DataStage8.5函数大全

[DataStage8.5函数大全 1](#_Toc302501093)

[一、 类型转换函数 3](#_Toc302501094)

[1. Char 4](#_Toc302501095)

[2. DateToString 4](#_Toc302501096)

[3. DateToDecimal 4](#_Toc302501097)

[4. DecimalToDate 5](#_Toc302501098)

[5. DecimalToDecimal 6](#_Toc302501099)

[6. DecimalToDFloat 6](#_Toc302501100)

[7. DecimalToString 7](#_Toc302501101)

[8. DecimalToTime 7](#_Toc302501102)

[9. DecimalToTimestamp 8](#_Toc302501103)

[10. DFloatToDecimal 9](#_Toc302501104)

[11. DfloatToStringNoExp 10](#_Toc302501105)

[12. IsValidDate 10](#_Toc302501106)

[13. IsValidTime 11](#_Toc302501107)

[14. IsValidTimestamp 11](#_Toc302501108)

[15. RawNumAt 11](#_Toc302501109)

[16. RawToString 12](#_Toc302501110)

[17. Seq 12](#_Toc302501111)

[18. SeqAt 12](#_Toc302501112)

[19. StringToDate 12](#_Toc302501113)

[20. StringToDecimal 13](#_Toc302501114)

[21. StringToRaw 13](#_Toc302501115)

[22. StringToTime 13](#_Toc302501116)

[23. StringToTimestamp 14](#_Toc302501117)

[24. StringToUstring 14](#_Toc302501118)

[25. TimestampToDate 14](#_Toc302501119)

[26. TimestampToDecimal 14](#_Toc302501120)

[27. TimestampToString 15](#_Toc302501121)

[28. TimestampToTime 16](#_Toc302501122)

[29. TimeToString 16](#_Toc302501123)

[30. TimeToDecimal 16](#_Toc302501124)

[31. UstringToString 17](#_Toc302501125)

[二、 字符串函数 17](#_Toc302501126)

[32. AlNum 17](#_Toc302501127)

[33. Alpha 17](#_Toc302501128)

[34. CompactWhiteSpace 18](#_Toc302501129)

[35. Compare 18](#_Toc302501130)

[36. CompareNoCase 18](#_Toc302501131)

[37. CompareNum 18](#_Toc302501132)

[38. CompareNumNoCase 19](#_Toc302501133)

[39. Convert 19](#_Toc302501134)

[40. Count 19](#_Toc302501135)

[41. Dcount 19](#_Toc302501136)

[42. DownCase 20](#_Toc302501137)

[43. DQuote 20](#_Toc302501138)

[44. Field 20](#_Toc302501139)

[45. Index 20](#_Toc302501140)

[46. Left 21](#_Toc302501141)

[47. Len 21](#_Toc302501142)

[48. Num 21](#_Toc302501143)

[49. PadString 21](#_Toc302501144)

[50. Right 21](#_Toc302501145)

[51. Soundex 22](#_Toc302501146)

[52. Space 22](#_Toc302501147)

[53. SQuote 22](#_Toc302501148)

[54. Str 22](#_Toc302501149)

[55. StripWhiteSpace 23](#_Toc302501150)

[56. Trim 23](#_Toc302501151)

[57. TrimB 24](#_Toc302501152)

[58. TrimF 24](#_Toc302501153)

[59. TrimLeadingTrailing 24](#_Toc302501154)

[60. UpCase 24](#_Toc302501155)

[三、 数字函数 25](#_Toc302501156)

[61. AsDouble 25](#_Toc302501157)

[62. AsFloat 25](#_Toc302501158)

[63. AsInteger 25](#_Toc302501159)

[64. MantissaFromDecimal 26](#_Toc302501160)

[65. MantissaFromDFloat 26](#_Toc302501161)

[四、 日期和时间函数 26](#_Toc302501162)

[66. CurrentDate 26](#_Toc302501163)

[67. CurrentTime 27](#_Toc302501164)

[68. CurrentTimeMS 27](#_Toc302501165)

[69. CurrentTimestamp 27](#_Toc302501166)

[70. CurrentTimestampMS 27](#_Toc302501167)

[71. DateFromDaysSince 28](#_Toc302501168)

[72. DateFromComponents 28](#_Toc302501169)

[73. DateFromJulianDay 28](#_Toc302501170)

[74. DateOffsetByComponents 28](#_Toc302501171)

[75. DaysSinceFromDate 29](#_Toc302501172)

[76. DaysInMonth 29](#_Toc302501173)

[77. DaysInYear 29](#_Toc302501174)

[78. DateOffsetByDays 30](#_Toc302501175)

[79. HoursFromTime 30](#_Toc302501176)

[80. JulianDayFromDate 30](#_Toc302501177)

[81. MicroSecondsFromTime 31](#_Toc302501178)

[82. MidnightSecondsFromTime 31](#_Toc302501179)

[83. MinutesFromTime 31](#_Toc302501180)

[84. MonthDayFromDate 31](#_Toc302501181)

[85. MonthFromDate 31](#_Toc302501182)

[86. NextWeekdayFromDate 32](#_Toc302501183)

[87. NthWeekdayFromDate 32](#_Toc302501184)

[88. PreviousWeekdayFromDate 32](#_Toc302501185)

[89. SecondsFromTime 33](#_Toc302501186)

[90. SecondsSinceFromTimestamp 33](#_Toc302501187)

[91. TimeDate 33](#_Toc302501188)

[92. TimeFromComponents 33](#_Toc302501189)

[93. TimeFromMidnightSeconds 34](#_Toc302501190)

[94. TimeOffsetByComponents 34](#_Toc302501191)

[95. TimeOffsetBySeconds 34](#_Toc302501192)

[96. TimestampFromDateTime 34](#_Toc302501193)

[97. TimestampFromSecondsSince 35](#_Toc302501194)

[98. TimestampFromTimet 35](#_Toc302501195)

[99. TimestampOffsetByComponents 35](#_Toc302501196)

[100. TimestampOffsetBySeconds 35](#_Toc302501197)

[101. TimetFromTimestamp 36](#_Toc302501198)

[102. WeekdayFromDate 36](#_Toc302501199)

[103. YeardayFromDate 36](#_Toc302501200)

[104. YearFromDate 37](#_Toc302501201)

[105. YearweekFromDate 37](#_Toc302501202)

[五、 Null 处理函数 37](#_Toc302501203)

[106. IsNotNull 37](#_Toc302501204)

[107. IsNull 38](#_Toc302501205)

[108. NullToEmpty 38](#_Toc302501206)

[109. NullToZero 38](#_Toc302501207)

[110. NullToValue 38](#_Toc302501208)

[111. SetNull 39](#_Toc302501209)

## 类型转换函数

类型转换函数用于更改参数的类型。

以下函数位于表达式编辑器的“类型转换”类别中。方括号表示参数是可选的。缺省日期格式为 %yyyy-%mm-%dd。

以下示例按照 Transformer 阶段的“派生”字段中所示来显示这些函数。

### Char

根据其数字代码值生成一个 ASCII 字符。您可以指定 allow8bits 参数来转换 8 位 ASCII 值（可选）。

* **输入**：code (number)，[allow8bits]
* **输出**：result (char)
* **示例**。以下示例将 ASCII 码 65 输出为字符 A。

Char(65)

### DateToString

返回给定日期的字符串表示。可以指定字符串的格式（可选）。

* **输入**：date (date)，[format (string)]
* **输出**：result (string)
* **示例**。以下示例会将 mylink.mydate 列中包含的日期输出为字符串。如果 mylink.mydate 包含日期 2009 年 8 月 18 日，那么输出字符串为“2009-08-18”：

DateToString(mylink.mydate)

以下示例会将 mylink.mydate 列中包含的日期输出为 dd:mm:yyyy 格式的字符串。如果 mylink.mydate 包含日期 2009 年 8 月 18 日，那么输出字符串为“18:08:2009”：

DateToString(mylink.mydate, "%dd:%mm:%yyyy")

### DateToDecimal

将给定的日期返回为压缩十进制值。如果您的目标十进制数指定了小数位，那么部分日期会在小数点后显示。您可以指定格式字符串来指定如何以十进制数形式存储日期（可选）。缺省格式字符串为“%yyyy%mm%dd”，因此，日期 2009-08-25 将存储为十进制数 20090825。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%yyyy-%mm-%dd”之类的格式字符串，这是因为压缩十进制值中不能存储连字符 (-)。以下标记对于与十进制值之间的转换有效：

%yyyy（四位数表示的年份）

%yy（两位数表示的年份）

%NNNNyy（截断的两位数年份）

%mm（两位数表示的月份）

%dd（两位数表示的一月中的第几天）

%ddd（三位数表示的一年中的第几天）

字面值数字 0 到 9 都有效。

* **输入**：basedate (date) [, format (string)]
* **输出**：converted\_date (decimal)
* **示例**。如果 mylink.basedate 列包含日期 2012-08-18，那么以下函数会将日期存储为十进制数 18082012：

DateToDecimal (mylink.basedate, "%dd%mm%yyyy")

如果 mylink.basedate 列包含日期 2012-08-18，并且目标列的长度为 10，小数位为 2，那么以下函数会将日期存储为十进制数 201208.18：

DateToDecimal (mylink.basedate)

### DecimalToDate

将给定的压缩十进制数返回为日期。在将十进制数转换为日期时，会忽略其符号和小数位。您可以指定格式字符串来指定如何以十进制数形式存储日期（可选）。缺省格式字符串为“%yyyy%mm%dd”，因此，日期 2009-08-25 将存储为十进制数 20090825。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%yyyy-%mm-%dd”之类的格式字符串，这是因为压缩十进制值中不能存储连字符 (-)。以下标记对于与十进制值之间的转换有效：

%yyyy（四位数表示的年份）

%yy（两位数表示的年份）

%NNNNyy（截断的两位数年份）

%mm（两位数表示的月份）

%dd（两位数表示的一月中的第几天）

%ddd（三位数表示的一年中的第几天）

字面值数字 0 到 9 都有效。

* **输入**：basedec (decimal) [, format (string)]
* **输出**：date
* **示例**。如果 mylink.mydecdata 列包含值 18082012，那么以下函数将返回日期 2012-08-18：

DecimalToDate (mylink.basedate, "%dd%mm%yyyy")

如果 mylink.mydecdata 列包含值 -201208.18，那么以下函数将返回日期 2012-08-18：

DecimalToDate (mylink.basedate)

### DecimalToDecimal

以具有目标列定义中指定精度和小数位的十进制表示形式返回给定的十进制数。参数 *rtype* 指定取整类型，并设置为以下某个值（可选）：

**ceil**。向正无穷方向对源字段取整。例如，1.4 -> 2，-1.6 -> -1。

**floor**。向负无穷方向对源字段取整。例如，1.6 -> 1，-1.4 -> -2。

**round\_inf**。通过向正无穷方向对正数值取整并向负无穷方向对负数值取整，向最接近的可表示值对源字段进行取整或截断。例如，1.4 -> 1，1.5 -> 2，-1.4 -> -1，-1.5 -> -2。

**trunc\_zero**。不论符号如何，废弃位于目标数据支持的最右侧小数位右侧的任何小数位。例如，如果目标数据是整数，那么截断所有小数位。如果目标数据是另一个具有更少小数位的小数，那么按照目标小数的小数位数进行取整或截断。例如，1.6 -> 1，-1.6 -> -1。

* **输入**：decimal (decimal) [,rtype (string)]
* **输出**：result (decimal)
* **示例**。如果 mylink.mydec 列包含十进制数 2.5345，那么以下函数将返回十进制数 00000002.54。

DecimalToDecimal(mylink.mydec,"ceil")

以下函数将返回十进制数 00000002.53。

DecimalToDecimal(mylink.mydec,"floor")

以下函数将返回十进制数 00000002.53。

DecimalToDecimal(mylink.mydec,"trunc\_zero")

以下函数将返回十进制数 00000002.53。

DecimalToDecimal(mylink.mydec,"round\_inf")

在所有这些示例中，目标十进制的长度为 10，小数位为 2。

### DecimalToDFloat

以 dfloat 表示形式返回给定的十进制数。参数“fix\_zero”指定所有零十进制值均视为有效（缺省情况下，全部由零组成的十进制数将视为无效）（可选）。

* **输入**：decimal (decimal) [,"fix\_zero"]
* **输出**：result (dfloat)
* **示例**。如果 mylink.mydec 列包含十进制数 00000004.00，那么以下函数将返回 dfloat 数 4.00000000000000000E+00。

DecimalToDFloat(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.00，那么以下函数将返回 dfloat 数 1.23440000000000000E+04。

DecimalToDFloat(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.120，那么以下函数将返回 dfloat 数 1.23441200000000010E+04。

DecimalToDFloat(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.120，那么以下函数将返回 dfloat 数 1.23441200000000010E+04。

DecimalToDFloat(mylink.mydec)

如果 mylink.mydec 列包含十进制数 00012344.000，那么以下函数将返回 dfloat 数 1.23440000000000000E+04。

DecimalToDFloat(mylink.mydec)

### DecimalToString

将给定的十进制数返回为字符串。参数“fix\_zero”指定所有零十进制值均视为有效（缺省情况下，全部由零组成的十进制数将视为无效）（可选）。这包括压缩十进制表示的符号位都为 0 以及都为内容数字的情况。该特性只在“fix\_zero”为 true 时才视为有效。

* **输入**：decimal (decimal) [,"fix\_zero"]
* **输出**：result (string)
* **示例**。如果 mylink.mydec 列包含十进制数 00000004.00，那么以下函数将返回字符串“4”：

DecimalToString(mylink.mydec,"suppress\_zero")

如果 mylink.mydec 列包含十进制数 00000004.00，那么以下函数将返回字符串“0000000000000000000000000004.0000000000”。

DecimalToString(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.00。那么以下函数将返回字符串“12344”。

DecimalToString(mylink.mydec,"suppress\_zero")

如果 mylink.mydec 列包含十进制数 00012344.00，那么以下函数将返回字符串“0000000000000000000000012344.0000000000”。

DecimalToString(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.120，那么以下函数将返回字符串“0000000000000000000000012344.1200000000”。

DecimalToString(mylink.mydec,"fix\_zero")

如果 mylink.mydec 列包含十进制数 00012344.120，那么以下函数将返回字符串“12344.12”：

DecimalToString(mylink.mydec,"suppress\_zero")

如果 mylink.mydec 列包含十进制数 00012344.120，那么以下函数将返回字符串“0000000000000000000000012344.120000000”。

DecimalToString(mylink.mydec)

如果 mylink.mydec 列包含十进制数 00012344.000，那么以下函数将返回字符串“0000000000000000000000012344.0000000000”。

DecimalToString(mylink.mydec)

### DecimalToTime

将给定的压缩十进制数返回为时间。您可以指定格式字符串来指定如何以十进制数形式存储时间（可选）。缺省格式字符串为“%hh%nn%ss”，因此，时间 14:03:22 将存储为十进制数 140322。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%hh:%nn:%ss”之类的格式字符串，这是因为压缩十进制值中不能存储冒号 (:)。以下标记对于与十进制值之间的转换有效：

%hh（使用 24 小时制的以两位数表示的小时数）

%nn（两位数表示的分钟数）

%ss（两位数表示的秒数）

%ss.*N*（由两位数表示的秒数加上所允许的小数位数。小数位数从 1 到 6，包括 1 和 6）。

字面值数字 0 到 9 都有效。

如果指定的格式包含微秒（例如，%ss.4），那么将在十进制值中推断小数点的位置。小数点的位置不必与指定的十进制小数位一致（例如，小数位 = 4）。

* **输入**：time (time) [, format (string)]
* **输出**：result (decimal)
* **示例**：如果 mylink.mytimedec 列包含十进制值 200658，那么以下函数将返回时间 20:06:58：

DecimalToTime(mylink.mytimedec)

如果 mylink.mytimedec 列包含十进制值 580620，那么以下函数将返回时间 20:06:58：

DecimalToTime(mylink.mytimedec, "%ss%nn%hh")

### DecimalToTimestamp

将给定的压缩十进制数返回为时间戳记。您可以指定格式字符串来指定如何以十进制数形式存储时间戳记（可选）。缺省格式字符串为“%yyyy%mm%dd%hh%nn%ss”，因此，时间戳记 2009-08-25 14:03:22 将存储为十进制数 20090825140322。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%yyyy/%mm/%dd%hh:%nn:%ss”之类的格式字符串，这是因为压缩十进制值中不能存储斜杠 (/) 和冒号 (:)。以下标记对于与十进制值之间的转换有效：

%yyyy（四位数表示的年份）

%yy（两位数表示的年份）

%NNNNyy（截断的两位数年份）

%mm（两位数表示的月份）

%dd（两位数表示的一月中的第几天）

%ddd（三位数表示的一年中的第几天）

%hh（使用 24 小时制的以两位数表示的小时数）

%nn（两位数表示的分钟数）

%ss（两位数表示的秒数）

%ss.*N*（由两位数表示的秒数加上所允许的小数位数。小数位数从 1 到 6，包括 1 和 6）。

字面值数字 0 到 9 都有效。

如果指定的格式包含微秒（例如，%ss.4），那么将在十进制值中推断小数点的位置。小数点的位置不必与指定的十进制小数位一致（例如，小数位 = 4）。

* **输入**：timestamp (timestamp) [, format (string)]
* **输出**：result (decimal)
* **示例**：如果 mylink.mytimestampdec 列包含值 19580818200658，那么以下函数将返回时间戳记 1958–08–18 20:06:58：

DecimalToTimestamp(mylink.mytimestampdec)

如果 mylink.mytimestampdec 列包含十进制值 200658580818，那么以下函数将返回时间戳记 1958–08–18 20:06:58：

DecimalToTimestamp(mylink.mytimestampdec, "%hh%nn%ss%yy%mm%dd")

### DFloatToDecimal

以十进制表示形式返回给定的 dfloat。参数 *rtype* 指定取整类型，并设置为以下某个值（可选）：

**ceil**。向正无穷方向对源字段取整。例如，1.4 -> 2，-1.6 -> -1。

**floor**。向负无穷方向对源字段取整。例如，1.6 -> 1，-1.4 -> -2。

**round\_inf**。通过向正无穷方向对正数值取整并向负无穷方向对负数值取整，向最接近的可表示值对源字段进行取整或截断。例如，1.4 -> 1，1.5 -> 2，-1.4 -> -1，-1.5 -> -2。

**trunc\_zero**。不论符号如何，废弃位于目标数据支持的最右侧小数位右侧的任何小数位。例如，如果目标数据是整数，那么截断所有小数位。如果目标数据是另一个具有更少小数位的小数，那么按照目标小数的小数位数进行取整或截断。例如，1.6 -> 1，-1.6 -> -1。

* **输入**：number (dfloat), [rtype (string)]
* **输出**：result (decimal)
* **示例**。如果 mylink.myfloat 列包含 dfloat 数 2.534，那么以下函数将返回十进制数 00000002.54。

DFloatToDecimal(mylink.mydec,"ceil")

如果 mylink.myfloat 列包含 dfloat 数 2.534，那么以下函数将返回十进制数 00000002.53。

DFloatToDecimal(mylink.mydec,"floor")

如果 mylink.myfloat 列包含 dfloat 数 2.534，那么以下函数将返回十进制数 00000002.53。

DFloatToDecimal(mylink.mydec,"trunc\_zero")

如果 mylink.myfloat 列包含 dfloat 数 2.534，那么以下函数将返回十进制数 00000002.53。

DFloatToDecimal(mylink.mydec,"round\_inf")

### DfloatToStringNoExp

使用指定小数位，以无指数的字符串表示形式返回给定的 dfloat。

* **输入**：number (dfloat), scale (string)
* **输出**：result (string)
* **示例**。如果 mylink.myfloat 列包含 dfloat 数 2.534，那么以下函数将返回字符串 00000002.50：

DfloatToStringNoExp(mylink.myfloat,2)

**IsValid**

返回给定的字符串对于给定类型是否有效。有效类型包括“date”、“decimal”、“dfloat”、“sfloat”、“int8”、“uint8”、“int16”、“uint16”、“int32”、“uint32”、“int64”、“uint64”、“raw”、“string”、“time”、“timestamp”和“ustring”。对于数据类型 date、time 和 timestamp，您可以指定格式字符串（可选）。格式字符串描述输入数据的格式与 date、time 或 timestamp 的缺省格式不同时，输入数据所使用的格式。date 的缺省格式为“%yyyy-%mm-%dd”。time 的缺省格式为“%hh:%mm:%ss”。timestamp 的缺省格式为“%yyyy-%mm-%dd %hh:%mm:%ss”。

* **输入**：type (string), teststring (string) [, format (string)]
* **输出**：result (int8)
* **示例**。如果 mylink.mystring 列包含字符串“1”，那么以下函数将返回值 1。

IsValid("int8",mylink.mystring)

如果 mylink.mystring 列包含字符串“380096.06”，那么以下函数将返回值 0。

IsValid("int8",mylink.mystring)

### IsValidDate

返回给定的值是否为有效的 date 类型。

* **输入**：testdate (date)
* **输出**：result (int8)
* **示例**。如果 mylink.mydate 列包含日期 2011-09-13，那么以下函数将返回值 1。

IsValidDate(mylink.mydate)

如果 mylink.mydate 列包含字符串“380096.06”，那么以下函数将返回值 0，这是因为转换后的字符串不是有效的 date 类型。

IsValidDate(StringTodate (mylink.mydate))

**IsValidDecimal**

返回给定的值是否为有效的 decimal 类型。如果将 allzerosflag 设置为 0，那么全零表示将视为无效。缺省情况下，allzerosflag 设置为 0。

* **输入**：testvalue (decimal) [, allzerosflag (uint8)]
* **输出**：result (int8)
* **示例**。如果 mylink.mynum 列包含值 310007.65，那么以下函数将返回值 1。

IsValidDecimal(mylink.mynum)

如果 mylink.mynum 列包含字符串“wake-robin”，那么以下函数将返回值 0，这是因为转换后的字符串不是有效的 decimal 类型。

IsValidDecimal(StringToDecimal (mylink.mynum))

### IsValidTime

返回给定的时间是否为有效的 time 类型。

* **输入**：testtime (time)
* **输出**：result (int8)
* **示例**。如果 mylink.mytime 列包含时间 23:09:22，那么以下函数将返回值 1：

IsValidTime(mylink.mytime)

如果 mylink.mydate 列包含字符串“IbnKayeed”，那么以下函数将返回值 0，这是因为转换后的字符串不是有效的 time 类型。

IsValidTime(StringToTime (mylink.mytime))

### IsValidTimestamp

返回给定的时间戳记是否为有效的 timestamp 类型。

* **输入**：testtimestamp (timestamp)
* **输出**：result (int8)
* **示例**。如果 mylink.mytimestamp 列包含时间 2011-09-13 23:09:22，那么以下函数将返回值 1：

IsValidTimestamp(mylink.mytimestamp)

如果 mylink.mytimestamp 列包含字符串“one of two”，那么以下函数将返回值 0，这是因为转换后的字符串不是有效的 timestamp 类型。

IsValidTimestamp(StringToTimestamp (mylink.mytimestamp))

### RawNumAt

返回指定 raw 字段中指定索引值的整数值。索引从 0 开始。

* **输入**：rawfield (raw), index (int32)
* **输出**：result (int32)
* **示例**。如果 mylink.myraw 列包含从字符串“hello”派生的 raw 值，那么以下函数将返回整数 0x68（字符 h 对应的 ASCII 码值）：

RawNumAt(mylink.myraw, 0)

如果 mylink.myraw 列包含从字符串“hello”派生的 raw 值，那么以下函数将返回 0，这是因为指定的索引超出了范围：

RawNumAt(mylink.myraw, 12)

### RawToString

以字符串表示形式返回给定的 raw 值。必须确保 raw 输入值包含一个字节序列，这些字节在使用输出字符串的目标字符集中为有效字符。例如，由于前导字节 0xE0 后应有范围在 [0x80..0xBF] 中的字节，因此 raw 值 { 0xE0 0x41 0x42 } 不是有效的 UTF-8 字符序列。如果将 raw 值 { xE0 x41 x42 } 传递给 RawToString 函数，那么在随后访问输出字符串时可能会出错，就像使用 UTF-8 对其进行编码一样。

* **输入**：rawfield (raw)
* **输出**：result (string)
* **示例**。如果 mylink.myraw 列包含值 { 0x31 0x31 0x30 0x35 0x32 0x32 0x30 0x39 }，那么以下函数将返回字符串“11052209”。

RawNumAt(mylink.myraw)

### Seq

根据 ASCII 字符生成数字代码值。您可以指定 allow8bits 参数来转换 8 位 ASCII 值（可选）。

* **输入**：Seq (char)
* **输出**：result (number)
* **示例**。以下示例将字符 A 输出为 ASCII 码 65。

Seq("A")

### SeqAt

返回给定字符串中指定位置上字符的数字代码点值。索引从 0 开始。如果指定的索引超出范围，那么该函数将返回 0。

* **输入**：basestring (string)，index (int32)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring 列包含字符串“horse”，那么以下函数将返回值 0x6F（即，字符 o 对应的 ASCII 码值）。

SeqAt(mylink.mystring, 1)

### StringToDate

以给定的格式返回给定字符串表示的日期。如果字符串包含缺省格式 yyyy-mm-dd 的日期，那么不必指定格式字符串。

* **输入**：string (string) [,format (string)]
* **输出**：result (date)
* **示例**：如果 mylink.mystring 列包含字符串“1958–08–18”，那么以下函数将返回日期 1958–08–18。

StringToDate(mylink.mystring)

如果 mylink.mystring 列包含字符串“18:08:1958”，那么以下函数将返回日期 1958–08–18

StringToDate(mylink.mystring,"%dd:%mm:%yyyy")

### StringToDecimal

以十进制表示形式返回给定的字符串。参数 *rtype* 指定取整类型，并设置为以下某个值（可选）：

**ceil**。向正无穷方向对源字段取整。例如，1.4 -> 2，-1.6 -> -1。

**floor**。向负无穷方向对源字段取整。例如，1.6 -> 1，-1.4 -> -2。

**round\_inf**。通过向正无穷方向对正数值取整并向负无穷方向对负数值取整，向最接近的可表示值对源字段进行取整或截断。例如，1.4 -> 1，1.5 -> 2，-1.4 -> -1，-1.5 -> -2。

**trunc\_zero**。不论符号如何，废弃位于目标数据支持的最右侧小数位右侧的任何小数位。例如，如果目标数据是整数，那么截断所有小数位。如果目标数据是另一个具有更少小数位的小数，那么按照目标小数的小数位数进行取整或截断。例如，1.6 -> 1，-1.6 -> -1。

* **输入**：string (string)，[rtype (string)]
* **输出**：result (decimal)
* **示例**。如果 mylink.mystring 列包含字符串“19982.22”，并且目标列定义为精度为 7，小数位为 2，那么以下函数将返回十进制数 19983.22。

StringToDecimal(mylink.mystring)

如果 mylink.mystring 列包含字符串“19982.2276”，并且目标列定义为精度为 7，小数位为 2，那么以下函数将返回十进制数 19983.23。

StringToDecimal(mylink.mystring,"ceil")

### StringToRaw

以 raw 表示形式返回字符串。

* **输入**：string (string)
* **输出**：result (raw)
* **示例**。如果 mylink.mystring 列包含字符串“hello”，并且目标列定义为二进制类型，那么以下函数将返回值 { 0x68 0x65 0x6C 0x6C 0x6F }。

StringToRaw(mylink.mystring)

### StringToTime

返回给定字符串的时间表示。

* **输入**：string (string)，[format (string)]
* **输出**：result (time)
* **示例**：如果 mylink.mystring 列包含字符串“20:06:58”，那么该函数将返回时间 20:06:58。

StringToTime(mylink.mystring)

如果 mylink.mystring 列包含字符串“20:6:58”，那么该函数将返回时间 20:06:58。

StringToTime(mylink.mystring,"%(h,s):$(n,s):$(s,s)")

### StringToTimestamp

返回给定字符串的时间表示。

* **输入**：string (string) [format (string)]
* **输出**：result (time)
* **示例**：如果 mylink.mystring 列包含字符串“1958–08–08 20:06:58”，那么该函数将返回时间戳记 1958–08–08 20:06:58。

StringToTimestamp(mylink.mystring)

如果 mylink.mystring 列包含字符串“8/ 8/1958 20: 6:58”，那么该函数将返回时间戳记 1958–08–08 20:06:58。

StringToTimestamp(mylink.mystring, "%(d,s)/%(m,s)/%yyyy%(h,s):$(n,s):$(s,s)")

### StringToUstring

返回给定字符串的 ustring，可以选择使用指定的映射（否则，将使用项目缺省值）。

* **输入**：string (string)，[mapname(string)]
* **输出**：result (ustring)
* **示例**：如果 mylink.mystring 列包含字符串“11052009”，那么以下函数将返回 ustring“11052009”

StringToUstring(mylink.mystring)

### TimestampToDate

返回给定时间戳记的日期。

* **输入**：timestamp (timestamp)
* **输出**：result (date)
* **示例**：如果 mylink.mytimestamp 列包含时间戳记 1958–08–18 20:06:58，那么以下函数将返回日期 1958–08–18：

TimestampToDate(mylink.mytimestamp)

### TimestampToDecimal

将给定的时间戳记返回为压缩十进制。您可以指定格式字符串来指定如何以十进制数形式存储时间戳记（可选）。缺省格式字符串为“%yyyy%mm%dd%hh%nn%ss”，因此，时间戳记 2009-08-25 14:03:22 将存储为十进制数 20090825140322。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%yyyy/%mm/%dd%hh:%nn:%ss”之类的格式字符串，这是因为压缩十进制值中不能存储斜杠 (/) 和冒号 (:)。以下标记对于与十进制值之间的转换有效：

%yyyy（四位数表示的年份）

%yy（两位数表示的年份）

%NNNNyy（截断的两位数年份）

%mm（两位数表示的月份）

%dd（两位数表示的一月中的第几天）

%ddd（三位数表示的一年中的第几天）

%hh（使用 24 小时制的以两位数表示的小时数）

%nn（两位数表示的分钟数）

%ss（两位数表示的秒数）

%ss.*N*（由两位数表示的秒数加上所允许的小数位数。小数位数从 1 到 6，包括 1 和 6）。

字面值数字 0 到 9 都有效。

如果指定的格式包含微秒（例如，%ss.4），那么将在十进制值中推断小数点的位置。小数点的位置不必与指定的十进制小数位一致（例如，小数位 = 4）。

* **输入**：timestamp (timestamp) [, format (string)]
* **输出**：result (decimal)
* **示例**：如果 mylink.mytimestamp 列包含时间戳记 1958–08–18 20:06:58，那么以下函数将返回十进制值 19580818200658：

TimestampToDecimal(mylink.mytimestamp)

如果 mylink.mytimestamp 列包含时间戳记 1958–08–18 20:06:58，那么以下函数将返回十进制值 200658580818：

TimestampToDecimal(mylink.mytimestamp, "%hh%nn%ss%yy%mm%dd")

### TimestampToString

返回给定时间戳记的字符串。

* **输入**：timestamp (timestamp) [format (string)]
* **输出**：result (string)
* **示例**：如果 mylink.mytimestamp 列包含时间戳记 1958–08–1820:06:58，那么该函数将返回字符串“1958–08–1820:06:58”。

TimestampToString(mylink.mytimestamp)

如果 mylink.mytimestamp 列包含时间戳记 1958–08–1820:06:58，那么该函数将返回字符串“18/08/1958 20:06:58”：

TimestampToString(mylink.mytimestamp, "%dd/%mm/%yyyy %hh:$nn:$ss")

### TimestampToTime

返回给定时间戳记的字符串表示。

* **输入**：timestamp (timestamp)
* **输出**：result (time)
* **示例**：如果 mylink.mytimestamp 列包含时间戳记 1958–08–1820:06:58，那么该函数将返回时间 20:06:58：

TimestampToTime(mylink.mytimestamp)

### TimeToString

返回给定时间的字符串。

* **输入**：timestamp (timestamp) [format (string)]
* **输出**：result (time)
* **示例**：如果 mylink.mytime 列包含时间 20:06:58，那么以下函数将返回字符串“20:06:58”：

TimeToString(mylink.mytime)

如果 mylink.mytime 列包含时间 20:06:58，那么以下函数将返回字符串“58:06:20”：

TimeToString(mylink.mytime, "%ss:$nn:$hh")

### TimeToDecimal

将给定的时间返回为压缩十进制。您可以指定格式字符串来指定如何以十进制数形式存储时间（可选）。缺省格式字符串为“%hh%nn%ss”，因此，时间 14:03:22 将存储为十进制数 140322。格式字符串只能指定包含数字的格式。例如，不能指定诸如“%hh:%nn:%ss”之类的格式字符串，这是因为压缩十进制值中不能存储冒号 (:)。以下标记对于与十进制值之间的转换有效：

%hh（使用 24 小时制的以两位数表示的小时数）

%nn（两位数表示的分钟数）

%ss（两位数表示的秒数）

%ss.*N*（由两位数表示的秒数加上所允许的小数位数。小数位数从 1 到 6，包括 1 和 6）。

字面值数字 0 到 9 都有效。

如果指定的格式包含微秒（例如，%ss.4），那么将在十进制值中推断小数点的位置。小数点的位置不必与指定的十进制小数位一致（例如，小数位 = 4）。

* **输入**：time (time) [, format (string)]
* **输出**：result (decimal)
* **示例**：如果 mylink.mytime 列包含时间 20:06:58，那么以下函数将返回十进制值 200658：

TimeToDecimal(mylink.mytime)

如果 mylink.mytime 列包含时间 20:06:58，那么以下函数将返回十进制值 580620：

TimeToDecimal(mylink.mytime, "%ss%nn%hh")

### UstringToString

返回给定 ustring 的字符串，可以选择使用指定的映射（否则，将使用项目缺省值）。

* **输入**：string (ustring) [, mapname(string)]
* **输出**：result (string)
* **示例**：如果 mylink.myustring 列包含 ustring“11052009”，那么以下函数将返回字符串“11052009”：

UstringToString(mylink.myustring)

## 字符串函数

字符串函数用于操作字符串。

以下函数位于表达式编辑器的“字符串”类别中。方括号表示参数是可选的。以下示例按照 Transformer 阶段的“派生”字段中所示来显示这些函数。

### AlNum

检查给定的字符串中是否只包含字母数字字符。

* **输入**：string (string)
* **输出**：true/false (int8)
* **示例**。如果 mylink.mystring1 包含字符串“OED\_75\_9\*E”，那么以下函数将返回值 -1 (false)。

AlNum(mylink.mystring1)

如果 mylink.mystring2 包含字符串“12 red roses”，那么以下函数将返回值 1 (true)。

AlNum(mylink.mystring2)

### Alpha

检查给定的字符串是否只包含字母字符。

* **输入**：string (string)
* **输出**：true/false (int8)
* **示例**。如果 mylink.mystring1 包含字符串“12 red roses”，那么以下函数将返回值 -1 (false)。

Alpha(mylink.mystring1)

如果 mylink.mystring2 包含字符串“twelve red roses”，那么以下函数将返回值 1 (true)。

Alpha(mylink.mystring2)

### CompactWhiteSpace

返回将所有连续空格减少到单个空格后得到的字符串。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“too   many    spaces”，那么以下函数将返回字符串“too many spaces”：

CompactWhiteSpace(mylink.mystring)

### Compare

比较用于排序的两个字符串。比较可以采用左对齐形式（缺省值），也可以采用右对齐形式。右对齐比较将指定字符串中的数字子串作为数字进行比较。在每个字符串中，数字字符串必须出现相同的字符位置。例如，对字符串 AB100 和 AB99 的右对齐比较结果表明 AB100 大于 AB99，这是因为 100 大于 99。对字符串 AC99 和 AB100 的右对齐比较结果表明 AC99 大于 AB100，这是因为 C 大于 B。

* **输入**：string1 (string)，string2 (string)，[justification (L or R)]
* **输出**：result (int8)，string1 小于 string2 时为 -1，两者相同时为 0，string1 大于 string2 时返回 1。
* **示例**。如果 mylink.mystring1 包含字符串“AB99”并且 mylink.mystring2 包含字符串“AB100”，那么以下函数将返回结果 1。

Compare(mylink.mystring1,mylink.mystring2,L)

如果 mylink.mystring1 包含字符串“AB99”并且 mylink.mystring2 包含字符串“AB100”，那么以下函数将返回结果 -1。

Compare(mylink.mystring1,mylink.mystring2,R)

### CompareNoCase

比较用于排序的两个字符串（忽略大小写）。

* **输入**：string1 (string)，string2 (string)
* **输出**：result (int8)，string1 小于 string2 时为 -1，两者相同时为 0，string1 大于 string2 时返回 1。
* **示例**。如果 mylink.mystring1 包含字符串“Chocolate Cake”并且 mylink.mystring2 包含字符串“chocolate cake”，那么以下函数将返回结果 1。

ComparNoCase(mylink.mystring1,mylink.mystring2)

### CompareNum

比较两个字符串中的前 *n* 个字符。

* **输入**：string1 (string)，string2 (string)，length (int16)
* **输出**：result (int8)，string1 小于 string2 时为 -1，两者相同时为 0，string1 大于 string2 时返回 1。
* **示例**。如果 mylink.mystring1 包含字符串“Chocolate”并且 mylink.mystring2 包含字符串“Choccy Treat”那么以下函数将返回结果 1。

ComparNum(mylink.mystring1,mylink.mystring2,4)

### CompareNumNoCase

比较两个字符串中的前 *n* 个字符（忽略大小写）。

* **输入**：string1 (string)，string2 (string)，length (int16)
* **输出**：result (int8)，string1 小于 string2 时为 -1，两者相同时为 0，string1 大于 string2 时返回 1。
* **示例**。如果 mylink.mystring1 包含字符串“chocolate”并且 mylink.mystring2 包含字符串“Choccy Treat”，那么以下函数将返回结果 1。

ComparNumNoCase(mylink.mystring1,mylink.mystring2,4)

### Convert

转换 *expression* 中指定的字符串中的字符。将 *fromlist* 中指定的字符转换为 *tolist* 中指定的字符。

* **输入**：fromlist (string)，tolist (string)，expression (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“NOW IS THE TIME”，那么以下函数将返回字符串“NOW YS XHE XYME”。

Convert("TI","XY",mylink.mystring1)

### Count

统计子串在字符串中出现的次数。

* **输入**：string (string)，substring (string)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars”，那么以下函数将返回 3。

Count(mylink.mystring1,"choc")

### Dcount

统计字符串中定界字段的数目。

* **输入**：string (string)，delimiter (string)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars”，那么以下函数将返回 3。

Dcount(mylink.mystring1,",")

### DownCase

将字符串中的所有大写字母转换为小写字母。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“CaMel cAsE”，那么以下函数将返回字符串“camel case”。

DownCase(mylink.mystring1)

### DQuote

在字符串两边加上双引号。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串 needs quotes，那么以下函数将返回字符串“needs quotes”。

DQuote(mylink.mystring1)

### Field

返回位于字符串中指定定界符之间的一个或多个子串。参数 *occurrence* 指定第几个出现的定界符用作终止符。参数 *number* 指定返回多少个子串（可选）。

* **输入**：string (string)，delimiter (string)，occurrence (int32), [number (int32)]
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars, chocolate dippers”，那么以下函数将返回字符串“chocolate ice cream”。

Field(mylink.mystring1,",",2)

如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars, chocolate dippers”，那么以下函数将返回字符串“chocolate ice cream, chocolate bars”。

Field(mylink.mystring1,",",2,2)

### Index

查找子串的起始字符位置。参数 *occurrence* 指定要查找第几个出现的子串。

* **输入**：string (string) substring (string) occurrence (int32)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars, chocolate dippers”，那么以下函数将返回值 18。

Index(mylink.mystring1,"chocolate",2)

### Left

返回字符串中最左侧的 *n* 个字符。

* **输入**：string (string) number (int32)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars, chocolate dippers”，那么以下函数将返回字符串“chocolate”。

Left(mylink.mystring1,9)

### Len

返回字符中字符串的长度。

* **输入**：string (string)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate”，那么以下函数将返回值 9。

Len(mylink.mystring1)

### Num

如果字符串可以转换为数字，那么返回 1。否则，返回 0。

* **输入**：string (string)
* **输出**：result (int32)
* **示例**。如果 mylink.mystring1 包含字符串“22”，那么以下函数将返回值 1。

Num(mylink.mystring1)

如果 mylink.mystring1 包含字符串“twenty two”，那么以下函数将返回值 0。

Num(mylink.mystring1)

### PadString

返回用指定的填充字符数填充的字符串。

* **输入**：string (string) padstring (string) padlength (int32)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“AB175”，那么以下函数将返回字符串“AB17500000”。

PadString(mylink.mystring1,"0",5)

### Right

返回字符串中最右侧的 *n* 个字符。

* **输入**：string (string) number (int32)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“chocolate drops, chocolate ice cream, chocolate bars, chocolate dippers”，那么以下函数将返回字符串“dippers”。

Right(mylink.mystring1,7)

### Soundex

返回一个代码，该代码用于根据 SOUNDEX 评估的标准开放式算法来识别发音（大致）相同的一组词。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“Griffin”，那么以下函数将返回代码“G615”。

Soundex(mylink.mystring1)

如果 mylink.mystring1 包含字符串“Griphin”，那么以下函数也将返回代码“G615”。

Soundex(mylink.mystring1)

### Space

返回包含 *n* 个空格字符的字符串。

* **输入**：length (int32)
* **输出**：result (string)
* **示例**。如果 mylink.mylength 包含数字 100，那么以下函数将返回包含 100 个空格字符的字符串。

Space(mylink.mylength)

### SQuote

将字符串用单引号括起来。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串 needs quotes，那么以下函数将返回字符串“needs quotes”。

SQuote(mylink.mystring1)

### Str

将字符串重复指定的次数。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串需要“choc”的字符串，那么以下函数将返回字符串“chocchocchocchocchoc”。

Str(mylink.mystring1,5)

### StripWhiteSpace

返回删除了字符串中所有空格字符后得到的该字符串。

* **输入**：string (string) repeats (int32)
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“too   many    spaces”，那么以下函数将返回字符串“toomanyspaces”：

StripWhiteSpace(mylink.mystring)

### Trim

除去所有前导空格、结尾空格和跳格，并将内部出现次数减少为 1。参数 *stripchar* 指定除空格或跳格之外的字符（可选）。参数 *options* 指定要执行的 Trim 操作的类型，并包含以下某一个或多个值（可选）：

**A** 除去所有出现的 *stripchar*

**B** 除去出现的前导和结尾 *stripchar*

**D** 除去前导、结尾和冗余空格字符

**E** 除去结尾空格字符

**F** 除去前导空格字符

**L** 除去所有出现的前导 *stripchar*

**R** 除去出现的前导、结尾和冗余 *stripchar*

**T**除去所有出现的结尾 *stripchar*

* **输入**：string (string) [stripchar (string)] [options (string)]
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“ String with whitespace ”，那么以下函数将返回字符串“String with whitespace”：

Trim(mylink.mystring)

如果 mylink.mystring 包含字符串“..Remove..redundant..dots....”，那么以下函数将返回字符串“Remove.redundant.dots”：

Trim(mylink.mystring,".")

如果 mylink.mystring 包含字符串“Remove..all..dots....”，那么以下函数将返回字符串“Removealldots”：

Trim(mylink.mystring,".","A")

如果 mylink.mystring 包含字符串“Remove..trailing..dots....”，那么以下函数将返回字符串“Remove..trailing..dots”：

Trim(mylink.mystring,".","T")

### TrimB

除去字符串中所有结尾空格和跳格。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“too many trailing spaces    ”，那么以下函数将返回字符串“too many trailing spaces”：

TrimB(mylink.mystring)

### TrimF

除去字符串中所有前导空格和跳格。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“    too many leading spaces”，那么以下函数将返回字符串“too many leading spaces”：

TrimF(mylink.mystring)

### TrimLeadingTrailing

除去字符串中所有前导空格、结尾空格和跳格。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring 包含字符串“ too many spaces ”，那么以下函数将返回字符串“too many spaces”：

TrimLeadingTrailing(mylink.mystring)

### UpCase

将字符串中的所有小写字母转换为大写字母。

* **输入**：string (string)
* **输出**：result (string)
* **示例**。如果 mylink.mystring1 包含字符串“CaMel cAsE”，那么以下函数将返回字符串“CAMEL CASE”。

UpCase(mylink.mystring1)

## 数字函数

数字函数用于抽取十进制数或浮点数中的尾数。表达式编辑器中的“数字”类别还包含类型为 casting 的函数，您可以使用该函数对数据类型为 double、float 或 integer 的数字进行强制类型转换。

方括号表示参数是可选的。以下示例按照 Transformer 阶段的“派生”字段中所示来显示这些函数。

当您使用数字字段执行数学运算时，类型为 casting 的函数可为您提供帮助。例如，如果在 Parallel Transformer 阶段中使用类型为 float 的输出列（派生自类型为 integer 的输入列）进行计算，那么无论其类型是否为 float，结果都将派生为 integer。如果希望使用整数操作数进行的计算得到非整数结果，那么可以使用类型为 casting 的函数来将整数操作数强制转换为非整数操作数。

### AsDouble

将给定数字视为双精度数。

* **输入**：number
* **输出**：number (double)
* **示例**。在以下表达式中，输入列 mynumber 包含一个整数，但是函数输出一个双精度数。如果 mylink.mynumber 包含值 56，那么以下两个函数等价，并返回值 1.29629629629629619E+01：
* AsDouble(56/4.32)

AsDouble(mylink.mynumber/4.32)

### AsFloat

将给定数字视为浮点数。

* **输入**：number
* **输出**：number (float)
* **示例**。在以下表达式中，输入列 mynumber 包含一个整数，但是函数输出一个浮点数。如果 mylink.mynumber 包含值 56，那么以下两个函数等价，并返回值 1.29629629629629619E+01：
* AsFloat(56/4.32)

AsFloat(mylink.mynumber/4.32)

### AsInteger

将给定数字视为整数。

* **输入**：number
* **输出**：number (integer)
* **示例**。在以下表达式中，输入列 mynumber 包含一个双精度数，但是函数输出一个整数。如果 mylink.mynumber 包含值 56，那么以下两个函数等价，并返回值 12：
* AsInteger(56/4.32)

AsInteger(mylink.mynumber/4.32)

### MantissaFromDecimal

返回给定十进制数的尾数。

* **输入**：number (decimal)
* **输出**：result (dfloat)
* **示例**。如果 mylink.number 包含数字 243.7675，那么以下两个函数等价，并返回值 7675：
* MantissaFromDecimal(243.7675)

MantissaFromDecimal(mylink.mynumber)

### MantissaFromDFloat

返回给定双精度浮点数的尾数。

* **输入**：number (dfloat)
* **输出**：result (dfloat)
* **示例**。如果 mylink.number 包含数字 1.234412000000000010E +4，那么以下两个函数返回值 1：

MantissaFromDFloat(mylink.mynumber)

## 日期和时间函数

您可以使用日期和时间函数对 Transformer 阶段中的日期和时间执行各种操作。

在自变量中指定日期、时间或时间戳记的函数使用具有特定格式的字符串：

* 对于日期，格式为 %*yyyy*-%*mm*-%*dd*
* 对于时间，格式为 %*hh*:%*nn*:%*ss*。如果扩展为包含微秒，那么格式为 %*hh*:%*nn*:%*ss.x*，其中 *x* 是为秒指定的小数位数。
* 对于时间戳记，格式为 %*yyyy*-%*mm*-%*dd* %*hh*:%*nn*:%*ss*。如果扩展为包含微秒，那么格式为 %*yyyy*-%*mm*-%*dd* %*hh*:%*nn*:%*ss.x*，其中 *x* 是为秒指定的小数位数。

在自变量中包含星期几的函数使用字符串来指定星期几。星期几可指定为三个字母的缩写，也可指定为完整名称。例如，字符串“thu”和“thursday”都是有效的。

以下函数位于表达式编辑器的日期和时间类别中。方括号表示参数是可选的。以下示例按照 Transformer 阶段的“派生”字段中所示来显示这些函数。

### CurrentDate

以日期格式返回作业运行的日期。

* **输入**：-
* **输出**：date
* **示例**。您可以使用该函数将包含日期的新列添加到 Transformer 阶段输出的数据：

CurrentDate()

### CurrentTime

以时间格式返回作业运行的时间。

* **输入**：-
* **输出**：time
* **示例**。您可以使用该函数将包含时间的新列添加到 Transformer 阶段输出的数据：

CurrentTime()

### CurrentTimeMS

以时间格式返回作业运行的时间，包含微秒。

* **输入**：-
* **输出**：time
* **示例**。您可以使用该函数将包含时间的新列添加到 Transformer 阶段输出的数据。必须将列元数据中的扩展字段设置为微秒以包含完整的时间：

CurrentTimeMS()

### CurrentTimestamp

以时间戳记格式返回作业运行的时间戳记（指定日期和时间）。

* **输入**：-
* **输出**：timestamp
* **示例**。您可以使用该函数将包含时间戳记的新列添加到 Transformer 阶段输出的数据：

CurrentTimestamp()

### CurrentTimestampMS

以时间戳记格式返回作业运行的时间戳记（指定日期和时间），时间部分包含微秒。

* **输入**：-
* **输出**：timestamp
* **示例**。您可以使用该函数将包含时间戳记的新列添加到 Transformer 阶段输出的数据。必须将列元数据中的扩展字段设置为微秒以包含完整的时间戳记：

CurrentTimestampMS()

### DateFromDaysSince

通过向基线日期添加一个整数来返回一个日期。该整数可以是负数，这样会返回比该基线日期更早的日期。

* **输入**：number (int32), [baseline\_date (date)]
* **输出**：date
* **示例**。如果 mylink.myintcol 包含整数 18250，mylink.mydatecol 包含日期 1958–08–18，那么以下三个函数是等效的，都返回日期 2008–08–05：
* DateFromDaysSince(18250,"1958-08-18")
* DateFromDaysSince(mylink.myintcol,"1958-08-18")

DateFromDaysSince(mylink.myintcol,mylink.mydatecol)

如果 mylink.mynegintcol 包含整数 -1，mylink.mydatecol 包含日期 1958–08–18，那么以下三个函数是等效的，都返回日期 1958–08–17：

DateFromDaysSince(-1,"1958-08-18")

DateFromDaysSince(mylink.mynegintcol,"1958-08-18")

DateFromDaysSince(mylink.mynegintcol,mylink.mydatecol)

### DateFromComponents

根据由三个单独的值指定的年月日来返回一个日期。

* **输入**：(int32), months (int32), dayofmonth (int32)
* **输出**：date
* **示例**。如果 mylink.yearcol 包含值 2010，mylink.monthcol 包含值 12，mylink.dayofmonthcol 包含值 2，那么以下两个函数是等效的，都返回日期 2010–12–02。
* DateFromComponents(2010, 12, 2)

DateFromComponents(mylink.yearcol, mylink.monthcol, mylink.dayofmonthcol)

### DateFromJulianDay

根据指定的儒略日返回一个日期。

* **输入**：julianday (uint32)
* **输出**：date
* **示例**。如果 mylink.myjulcol 包含值 2454614，那么以下两个函数是等效的，都返回日期 2008–05–27。
* DateFromJulianDay(2454614)

DateFromJulianDay(mylink.myjulcol)

### DateOffsetByComponents

返回指定的日期应用了指定的年偏移量、月偏移量和日偏移量（每个偏移量都以单独的值指定）之后所得到的日期。每个偏移值都可以是正数、零或负数。

* **输入**：basedate (date), year\_offset (int32), month\_offset (int32), dayofmonth\_offset (int32)
* **输出**：date
* **示例**。如果 mylink.basedate 包含 2011-08-18，mylink.yearos 包含值 2，mylink.monthos 包含值 0，mylink.dayofmonthosol 包含值 0，那么以下两个函数是等效的，都返回日期 2013–08–18。
* DateOffsetByComponents("2011-08-18", 2011, 8, 18)
* DateOffsetByComponents(mylink.basedate, mylink.yearos, mylink.monthos,

mylink.dayofmonthos)

* 如果 mylink.basedate 包含 2011-08-18，mylink.yearos 包含值 -2，mylink.monthos 包含值 0，mylink.dayofmonthosol 包含值 0，那么以下两个函数是等效的，都返回日期 2009–08–18。
* DateOffsetByComponents("2011-08-18", 2011, 8, 18)
* DateOffsetByComponents(mylink.basedate, mylink.yearos, mylink.monthos,

mylink.dayofmonthos)

### DaysSinceFromDate

返回源日期到指定日期之间的天数。

* **输入**：source\_date, given\_date
* **输出**：days\_since (int32)
* **示例**。如果 mylink.mysourcedate 包含日期 1958–08–18，mylink.mygivendate 包含日期 2008–08–18，那么以下两个函数是等效的，都返回整数值 18263。
* DaysSinceFromDate(mylink.mysourcedate, mylink.mygivendate)

DaysSinceFromDate("1958-08-18","2008-08-18")

### DaysInMonth

返回指定的基本日期中月份的天数。

* **输入**：basedate (date)
* **输出**：daysinmonth (int32)
* **示例**。如果 mylink.mysourcedate 包含日期 1958–08–18，那么以下两个函数是等效的，都返回整数值 31。
* DaysInMonth(mylink.mysourcedate)

DaysInMonth("1958-08-18")

### DaysInYear

返回指定的基本日期中年份的天数。

* **输入**：basedate (date)
* **输出**：daysinyearh (int32)
* **示例**。如果 mylink.mysourcedate 包含日期 2012–08–18，那么以下两个函数是等效的，都返回整数值 366。
* DaysInYear(mylink.mysourcedate)

DaysInYear("2012-08-18")

如果 mylink.mysourcedate 包含日期 2011–08–18，那么以下两个函数是等效的，都返回整数值 365。

DaysInYear(mylink.mysourcedate)

DaysInYear("2011-08-18")

### DateOffsetByDays

返回指定的日期应用了指定天数之后所得到的日期。偏移值可以是正数、零或负数。

* **输入**：basedate (date), dayoffset (int32)
* **输出**：date
* **示例**。如果 mylink.basedate 包含 2011-08-18，mylink.dayoffset 包含值 2，那么以下两个函数是等效的，都返回日期 2011–08–20。
* DateOffsetByDays("2011-08-18", 2)

DateOffsetByDays(mylink.basedate, mylink.dayoffset)

* 如果 mylink.basedate 包含 2011-08-18，mylink.dayoffset 包含值 -31，那么以下两个函数是等效的，都返回日期 2011–07–18。
* DateOffsetByDays("2011-08-18", -31)

DateOffsetByDays(mylink.basedate, mylink.dayoffset)

### HoursFromTime

返回时间的小时部分。

* **输入**：time
* **输出**：hours (int8)
* **示例**。如果 mylink.mytime 包含时间 22:30:00，那么以下两个函数是等效的，都返回整数值 22。
* HoursFromTime(mylink.mytime)

HoursFromTime("22:30:00")

### JulianDayFromDate

根据指定的日期返回对应的儒略日。

* **输入**：date
* **输出**：julianday (int32)
* **示例**。如果 mylink.mydate 包含日期 2008–05–27，那么以下两个函数是等效的，都返回值 2454614。
* JulianDayFromDate("2008–05–27")

JulianDayFromDate(mylink.mydate)

### MicroSecondsFromTime

返回时间的微秒部分。

* **输入**：time
* **输出**：microseconds (int32)
* **示例**。如果 mylink.mytime 包含时间 22:30:00.32，那么以下函数返回值 320000：

MicroSecondsFromTime(mylink.mytime)

### MidnightSecondsFromTime

返回从午夜到指定时间的秒数。

* **输入**：time
* **输出**：seconds (int8)
* **示例**。如果 mylink.mytime 包含时间 00:30:52，那么以下两个函数是等效的，都返回值 1852：
* MidnightSecondsFromTime("00:30:52")

MidnightSecondsFromTime(mylink.mytime)

### MinutesFromTime

返回时间的分钟部分。

* **输入**：time
* **输出**：minutes (int8)
* **示例**。如果 mylink.mytime 包含时间 22:30:52，那么以下两个函数是等效的，都返回值 30：
* MinutesFromTime("22:30:52")

MinutesFromTime(mylink.mytime)

### MonthDayFromDate

返回指定日期的日部分。

* **输入**：date
* **输出**：day (int8)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 18：
* MonthDayFromDate("2008-08-18")

MonthDayFromDate(mylink.mydate)

### MonthFromDate

返回指定日期的月份。

* **输入**：date
* **输出**：month\_number (int8)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 8：
* MonthFromDate("2008-08-18")

MonthDayDate(mylink.mydate)

### NextWeekdayFromDate

返回源日期之后第一个指定的星期几的日期。星期几可以用完整名称指定，例如 thursday；也可以用三个字母缩写指定，例如 thu。

* **输入**：sourcedate (date), day\_of\_week (string)
* **输出**：date
* **示例**。如果 mylink.mysourcedate 包含日期 2008-08-18，那么以下两个函数是等效的，都将返回值 2008–08–21：
* NextWeekdayFromDate("2008-08-18", "thursday")

NextWeekdayFromDate(mylink.mysourcedate, "thu")

### NthWeekdayFromDate

返回从源日期偏移指定星期数后的指定星期几的日期。星期几可以用完整名称指定，例如 thursday；也可以用三个字母缩写指定，例如 thu。偏移量可以是正数、负数或零。

* **输入**：basedate (date), day\_of\_week (string), week\_offset (int32)
* **输出**：date
* **示例**。如果 mylink.mydate 包含日期 2009-08-18，那么以下两个函数是等效的，都返回值 2009–08–27：
* NthWeekdayFromDate("2009-08-18", "thursday", 1)

NthWeekdayFromDate(mylink.mydate, "thu", 1)

如果 mylink.mydate 包含日期 2009-08-18，那么以下两个函数是等效的，都返回值 2009–08–06：

NthWeekdayFromDate("2009-08-18", "thursday", -2)

NthWeekdayFromDate(mylink.mydate, "thu", -2)

### PreviousWeekdayFromDate

返回源日期之前第一个指定的星期几的日期。星期几可以用完整名称指定，例如 thursday；也可以用三个字母缩写指定，例如 thu。

* **输入**：sourcedate, day\_of\_week (string)
* **输出**：date
* **示例**。如果 mylink.mysourcedate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 2008–08–14：
* PreviousWeekdayFromDate("2008-08-18", "thursday")

PreviousWeekdayFromDate(mylink.mysourcedate, "thu")

### SecondsFromTime

返回时间的秒部分。

* **输入**：time
* **输出**：seconds (dfloat)
* **示例**。如果 mylink.mytime 包含时间 22:30:52，那么以下两个函数是等效的，都返回值 52：
* SecondsFromTime("22:30:52")

SecondsFromTime(mylink.mytime)

### SecondsSinceFromTimestamp

返回两个时间戳记之间的秒数。

* **输入**：timestamp, timestamp\_base
* **输出**：seconds (dfloat)
* **示例**。如果 mylink.mytimestamp 包含时间戳记 2008–08–18 22:30:52，mylink.mytimestamp\_base 包含时间戳记 2008–08–19 22:30:52，那么以下两个函数是等效的，都返回值 -86400：
* SecondsSinceFromTimestamp("2008–08–18 22:30:52","2008–08–19 22:30:52")

SecondsSinceFromTimestamp(mylink.mytimestamp, mylink.mytimestamp\_base)

### TimeDate

以格式字符串形式返回系统时间和日期。

* **输入**：-
* **输出**：system\_time\_and\_date (string)
* **示例**。如果作业在 2008 年 6 月 20 日下午 4:21 运行，那么以下函数将返回字符串“16:21:48 20 Jun 2008”。

TimeDate()

### TimeFromComponents

按照以四个单独的值指定的小时、分钟、秒和微秒组成的时间。

* **输入**：hours (int32), minutes (int32), seconds (int32), microseconds (int32)
* **输出**：time
* **示例**。如果 mylink.hourcol 包含值 10，mylink.mincol 包含值 12，mylink.seccol 包含值 2，mylink.mseccol 包含 0，那么以下两个函数是等效的，都返回时间 10:12:02.0：
* TimeFromComponents(10, 12, 2, 0)

TimeFromComponents(mylink.hourcol, mylink.mincol, mylink.seccol, mylink.mseccol)

### TimeFromMidnightSeconds

返回自午夜起指定秒数之后的时间。

* **输入**：seconds (dfloat)
* **输出**：time
* **示例**。如果 mylink.mymidnightseconds 包含值 240，那么以下两个函数是等效的，都返回值 00:04:00：
* TimeFromMidnightSeconds("240")

TimeFromMidnightSeconds(mylink.mymidnightseconds)

### TimeOffsetByComponents

返回指定的时间应用了指定的小时偏移量、分钟偏移量和秒偏移量（每个偏移量都以单独的值指定）之后所得到的时间。秒偏移量可以包含小数。

* **输入**：basetime (time), hour\_offset (int32), minute\_offset (int32), second\_offset (dfloat)
* **输出**：time
* **示例**。如果 mylink.basetime 包含 14:05:29，mylink.houros 包含值 2，mylink.minos 包含值 0，mylink.secos 包含值 20，那么以下两个函数是等效的，都返回时间 16:05:49。
* TimeOffsetByComponents("14:05:29", 2, 0, 20)

TimeOffsetByComponents(mylink.basetime, mylink.houros, mylink.minos, mylink.secos)

### TimeOffsetBySeconds

返回指定的时间应用了指定的秒偏移量之后所得到的时间。秒偏移量可以包含小数。

* **输入**：basetime (time), second\_offset (dfloat)
* **输出**：time
* **示例**。如果 mylink.basetime 包含 14:05:29.30，mylink.secos 包含值 2.5，那么以下两个函数是等效的，都返回时间 14:05:31.80：
* TimeOffsetByComponents("14:05:29.30", 2.5)

TimeOffsetByComponents(mylink.basetime, mylink.secos)

### TimestampFromDateTime

返回指定的日期和时间的时间戳记。

* **输入**：date, time
* **输出**：timestamp
* **示例**。如果 mylink.mydate 包含日期 2008–08–18，mylink.mytime 包含时间 22:30:52，那么以下两个函数是等效的，都返回时间戳记 2008–08–18 22:30:52：
* TimestampFromDateTime("2008–08–18","22:30:52")

TimestampFromDateTime(mylink.mydate,mylink.mytime)

### TimestampFromSecondsSince

返回基本时间戳记加上指定秒数之后所得到的时间戳记。

* **输入**：seconds (dfloat), [base\_timestamp]
* **输出**：timestamp
* **示例**。如果 mylink.myseconds 包含值 2563，mylink.timestamp\_base 包含时间戳记 2008–08–18 22:30:52，那么以下两个函数是等效的，都返回时间戳记 2008–08–18 23:13:35：
* TimestampFromSecondsSince("2563","2008–08–18 22:30:52")

TimestampFromSecondsSince(mylink.myseconds,mylink.timestamp\_base)

### TimestampFromTimet

按照指定的 UNIX time\_t 值返回时间戳记。

* **输入**：timet (int32)
* **输出**：timestamp
* **示例**。如果 mylink.mytimet 包含值 1234567890，那么以下两个函数是等效的，都返回时间戳记 2009–02–13 23:31:30：
* TimestampFromTimet("1234567890")

TimestampFromTimet(mylink.mytimet)

### TimestampOffsetByComponents

返回指定的时间戳记应用了指定的年偏移量、月偏移量、日偏移量、小时偏移量、分钟偏移量和秒偏移量（每个偏移量都以单独的值指定）之后所得到的时间戳记。秒偏移量可以包含小数。

* **输入**：basetimestamp (timestamp), year\_offset (int32), month\_offset (int32), dayofmonth\_offset (int32), hour\_offset (int32), minute\_offset (int32), second\_offset (dfloat)
* **输出**：timestamp
* **示例**。如果 mylink.basetimestamp 包含 2009-08-18 14:05:29，mylink.yearos 包含 0，mylink.monthos 包含值 2，mylink.dayos 包含值 -4，mylink.houros 包含值 2，mylink.minos 包含值 0，mylink.secos 包含值 20，那么以下两个函数是等效的，都返回时间戳记 2009-10-14 16:05:49。
* TimestampOffsetByComponents("2009-08-18 14:05:29", 0, 2, -4, 2, 0, 20)
* TimestampOffsetByComponents(mylink.basetimestamp, mylink.houros,

mylink.minos, mylink.secos)

### TimestampOffsetBySeconds

返回指定的时间戳记应用了指定的秒偏移量之后所得到的时间戳记。秒偏移量可以包含小数。

* **输入**：basetimestamp (timestamp), second\_offset (dfloat)
* **输出**：timestamp
* **示例**。如果 mylink.basetimestamp 包含 2009-08-18 14:05:29，mylink.secos 包含值 32760，那么以下两个函数是等效的，都返回时间戳记 2009-08-18 23:11:29：
* TimeOffsetBySeconds("2009-08-18 14:05:29", 32760)
* TimeOffsetBySeconds

(mylink.basetimestamp, mylink.secos)

### TimetFromTimestamp

返回指定的时间戳记的 UNIX time\_t 值。

* **输入**：timestamp
* **输出**：timet (int32)
* **示例**。如果 mylink.mytimestamp 包含值 2009–02–13 23:31:30，那么以下两个函数是等效的，都返回值 1234567890：
* TimestampFromTimet("2009–02–13 23:31:30")

TimestampFromTimet(mylink.mytimestamp)

### WeekdayFromDate

返回指定日期距离所在星期第一天的天数。可以选择用 Origin\_day 指定每周第一天是星期几，缺省情况下每周第一天是星期日 (Sunday)。

* **输入**：date, [origin\_day]
* **输出**：day (int8)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 1：
* WeekdayFromDate("2008-08-18")

WeekdayFromDate(mylink.mydate)

如果 mylink.mydate 包含日期 2008-08-18，mylink.origin\_day 包含 saturday，那么以下两个函数是等效的，都返回值 2：

WeekdayFromDate("2008-08-18","saturday")

WeekdayFromDate(mylink.mydate,mylink.origin\_day)

### YeardayFromDate

返回指定的日期在该年份中所处的天数。

* **输入**：date
* **输出**：day (int16)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 231：
* YeardayFromDate("2008-08-18")

YeardayFromDate(mylink.mydate)

### YearFromDate

返回指定的日期所在年份。

* **输入**：date
* **输出**：year (int16)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 2008：
* YearFromDate("2008-08-18")

YearFromDate(mylink.mydate)

### YearweekFromDate

返回指定的日期在该年份中所处的星期数。

* **输入**：date
* **输出**：week (int16)
* **示例**。如果 mylink.mydate 包含日期 2008-08-18，那么以下两个函数是等效的，都返回值 33：
* YearweekFromDate("2008-08-18")

YearweekFromDate(mylink.mydate)

## Null 处理函数

您可以在 Transformer 阶段中使用 Null 处理函数来处理派生中的 Null 值。

如果在输出列表达式中使用输入列，任意输入列中的 Null 值会导致将一个 Null 值写入到输出列中。但是，您可以使用 Null 处理函数来显式处理 Null 值。

“Null 处理”类别中包含了以下函数。方括号表示参数是可选的。以下示例按照 Transformer 阶段的“派生”字段中所示来显示这些函数。

### IsNotNull

当表达式无法求出 Null 值时返回 true。

* **输入**：任意值
* **输出**：true/false (int8)
* **示例**。如果某输出列的“派生”字段包含以下代码，那么 Transformer 阶段会检查名为 mylink.mycolumn 的输入列是否包含 Null 值。如果输入列不包含 Null 值，那么输出列将包含该输入列的值。如果输入列包含 Null 值，那么输出列将包含字符串 NULL。

If IsNotNull(mylink.mycolumn) Then mylink.mycolumn Else "NULL"

### IsNull

当表达式的求值结果为 Null 值时返回 true。

* **输入**：任意值
* **输出**：true/false (int8)
* **示例**。如果某输出列的“派生”字段包含以下代码，那么 Transformer 阶段会检查名为 mylink.mycolumn 的输入列是否包含 Null 值。如果输入列包含 Null 值，那么输出列将包含字符串 NULL。如果输入列不包含 Null 值，那么输出列将包含该输入列的值。

If IsNull(mylink.mycolumn) Then "NULL" Else mylink.mycolumn

### NullToEmpty

输入列为 NULL 时返回空字符串，否则返回输入列值。

* **输入**：输入列
* **输出**：输入列值或空字符串
* **示例**。如果某输出列的“派生”字段包含以下代码，那么 Transformer 阶段会检查名为 mylink.mycolumn 的输入列是否包含 Null 值。如果输入列包含 Null 值，那么输出列将包含空字符串。如果输入列包含 Null 值，那么输出列将包含该输入列的值。

NullToEmpty(mylink.mycolumn)

### NullToZero

输入列为 NULL 时返回零，否则返回输入列值。

* **输入**：输入列
* **输出**：输入列值或零
* **示例**。如果某输出列的“派生”字段包含以下代码，那么 Transformer 阶段会检查名为 mylink.mycolumn 的输入列是否包含 Null 值。如果输入列包含 Null 值，那么输出列将包含零。如果输入列包含 Null 值，那么输出列将包含该输入列的值。

NullToZeroy(mylink.mycolumn)

### NullToValue

输入列为 NULL 时返回指定值，否则返回输入列值。

* **输入**：输入列，*value*
* **输出**：输入列值或 *value*
* **示例**。如果某输出列的“派生”字段包含以下代码，那么 Transformer 阶段会检查名为 mylink.mycolumn 的输入列是否包含 Null 值。如果输入列包含 Null 值，那么输出列将包含 42。如果输入列包含 Null 值，那么输出列将包含该输入列的值。

NullToValue(mylink.mycolumn,42)

### SetNull

将 Null 值赋给目标列。

* **输入**：-
* **输出**：-
* **示例**。如果输出列的“派生”字段包含以下代码，那么 Transformer 阶段会将输出列设置为 NULL：

setnull()

## 线头标志区分

