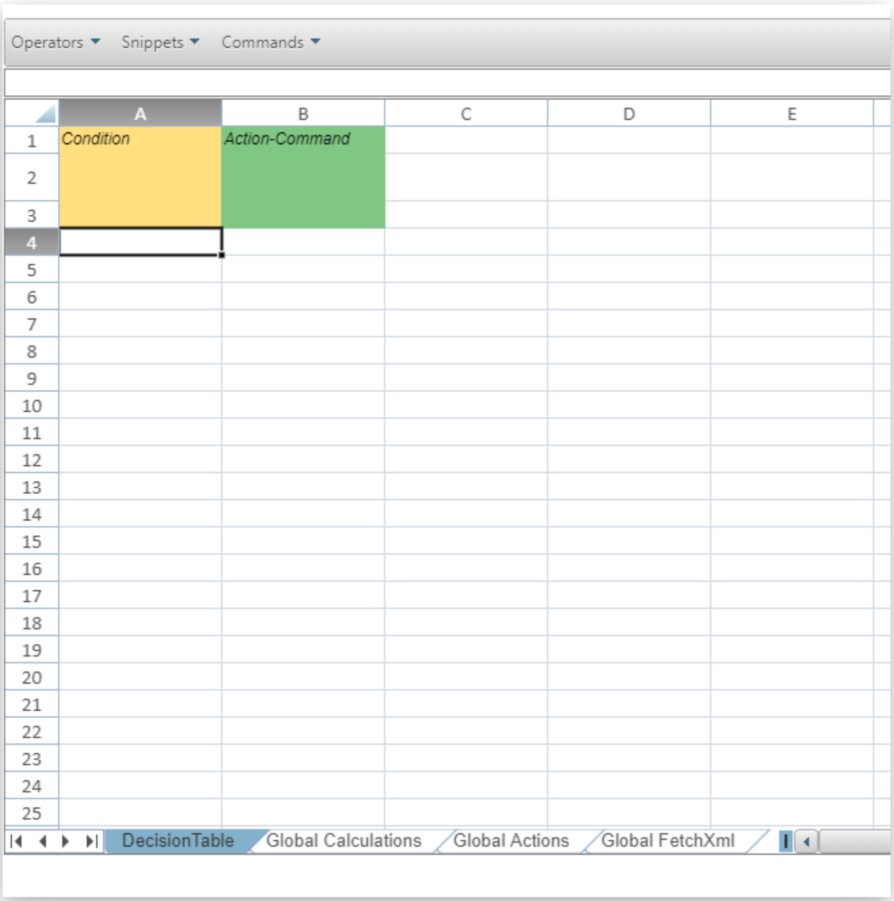
Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions****, please read the above articles if you need detailed configuration steps.*

Where to find Global FetchXml

To show the **Global FetchXml** sheet, open a Decision Table, right-click and select **Operations > Toggle Advanced Mode**

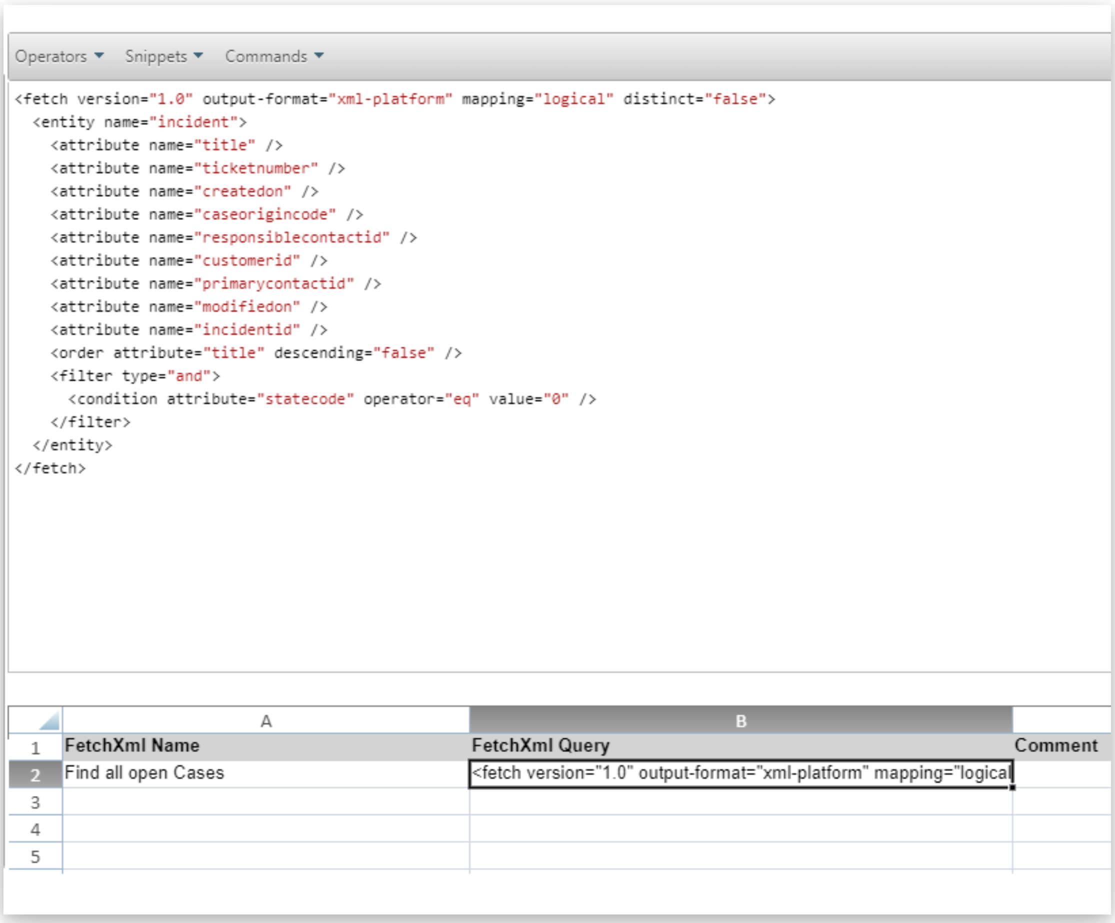


You will then see the **Global FetchXml** sheet (along with other Advance Mode sheets like **Global Calculation**)



How to setup Global Fetch XML

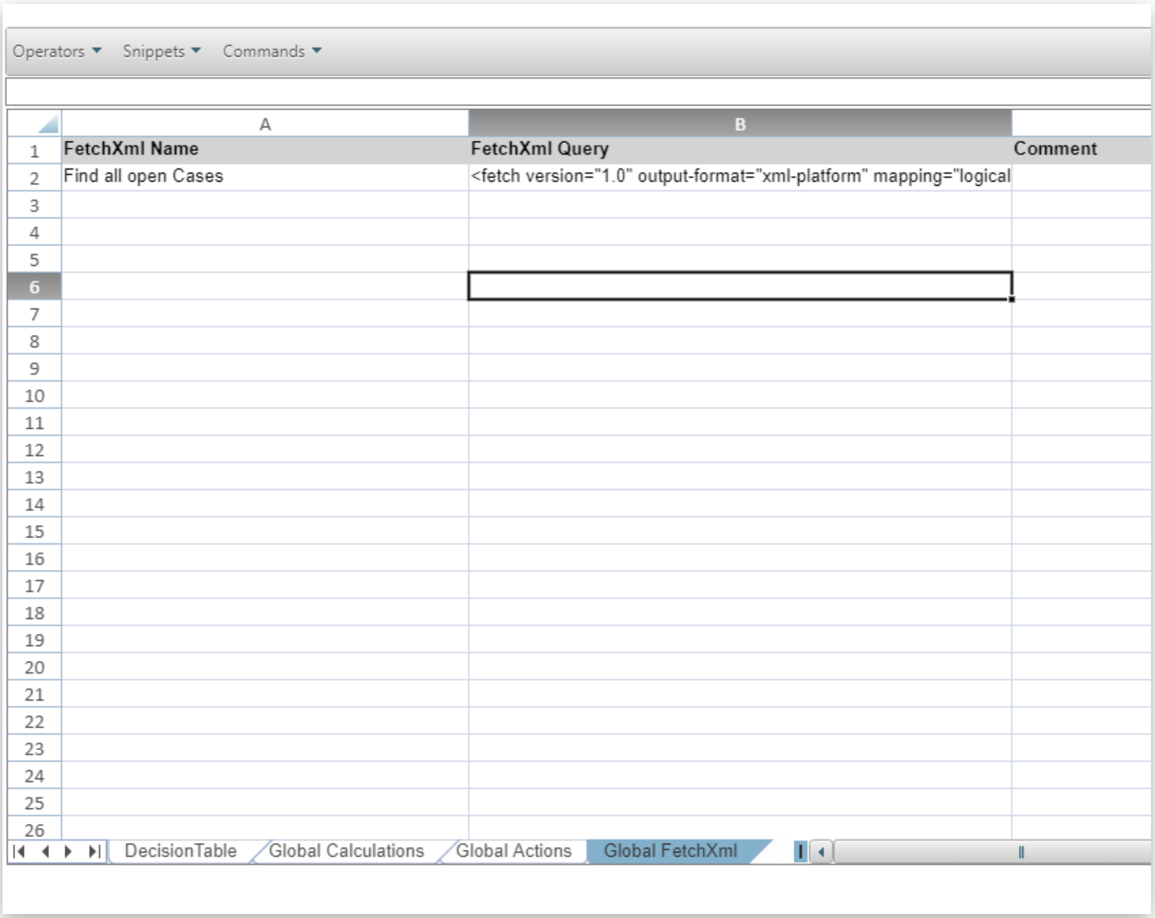
- In a cell in column **A**, give the **Fetch XML** a name
- Then expand the advanced editor for the corresponding cell in column **B**
- Enter the **Fetch XML** - this example is a query to pull back all active cases (if required you can parameterize the Fetch XML query using {0}, {1}... {n} placeholders)



```
<fetch version="1.0" output-format="xml-platform" mapping="logical" distinct="false">
  <entity name="incident">
    <attribute name="title" />
    <attribute name="ticketnumber" />
    <attribute name="createdon" />
    <attribute name="caseorigincode" />
    <attribute name="responsiblecontactid" />
    <attribute name="customerid" />
    <attribute name="primarycontactid" />
    <attribute name="modifiedon" />
    <attribute name="incidentid" />
    <order attribute="title" descending="false" />
    <filter type="and">
      <condition attribute="statecode" operator="eq" value="0" />
    </filter>
  </entity>
</fetch>
```

	A	B	
1	FetchXml Name	FetchXml Query	Comment
2	Find all open Cases	<fetch version="1.0" output-format="xml-platform" mapping="logical	
3			
4			
5			

- Collapse the Advanced Editor and click Save

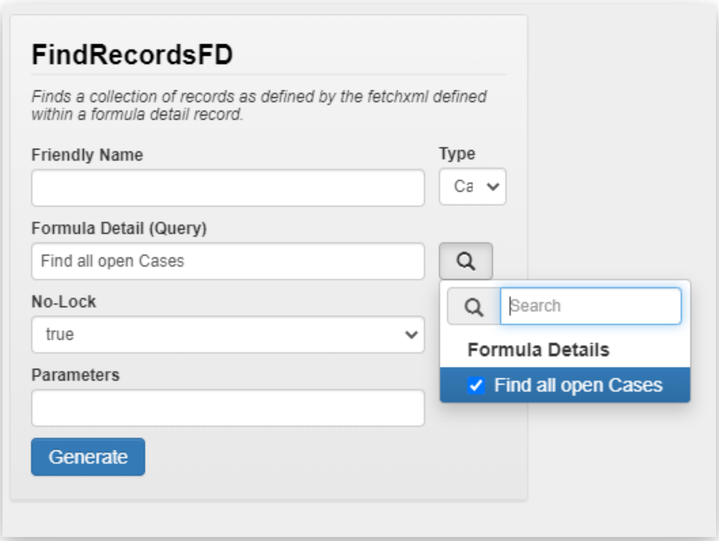


	A	B	
1	FetchXml Name	FetchXml Query	Comment
2	Find all open Cases	<fetch version="1.0" output-format="xml-platform" mapping="logical	

- Go to the **Source** tab and select the refresh icon
- Go to the **Fetch XML tab**, you will see your new **Fetch XML**
- Now you will be able to reference this **Fetch XML**

Example

Below is an example showing how you can reference the **Fetch XML** you specify on the Global Fetch XML sheet. The scenario shows how we can loop over the records returned by the Fetch XML query and update the records using a Decision Table).


FindRecordsFD

Finds a collection of records as defined by the fetchxml defined within a formula detail record.

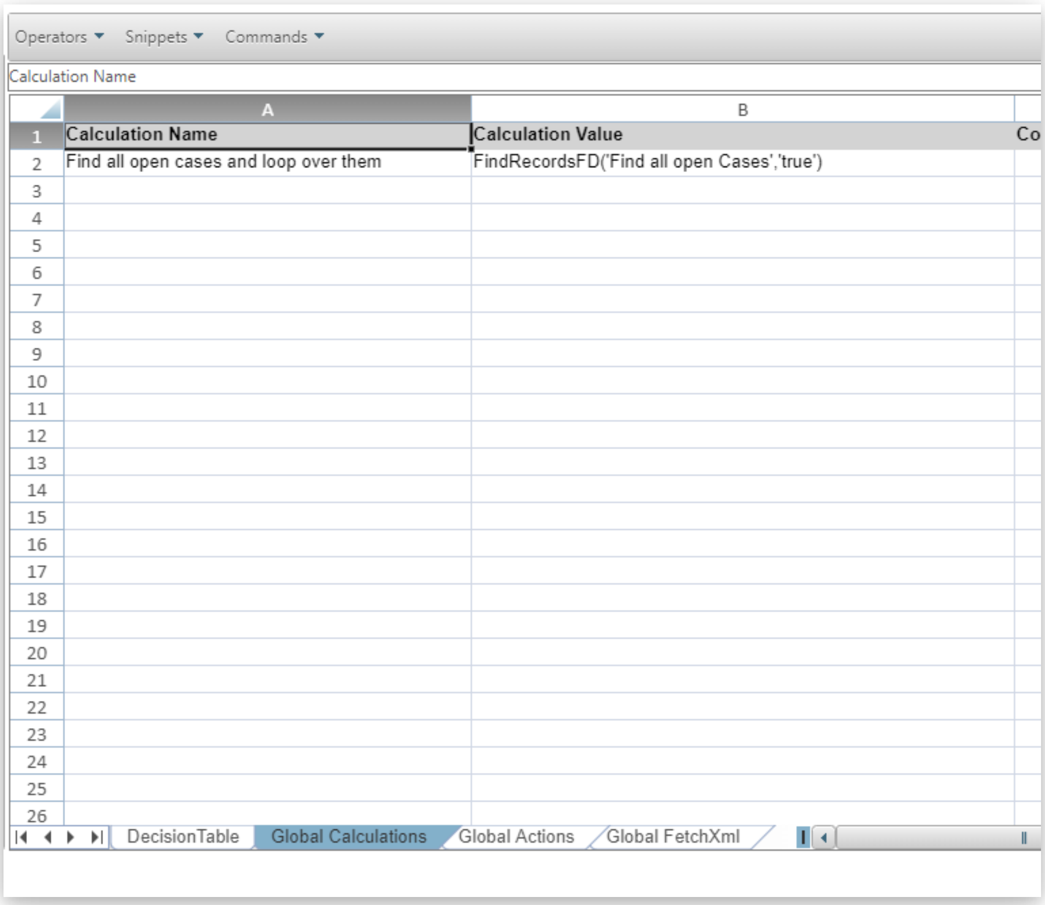
Friendly Name:

Type: Ca

Formula Detail (Query): Find all open Cases

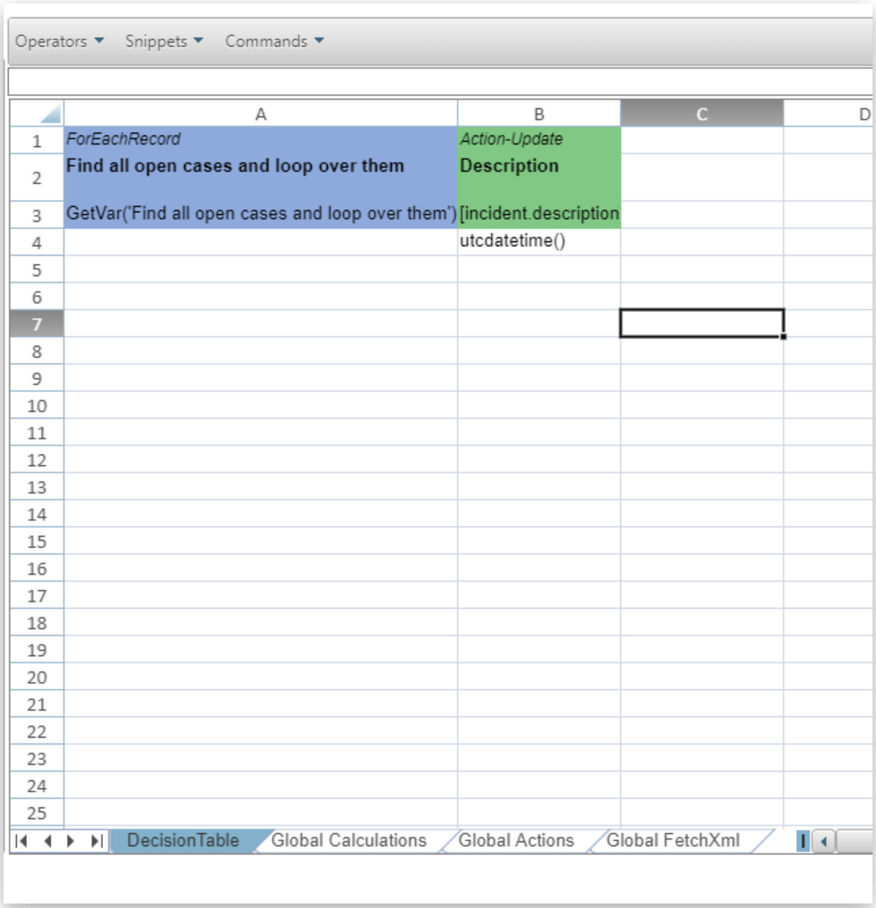
No-Lock: true

Parameters:



Calculation Name

	A	B	Co
1	Calculation Name	Calculation Value	Co
2	Find all open cases and loop over them	FindRecordsFD('Find all open Cases','true')	



	A	B	C	D
1	*ForEachRecord*	*Action-Update*		
2	**Find all open cases and loop over them**	**Description**		
3	GetVar('Find all open cases and loop over them')	[incident.description		
4		utcdatetime()		

DT - How to - 13 - Global Actions sheet

[TOC]

Overview

The **Global Actions** sheet is displayed when using the Decision Tables Advanced Mode. It allows the user to quickly add and edit Actions that can be referenced by Decision Table sheets.

For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions,*** *please read the above articles if you need detailed configuration steps.*

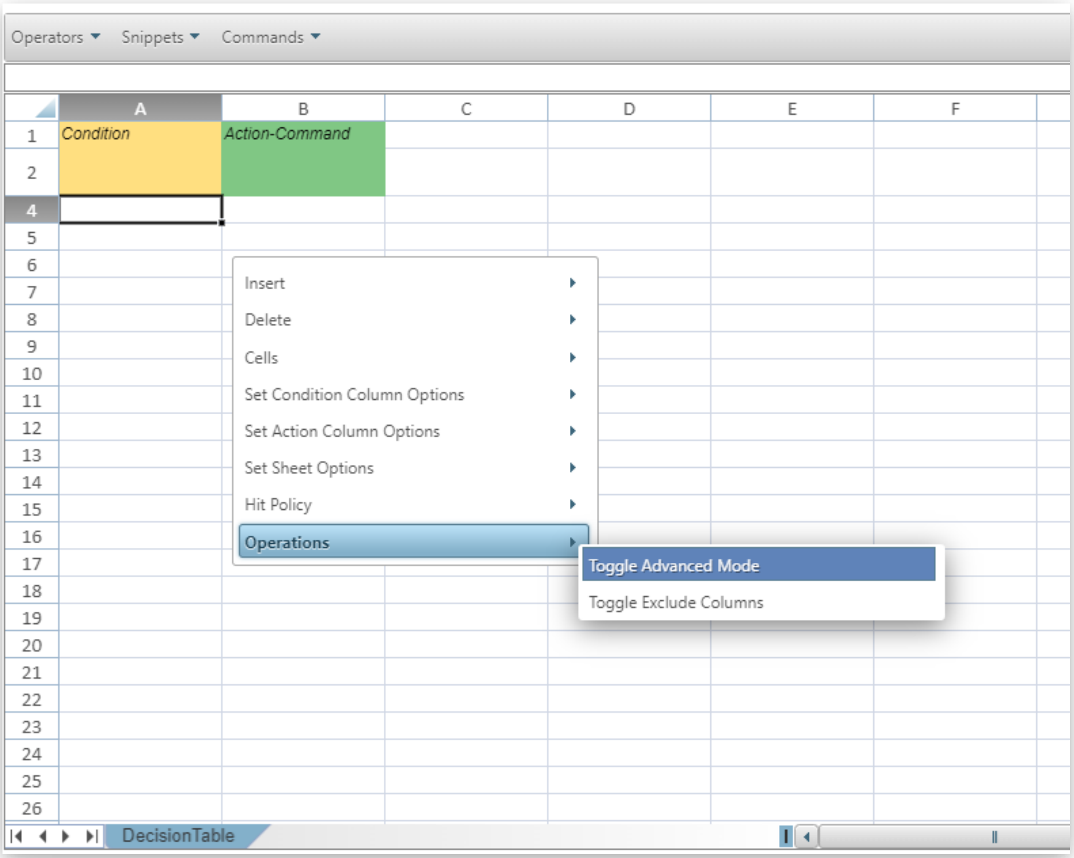
Difference between a **Global Action** and a **Global Calculation**

A **Global Action** will only be executed when referenced elsewhere in the Decision Table.

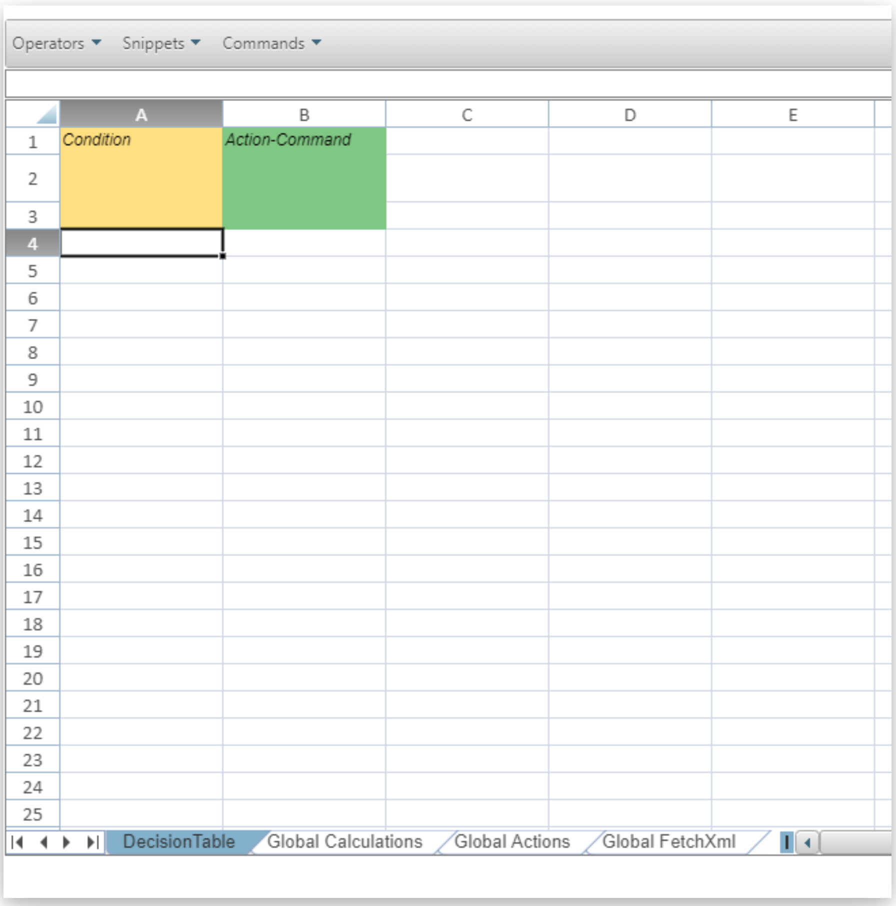
A **Global Calculation** sheet will be executed at the start of the formula regardless of whether or not it is referenced.

Where to find Global Actions

To show the **Global Actions** sheet, open a Decision Table, right-click and select **Operations** > **Toggle Advanced Mode**

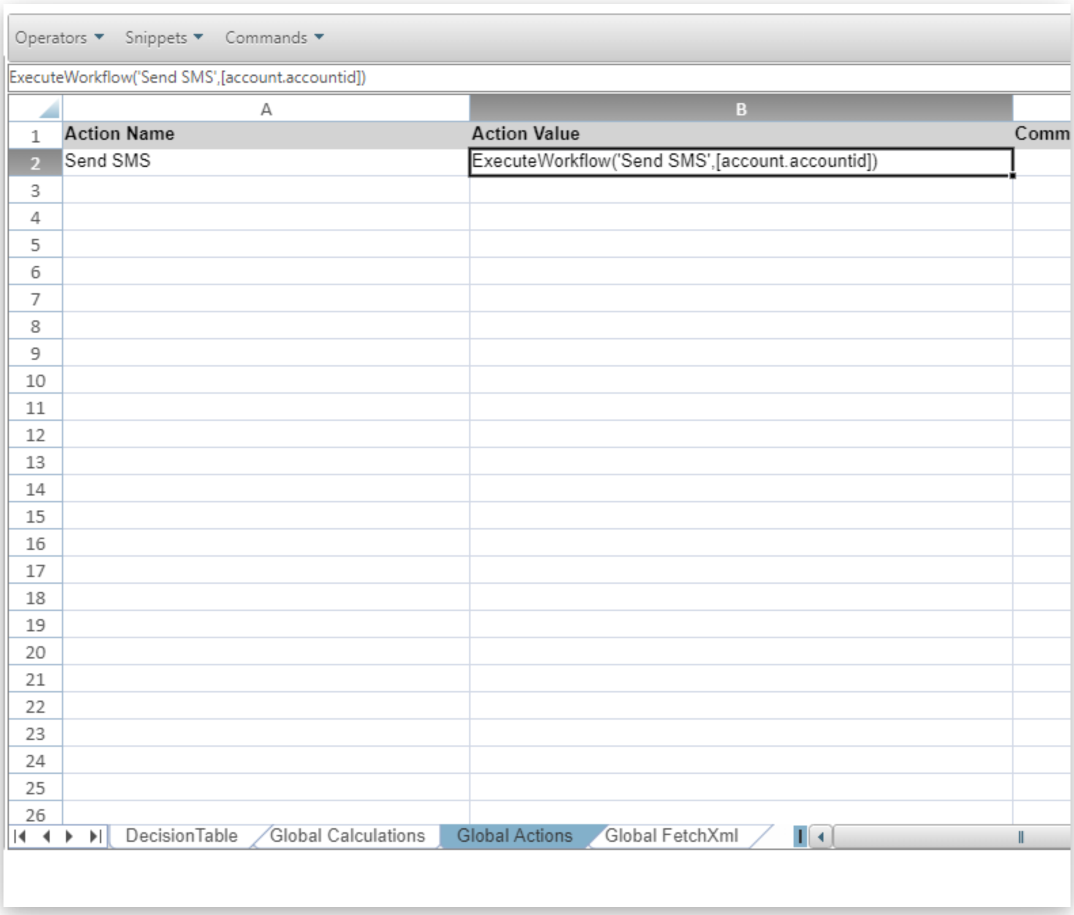


You will then see the **Global Actions** sheet (along with other sheets like **Global Calculations** and **Global FetchXml**)



How to setup Global Actions

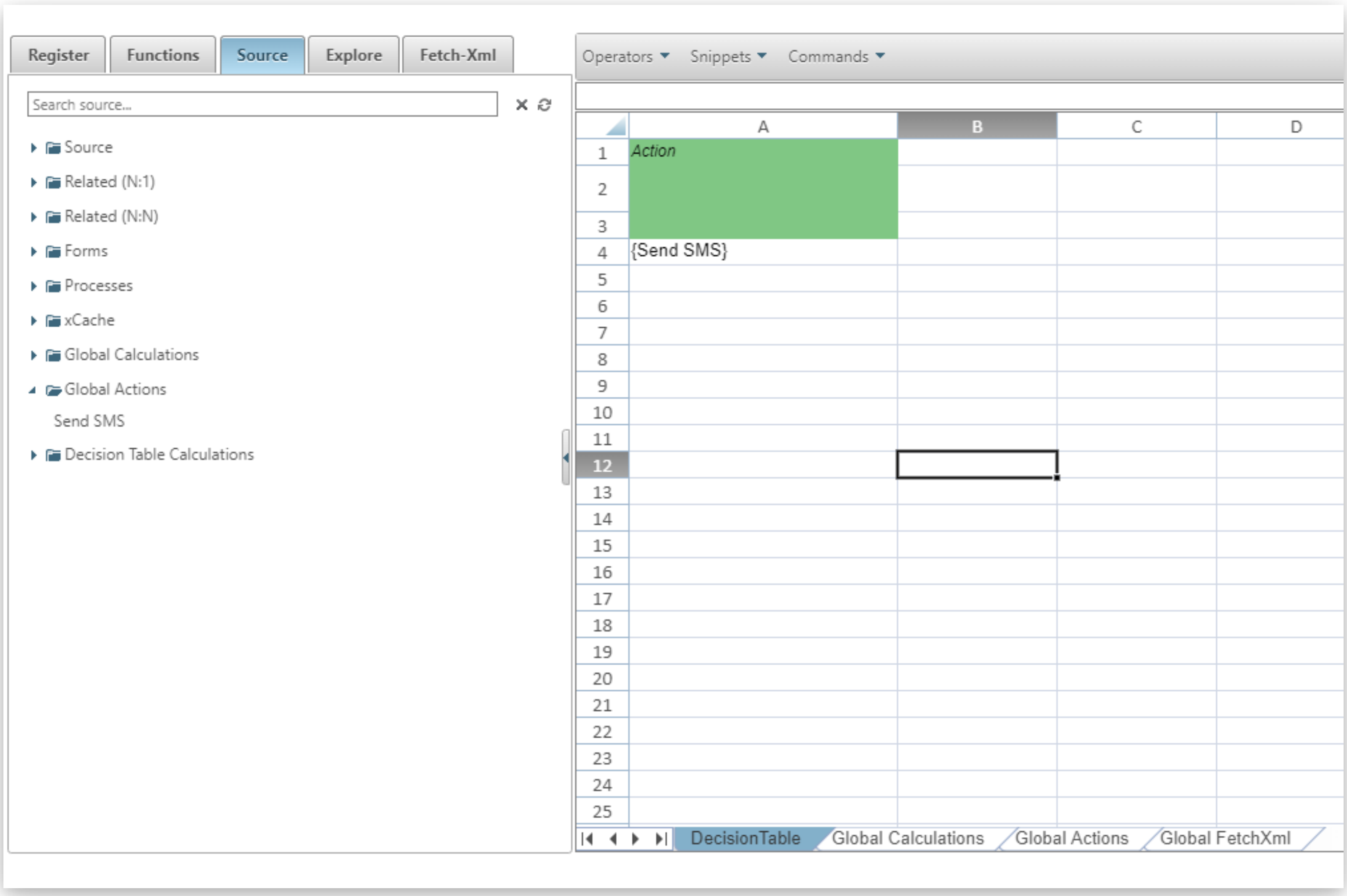
- In a cell in column **A**, give your **Action** a name.
- Then expand the Advanced Editor for the corresponding cell in column **B**
- Enter your Action like below (in our example we wish to execute a workflow called **Send SMS** for an **Account**):



```
ExecuteWorkflow('Send SMS',[account.accountid])
```

	A	B	
1	Action Name	Action Value	Comm
2	Send SMS	ExecuteWorkflow('Send SMS',[account.accountid])	

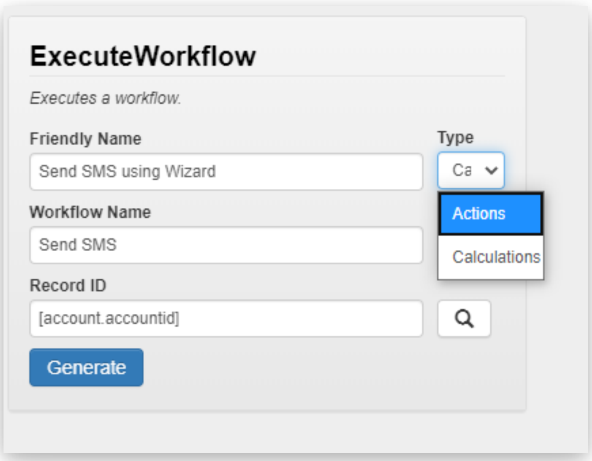
- Collapse the Advanced Editor and click **Save**
- Go to the **Source** tab and select the refresh icon
- Expand **Global Actions**, you will see your **Action**
- Simply click on this node to add a reference to this **Action** within a Decision Table:



	A	B	C	D
1	*Action*			
2				
3				
4	{Send SMS}			

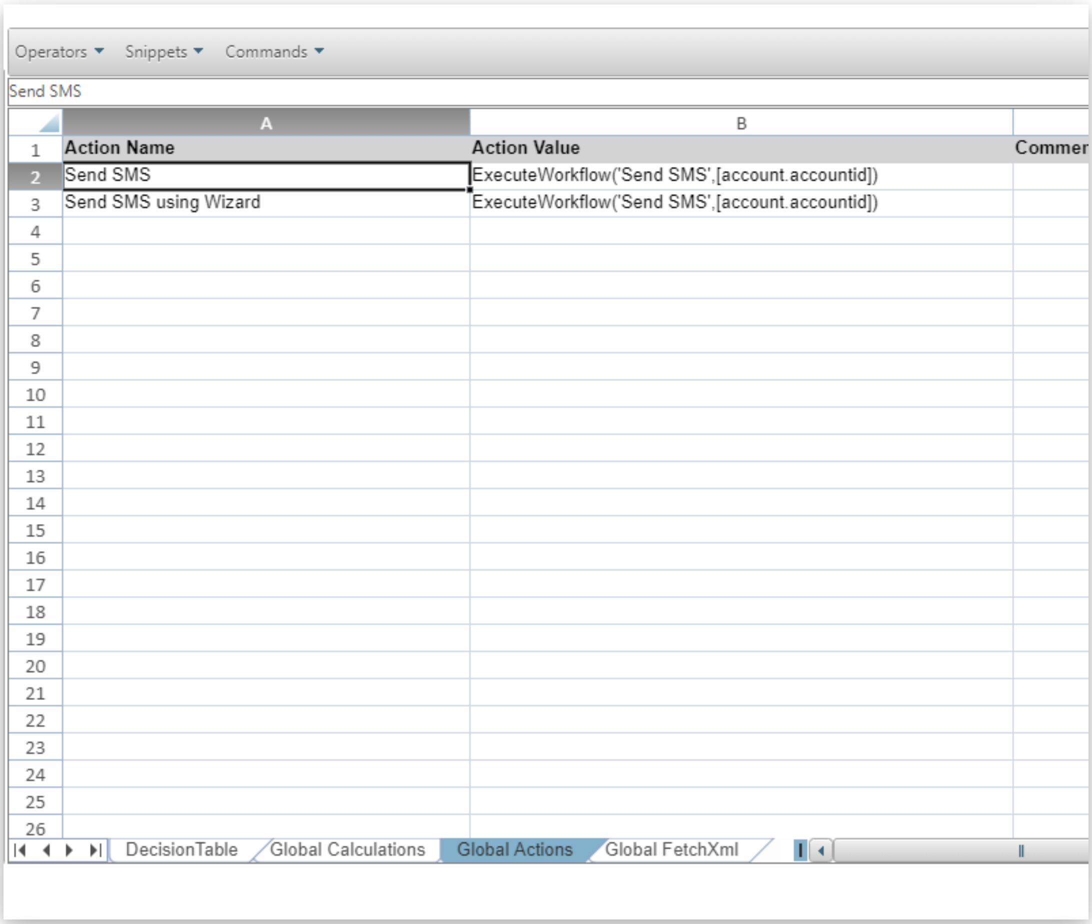
How to setup Global Action using a wizard

When you are using the function wizard in a **Decision Table**, you can give that function a friendly name and choose whether this reference is stored as a **Global Action** or a **Global Calculation**.



Friendly Name: Send SMS using Wizard
Workflow Name: Send SMS
Record ID: [account.accountid]

After selecting **Generate**, this function will be automatically added to the **Global Actions** or **Global Calculations** sheet depending on the Type selected:



	A	B	
1	Action Name	Action Value	Commer
2	Send SMS	ExecuteWorkflow('Send SMS',[account.accountid])	
3	Send SMS using Wizard	ExecuteWorkflow('Send SMS',[account.accountid])	

DT - How to - 14 - Global Calculation sheet

[TOC]

Overview

The **Global Calculation** sheet is displayed when using the Decision Tables Advanced Mode. It allows the user to quickly add and edit Calculations that can be referenced by Decision Table sheets.

For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions****, please read the above articles if you need detailed configuration steps.*

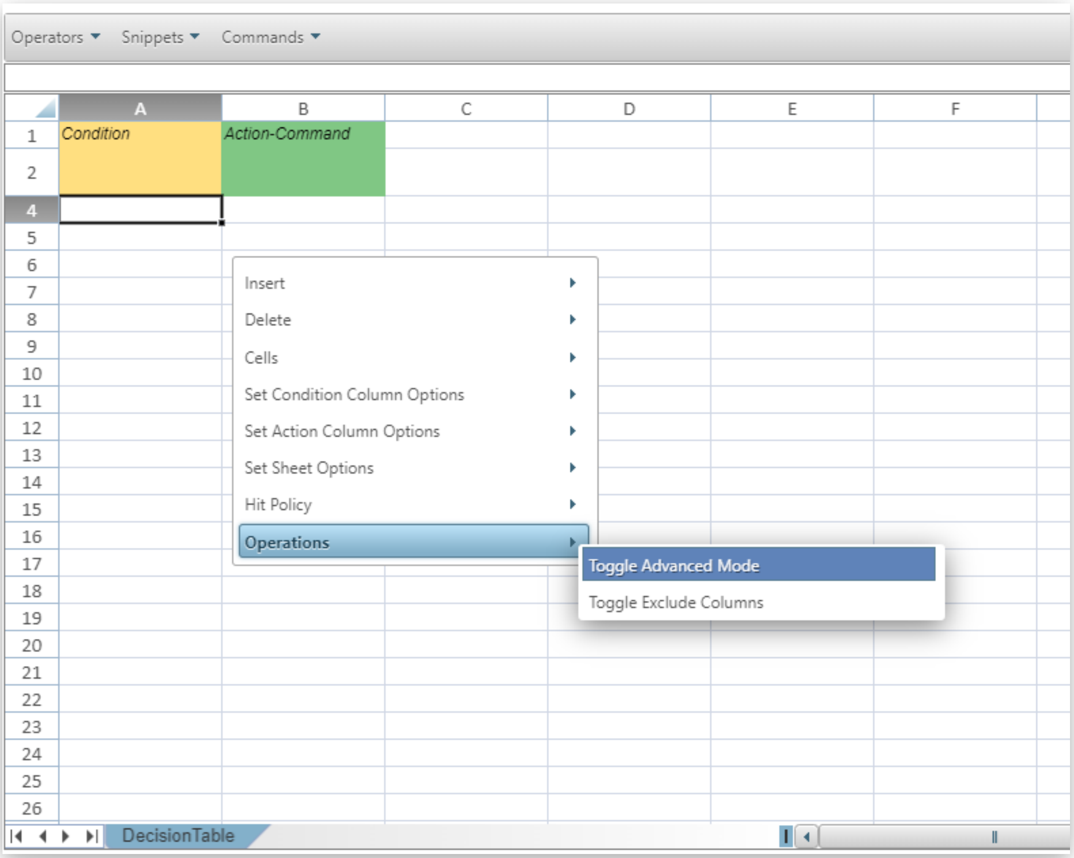
Difference between a **Global Action** and a **Global Calculation**

A **Global Action** will only be executed when referenced elsewhere in the Decision Table.

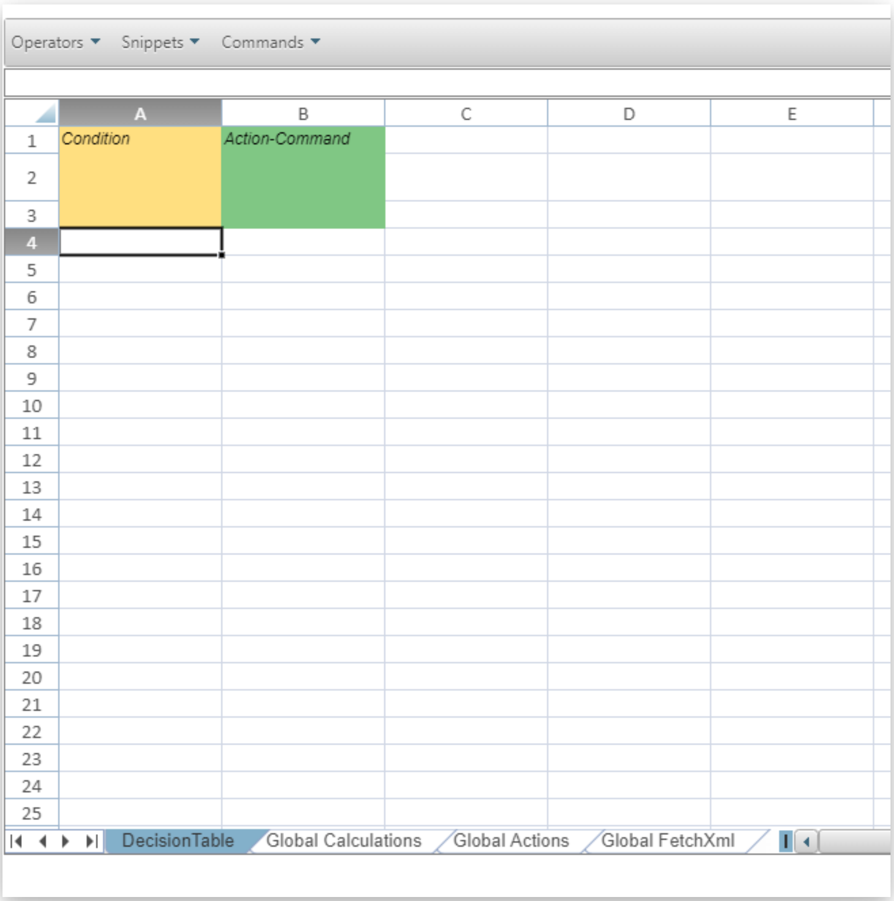
A **Global Calculation** sheet will be executed at the start of the formula regardless of whether or not it is referenced.

Where to find Global Calculations

To show the **Global Calculations** sheet, open a Decision Table, right-click and select **Operations** > **Toggle Advanced Mode**

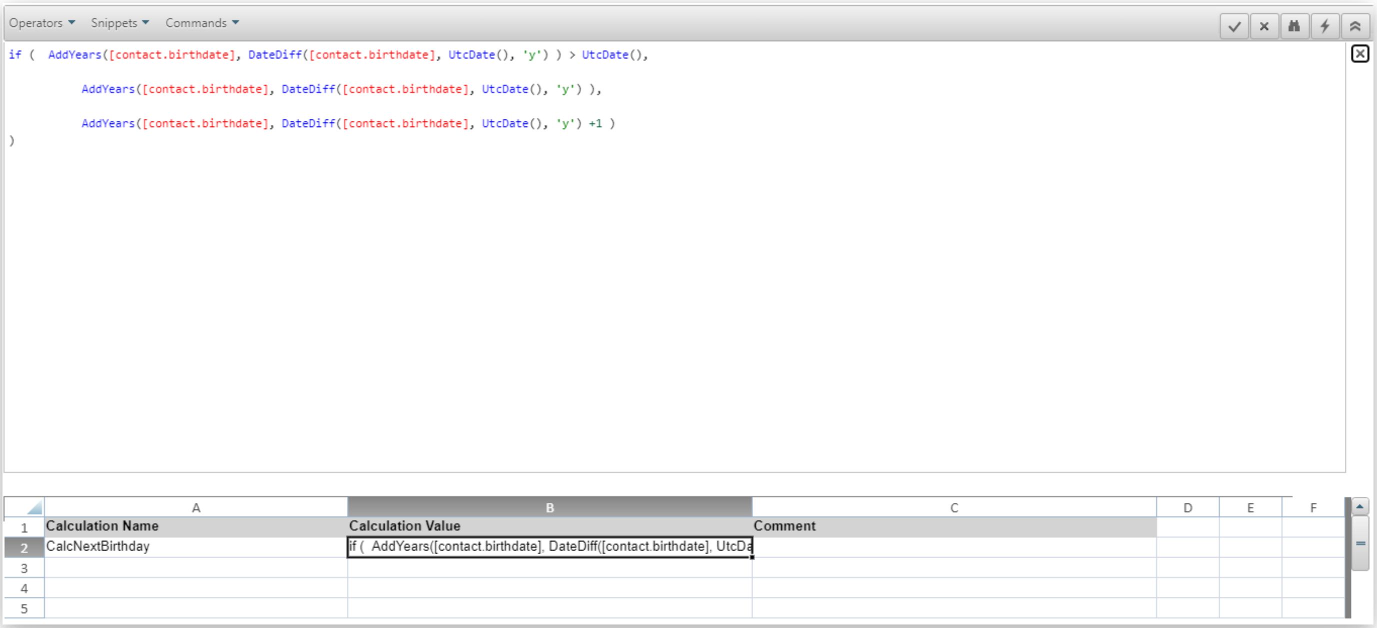


You will then see the **Global Calculations** sheet (along with other sheets like **Global Actions** and **Global FetchXml**)



How to setup Global Calculations

- In a cell in column **A**, give your **Calculation** a name.
- Then expand the Advanced Editor for the corresponding cell in column **B**
- Enter in your Calculation like below (in our example we are determining the date for someone's next birthday based on the current date)

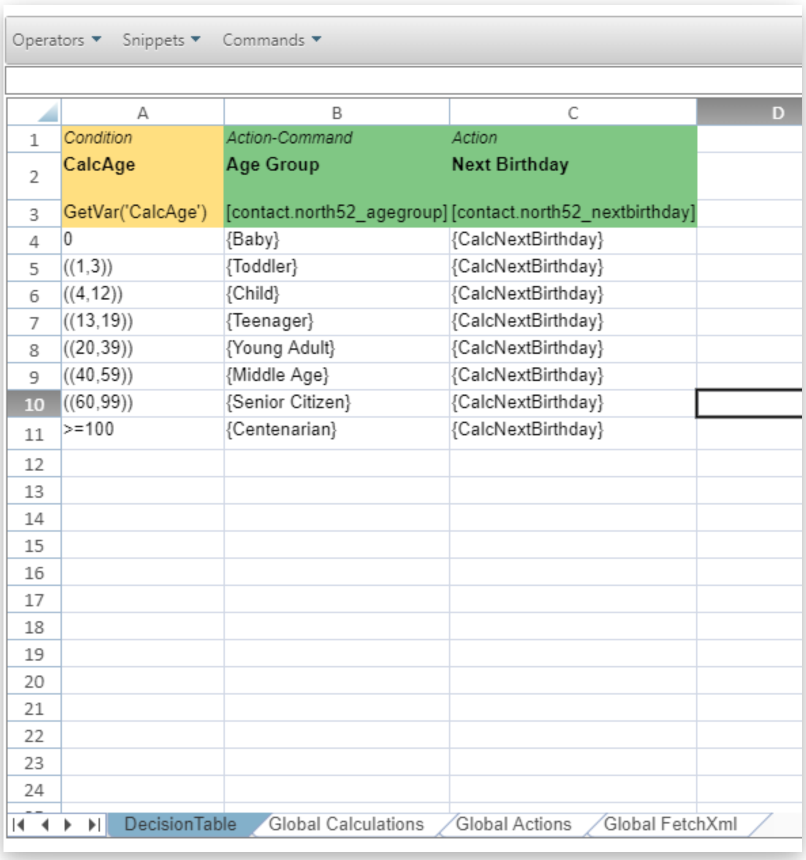


```
if (  AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') ) > UtcDate(),

          AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') ),

          AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') +1 )
)
```

	A	B	C	D	E	F
1	Calculation Name	Calculation Value	Comment			
2	CalcNextBirthday	if (AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDa				
3						
4						
5						

- Collapse the Advanced Editor and click **Save**
- Add any other Calculations you may need
- Go to the **Source** tab and select the refresh icon
- Expand **Global Calculation**, you will see your new **Calculation**

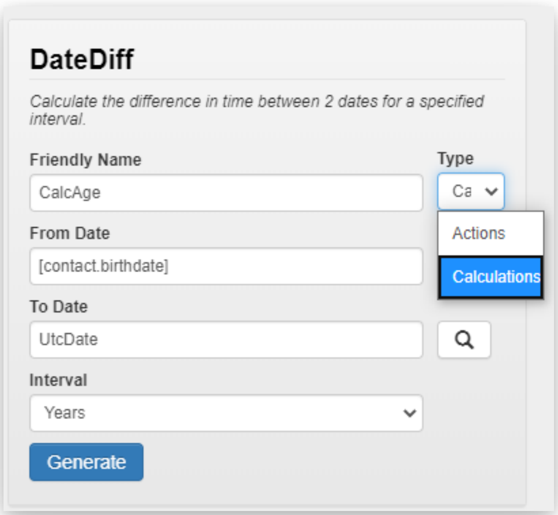
- Now you will be able to reference this **Calculation** in any sheet



	A	B	C	D
1	*Condition*	*Action-Command*	*Action*	
2	**CalcAge**	**Age Group**	**Next Birthday**	
3	GetVar('CalcAge')	[contact.north52_agegroup]	[contact.north52_nextbirthday]	
4	0	{Baby}	{CalcNextBirthday}	
5	((1,3))	{Toddler}	{CalcNextBirthday}	
6	((4,12))	{Child}	{CalcNextBirthday}	
7	((13,19))	{Teenager}	{CalcNextBirthday}	
8	((20,39))	{Young Adult}	{CalcNextBirthday}	
9	((40,59))	{Middle Age}	{CalcNextBirthday}	
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}	
11	>=100	{Centenarian}	{CalcNextBirthday}	

How to setup Global **Calculation** using a wizard

When you are using the function wizard in a **Decision Table**, you can give that function a friendly name and choose whether this reference is stored as a **Global Action** or a **Global Calculation**.



DateDiff

Calculate the difference in time between 2 dates for a specified interval.

Friendly Name	CalcAge
Type	Ca (Actions / Calculations)
From Date	[contact.birthdate]
To Date	UtcDate
Interval	Years

After selecting **Generate**, this function will be automatically added to the **Global Actions** or **Global Calculations** sheet depending on the Type selected.

DT - How to - 15 - Source tab

[TOC]

Overview

The **Source** tab allows users to reference fields (attributes) on a record and fields on records related to the primary record.

It also allows users to reference environmental values such as **Processes**, **xCache** records, **Global Calculations**, **Global Actions** and **Decision Table Calculations**.

The Source tab can be used in conjunction with **Classic** or **Decision Tables**, in this article we will focus on its use in **Decision Tables**.

For this article it is assumed that you have at least basic familiarity with **Decision Tables** and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

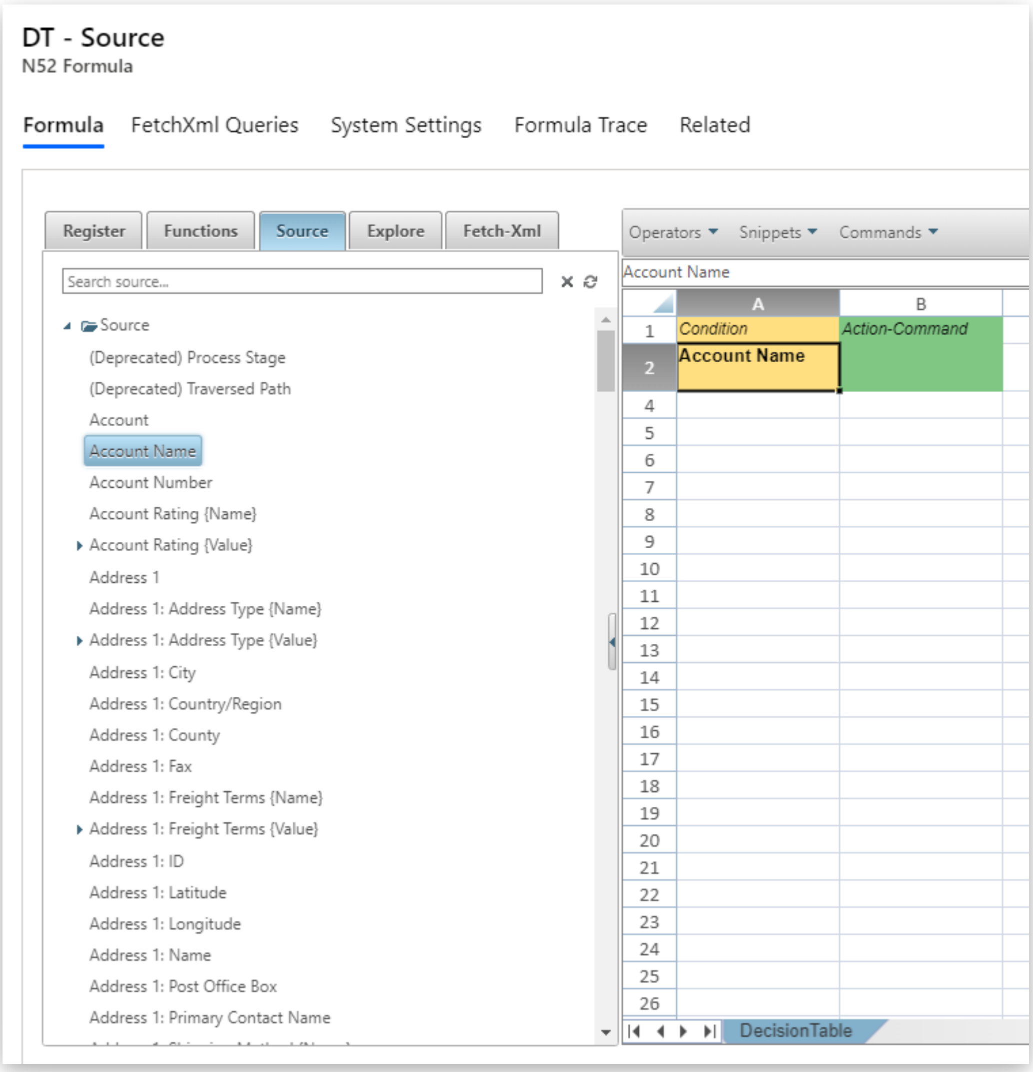
Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions,*** *please read the above articles if you need detailed configuration steps.*

The following parts of the Source tab are explained below:

- Source
- Related (N:1)
- Related (N:N)
- Forms
- Processes
- xCache
- Global Calculations
- Global Actions
- Decision Table Calculations

Source

The **Source** tree shows all the fields on an record. Clicking a field will place a reference to the field in the selected cell of the Decision Table:

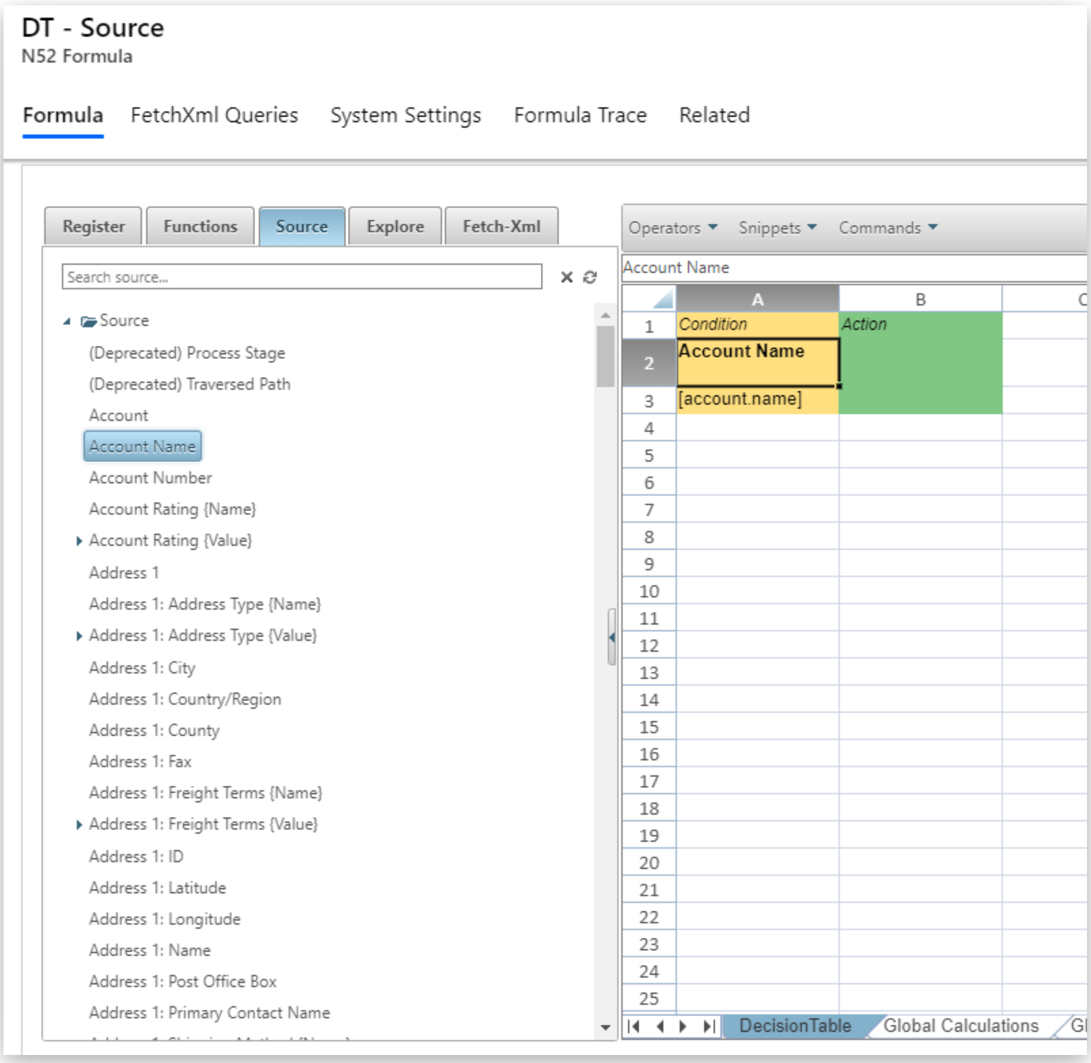

DT - Source
N52 Formula

Source
- (Deprecated) Process Stage
- (Deprecated) Traversed Path
- Account
- Account Name
- Account Number
- Account Rating {Name}
- Account Rating {Value}
- Address 1
- Address 1: Address Type {Name}
- Address 1: Address Type {Value}
- Address 1: City
- Address 1: Country/Region
- Address 1: County
- Address 1: Fax
- Address 1: Freight Terms {Name}
- Address 1: Freight Terms {Value}
- Address 1: ID
- Address 1: Latitude
- Address 1: Longitude
- Address 1: Name
- Address 1: Post Office Box
- Address 1: Primary Contact Name

Account Name

	A	B
1	Condition	Action-Command
2	Account Name	

Right-click and select **Operations** > **Toggle Advanced Mode** to reveal Row 3 as shown below:


DT - Source

N52 Formula

Formula FetchXml Queries System Settings Formula Trace Related

Register Functions Source Explore Fetch-Xml

Operators Snippets Commands

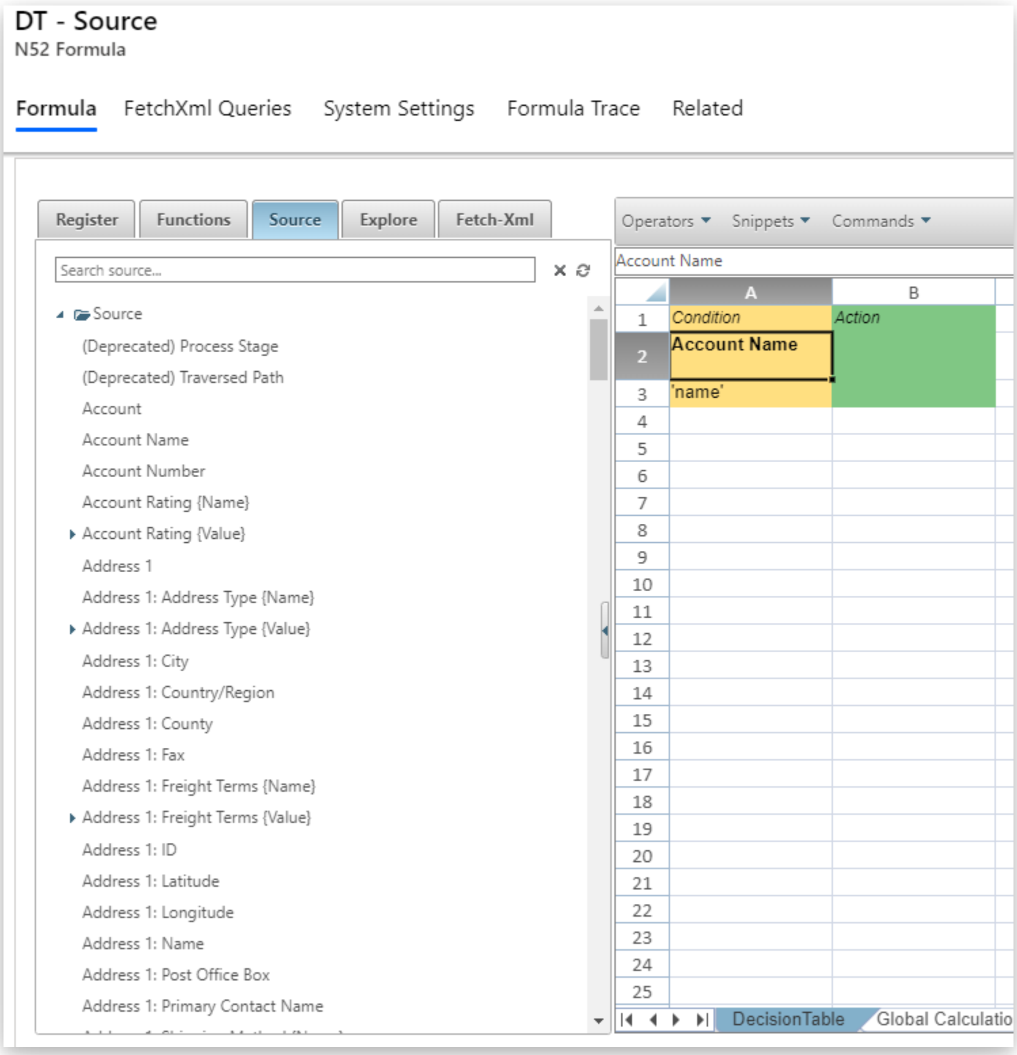
Account Name

Source
- (Deprecated) Process Stage
- (Deprecated) Traversed Path
- Account
- Account Name
- Account Number
- Account Rating {Name}
- Account Rating {Value}
- Address 1
- Address 1: Address Type {Name}
- Address 1: Address Type {Value}
- Address 1: City
- Address 1: Country/Region
- Address 1: County
- Address 1: Fax
- Address 1: Freight Terms {Name}
- Address 1: Freight Terms {Value}
- Address 1: ID
- Address 1: Latitude
- Address 1: Longitude
- Address 1: Name
- Address 1: Post Office Box
- Address 1: Primary Contact Name

	A	B
1	*Condition*	*Action*
2	**Account Name**	
3	[account.name]	

DecisionTable Global Calculations

If the field's logical name is needed instead, **Ctrl-click** the field name in the tree to place the field's logical name into the cell:


DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Source
- (Deprecated) Process Stage
- (Deprecated) Traversed Path
- Account
- Account Name
- Account Number
- Account Rating {Name}
- Account Rating {Value}
- Address 1
- Address 1: Address Type {Name}
- Address 1: Address Type {Value}
- Address 1: City
- Address 1: Country/Region
- Address 1: County
- Address 1: Fax
- Address 1: Freight Terms {Name}
- Address 1: Freight Terms {Value}
- Address 1: ID
- Address 1: Latitude
- Address 1: Longitude
- Address 1: Name
- Address 1: Post Office Box
- Address 1: Primary Contact Name

Account Name

	A	B
1	Condition	Action
2	Account Name	
3	'name'	

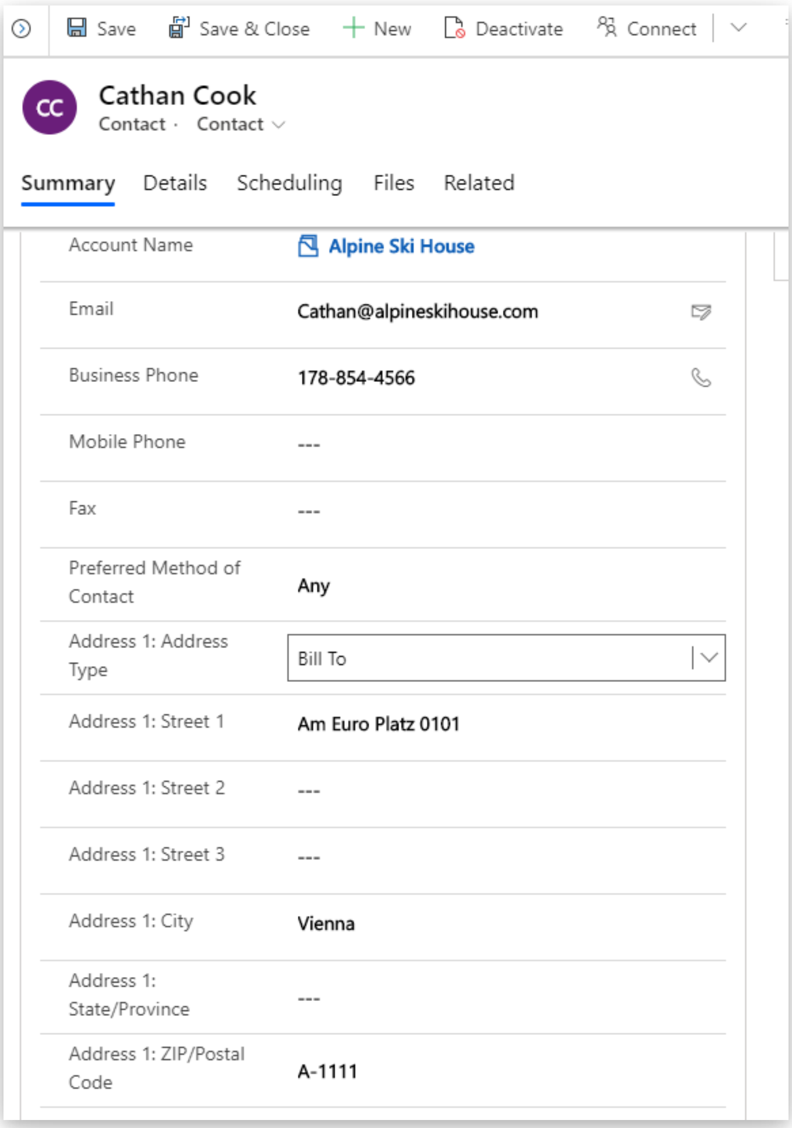
Option Set Field: Name and Value

Option Set fields can have different iterations depending on their type: **Value** and **Name**

- The **Value** iteration references the value stored in the database (e.g. a number for an option set item, a GUID for a lookup). It is the iteration used when updating an record or when interacting with the system
- The **Name** iteration references the user friendly equivalent of value displayed to the end user

Example:

A **Contact** record with the **Bill To** option selected for the option set field **Address 1: Address Type**:

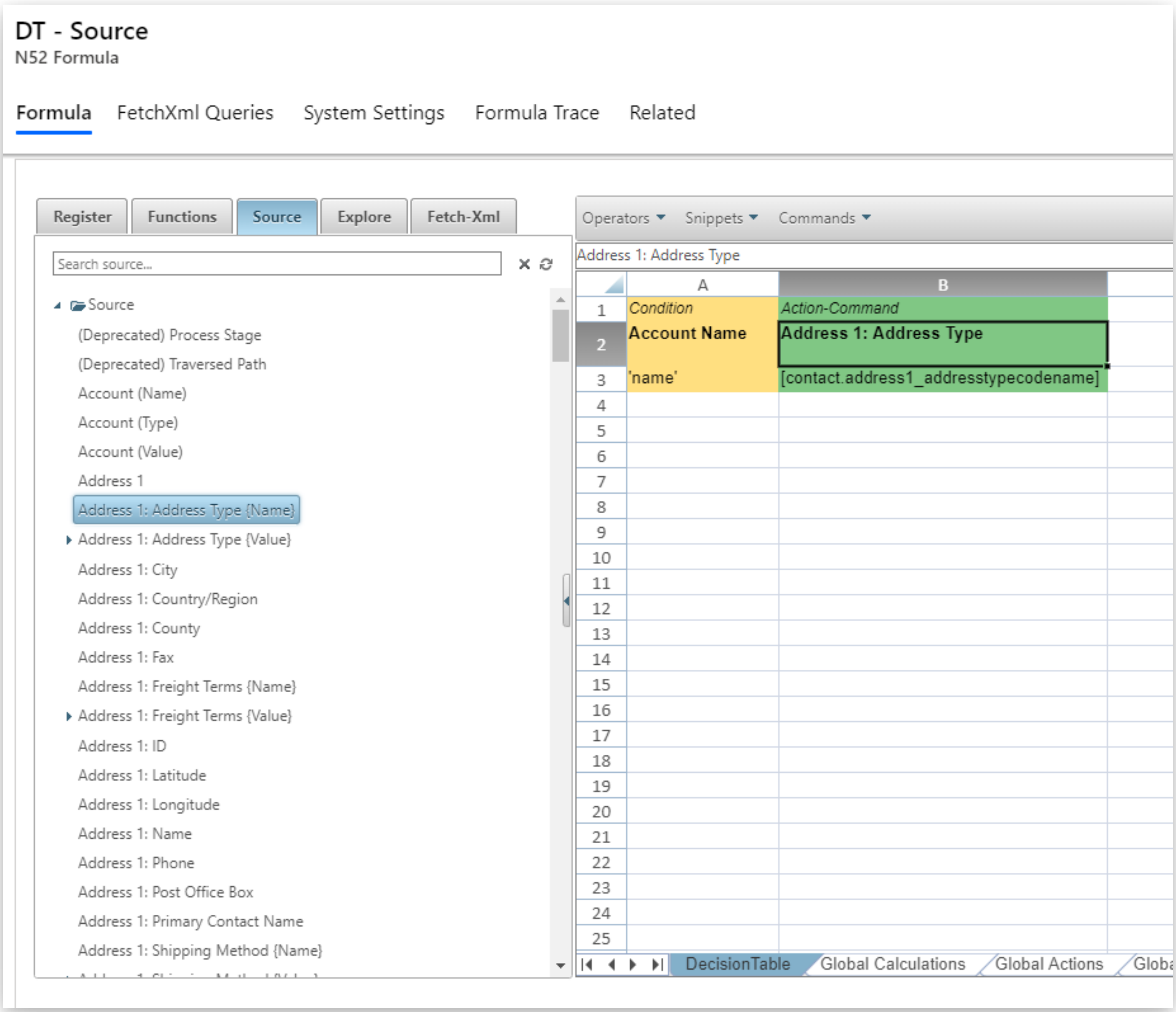

Save | Save & Close | New | Deactivate | Connect

Cathan Cook
Contact · Contact

Summary | Details | Scheduling | Files | Related

Account Name	Alpine Ski House
Email	Cathan@alpineskihouse.com
Business Phone	178-854-4566
Mobile Phone	---
Fax	---
Preferred Method of Contact	Any
Address 1: Address Type	Bill To
Address 1: Street 1	Am Euro Platz 0101
Address 1: Street 2	---
Address 1: Street 3	---
Address 1: City	Vienna
Address 1: State/Province	---
Address 1: ZIP/Postal Code	A-1111

Selecting **Address 1: Address Type {Name}** will reference the string **Bill To**


DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Register | Functions | Source | Explore | Fetch-Xml

Operators ▾ Snippets ▾ Commands ▾

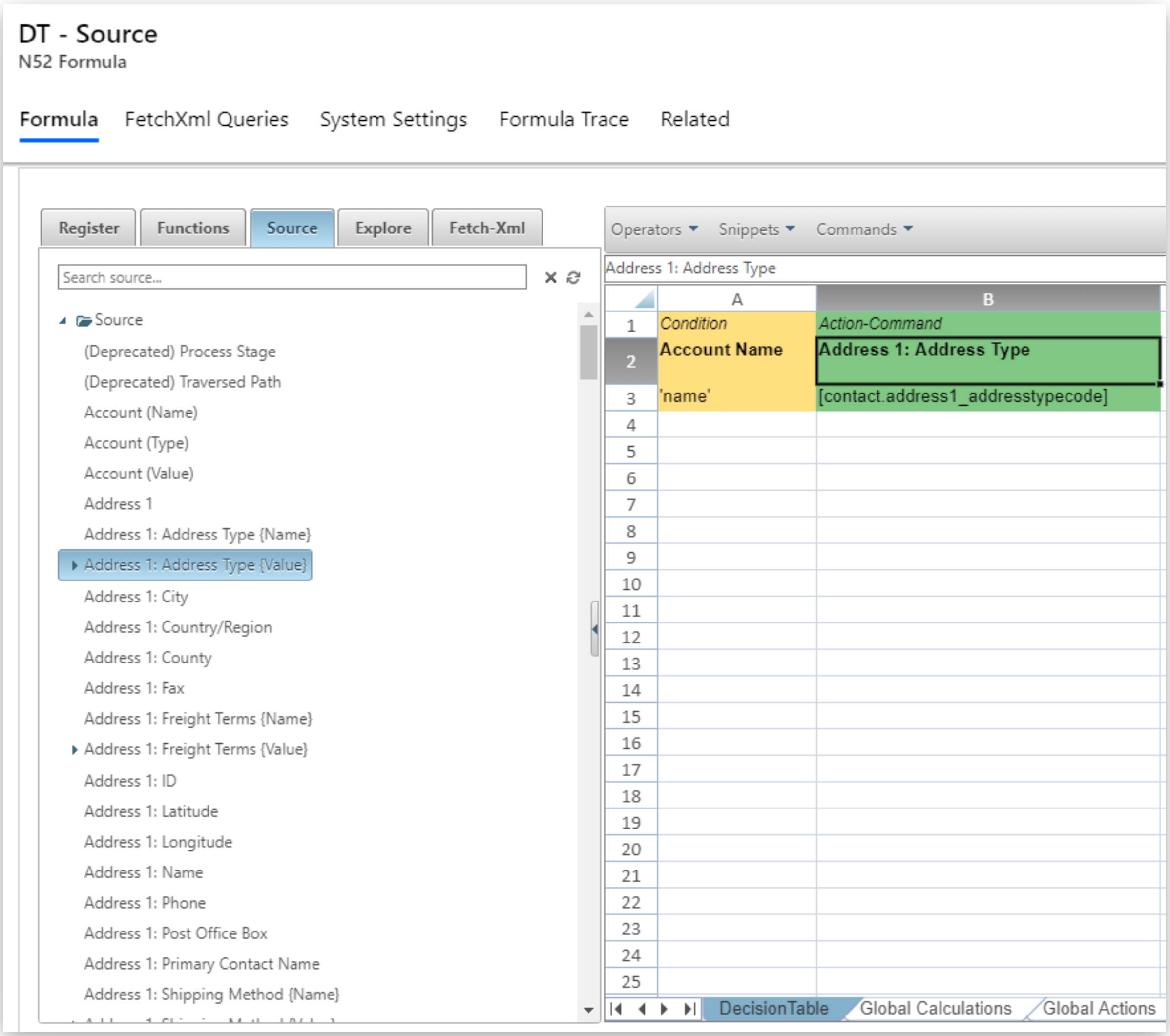
Address 1: Address Type

Source
- (Deprecated) Process Stage
- (Deprecated) Traversed Path
- Account (Name)
- Account (Type)
- Account (Value)
- Address 1
- Address 1: Address Type {Name}
- Address 1: Address Type {Value}
- Address 1: City
- Address 1: Country/Region
- Address 1: County
- Address 1: Fax
- Address 1: Freight Terms {Name}
- Address 1: Freight Terms {Value}
- Address 1: ID
- Address 1: Latitude
- Address 1: Longitude
- Address 1: Name
- Address 1: Phone
- Address 1: Post Office Box
- Address 1: Primary Contact Name
- Address 1: Shipping Method {Name}

	A	B
1	*Condition*	*Action-Command*
2	**Account Name**	**Address 1: Address Type**
3	'name'	[contact.address1_addresstypecodename]

DecisionTable | Global Calculations | Global Actions

However selecting **Address 1: Address Type {Value}** will reference the **Bill To's value** which is **1**


DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Source
- (Deprecated) Process Stage
- (Deprecated) Traversed Path
- Account (Name)
- Account (Type)
- Account (Value)
- Address 1
- Address 1: Address Type {Name}
- Address 1: Address Type {Value}
- Address 1: City
- Address 1: Country/Region
- Address 1: County
- Address 1: Fax
- Address 1: Freight Terms {Name}
- Address 1: Freight Terms {Value}
- Address 1: ID
- Address 1: Latitude
- Address 1: Longitude
- Address 1: Name
- Address 1: Phone
- Address 1: Post Office Box
- Address 1: Primary Contact Name
- Address 1: Shipping Method {Name}

Address 1: Address Type

	A	B
1	Condition	Action-Command
2	Account Name	Address 1: Address Type
3	'name'	[contact.address1_addresstypecode]

DecisionTable | Global Calculations | Global Actions

Note: Most times you will want to reference the Option set Value - it is this value that needs to be updated/referenced when interacting with records.

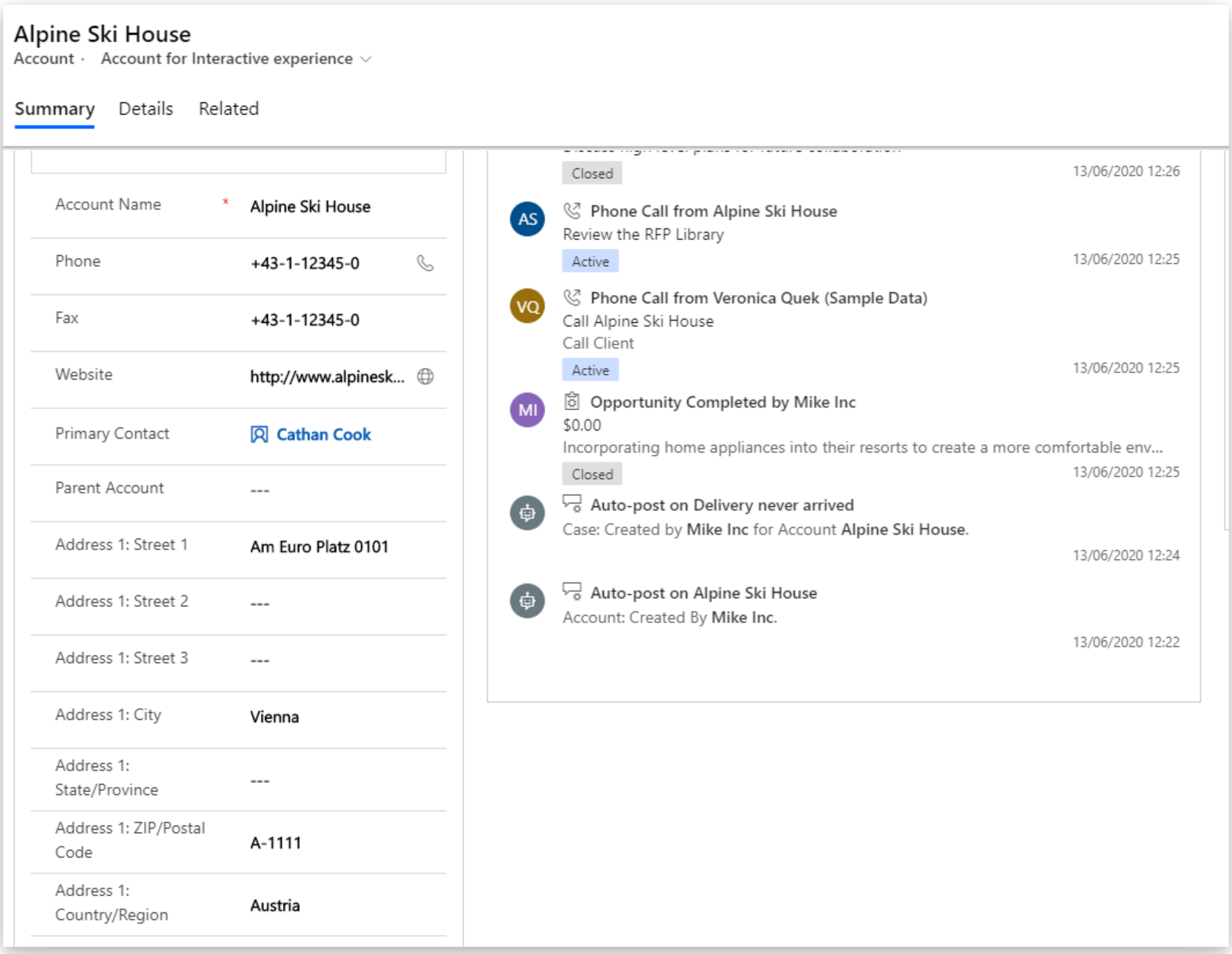
Lookup Field: Name, Value or Type

Lookup fields can have different iterations depending on their type: **Value, Name** and **Type**

- The **Value** iteration references the value stored in the database (e.g. a number for an option set item, a GUID for a lookup). It is the iteration used when updating an record or when interacting with the system
- The **Name** iteration references the user friendly equivalent of value displayed to the end user
- The **Type** iteration references the entity type, e.g. contact, account, lead, etc

Example:

The **Primary Contact** field on the **Account** entity:



Alpine Ski House
Account · Account for Interactive experience

Summary Details Related

Account Name *	Alpine Ski House
Phone	+43-1-12345-0
Fax	+43-1-12345-0
Website	http://www.alpinesk...
Primary Contact	Cathan Cook
Parent Account	---
Address 1: Street 1	Am Euro Platz 0101
Address 1: Street 2	---
Address 1: Street 3	---
Address 1: City	Vienna
Address 1: State/Province	---
Address 1: ZIP/Postal Code	A-1111
Address 1: Country/Region	Austria

Closed 13/06/2020 12:26

AS Phone Call from Alpine Ski House
Review the RFP Library
Active 13/06/2020 12:25

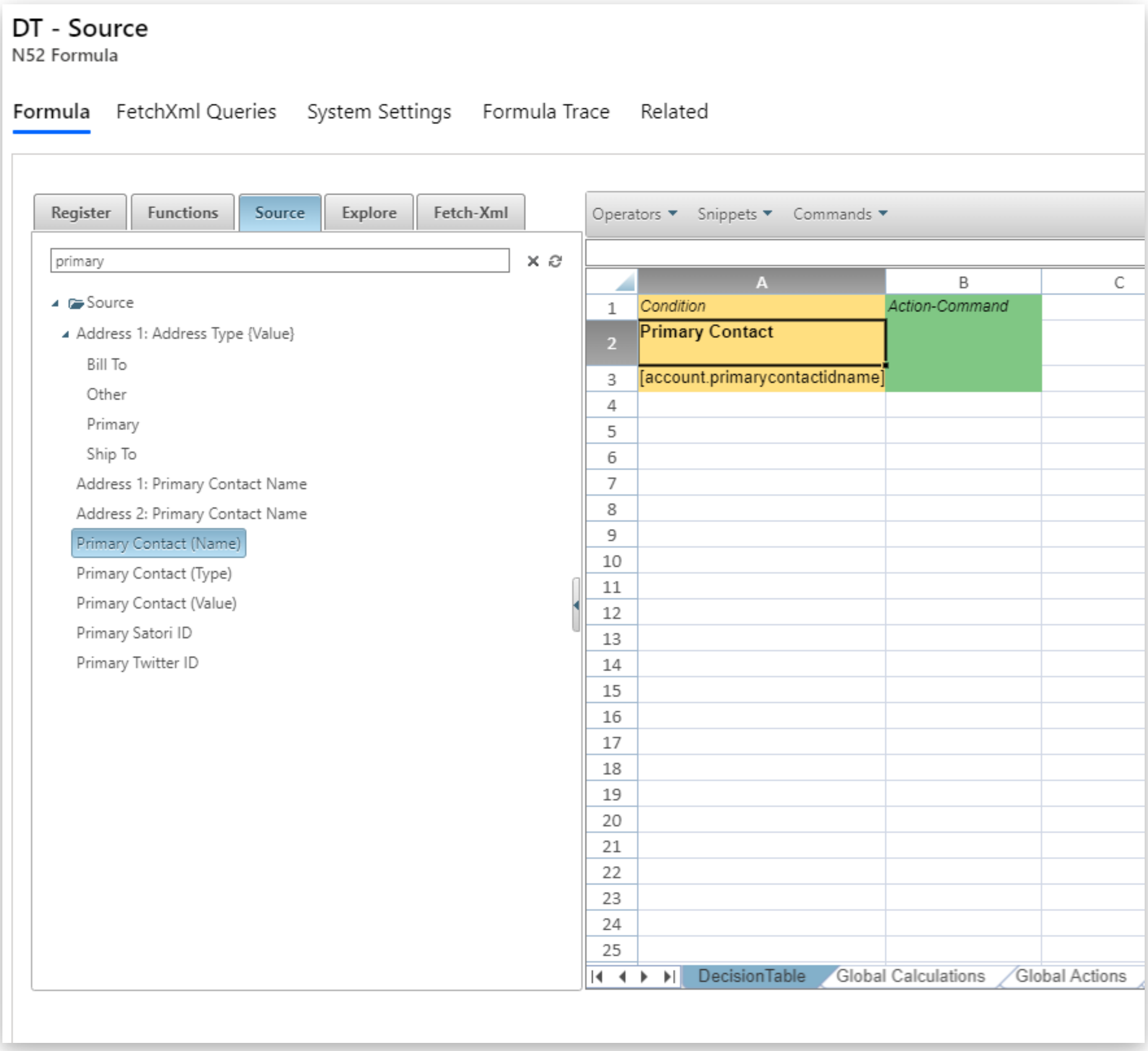
VQ Phone Call from Veronica Quek (Sample Data)
Call Alpine Ski House
Call Client
Active 13/06/2020 12:25

MI Opportunity Completed by Mike Inc
$0.00
Incorporating home appliances into their resorts to create a more comfortable env...
Closed 13/06/2020 12:25

Auto-post on Delivery never arrived
Case: Created by **Mike Inc** for Account **Alpine Ski House**.
13/06/2020 12:24

Auto-post on Alpine Ski House
Account: Created By **Mike Inc**.
13/06/2020 12:22

Selecting **Primary Contact {Name}** will reference **Cathan Cooke**



DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Register | Functions | Source | Explore | Fetch-Xml

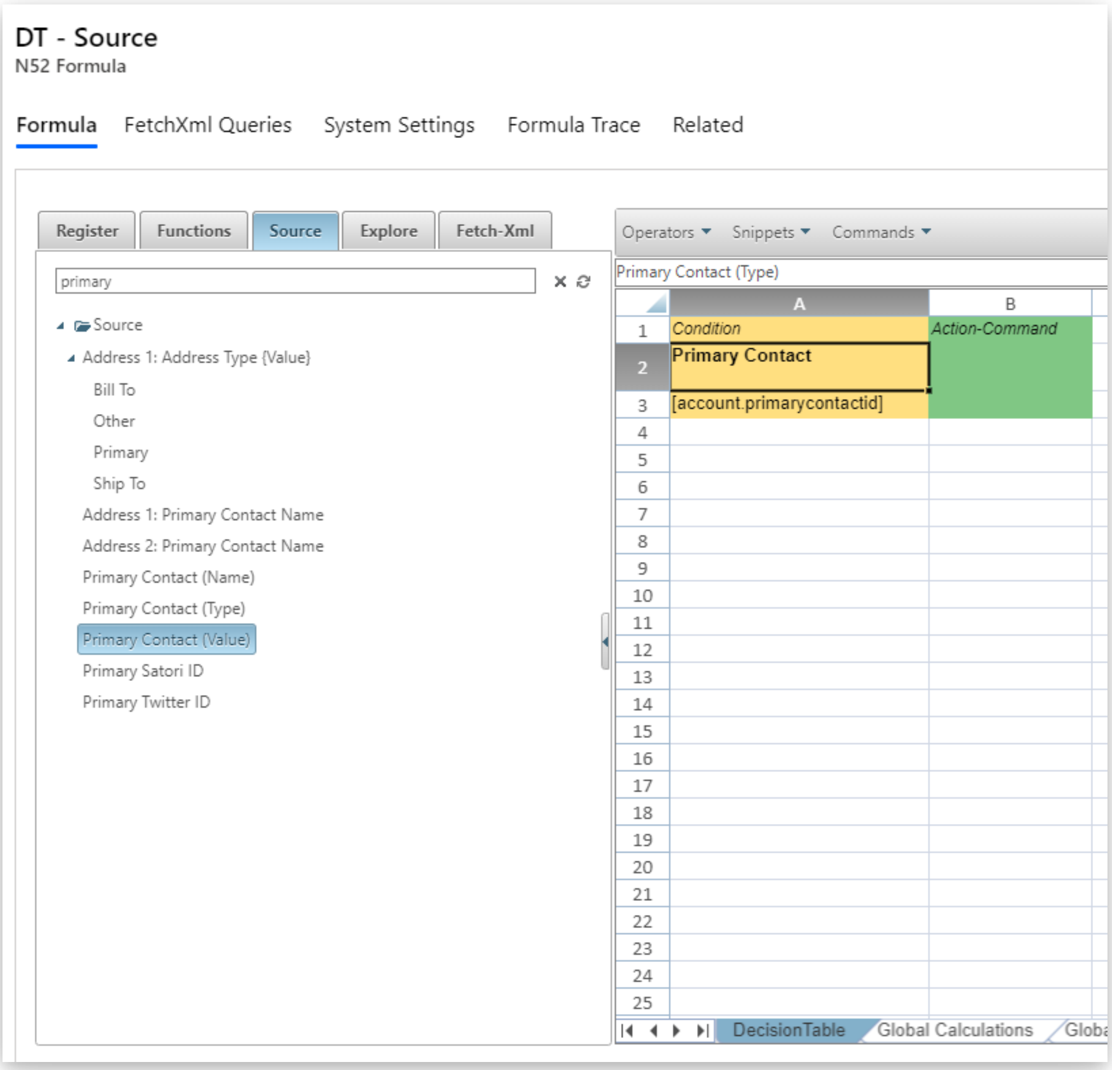
primary

- Source
  - Address 1: Address Type {Value}
    - Bill To
    - Other
    - Primary
    - Ship To
  - Address 1: Primary Contact Name
  - Address 2: Primary Contact Name
  - Primary Contact (Name)
  - Primary Contact (Type)
  - Primary Contact (Value)
  - Primary Satori ID
  - Primary Twitter ID

	A	B	C
1	Condition	Action-Command	
2	Primary Contact		
3	[account.primarycontactidname]		

DecisionTable | Global Calculations | Global Actions

Selecting **Primary Contact {Value}** will reference the GUID of the **Contact Cathan Cooke:**



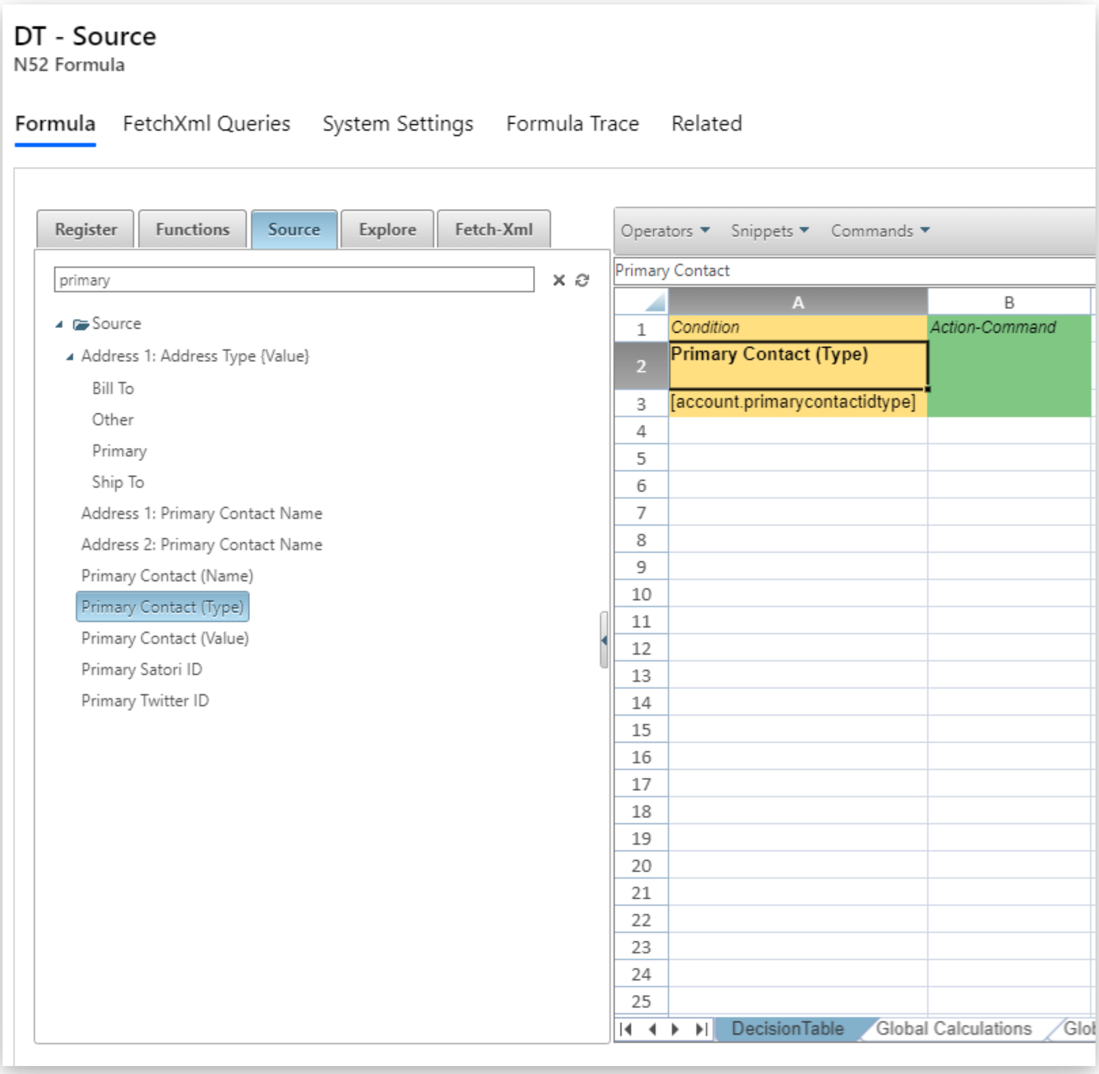
primary

- Source
  - Address 1: Address Type {Value}
    - Bill To
    - Other
    - Primary
    - Ship To
  - Address 1: Primary Contact Name
  - Address 2: Primary Contact Name
  - Primary Contact (Name)
  - Primary Contact (Type)
  - Primary Contact (Value)
  - Primary Satori ID
  - Primary Twitter ID

Primary Contact (Type)

	A	B
1	Condition	Action-Command
2	Primary Contact	
3	[account.primarycontactid]	

Selecting the **Primary Contact {Type}** will reference the entity type of the contact being referred to - in this case it will return **contact**:


DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Register | Functions | Source | Explore | Fetch-Xml

primary

- Source
  - Address 1: Address Type (Value)
    - Bill To
    - Other
    - Primary
    - Ship To
  - Address 1: Primary Contact Name
  - Address 2: Primary Contact Name
  - Primary Contact (Name)
  - Primary Contact (Type)
  - Primary Contact (Value)
  - Primary Satori ID
  - Primary Twitter ID

Operators | Snippets | Commands

Primary Contact

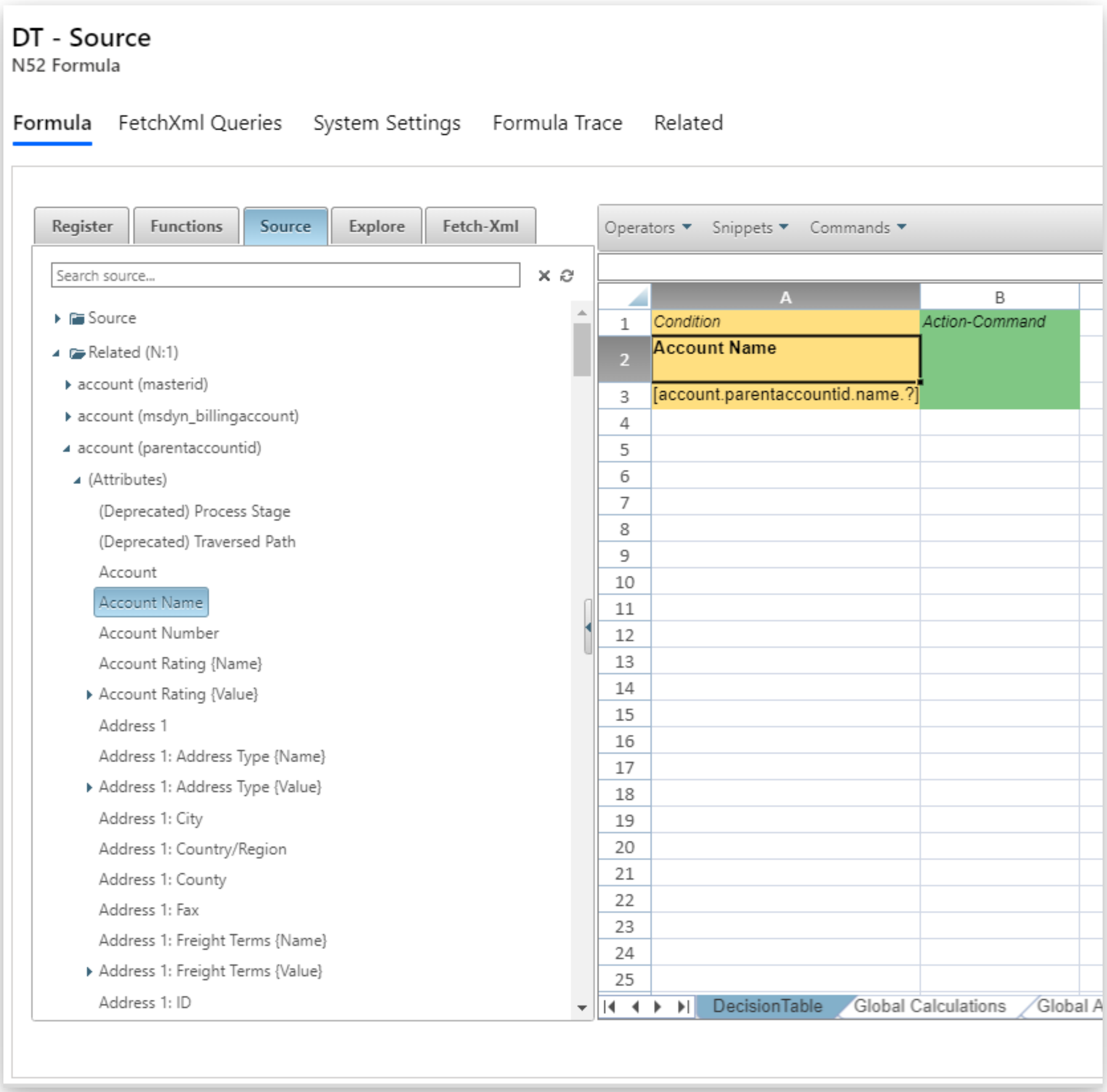
	A	B
1	*Condition*	*Action-Command*
2	**Primary Contact (Type)**	
3	[account.primarycontactidtype]	

DecisionTable | Global Calculations

***Note**: Most times you will want to reference the lookup Value - it is this value that needs to be updated/referenced when interacting with records.*

Related (N:1)

The **Related (N:1)** tree shows the N:1 related entities of the record. From this tree a you can access the related entities fields via the **Attributes** branch:


DT - Source
N52 Formula

Formula | FetchXml Queries | System Settings | Formula Trace | Related

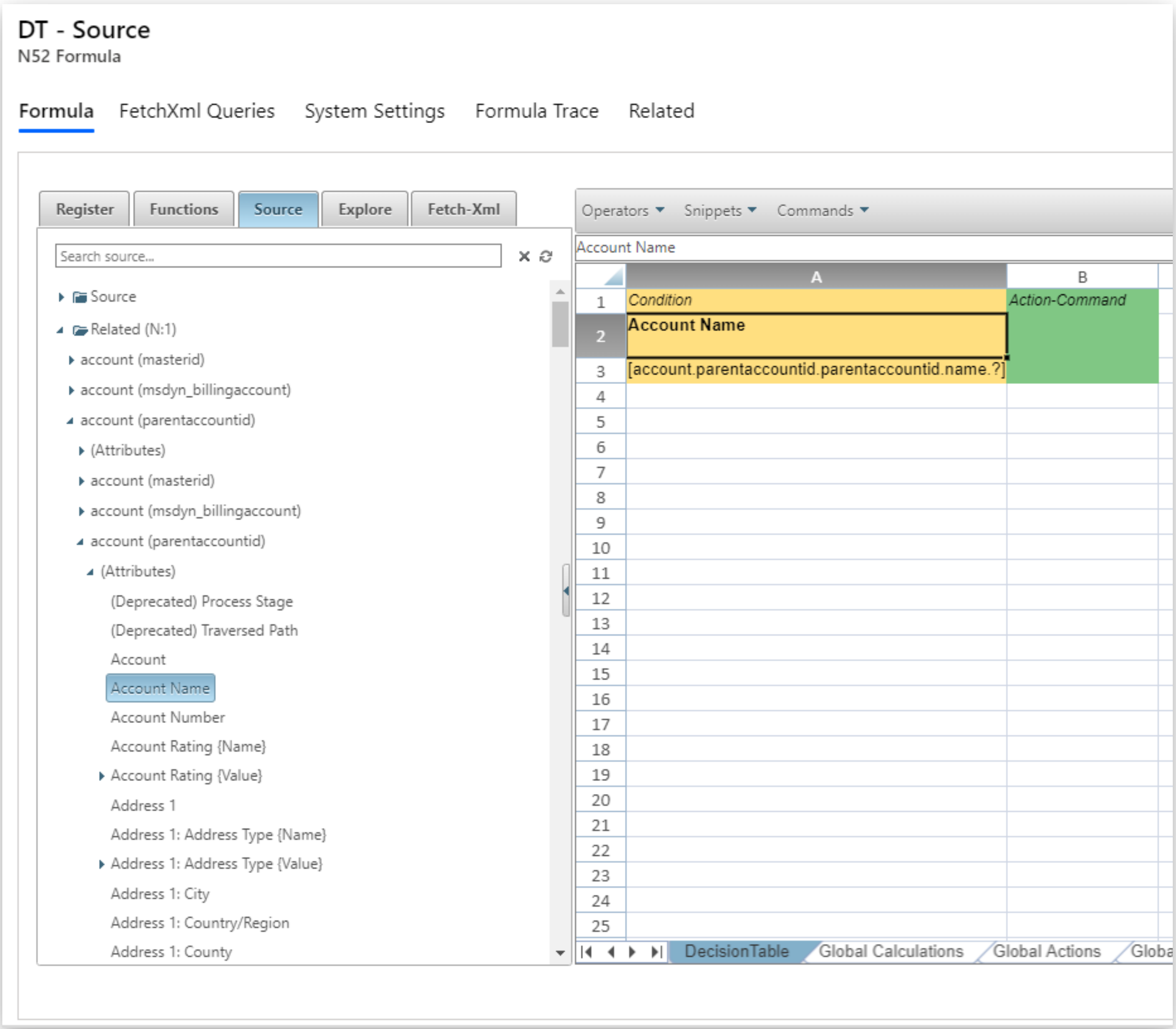
Register | Functions | Source | Explore | Fetch-Xml

- Source
- Related (N:1)
  - account (masterid)
  - account (msdyn_billingaccount)
  - account (parentaccountid)
    - (Attributes)
      - (Deprecated) Process Stage
      - (Deprecated) Traversed Path
      - Account
      - Account Name
      - Account Number
      - Account Rating {Name}
      - Account Rating {Value}
      - Address 1
      - Address 1: Address Type {Name}
      - Address 1: Address Type {Value}
      - Address 1: City
      - Address 1: Country/Region
      - Address 1: County
      - Address 1: Fax
      - Address 1: Freight Terms {Name}
      - Address 1: Freight Terms {Value}
      - Address 1: ID

	A	B
1	*Condition*	*Action-Command*
2	**Account Name**	
3	[account.parentaccountid.name.?]	

DecisionTable | Global Calculations | Global A

You can also access the related entities' related entities. In the screenshot below, the grandparent **Account's** name is referenced:


DT - Source
N52 Formula

- Source
- Related (N:1)
  - account (masterid)
  - account (msdyn_billingaccount)
  - account (parentaccountid)
    - (Attributes)
    - account (masterid)
    - account (msdyn_billingaccount)
    - account (parentaccountid)
      - (Attributes)
        - (Deprecated) Process Stage
        - (Deprecated) Traversed Path
        - Account
        - Account Name
        - Account Number
        - Account Rating {Name}
        - Account Rating {Value}
        - Address 1
        - Address 1: Address Type {Name}
        - Address 1: Address Type {Value}
        - Address 1: City
        - Address 1: Country/Region
        - Address 1: County

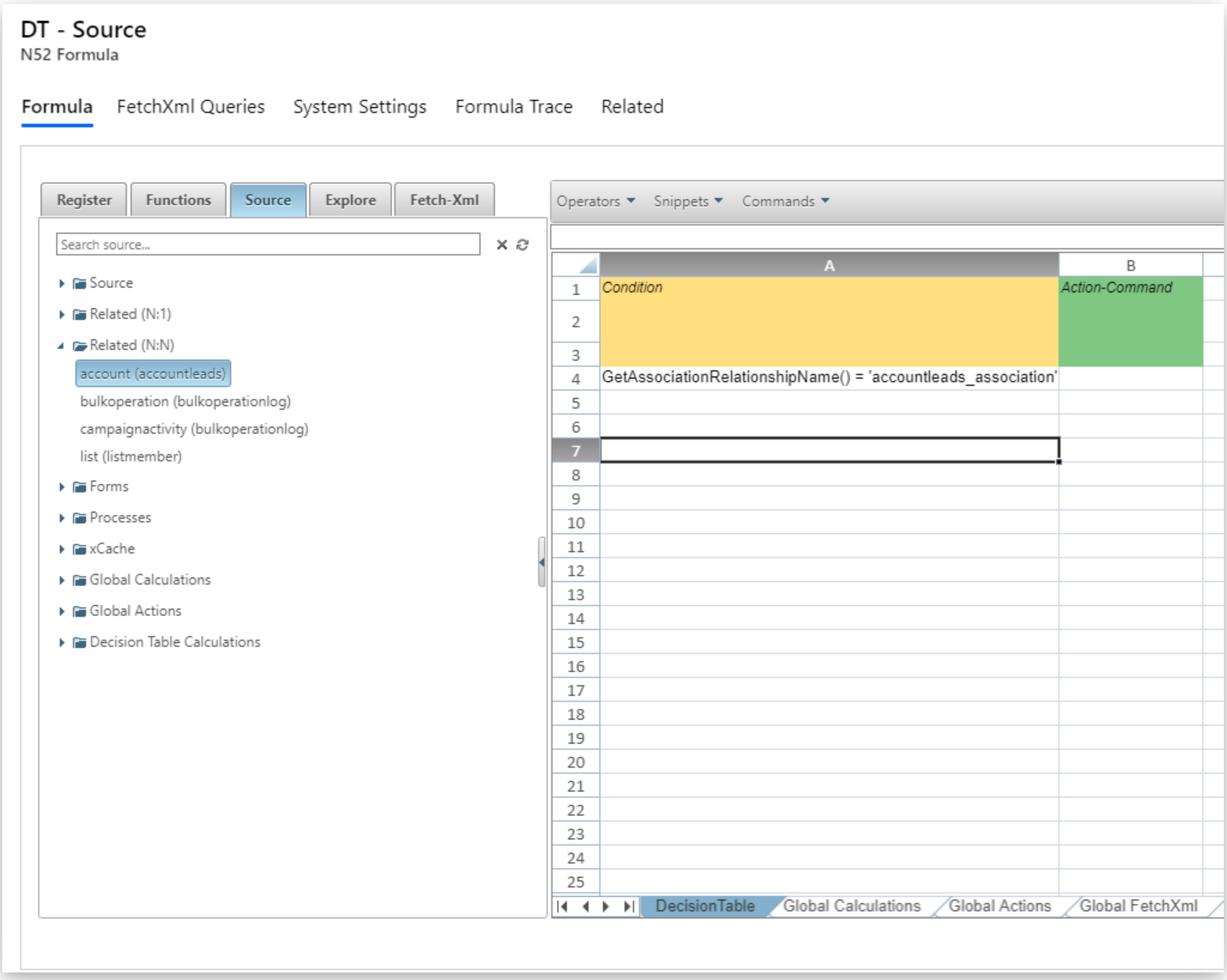
Account Name

	A	B
1	Condition	Action-Command
2	Account Name	
3	[account.parentaccountid.parentaccountid.name.?]	

This can continue as far down the relationship hierarchy as you need to go.

Related (N:N)

The **Related (N:N)** tree provides access to the name of the N:N relationships:

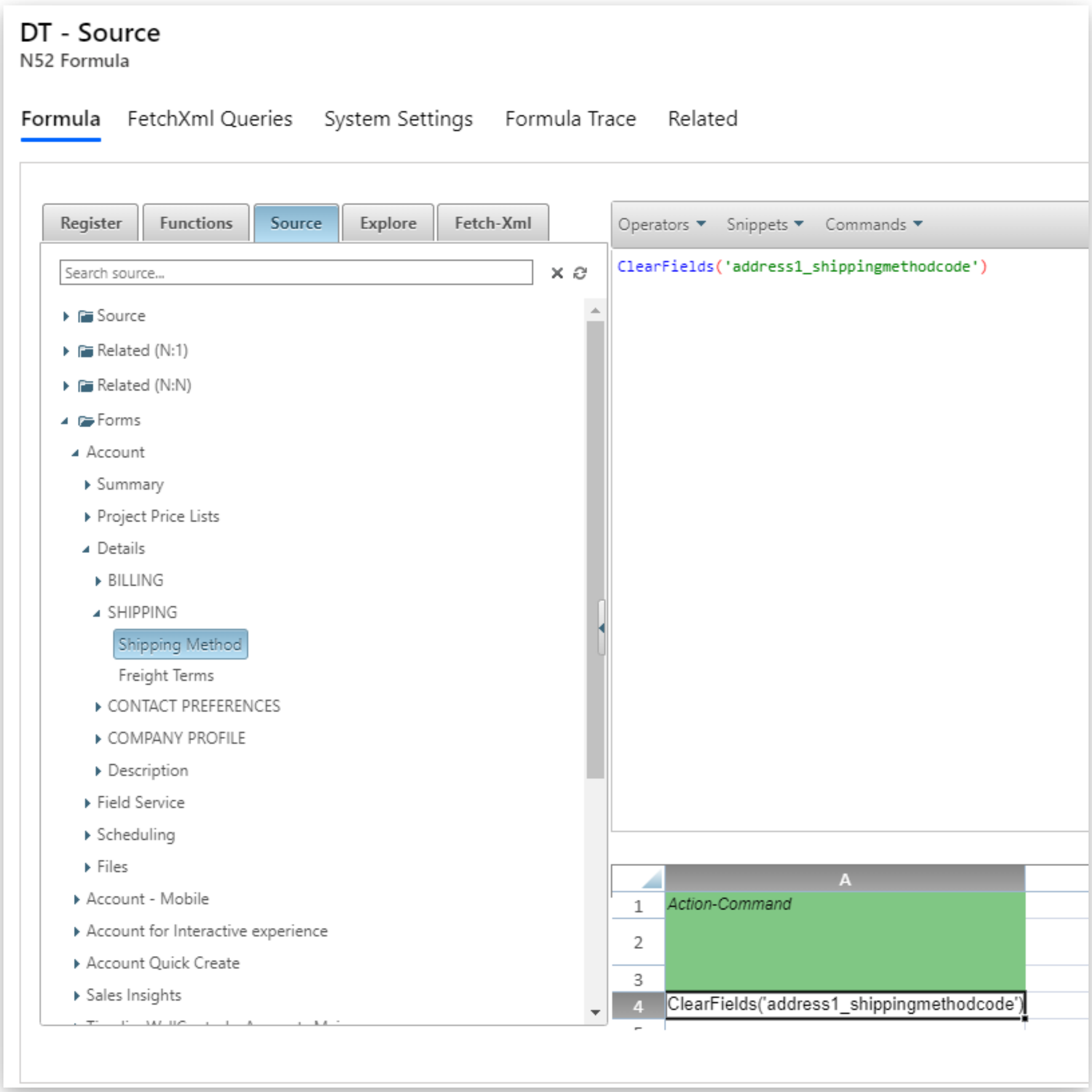


DT - Source
N52 Formula

	A	B
1	Condition	Action-Command
2		
3		
4	GetAssociationRelationshipName() = 'accountleads_association'	

Forms

The **Forms** tree provides access to the logical names of the source entity's forms and their **Section/Tabs/Fields**:

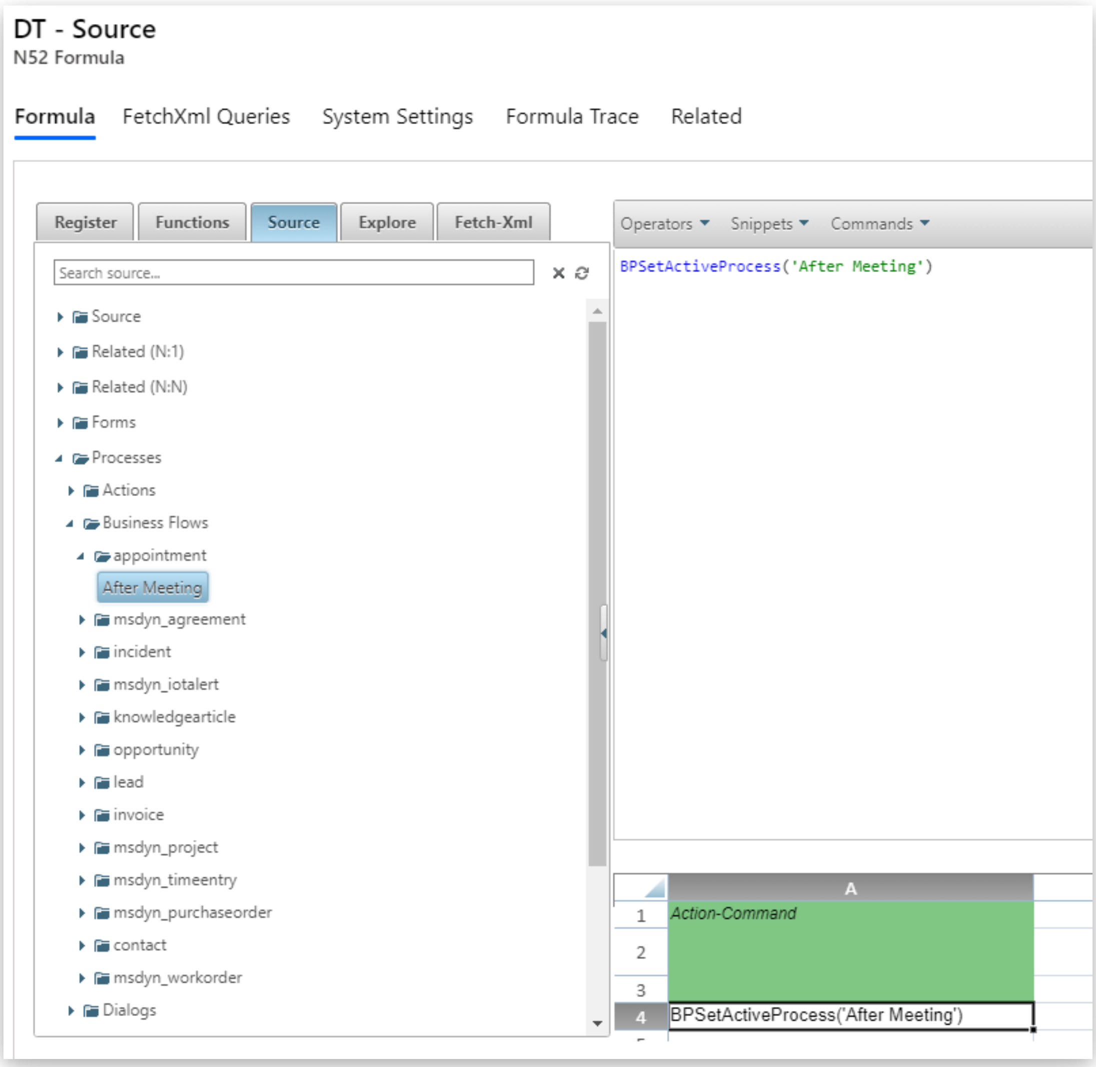


```
ClearFields('address1_shippingmethodcode')
```

	A
1	*Action-Command*
2	
3	
4	ClearFields('address1_shippingmethodcode')

Processes

The **Process** tree provides access to the names of the active Processes in the instance:

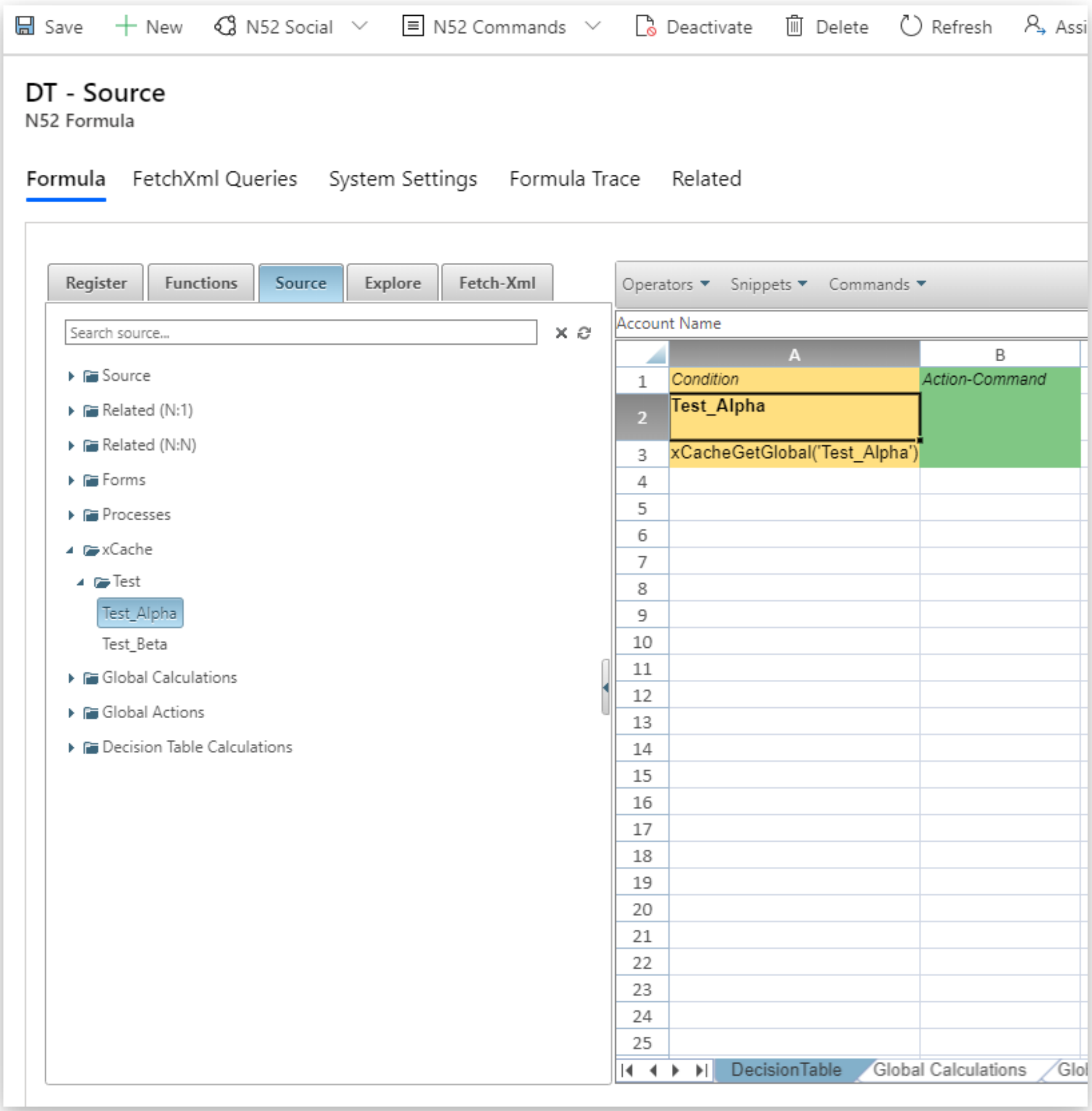


```
BPSetActiveProcess('After Meeting')
```

	A
1	*Action-Command*
2	
3	
4	BPSetActiveProcess('After Meeting')

xCache

The **xCache** tree provides access to the active **xCache** records that are stored in the instance:

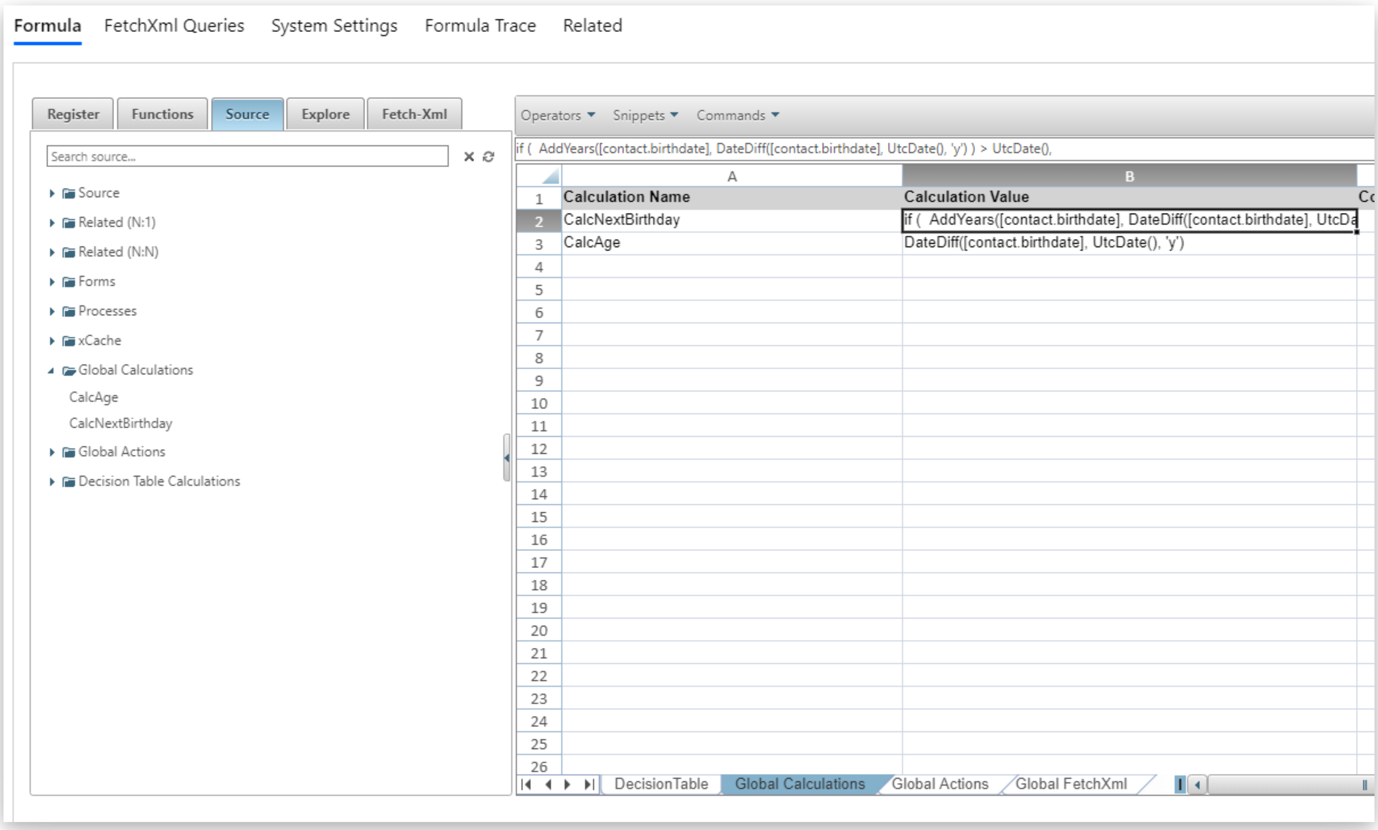


	A	B
1	Condition	Action-Command
2	Test_Alpha	
3	xCacheGetGlobal('Test_Alpha')	

For more information on **xCache** records, please see the xCache training course.

Global Calculations

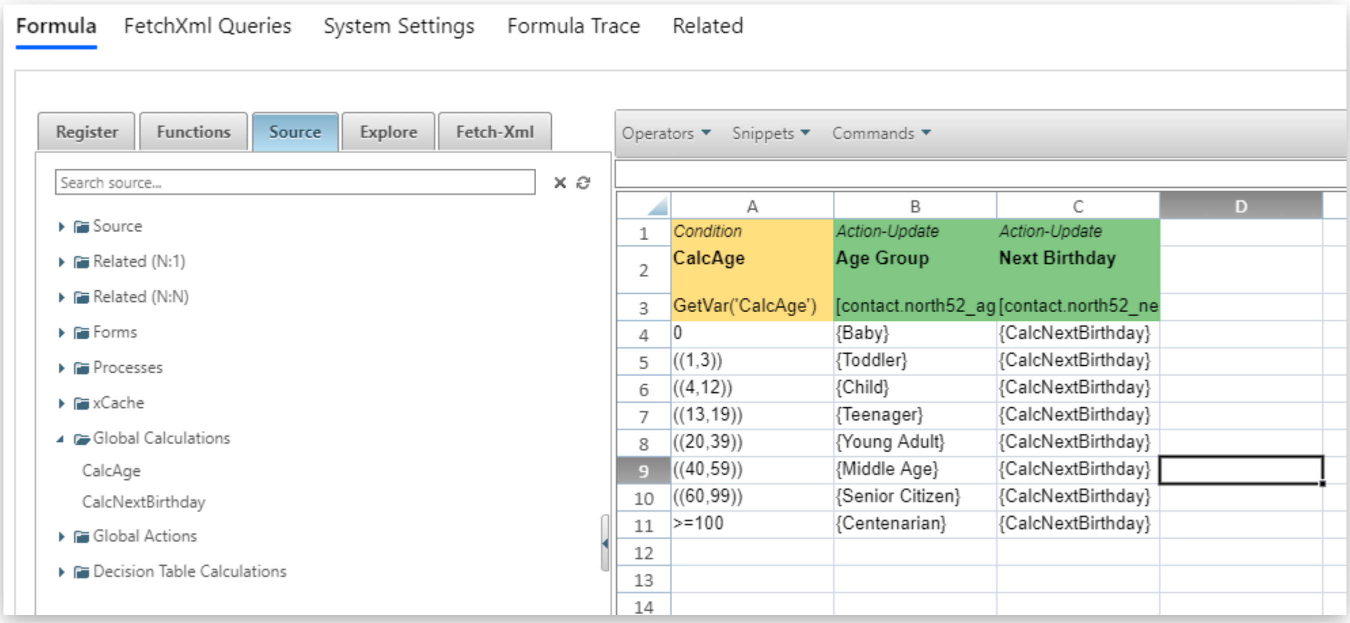
The **Global Calculation** node provides access to the Global Calculations that have been created and stored in the **Global Calculation sheet** for this Decision Table Formula.



```
if ( AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') ) > UtcDate(),
```

	A	B
1	Calculation Name	Calculation Value
2	CalcNextBirthday	if (AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDa
3	CalcAge	DateDiff([contact.birthdate], UtcDate(), 'y')

Referencing the Global Calculation in a Decision Table sheet:



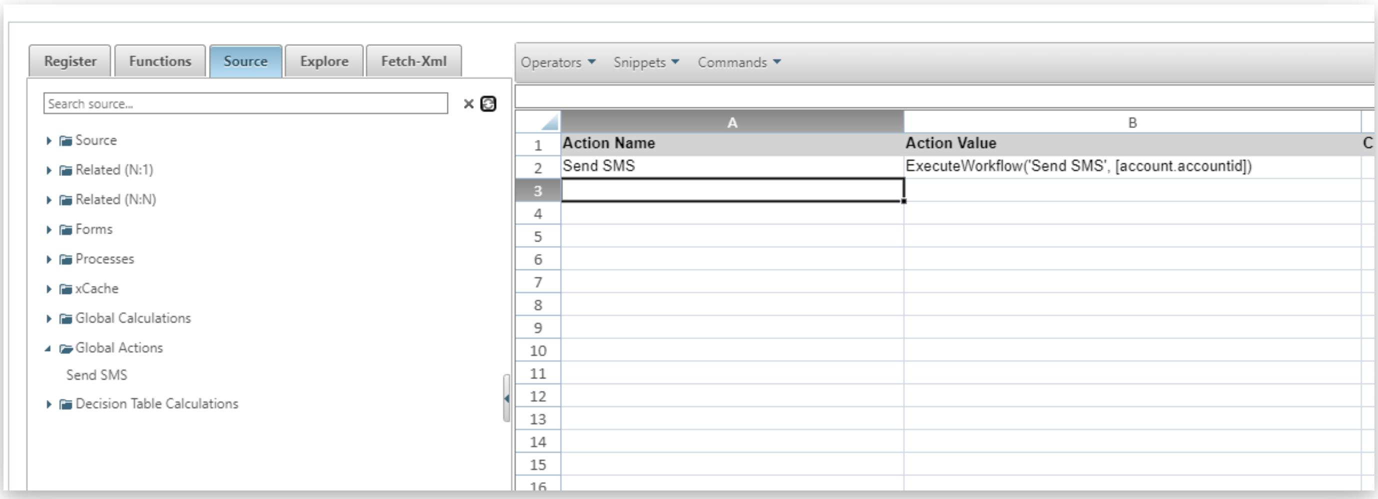
	A	B	C	D
1	*Condition*	*Action-Update*	*Action-Update*	
2	**CalcAge**	**Age Group**	**Next Birthday**	
3	GetVar('CalcAge')	[contact.north52_ag	[contact.north52_ne	
4	0	{Baby}	{CalcNextBirthday}	
5	((1,3))	{Toddler}	{CalcNextBirthday}	
6	((4,12))	{Child}	{CalcNextBirthday}	
7	((13,19))	{Teenager}	{CalcNextBirthday}	
8	((20,39))	{Young Adult}	{CalcNextBirthday}	
9	((40,59))	{Middle Age}	{CalcNextBirthday}	
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}	
11	>=100	{Centenarian}	{CalcNextBirthday}	

*Note: To create or edit a **Global Calculation**, right-click and select **Operations** > **Toggle Advanced Mode** and the **Global Calculation** sheet will be shown (you can also press F4).*

For more information on **Global Calculations**, please review the Global Calculations 'how to' knowledge base article.

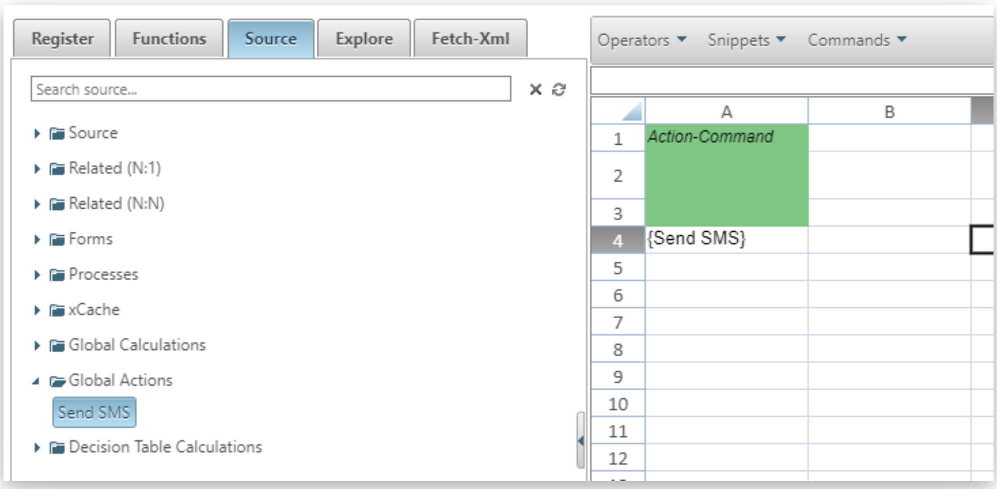
Global Actions

The **Global Actions** node provides access to the **Global Actions** that have been created and stored in the **Global Actions sheet** for this Decision Table Formula.



	A	B
1	Action Name	Action Value
2	Send SMS	ExecuteWorkflow('Send SMS', [account.accountid])

Referencing the **Global Action** in a **Decision Table** sheet:



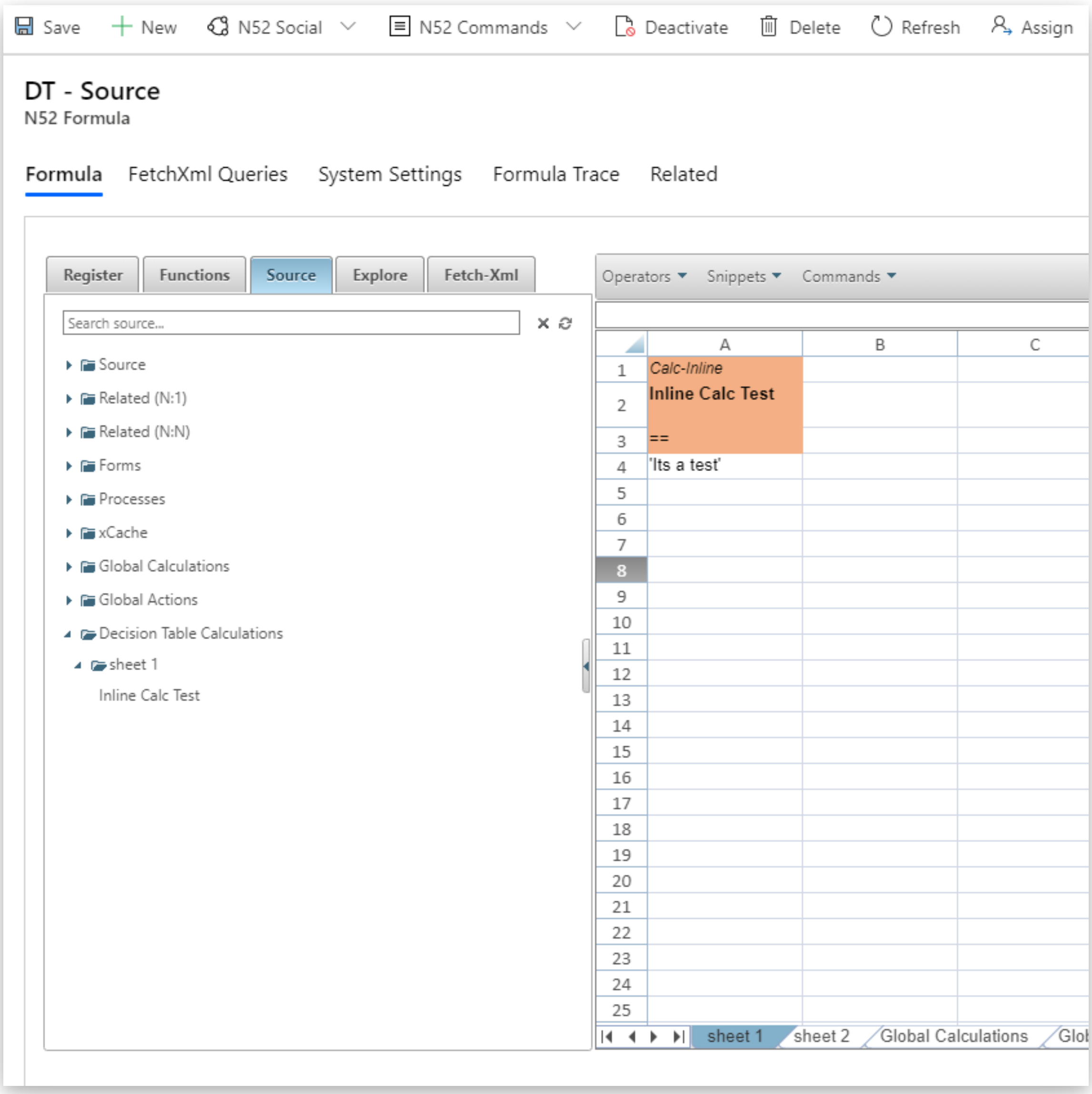
	A	B
1	Action-Command	
4	{Send SMS}	

Note: *To create or edit* ***Global Actions****, right-click and select* ***Operations > Toggle Advanced Mode****, and the* ***Global Actions*** *sheet will be shown (you can also press F4).*

For more information on Global Actions, please review the Global Actions 'how to' knowledge base article.

Decision Table Calculations

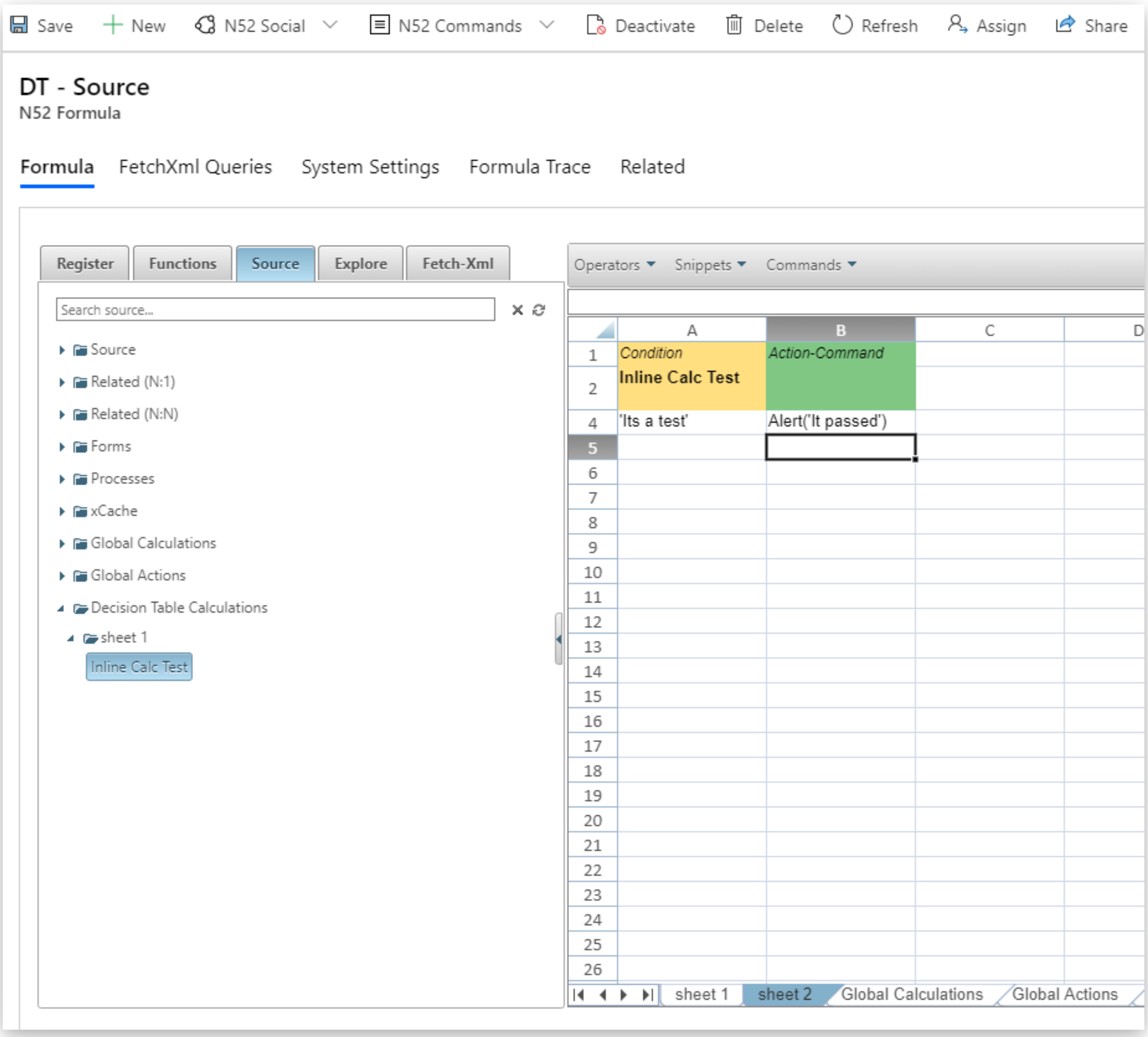
The Decision Table Calculation node provides access to the calculations that have been created inline on a Decision Table sheet (Inline Calculation, Table Calculation, Increment Calculation, Decrement Calculation or SetVarConcat Calculation):



DT - Source
N52 Formula

	A	B	C
1	*Calc-Inline*		
2	**Inline Calc Test**		
3	==		
4	'Its a test'		

Also, whenever an Inline Calculation, Table Calculation, Increment Calculation, Decrement Calculation or SetVarConcat Calculation is set, it will appear here for referencing



	A	B	C	D
1	*Condition*	*Action-Command*		
2	**Inline Calc Test**			
4	'Its a test'	Alert('It passed')		
5				

DT - How to - 16 - Use the Validator

[TOC]

Overview

The **Validator** allows you to run a basic and an advanced syntax check on your Formula. It will scan and highlight errors in the Formula structure.

This can be used in conjunction with **Classic** or **Decision Tables -** in this article we will focus on its use in **Decision Tables**.

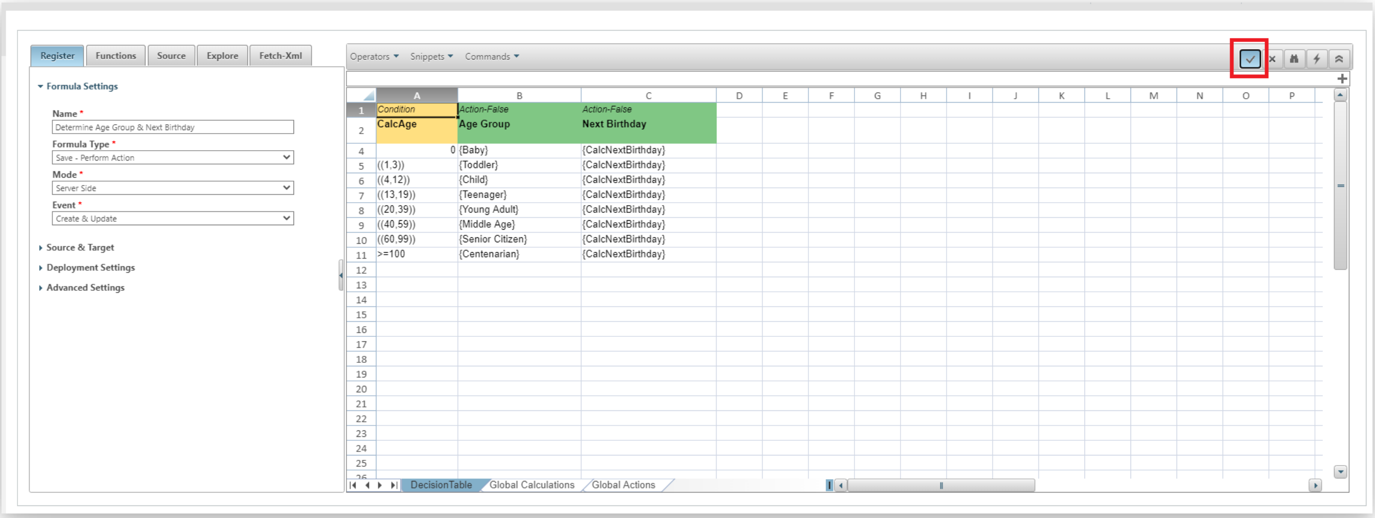
For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions,*** *please read the above articles if you need detailed configuration steps.*

Validator

You can locate the Validator on the top right of the editor command bar as shown below:



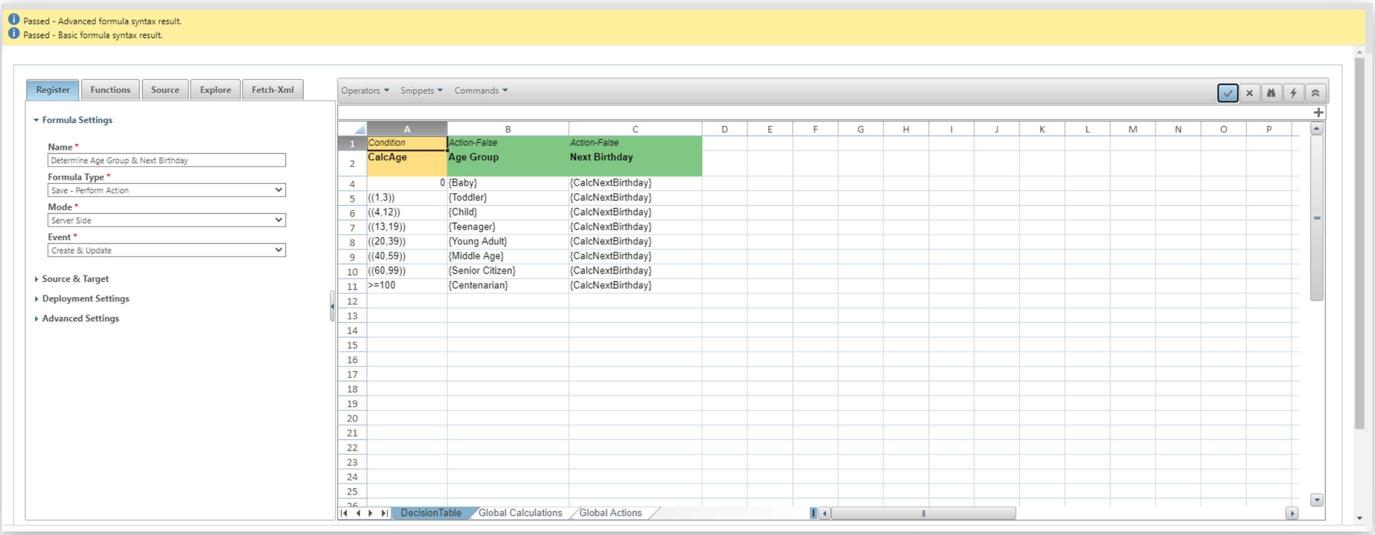
Formula Settings

Name *: Determine Age Group & Next Birthday
Formula Type *: Save - Perform Action
Mode *: Server Side
Event *: Create & Update

Source & Target
Deployment Settings
Advanced Settings

	A	B	C
1	*Condition*	*Action-False*	*Action-False*
2	**CalcAge**	**Age Group**	**Next Birthday**
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}

For the Validator to work, changes must be saved before selecting it:



Passed - Advanced formula syntax result.
Passed - Basic formula syntax result.

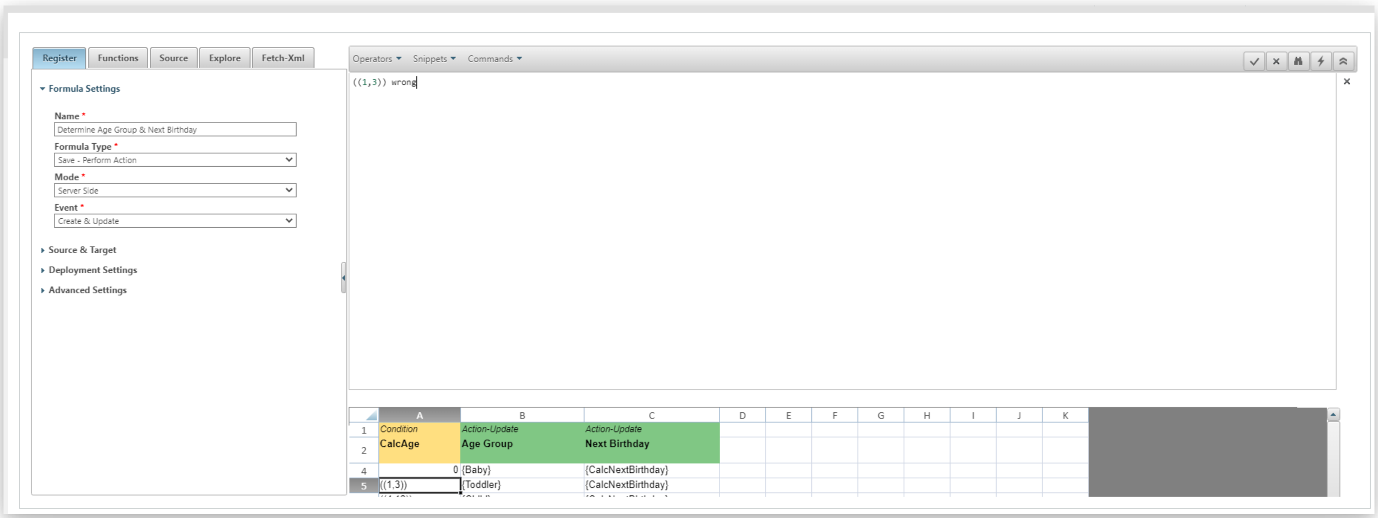
Formula Settings

Name *: Determine Age Group & Next Birthday
Formula Type *: Save - Perform Action
Mode *: Server Side
Event *: Create & Update

Source & Target
Deployment Settings
Advanced Settings

	A	B	C
1	*Condition*	*Action-False*	*Action-False*
2	**CalcAge**	**Age Group**	**Next Birthday**
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}

This includes the Advanced Editor as well - it must be collapsed and saved before a syntax check can be performed:



Register | Functions | Source | Explore | Fetch-Xml

Formula Settings

Name *
Determine Age Group & Next Birthday

Formula Type *
Save - Perform Action

Mode *
Server Side

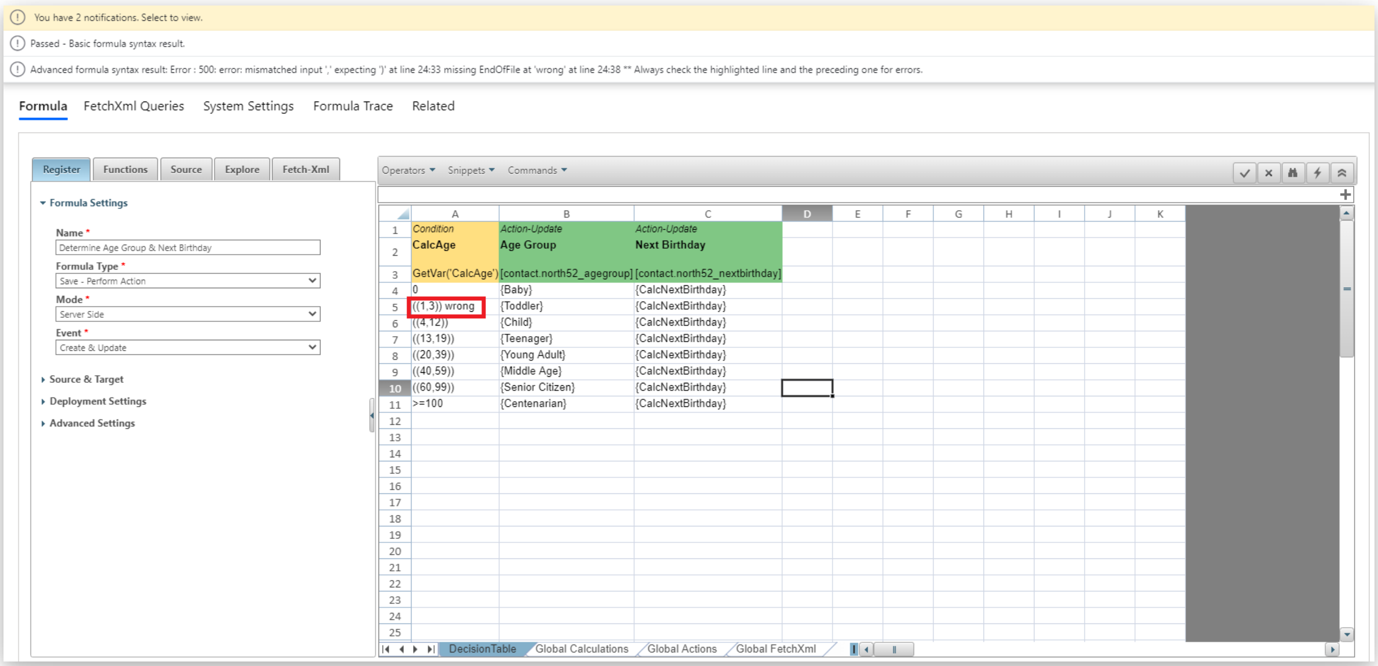
Event *
Create & Update

Source & Target
Deployment Settings
Advanced Settings

```
((1,3)) wrong
```

	A	B	C
1	Condition	Action-Update	Action-Update
2	CalcAge	Age Group	Next Birthday
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}

An example of an Error shown by the Validator:



You have 2 notifications. Select to view.

Passed - Basic formula syntax result.

Advanced formula syntax result: Error : 500: error: mismatched input ',' expecting ')' at line 24:33 missing EndOfFile at 'wrong' at line 24:38 ** Always check the highlighted line and the preceding one for errors.

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Register | Functions | Source | Explore | Fetch-Xml

Formula Settings

Name *
Determine Age Group & Next Birthday

Formula Type *
Save - Perform Action

Mode *
Server Side

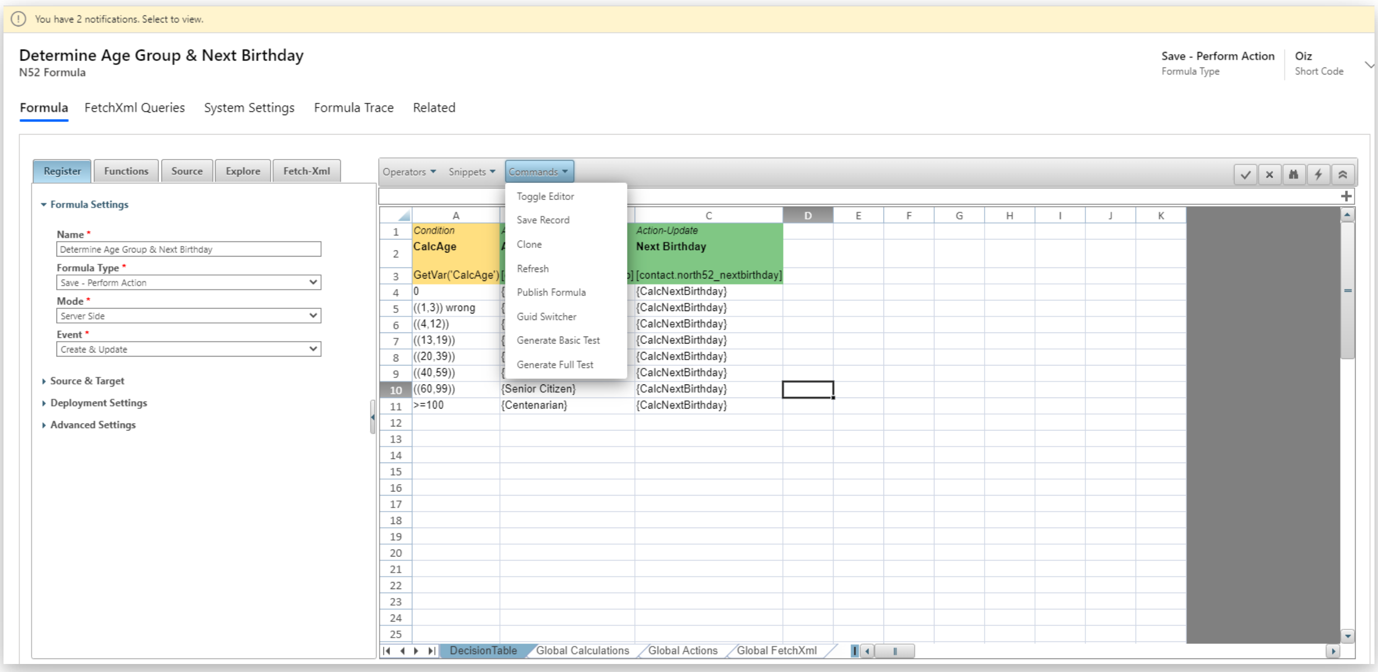
Event *
Create & Update

Source & Target
Deployment Settings
Advanced Settings

	A	B	C
1	Condition	Action-Update	Action-Update
2	CalcAge	Age Group	Next Birthday
3	GetVar('CalcAge')	[contact.north52_agegroup]	[contact.north52_nextbirthday]
4	0	{Baby}	{CalcNextBirthday}
5	((1,3)) wrong	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}

DecisionTable | Global Calculations | Global Actions | Global FetchXml

We can see the error highlighted by toggling the editor to classic mode:


You have 2 notifications. Select to view.

Determine Age Group & Next Birthday
N52 Formula

Save - Perform Action (Formula Type) | Oiz (Short Code)

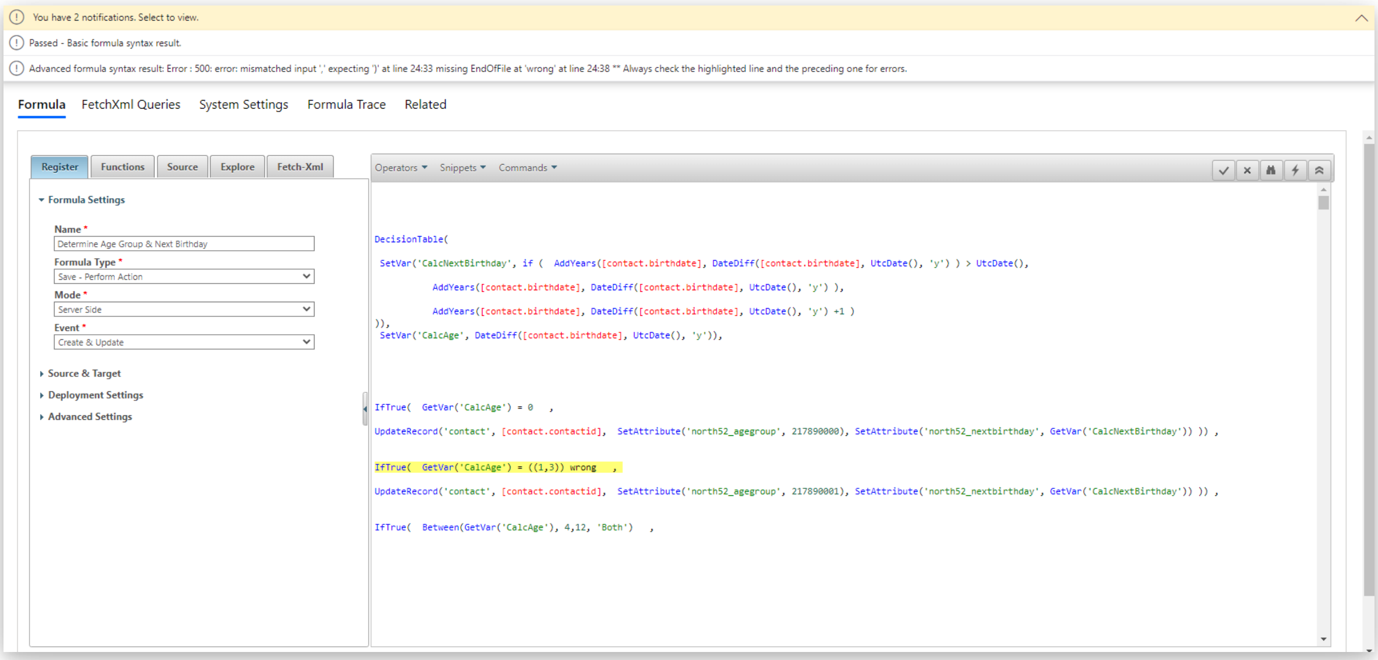
Formula | FetchXml Queries | System Settings | Formula Trace | Related

Formula Settings

- Name *: Determine Age Group & Next Birthday
- Formula Type *: Save - Perform Action
- Mode *: Server Side
- Event *: Create & Update

Source & Target
Deployment Settings
Advanced Settings

	A	B	C
1	*Condition*	*A*	*Action-Update*
2	**CalcAge**	**A**	**Next Birthday**
3	GetVar('CalcAge')	[	[contact.north52_nextbirthday]
4	0	{	{CalcNextBirthday}
5	((1,3)) wrong	{	{CalcNextBirthday}
6	((4,12))	{	{CalcNextBirthday}
7	((13,19))	{	{CalcNextBirthday}
8	((20,39))	{	{CalcNextBirthday}
9	((40,59))	{	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}


You have 2 notifications. Select to view.

Passed - Basic formula syntax result.

Advanced formula syntax result: Error : 500: error: mismatched input ',' expecting ')' at line 24:33 missing EndOfFile at 'wrong' at line 24:38 ** Always check the highlighted line and the preceding one for errors.

Formula | FetchXml Queries | System Settings | Formula Trace | Related

Formula Settings

- Name *: Determine Age Group & Next Birthday
- Formula Type *: Save - Perform Action
- Mode *: Server Side
- Event *: Create & Update

Source & Target
Deployment Settings
Advanced Settings

```
DecisionTable(

 SetVar('CalcNextBirthday', if (  AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') ) > UtcDate(),

          AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') ),

          AddYears([contact.birthdate], DateDiff([contact.birthdate], UtcDate(), 'y') +1 )
)),
 SetVar('CalcAge', DateDiff([contact.birthdate], UtcDate(), 'y')),



IfTrue(  GetVar('CalcAge') = 0   ,

UpdateRecord('contact', [contact.contactid],  SetAttribute('north52_agegroup', 217890000), SetAttribute('north52_nextbirthday', GetVar('CalcNextBirthday')) )) ,


IfTrue(  GetVar('CalcAge') = ((1,3)) wrong   ,

UpdateRecord('contact', [contact.contactid],  SetAttribute('north52_agegroup', 217890001), SetAttribute('north52_nextbirthday', GetVar('CalcNextBirthday')) )) ,


IfTrue(  Between(GetVar('CalcAge'), 4,12, 'Both')   ,
```

DT - How to - 17 - Tester (execute Formula from the editor)

[TOC]

Overview

The **Tester** allows users to quickly execute and test a Formula from within the Formula editor.

This can be used in conjunction with Classic Formulas or Decision Tables, however, in this article we focus on its use in Decision Tables.

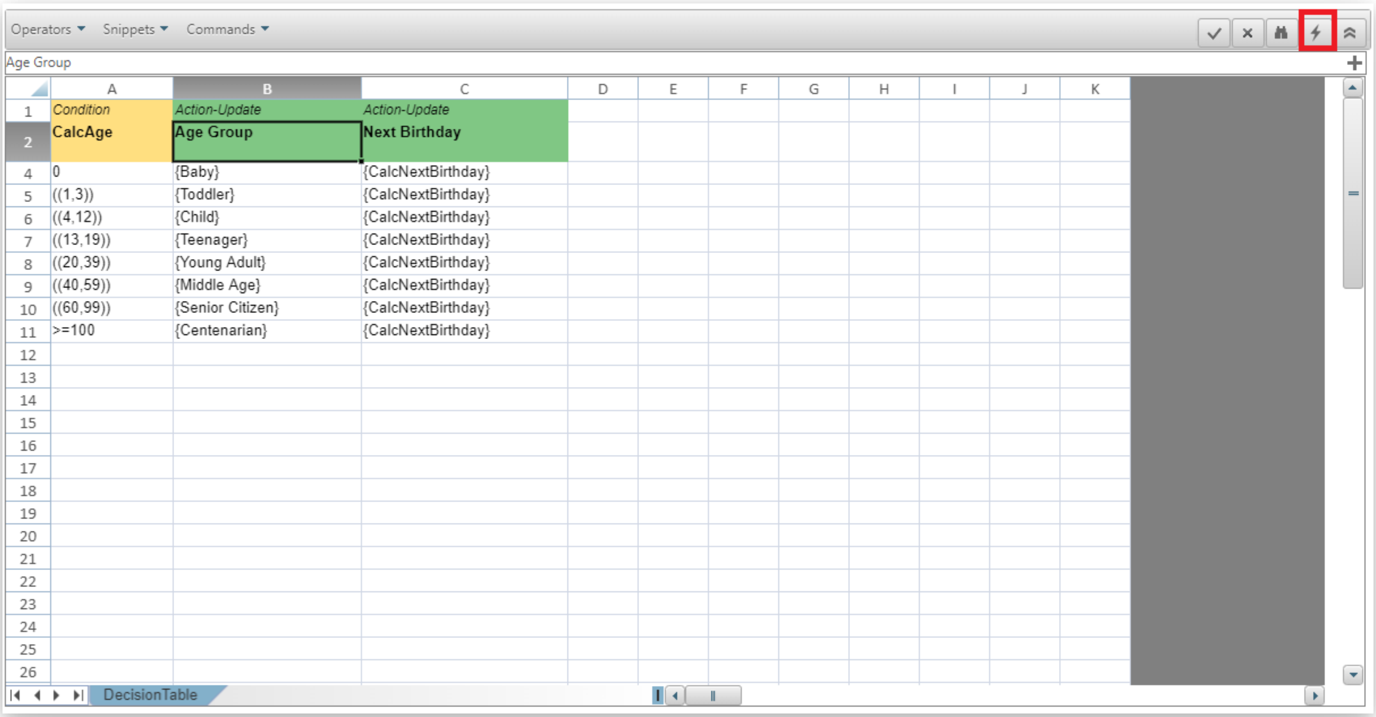
For this article it is assumed that you have at least basic familiarity with Decision Tables and/or have read the following articles:

- Everything you need to know about Decision Table Conditions
- How to set up your Actions!

Note: *We will not detail step-by-step instructions here on how to set up* ***Conditions*** *or* ***Actions****, please read the above articles if you need detailed configuration steps.*

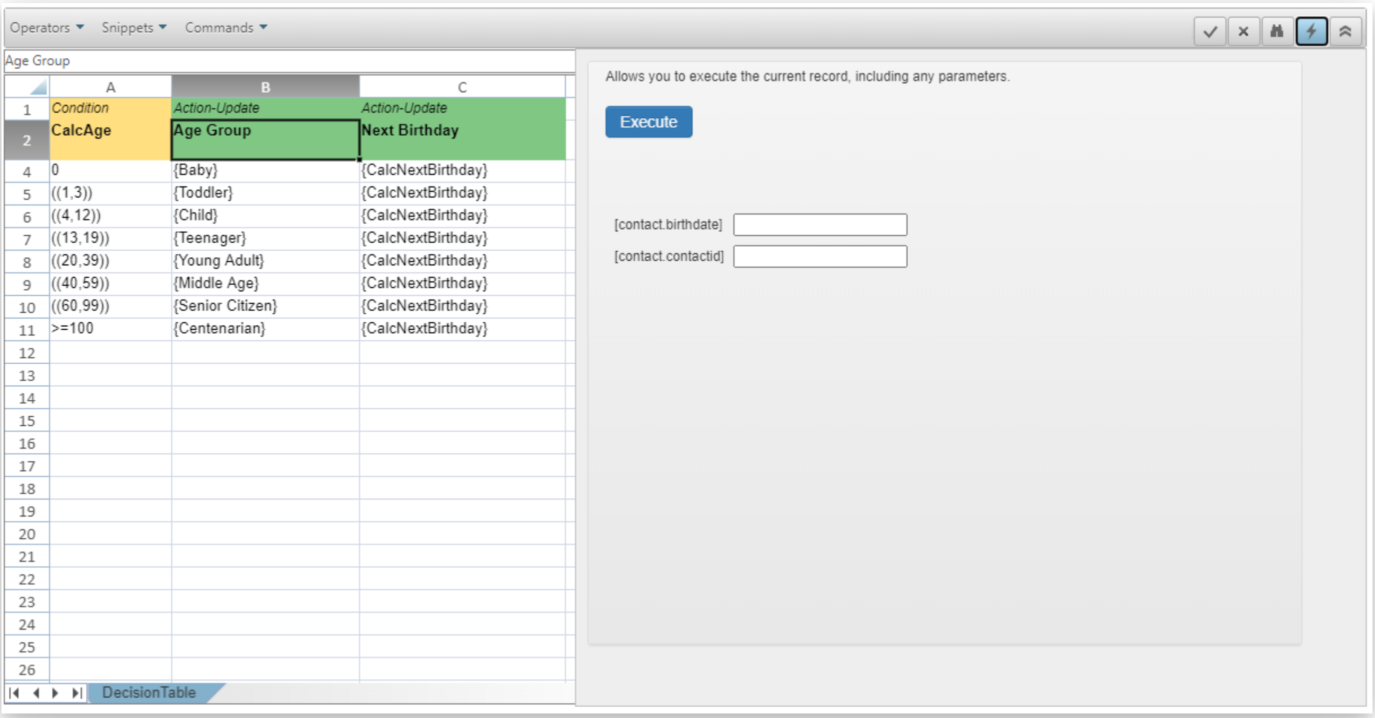
How to use the Tester

You can locate the Tester in the top right corner of the editor as shown below:



	A	B	C
1	Condition	Action-Update	Action-Update
2	CalcAge	Age Group	Next Birthday
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}

Note that the Formula must be saved before the Tester can be executed.



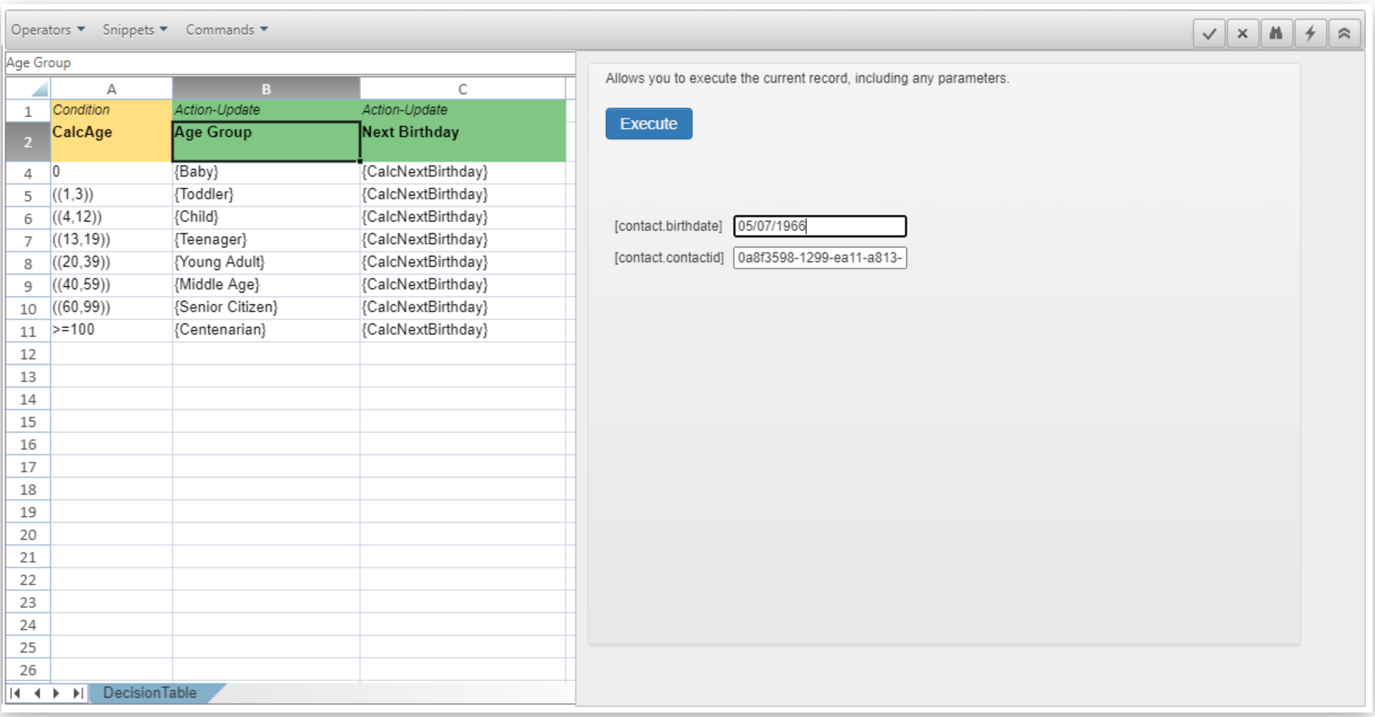
Age Group

	A	B	C
1	*Condition*	*Action-Update*	*Action-Update*
2	**CalcAge**	**Age Group**	**Next Birthday**
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

Allows you to execute the current record, including any parameters.

[contact.birthdate]	
[contact.contactid]	

The Tester will request any references to data it needs and execute the Formula using the values supplied: (***Note:*** *some formulas require the unique ID of the record. You can typically find this in the URL of the record or by using a browser extension*)



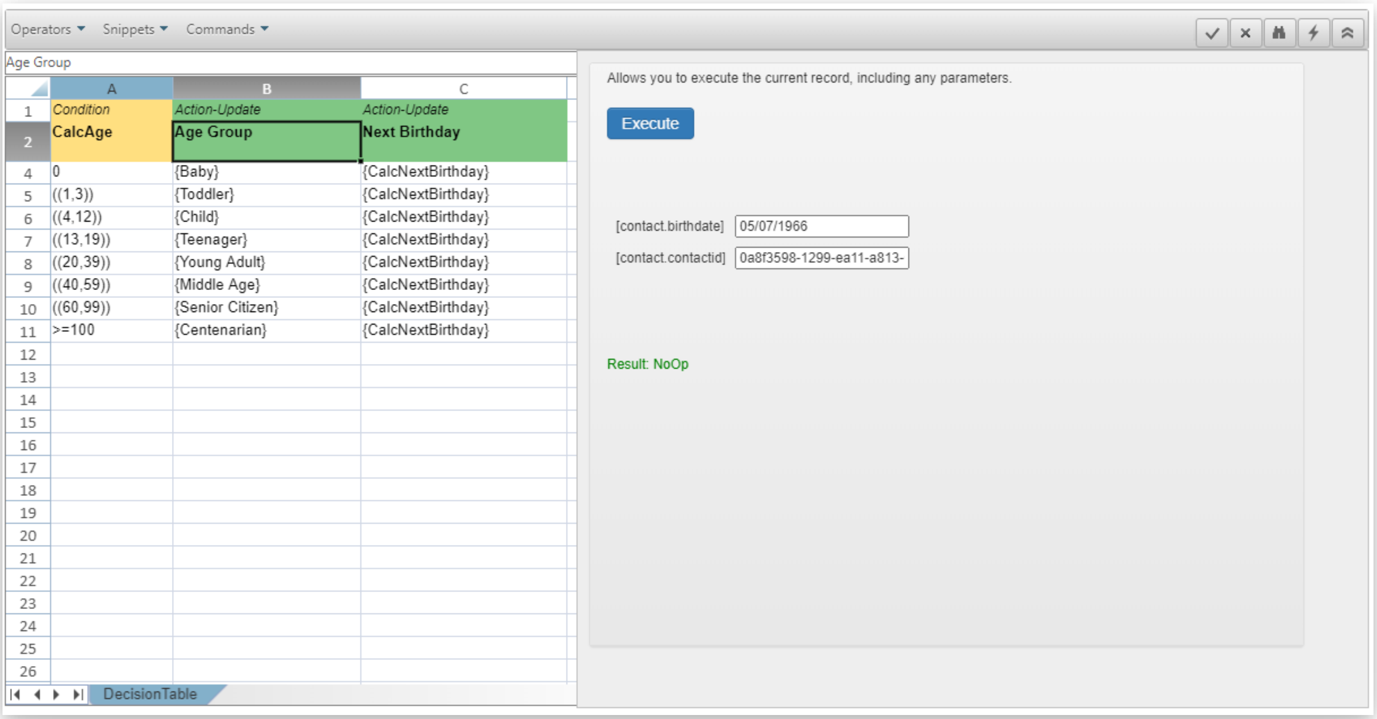
Age Group

	A	B	C
1	*Condition*	*Action-Update*	*Action-Update*
2	**CalcAge**	**Age Group**	**Next Birthday**
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

Allows you to execute the current record, including any parameters.

[contact.birthdate]	05/07/1966
[contact.contactid]	0a8f3598-1299-ea11-a813-

A successful execution will return result message, in this case **NoOp** (which means No Operation):



	A	B	C
1	Condition	Action-Update	Action-Update
2	CalcAge	Age Group	Next Birthday
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

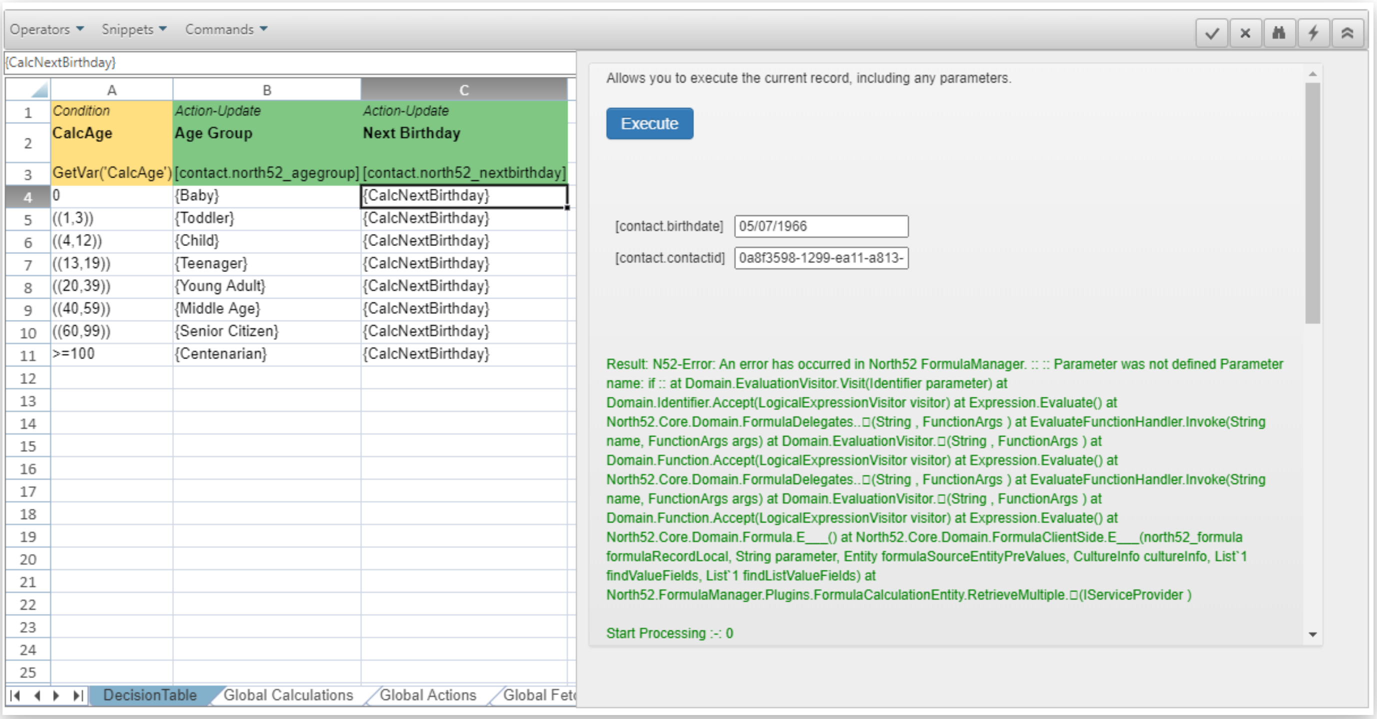
Allows you to execute the current record, including any parameters.

Execute

[contact.birthdate]	05/07/1966
[contact.contactid]	0a8f3598-1299-ea11-a813-

Result: NoOp

If there is an error in the formula, a trace log will show up like in the image below. For this example an IF statement without any parameters was entered to purposefully break it:



	A	B	C
1	Condition	Action-Update	Action-Update
2	CalcAge	Age Group	Next Birthday
3	GetVar('CalcAge')	[contact.north52_agegroup]	[contact.north52_nextbirthday]
4	0	{Baby}	{CalcNextBirthday}
5	((1,3))	{Toddler}	{CalcNextBirthday}
6	((4,12))	{Child}	{CalcNextBirthday}
7	((13,19))	{Teenager}	{CalcNextBirthday}
8	((20,39))	{Young Adult}	{CalcNextBirthday}
9	((40,59))	{Middle Age}	{CalcNextBirthday}
10	((60,99))	{Senior Citizen}	{CalcNextBirthday}
11	>=100	{Centenarian}	{CalcNextBirthday}
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

Allows you to execute the current record, including any parameters.

Execute

[contact.birthdate]	05/07/1966
[contact.contactid]	0a8f3598-1299-ea11-a813-

```
Result: N52-Error: An error has occurred in North52 FormulaManager. :: :: Parameter was not defined Parameter
name: if :: at Domain.EvaluationVisitor.Visit(Identifier parameter) at
Domain.Identifier.Accept(LogicalExpressionVisitor visitor) at Expression.Evaluate() at
North52.Core.Domain.FormulaDelegates..□(String , FunctionArgs ) at EvaluateFunctionHandler.Invoke(String
name, FunctionArgs args) at Domain.EvaluationVisitor.□(String , FunctionArgs ) at
Domain.Function.Accept(LogicalExpressionVisitor visitor) at Expression.Evaluate() at
North52.Core.Domain.FormulaDelegates..□(String , FunctionArgs ) at EvaluateFunctionHandler.Invoke(String
name, FunctionArgs args) at Domain.EvaluationVisitor.□(String , FunctionArgs ) at
Domain.Function.Accept(LogicalExpressionVisitor visitor) at Expression.Evaluate() at
North52.Core.Domain.Formula.E___() at North52.Core.Domain.FormulaClientSide.E___(north52_formula
formulaRecordLocal, String parameter, Entity formulaSourceEntityPreValues, CultureInfo cultureInfo, List`1
findValueFields, List`1 findListValueFields) at
North52.FormulaManager.Plugins.FormulaCalculationEntity.RetrieveMultiple.□(IServiceProvider )

Start Processing :-: 0
```