Available expression Properties:

|  |  |  |
| --- | --- | --- |
| StartRow | int | the row # of the first data point on the current sheet. All indexing starts with 1. |
| StartColumn | int | the column # of the first data point on the current sheet |
| EndRow | int | the row # of the last data point on the current sheet |
| EndColumn | int | the column # of the last data point on the current sheet |
| SheetName | string | the name of the current sheet |
| WorkbookName | string | the name of the current workbook |
| WorkbookFullPath | string | The full path to the current workbook |
| Date | string | The current date in yyyy-mm-dd format |
| Datetime | string | The current date and time in yyyy-MM-dd-HHmmss format |
| TemplateName | string | The name of the excel template used for creating the current excel workbook, assumes the same value as WorkbookName when importing |
| DatasetCode | string | Sdmx Id of Dataflow involved for the current coordinate mapping |
|  |  |  |

properties available on Cell objects, which can be created with Cell() functions defined below:

If these properties used without a Cell object, a reference the **current Cell in the current worksheet** is assumed

|  |  |  |
| --- | --- | --- |
| Row | int | the (1-based) row index |
| Column | int | the (1-based) column index |
| CurrentCell | Cell | Reference to the currently active cell, as iteration is processed through the massive of cells from Top-left to Bottom-Right |
| CellAbove | Cell | the cell to the top |
| CellBelow | Cell | the cell below |
| RightCell | Cell | the cell to the right |
| LeftCell | Cell | The cell to the left |
| Value | string | the content of the cell, all cell values are assumed to be strings |
| FormattedValue | string | The content of the cell in html format, if the cell contains formatted text |
| Comments | string | the comment on the cell |
| IsError | bool | True if the current cell is considered as an error by Excel such as #DIV/0! |
|  |  |  |

## Functions:

Parameters are mandatory exept parameters in brackets “[“, “]” that are optional and can be omitted.

|  |  |  |
| --- | --- | --- |
| Cell( rowIndex, columnIndex, [sheetName] ) | Cell | These are functions to create a cell reference anywhere in the current workbook. cellreference parameter is cell address in excel format, can be something like “A1” “M12”, “Sheet 1!Q15”, etc. |
| Cell( “cellReference”) | Cell |
| FindCells( "range", "search text" [, isregex = false ] [, notmatching = false] ) | Cells | this function evaluates to a cell collection.  All the cells in the standard excel range 'range' (ex. 'A1:A100') definition is considered from left to right, top to bottom  If notMatching parameter is set to true, all the cells that DOES NOT MATCH are returned, instead of matching ones  All the cell search functions IGNORE WHITE SPACE at the beginning and at the end of the value, both in the cell values and also in the search text. Furhermore string value “” and null are considered equal. |
| FindFirstCell( "range", "search text" [, isregex = false ] [, notmatching = false] ) | Cell | Same as FindCells()but returns the first cell found, instead. |
| FindLastCell( "range", "search text" [, isregex = false ] [, notmatching = false] ) | Cell | Same as FindCells()but returns the last cell found, instead. |
| FirstNonEmptyCell( "range" ) | Cell | returns the first nonempty and non-null cell in the range |
| LastNonEMptyCell( "range" ) | Cell | Same as FirstNonEmptyCell(), but returns the last one, instead. |
| FindRows ( "range", "search text" [, isregex = false ] [, notmatching = false] ) | Int[] | Works same as FindCells() function but returns a collection of row indexes of matching cells, instead |
| FindColumns( "range", "search text" [, isregex = false ] [, notmatching = false] ) | Int[] | Works same as FindCells() function but returns a collection of column indexes of matching cells, instead |
| FirstNonNull( item1, item2 [, ... ] ) | object | Returns the first non-null + non-empty item in the parameter list |
| Trim(“input” [, trimInside = false ] ) | string | Removes the beginning and trailing whitespace characters from its input. If the trimInside = true, it will replace whitespace characters if more than 1 space in a row with 1 space char. For example “ 12 345 “ would become “12 345” |
| LTrim(“input”) | string | Same as Trim() but only the starting whitespace characters are removed |
| RTrim(“input”) | string | Same as Trim() but only the trailing whitespace characters are removed |
| TakeLeft( "text", count ): | string | Takes the first “count” characters from the given “text” |
| TakeRight ( "text", count ): | string | Takes the last “count” characters from the given “text” |
| TakeCut ( "text", startIndex [, count ] ): | string | Takes a cut staring from startIndex(1-based) from the given index. If a count parameter is given takes a maximum of count characters, depending on the size of the text |
| TakeCutUntil( “text”, startIndex [, until =”stop”] | string | Take a substring of the “text” starting from startIndex until it finds the given “stop” characters,excluding stop chars. If until parameter is not used or not found in the “text” a substring of the “text” is taken until to the end of the string |
| IsMatch("pattern" = null, "value" = null [ , isRegex = true ] ) | bool | Does the given “value” match the given text “pattern”. If isregex parameter is false, normal text matching is done, but leading and trailing whitespace is ignored |
| TextContains(“pattern”, “input”) | bool | Does the “input” contain “pattern”? character casing is ignored |
| IsNumeric( "value") | bool | Is the given value numeric? |
| ConvertToNumber(“input” [, “decimalSeparator”]) | int | This function attempts to convert the given textual value to a number by removing all non-numeric characters within the input. If the input is still not convertible to a number a NaN (not-a-number) value is returned. |
| IsEmpty( “value”) | bool | Is the given value considered empty (white space or null)? |
| Replace( “input”, “search”, “replacement”) | string | Returns a new string, replacing the “search” with’n “input” with “replacement” |
| TextContains(“pattern|,”input” ) | bool | Returns true if the “”input” contains “pattern”, false otherwise |
| AreEqual(“str1”, “str2”) | bool | Are the two strings equal, ignoring character casing? |
| If( condition, trueResult, falseResult) | object | Similar to Excel If() function, returns trueResult parameter if the condition is true, falseResult otherwise |
| Case( condition1, result1, condition2, result2, …) | object | Similar to If() function, however, there are multiple conditions that can be tested. The conditions are tested in the given order and the result of first matching condition is returned |

## Example EDDs:

<?xml version='1.0' encoding='utf-8'?>

<Excel-Data-Description>

<Coord-Mapping Name='264D\_264\_SALDI Mapping'

Dataflow='264D\_264\_SALDI'

Agency='IT1'

Version='1.0'

ExcludeNulls='false'

IsActive='true'>

<Worksheets>

<Worksheet Name='Test-Worksheet' />

</Worksheets>

<Top-Left Coord='M6' />

<Bottom-Right Coord='M105' />

<Bottom-Right Coord="Cell(LastNonEmptyCell(&quot;M5:M500000&quot;).Row, 13)" IsExpression="true" />

<Dimension Code='TIME\_PERIOD' Value='Cell(Row, Column-10).Value' />

<Dimension Code='FREQ' Value='Cell(Row, Column-9).Value' />

<Dimension Code='REF\_AREA' Value='Cell(Row, Column-8).Value' />

<Dimension Code='IND\_TYPE' Value='Cell(Row, Column-7).Value' />

<Dimension Code='ADJUSTMENT' Value='Cell(Row, Column-6).Value' />

<Dimension Code='MEASURE' Value='Cell(Row, Column-5).Value' />

<Dimension Code='ETA' Value='Cell(Row, Column-4).Value' />

<Dimension Code='SESSO' Value='Cell(Row, Column-3).Value' />

<Dimension Code='GRADO\_ISTRUZ' Value='Cell(Row, Column-2).Value' />

<Dimension Code='CATEG\_PROF' Value='Cell(Row, Column-1).Value' />

<ObsAttribute Code='OBS\_STATUS' Value='Cell(Row, Column+1).Value' />

<ObsAttribute Code='DECIMALS' Value='Cell(Row, Column+2).Value' />

<ObsAttribute Code='NOTA\_EN' Value='Cell(Row, Column+3).Value' />

</Coord-Mapping>

</Excel-Data-Description>

|  |
| --- |
| <?xml version='1.0' encoding='utf-8'?>  <Excel-Data-Description>  <Coord-Mapping Name="My\_EDD" Dataflow="DB\_CODE" Agency="OECD" Version="1.0" ExcludeNulls="false" IsActive="true"> |
| <Worksheets> |
| <Worksheet Name="Results"/> |
| </Worksheets> |
| <Top-Left Coord="D12"/> |
| <Bottom-Right Coord="O24"/> |
| <Exclude> |
| <Row Index='FindRows("B:B", "", false, false)' IsExpression="true"/> |
| <Row Index='FindRows("B:B", "CONTROL", false, false)' IsExpression="true"/> |
| <Column Index='FindColumns("D10:Z10", "", false, false)' IsExpression="true"/> |
| <Cell Condition='IsMatch(Value, "n/a", false)' IsExpression="true"/> |
| </Exclude> |
| <ObsAttribute Coord="RightCell"/> |
| <Dimension Code="COUNTRY" Coord='Cell("B5")' IsExpression="false"/> |
| <Dimension Code="SECTOR" Coord="Cell(Row, 1)" Coord1="Cell(Row-1, 1)" Coord2="Cell(Row-2, 1)" Coord3="Cell(Row-3, 1)" Coord4="Cell(Row-4, 1)"/> |
| <Dimension Code="TYPE" Coord="Cell(Row, 2)"/> |
| <Dimension Code="TIME" Coord="Cell(8, Column)" Coord1="Cell(8, Column-2)" Coord2="Cell(8, Column-4)"/> |
| <Dimension Code="SOURCE" Coord="Cell(10, Column)"/> |
| <Dimension Code="MEASURE" Value="NATCUR" IsExpression="false"/> |
| </Coord-Mapping>  </Excel-Data-Description> |