Description domain sepecific languages for DataTransfer

# Table of contents

[Table of contents 2](#_Toc158101067)

[1 Introduction 3](#_Toc158101068)

[2 Variables 4](#_Toc158101069)

[2.1 Literals 4](#_Toc158101070)

[2.2 Relations 4](#_Toc158101071)

[2.3 Arithmetic operators 4](#_Toc158101072)

[2.4 Bitwise operators 4](#_Toc158101073)

[2.5 String operators 4](#_Toc158101074)

[2.6 Logic operators 4](#_Toc158101075)

[2.7 Functions 5](#_Toc158101076)

[2.7.1 Mathematics 5](#_Toc158101077)

[2.7.2 String 5](#_Toc158101078)

[2.7.3 Date 6](#_Toc158101079)

[2.7.4 Conversions 7](#_Toc158101080)

[2.7.5 Logic 7](#_Toc158101081)

[3 Custom SQL 9](#_Toc158101082)

# Introduction

The Datatransfer uses so called DSLs (domain specific languages) to query various data sources and to declare variables. The default DSL syntax is described here.

# Variables

**The Datatransfer allows the usage of variables in TransferTableJobs with the XML element <variable>.**

**The type and name of the variable is assigned by xml attributes.**

**Setting the value can be done in 3 ways:**

* **Constanct – value attribute**
* **SELECT expression**
  + **Attribute selectContext defines where to execute the sql query – target or source**
  + **Attribute selectStmt sets the sql statement**
  + **Variables can be used with ${{VarName}} – Conversions and brace symbols are not needed and applied automatically depending on data type** 
    - **String var value hello becomes ‘hello‘**
* **Expression**
  + **Uses the dsl itsself to define the value of the variable**

**The expression language has the following specification**

## ****Literals****

Strings are separated with single quotes ‘. Escaping is allowed like \` \n \r \t \\ (like C#)

Numbers are written without any braces/just plain.

Decimals are separated with . .

Boolean literals are plain “true” and “false”.

Null literal is plain „null“.

## ****Relations****

The following relationen are allowed:

<, <=, >, >=, =, ==, !=, <>

## Arithmetic operators

+, -, \*, /, %

## Bitwise operators

&, |

## String operators

+

## Logic operators

&&, ||, and, or

## Functions

The Expression language ist a functional language. Functions are called with name and parameters. Example

Funtcion\_name( param1, param2, …, param n)

The language is case insensitive. In effect, it´s not important if function names are written upper- or lowercase.

The following functions are valid.

### Mathematics

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Description** |
| Sin | Number X  Returns Number | Calculates sine of x |
| Cos | Number X  Returns Number | Calculates cosine of x |
| Tan | Number X  Returns Number | Calculates tangent of x |
| Abs | Number X  Returns Number | Calculates absolute of x |
| Pi | N/A  Returns Number | PI |
| Ceiling | Number X  Returns Number | Rounds the number x up to the next integer |
| Floor | Number X  Returns Number | Rounds the number x down to the next integer |
| Round / rnd | Number X  [integer precision]  Returns Number | Rounds the number x to the next nearest integer.  If precision is set, the rounding is calculated on the n´th decimal position |
| Max | Number X  Number Y  Returns Number | Returns the maximum of [X] and [Y] |
| Min | Number X  Number Y  Returns Number | Returns the minimum of [X] and [Y] |

### String

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Description** |
| ToUpper | String Input  Returns String | Converts all lowercase letters of [input] to uppercase |
| ToLower | String Input  Returns String | Converts all uppercase letters of [input] to lowercase |
| IndexOf | String Input  String SearchString  Returns Number | Searches in string [input] for occurence of [SearchString] and returns the index position.  If the string isn´t found the function returns -1 |
| Replace | String Input  String SearchString  String ReplaceString  Returns String | Search and replaces all occurences of [SearchString] in [input] with the replacement string [ReplaceString]. |
| Substring | String Input  Number startIndex  [Number Length]  Returns String | Returns a substring of [input]. This is determined from the the start position [startIndex] to the end or if given [Length] in the given [Length] from [startindex]. |
| strContains / contains | String Input  String SearchString  Returns Bool | Checks if [input] occurs in string [SearchString].  If so, returns true, else false. |
| strLeft / left | String Input  String SearchString  Returns String | Searchs from the left for the first occurrence of [SearchString] in [input] and returns the substring before the found occurence. If [SearchString] is not found the return is an empty string. |
| strRight / right | String Input  String SearchString  Returns String | Searchs from the left for the first occurrence of [SearchString] in [input] and returns the substring after the found occurence. If [SearchString] is not found the return is an empty string. |
| strMid / Mid | String Input  String SearchStringStart  String SearchStringEnd  Returns String | Searchs from the left for the first occurrence of [SearchStringStart] in [Input]. Then it searches from there for the first occurrence of [SearchStringEnde]. The substring between both findings is returned.  If [SearchStringStart] or [SearchStringEnd] is not found the return is an empty string. |
| startsWith | String Input  String SearchString  Returns Bool | Checks if [input] starts with [SearchString]. If so, returns true, else false. |
| endsWith | String Input  String SearchString  Returns Bool | Checks if [input] ends with [SearchString]. If so, returns true, else false. |

### Date

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Description** |
| Date | Number Ticks  OR  Number year  Number month  Number day  [Number hour  Number minute  Number second]  Returns Date | Creates a date from the count of [Ticks] (see .NET documentation for DateTime)  OR  Creates a date composed of the date part parameters [year] [month] [day].  [hour], [minute] and [second] are optional and can be used to set the time as well. |
| AdjustSeconds | Date Input  Number count  Returns Date | Adds [count] seconds to the date returns the calculated date. Negative [count] is subtracted. |
| Adjustminutes | Date Input  Number count  Returns Date | Adds [count] minutes to the date returns the calculated date.  Negative [count] is subtracted. |
| AdjustHours | Date Input  Number count  Returns Date | Adds [count] hours to the date returns the calculated date.  Negative [count] is subtracted. |
| AdjustDays | Date Input  Number count  Returns Date | Adds [count] days to the date returns the calculated date.  Negative [count] is subtracted. |
| AdjustMonths | Date Input  Number count  Returns Date | Adds [count] months to the date returns the calculated date.  Negative [count] is subtracted. |
| AdjustYears | Date Input  Number count  Returns Date | Adds [count] years to the date returns the calculated date.  Negative [count] is subtracted. |
| Second | Date Input  Returns Number | Returns the second date time part of [input] |
| Minute | Date Input  Returns Number | Returns the minute date time part of [input] |
| Hour | Date Input  Returns Number | Returns the hour date time part of [input] |
| Day | Date Input  Returns Number | Returns the day date time part of [input] |
| Month | Date Input  Returns Number | Returns the month date time part of [input] |
| Year | Date Input  Returns Number | Returns the year date time part of [input] |

### Conversions

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Description** |
| cstr | Type neutral [input]  Returns String | Converts [input] to string and returns it. |
| Cbool | Type neutral [input]  Returns Bool | Converts [input] to boolean and returns it. If not possible an error is returned. |
| Cint | Type neutral [input]  Returns Ganznumber | Converts [input] to integer and returns it. If not possible an error is returned. |
| Cdbl | Type neutral [input]  Returns Number | Converts [input] to double/number and returns it. If not possible an error is returned. |
| Cdate | Type neutral [input]  Returns Date | Converts [input] to date and returns it. If not possible an error is returned. |
| cChar | Type neutral [input]  Returns Zeichen | Converts [input] to character and returns it. If not possible an error is returned. |

### Logic

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Description** |
| If / iif,/ case / casewhen | Bool Condition1  Type neutral Result1  [Bool Condition 2-n  Type neutral Result 2-n]  Type neutral ElseResult  Returns Type neutral | Checks if [condition1] is true and if so, returns [Result1].  Afterwards check the following conditions in the order of occurrence and return the result for the first true case/if. If no condition is true, return [ElseResult]. |
| Nvl | Type neutral [input]  Type neutral [ElseResult]  Returns Type neutral | Checks if [input] is null. If so, returns [ElseResult], otherwise return [input] |
| Not | Bool Input  Returns bool | Changes the boolean value of [input] from true to false or reverse and returns it. |

# Custom SQL

Custom SQL allows SQL syntax on non-SQL data sources which do not support SQL.

One characteristic is that you have to name all tables and calculated columns with a name.

Names of columns or “tables” are named with “AS Name”.

i.e SELECT 1 as Column from Table as T

Simple joins with “=” are accepted

i.e. Select T1.Key from Tab1 as T1 inner join Tab2 T2 on T1.Key = T2.Key

For multiple tables every column has to be specified full qualified. (this means table.column)

Variables can be inserted with the data binding expression ${{Varname}}

i.e. SELECT 3 + ${{NumberVar}} as Calc from Tab AS T1