APRA Connect

Expression Functions Guide

Version 1.0 (released June 2021)

Contents

Overview

Information on the expression functions should be used in conjunction with the published taxonomy artefacts.

Expression functions exist to execute standard logical/mathematical/date/string functions. Expressions typically have a *context* in which they run. For example, a validation rule that is applied to items within a Schema (e.g. A > B + C), will have that Schema as its context. This enables the Expression to easily reference local Schema Instance data. A validation Expression of "*[*A] > (*[B] + [C])*" uses square brackets to indicate a data reference.

A Schema represents a self-contained hierarchical data model.

Expression syntax is also straightforward and based loosely on the Excel formula syntax. Functions are used in the same way as Excel with the function name being followed by a parenthesis and a list of comma-separated arguments, e.g. 'Round([FixedAssets], 1)'. The arguments can themselves be data or the result of other executed functions.

Local data references use square brackets with the name of the Item to be referenced (for example "[FixedAssets]"). Non-contextual, non-local data, such as properties of the regulated entity or values from other returns, use an extended syntax which first specifies the Schema (or return) and then the data within the return. For example:

[@schema=BalanceSheet, @item=FixedAssets]

Standard operators can also be used in Expressions, such as +, -, *, ^, and <, along with constants (static values such as '3') for all data types.

Logical Operators for Expression Functions

The table below describes the logical operators that are used in APRA Connect.

Name	Return Type	Description
=	Boolean	**A=B** Check if value A is equal to value B. A and B can be replaced with a constant, expression or Schema Item of the following types: • Number • String • Boolean • Date • Enumeration Option A and B must be of the same data type.
<	Boolean	**A<B** Check if value A is less than value B. A and B can be replaced with a constant, expression or Schema Item of the following types: • Number • Date A and B must be of the same data type.
<=	Boolean	**A<=B** Check if value A is less than value B. A and B can be replaced with a constant, expression or Schema Item of the following types: • Number • Date A and B must be of the same data type.
>	Boolean	**A>B** Check if value A is greater than value B. A and B can be replaced with a constant, expression or Schema Item of the following types: • Number • Date A and B must be of the same data type.

Name	Return Type	Description
>=	Boolean	**A>=B** Check if value A is greater than or equal to the value B. A and B can be replaced with a constant, expression or Schema Item of the following types: • Number • Date A and B must be of the same data type.
and	Boolean	**and(A,B...)** This expression returns true if all arguments are true. The expression returns false if any of the arguments are false.
	Boolean	**A and B** The above expression demonstrates an alternative way of using the **and** operator. This expression returns true if all arguments are true. The expression returns false if any of the arguments are false.
If	Any	**If(A, B, C)** Checks whether a logical condition has been met, and returns one value if true and another if false. In the above signature, A represents the condition which must return either true or false (i.e. a Boolean value). If condition A evaluates to true, the resulting value of expression B is returned. If condition A evaluates to false, the resulting value of expression C is returned. Expressions B and C may be a function returning any data type or may be specific values to be returned such as strings or numbers. **Note:** Either expression B or C may be replaced by another If expression, e.g. **If (A, B, If(A, B, C))** In the above example, if condition A evaluates to true, the result of expression B is returned. However, if condition A evaluates to false, a further If expression may be evaluated.
not	Boolean	**not(A)** This expression changes the value of A from false to true, or from true to false. Both A and B must return a Boolean value.
or	Boolean	**or(A,B...)** This expression returns true if any one of the arguments (A, B etc.) is true, otherwise false.
	Boolean	**A or B** An alternative way of using the **or** operator. This expression returns true if any one of the arguments (A, B etc.) is true, otherwise false.

Name	Return Type	Description
false	Boolean	**false** This expression returns the logical value false. This expression may also be usefully employed as a part of a schema validation rule in order to ensure the associated rule fails if the condition returns false. **If (A, B, false)** In the above example, if condition A returns true, then expression B is evaluated with a value returned. If condition A returns false, the value false will be returned which can be used to fail a validation rule.
true	Boolean	**true** This expression returns the logical value true. This expression may also be usefully employed as a part of a schema or form validation rule in order to ensure the associated rule passes if the condition returns true. **If (A, true, C)** In the above example, if condition A returns true, then the whole expression evaluates to true and can be used to pass a schema or form validation rule. If condition A returns false, the expression C will be evaluated.

Date Expression Functions

The table below includes date related expression functions that are used in APRA Connect.

Name	Return Type	Description
AddDays	Date	**AddDays(date, days)** Returns a date with the specified number of days added to it. If a negative number of days is passed in, this number of days will be subtracted from the date i.e. a date in the past will be returned. **Example:** To add 5 days to the reporting end date: AddDays([ReportingEndDate], 5)
AddMonths	Date	**AddMonths(date, months)** Returns a date with a specified number of months added to it. If a negative number of months is passed in, this number of months will be subtracted from the date i.e. a date in the past will be returned. This may be used to get the date of a prior period. **Example:** To get the previous quarter, the expression would be: GetLastDayOfMonth(AddMonths([ReportingEndDate], -3))
AddYears	Date	**AddYears(date, years)** Returns a date with a specified number of years added to it. If a negative number of years is passed in, this number of years will be subtracted from the date i.e. a date in the past will be returned. This may be used to get the date of a prior period. **Example:** To get the same date for the previous year, the expression would be: AddYears([ReportingEndDate], -1)
AddBusinessDays	Date	**AddBusinessDays(startDate, daysToAdd)** Returns a date with the specified number of business days added to it. Similar to AddDays except that it will skip weekends and any days that have been defined in the non_business_days database table. If a negative number of days is passed in, this number of days will be subtracted from the date i.e. a date in the past will be returned. **Example:** AddBusinessDays(Now(), 3) If Now() is the Monday 24th November, with the 25th and 26th as non-business days, then because Saturday 29th and Sunday 30th are weekend days, then Monday 1st December is returned.

Name	Return Type	Description
AddBusinessDays	Date	**AddBusinessDays(startDate, daysToAdd, administrativeDivision)** Returns a date with the specified number of business days for the given administrative division (e.g. region, state, province, country) added to it. Similar to above, except that in addition to skipping weekends and any days that have been defined in the non_business_days database table, it will skip non-business days for the given administrative division. If a negative number of days is passed in, this number of days will be subtracted from the date i.e. a date in the past will be returned. **Example:** AddBusinessDays(Now(), 3, 'Ireland') If Now() is the Monday 24th November, with the 25th and 26th as non-business days for all divisions, with Ireland having an additional non-business day of 1st December, then because Saturday 29th and Sunday 30th are weekend days, then Monday 2nd December is returned.
Date	Date	**Date(year, month, day)** Returns a date containing the given year, month and day. **Example:** Date(2020,12,31) will return the date 31st December 2020.
Day	Number	**Day(date)** Returns the day part of the provided date (1-31). **Example:** Day(Date(2020,12,31)) will return 31 as the day part of 31st December 2020.
GetDatesInRange	Array[Date]	**GetDatesInRange(fromDate, toDate, includeWeekends)** Returns an array of Dates between the specified to and from dates. If the Boolean parameter includeWeekends is false, then dates that fall on a weekend will be excluded.
GetLastBusinessDayOfMonth	Date	**GetLastBusinessDayOfMonth(initialDate)** Gets the last business day of a month by retrieving a list of non-business days and also weekends for a particular month. This function works by getting the last day of the month first and then computing from the last day of month whether it is a business day. If it is a business day, it returns that as the last business day. Otherwise, it goes to the previous business day and returns that day as the last business day of the month.

GetLastDayOfMonth	Date	**GetLastDayOfMonth(date)**
		Returns the date of the last day of the month for the given date, e.g. if 10/12/2010 was passed in, the return value would be 31/12/2010.
		Example:
		GetLastDayOfMonth([ReportingEndDate])

Name	Return Type	Description
GetMonthName	String	**GetMonthName(month)** Gets the translated name of a particular month from a specified number i.e. in an English culture 1 would return January, 12 would return December etc.

GetMonthsAndYearsInRange	Array[Date]	**GetMonthsAndYearsInRange(fromDate, toDate)**
		Returns an array of the dates including the first day of each month and year between the specified dates.
		Example:
		GetMonthsAndYearsInRange([PeriodStartDate],[PeriodEndDate])

GetQuarter	Number	**GetQuarter(date)** Returns the Quarter (Quarter 1 to 4) as a number, to which a valid date provided belongs. This is based on the calendar year, i.e. 1 from January-March, 2 from April-June, 3 for JulySeptember and 4 for October-December. **Example:** GetQuarter([ReportingEndDate]) If the item ReportingEndDate has a value of the 31st July 2015, then 3 will be returned as the reporting end date falls in the 3rd quarter.
GetQuartersInRange	Array[Date]	**GetQuartersInRange(fromDate, toDate)** Returns an array of end of quarter Dates between the specified to and from dates. This is based on the calendar year, returning 31st March, 30th June, 30th September or 31st December as end dates for a quarter if they fall within the specified dates. **Example:** GetQuartersInRange([PeriodStartDate],[PeriodEndDate]) If the item PeriodStartDate has a value of the 1st January 2020, and PeriodEndDate has a value of 1st January 2021, 4 dates will be returned: • 31st March 2020 • 30th June 2020 • 30th September 2020 • 31st December 2020
GetYearsInRange	Array[Number]	**GetYearsInRange(fromDate, toDate)** Returns an array of years between the given dates, inclusive of both years. **Example:** GetYearsInRange([PeriodStartDate],[PeriodEndDate]) If the item PeriodStartDate has a value of the 1st January 2020, and PeriodEndDate has a value of 1st January 2021, 2 values will be returned: 2020 and 2021.

Name	Return Type	Description
MaxOfListDateValues	Date	**MaxOfListDateValues(A)** Returns the latest date from an array of date values.
MaxA	Date	**MaxA(dateArray)** Returns the latest value in an array of dates.
MinA	Date	**MinA(dateArray)** Returns the earliest value in an array of dates.
Month	Number	**Month(date)** Returns the month part of the provided date (1-12). **Example:** Month([ReportingEndDate]) If the item ReportingEndDate has a value of 1st October 2015, then 10 will be returned.
Now	Date	**Now()** Returns the current date with the time component set to 00:00:00.
Year	Number	**Year(date)** Returns the year part of the provided date (e.g. 1980) **Example:** Year([ReportingEndDate]) If the item ReportingEndDate has a value of 1st October 2015, then 2015 will be returned.
UTCOADate	Number	**UTCOADate()** Returns the current UTC date and time using the Microsoft OADate decimal format. An OLE Automation date is implemented as a floating-point number whose integral component is the number of days before or after midnight, 30 December 1899, and whose fractional component. represents the time on that day divided by 24. **Example:** UTCOADate() 7th July 2018 1:34:32 PM returns 43286.5656488773

Dimensional Data Expression Functions

The following table describes dimensional data expression functions that are in APRA Connect

Name	Return Type	Description
InitialiseListValues	N/A	**InitialiseListValues(splitCharacter, A)** Initialises the items in list Split Character - Character used to split the columns in each row A - The input is an array of strings. MakeStringArray function needs to be used to provide an array of strings where each string will have the list item values separated by the splitCharacter. For example, if a list has 3 primary keys - Region, Currency, Department, 3 rows of initial values can be specified using the below expression: InitialiseListValues(',', MakeStringArray('EU, EUR, Agriculture;EU, EUR, Textiles; EU, EUR, Others',';'))
IsValidAggregate	Boolean	**IsValidAggregate(valuesItemName, totalItemName, totalKey, aggregateKeys, percentThreshold, valueThreshold, [@context_instance])** Validates a dimensional aggregation, where one value of the "total" item, should equal the sum of a set of other item values in the list. The items for summation and comparison are given by the parameters "valueItemName" and "totalItemName". The key value for matching the dimensional total is given in “totalKey” and key values for matching dimensional disaggregated details are given in the array “aggregateKeys”. Finally, the two totals are compared based on the percentage or value thresholds specified by “percentThreshold” and “valueThreshold” parameters.
GetRowNumber	Number	**GetRowNumber([@Repeat_Group_Instance_Guid])** This function retrieves the current row number from within a list. This expression function can only be used on a list.

Name	Return Type	Description
Rollup	Number	**Rollup(numericItem, enumItem, EnumerationName, EnumKey)** Allows the aggregation of dimensional data by enumeration. For example, in a dimensional form where values of numeric dimensions, the Rollup expression can be used to aggregate by the specific dimensions e.g. country, currency. This function filters the rows of a List, based on the value of 'enumItem', and calculates the sum of 'numericItem' values of the filtered rows. A filter is passed when the enumItem's value is either 'EnumerationKey' if 'EnumerationKey' is a leaf in the enumeration or one of its leaf childs in the hierarchy of the enumeration otherwise. Only leaf nodes of the enumeration will be aggregated (Elements of the enumeration with children nodes will not be taken in account). Note that EnumerationKey refers to the Key of the enumeration instead of its value: if your Enumeration data type has different key and value elements (i.e.: Key='1', Value='Ireland') you will need to specify the Key ('1'). 'numericItem' and 'enumItem' must belong to the same List, or to descendant Groups of the same List. They cannot belong to different Lists, even when one of the Lists is a descentant of the other. 'enumItem' must be an enumeration marked as Primary Key of the list. The 'EnumerationName' should be the name of the data type of 'enumItem'. It is mandatory to use one filter per dimension where applicable. This means all enumeration items that are a Primary Key of the list must explicitly be filtered. Use the 'Root' keyword if you do not want to filter by any particular dimension, but include the filter in the Rollup call anyway (the order is not relevant). Only the rows that pass all filters will be aggregated in the final sum. Different filters must be applied using different enumeration items. Rollup([all numericItem], [all enumItem1], 'EnumerationName1', 'EnumerationKey1', [all enumItem2], 'EnumerationName2', 'EnumerationKey2')

Enumeration Expression Functions

The table below includes enumeration related expression functions that are used in APRA Connect:

Name	Return Type	Description
GetEnumeration Options	Array[Enum]	**GetEnumerationOptions([@data_types], name)** Returns an array containing all the options (key, value) for the enumeration with the given name. **[@data_types]** is a keyword that should be passed into the expression exactly as is.
		GetEnumerationOptions([@data_types], name, excludeKeys) Returns an array containing all the options (key, value) for the enumeration with the given name except for the options in the given array of keys to exclude. **Example:** GetEnumerationOptions([@data_types], ‘Currency’, MakeStringArray(‘EUR,GBP’,’,’)) This will return all options from the ‘Currency’ enumeration data type, excluding ‘EUR’ and ‘GBP’. [@data_types] is a keyword that should be passed into the expression exactly as is.
GetEnumeration OptionsSpecified	Array[Enum]	**GetEnumerationOptionsSpecified([@data_types], name, specifiedKeys)** Returns an array containing all the options for the enumeration with the given name. **Example:** GetEnumerationOptionsSpecified([@data_types], ‘Currency’, MakeStringArray(‘EUR,GBP,AUD,CAD’,’,’)) Given an Enumeration Data Type called 'Currency' exists in the project having Options with each one of the listed keys, this will return an array with only the enumeration options ‘EUR’, ‘GBP’, ‘AUD’ and ‘CAD’ extracted from the ‘Currency’ enumeration data type. **[@data_types]** is a keyword that should be passed into the expression exactly as is.
GetKey	String	**GetKey(Enumeration)** Returns the key of an enumeration option.
GetText	String	**GetText(Enumeration)** Returns the text of an enumeration option.
Option	Enum option	**Option([@data_types],name,key)** Returns an enumeration option with the given name and key. **[@data_types]** is a keyword that should be passed into the expression exactly as is.

Lookup & Reference Expression Functions

The following table describes lookup & reference functions that are in APRA Connect:

Name	Return Type	Description
BuildBoolForeignReference	Foreign Reference	**BuildBoolForeignReference(stringPath, [@context_instance])** Returns a foreign reference to a boolean schema item based on the path passed in. Used to dynamically build up a reference to another schema element based on a string you enter and/or text entered by users in runtime. **Example:** BuildNumberForeignReference('[@schema=SP @item=root/Section1/'+[parameterItemText], [@context_instance]) If the value of [parameterItemText] is a string value 'PassportReveived', this expression will return a foreign reference to the boolean item [PassportReceived] within the SP schema under the root group Section1. **Note:** [@context_instance] is a keyword parameter that must match the exact syntax.
BuildDateForeignReference	Foreign Reference	**BuildDateForeignReference(stringPath, [@context_instance])** Similar to BuildBoolForeignReference, but can be used to return a foreign **Date** schema item.
BuildEnumForeignReference	Foreign Reference	**BuildEnumForeignReference(stringPath, [@context_instance])** Similar to BuildBoolForeignReference, but can be used to return a foreign **Enumeration** schema item.
BuildNumberForeignReference	Foreign Reference	**BuildNumberForeignReference(stringPath, [@context_instance])** Similar to BuildBoolForeignReference, but can be used to return a foreign **Number** schema item.
BuildStringForeignReference	Foreign Reference	**BuildStringForeignReference(stringPath, [@context_instance])** Similar to BuildBoolForeignReference, but can be used to return a foreign **String** schema item.
BuildListForeignReference	Foreign Reference	**BuildListForeignReference(stringPath, [@context_instance])** Similar to BuildGroupForeignReference, but can be used to return a foreign schema List.

Name	Return Type	Description
Contains	Boolean	**Contains(listItem, value)** Determines if parameter 'value' is contained within the array of items from a list. List item and value must be of the same base data type.
BuildGroupForeignReference	Foreign Reference	**BuildGroupForeignReference(stringPath, [@context_instance])** Similar to BuildBoolForeignReference, but can be used to return a foreign schema Group. **Example:** ElementExists(BuildGroupForeignReference('[@schema=SP @item=root/'+[parameterGroupText], [@context_instance])) If [parameterGroupText] is 'Section1', this expression will check if the root/Section1 Group element exists within the foreign schema called SP.
ContainsFilter	Boolean	**ContainsFilter(listItem, item, operator, value)** Determines if parameter 'value' is contained within the array of items from a list compared against an item. Both 'listItem' and 'item' must be local items. This function also allows specifying an *operator* other than the default ('=') while comparing the value. It is possible to filter by multiple item-operator-value pairs by adding more the pairs at the end (E.g: Filter([all myItem], [itemA], valueA, [itemB], '<', valueB, [itemC], '>', values).
ElementExists	Boolean	**ElementExists(SchemaElement)** This function checks if a Schema element (Group, Item or List) exists; in the case of an Item element it checks that the Item exists AND has a value. This function can be used in Schema validation rules to determine whether the rule should be run or not. The 'element' parameter can reference an element in the local Schema or a foreign Schema.
ElementExists	Boolean	**ElementExists(endDate, firmId, foreignParameter)** This function returns true or false if an element(Group, Item or List) exists for the specified reporting end date for the specified firm.
ElementExists	Boolean	**ElementExists(endDate, firmId, foreignParameter, filterByForeignParameter, filterByValue)** This function returns true or false if an element(Group, Item or List) exists under a List for the specified reporting end date for the specified firm. Replace 'filterByForeignParameter' with the Item you would like to filter by and replace 'filterByValue' with a value you would like to match it to. To add multiple filter parameters, just add more filter parameter\value pairs.

Name	Return Type	Description
Filter	Any	**Filter(listItem, item, value)** This function filters items in a list where another item has a specified value. The filter item can be the same item, a different item in the same list, or somewhere else in the Schema entirely. Both items must be in the same Schema. It is possible to filter by multiple item-value pairs simply by adding more at the end, e.g. Filter(listItem, item, value, item2, value2,...)
Filter	Any	**Filter(listItem, item, operator, value)** This function filters items in a list where another item has a specified value. The filter item can be the same item, a different item in the same list, or somewhere else in the Schema entirely. Both items must be in the same Schema. This derivation of the function allows you to specify an operator other than the default '='. Examples of valid operators include '=', '<' and '>'. It is possible to filter by multiple item-operator-value pairs simply by adding more at the end, e.g. Filter(listItem, itemA, valueA, itemB, '<', valueB, itemC, '>', valueC)
ExecuteSQL	2-D Array of Any type (except file)	**ExecuteSQL(selectStatement, parameter1, parameter2...)** Executes a SQL SELECT statement 'selectStatement' against the database to return data from submitted returns or from entity profiles. Familiarity with SQL and the database structure is required to use this function. The first parameter, 'selectStatement', is a string which contains basic SQL keywords and may contain local, foreign or shortcut parameter references to Schema elements. These Schema element references will be converted to SQL before executing the 'selectStatement'. To refine the data being returned, any number of placeholders {0}, {1}, {2} etc. are allowed in the 'selectStatement' to be replaced by the parameters passed into ExecuteSQL.
GetCurrentLanguage	String	**GetCurrentLanguage()** Returns a string representing the current culture's short name (eg. 'en-GB')
GetDataValue	Any	**GetDataValue(foreignParameter)** This function returns a single value from another schema and is commonly used to run cross-formset validation rules. Where the value being sought is in another schema in the current submission pack, only the foreignParameter parameter is needed.
GetDataValue		**GetDataValue(endDate, firmID, foreignItem)** This function returns a single value from another schema and is commonly used to run cross-time or cross-return validation rules. Where the value being sought is from a separate return and/or time period, use the endDate and firmID to uniquely identify where to get the value from.

Name	Return Type	Description
GetDataValue	Any	**GetDataValue(endDate, firmID, categoryIDs, foreignParameter)** This function returns a single value from another schema and is commonly used to run cross-time or cross-return validation rules. Where the value being sought is in another schema in the current submission pack, only the foreignParameter parameter is needed. Where the value being sought is from a separate return and/or time period, use the endDate and firmID (and optionally the categoryIDs) to uniquely identify where to get the value from.
GetDataValue	Any	**GetDataValue(endDate, firmId, categoryIDs, foreignParameter, filterByForeignParameter, filterByValue)** This function returns a single value from another schema, filtered by another value, and is commonly used to run cross-time or cross-return validation rules. The Item referenced by 'foreignParameter' should be the value to return, the results of which are filtered by the filter parameters. Replace 'filterByForeignReference' with the Item you would like to filter by and replace 'filterByValue' with a value you would like to match. To use multiple filter parameters, use the AND operator.
GetDataValues	Array[Any]	**GetDataValues(endDate, foreignParameter)** This function returns a value for the specified end date for all firms. The value to return should replace 'foreignParameter' and the 'endDate' should be replaced by the specified date.
GetDataValues	Array[Any]	**GetDataValues(endDate, firmIds, foreignParameter)** This function returns a value for the specified end date for the List of firm ids. The value to return should replace 'foreignParameter', the 'endDate' should be replaced by the specified date and 'firmId' should be replaced with a List of firms.
GetDataValues	Array[Any]	**GetDataValues(endDate, foreignParameter,filterByForeignParameter, filterByValue)** This function returns multiple values for the specified end date and filter parameters for all firms. The Item referenced by 'foreignParameter' must be in a List, the results of which are filtered by a filter parameter\value pair. Replace 'filterByForeignParameter' with the Item you would like to filter by and replace 'filterByValue' with a value you would like to match it to. To add multiple filter parameters, just add more filter parameter\value pairs.

GetEnvironment Setting	String	**GetEnvironmentSetting(environmentSetting)**
		Returns a string value for the specified setting in InReg.config.
		Returns an empty string if the setting is not found. Example:
		GetEnvironmentSetting('ExternalWebsiteUrl')

Name	Return Type	Description
GetProfileValue	Any	**GetProfileValue(firmId, foreignParameter)** A function to retrieve a piece of data from another Schema instance. The first parameter 'firmId' is used to find the Schema instance. The second parameter 'foreignParameter' is the Item whose value is returned from the Schema instance. The Item referenced by 'foreignParameter' cannot be in a List.
GetProfileValue	Any	**GetProfileValue(firmId, ForeignParameter,filterByForeignParameter, filterByValue)** A function to retrieve a piece of data from another Schema instance. The first parameter 'firmId' is used to find the Schema instance. The second parameter 'foreignParameter' is the Item whose value is returned from the Schema instance. The Item referenced by 'foreignParameter' must be in a List, the results of which are filtered by the two filter parameters 'filterByForeignReference' and 'filterByValue'.
GetProfileValues	Array[Any]	**GetProfileValues(firmId, foreignParameter)** This function returns multiple values for the specified firm for the and filter parameters. The Item referenced by 'foreignParameter' must be in a List, the results of which are filtered by a filter parameter/value pair. Replace 'filterByForeignParameter' with the Item you would like to filter by and replace 'filterByValue' with a value you would like to match it to. To add multiple filter parameters, just add more filter parameter/value pairs.
GetProfileValues	Array[Any]	**GetProfileValues(firmId, foreignParameter,filterByForeignParameter, filterByValue)** This function returns multiple values for the specified firm for the and filter parameters. The Item referenced by 'foreignParameter' must be in a List, the results of which are filtered by a filter parameter/value pair. Replace 'filterByForeignParameter' with the Item you would like to filter by and replace 'filterByValue' with a value you would like to match it to. To add multiple filter parameters, just add more filter parameter/value pairs.

Mathematical & Statistical Expression Functions

The table below includes expression functions related to enumeration data types that are in APRA Connect

Name	Return Type	Description
^	Number	**A^B** Exponentiation – returns A raised to the power of B.
+	Number	**A + B** Adds one number (A) to another (B).
-	Number	**A - B** Subtracts one number (B) from another (A).
/	Number	**A / B** Divides the number A by the number B. Division by zero returns null so if it is a possibility and requires handling, wrap the expression in an Nz i.e. Nz(A/B, 0) will return 0 if B is 0
*	Number	**A * B** Multiplies one number (A) by another number (B).
Absolute	Number	**Absolute(number)** Returns the absolute value of the number provided.
Avg	Number	**Avg(numberArray)** Returns the average of an array of numbers.
Count	Number	**Count(ListItem)** Counts the number of instances in a List or number of items in an Array. **Example:** A list [Countries] exists containing an enumeration item called [Country]. The following expression will return the number of **Note:** File types are not supported.
CountNonNullValues	Number	**CountNonNullValues(ListItem)** Counts the number of non-null value instances in a List or number of non-null items in an Array. **Note:** Only Date, Number and String types are supported. Boolean, Enumeration and File are not supported.

Name	Return Type	Description
EqualWithinThreshold	Boolean	**EqualWithinThreshold(firstValue, secondValue, percentageThreshold, numericThreshold)** Returns true if the first value is equal to, within a certain tolerance of, the second value. The first value must be within one or the other of these thresholds: • the third parameter is a percentage amount (e.g. 5 means the firstvalue must be within a range +/-5% on the second value). • the fourth parameter is a numeric threshold (e.g. 20 means the firstvalue must be within a range +/-20 on the second value). EqualWithinThreshold(1035,1000,5,20) This will return true as 1035 is within +/- 5% of 1000 (950-1050), even though it is not within +/- 20 (980-1020)
EqualWithinThreshold	Boolean	**EqualWithinThreshold(firstValue,secondValue,lowerPercentageThreshold, lowerNumericThreshold, upperPercentageThreshold, upperNumericThreshold, excludeEquality)** Returns true if the first value is equal to, within a certain tolerance of, the second value. The first value must be within one or the other of these thresholds: • the third and fifth parameters are percentage amounts, lower andupper bound (e.g. -5 and 10 means the first value must be within the range [-5%, +10%] of the second value). • the fourth and sixth parameters are numeric thresholds, lower andupper bound (e.g. 0 and 20 means the first value must be within the range [0, +20] of the second value). • the seventh parameter is a Boolean, indicates whether to exclude the equality from the comparison. Its default value is false (e.g. true means the result will be false if the first and the second value are equal). • The lower percentage bound must be lower or equal than the upperpercentage bound. • The lower numeric bound must be lower or equal than the upper numeric bound. EqualWithinThreshold(1035,1000,1,-10,5,20,true) This will return true as 1035 is within [1%, +5%] of 1000 (1010-1050), even though it is not within [-10, +20] (990-1020)
Exp	Number	**Exp(number)** Returns the exponential value of the provided number.
Factorial	Number	**Factorial(number)** Returns the factorial of the given number.
ForAll	Boolean	**ForAll(group, comparator, value)** A function that checks all numeric items in a referenced group against a numeric value, using a comparator. Disabled items are ignored. The expression only returns true if the comparison is true for all items checked or if the group doesn't exist. The comparators allowed are '>=', '=', '<=', '>', '<', and '!='.

Name	Return Type	Description
ForAll	Boolean	**ForAll(foreignGroup, comparator, value)** Similar to above, but for a group in a foreign schema. Note, the foreign schema must exist in the same return as the schema where this expression function is used.
		ForAll(endDate, firmId, foreignGroup, comparator, value) Similar to above, except that this expression is used when the foreign schema group is in a different return to the current schema where this expression function is used.
GenerateSequentialNumber	Number	**GenerateSequentialNumber(key)** This function generates the next number in a particular sequence (for the given key). When this function is used it will increment the previously generated number by 1, e.g. **Example:** GenerateSequentialNumber("Unique_Reference") will generate 1 then 2 then 3 and so on. Caution should be used in setting this value on a calculated field, as this will increment each time that a form is viewed.
GenerateSequentialNumberLegacy	Number	**GenerateSequentialNumberLegacy(key)** **WARNING**: This function can cause snapshot issues if called many times in quick succession. ONLY use this function if you understand the consequences, in most situations GenerateSequentialNumber should be used instead. This function generates a unique number corresponding to the given key. When this function is used it will increment the previously generated number by 1, e.g. GenerateSequentialNumberLegacy("Unique_Reference") will generate 1 then 2 then 3 and so on. DO NOT use this function in a calculated value or any other place it can be called frequently; including for data transfers. This function should only be used on firm profiles.
GenerateGUID	Number	**GenerateGuid()** Generates a new globally unique identifier (GUID). A GUID is represented as 32 hexadecimal (base 16) digits in the format xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, e.g. 123e4567-e89b-12d3a456-426655440000.
Ln	Number	**Ln(A)** Returns the natural logarithm value of the provided number. If the number is negative, null is returned.
Max	Number	**Max(A,B)** Returns the largest value, either A or B
MaxA	Number	**MaxA(numberArray)** Returns the largest value in an array of numbers.

Name	Return Type	Description
MedianAbsolute Deviation	Number	**MedianAbsoluteDeviation(numberArray)** Returns the median absolute deviation of an array of numbers. To get the median absolute deviation, the array values are sorted in increasing order and the median value is found. The absolute deviation from the median is then calculated for each value. The array of absolute deviations is again sorted, and the median absolute deviation is returned. **Note:** null values in the array are ignored. If the array is empty or has only one value, null is returned.
Min	Number	**Min(A,B)** Returns the smallest value, either A or B.
MinA	Number	**MinA(numberArray)** Returns the smallest value in an array of numbers.
Modulus11	Number	**Modulus11(A)** Returns true if the specified string/number satisfies Modulus 11, false otherwise.
Round	Number	**Round(number, NumberOfDigits)** Rounds a number to a specified number of digits.
Sign	Number	**Sign(A)** Returns a number indicating the sign of the input value. The return value is -1 if the input value is less than zero. 0 if the value is equal to zero or null. and 1 if the value is greater than zero.
SQRT	Number	**SQRT(number)** Returns the square root of the number provided
StDev	Number	**StDev(numberArray)** Returns the standard deviation of an array of numbers.
Sum	Number	**Sum(ListNumberItem)** Adds all the numbers in a list of numbers.
ToNumber	Number	**ToNumber(string)** Converts its string parameter to a number (if possible, otherwise returns null).
UnaryMinus	Number	**UnaryMinus(A)** Returns the negative of numerical parameter A.

Reporting Entity Expression Functions

The table below describes the expression functions to retrieve information about a reporting entity or an entity group that are in APRA Connect.

Note:
In all expression functions where firmId is a parameter, this a number that is normally populated by one of the following expressions:

- GetFirmID in the context of a return
- GetFirmIdForProfile in the context of a profile
- GetFirmsForReport in the context of a standard or aggregate report

Name	Return Type	Description
GetFirmID	Number	**GetFirmID([@Schema_Instance_Guid])** Returns the ID of the reporting entity for the current Schema Instance. **Note**: This expression function can only be used with return form sets. As this value does not change once a form is created, it is best practice to use this as an initial value so that it is not re-calculated every time you view the form. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format. **Example:** GetFirmID([@Schema_Instance_Guid])
GetFirmIdForProfile	Number	**GetFirmIdForProfile([@Schema_Instance_Guid])** Gets the ID for the reporting entity for the current Profile Schema Instance. **Note**: This expression function cannot be used as an initial value on the profile as this will result in a value of 0 being returned. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format. **Example:** GetFirmIdForProfile([@Schema_Instance_Guid])
GetFirmsForReport	Array[Number]	**GetFirmsForReport([@Schema_Instance_Guid])** Gets the IDs of the reporting entities selected for the current Standard or Aggregate Report. This will include regulated entities that are members of any groups selected when the report is generated. **Note:** This expression function can only be used with Standard Reports and Formsets designed to be used for Aggregate Reports. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format.

Name	Return Type	Description
GetFirmsForReport	Array[Number]	**GetFirmsForReport([@Schema_Instance_Guid], membershipDate)** Gets the IDs of the reporting entities selected for the current Standard or Aggregate Report. This will include regulated entities that are members of the groups selected on the given membershipDate. membershipDate is a date type. This expression is identical to above if membershipDate is passed in as Now(). **Note:** This expression function can only be used with Standard and Aggregate Reports. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format.
GetFirmAndGroupCodesForReport	Array[String]	**GetFirmAndGroupCodesForReport([@Schema_Instance_Guid])** Returns a comma-separated list of the reporting entity references and Group descriptions that were selected when generating the current Standard or Aggregate Report. If a Group is selected when generating the report, only the Group description is returned, and not the reference for each Reporting Entity that is a member within the group, unless the Reporting Entity is specifically selected when generating the report. The results are ordered by entity references alphabetically, followed by the group descriptions alphabetically. For multi-lingual projects, the references returned will be in the language selected when the report was generated. **Note:** This expression function can only be used with Standard and Aggregate Reports. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format. In the case of Group Descriptions, **Example:** When a standard report is generated, if the following were selected: **Regulated Entities:** National Bank (ref=REG112), Big Bank (ref=REG111) **Entity Groups:** Foreign (desc=GRP123), Domestic (desc=GRP124) The following expression: GetFirmAndGroupNamesForReport([@Schema_Instance_Guid]) will return "REG111, REG112, GRP123, GRP124"

Name	Return Type	Description
GetFirmAndGroupNamesForReport	Array[String]	**GetFirmAndGroupNamesForReport([@Schema_Instance_Guid])** Returns a comma-separated list of the Reporting Entity and Group names that were selected when generating the current Standard or Aggregate Report. If a Group is selected when generating the report, only the Group name is returned, and not each Reporting Entity that is a member within the group, unless the Reporting Entity is specifically selected when generating the report. For multi-lingual projects, the names returned will be in the language selected when the report was generated. **Note:** This expression function can only be used with Standard and Aggregate Reports. When using for Aggregate Reports, make sure to check if the current formset instance is a report using the IsReport expression function. **[@Schema_Instance_Guid]** is a placeholder parameter that must be passed to the expression function in this **exact** format. **Example:** This may be used to display all of the firms and groups that were selected when generating a standard report. If the following were selected: **Regulated Entities:** National Bank, Big Bank **Entity Groups:** Foreign, Domestic The following expression: GetFirmAndGroupNamesForReport([@Schema_Instance_Guid]) will return "Domestic, Foreign, Big Bank, National Bank"
IsFirmInGroup	Boolean	**IsFirmInGroup(firmId, groupName)** This function returns true if the regulated entity with the given ID (number) is currently a member of the group with the given name. For multi-lingual projects, the group name must be in the **default language** for the project, regardless of the language selected in VSC when the report was generated. False will be returned if: • The reporting entity is not currently a member of this group • The reporting entity or group does not exist • The reporting entity or group has been deleted **Example:** Taking the example of the following entity groups: Domestic Banks group has members National Bank and First Bank. Foreign Banks group has members Worldwide Bank, Multinational Bank. The following expression could be used as a dependency on a form section that only applies to domestic banks: IsFirmInGroup([FirmID], 'Domestic Banks') will result in true for First Bank and false for Worldwide Bank.

Name	Return Type	Description
IsFirmInGroup	Boolean	**IsFirmInGroup(firmId, groupDescription, membershipDate)** This function returns true if the reporting entity with the given ID (number) is a member of a group with a matching description for a specific date. For multi-lingual projects, the group description must be in the **default language** for the project, regardless of the language selected in VSC when the report was generated. False will be returned if: • The reporting entity is not a member of the group on the date specified • The reporting entity or group with this description does not exist • The reporting entity or group with this description has been deleted **Note:** Group descriptions should be unique **Example:** Taking the example of the following entity groups: Domestic Banks group (Description=123456) currently has members National Bank and First Bank. Foreign Banks group (Description=123555) currently has members Worldwide Bank, Multinational Bank. The following expression could be used as a dependency on a form section that only applies to domestic banks: IsFirmInGroup([FirmID], '123456', Now()) will result in true for First Bank and false for Worldwide Bank.
IsFirmInGroup	Boolean	**IsFirmInGroup(firmId, groupId, membershipDate)** This function returns true if the reporting entity with the given Firm ID (number) is a member of a group with the given group ID (number) for a specific date. False will be returned if: • The reporting entity is not a member of the group on the date specified • The reporting entity or group does not exist • The reporting entity or group has been deleted **Example:** Taking the example of the following entity groups: Domestic Banks group (ID=1) currently has members National Bank and First Bank. Foreign Banks group (ID=2) currently has members Worldwide Bank, Multinational Bank. The following expression could be used as a dependency on a form section that only applies to domestic banks: IsFirmInGroup([FirmID], 1, Now()) will result in true for First Bank and false for Worldwide Bank.

Name	Return Type	Description
IsParent	Boolean	**IsParent(firmId, asOfDate, isUltimate)** A function that returns true if the firmID passed to the function is a parent firm as of Date specified in the function. The IsUltimate flag serves as an AND condition which when combined with other parameters returns true only if the Reporting Entity of any other entity.
IsSubsidiary	Boolean	**IsSubsidiary(firmId, asOfDate, isUltimate)** A function that returns true if the firmId passed to the function is a subsidiary as of Date specified in the function. The IsUltimate flag serves as an AND condition which when combined with other parameters returns true only if the Reporting Entity is both a subsidiary as of Date and is not a parent of another entity.
	Boolean	**IsSubsidiary(firmId, asOfDate, parentFirmIds)** A function that returns true if the firmId is a subsidiary of any of the parent firmIds passed to this function on the Date specified.
GetFirmFiscalYearEnd	Date	**GetFirmFiscalYearEnd(firmId, asOfDate)** Returns the fiscal year-end date for a reporting entity with the given ID (number) for the date passed in. **Returns:** The full date as stored in VSAB, which includes the year recorded as the year at the time of when the fiscal year end was set.
GetFirmName	String	**GetFirmName(firmId)** Returns the name of the firm for the Firm ID passed in. The name returned will reflect the name of the reporting entity for the current language when the expression is executed. For example, if the name is displayed on a form, report or profile, this will be the name of the regulated entity in the language selected by the VP or VSC user. **Example:** An entity has the name 'Big Bank' in English and 'La Grande Banque' in French. An item exists in the schema called FirmID that has an initial value of GetFirmID([@Schema_Instance_Guid]). If the user is working in English and a calculated value exists on a form displaying Name with the expression: GetFirmName([FirmID]) will return 'Big Bank'. If the user switches to French and reviews the same form, as the item is a calculated value, 'La Grande Banque' will be displayed.

Name	Return Type	Description
GetFirmReference	String	**GetFirmReference(firmId)** Returns the reference for the entity for the Firm ID passed in. If no reference has been given to the entity, a blank string will be returned. **Example:** An item exists in the return schema called FirmID that has an initial value of GetFirmID([@Schema_Instance_Guid]). The reference may be displayed on a form using the expression GetFirmName([FirmID])
GetFirmStatus	String	**GetFirmStatus(firmId)** Returns the current status (e.g. 'Active') for the Firm ID passed in. Note: In a multi-lingual project, the status will be returned in the current language selected when the expression executes. **Example:** If the system has been configured with the states 'Active', 'Registering', 'Deregistered' and 'Suspended', and the entity 'Big Bank' state has been updated in VSC to 'Suspended'. In this example, the profile has a form with shows the entity status with the expression: GetFirmStatus(GetFirmIDForProfile([@Schema_Instance_Guid])) 'Suspended' will be returned.
GetFirmType	String	**GetFirmType(firmId)** Returns the current type (e.g. 'Company', 'Holding') for the Firm ID passed in. **Note:** In a multi-lingual project, the type will be returned in the current language selected when the expression executes. **Example:** If the system has been configured with the states 'Holding', 'Public' and 'Non-Holding', and the entity 'Big Bank' state has been updated in VSC to 'Non-Holding'. In this example, the profile has a form with shows the entity type with the expression: **GetFirmType**(GetFirmIDForProfile([@Schema_Instance_Guid])) 'Non-Holding' will be returned

Name	Return Type	Description
GetFirmIDByName	Number	**GetFirmIDByName(nameValue, cultureCode)** Returns the Firm ID of the entity identified by the given name, for the language selected by the cultureCode Parameter. **Note:** The culture code must be a configured project language. The culture code is an optional parameter. If this parameter is not used, the function will use the current language selected when the expression executes. **Example:** If a schema has an item 'AssociatedFirmName' containing the name of an associated firm, and 'en-IE' is the code of a project language. **GetFirmIDByName**([AssociatedFirmName], 'en-IE') Returns the ID of the entity with its name stored in the 'AssociatedFirmName' schema item, using the 'en-IE' culture. If there is no such entity, null is returned.
GetFirmIDByReference	Number	**GetFirmIDByReference(referenceValue)** Returns the Firm ID of the entity identified by the given reference. **Note:** Note entity references require configuration to be deemed mandatory and unique. If the expression finds more than one entity with the same reference it will return null. **Example:** If a schema has an item 'AssociatedFirmReference' containing the reference of an associated firm. **GetFirmIDByReference**([AssociatedFirmReference]) Returns the ID of the entity with its reference stored in the 'AssociatedFirmReference' schema item. If there is no such entity or if more than one entity is found, null is returned.

Name	Return Type	Description
BuildEntityUrl	URL	**BuildEntityUrl(entityId)** Returns the relative URL for the entity profile in Vizor Supervision Centre for the given entity. The entityId parameter can be replaced with a number expression or a numeric schema item to calculate the entity id the url is required to be built for. **Note:** This is intended to work on Vizor Supervision Centre only. If used on a form that is display on Vizor Portal, an empty URL is returned This is a server side only expression which means that the expression will only be evaluated when the form is opened i.e. if the entity ID is selected on the same form where the expression is located, the expression will not be updated until the form is saved and re-opened. If the entity ID does not exist, this expression function will return an empty URL i.e. no URL will be generated. This expression function cannot be used on from Standard or Aggregate Reports. **Example:** A schema item exists called EntityRelationship_Name that contains the name an entity in the system. The following expression will retrieve this entity's ID and pass into BuildEntityUrl, giving the URL to link to the entity profile specified in EntityRelationship_Name: BuildEntityUrl(GetFirmIDByName([EntityRelationship_Name])) This could be used in a dynamic hyperlink control to link a form to another entity's profile.

Return Expression Functions

Expression functions to retrieve information related to returns that are in APRA Connect.

Name	Return Type	Description
GetApprovedByUserID	Number	**GetApprovedByUserId([@Schema_Instance_Guid])** Retrieves the User ID of the user that approved the current return. Null is returned if the return has not been approved. **Example:** This could be used in conjunction with GetUsernameByUserId to display the name of a user who approved the return: GetUsernameByUserId(GetApprovedByUserId([@Schema_Instance_Guid]) **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetCategoryIds	Array[String]	**GetCategoryIds([@Schema_Instance_Guid])** Retrieves an array of Category IDs for the current return as a string array. An empty string array will be returned if the return does not have any categories associated with it. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetCategoryNames	Array[String]	**GetCategoryNames(categoryIdsArray, language)** Retrieves the Category Names for the category id's provided in a string array for the given language. An empty string array will be returned if the return does not have any category associated with it. **Example:** The following expression will retrieve all category names for English-Ireland. GetCategoryNames([all CategoryId],'en-IE')
GetCategoryOption	String	**GetCategoryOption([@Schema_Instance_Guid], [@Category_Id])** This function will return the category option associated with the current return as a string, an empty string will be returned if the return does not have any category associated with it. **Note:** **[@Schema_Instance_Guid] and [@Category_Id]** are placeholder parameters which must be passed to the expression function in this exact format.

Name	Return Type	Description
GetCurrentSchemaInstanceGUID	String	**GetCurrentSchemaInstanceGUID([@Schema_Instance_Guid])** Retrieves the GUID that uniquely identifies the current Schema Instance associated with this return. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetMostRecentDate	Date	**GetMostRecentDate(firmId, foreignSchema)** Retrieves the Reporting End Date of the most recently submitted return for the current schema type and regulated entity. **Note:** The parameter for foreignSchema must take the form of [@schema=Schema Name, @group=/SchemaName]. You will get this parameter by selecting 'Foreign Reference' and selecting the root folder of the required schema.
GetReportingEndDate	Date	**GetReportingEndDate([@Schema_Instance_Guid])** Retrieves the Reporting End Date for the current return. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSchemaRules	Arrray[String]	**GetSchemaRules(schemaName)** Returns an array of rule names for the given schema. For dual language systems, the names will be returned in the current language. **Example:** GetSchemaRules('AnnualInsurance') will return an array containing the names of all schema rules for the AnnualInsurance schema e.g. 'Total Capital Reserves does not tally with sum of component elements', 'Implausible Variance in Total Capital Reserve', etc...
GetSubmissionPackApprovalDate	Date	**GetSubmissionPackApprovalDate([@Schema_Instance_Guid])** Retrieves the approval date (if exists) of this return revision. If this revision has not been approved, a null value will be returned. This could be used to display the approval date on a supervisor-only form, or to transfer the approval date to the Corporate Profile when a notification form is approved e.g. date of approval of a new director. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.

Name	Return Type	Description
GetSubmissionPackDueDate	Date	**GetSubmissionPackDueDate([@Schema_Instance_Guid])** Retrieves the Due Date for the current return. If a due date has not been specified, a null value will be returned. For example, the due date may not be set for ad hoc returns created from Vizor Portal e.g. update director details. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackInstanceId	Number	**GetSubmissionPackInstanceId([@Schema_Instance_Guid])** Retrieves a unique ID for the current return i.e. submission pack instance. This may be used to pass a unique identifier to 3rd party systems. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackVersionId	Number	**GetSubmissionPackVersionId([@Schema_Instance_Guid])** Retrieves a unique ID for the current revision of the return i.e. submission pack version. This may be used to pass a unique identifier to 3rd party systems. **Note:** This expression function will not work as an initial value for a schema item within the return. **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackVersionNo	String	**GetSubmissionPackVersionNo([@Schema_Instance_Guid])** Retrieves the revision number (as a string) for the current return i.e. "0.1", "1.0", "2.0". **Note:** This expression function will return null if used as an initial value for a schema item within the return, or for any expression within a profile or report. **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackLastModifiedByUserId	Number	**GetSubmissionPackLastModifiedByUserId([@Schema_Instance_Guid])** Retrieves the ID of the user who last modified a return. **Example:** This could be used in conjunction with GetUsernameByUserId to display the name of a user who last modified a return on a form: GetUsernameByUserId(GetSubmissionPackLastModifiedByUserId([@Schema_Instance_Guid]) **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.

Name	Return Type	Description
GetSubmissionPackName	String	**GetSubmissionPackName([@Schema_Instance_Guid])** Retrieves the name of the return in the current language. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackReference	String	GetSubmissionPackReference([@Schema_Instance_Guid]) Retrieves the reference associated with the return (as a string). Note: [@Schema_Instance_Guid] is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmissionPackSubmittedDate	Date	**GetSubmissionPackSubmittedDate([@Schema_Instance_Guid])** Retrieves the date of submission for the current return. If the return has not been submitted, null will be returned. **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
GetSubmittedByUserId	Number	**GetSubmittedByUserId([@Schema_Instance_Guid])** Retrieves the ID of the user who submitted a return. If the return has not been submitted, null will be returned. **Example:** This could be used in conjunction with GetUsernameByUserId to display the name of a user who submitted a return on another form. GetUsernameByUserId(GetSubmittedByUserId([@Schema_Instance_Guid]) **Note:** **[@Schema_Instance_Guid]** is a placeholder parameter which must be passed to the expression function in this exact format.
IsReport	Boolean	**IsReport([@Schema_Instance_Guid])** This expression is used when a form set can be aggregated. True is returned when this expression is evaluated on an aggregate report, and false when the expression is evaluated on a return.

Text Expression Functions

Name	Return Type	Description
=	Boolean	**[StringA]=[StringB]** Returns true if string A equals string B exactly. This comparison is case sensitive, so it will return false for 'ABC'='abc'.
+	String	**'Date of Birth: ' + ToString([DOB])** Concatenates two strings together & is often used in the Dynamic Text control, for example 'Date of Birth: 12/01/1965'.
FormatNumber	String	**FormatNumber(number, decimalPlaces)** Returns a formatted string rounded to the specified number of decimal places.
Trim	String	**Trim(textToTrim)** A function for removing blank spaces from the start and end of the given text.
Len	Number	**Len(string)** Returns the length of the provided string.
AddZeroPadding	String	**AddZeroPadding(number, digits)** Returns the string containing the number in the first parameter padded to the length defined in the second parameter. The padding will consist of leading zeros. For example, AddZeroPadding(13, 7) will result in a 7 digit value ending in 13. In this example, as only 2 digits are used for the first parameter, the remaining 5 digits will be zero which results in the string "0000013". If the number in the first parameter exceeds the number of digits, then a string containing the number is returned. For example, AddZeroPadding(123456789, 7) will return the string "123456789".
MatchRegEx	Boolean	**MatchRegEx(string, pattern, ignoreCase)** This function returns true if the string used in the function matches the regular expression. Enter true as a parameter to ignore case in looking for a match.
Replace	String	**Replace(input, oldValue, newValue, ignoreCase)** This expression function is used to provide String.Replace(input, oldvalue, newValue, IgnoreCase) functionality. The IgnoreCase flag provides for case sensitivity. If the input arguments are null, the expression will return null. For example, Replace('abcde','cd','xyz', true) will return 'abxyze'.

Name	Return Type	Description
ToString	String String	**ToString(boolean)** Converts a boolean to a string e.g. 'True' or 'False'. **ToString(date),** **ToString(date, dateFormat)** Converts a date to a string, with the option to specify the date format to be returned. If no date format is given, the day, month and year will be provided in your current culture. Examples: For the following examples, [PeriodEndDate] is 31st December 2019 ToString([PeriodEndDate]) As no format is provided, the culture 'en-GB' will return '31/12/2009' and 'en-US' will return '12/31/2019' ToString([PeriodEndDate],'yyyy-MM-dd') '2019-12-31' is returned regardless of the user's culture. ToString([PeriodEndDate],'ddd, dd MMM yyy HH:mm:ss') 'Tue, 31 Dec 2019 00:00:00' is returned for English languages. The following table provides a full list of options:

Key	Description
d	Represents the day of the month as a number from 1 through 31
dd	Represents the day of the month as a number from 01 to 31
ddd	Represents the abbreviated name of the day in the current language. For example, in English this would be Mon, Tues, Weds etc
dddd	Represents the full name of the day in the current language. For example, in English this would be Monday, Tuesday etc
M	The number for the month i.e March = 3
MM	The number for the month with 2 digits i.e. March = 03
MMM	Abbreviated name for the month in the current language. For example, in English this would be Jan, Feb etc
MMMM	The full name of the month in the current language for example in English this would be January, February
y	The number for the year with no leading zero e.g. 19 for 2019
yy	The number the year with leading zero e.g. 019 for 2019
yyyy	Thenumber for the year e.g. 2019

Name	Return Type	Description
		h 12-hour clock e.g. 4 hh 12- hour clock with 2 digits e.g. 04 H 24-hour clock e.g. 16 HH 24-hour clock with 2 digits e.g. 16 m Minutes mm Minutes with 2 digits s Seconds ss Seconds with 2 digits t Abbreviated AM/PM e.g. A/P tt AM or PM
ToStringArray	Array[String]	**ToStringArray(arrayOfEnumerationOptions)** Takes an array of enumeration options and returns an array of strings where the strings represent the label of the enumeration options. For multi-lingual projects, the labels will appear in the language that the user has selected. Example: A List exists containing a schema item called [PremisesCountry] exists. This is an enumeration datatype called 'Countries' with the following options: Key / Label (English) / Label (French) IE / Ireland / Irlande GB / United Kingdom / Royaume Uni FR / France / France IT / Italy / Italie If a table is completed with all 4 countries, the following expression ToStringArray([all PremisesCountry]) will return the following array when working in French 'Irlande', 'Royaume Uni' 'France', 'Italie' and the following array when working in English 'Ireland', 'United Kingdom', 'France', and 'Italy'

Name	Return Type	Description
Substring	String	**Substring(input, startIndex, length)** This expression function returns part of the string (i.e. the substring) specified in its first parameter, e.g. substring('hello world', 6, 5) will return 'world'. The second parameter indicates the first character to return which can be any character from the submitted string. Note: The function uses 0-based indexes, i.e. counting always begins at 0. The character at index 0 in the above example is 'h', index 1 is 'e', index 2 is 'l' etc. The third parameter specifies the number of characters to return, i.e. the length of the string to be returned. If the start or length arguments are out of range the expression will return null.
GetFilenameWithoutExtension	String	**GetFilenameWithoutExtension(string)** This expression accepts a string parameter and returns the file name excluding the file extension. **Example:** GetFilenameWithoutExtension('samplefile.jpg') This will return 'samplefile'.
GetFileExtension	String	**GetFileExtension(string)** This expression accepts a string parameter and returns the file extension. **Example:** GetFileExtension('samplefile.jpg') This will return '.jpg'.
EncodeURIComponent	String	**EncodeURIComponent(string)** The expression accepts a string parameter and URI encodes the characters. **Example:** EncodeURIComponent('_'__<_>_{_}_?_/_#_&_~_,_:_@_=_$_+_') This will return '_%27_%5c_%3c_%3e_%7b_%7d_%3f_%2f_%23_%26_%7e_%2c_%3a_%40_%3d_%24_%2b_'.

Utility Expression Functions

Name	Return Type	Description
FromArray	Any	**FromArray(index, array)** Returns the value at a given index from an array of values. A value may be extracted from the array containing values of the same base data type: • Bool • Date • Enumeration • Number • String File data type is not currently supported. Null will be returned if the index is outside the bounds of the array. Note, that the array is zero-based i.e. 0 will return the first string in the array, 1, the second item in the array.
IsDisabled	Boolean	**IsDisabled(GroupOrList)** Determines if an instance of a Schema Group or List is disabled. True is returned if the Schema Group or List is enabled, otherwise false.
IsNull	Boolean	**IsNull(item)** Returns true if the given parameter is null, otherwise false. The following data types are supported as input parameters: • Bool • Date • Number • String File and enumeration types are not currently supported.
Intersect	Array[Any]	**Intersect(arrayA, arrayB)** Gets the intersection of two arrays of values. The two parameters are the arrays. The following data types are supported as input parameters: • Bool • Date • Enumeration • Number • String File data type is not currently supported.

Name	Return Type	Description
MakeStringArray	Array[String]	**MakeStringArray(string, splitCharacter)** Splits the given string into an array of strings using the given character to perform the split. For example: MakeStringArray('this is a sentence', ' ') will return an array of: • this • is • a • sentence Note, that if the splitCharacter is null or empty, each character will be returned in a string of arrays. For example: MakeStringArray('this is a sentence', '') will return an array of:'t','h','i','s', ' ','i','s', ' ','a',' ','s','e','n','t','e','n','c','e'.
MakeStringFromArray	String	**MakeStringFromArray(array, splitCharacter)** Converts the array of strings in the first parameter into a single string using the second parameter as a separator. For example, the array of characters 't','h','i','s', ' ','i','s', ' ','a',' ','s','e','n','t','e','n','c','e' will return a single string 'this is a sentence'. If the string array parameter is empty, an empty string will be returned.

MakeNumberArray	Array[Number]	**MakeNumberArray(arrayOfNumbers, schemaItems)**
		This will return an array of numbers, either by referencing multiple Schema Items or raw numbers in its parameters. These are a comma-separated list where each entry is either a schema item with a number data type or a numeric constant.
		MakeNumberArray([A1],[A2],...,[Ax])

MakeBoolArray	Array[Bool]	**MakeBoolArray(arrayOfBooleans, schemaItems)**
		This will return an array of boolean values, either by referencing multiple Schema Items or raw boolean values in its parameters. These are a comma separated list where each entry is a Boolean value.
		MakeBoolArray([Bool1],[Bool2],...,[BoolX])
		MakeBoolArray(true,false,...,true)

MakeDateArray	Array[Date]	**MakeDateArray(arrayOfDates, schemaItems)**
		This will return an array of dates, either by referencing multiple Schema Items or raw dates in its parameters. These are a comma separated list where each entry is a date value.
		MakeDateArray([date1],[date2],...,[dateX])
		MakeDateArray(Date(2020,02,02),...,Date(2020,02,10))

Name	Return Type	Description

MakeEnumerationArray	Array[Option]	**MakeEnumerationArray(arrayOfOptions,schemaItems)**
		This will return an array of enumeration options, either by referencing multiple Schema Items or raw enumeration options in its parameters. These are a comma separated list where each entry is an enumeration option.
		MakeEnumerationArray([Enum1],[Enum2],...,[EnumX])
		MakeEnumerationArray(Option([@data_types],'Country','IE') ,...,Option([@data_types],'Country','FR'))

Make2DArray	Array[Any] [Any]	**Make2DArray(array1,array2,...,arrayX)** This returns a two-dimensional array of mixed type. The parameters are a comma-separated list of one-dimensional arrays of any type. For example, to create a 2D array as below: [12, 'John'], [14, 'Jane'], [18, 'June'] The expression function would be used in the format: Make2DArray(MakeNumberArray(12,14,18),MakeStringArray('John ,Jane,June',','))) The primary use of this function is to initialize lists: The 1st item of each array parameter would make the first row of the list, the 2nd item of each array would make the 2nd row etc.
Nz	Any	**Nz(A, B)** Returns A if A is not null, otherwise returns B. A and B must be the same type. The following data types are supported as input parameters: • Bool • Date • Number • String File and enumeration types are not currently supported.
ReverseArray	Array[Any]	**ReverseArray(array)** Reverses the order of an array. The following data types are supported as input parameters: • Bool • Date • Enumeration • Number • String File data type is not currently supported.

Name	Return Type	Description
Union	Array[Any]	**Union(arrayA, arrayB, excludeDuplicates)** Gets the union of two arrays of values. The first two parameters are the arrays while the third parameter determines whether or not to remove duplicate values from the result or not. The following data types are supported as input parameters: • Bool • Date • Enumeration • Number • String File data type is not currently supported.
Join	Array[String]	**Join(segments)** Returns a string array in which each entry is a concatenation of a particular index of the passed in string arrays. It will accept a list of objects. Each object must be either of type string or string array For example, the expression Join(MakeStringArray('a,b,c', ','),'_','21') will return an array of: a_21, b_21 and c_21
ArrayMerge	Array[String]	**ArrayMerge(valuesA, valuesB)** Returns a merged string array in which each entry is the corresponding index value of either A or B from the passed in parameters. If valueA is NULL or Empty (including whitespace) then valueB will be used. Three kind of scenarios will work for the second parameter: 1: it can be a string in which case the same string is always used as the substitution. 2: it can be a string array of length one in which case the string at index zero is always used as the substitution. 3: It can be a string array of equal length to the first parameter, in which case the value in the corresponding index of the first parameter's 'NULL or Empty' will be used as the substitution. **Examples:** ArrayMerge(MakeStringArray('valueA1,,ValueA3', ','), MakeStringArray('valueB1,valueB2,ValueB3', ',')) will return an array of: valueA1, valueB2, valueA3 ArrayMerge(MakeStringArray('valueA1,,,ValueA3', ','), 'ValueB') will return an array of: valueA1, valueB, valueB, valueA3 ArrayMerge(MakeStringArray('valueA1,,ValueA3', ','), ['ValueB']) will return an array of: valueA1, valueB, valueA3

Custom Expression

Name	Return Type	Description
ValidateList	Array[String]	**ValidateList(listParameter)** This function returns a list of errors found when checking holes in a list. It's used in conjunction with a holes file stored under the schema custom attribute name **Holes Constraint File**. This will validate so that only the pre-set list of explicitly defined combinations can be entered. An example use case could be a list that contains 3 items and is intended to record amounts for all European countries in their native currency: Country Currency Amount The associated holes file will store all possible combinations that can be entered like below Country: Ireland Currency: Euro Amount: 1 Country: France Currency: Euro Amount: 1 etc... The above 2 sets of combinations mean that an Amount is an allowed value to be entered under Ireland-Euro or France-Euro. If the following combination is attempted then it will not be in the holes file and therefore the validation will fail: Country: France Currency: Dollar Amount: 155798.98 Each item that has a predefined set of values (like the country and currency enumerations) will include these actual values in the holes file. Each item that doesn't have a predefined set of values (like amount) will have a 1 or a 0 to indicate if a value can or cannot be entered for that specific row.

Name	Return Type	Description
ValidateListFailureMessages	Array[String]	**ValidateListFailureMessages(listParameter)** This function returns a list of error messages found when checking holes in a list. It should be used in a rule failure message expression where the rule expression utilises the ValidateList expression function. This function returns a list of errors found when checking holes in a list. It's used in conjunction with a holes file stored under the schema custom attribute name **Holes Constraint File**. This will validate so that only the pre-set list of explicitly defined combinations can be entered. An example use case could be a list that contains 3 items and is intended to record amounts for all European countries in their native currency: Country Currency Amount The associated holes file will store all possible combinations that can be entered like below Country: Ireland Currency: Euro Amount: 1 Country: France Currency: Euro Amount: 1 etc... The above 2 sets of combinations mean that an Amount is an allowed value to be entered under Ireland-Euro or France-Euro If the following combination is attempted then it will not be in the holes file and therefore the validation will fail: Country: France Currency: Dollar Amount: 155798.98 Each item that has a predefined set of values (like the country and currency enumerations) will include these actual values in the holes file. Each item that doesn't have a predefined set of values (like amount) will have a 1 or a 0 to indicate if a value can or cannot be entered for that specific row.