smarter drafter PRO
by tensis

Calculations – Advanced Examples

Module 15 - Calculations

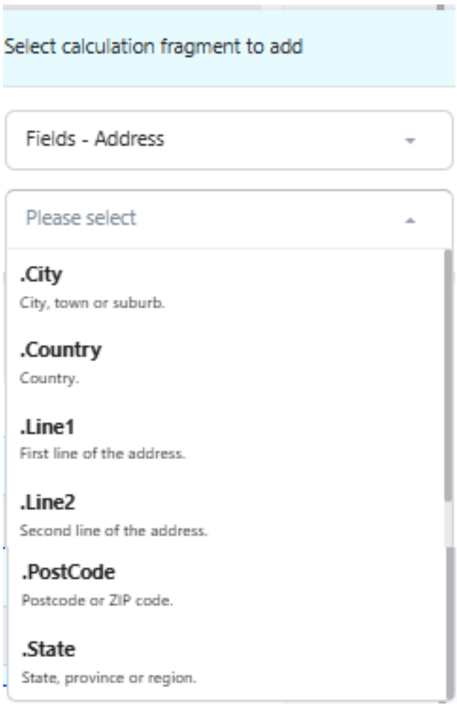
Version 1.0

Table of Contents

Note: This guide is a quick reference tool providing examples of calculations. For detailed instructions and a general overview of calculations, please review the Calculations - Introduction and Basic Examples guide.

1. Addresses

The preset calculations for addresses assist in extracting specific components of the address, e.g.:



Select calculation fragment to add

Fields - Address

Please select

.City
City, town or suburb.

.Country
Country.

.Line1
First line of the address.

.Line2
Second line of the address.

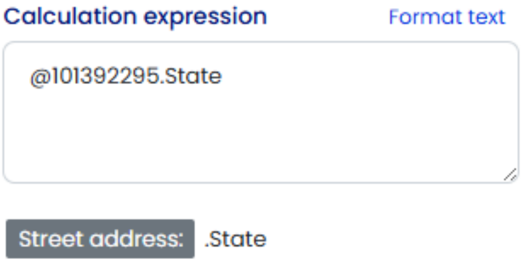
.PostCode
Postcode or ZIP code.

.State
State, province or region.

You can select the component of an address via the Word Addin to insert these details as a field in your template. You would only need to create a calculation to extract a specific component in a few circumstances.

1.1 Extract the State of an Address

Create a text field and use the Calculation Tool. Select the **Fields - Address** data type and the **State** calculation type. Insert the address field ID to build the expression.



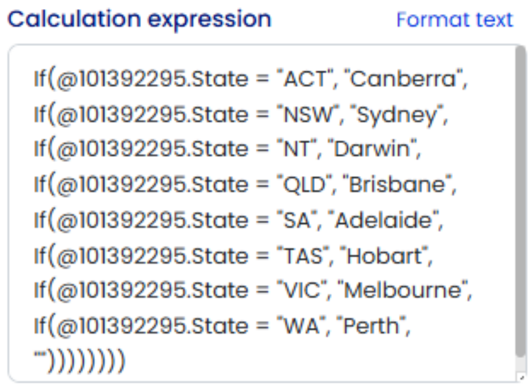
Calculation expression

@101392295.State

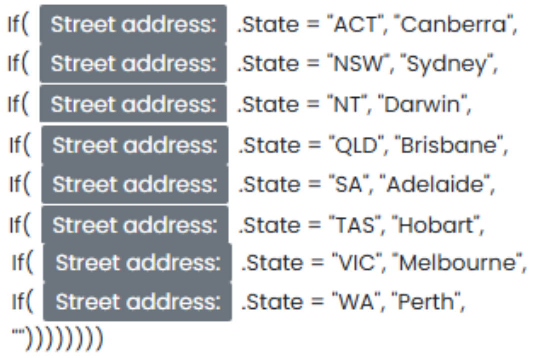
Street address: .State

1.2 Conditional Field Using State to Return the Capital City

Use a text field and write an expression to use the address state to determine another factor, e.g. the corresponding capital city:



```
If(@101392295.State = "ACT", "Canberra",
If(@101392295.State = "NSW", "Sydney",
If(@101392295.State = "NT", "Darwin",
If(@101392295.State = "QLD", "Brisbane",
If(@101392295.State = "SA", "Adelaide",
If(@101392295.State = "TAS", "Hobart",
If(@101392295.State = "VIC", "Melbourne",
If(@101392295.State = "WA", "Perth",
""))))))))
```



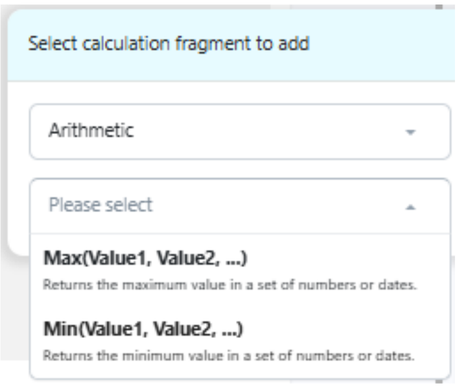
```
If( Street address: .State = "ACT", "Canberra",
If( Street address: .State = "NSW", "Sydney",
If( Street address: .State = "NT", "Darwin",
If( Street address: .State = "QLD", "Brisbane",
If( Street address: .State = "SA", "Adelaide",
If( Street address: .State = "TAS", "Hobart",
If( Street address: .State = "VIC", "Melbourne",
If( Street address: .State = "WA", "Perth",
""))))))))
```

Note: An alternative to manually writing this expression would be to use a layered/nested IF THEN ELSE statement by building this out in a conditional calculation using the Calculation Builder tool.

State fields are often the basis of calculations and are used to determine many other factors in documents that differ by state, e.g. lodgement fees, legislation, limitations etc. While conditional rules could achieve the same outcome, in this instance you would need a rule for each state – so 8 conditional rules – rather than a single calculation.

2. Arithmetic – Identify Min/Max Values

The preset calculations available for arithmetic are:



Select calculation fragment to add

Arithmetic

Please select

Max(Value1, Value2, ...)
Returns the maximum value in a set of numbers or dates.

Min(Value1, Value2, ...)
Returns the minimum value in a set of numbers or dates.

- Max will return the field that has the maximum value out of the specified fields
- Min will return the field that has the minimum value out of the specified fields

Both options require two or more number or date/time fields to calculate the min/max, and all fields must be the same type, eg all numbers or all date/times.

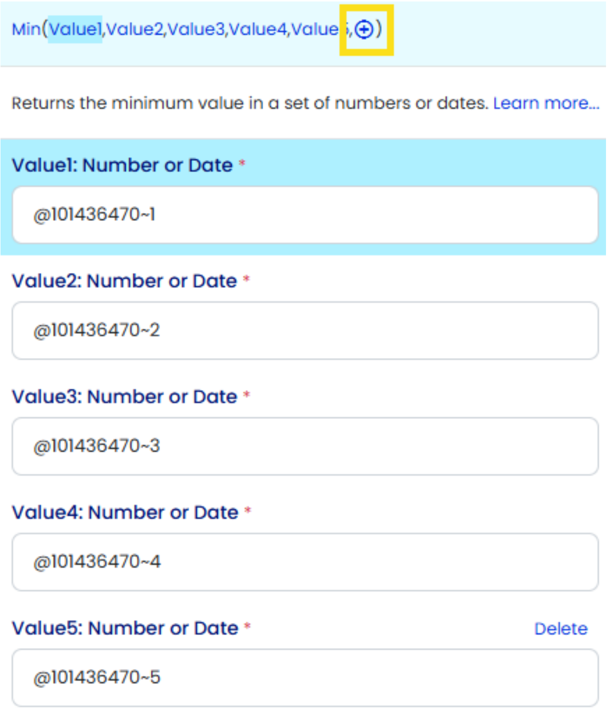
Use a date or number field, depending on the format of the data to be returned.

2.1 Field to Compare DOBs and Determine When the First Child was Born

Create a date field and use the Calculation Builder to create an expression to compare the children's dates of birth and determine when the first child was born.

Select the Min option and then add the relevant fields as the values to be compared.

Click **+** to add more fields/values to compare.



Min(Value1,Value2,Value3,Value4,Value5,⊕)

Returns the minimum value in a set of numbers or dates. Learn more...

Value1: Number or Date *
@101436470~1

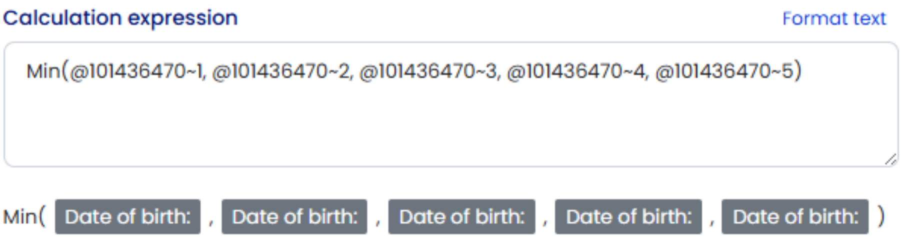
Value2: Number or Date *
@101436470~2

Value3: Number or Date *
@101436470~3

Value4: Number or Date *
@101436470~4

Value5: Number or Date * Delete
@101436470~5

The calculation expression will be inserted.



Calculation expression Format text

```
Min(@101436470~1, @101436470~2, @101436470~3, @101436470~4, @101436470~5)
```

Min(Date of birth: , Date of birth: , Date of birth: , Date of birth: , Date of birth:)

Note: This example uses DOBs from a repeating section and has to refer to each specific repeat instance. In this situation, it is important to set a realistic min/max number of repeats to limit the extent of this calculation on a repeat.

3. Concatenating Text

Concatenating text is a calculation expression that joins both fields and static text together.

As the result of the field is a string of text, use a text field for this type of calculation.

To concatenate text, we need to use certain characters within the expression so that it can be interpreted correctly:

+ to join the pieces together

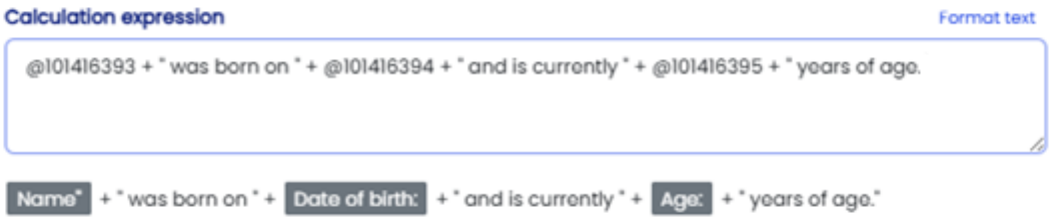
" " to identify static text

@... to enter field IDs

The expressions for concatenated text calculations need to be written manually.

The expression below creates the sentence:

Jane was born on 15 March 2001 and is currently 24 years of age.



```
@101416393 + " was born on " + @101416394 + " and is currently " + @101416395 + " years of age.
```

Name* + " was born on " + Date of birth: + " and is currently " + Age: + " years of age."

It is likely that this sentence would be created in the body of the template – BUT if this text were the product of an IF condition, then you may use it in a calculation.

You may consider using conditional rules where conditioning is required, though again it depends on how layered and how many conditions there are and whether a single calculation will achieve what would instead require multiple fields / conditional rules.

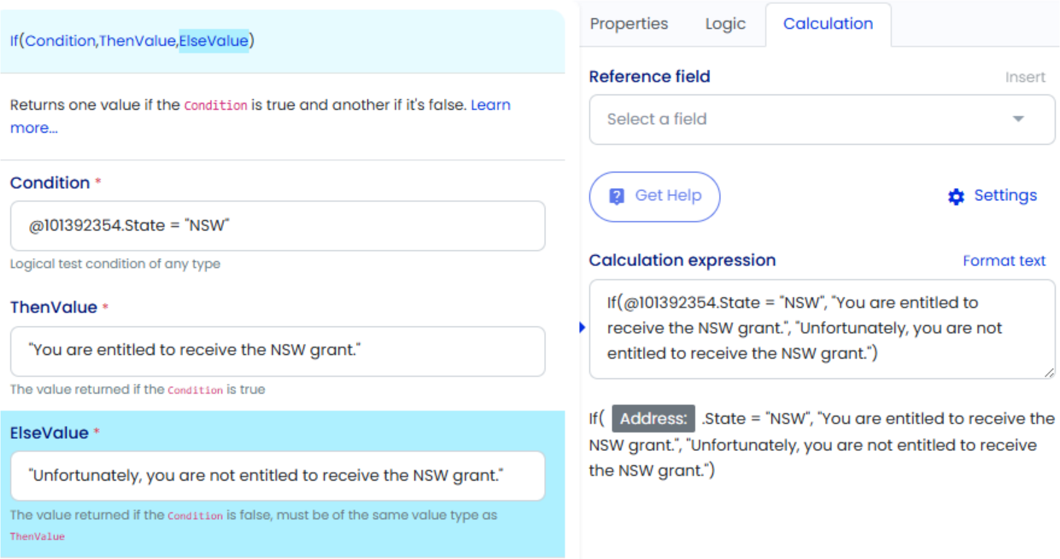
4. Conditional: IF - THEN - ELSE

4.1 Basic IF – THEN – ELSE Condition

Create a new field, selecting the appropriate field type depending on the type of data the calculation is to return.

Use the Calculation Tool and select the **conditional** data type and build the expression.

The output of this example is based on whether or not the client lives in NSW.



If(Condition,ThenValue,ElseValue)

Returns one value if the Condition is true and another if it's false. Learn more...

Condition *

@101392354.State = "NSW"

Logical test condition of any type

ThenValue *

"You are entitled to receive the NSW grant."

The value returned if the Condition is true

ElseValue *

"Unfortunately, you are not entitled to receive the NSW grant."

The value returned if the Condition is false, must be of the same value type as ThenValue

Reference field

Select a field

Calculation expression

If(@101392354.State = "NSW", "You are entitled to receive the NSW grant.", "Unfortunately, you are not entitled to receive the NSW grant.")

If(Address: .State = "NSW", "You are entitled to receive the NSW grant.", "Unfortunately, you are not entitled to receive the NSW grant.")

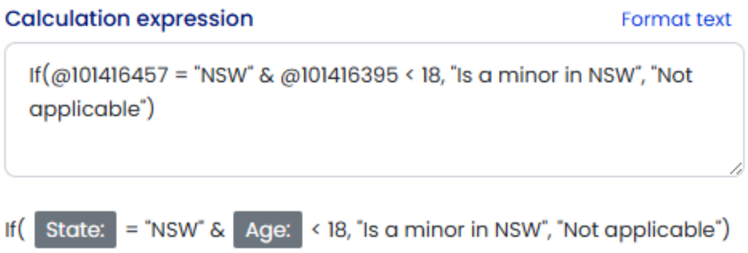
If the client lives in NSW - the text confirming that they are ENTITLED to a grant will appear
ELSE – meaning they don't live in NSW - the text confirming that they are NOT ENTITLED to a grant will appear.

4.2 Using Multiple Conditions within a Single Expression

A single expression can contain multiple conditions so that a combination of criteria can be assigned.

This example uses an IF statement to check a combination of answers to determine the result.

EG: **IF** the child lives in NSW **AND** is under 18, return 'Is a minor in NSW' ELSE return 'Not applicable'.



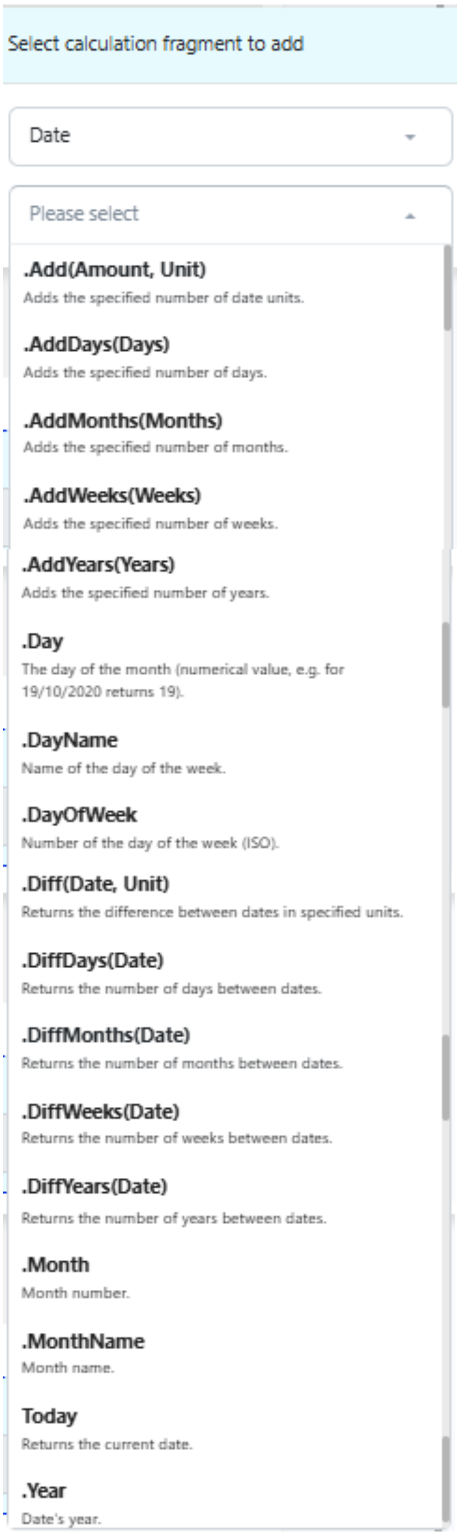
```
If(@101416457 = "NSW" & @101416395 < 18, "Is a minor in NSW", "Not applicable")
```

```
If( State: = "NSW" & Age: < 18, "Is a minor in NSW", "Not applicable")
```

Note: Expressions can be built applying the same principles for OR and NOT condition combinations. The expressions can be layered/nested and as complex as you are able to create.

5. Dates

The preset calculations available for dates are:



Select calculation fragment to add

Date

Please select

.Add(Amount, Unit)
Adds the specified number of date units.

.AddDays(Days)
Adds the specified number of days.

.AddMonths(Months)
Adds the specified number of months.

.AddWeeks(Weeks)
Adds the specified number of weeks.

.AddYears(Years)
Adds the specified number of years.

.Day
The day of the month (numerical value, e.g. for 19/10/2020 returns 19).

.DayName
Name of the day of the week.

.DayOfWeek
Number of the day of the week (ISO).

.Diff(Date, Unit)
Returns the difference between dates in specified units.

.DiffDays(Date)
Returns the number of days between dates.

.DiffMonths(Date)
Returns the number of months between dates.

.DiffWeeks(Date)
Returns the number of weeks between dates.

.DiffYears(Date)
Returns the number of years between dates.

.Month
Month number.

.MonthName
Month name.

Today
Returns the current date.

.Year
Date's year.

Calculations can be used to manipulate dates fields and return alternate data, eg:

- Split out date components
  - return the day number, day of week, month, year etc
  - use a **text** field for the calc

- Calculation additions and subtractions
  - returns the resulting date
  - use a **date** field for the calc

- Calculate time between dates
  - measure in days, weeks, months or years
  - returns difference between the dates in the selected measurement
  - use a **number** field for the calc

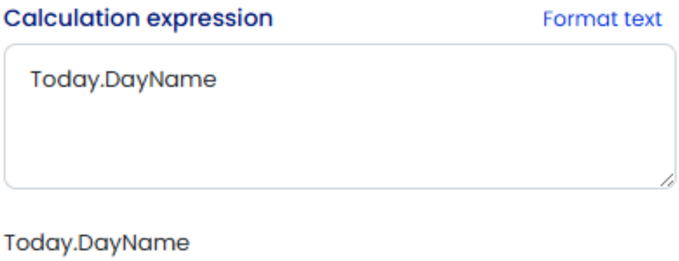
Note: See some simple date calculation examples in the Calculations - Introduction and Basic Examples guide.

5.1 Extract the Day of the Week

Create a text field and use the Calculation Tool. Select the **Date** data type and **DayName** calculation type to build the expression.

The expression below will return Monday, Tuesday, Wednesday etc based on the target date field used.

The calculation expression will be inserted.



Calculation expression

Today.DayName

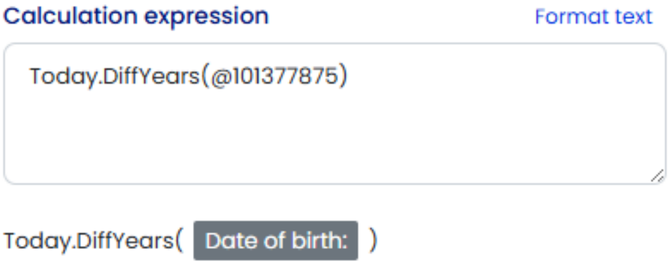
Today.DayName

5.1 Calculate Age

Create a number field and use the Calculation Tool. Select the **Date** data type and **DiffYears** calculation type to build the expression.

The expression below will return a number reflecting the age by comparing today's date with the date of birth, and then counting the years in-between. The calculation will determine whether the date has passed in the current year to return an accurate result.

The calculation expression will be inserted.



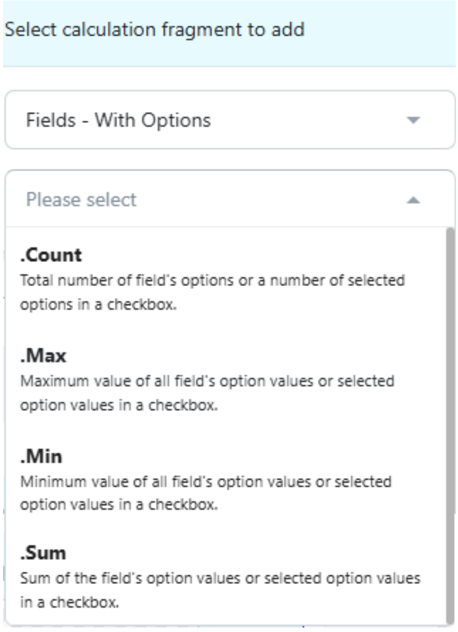
```
Today.DiffYears(@101377875)
```

Today.DiffYears(Date of birth:)

6. Fields with Select Options

Fields – With Options data types perform calculations based on responses to option fields, eg radio button, drop-down select and checkbox fields.

The preset calculations available for option fields are:



Select calculation fragment to add

Fields - With Options

Please select

.Count
Total number of field's options or a number of selected options in a checkbox.

.Max
Maximum value of all field's option values or selected option values in a checkbox.

.Min
Minimum value of all field's option values or selected option values in a checkbox.

.Sum
Sum of the field's option values or selected option values in a checkbox.

- Count
  Count the number of options available in a radio/select/checkbox list
  and/or
  Count the number of options selected in a checkbox multi select list

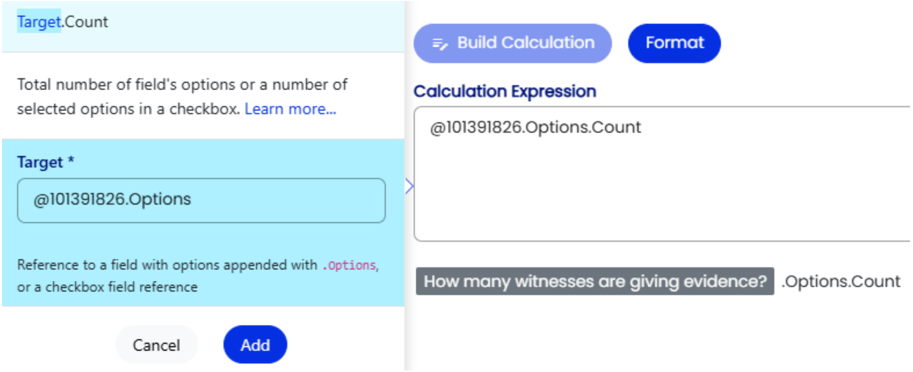
- Max
  Display the option with the maximum value in a radio/select/checkbox list
  and/or
  Return the value of the maximum option selected in a checkbox multi select list

- Min
  Display the option with the minimum value in a radio/select/checkbox list
  and/or
  Return the value of the minimum option selected in a checkbox multi select list

- Sum
  Calculate the sum of all options in a radio/select/checkbox list
  and/or
  Calculate the sum of the options selected in a checkbox multi select list

6.1 Count the Number of Options Available

Create a number field and use the Calculation Tool. Select the **Fields – With Options** data type and **Count** calculation type to build the expression.



Target.Count

Total number of field's options or a number of selected options in a checkbox. Learn more...

Target *

@101391826.Options

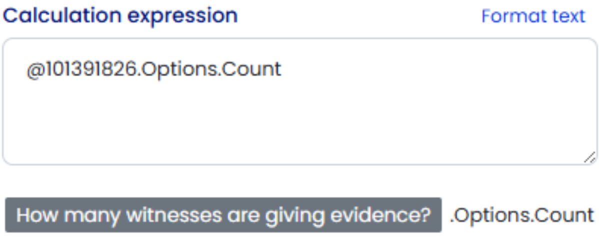
Reference to a field with options appended with `.Options`, or a checkbox field reference

Calculation Expression

@101391826.Options.Count

How many witnesses are giving evidence? .Options.Count

The calculation expression will be inserted.



Calculation expression

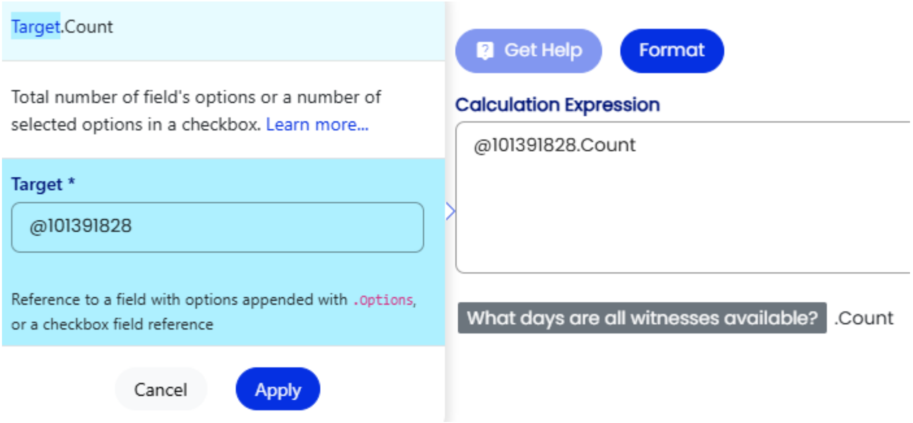
@101391826.Options.Count

How many witnesses are giving evidence? .Options.Count

6.2 Count the Number of Options the Form Filler Selected

Note: This calculation can only be performed on a checkbox field which is multi-select.

Create a number field and use the Calculation Tool. Select the **Fields – With Options** data type and **Count** calculation type to build the expression.



Target.Count

Total number of field's options or a number of selected options in a checkbox. Learn more...

Target *

@101391828

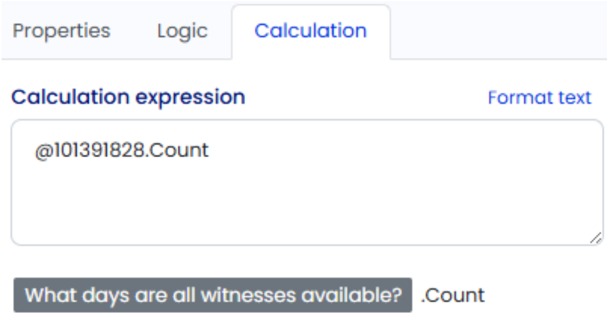
Reference to a field with options appended with `.Options`, or a checkbox field reference

Calculation Expression

@101391828.Count

What days are all witnesses available? .Count

The calculation expression will be inserted.

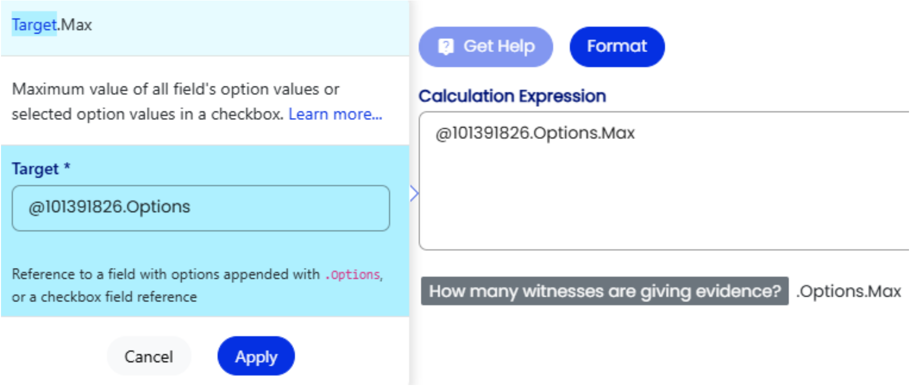


@101391828.Count

What days are all witnesses available? .Count

6.3 Identify the Maximum Value in a List

Create a number field and use the Calculation Tool. Select the **Fields – With Options** data type and **Max** calculation type to build the expression.



Target.Max

Maximum value of all field's option values or selected option values in a checkbox. Learn more...

Target *

@101391826.Options

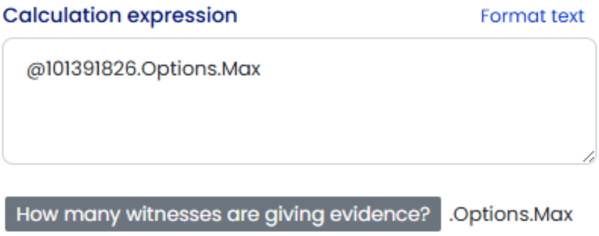
Reference to a field with options appended with .options, or a checkbox field reference

Calculation Expression

@101391826.Options.Max

How many witnesses are giving evidence? .Options.Max

The calculation expression will be inserted.

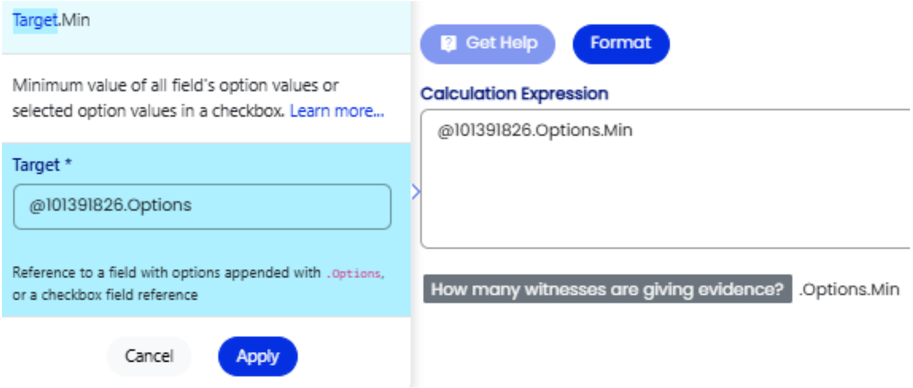


@101391826.Options.Max

How many witnesses are giving evidence? .Options.Max

6.4 Identify the Minimum Value in a List

Create a number field and use the Calculation Tool. Select the **Fields – With Options** data type and **Min** calculation type to build the expression.



Target.Min

Minimum value of all field's option values or selected option values in a checkbox. Learn more...

Target *

@101391826.Options

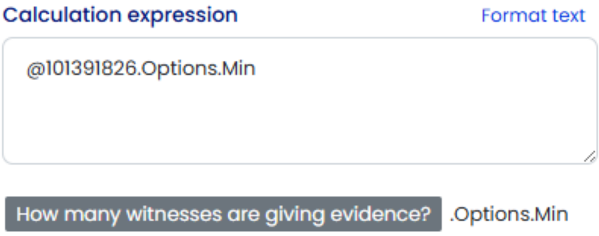
Reference to a field with options appended with `.Options`, or a checkbox field reference

Calculation Expression

@101391826.Options.Min

How many witnesses are giving evidence? .Options.Min

The calculation expression will be inserted.



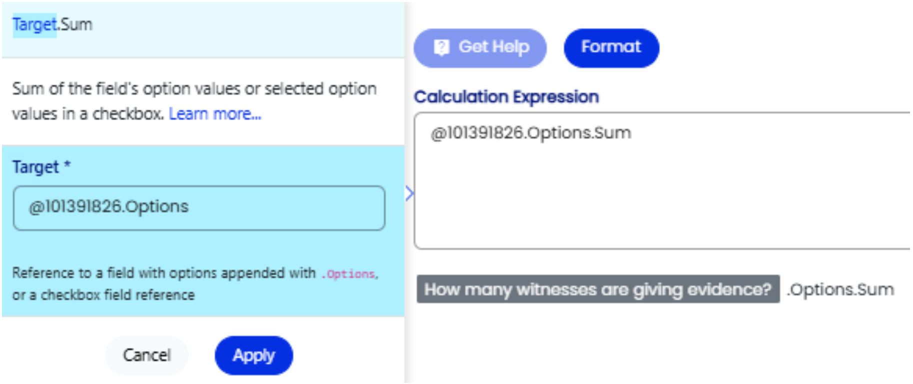
Calculation expression

@101391826.Options.Min

How many witnesses are giving evidence? .Options.Min

6.5 Calculate the Sum of List Values

Create a number field and use the Calculation Tool. Select the **Fields – With Options** data type and **Sum** calculation type to build the expression.



Target.Sum

Sum of the field's option values or selected option values in a checkbox. Learn more...

Target *

@101391826.Options

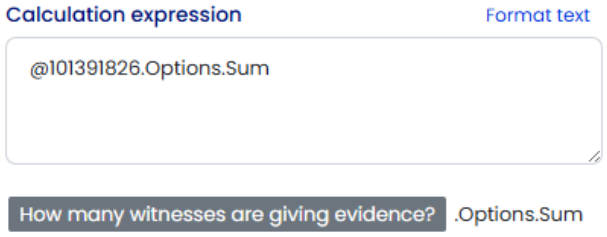
Reference to a field with options appended with `.Options`, or a checkbox field reference

Calculation Expression

@101391826.Options.Sum

How many witnesses are giving evidence? .Options.Sum

The calculation expression will be inserted.



Calculation expression

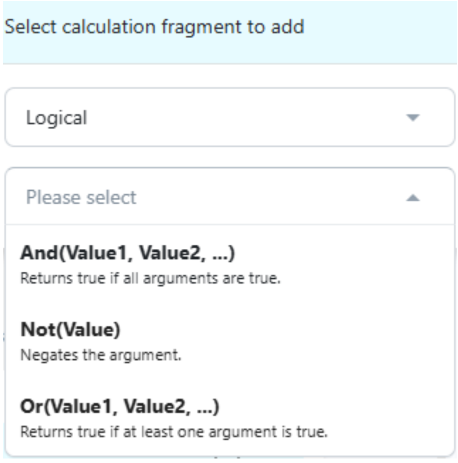
@101391826.Options.Sum

How many witnesses are giving evidence? .Options.Sum

7. Logical: AND - OR - NOT

Logical calculations check whether conditions have been met and return a true or false value.

The calculations can use AND - NOT – OR operators to build the conditions to be met.



Select calculation fragment to add

Logical

Please select

And(Value1, Value2, ...)
Returns true if all arguments are true.

Not(Value)
Negates the argument.

Or(Value1, Value2, ...)
Returns true if at least one argument is true.

These fields will check the specified conditions and:

If the conditions ARE MET - they will return 1 – meaning true.

If the conditions ARE NOT MET - they will return 0 – meaning false.

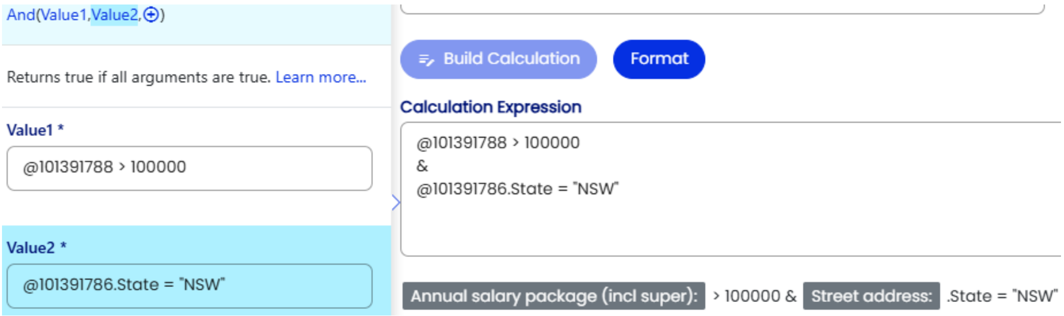
This type of response is called a boolean true/false field.

Note: By returning a 1 or 0, logical fields are easy to use in other calculations, logic and conditional rules by having a simple value to use in expressions.

7.1 Boolean Fields: True or False

This example is going to check if the contact earns over $100,000 AND lives in NSW.

Create a text field and use the Calculation Tool. Select the **Logical** data type and the **AND** calculation type to build the expression.



And(Value1,Value2,⊕)

Returns true if all arguments are true. Learn more...

Value1 *

@101391788 > 100000

Value2 *

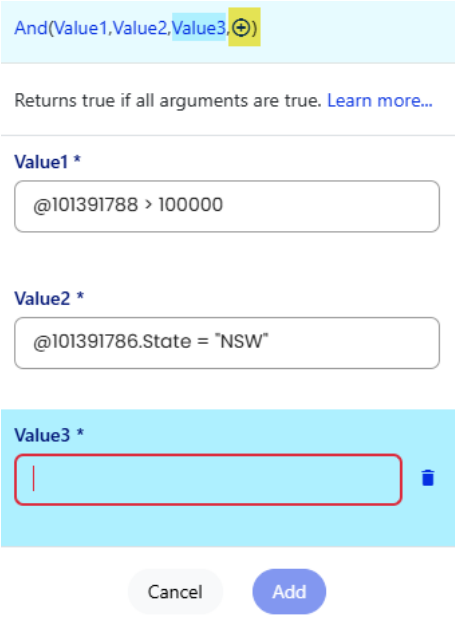
@101391786.State = "NSW"

Calculation Expression

```
@101391788 > 100000
&
@101391786.State = "NSW"
```

Annual salary package (incl super): > 100000 & Street address: .State = "NSW"

Add the conditions to be met. Click the + to add more conditions.



And(Value1,Value2,Value3,⊕)

Returns true if all arguments are true. Learn more...

Value1 *

@101391788 > 100000

Value2 *

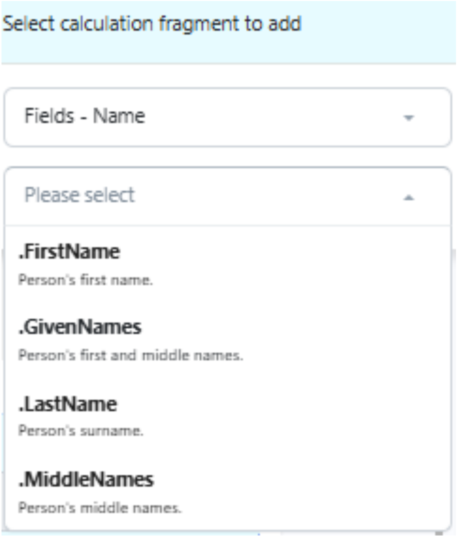
@101391786.State = "NSW"

Value3 *

Note: OR and NOT boolean fields are built as above, other than selecting the alternate calculation type.

8. Names

The preset calculations for names assist in extracting specific components of a name, e.g.:



Select calculation fragment to add

Fields - Name

Please select

.FirstName
Person's first name.

.GivenNames
Person's first and middle names.

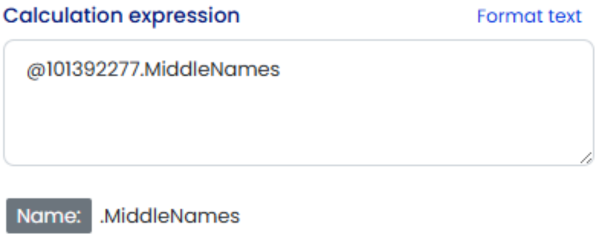
.LastName
Person's surname.

.MiddleNames
Person's middle names.

You can select the component of a name via the Word Addin to insert these details as a field in your template. You would only need to create a calculation to extract the middle or given names, which aren't available components in the Addin.

8.1 Extract the Middle Name

Create a text field and use the Calculation Tool. Select the **Name** data type and the **MiddleNames** calculation type. Insert the name field ID to build the expression.



Calculation expression

```
@101392277.MiddleNames
```

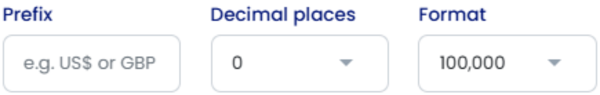
Name: .MiddleNames

9. Numbers

Calculations can be used to perform mathematics and output a number, but they can also check and convert data entered to produce a text response.

9.1 Rounding

Create a number field and set the decimal places to zero.

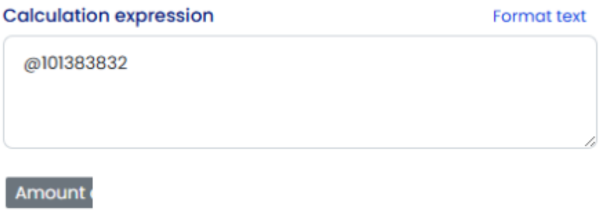


Prefix: e.g. US$ or GBP

Decimal places: 0

Format: 100,000

In the calculation, insert the field ID of the target number.



Calculation expression

@101383832

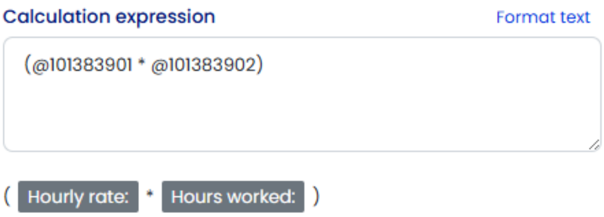
9.2 Convert to Words

Conversion to words takes 2 steps:

1. Create a number field that has no prefix and decimals set to nil.
   In the calculation, insert the field ID of the target number field to be converted.
2. Create a text field.
   In the calculation, insert the field ID of the new field created above.

9.3 Sums

Write the expression by setting the formula and inserting field IDs.



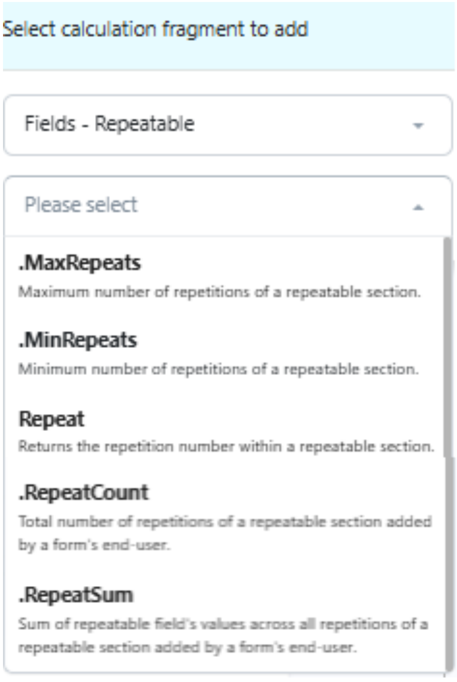
Calculation expression

(@101383901 * @101383902)

Note: The mathematical expressions in calculations are unlimited. Use the same formula styles as Excel.

10. Repeats

The preset calculations available for repeats are:



Select calculation fragment to add

Fields - Repeatable

Please select

.MaxRepeats
Maximum number of repetitions of a repeatable section.

.MinRepeats
Minimum number of repetitions of a repeatable section.

Repeat
Returns the repetition number within a repeatable section.

.RepeatCount
Total number of repetitions of a repeatable section added by a form's end-user.

.RepeatSum
Sum of repeatable field's values across all repetitions of a repeatable section added by a form's end-user.

- MaxRepeats
  Returns the value set in the repeating section as the maximum number of repeats the user can add

- MinRepeats
  Returns the value set in the repeating section as the minimum number of repeats the user must add

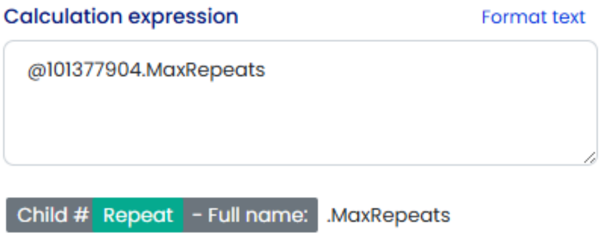
- Repeat
  Returns the repeat instance number of the repeat
  EG the first repeat will return 1, the second repeat will return 2 etc.

- RepeatCount
  Returns the number of repeats the user has entered

- RepeatSum
  Calculates the sum of the repeated field selected
  EG a repeating section to enter invoices issued with a field for the invoiced amount can calculate the total sum of all invoices entered

10.1 Return the Maximum Repeats set in the Repeating Section

Create a number field and use the Calculation Tool. Select the **Fields – Repeatable** data type and the **MaxRepeats** calculation type to build the expression.

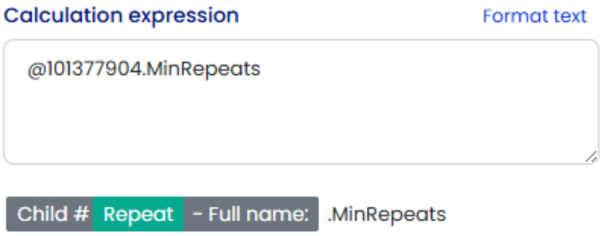


```
@101377904.MaxRepeats
```

Child # Repeat - Full name: .MaxRepeats

10.2 Return the Minimum Repeats set in the Repeating Section

Create a number field and use the Calculation Tool. Select the **Fields – Repeatable** data type and the **MinRepeats** calculation type to build the expression.

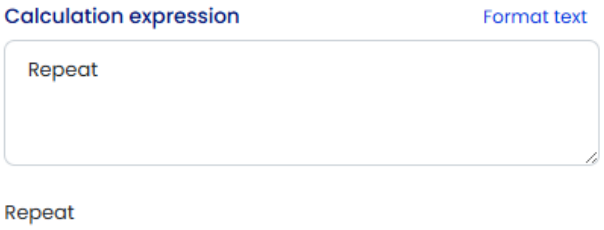


```
@101377904.MinRepeats
```

Child # Repeat - Full name: .MinRepeats

10.3 Return the Repetition Instance Number

Create a number field and use the Calculation Tool. Select the **Fields – Repeatable** data type and the **Repeat** calculation type to build the expression.

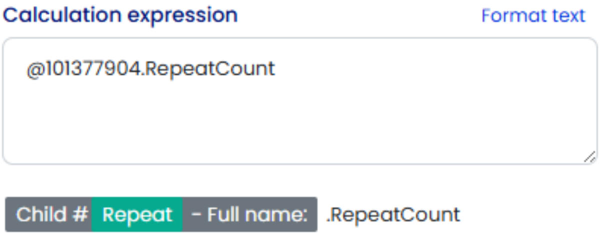


```
Repeat
```

Repeat

10.4 Count the Number of Repeats the User has Entered

Create a number field and use the Calculation Tool. Select the **Fields – Repeatable** data type and the **RepeatCount** calculation type to build the expression.

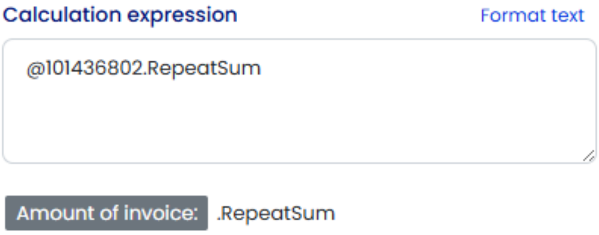


```
@101377904.RepeatCount
```

Child # Repeat - Full name: .RepeatCount

10.5 Add the Total Sum of a Repeatable Field

Create a number field and use the Calculation Tool. Select the **Fields – Repeatable** data type and the **RepeatSum** calculation type to build the expression.



```
@101436802.RepeatSum
```

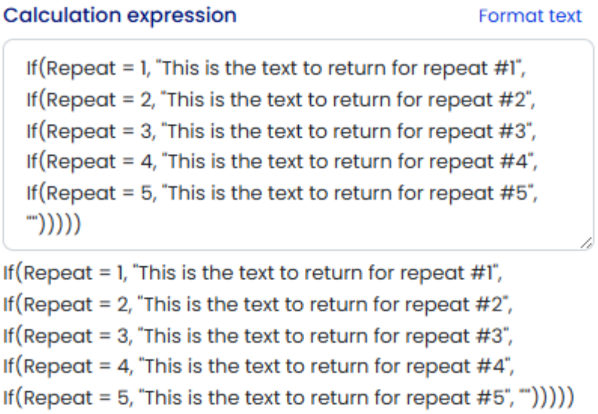
Amount of invoice: .RepeatSum

10.6 Calculation to do Something Different for Each Repeat

You can create a calc that will return a different result for each separate repeat, eg:

If this is repeat 1, do THIS
If this is repeat 2, do THIS instead … etc

The expression for this calculation is:



```
If(Repeat = 1, “This is the text to return for repeat #1”,
If(Repeat = 2, “This is the text to return for repeat #2”,
If(Repeat = 3, “This is the text to return for repeat #3”,
If(Repeat = 4, “This is the text to return for repeat #4”,
If(Repeat = 5, “This is the text to return for repeat #5”,
“”)))))
```

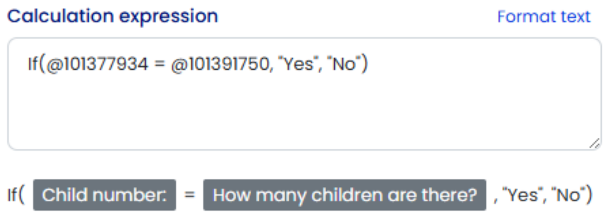
If(Repeat = 1, “This is the text to return for repeat #1”,
If(Repeat = 2, “This is the text to return for repeat #2”,
If(Repeat = 3, “This is the text to return for repeat #3”,
If(Repeat = 4, “This is the text to return for repeat #4”,
If(Repeat = 5, “This is the text to return for repeat #5”, “”)))))

10.7 Check if the Repeat is the Last Repeat

In this example:
Child number is a calculation that returns the specific instance, e.g. 1, 2, 3 etc
How many children are there? Is a calculation that returns the total number of children, e.g. 4

The expression checks IF the repeat instance is the same as the total, which will return YES to say that particular repeat is the last repeat.

This field is very useful when inserting the word ‘and’ between parties in court documents, deeds, agreements etc and can determine not to put the word ‘and’ after the last instance.



```
If(@101377934 = @101391750, "Yes", "No")
```

```
If( Child number: = How many children are there? , "Yes", "No")
```

10.8 Referring to specific repeat instances in a calculation

Calculations can look at repeated data collectively and also pinpoint a specific instance.

In this example, we will create a calculation that is going to look at a group of properties and determine what states they are in. We want the calculation to determine the unique states and exclude multiple references for the properties that are in the same state.

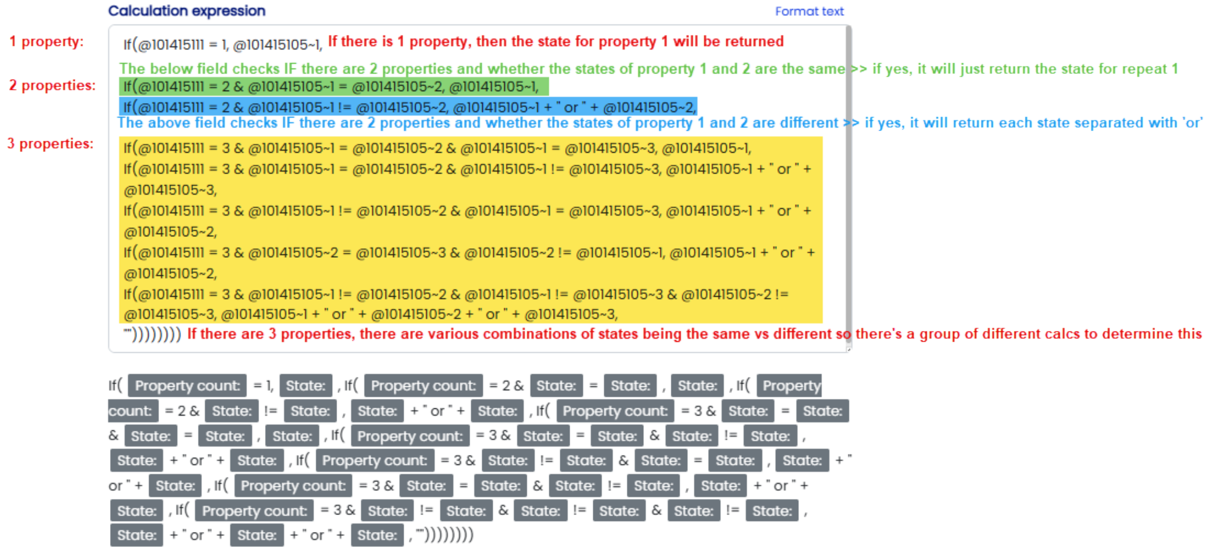
The repeatable field will allow for up to 3 properties to be entered. Depending on how many properties are added, it will calculate the unique values:

- If there is 1 property, then return the state of that property
- If there are 2 properties and they're both in NSW, then return 'NSW'
- If there are 2 properties - one in NSW + one in QLD - then return 'NSW and QLD'
- If there are 3 properties and they're all in VIC, then return 'VIC'
- If there are 3 properties - one in NSW + one in QLD + one in VIC - then return 'NSW, QLD and VIC' etc

Repeat instances are specified by adding a tilda and the repeat number to the end of the field ID number.
EG The property address field ID is "@101415104" but if we wanted to point to address number 2, the field ID would be "@101415104~2".

When referring to specific repeats it's important to build a safety net to ensure the specific repeat actually exists. In the example below, you'll see the first IF statement is checking for the number of properties to then determine how to proceed.

The calculation contains multiple IF statements to cater for the various possibilities and combination of states. Each IF statement first checks how many properties there are >> then compares the relevant states >> then returns the unique results.



Calculation expression

1 property:

```
If(@101415111 = 1, @101415105~1,
```

If there is 1 property, then the state for property 1 will be returned

2 properties:

The below field checks IF there are 2 properties and whether the states of property 1 and 2 are the same >> if yes, it will just return the state for repeat 1

```
If(@101415111 = 2 & @101415105~1 = @101415105~2, @101415105~1,
If(@101415111 = 2 & @101415105~1 != @101415105~2, @101415105~1 + " or " + @101415105~2,
```

The above field checks IF there are 2 properties and whether the states of property 1 and 2 are different >> if yes, it will return each state separated with 'or'

3 properties:

```
If(@101415111 = 3 & @101415105~1 = @101415105~2 & @101415105~1 = @101415105~3, @101415105~1,
If(@101415111 = 3 & @101415105~1 = @101415105~2 & @101415105~1 != @101415105~3, @101415105~1 + " or " +
@101415105~3,
If(@101415111 = 3 & @101415105~1 != @101415105~2 & @101415105~1 = @101415105~3, @101415105~1 + " or " +
@101415105~2,
If(@101415111 = 3 & @101415105~2 = @101415105~3 & @101415105~2 != @101415105~1, @101415105~1 + " or " +
@101415105~2,
If(@101415111 = 3 & @101415105~1 != @101415105~2 & @101415105~1 != @101415105~3 & @101415105~2 !=
@101415105~3, @101415105~1 + " or " + @101415105~2 + " or " + @101415105~3,
""))))))))
```

If there are 3 properties, there are various combinations of states being the same vs different so there's a group of different calcs to determine this

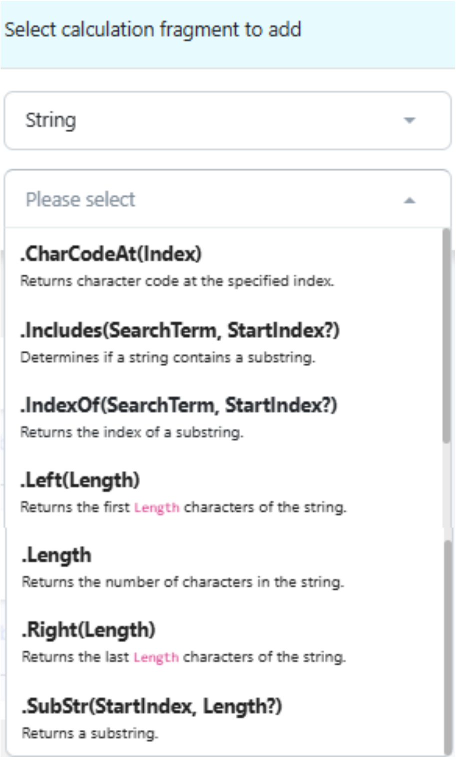
```
If( Property count: = 1, State: , If( Property count: = 2 & State: = State: , State: , If( Property
count: = 2 & State: != State: , State: + " or " + State: , If( Property count: = 3 & State: = State:
& State: = State: , State: , If( Property count: = 3 & State: = State: & State: != State: ,
State: + " or " + State: , If( Property count: = 3 & State: != State: & State: = State: , State: + "
or " + State: , If( Property count: = 3 & State: = State: & State: != State: , State: + " or " +
State: , If( Property count: = 3 & State: != State: & State: != State: & State: != State: ,
State: + " or " + State: + " or " + State: , ""))))))))
```

This example is quite complex and may be something to review when you are experienced and confident with calculations.

TIP: Ensure careful placement of commas and brackets when building complex calcs.

11. Strings / Text

The preset calculations available for strings are:



Select calculation fragment to add

String

Please select

.CharCodeAt(Index)
Returns character code at the specified index.

.Includes(SearchTerm, StartIndex?)
Determines if a string contains a substring.

.IndexOf(SearchTerm, StartIndex?)
Returns the index of a substring.

.Left(Length)
Returns the first Length characters of the string.

.Length
Returns the number of characters in the string.

.Right(Length)
Returns the last Length characters of the string.

.SubStr(StartIndex, Length?)
Returns a substring.

Note: “String” is term that means a series of characters/text.

11.1 IndexOf

The IndexOf() method finds a defined search term within a target string, and compares it to an index. IndexOf() will return the index of the first character of the search term's first occurrence within the target string or zero if the search term is not found. It will return zero when the search term is an empty string or a field that is unanswered or hidden by logic.

For example: A field in your form asks a form filler to list their assets. You could create an IndexOf calculation to identify if they have listed “real estate”.

The final syntax would look like this: @123456789.IndexOf("real estate") = 1.

We can continue to add other asset classes, and continue the hidden field attached to the calculation would be populated by the number one if the form filler entered the words “real estate”.

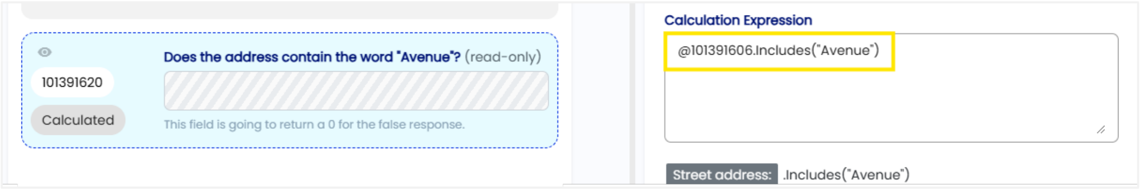
11.2 Includes

The Includes() method is very similar to the IndexOf() method. The only difference is that the Includes method returns 1 (representing true) if the search term is found within the target string, instead of the index of the search term itself.

Includes() will return zero (representing false) if the search term is not found, including when the search term is an empty string or a field that is unanswered or hidden by logic.

For example, in the calculation below, we are searching a field (ID: 101391606) which contains part of an address (Street or Avenue).

- The resulting expression: @101391606.Includes("Avenue") = 1.
- We could add a second calculation for a @101391606.Includes("Street") = 0.



101391620

Calculated

Does the address contain the word "Avenue"? (read-only)

This field is going to return a 0 for the false response.

Calculation Expression

@101391606.Includes("Avenue")

Street address: .Includes("Avenue")

11.3 CharCodeAt

The CharCodeAt() method returns a UTF-16 decimal code unit of the character at the specified index. For characters with codes between 0 and 127 UTF-16 code unit is the same as the ASCII code. For example, 'A' = 65, 'Z' = 90, 'a' = 97 and 'z' = 122.

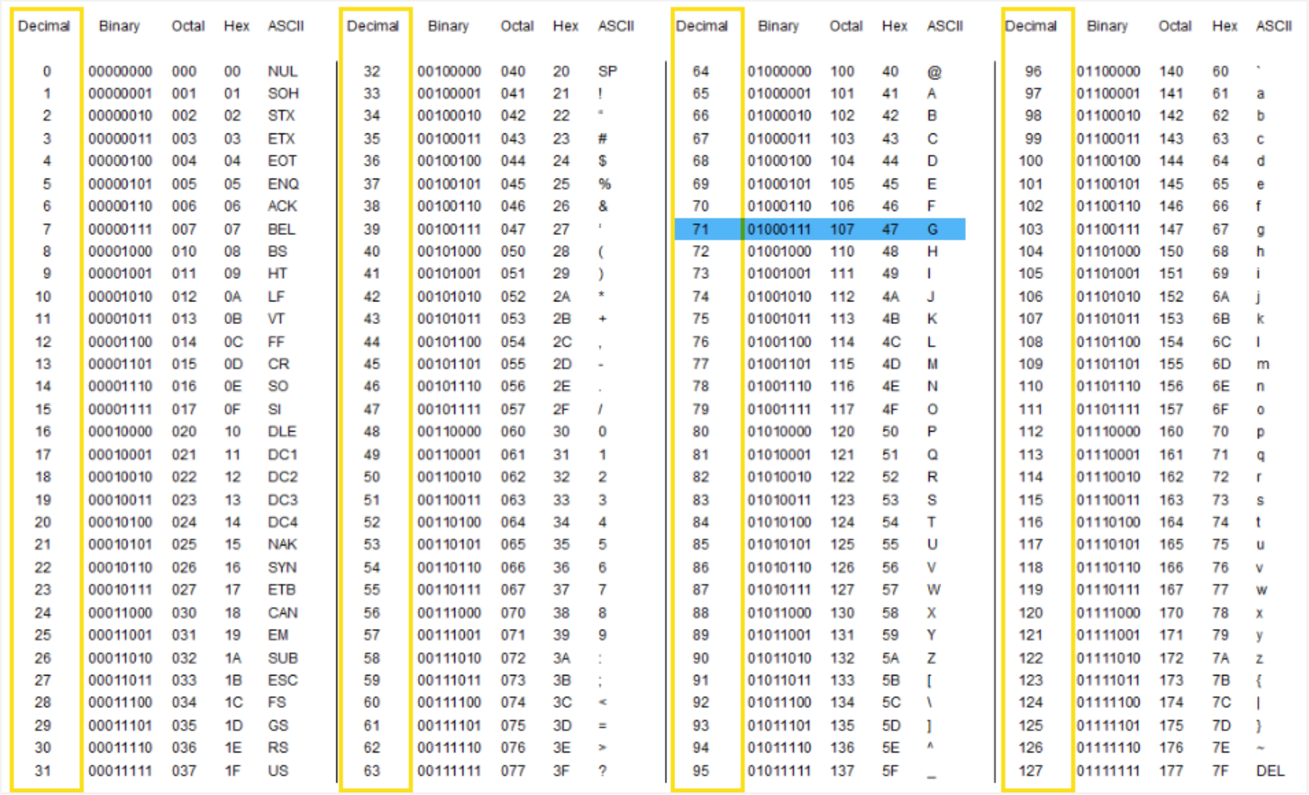
If the index is out of range or a field hidden by logic or unanswered, the method returns null. CharCodeAt() is useful for validating text field answers when a specified format is required – for example, a specific combination of letters and numbers or a slash '/' or dash '-' in a specific position.

For example: A field (ref: 101391605) captures a first name. The name entered is Gary.

- Syntax: @101391605.CharCodeAt(1)

The output in the calculation field = 71.

Here's the full code used:



Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

11.4 SubStr

The SubStr() method extracts a substring of up to a defined number of characters from the target string starting at a certain point. If the starting point is negative, the counting starts from the end of the string.

The length of the string is optional and if it is:

- not provided,
- exceeds actual string length, or
- is a field hidden by logic or unanswered.

All characters from the starting point to the end of the target string are returned.

For example: This string is going to return characters 10-12 of the address. The address is:
Level 2, 50 George Street, Parramatta NSW 2150

- Syntax: @101391606.SubStr(10, 2)
- Output: 50, because the string starting point is 10, and the string length is 2.

11.5 Left/Right

The **Left()** method returns the first number of characters of a string – as in, it starts on the **left**. If the number of characters is an unanswered field or is less than zero it's treated as if it was zero. If it exceeds the string's length, it's treated as if it was equal to the length.

For example: a field (id: 101391606) has captured an address and we want to return the first 7 characters of the address. The address is: **Level 2, 50 George Street, Parramatta NSW 2150**

- Syntax: @101391606.Left(7)
- Output: Level 2

The **Right()** method returns the last number of characters of a string – it starts on the **right**.

For example: a field (id: 101391606) has captured an address and we want to extract the last 4 characters of the address. The address is: **Level 2, 50 George Street, Parramatta NSW 2150**

- Syntax: @101391606.Right(4)
- Output: 2150

11.6 Length

Returns the strings length, i.e. the number of characters that make up the string.

For example: a field (id: 101391606) has captured an address and we want to return the length of the address. The address is: **Level 2, 50 George Street, Parramatta NSW 2150**

- Syntax: @101391606.Length
- Output: 46

12. Writing Expressions

The ability to write complex expressions is not a requirement of building calculations. If you are an experienced automator or coder then you may be able to apply that knowledge and write syntax rather than use the builder.

Smarter Drafter can interpret HTML though we don’t recommend writing your expressions and applying HTML commands.