

root

[Go Up](#)

Table of Contents

BLAS.ecl
BundleBase.ecl
Date.ecl
File.ecl
math.ecl
Metaphone.ecl
str.ecl
Uni.ecl

BLAS

[Go Up](#)

IMPORTS

lib_eclblas |

DESCRIPTIONS

MODULE BLAS

BLAS

No Documentation Found

Children

1. [Types](#) : No Documentation Found
2. [ICellFunc](#) : Function prototype for Apply2Cell
3. [Apply2Cells](#) : Iterate matrix and apply function to each cell
4. [dasum](#) : Absolute sum, the 1 norm of a vector
5. [daxpy](#) : $\alpha * X + Y$
6. [dgemm](#) : $\alpha * \text{op}(A) \text{op}(B) + \beta * C$ where $\text{op}()$ is transpose
7. [dgetf2](#) : Compute LU Factorization of matrix A
8. [dpotf2](#) : DPOTF2 computes the Cholesky factorization of a real symmetric positive definite matrix A
9. [dscal](#) : Scale a vector alpha

10. [dsyrk](#) : Implements symmetric rank update C
11. [dtrsm](#) : Triangular matrix solver
12. [extract_diag](#) : Extract the diagonal of the matrix
13. [extract_tri](#) : Extract the upper or lower triangle
14. [make_diag](#) : Generate a diagonal matrix
15. [make_vector](#) : Make a vector of dimension m
16. [trace](#) : The trace of the input matrix

MODULE Types

[BLAS](#) \

	Types
--	-------

No Documentation Found

Children

1. [value_t](#) : No Documentation Found
2. [dimension_t](#) : No Documentation Found
3. [matrix_t](#) : No Documentation Found
4. [Triangle](#) : No Documentation Found
5. [Diagonal](#) : No Documentation Found
6. [Side](#) : No Documentation Found

ATTRIBUTE [value_t](#)

[BLAS](#) \ [Types](#) \

	value_t
--	-------------------------

No Documentation Found

RETURN REAL8 —

ATTRIBUTE dimension_t

[BLAS](#) \ [Types](#) \

	dimension_t
--	-------------

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE matrix_t

[BLAS](#) \ [Types](#) \

	matrix_t
--	----------

No Documentation Found

RETURN SET (REAL8) —

ATTRIBUTE Triangle

[BLAS](#) \ [Types](#) \

	Triangle
--	----------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Diagonal

[BLAS](#) \ [Types](#) \

	Diagonal
--	----------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Side

[BLAS](#) \ [Types](#) \

	Side
--	------

No Documentation Found

RETURN UNSIGNED1 —

FUNCTION ICellFunc

[BLAS](#) \

Types.value_t	ICellFunc
(Types.value_t v, Types.dimension_t r, Types.dimension_t c)	

Function prototype for Apply2Cell.

PARAMETER v ||| REAL8 — the value

PARAMETER c ||| UNSIGNED4 — the column ordinal

PARAMETER r ||| UNSIGNED4 — the row ordinal

RETURN REAL8 — the updated value

FUNCTION Apply2Cells

BLAS \

Types.matrix_t	Apply2Cells
(Types.dimension_t m, Types.dimension_t n, Types.matrix_t x, ICellFunc f)	

Iterate matrix and apply function to each cell

PARAMETER f ||| FUNCTION [REAL8 , UNSIGNED4 , UNSIGNED4] (REAL8) — function to apply

PARAMETER x ||| SET (REAL8) — matrix

PARAMETER n ||| UNSIGNED4 — number of columns

PARAMETER m ||| UNSIGNED4 — number of rows

RETURN SET (REAL8) — updated matrix

FUNCTION dasum

BLAS \

Types.value_t	dasum
(Types.dimension_t m, Types.matrix_t x, Types.dimension_t incx, Types.dimension_t skipped=0)	

Absolute sum, the 1 norm of a vector.

PARAMETER skipped ||| UNSIGNED4 — default is zero, the number of entries stepped over to get to the first entry

PARAMETER incx ||| UNSIGNED4 — the increment for x, 1 in the case of an actual vector

PARAMETER x ||| SET (REAL8) — the column major matrix holding the vector

PARAMETER m ||| UNSIGNED4 — the number of entries

RETURN REAL8 — the sum of the absolute values

FUNCTION daxpy

BLAS \

Types.matrix_t	daxpy
(Types.dimension_t N, Types.value_t alpha, Types.matrix_t X, Types.dimension_t incX, Types.matrix_t Y, Types.dimension_t incY, Types.dimension_t x_skipped=0, Types.dimension_t y_skipped=0)	

$\alpha * X + Y$

PARAMETER y_skipped ||| UNSIGNED4 — number of entries skipped to get to the first Y

PARAMETER X ||| SET (REAL8) — the column major matrix holding the vector X

PARAMETER incX ||| UNSIGNED4 — the increment or stride for the vector

PARAMETER N ||| UNSIGNED4 — number of elements in vector

PARAMETER alpha ||| REAL8 — the scalar multiplier

PARAMETER Y ||| SET (REAL8) — the column major matrix holding the vector Y

PARAMETER incY ||| UNSIGNED4 — the increment or stride of Y

PARAMETER x_skipped ||| UNSIGNED4 — number of entries skipped to get to the first X

RETURN SET (REAL8) — the updated matrix

FUNCTION dgemm

BLAS \

<code>Types.matrix_t</code>	<code>dgemm</code>
<pre>(BOOLEAN transposeA, BOOLEAN transposeB, Types.dimension_t M, Types.dimension_t N, Types.dimension_t K, Types.value_t alpha, Types.matrix_t A, Types.matrix_t B, Types.value_t beta=0.0, Types.matrix_t C=[])</pre>	

$\alpha * \text{op}(A) \text{op}(B) + \beta * C$ where $\text{op}()$ is transpose

PARAMETER B ||| SET (REAL8) — matrix B

PARAMETER A ||| SET (REAL8) — matrix A

PARAMETER transposeB ||| BOOLEAN — true when transpose of B is used

PARAMETER K ||| UNSIGNED4 — number of columns/rows for the multiplier/multiplicand

PARAMETER transposeA ||| BOOLEAN — true when transpose of A is used

PARAMETER alpha ||| REAL8 — scalar used on A

PARAMETER beta ||| REAL8 — scalar for matrix C

PARAMETER C ||| SET (REAL8) — matrix C or empty

PARAMETER M ||| UNSIGNED4 — number of rows in product

PARAMETER N ||| UNSIGNED4 — number of columns in product

RETURN SET (REAL8) —

FUNCTION dgetf2

BLAS \

<code>Types.matrix_t</code>	<code>dgetf2</code>
<pre>(Types.dimension_t m, Types.dimension_t n, Types.matrix_t a)</pre>	

Compute LU Factorization of matrix A.

PARAMETER n ||| UNSIGNED4 — number of columns of A

PARAMETER m ||| UNSIGNED4 — number of rows of A

PARAMETER a ||| SET (REAL8) — No Doc

RETURN SET (REAL8) — composite matrix of factors, lower triangle has an implied diagonal of ones. Upper triangle has the diagonal of the composite.

FUNCTION dpotf2

BLAS \

Types.matrix_t	dpotf2
(Types.Triangle tri, Types.dimension_t r, Types.matrix_t A, BOOLEAN clear=TRUE)	

DPOTF2 computes the Cholesky factorization of a real symmetric positive definite matrix A. The factorization has the form $A = U^{**T} * U$, if UPLO = 'U', or $A = L * L^{**T}$, if UPLO = 'L', where U is an upper triangular matrix and L is lower triangular. This is the unblocked version of the algorithm, calling Level 2 BLAS.

PARAMETER tri ||| UNSIGNED1 — indicate whether upper or lower triangle is used

PARAMETER A ||| SET (REAL8) — the square matrix

PARAMETER r ||| UNSIGNED4 — number of rows/columns in the square matrix

PARAMETER clear ||| BOOLEAN — clears the unused triangle

RETURN SET (REAL8) — the triangular matrix requested.

FUNCTION dscal

BLAS \

<code>Types.matrix_t</code>	<code>dscal</code>
(Types.dimension_t N, Types.value_t alpha, Types.matrix_t X, Types.dimension_t incX, Types.dimension_t skipped=0)	

Scale a vector alpha

PARAMETER alpha ||| REAL8 — the scaling factor

PARAMETER X ||| SET (REAL8) — the column major matrix holding the vector

PARAMETER incX ||| UNSIGNED4 — the stride to get to the next element in the vector

PARAMETER N ||| UNSIGNED4 — number of elements in the vector

PARAMETER skipped ||| UNSIGNED4 — the number of elements skipped to get to the first element

RETURN SET (REAL8) — the updated matrix

FUNCTION dsyrk

BLAS \

<code>Types.matrix_t</code>	<code>dsyrk</code>
(Types.Triangle tri, BOOLEAN transposeA, Types.dimension_t N, Types.dimension_t K, Types.value_t alpha, Types.matrix_t A, Types.value_t beta, Types.matrix_t C, BOOLEAN clear=FALSE)	

Implements symmetric rank update C

PARAMETER tri ||| UNSIGNED1 — update upper or lower triangle

PARAMETER A ||| SET (REAL8) — the update matrix, either NxK or KxN

PARAMETER K ||| UNSIGNED4 — number of columns in the update matrix or transpose

PARAMETER transposeA ||| BOOLEAN — Transpose the A matrix to be NxK

PARAMETER clear ||| BOOLEAN — clear the triangle that is not updated. BLAS assumes that symmetric matrices have only one of the triangles and this option lets you make that true.

PARAMETER alpha ||| REAL8 — the alpha scalar

PARAMETER beta ||| REAL8 — the beta scalar

PARAMETER C ||| SET (REAL8) — the matrix to update

PARAMETER N ||| UNSIGNED4 — number of rows

RETURN SET (REAL8) —

FUNCTION dtrsm

BLAS \

Types.matrix_t	dtrsm
<pre>(Types.Side side, Types.Triangle tri, BOOLEAN transposeA, Types.Diagonal diag, Types.dimension_t M, Types.dimension_t N, Types.dimension_t lda, Types.value_t alpha, Types.matrix_t A, Types.matrix_t B)</pre>	

Triangular matrix solver. $\text{op}(A) X = \alpha B$ or $X \text{op}(A) = \alpha B$ where op is Transpose, X and B is $M \times N$

PARAMETER B ||| SET (REAL8) — the matrix of values for the solve

PARAMETER tri ||| UNSIGNED1 — Says whether A is Upper or Lower triangle

PARAMETER A ||| SET (REAL8) — a triangular matrix

PARAMETER M ||| UNSIGNED4 — number of rows

PARAMETER transposeA ||| BOOLEAN — is $\text{op}(A)$ the transpose of A

PARAMETER side ||| UNSIGNED1 — side for A, Side.Ax is $\text{op}(A) X = \alpha B$

PARAMETER alpha ||| REAL8 — the scalar multiplier for B

PARAMETER lda ||| UNSIGNED4 — the leading dimension of the A matrix, either M or N

PARAMETER diag ||| UNSIGNED1 — is the diagonal an implied unit diagonal or supplied

PARAMETER N ||| UNSIGNED4 — number of columns

RETURN SET (REAL8) — the matrix of coefficients to get B.

FUNCTION `extract_diag`

BLAS \

<code>Types.matrix_t</code>	<code>extract_diag</code>
<code>(Types.dimension_t m, Types.dimension_t n, Types.matrix_t x)</code>	

Extract the diagonal of the matrix

PARAMETER `x` ||| SET (REAL8) — matrix from which to extract the diagonal

PARAMETER `n` ||| UNSIGNED4 — number of columns

PARAMETER `m` ||| UNSIGNED4 — number of rows

RETURN SET (REAL8) — diagonal matrix

FUNCTION `extract_tri`

BLAS \

<code>Types.matrix_t</code>	<code>extract_tri</code>
<code>(Types.dimension_t m, Types.dimension_t n, Types.Triangle tri, Types.Diagonal dt, Types.matrix_t a)</code>	

Extract the upper or lower triangle. Diagonal can be actual or implied unit diagonal.

PARAMETER `dt` ||| UNSIGNED1 — Use Diagonal.NotUnitTri or Diagonal.UnitTri

PARAMETER `tri` ||| UNSIGNED1 — Upper or Lower specifier, Triangle.Lower or Triangle.Upper

PARAMETER `a` ||| SET (REAL8) — Matrix, usually a composite from factoring

PARAMETER `n` ||| UNSIGNED4 — number of columns

PARAMETER `m` ||| UNSIGNED4 — number of rows

RETURN SET (REAL8) — the triangle

FUNCTION `make_diag`

BLAS \

<code>Types.matrix_t</code>	<code>make_diag</code>
<code>(Types.dimension_t m, Types.value_t v=1.0, Types.matrix_t X=[])</code>	

Generate a diagonal matrix.

PARAMETER X ||| SET (REAL8) — optional input of diagonal values, multiplied by v.

PARAMETER v ||| REAL8 — option value, defaults to 1

PARAMETER m ||| UNSIGNED4 — number of diagonal entries

RETURN SET (REAL8) — a diagonal matrix

FUNCTION `make_vector`

BLAS \

<code>Types.matrix_t</code>	<code>make_vector</code>
<code>(Types.dimension_t m, Types.value_t v=1.0)</code>	

Make a vector of dimension m

PARAMETER v ||| REAL8 — the values, defaults to 1

PARAMETER m ||| UNSIGNED4 — number of elements

RETURN SET (REAL8) — the vector

FUNCTION **trace**

BLAS \

<code>Types.value_t</code>	trace
<code>(Types.dimension_t m, Types.dimension_t n, Types.matrix_t x)</code>	

The trace of the input matrix

PARAMETER **x** ||| SET (REAL8) — the matrix

PARAMETER **n** ||| UNSIGNED4 — number of columns

PARAMETER **m** ||| UNSIGNED4 — number of rows

RETURN **REAL8** — the trace (sum of the diagonal entries)

BundleBase

[Go Up](#)

DESCRIPTIONS

MODULE BundleBase

	BundleBase
--	------------

No Documentation Found

Children

1. [PropertyRecord](#) : No Documentation Found
 2. [Name](#) : No Documentation Found
 3. [Description](#) : No Documentation Found
 4. [Authors](#) : No Documentation Found
 5. [License](#) : No Documentation Found
 6. [Copyright](#) : No Documentation Found
 7. [DependsOn](#) : No Documentation Found
 8. [Version](#) : No Documentation Found
 9. [Properties](#) : No Documentation Found
 10. [PlatformVersion](#) : No Documentation Found
-

RECORD PropertyRecord

[BundleBase](#) \

	PropertyRecord
--	-----------------------

No Documentation Found

FIELD key ||| UTF8 — No Doc

FIELD value ||| UTF8 — No Doc

ATTRIBUTE Name

[BundleBase](#) \

STRING	Name
---------------	-------------

No Documentation Found

RETURN STRING —

ATTRIBUTE Description

[BundleBase](#) \

UTF8	Description
-------------	--------------------

No Documentation Found

RETURN UTF8 —

ATTRIBUTE Authors

[BundleBase](#) \

SET OF UTF8	Authors
-------------	---------

No Documentation Found

RETURN SET (UTF8) —

ATTRIBUTE License

[BundleBase](#) \

UTF8	License
------	---------

No Documentation Found

RETURN UTF8 —

ATTRIBUTE Copyright

[BundleBase](#) \

UTF8	Copyright
------	-----------

No Documentation Found

RETURN UTF8 —

ATTRIBUTE DependsOn

[BundleBase](#) \

SET OF STRING	DependsOn
---------------	-----------

No Documentation Found

RETURN SET (STRING) —

ATTRIBUTE Version

[BundleBase](#) \

STRING	Version
--------	---------

No Documentation Found

RETURN STRING —

ATTRIBUTE Properties

[BundleBase](#) \

Properties

No Documentation Found

RETURN DICTIONARY (PropertyRecord) —

ATTRIBUTE PlatformVersion

BundleBase \

STRING	PlatformVersion
--------	-----------------

No Documentation Found

RETURN STRING —

Date

[Go Up](#)

IMPORTS

DESCRIPTIONS

MODULE Date

	Date
--	------

No Documentation Found

Children

1. [Date_rec](#) : No Documentation Found
2. [Date_t](#) : No Documentation Found
3. [Days_t](#) : No Documentation Found
4. [Time_rec](#) : No Documentation Found
5. [Time_t](#) : No Documentation Found
6. [Seconds_t](#) : No Documentation Found
7. [DateTime_rec](#) : No Documentation Found
8. [Timestamp_t](#) : No Documentation Found
9. [Year](#) : Extracts the year from a date type
10. [Month](#) : Extracts the month from a date type
11. [Day](#) : Extracts the day of the month from a date type
12. [Hour](#) : Extracts the hour from a time type
13. [Minute](#) : Extracts the minutes from a time type

14. [Second](#) : Extracts the seconds from a time type
15. [DateFromParts](#) : Combines year, month day to create a date type
16. [TimeFromParts](#) : Combines hour, minute second to create a time type
17. [SecondsFromParts](#) : Combines date and time components to create a seconds type
18. [SecondsToParts](#) : Converts the number of seconds since epoch to a structure containing date and time parts
19. [TimestampToSeconds](#) : Converts the number of microseconds since epoch to the number of seconds since epoch
20. [IsLeapYear](#) : Tests whether the year is a leap year in the Gregorian calendar
21. [IsDateLeapYear](#) : Tests whether a date is a leap year in the Gregorian calendar
22. [FromGregorianYMD](#) : Combines year, month, day in the Gregorian calendar to create the number days since 31st December 1BC
23. [ToGregorianYMD](#) : Converts the number days since 31st December 1BC to a date in the Gregorian calendar
24. [FromGregorianDate](#) : Converts a date in the Gregorian calendar to the number days since 31st December 1BC
25. [ToGregorianDate](#) : Converts the number days since 31st December 1BC to a date in the Gregorian calendar
26. [DayOfYear](#) : Returns a number representing the day of the year indicated by the given date
27. [DayOfWeek](#) : Returns a number representing the day of the week indicated by the given date
28. [IsJulianLeapYear](#) : Tests whether the year is a leap year in the Julian calendar
29. [FromJulianYMD](#) : Combines year, month, day in the Julian calendar to create the number days since 31st December 1BC
30. [ToJulianYMD](#) : Converts the number days since 31st December 1BC to a date in the Julian calendar
31. [FromJulianDate](#) : Converts a date in the Julian calendar to the number days since 31st December 1BC
32. [ToJulianDate](#) : Converts the number days since 31st December 1BC to a date in the Julian calendar
33. [DaysSince1900](#) : Returns the number of days since 1st January 1900 (using the Gregorian Calendar)
34. [ToDaysSince1900](#) : Returns the number of days since 1st January 1900 (using the Gregorian Calendar)
35. [FromDaysSince1900](#) : Converts the number days since 1st January 1900 to a date in the Julian calendar
36. [YearsBetween](#) : Calculate the number of whole years between two dates

37. [MonthsBetween](#) : Calculate the number of whole months between two dates
38. [DaysBetween](#) : Calculate the number of days between two dates
39. [DateFromDateRec](#) : Combines the fields from a `Date_rec` to create a `Date_t`
40. [DateFromRec](#) : Combines the fields from a `Date_rec` to create a `Date_t`
41. [TimeFromTimeRec](#) : Combines the fields from a `Time_rec` to create a `Time_t`
42. [DateFromDateTimeRec](#) : Combines the date fields from a `DateTime_rec` to create a `Date_t`
43. [TimeFromDateTimeRec](#) : Combines the time fields from a `DateTime_rec` to create a `Time_t`
44. [SecondsFromDateTimeRec](#) : Combines the date and time fields from a `DateTime_rec` to create a `Seconds_t`
45. [FromStringToDate](#) : Converts a string to a `Date_t` using the relevant string format
46. [FromString](#) : Converts a string to a date using the relevant string format
47. [FromStringToTime](#) : Converts a string to a `Time_t` using the relevant string format
48. [MatchDateString](#) : Matches a string against a set of date string formats and returns a valid `Date_t` object from the first format that successfully parses the string
49. [MatchTimeString](#) : Matches a string against a set of time string formats and returns a valid `Time_t` object from the first format that successfully parses the string
50. [DateToString](#) : Formats a date as a string
51. [TimeToString](#) : Formats a time as a string
52. [SecondsToString](#) : Converts a `Seconds_t` value into a human-readable string using a format template
53. [ToString](#) : Formats a date as a string
54. [ConvertDateFormat](#) : Converts a date from one format to another
55. [ConvertFormat](#) : Converts a date from one format to another
56. [ConvertTimeFormat](#) : Converts a time from one format to another
57. [ConvertDateFormatMultiple](#) : Converts a date that matches one of a set of formats to another
58. [ConvertFormatMultiple](#) : Converts a date that matches one of a set of formats to another
59. [ConvertTimeFormatMultiple](#) : Converts a time that matches one of a set of formats to another
60. [AdjustDate](#) : Adjusts a date by incrementing or decrementing year, month and/or day values
61. [AdjustDateBySeconds](#) : Adjusts a date by adding or subtracting seconds
62. [AdjustTime](#) : Adjusts a time by incrementing or decrementing hour, minute and/or second values
63. [AdjustTimeBySeconds](#) : Adjusts a time by adding or subtracting seconds

- 64. [AdjustSeconds](#) : Adjusts a Seconds_t value by adding or subtracting years, months, days, hours, minutes and/or seconds
 - 65. [AdjustCalendar](#) : Adjusts a date by incrementing or decrementing months and/or years
 - 66. [IsLocalDaylightSavingsInEffect](#) : Returns a boolean indicating whether daylight savings time is currently in effect locally
 - 67. [LocalTimeZoneOffset](#) : Returns the offset (in seconds) of the time represented from UTC, with positive values indicating locations east of the Prime Meridian
 - 68. [CurrentDate](#) : Returns the current date
 - 69. [Today](#) : Returns the current date in the local time zone
 - 70. [CurrentTime](#) : Returns the current time of day
 - 71. [CurrentSeconds](#) : Returns the current date and time as the number of seconds since epoch
 - 72. [CurrentTimestamp](#) : Returns the current date and time as the number of microseconds since epoch
 - 73. [DatesForMonth](#) : Returns the beginning and ending dates for the month surrounding the given date
 - 74. [DatesForWeek](#) : Returns the beginning and ending dates for the week surrounding the given date (Sunday marks the beginning of a week)
 - 75. [IsValidDate](#) : Tests whether a date is valid, both by range-checking the year and by validating each of the other individual components
 - 76. [IsValidGregorianCalendar](#) : Tests whether a date is valid in the Gregorian calendar
 - 77. [IsValidTime](#) : Tests whether a time is valid
 - 78. [CreateDate](#) : A transform to create a Date_rec from the individual elements
 - 79. [CreateDateFromSeconds](#) : A transform to create a Date_rec from a Seconds_t value
 - 80. [CreateTime](#) : A transform to create a Time_rec from the individual elements
 - 81. [CreateTimeFromSeconds](#) : A transform to create a Time_rec from a Seconds_t value
 - 82. [CreateDateTime](#) : A transform to create a DateTime_rec from the individual elements
 - 83. [CreateDateTimeFromSeconds](#) : A transform to create a DateTime_rec from a Seconds_t value
-

RECORD Date_rec

Date \

Date_rec

No Documentation Found

FIELD year ||| INTEGER2 — No Doc

FIELD month ||| UNSIGNED1 — No Doc

FIELD day ||| UNSIGNED1 — No Doc

ATTRIBUTE Date_t

Date \

Date_t

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE Days_t

Date \

Days_t

No Documentation Found

RETURN INTEGER4 —

RECORD Time_rec

Date \

	Time_rec
--	----------

No Documentation Found

FIELD minute ||| UNSIGNED1 — No Doc

FIELD second ||| UNSIGNED1 — No Doc

FIELD hour ||| UNSIGNED1 — No Doc

ATTRIBUTE Time_t

Date \

	Time_t
--	--------

No Documentation Found

RETURN UNSIGNED3 —

ATTRIBUTE Seconds_t

Date \

	Seconds_t
--	-----------

No Documentation Found

RETURN INTEGER8 —

RECORD DateTime_rec

Date \

	DateTime_rec
--	--------------

No Documentation Found

FIELD year ||| INTEGER2 — No Doc

FIELD second ||| UNSIGNED1 — No Doc

FIELD hour ||| UNSIGNED1 — No Doc

FIELD minute ||| UNSIGNED1 — No Doc

FIELD month ||| UNSIGNED1 — No Doc

FIELD day ||| UNSIGNED1 — No Doc

ATTRIBUTE Timestamp_t

Date \

	Timestamp_t
--	-------------

No Documentation Found

RETURN INTEGER8 —

FUNCTION Year

Date \

INTEGER2	Year
(Date_t date)	

Extracts the year from a date type.

PARAMETER date ||| UNSIGNED4 — The date.

RETURN INTEGER2 — An integer representing the year.

FUNCTION Month

Date \

UNSIGNED1	Month
(Date_t date)	

Extracts the month from a date type.

PARAMETER date ||| UNSIGNED4 — The date.

RETURN UNSIGNED1 — An integer representing the year.

FUNCTION Day

Date \

UNSIGNED1	Day
(Date_t date)	

Extracts the day of the month from a date type.

PARAMETER date ||| UNSIGNED4 — The date.

RETURN UNSIGNED1 — An integer representing the year.

FUNCTION Hour

Date \

UNSIGNED1	Hour
(Time_t time)	

Extracts the hour from a time type.

PARAMETER time ||| UNSIGNED3 — The time.

RETURN UNSIGNED1 — An integer representing the hour.

FUNCTION Minute

Date \

UNSIGNED1	Minute
(Time_t time)	

Extracts the minutes from a time type.

PARAMETER time ||| UNSIGNED3 — The time.

RETURN UNSIGNED1 — An integer representing the minutes.

FUNCTION Second

Date \

UNSIGNED1	Second
(Time_t time)	

Extracts the seconds from a time type.

PARAMETER time ||| UNSIGNED3 — The time.

RETURN UNSIGNED1 — An integer representing the seconds.

FUNCTION DateFromParts

Date \

Date_t	DateFromParts
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day)	

Combines year, month day to create a date type.

PARAMETER year ||| INTEGER2 — The year (0-9999).

PARAMETER month ||| UNSIGNED1 — The month (1-12).

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth).

RETURN UNSIGNED4 — A date created by combining the fields.

FUNCTION TimeFromParts

Date \

Time_t	TimeFromParts
(UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second)	

Combines hour, minute second to create a time type.

PARAMETER minute ||| UNSIGNED1 — The minute (0-59).

PARAMETER second ||| UNSIGNED1 — The second (0-59).

PARAMETER hour ||| UNSIGNED1 — The hour (0-23).

RETURN **UNSIGNED3** — A time created by combining the fields.

FUNCTION SecondsFromParts

Date \

Seconds_t	SecondsFromParts
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day, UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second, BOOLEAN is_local_time = FALSE)	

Combines date and time components to create a seconds type. The date must be represented within the Gregorian calendar after the year 1600.

PARAMETER year ||| INTEGER2 — The year (1601-30827).

PARAMETER second ||| UNSIGNED1 — The second (0-59).

PARAMETER hour ||| UNSIGNED1 — The hour (0-23).

PARAMETER minute ||| UNSIGNED1 — The minute (0-59).

PARAMETER month ||| UNSIGNED1 — The month (1-12).

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth).

PARAMETER is_local_time ||| BOOLEAN — TRUE if the datetime components are expressed in local time rather than UTC, FALSE if the components are expressed in UTC. Optional, defaults to FALSE.

RETURN **INTEGER8** — A Seconds_t value created by combining the fields.

MODULE SecondsToParts

Date \

	SecondsToParts
(Seconds_t seconds)	

Converts the number of seconds since epoch to a structure containing date and time parts. The result must be representable within the Gregorian calendar after the year 1600.

PARAMETER seconds ||| INTEGER8 — The number of seconds since epoch.

RETURN — Module with exported attributes for year, month, day, hour, minute, second, day_of_week, date and time.

Children

1. [Year](#) : No Documentation Found
2. [Month](#) : No Documentation Found
3. [Day](#) : No Documentation Found
4. [Hour](#) : No Documentation Found
5. [Minute](#) : No Documentation Found
6. [Second](#) : No Documentation Found
7. [day_of_week](#) : No Documentation Found
8. [date](#) : Combines year, month day to create a date type
9. [time](#) : Combines hour, minute second to create a time type

ATTRIBUTE Year

[Date](#) \ [SecondsToParts](#) \

INTEGER2	Year
-----------------	-------------

No Documentation Found

RETURN INTEGER2 —

ATTRIBUTE Month

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	Month
-----------	-------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Day

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	Day
-----------	-----

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Hour

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	Hour
-----------	------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Minute

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	Minute
-----------	--------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Second

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	Second
-----------	--------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE day_of_week

[Date](#) \ [SecondsToParts](#) \

UNSIGNED1	day_of_week
-----------	-------------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE **date**

[Date](#) \ [SecondsToParts](#) \

Date_t	date
---------------	-------------

Combines year, month day to create a date type.

PARAMETER year ||| — The year (0-9999).

PARAMETER month ||| — The month (1-12).

PARAMETER day ||| — The day (1..daysInMonth).

RETURN **UNSIGNED4** — A date created by combining the fields.

ATTRIBUTE **time**

[Date](#) \ [SecondsToParts](#) \

Time_t	time
---------------	-------------

Combines hour, minute second to create a time type.

PARAMETER minute ||| — The minute (0-59).

PARAMETER second ||| — The second (0-59).

PARAMETER hour ||| — The hour (0-23).

RETURN **UNSIGNED3** — A time created by combining the fields.

FUNCTION TimestampToSeconds

Date \

Seconds_t	TimestampToSeconds
(Timestamp_t timestamp)	

Converts the number of microseconds since epoch to the number of seconds since epoch.

PARAMETER timestamp ||| INTEGER8 — The number of microseconds since epoch.

RETURN INTEGER8 — The number of seconds since epoch.

FUNCTION IsLeapYear

Date \

BOOLEAN	IsLeapYear
(INTEGER2 year)	

Tests whether the year is a leap year in the Gregorian calendar.

PARAMETER year ||| INTEGER2 — The year (0-9999).

RETURN BOOLEAN — True if the year is a leap year.

FUNCTION IsDateLeapYear

Date \

BOOLEAN	IsDateLeapYear
(Date_t date)	

Tests whether a date is a leap year in the Gregorian calendar.

PARAMETER date ||| UNSIGNED4 — The date.

RETURN **BOOLEAN** — True if the year is a leap year.

FUNCTION FromGregorianYMD

Date \

Days_t	FromGregorianYMD
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day)	

Combines year, month, day in the Gregorian calendar to create the number days since 31st December 1BC.

PARAMETER year ||| INTEGER2 — The year (-4713..9999).

PARAMETER month ||| UNSIGNED1 — The month (1-12). A missing value (0) is treated as 1.

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth). A missing value (0) is treated as 1.

RETURN **INTEGER4** — The number of elapsed days (1 Jan 1AD = 1)

MODULE ToGregorianYMD

Date \

	ToGregorianYMD
(Days_t days)	

Converts the number days since 31st December 1BC to a date in the Gregorian calendar.

PARAMETER days ||| INTEGER4 — The number of elapsed days (1 Jan 1AD = 1)

RETURN — Module containing Year, Month, Day in the Gregorian calendar

Children

1. [year](#) : No Documentation Found
 2. [month](#) : No Documentation Found
 3. [day](#) : No Documentation Found
-

ATTRIBUTE year

[Date](#) \ [ToGregorianYMD](#) \

	year
--	------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE month

[Date](#) \ [ToGregorianYMD](#) \

	month
--	-------

No Documentation Found

RETURN INTEGER8 —

ATTRIBUTE day

[Date](#) \ [ToGregorianYMD](#) \

	day
--	------------

No Documentation Found

RETURN **INTEGER8** —

FUNCTION **FromGregorianDate**

Date \

Days_t	FromGregorianDate
(Date_t date)	

Converts a date in the Gregorian calendar to the number days since 31st December 1BC.

PARAMETER date ||| **UNSIGNED4** — The date (using the Gregorian calendar)

RETURN **INTEGER4** — The number of elapsed days (1 Jan 1AD = 1)

FUNCTION **ToGregorianDate**

Date \

Date_t	ToGregorianDate
(Days_t days)	

Converts the number days since 31st December 1BC to a date in the Gregorian calendar.

PARAMETER days ||| **INTEGER4** — The number of elapsed days (1 Jan 1AD = 1)

RETURN **UNSIGNED4** — A Date_t in the Gregorian calendar

FUNCTION DayOfYear

Date \

UNSIGNED2	DayOfYear
(Date_t date)	

Returns a number representing the day of the year indicated by the given date. The date must be in the Gregorian calendar after the year 1600.

PARAMETER date ||| UNSIGNED4 — A Date_t value.

RETURN UNSIGNED2 — A number (1-366) representing the number of days since the beginning of the year.

FUNCTION DayOfWeek

Date \

UNSIGNED1	DayOfWeek
(Date_t date)	

Returns a number representing the day of the week indicated by the given date. The date must be in the Gregorian calendar after the year 1600.

PARAMETER date ||| UNSIGNED4 — A Date_t value.

RETURN UNSIGNED1 — A number 1-7 representing the day of the week, where 1 = Sunday.

FUNCTION IsJulianLeapYear

Date \

BOOLEAN	IsJulianLeapYear
(INTEGER2 <i>year</i>)	

Tests whether the year is a leap year in the Julian calendar.

PARAMETER year ||| INTEGER2 — The year (0-9999).

RETURN **BOOLEAN** — True if the year is a leap year.

FUNCTION FromJulianYMD

Date \

Days_t	FromJulianYMD
(INTEGER2 <i>year</i> , UNSIGNED1 <i>month</i> , UNSIGNED1 <i>day</i>)	

Combines year, month, day in the Julian calendar to create the number days since 31st December 1BC.

PARAMETER year ||| INTEGER2 — The year (-4800..9999).

PARAMETER month ||| UNSIGNED1 — The month (1-12).

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth).

RETURN **INTEGER4** — The number of elapsed days (1 Jan 1AD = 1)

MODULE ToJulianYMD

Date \

	ToJulianYMD
(Days_t <i>days</i>)	

Converts the number days since 31st December 1BC to a date in the Julian calendar.

PARAMETER days ||| INTEGER4 — The number of elapsed days (1 Jan 1AD = 1)

RETURN — Module containing Year, Month, Day in the Julian calendar

Children

1. [Day](#) : No Documentation Found
2. [Month](#) : No Documentation Found
3. [Year](#) : No Documentation Found

ATTRIBUTE Day

[Date](#) \ [ToJulianYMD](#) \

UNSIGNED1	Day
------------------	-----

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Month

[Date](#) \ [ToJulianYMD](#) \

UNSIGNED1	Month
------------------	-------

No Documentation Found

RETURN UNSIGNED1 —

ATTRIBUTE Year

Date \ ToJulianYMD \

INTEGER2	Year
----------	------

No Documentation Found

RETURN INTEGER2 —

FUNCTION FromJulianDate

Date \

Days_t	FromJulianDate
(Date_t date)	

Converts a date in the Julian calendar to the number days since 31st December 1BC.

PARAMETER date ||| UNSIGNED4 — The date (using the Julian calendar)

RETURN INTEGER4 — The number of elapsed days (1 Jan 1AD = 1)

FUNCTION ToJulianDate

Date \

Date_t	ToJulianDate
(Days_t days)	

Converts the number days since 31st December 1BC to a date in the Julian calendar.

PARAMETER days ||| INTEGER4 — The number of elapsed days (1 Jan 1AD = 1)

RETURN UNSIGNED4 — A Date_t in the Julian calendar

FUNCTION DaysSince1900

Date \

Days_t	DaysSince1900
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day)	

Returns the number of days since 1st January 1900 (using the Gregorian Calendar)

PARAMETER year ||| INTEGER2 — The year (-4713..9999).

PARAMETER month ||| UNSIGNED1 — The month (1-12). A missing value (0) is treated as 1.

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth). A missing value (0) is treated as 1.

RETURN INTEGER4 — The number of elapsed days since 1st January 1900

FUNCTION ToDaysSince1900

Date \

Days_t	ToDaysSince1900
(Date_t date)	

Returns the number of days since 1st January 1900 (using the Gregorian Calendar)

PARAMETER date ||| UNSIGNED4 — The date

RETURN INTEGER4 — The number of elapsed days since 1st January 1900

FUNCTION FromDaysSince1900

Date \

Date_t	FromDaysSince1900
(Days_t days)	

Converts the number days since 1st January 1900 to a date in the Julian calendar.

PARAMETER days ||| INTEGER4 — The number of elapsed days since 1st Jan 1900

RETURN UNSIGNED4 — A Date_t in the Julian calendar

FUNCTION YearsBetween

Date \

INTEGER	YearsBetween
(Date_t from, Date_t to)	

Calculate the number of whole years between two dates.

PARAMETER from ||| UNSIGNED4 — The first date

PARAMETER to ||| UNSIGNED4 — The last date

RETURN INTEGER8 — The number of years between them.

FUNCTION MonthsBetween

Date \

INTEGER	MonthsBetween
(Date_t from, Date_t to)	

Calculate the number of whole months between two dates.

PARAMETER from ||| UNSIGNED4 — The first date

PARAMETER to ||| UNSIGNED4 — The last date

RETURN INTEGER8 — The number of months between them.

FUNCTION DaysBetween

Date \

INTEGER	DaysBetween
(Date_t from, Date_t to)	

Calculate the number of days between two dates.

PARAMETER from ||| UNSIGNED4 — The first date

PARAMETER to ||| UNSIGNED4 — The last date

RETURN INTEGER8 — The number of days between them.

FUNCTION DateFromDateRec

Date \

Date_t	DateFromDateRec
(Date_rec date)	

Combines the fields from a Date_rec to create a Date_t

PARAMETER date ||| ROW (Date_rec) — The row containing the date.

RETURN UNSIGNED4 — A Date_t representing the combined values.

FUNCTION DateFromRec

Date \

Date_t	DateFromRec
(Date_rec date)	

Combines the fields from a Date_rec to create a Date_t

PARAMETER date ||| ROW (Date_rec) — The row containing the date.

RETURN UNSIGNED4 — A Date_t representing the combined values.

FUNCTION TimeFromTimeRec

Date \

Time_t	TimeFromTimeRec
(Time_rec time)	

Combines the fields from a Time_rec to create a Time_t

PARAMETER time ||| ROW (Time_rec) — The row containing the time.

RETURN UNSIGNED3 — A Time_t representing the combined values.

FUNCTION DateFromDateTimeRec

Date \

Date_t	DateFromDateTimeRec
(DateTime_rec datetime)	

Combines the date fields from a DateTime_rec to create a Date_t

PARAMETER datetime ||| ROW (DateTime_rec) — The row containing the datetime.

RETURN UNSIGNED4 — A Date_t representing the combined values.

FUNCTION TimeFromDateTimeRec

Date \

Time_t	TimeFromDateTimeRec
(DateTime_rec datetime)	

Combines the time fields from a DateTime_rec to create a Time_t

PARAMETER datetime ||| ROW (DateTime_rec) — The row containing the datetime.

RETURN UNSIGNED3 — A Time_t representing the combined values.

FUNCTION SecondsFromDateTimeRec

Date \

Seconds_t	SecondsFromDateTimeRec
(DateTime_rec datetime, BOOLEAN is_local_time = FALSE)	

Combines the date and time fields from a DateTime_rec to create a Seconds_t

PARAMETER datetime ||| ROW (DateTime_rec) — The row containing the datetime.

PARAMETER is_local_time ||| BOOLEAN — TRUE if the datetime components are expressed in local time rather than UTC, FALSE if the components are expressed in UTC. Optional, defaults to FALSE.

RETURN INTEGER8 — A Seconds_t representing the combined values.

FUNCTION FromStringToDate

Date \

Date_t	FromStringToDate
(STRING date_text, VARSTRING format)	

Converts a string to a Date_t using the relevant string format. The resulting date must be representable within the Gregorian calendar after the year 1600.

PARAMETER date_text ||| STRING — The string to be converted.

PARAMETER format ||| VARSTRING — The format of the input string. (See documentation for strftime)

RETURN UNSIGNED4 — The date that was matched in the string. Returns 0 if failed to match or if the date components match but the result is an invalid date. Supported characters: %B Full month name %b or %h Abbreviated month name %d Day of month (two digits) %e Day of month (two digits, or a space followed by a single digit) %m Month (two digits) %t Whitespace %y year within century (00-99) %Y Full year (yyyy) %j Julian day (1-366) Common date formats American '%m/%d/%Y' mm/dd/yyyy Euro '%d/%m/%Y' dd/mm/yyyy Iso format '%Y-%m-%d' yyyy-mm-dd Iso basic 'Y%m%d' yyyymmdd '%d-%b-%Y' dd-mon-yyyy e.g., '21-Mar-1954'

FUNCTION FromString

Date \

Date_t	FromString
(STRING date_text, VARSTRING format)	

Converts a string to a date using the relevant string format.

PARAMETER date_text ||| STRING — The string to be converted.

PARAMETER format ||| VARSTRING — The format of the input string. (See documentation for strftime)

RETURN UNSIGNED4 — The date that was matched in the string. Returns 0 if failed to match.

FUNCTION FromStringToTime

Date \

Time_t	FromStringToTime
(STRING time_text, VARSTRING format)	

Converts a string to a Time_t using the relevant string format.

PARAMETER date_text ||| — The string to be converted.

PARAMETER format ||| VARSTRING — The format of the input string. (See documentation for strftime)

PARAMETER time_text ||| STRING — No Doc

RETURN UNSIGNED3 — The time that was matched in the string. Returns 0 if failed to match.
Supported characters: %H Hour (two digits) %k (two digits, or a space followed by a single digit)
%M Minute (two digits) %S Second (two digits) %t Whitespace

FUNCTION MatchDateString

Date \

Date_t	MatchDateString
(STRING date_text, SET OF VARSTRING formats)	

Matches a string against a set of date string formats and returns a valid Date_t object from the first format that successfully parses the string.

PARAMETER formats ||| SET (VARSTRING) — A set of formats to check against the string. (See documentation for strftime)

PARAMETER date_text ||| STRING — The string to be converted.

RETURN UNSIGNED4 — The date that was matched in the string. Returns 0 if failed to match.

FUNCTION MatchTimeString

Date \

Time_t	MatchTimeString
(STRING time_text, SET OF VARSTRING formats)	

Matches a string against a set of time string formats and returns a valid Time_t object from the first format that successfully parses the string.

PARAMETER formats ||| SET (VARSTRING) — A set of formats to check against the string. (See documentation for strftime)

PARAMETER time_text ||| STRING — The string to be converted.

RETURN UNSIGNED3 — The time that was matched in the string. Returns 0 if failed to match.

FUNCTION DateToString

Date \

STRING	DateToString
(Date_t date, VARSTRING format = '%Y-%m-%d')	

Formats a date as a string.

PARAMETER date ||| UNSIGNED4 — The date to be converted.

PARAMETER format ||| VARSTRING — The format template to use for the conversion; see strftime() for appropriate values. The maximum length of the resulting string is 255 characters. Optional; defaults to '%Y-%m-%d' which is YYYY-MM-DD.

RETURN STRING — Blank if date cannot be formatted, or the date in the requested format.

FUNCTION TimeToString

Date \

STRING	TimeToString
(Time_t time, VARSTRING format = '%H:%M:%S')	

Formats a time as a string.

PARAMETER time ||| UNSIGNED3 — The time to be converted.

PARAMETER format ||| VARSTRING — The format template to use for the conversion; see strftime() for appropriate values. The maximum length of the resulting string is 255 characters. Optional; defaults to '%H:%M:%S' which is HH:MM:SS.

RETURN STRING — Blank if the time cannot be formatted, or the time in the requested format.

FUNCTION SecondsToString

Date \

STRING	SecondsToString
(Seconds_t seconds, VARSTRING format = '%Y-%m-%dT%H:%M:%S')	

Converts a Seconds_t value into a human-readable string using a format template.

PARAMETER format ||| VARSTRING — The format template to use for the conversion; see strftime() for appropriate values. The maximum length of the resulting string is 255 characters. Optional; defaults to '%Y-%m-%dT%H:%M:%S' which is YYYY-MM-DDTHH:MM:SS.

PARAMETER seconds ||| INTEGER8 — The seconds since epoch.

RETURN STRING — The converted seconds as a string.

FUNCTION ToString

Date \

STRING	ToString
(Date_t date, VARSTRING format)	

Formats a date as a string.

PARAMETER date ||| UNSIGNED4 — The date to be converted.

PARAMETER format ||| VARSTRING — The format the date is output in. (See documentation for strftime)

RETURN STRING — Blank if date cannot be formatted, or the date in the requested format.

FUNCTION ConvertDateFormat

Date \

STRING	ConvertDateFormat
(STRING date_text, VARSTRING from_format='%m/%d/%Y', VARSTRING to_format='%Y%m%d')	

Converts a date from one format to another

PARAMETER from_format ||| VARSTRING — The format the date is to be converted from.

PARAMETER date_text ||| STRING — The string containing the date to be converted.

PARAMETER to_format ||| VARSTRING — The format the date is to be converted to.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION ConvertFormat

Date \

STRING	ConvertFormat
(STRING date_text, VARSTRING from_format='%m/%d/%Y', VARSTRING to_format='%Y%m%d')	

Converts a date from one format to another

PARAMETER from_format ||| VARSTRING — The format the date is to be converted from.

PARAMETER date_text ||| STRING — The string containing the date to be converted.

PARAMETER to_format ||| VARSTRING — The format the date is to be converted to.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION ConvertTimeFormat

Date \

STRING	ConvertTimeFormat
(STRING time_text, VARSTRING from_format='%H%M%S', VARSTRING to_format='%H:%M:%S')	

Converts a time from one format to another

PARAMETER from_format ||| VARSTRING — The format the time is to be converted from.

PARAMETER to_format ||| VARSTRING — The format the time is to be converted to.

PARAMETER time_text ||| STRING — The string containing the time to be converted.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION ConvertDateFormatMultiple

Date \

STRING	ConvertDateFormatMultiple
(STRING date_text, SET OF VARSTRING from_formats, VARSTRING to_format='%Y%m%d')	

Converts a date that matches one of a set of formats to another.

PARAMETER from_formats ||| SET (VARSTRING) — The list of formats the date is to be converted from.

PARAMETER date_text ||| STRING — The string containing the date to be converted.

PARAMETER to_format ||| VARSTRING — The format the date is to be converted to.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION ConvertFormatMultiple

Date \

STRING	ConvertFormatMultiple
(STRING date_text, SET OF VARSTRING from_formats, VARSTRING to_format='%Y%m%d')	

Converts a date that matches one of a set of formats to another.

PARAMETER from_formats ||| SET (VARSTRING) — The list of formats the date is to be converted from.

PARAMETER date_text ||| STRING — The string containing the date to be converted.

PARAMETER to_format ||| VARSTRING — The format the date is to be converted to.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION ConvertTimeFormatMultiple

Date \

STRING	ConvertTimeFormatMultiple
(STRING time_text, SET OF VARSTRING from_formats, VARSTRING to_format='%H:%m:%s')	

Converts a time that matches one of a set of formats to another.

PARAMETER from_formats ||| SET (VARSTRING) — The list of formats the time is to be converted from.

PARAMETER to_format ||| VARSTRING — The format the time is to be converted to.

PARAMETER time_text ||| STRING — The string containing the time to be converted.

RETURN STRING — The converted string, or blank if it failed to match the format.

FUNCTION AdjustDate

Date \

Date_t	AdjustDate
(Date_t date, INTEGER2 year_delta = 0, INTEGER4 month_delta = 0, INTEGER4 day_delta = 0)	

Adjusts a date by incrementing or decrementing year, month and/or day values. The date must be in the Gregorian calendar after the year 1600. If the new calculated date is invalid then it will be normalized according to mktime() rules. Example: 20140130 + 1 month = 20140302.

PARAMETER date ||| UNSIGNED4 — The date to adjust.

PARAMETER month_delta ||| INTEGER4 — The requested change to the month value; optional, defaults to zero.

PARAMETER year_delta ||| INTEGER2 — The requested change to the year value; optional, defaults to zero.

PARAMETER day_delta ||| INTEGER4 — The requested change to the day of month value; optional, defaults to zero.

RETURN **UNSIGNED4** — The adjusted Date_t value.

FUNCTION AdjustDateBySeconds

Date \

Date_t	AdjustDateBySeconds
(Date_t date, INTEGER4 seconds_delta)	

Adjusts a date by adding or subtracting seconds. The date must be in the Gregorian calendar after the year 1600. If the new calculated date is invalid then it will be normalized according to mktime() rules. Example: 20140130 + 172800 seconds = 20140201.

PARAMETER date ||| **UNSIGNED4** — The date to adjust.

PARAMETER seconds_delta ||| **INTEGER4** — The requested change to the date, in seconds.

RETURN **UNSIGNED4** — The adjusted Date_t value.

FUNCTION AdjustTime

Date \

Time_t	AdjustTime
(Time_t time, INTEGER2 hour_delta = 0, INTEGER4 minute_delta = 0, INTEGER4 second_delta = 0)	

Adjusts a time by incrementing or decrementing hour, minute and/or second values. If the new calculated time is invalid then it will be normalized according to mktime() rules.

PARAMETER time ||| **UNSIGNED3** — The time to adjust.

PARAMETER second_delta ||| **INTEGER4** — The requested change to the second of month value; optional, defaults to zero.

PARAMETER hour_delta ||| INTEGER2 — The requested change to the hour value; optional, defaults to zero.

PARAMETER minute_delta ||| INTEGER4 — The requested change to the minute value; optional, defaults to zero.

RETURN UNSIGNED3 — The adjusted Time_t value.

FUNCTION AdjustTimeBySeconds

Date \

Time_t	AdjustTimeBySeconds
(Time_t time, INTEGER4 seconds_delta)	

Adjusts a time by adding or subtracting seconds. If the new calculated time is invalid then it will be normalized according to mktime() rules.

PARAMETER time ||| UNSIGNED3 — The time to adjust.

PARAMETER seconds_delta ||| INTEGER4 — The requested change to the time, in seconds.

RETURN UNSIGNED3 — The adjusted Time_t value.

FUNCTION AdjustSeconds

Date \

Seconds_t	AdjustSeconds
(Seconds_t seconds, INTEGER2 year_delta = 0, INTEGER4 month_delta = 0, INTEGER4 day_delta = 0, INTEGER4 hour_delta = 0, INTEGER4 minute_delta = 0, INTEGER4 second_delta = 0)	

Adjusts a Seconds_t value by adding or subtracting years, months, days, hours, minutes and/or seconds. This is performed by first converting the seconds into a full date/time structure, applying any delta

values to individual date/time components, then converting the structure back to the number of seconds. This interim date must lie within Gregorian calendar after the year 1600. If the interim structure is found to have an invalid date/time then it will be normalized according to mktime() rules. Therefore, some delta values (such as "1 month") are actually relative to the value of the seconds argument.

PARAMETER month_delta ||| INTEGER4 — The requested change to the month value; optional, defaults to zero.

PARAMETER second_delta ||| INTEGER4 — The requested change to the second of month value; optional, defaults to zero.

PARAMETER seconds ||| INTEGER8 — The number of seconds to adjust.

PARAMETER day_delta ||| INTEGER4 — The requested change to the day of month value; optional, defaults to zero.

PARAMETER minute_delta ||| INTEGER4 — The requested change to the minute value; optional, defaults to zero.

PARAMETER year_delta ||| INTEGER2 — The requested change to the year value; optional, defaults to zero.

PARAMETER hour_delta ||| INTEGER4 — The requested change to the hour value; optional, defaults to zero.

RETURN INTEGER8 — The adjusted Seconds_t value.

FUNCTION AdjustCalendar

Date \

Date_t	AdjustCalendar
	(Date_t date, INTEGER2 year_delta = 0, INTEGER4 month_delta = 0, INTEGER4 day_delta = 0)

Adjusts a date by incrementing or decrementing months and/or years. This routine uses the rule outlined in McGinn v. State, 46 Neb. 427, 65 N.W. 46 (1895): "The term calendar month, whether employed in statutes or contracts, and not appearing to have been used in a different sense, denotes a period terminating with the day of the succeeding month numerically corresponding to the day of its beginning, less one. If there be no corresponding day of the succeeding month, it terminates with the last day thereof." The internet suggests similar legal positions exist in the Commonwealth and Germany. Note that day adjustments are performed after year and month adjustments using the preceding rules. As an example, Jan. 31, 2014 + 1 month will result in Feb. 28, 2014; Jan. 31, 2014 + 1 month + 1 day will result in Mar. 1, 2014.

PARAMETER date ||| UNSIGNED4 — The date to adjust, in the Gregorian calendar after 1600.

PARAMETER month_delta ||| INTEGER4 — The requested change to the month value; optional, defaults to zero.

PARAMETER year_delta ||| INTEGER2 — The requested change to the year value; optional, defaults to zero.

PARAMETER day_delta ||| INTEGER4 — The requested change to the day value; optional, defaults to zero.

RETURN UNSIGNED4 — The adjusted Date_t value.

FUNCTION IsLocalDaylightSavingsInEffect

Date \

BOOLEAN	IsLocalDaylightSavingsInEffect
()	

Returns a boolean indicating whether daylight savings time is currently in effect locally.

RETURN BOOLEAN — TRUE if daylight savings time is currently in effect, FALSE otherwise.

FUNCTION LocalTimeZoneOffset

Date \

INTEGER4	LocalTimeZoneOffset
()	

Returns the offset (in seconds) of the time represented from UTC, with positive values indicating locations east of the Prime Meridian. Given a UTC time in seconds since epoch, you can find the local time by adding the result of this function to the seconds.

RETURN INTEGER4 — The number of seconds offset from UTC.

FUNCTION CurrentDate

Date \

Date_t	CurrentDate
(BOOLEAN in_local_time = FALSE)	

Returns the current date.

PARAMETER in_local_time ||| BOOLEAN — TRUE if the returned value should be local to the cluster computing the date, FALSE for UTC. Optional, defaults to FALSE.

RETURN UNSIGNED4 — A Date_t representing the current date.

FUNCTION Today

Date \

Date_t	Today
()	

Returns the current date in the local time zone.

RETURN UNSIGNED4 — A Date_t representing the current date.

FUNCTION CurrentTime

Date \

Time_t	CurrentTime
(BOOLEAN in_local_time = FALSE)	

Returns the current time of day

PARAMETER in_local_time ||| BOOLEAN — TRUE if the returned value should be local to the cluster computing the time, FALSE for UTC. Optional, defaults to FALSE.

RETURN UNSIGNED3 — A Time_t representing the current time of day.

FUNCTION CurrentSeconds

Date \

Seconds_t	CurrentSeconds
(BOOLEAN in_local_time = FALSE)	

Returns the current date and time as the number of seconds since epoch.

PARAMETER in_local_time ||| BOOLEAN — TRUE if the returned value should be local to the cluster computing the time, FALSE for UTC. Optional, defaults to FALSE.

RETURN INTEGER8 — A Seconds_t representing the current time in UTC or local time, depending on the argument.

FUNCTION CurrentTimestamp

Date \

Timestamp_t	CurrentTimestamp
(BOOLEAN in_local_time = FALSE)	

Returns the current date and time as the number of microseconds since epoch.

PARAMETER in_local_time ||| BOOLEAN — TRUE if the returned value should be local to the cluster computing the time, FALSE for UTC. Optional, defaults to FALSE.

RETURN INTEGER8 — A Timestamp_t representing the current time in microseconds in UTC or local time, depending on the argument.

MODULE DatesForMonth

[Date](#) \

	DatesForMonth
(Date_t as_of_date = CurrentDate(FALSE))	

Returns the beginning and ending dates for the month surrounding the given date.

PARAMETER as_of_date ||| UNSIGNED4 — The reference date from which the month will be calculated. This date must be a date within the Gregorian calendar. Optional, defaults to the current date in UTC.

RETURN — Module with exported attributes for startDate and endDate.

Children

1. [startDate](#) : No Documentation Found
2. [endDate](#) : No Documentation Found

ATTRIBUTE startDate

[Date](#) \ [DatesForMonth](#) \

Date_t	startDate
--------	-----------

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE endDate

[Date](#) \ [DatesForMonth](#) \

<code>Date_t</code>	<code>endDate</code>
---------------------	----------------------

No Documentation Found

RETURN `UNSIGNED4` —

MODULE `DatesForWeek`

`Date` \

<code>DatesForWeek</code>
<code>(Date_t as_of_date = CurrentDate(FALSE))</code>

Returns the beginning and ending dates for the week surrounding the given date (Sunday marks the beginning of a week).

PARAMETER `as_of_date` ||| `UNSIGNED4` — The reference date from which the week will be calculated. This date must be a date within the Gregorian calendar. Optional, defaults to the current date in UTC.

RETURN — Module with exported attributes for `startDate` and `endDate`.

Children

1. `startDate` : No Documentation Found
2. `endDate` : No Documentation Found

ATTRIBUTE `startDate`

`Date` \ `DatesForWeek` \

<code>Date_t</code>	<code>startDate</code>
---------------------	------------------------

No Documentation Found

RETURN UNSIGNED4 —

ATTRIBUTE endDate

Date \ DatesForWeek \

Date_t	endDate
--------	---------

No Documentation Found

RETURN UNSIGNED4 —

FUNCTION IsValidDate

Date \

BOOLEAN	IsValidDate
(Date_t date, INTEGER2 yearLowerBound = 1800, INTEGER2 yearUpperBound = 2100)	

Tests whether a date is valid, both by range-checking the year and by validating each of the other individual components.

PARAMETER date ||| UNSIGNED4 — The date to validate.

PARAMETER yearUpperBound ||| INTEGER2 — The maximum acceptable year. Optional; defaults to 2100.

PARAMETER yearLowerBound ||| INTEGER2 — The minimum acceptable year. Optional; defaults to 1800.

RETURN BOOLEAN — TRUE if the date is valid, FALSE otherwise.

FUNCTION IsValidGregorianDate

Date \

BOOLEAN	IsValidGregorianDate
(Date_t date)	

Tests whether a date is valid in the Gregorian calendar. The year must be between 1601 and 30827.

PARAMETER date ||| UNSIGNED4 — The Date_t to validate.

RETURN BOOLEAN — TRUE if the date is valid, FALSE otherwise.

FUNCTION IsValidTime

Date \

BOOLEAN	IsValidTime
(Time_t time)	

Tests whether a time is valid.

PARAMETER time ||| UNSIGNED3 — The time to validate.

RETURN BOOLEAN — TRUE if the time is valid, FALSE otherwise.

TRANSFORM CreateDate

Date \

Date_rec	CreateDate
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day)	

A transform to create a Date_rec from the individual elements

PARAMETER year ||| INTEGER2 — The year

PARAMETER month ||| UNSIGNED1 — The month (1-12).

PARAMETER day ||| UNSIGNED1 — The day (1..daysInMonth).

RETURN Date_rec — A transform that creates a Date_rec containing the date.

TRANSFORM CreateDateFromSeconds

Date \

<u>Date_rec</u>	CreateDateFromSeconds
(Seconds_t seconds)	

A transform to create a Date_rec from a Seconds_t value.

PARAMETER seconds ||| INTEGER8 — The number seconds since epoch.

RETURN Date_rec — A transform that creates a Date_rec containing the date.

TRANSFORM CreateTime

Date \

<u>Time_rec</u>	CreateTime
(UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second)	

A transform to create a Time_rec from the individual elements

PARAMETER minute ||| UNSIGNED1 — The minute (0-59).

PARAMETER second ||| UNSIGNED1 — The second (0-59).

PARAMETER hour ||| UNSIGNED1 — The hour (0-23).

RETURN `Time_rec` — A transform that creates a `Time_rec` containing the time of day.

TRANSFORM `CreateTimeFromSeconds`

`Date \`

<code>Time_rec</code>	<code>CreateTimeFromSeconds</code>
<code>(Seconds_t seconds)</code>	

A transform to create a `Time_rec` from a `Seconds_t` value.

PARAMETER `seconds` ||| `INTEGER8` — The number seconds since epoch.

RETURN `Time_rec` — A transform that creates a `Time_rec` containing the time of day.

TRANSFORM `CreateDateTime`

`Date \`

<code>DateTime_rec</code>	<code>CreateDateTime</code>
<code>(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day, UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second)</code>	

A transform to create a `DateTime_rec` from the individual elements

PARAMETER `year` ||| `INTEGER2` — The year

PARAMETER `second` ||| `UNSIGNED1` — The second (0-59).

PARAMETER `hour` ||| `UNSIGNED1` — The hour (0-23).

PARAMETER `minute` ||| `UNSIGNED1` — The minute (0-59).

PARAMETER `month` ||| `UNSIGNED1` — The month (1-12).

PARAMETER `day` ||| `UNSIGNED1` — The day (1..daysInMonth).

RETURN `DateTime_rec` — A transform that creates a `DateTime_rec` containing date and time components.

TRANSFORM `CreateDateTimeFromSeconds`

`Date \`

<code>DateTime_rec</code>	<code>CreateDateTimeFromSeconds</code>
<code>(Seconds_t seconds)</code>	

A transform to create a `DateTime_rec` from a `Seconds_t` value.

PARAMETER `seconds` `|||` `INTEGER8` — The number seconds since epoch.

RETURN `DateTime_rec` — A transform that creates a `DateTime_rec` containing date and time components.

File

[Go Up](#)

IMPORTS

lib_fileservices |

DESCRIPTIONS

MODULE File

File

No Documentation Found

Children

1. [FsFilenameRecord](#) : A record containing information about filename
2. [FsLogicalFileName](#) : An alias for a logical filename that is stored in a row
3. [FsLogicalFileNameRecord](#) : A record containing a logical filename
4. [FsLogicalFileInfoRecord](#) : A record containing information about a logical file
5. [FsLogicalSuperSubRecord](#) : A record containing information about a superfile and its contents
6. [FsFileRelationshipRecord](#) : A record containing information about the relationship between two files
7. [RECFMV_RECSIZE](#) : Constant that indicates IBM RECFM V format file
8. [RECFMVB_RECSIZE](#) : Constant that indicates IBM RECFM VB format file
9. [PREFIX_VARIABLE_RECSIZE](#) : Constant that indicates a variable little endian 4 byte length prefixed file

10. [PREFIX_VARIABLE_BIGENDIAN_RECSIZE](#) : Constant that indicates a variable big endian 4 byte length prefixed file
11. [FileExists](#) : Returns whether the file exists
12. [DeleteLogicalFile](#) : Removes the logical file from the system, and deletes from the disk
13. [SetReadOnly](#) : Changes whether access to a file is read only or not
14. [RenameLogicalFile](#) : Changes the name of a logical file
15. [ForeignLogicalFileName](#) : Returns a logical filename that can be used to refer to a logical file in a local or remote dali
16. [ExternalLogicalFileName](#) : Returns an encoded logical filename that can be used to refer to a external file
17. [GetFileDescription](#) : Returns a string containing the description information associated with the specified filename
18. [SetFileDescription](#) : Sets the description associated with the specified filename
19. [RemoteDirectory](#) : Returns a dataset containing a list of files from the specified machineIP and directory
20. [LogicalFileList](#) : Returns a dataset of information about the logical files known to the system
21. [CompareFiles](#) : Compares two files, and returns a result indicating how well they match
22. [VerifyFile](#) : Checks the system datastore (Dali) information for the file against the physical parts on disk
23. [AddFileRelationship](#) : Defines the relationship between two files
24. [FileRelationshipList](#) : Returns a dataset of relationships
25. [RemoveFileRelationship](#) : Removes a relationship between two files
26. [GetColumnMapping](#) : Returns the field mappings for the file, in the same format specified for the SetColumnMapping function
27. [SetColumnMapping](#) : Defines how the data in the fields of the file must be transformed between the actual data storage format and the input format used to query that data
28. [EncodeRfsQuery](#) : Returns a string that can be used in a DATASET declaration to read data from an RFS (Remote File Server) instance (e.g
29. [RfsAction](#) : Sends the query to the rfs server
30. [MoveExternalFile](#) : Moves the single physical file between two locations on the same remote machine
31. [DeleteExternalFile](#) : Removes a single physical file from a remote machine
32. [CreateExternalDirectory](#) : Creates the path on the location (if it does not already exist)
33. [GetLogicalFileAttribute](#) : Returns the value of the given attribute for the specified logicalfilename

34. [ProtectLogicalFile](#) : Toggles protection on and off for the specified logicalfilename
35. [DfuPlusExec](#) : The DfuPlusExec action executes the specified command line just as the DfuPlus.exe program would do
36. [fSprayFixed](#) : Sprays a file of fixed length records from a single machine and distributes it across the nodes of the destination group
37. [SprayFixed](#) : Same as fSprayFixed, but does not return the DFU Workunit ID
38. [fSprayVariable](#) : No Documentation Found
39. [SprayVariable](#) : No Documentation Found
40. [fSprayDelimited](#) : Sprays a file of fixed delimited records from a single machine and distributes it across the nodes of the destination group
41. [SprayDelimited](#) : Same as fSprayDelimited, but does not return the DFU Workunit ID
42. [fSprayXml](#) : Sprays an xml file from a single machine and distributes it across the nodes of the destination group
43. [SprayXml](#) : Same as fSprayXml, but does not return the DFU Workunit ID
44. [fDespray](#) : Copies a distributed file from multiple machines, and desprays it to a single file on a single machine
45. [Despray](#) : Same as fDespray, but does not return the DFU Workunit ID
46. [fCopy](#) : Copies a distributed file to another distributed file
47. [Copy](#) : Same as fCopy, but does not return the DFU Workunit ID
48. [fReplicate](#) : Ensures the specified file is replicated to its mirror copies
49. [Replicate](#) : Same as fReplicated, but does not return the DFU Workunit ID
50. [fRemotePull](#) : Copies a distributed file to a distributed file on remote system
51. [RemotePull](#) : Same as fRemotePull, but does not return the DFU Workunit ID
52. [fMonitorLogicalFileName](#) : Creates a file monitor job in the DFU Server
53. [MonitorLogicalFileName](#) : Same as fMonitorLogicalFileName, but does not return the DFU Workunit ID
54. [fMonitorFile](#) : Creates a file monitor job in the DFU Server
55. [MonitorFile](#) : Same as fMonitorFile, but does not return the DFU Workunit ID
56. [WaitDfuWorkunit](#) : Waits for the specified DFU workunit to finish
57. [AbortDfuWorkunit](#) : Aborts the specified DFU workunit
58. [CreateSuperFile](#) : Creates an empty superfile
59. [SuperFileExists](#) : Checks if the specified filename is present in the Distributed File Utility (DFU) and is a SuperFile

60. [DeleteSuperFile](#) : Deletes the superfile
61. [GetSuperFileSubCount](#) : Returns the number of sub-files contained within a superfile
62. [GetSuperFileSubName](#) : Returns the name of the Nth sub-file within a superfile
63. [FindSuperFileSubName](#) : Returns the position of a file within a superfile
64. [StartSuperFileTransaction](#) : Starts a superfile transaction
65. [AddSuperFile](#) : Adds a file to a superfile
66. [RemoveSuperFile](#) : Removes a sub-file from a superfile
67. [ClearSuperFile](#) : Removes all sub-files from a superfile
68. [RemoveOwnedSubFiles](#) : Removes all soley-owned sub-files from a superfile
69. [DeleteOwnedSubFiles](#) : Legacy version of RemoveOwnedSubFiles which was incorrectly named in a previous version
70. [SwapSuperFile](#) : Swap the contents of two superfiles
71. [ReplaceSuperFile](#) : Removes a sub-file from a superfile and replaces it with another
72. [FinishSuperFileTransaction](#) : Finishes a superfile transaction
73. [SuperFileContents](#) : Returns the list of sub-files contained within a superfile
74. [LogicalFileSuperOwners](#) : Returns the list of superfiles that a logical file is contained within
75. [LogicalFileSuperSubList](#) : Returns the list of all the superfiles in the system and their component sub-files
76. [fPromoteSuperFileList](#) : Moves the sub-files from the first entry in the list of superfiles to the next in the list, repeating the process through the list of superfiles
77. [PromoteSuperFileList](#) : Same as fPromoteSuperFileList, but does not return the DFU Workunit ID

RECORD **FsFilenameRecord**

[File](#) \

	FsFilenameRecord
--	-------------------------

A record containing information about filename. Includes name, size and when last modified. export
 FsFilenameRecord := RECORD string name; integer8 size; string19 modified; END;

FIELD size ||| INTEGER8 — No Doc

FIELD name ||| STRING — No Doc

FIELD modified ||| STRING19 — No Doc

ATTRIBUTE FsLogicalFileName

File \

FsLogicalFileName

An alias for a logical filename that is stored in a row.

RETURN STRING —

RECORD FsLogicalFileNameRecord

File \

FsLogicalFileNameRecord

A record containing a logical filename. It contains the following fields:

FIELD name ||| STRING — The logical name of the file;

RECORD FsLogicalFileInfoRecord

File \

FsLogicalFileInfoRecord

A record containing information about a logical file.

FIELD size ||| INTEGER8 — Number of bytes in the file (before compression)

FIELD superfile ||| BOOLEAN — Is this a superfile?

FIELD rowcount ||| INTEGER8 — Number of rows in the file.

FIELD name ||| STRING — No Doc

FIELD owner ||| STRING — No Doc

FIELD modified ||| STRING19 — No Doc

FIELD cluster ||| STRING — No Doc

RECORD FsLogicalSuperSubRecord

File \

FsLogicalSuperSubRecord

A record containing information about a superfile and its contents.

FIELD subname ||| STRING — The name of the sub-file

FIELD supername ||| STRING — The name of the superfile

RECORD FsFileRelationshipRecord

File \

FsFileRelationshipRecord

A record containing information about the relationship between two files.

FIELD payload ||| BOOLEAN — Indicates whether the primary or secondary are payload INDEXes.

FIELD primaryfile ||| STRING — The logical filename of the primary file

FIELD primaryflds ||| STRING — The name of the primary key field for the primary file. The value "fileposition" indicates the secondary is an INDEX that must use FETCH to access non-keyed fields.

FIELD secondaryfile ||| STRING — The logical filename of the secondary file.

FIELD kind ||| STRING — The type of relationship between the primary and secondary files. Containing either 'link' or 'view'.

FIELD secondaryflds ||| STRING — The name of the foreign key field relating to the primary file.

FIELD cardinality ||| STRING — The cardinality of the relationship. The format is <primary>:<secondary>. Valid values are "1" or "M".</secondary></primary>

FIELD description ||| STRING — The description of the relationship.

ATTRIBUTE RECFMV_RECSIZE

File \

RECFMV_RECSIZE

Constant that indicates IBM RECFM V format file. Can be passed to SprayFixed for the record size.

RETURN INTEGER4 —

ATTRIBUTE RECFMVB_RECSIZE

File \

RECFMVB_RECSIZE

Constant that indicates IBM RECFM VB format file. Can be passed to SprayFixed for the record size.

RETURN INTEGER4 —

ATTRIBUTE PREFIX_VARIABLE_RECSIZE

File \

INTEGER4	PREFIX_VARIABLE_RECSIZE
----------	-------------------------

Constant that indicates a variable little endian 4 byte length prefixed file. Can be passed to SprayFixed for the record size.

RETURN INTEGER4 —

ATTRIBUTE PREFIX_VARIABLE_BIGENDIAN_RECSIZE

File \

INTEGER4	PREFIX_VARIABLE_BIGENDIAN_RECSIZE
----------	-----------------------------------

Constant that indicates a variable big endian 4 byte length prefixed file. Can be passed to SprayFixed for the record size.

RETURN INTEGER4 —

FUNCTION FileExists

File \

boolean	FileExists
---------	------------

(varstring lfn, boolean physical=FALSE)

Returns whether the file exists.

PARAMETER lfn ||| VARSTRING — The logical name of the file.

PARAMETER physical ||| BOOLEAN — Whether to also check for the physical existence on disk. Defaults to FALSE.

RETURN BOOLEAN — Whether the file exists.

FUNCTION DeleteLogicalFile

File \

DeleteLogicalFile
(varstring lfn, boolean allowMissing=FALSE)

Removes the logical file from the system, and deletes from the disk.

PARAMETER lfn ||| VARSTRING — The logical name of the file.

PARAMETER allowMissing ||| BOOLEAN — Whether to suppress an error if the filename does not exist. Defaults to FALSE.

RETURN —

FUNCTION SetReadOnly

File \

SetReadOnly
(varstring lfn, boolean ro=TRUE)

Changes whether access to a file is read only or not.

PARAMETER ro ||| BOOLEAN — Whether updates to the file are disallowed. Defaults to TRUE.

PARAMETER lfn ||| VARSTRING — The logical name of the file.

RETURN —

FUNCTION RenameLogicalFile

File \

	RenameLogicalFile
(varstring oldname, varstring newname)	

Changes the name of a logical file.

PARAMETER newname ||| VARSTRING — The new logical name of the file.

PARAMETER oldname ||| VARSTRING — The current name of the file to be renamed.

RETURN —

FUNCTION ForeignLogicalFileName

File \

varstring	ForeignLogicalFileName
(varstring name, varstring foreigndali="", boolean abspath=FALSE)	

Returns a logical filename that can be used to refer to a logical file in a local or remote dali.

PARAMETER foreigndali ||| VARSTRING — The IP address of the foreign dali used to resolve the file. If blank then the file is resolved locally. Defaults to blank.

PARAMETER name ||| VARSTRING — The logical name of the file.

PARAMETER abspath ||| BOOLEAN — Should a tilde (~) be prepended to the resulting logical file name. Defaults to FALSE.

RETURN VARSTRING —

FUNCTION ExternalLogicalFileName

File \

varstring	ExternalLogicalFileName
(varstring location, varstring path, boolean abspath=TRUE)	

Returns an encoded logical filename that can be used to refer to a external file. Examples include directly reading from a landing zone. Upper case characters and other details are escaped.

PARAMETER location ||| VARSTRING — The IP address of the remote machine. '.' can be used for the local machine.

PARAMETER path ||| VARSTRING — The path/name of the file on the remote machine.

PARAMETER abspath ||| BOOLEAN — Should a tilde (~) be prepended to the resulting logical file name. Defaults to TRUE.

RETURN VARSTRING — The encoded logical filename.

FUNCTION GetFileDescription

File \

varstring	GetFileDescription
(varstring lfn)	

Returns a string containing the description information associated with the specified filename. This description is set either through ECL watch or by using the FileServices.SetFileDescription function.

PARAMETER lfn ||| VARSTRING — The logical name of the file.

RETURN VARSTRING —

FUNCTION SetFileDescription

File \

	SetFileDescription
(varstring lfn, varstring val)	

Sets the description associated with the specified filename.

PARAMETER val ||| VARSTRING — The description to be associated with the file.

PARAMETER lfn ||| VARSTRING — The logical name of the file.

RETURN —

FUNCTION RemoteDirectory

File \

dataset(FsFilenameRecord)	RemoteDirectory
(varstring machineIP, varstring dir, varstring mask='*', boolean recurse=FALSE)	

Returns a dataset containing a list of files from the specified machineIP and directory.

PARAMETER mask ||| VARSTRING — The filemask specifying which files to include in the result.
Defaults to '*' (all files).

PARAMETER machineIP ||| VARSTRING — The IP address of the remote machine.

PARAMETER recurse ||| BOOLEAN — Whether to include files from subdirectories under the directory. Defaults to FALSE.

PARAMETER directory ||| — The path to the directory to read. This must be in the appropriate format for the operating system running on the remote machine.

PARAMETER dir ||| VARSTRING — No Doc

RETURN TABLE (FsFilenameRecord) —

FUNCTION LogicalFileList

File \

<code>dataset(FsLogicalFileInfoRecord)</code>	LogicalFileList
<pre>(varstring namepattern='*', boolean includenormal=TRUE, boolean includesuper=FALSE, boolean unknownszero=FALSE, varstring foreigndali="")</pre>	

Returns a dataset of information about the logical files known to the system.

PARAMETER unknownszero ||| BOOLEAN — Whether to set file sizes that are unknown to zero(0) instead of minus-one (-1). Defaults to FALSE.

PARAMETER namepattern ||| VARSTRING — The mask of the files to list. Defaults to '*' (all files).

PARAMETER foreigndali ||| VARSTRING — The IP address of the foreign dali used to resolve the file. If blank then the file is resolved locally. Defaults to blank.

PARAMETER includesuper ||| BOOLEAN — Whether to include SuperFiles. Defaults to FALSE.

PARAMETER includenormal ||| BOOLEAN — Whether to include 'normal' files. Defaults to TRUE.

RETURN TABLE (FsLogicalFileInfoRecord) —

FUNCTION CompareFiles

File \

INTEGER4	CompareFiles
<pre>(varstring lfn1, varstring lfn2, boolean logical_only=TRUE, boolean use_crcs=FALSE)</pre>	

Compares two files, and returns a result indicating how well they match.

PARAMETER logical_only ||| BOOLEAN — Whether to only compare logical information in the system datastore (Dali), and ignore physical information on disk. [Default TRUE]

PARAMETER file1 ||| — The logical name of the first file.

PARAMETER file2 ||| — The logical name of the second file.

PARAMETER use_crcs ||| BOOLEAN — Whether to compare physical CRCs of all the parts on disk. This may be slow on large files. Defaults to FALSE.

PARAMETER lfn2 ||| VARSTRING — No Doc

PARAMETER lfn1 ||| VARSTRING — No Doc

RETURN **INTEGER4** — 0 if file1 and file2 match exactly 1 if file1 and file2 contents match, but file1 is newer than file2 -1 if file1 and file2 contents match, but file2 is newer than file1 2 if file1 and file2 contents do not match and file1 is newer than file2 -2 if file1 and file2 contents do not match and file2 is newer than file1

FUNCTION VerifyFile

File \

varstring	VerifyFile
(varstring lfn, boolean usecrcs)	

Checks the system datastore (Dali) information for the file against the physical parts on disk.

PARAMETER use_crcs ||| — Whether to compare physical CRCs of all the parts on disk. This may be slow on large files.

PARAMETER lfn ||| VARSTRING — The name of the file to check.

PARAMETER usecrcs ||| BOOLEAN — No Doc

RETURN **VARSTRING** — 'OK' - The file parts match the datastore information 'Could not find file: <filename>' - The logical filename was not found 'Could not find part file: <partname>' - The partname was not found 'Modified time differs for: <partname>' - The partname has a different timestamp 'File size differs for: <partname>' - The partname has a file size 'File CRC differs for: <partname>' - The partname has a different CRC</partname></partname></partname></partname></filename>

FUNCTION AddFileRelationship

File \

AddFileRelationship
(varstring primary, varstring secondary, varstring primaryflds, varstring secondaryflds, varstring kind='link', varstring cardinality, boolean payload, varstring description="")

Defines the relationship between two files. These may be DATASETs or INDEXes. Each record in the primary file should be uniquely defined by the primaryfields (ideally), preferably efficiently. This information is used by the roxie browser to link files together.

PARAMETER primaryfields ||| — The name of the primary key field for the primary file. The value "___fileposition___" indicates the secondary is an INDEX that must use FETCH to access non-keyed fields.

PARAMETER payload ||| BOOLEAN — Indicates whether the primary or secondary are payload INDEXes.

PARAMETER description ||| VARSTRING — The description of the relationship.

PARAMETER secondary ||| VARSTRING — The logical filename of the secondary file.

PARAMETER cardinality ||| VARSTRING — The cardinality of the relationship. The format is <primary>:<secondary>. Valid values are "1" or "M".</secondary></primary>

PARAMETER relationship ||| — The type of relationship between the primary and secondary files. Containing either 'link' or 'view'. Default is "link".

PARAMETER secondaryfields ||| — The name of the foreign key field relating to the primary file.

PARAMETER primary ||| VARSTRING — The logical filename of the primary file.

PARAMETER primaryflds ||| VARSTRING — No Doc

PARAMETER kind ||| VARSTRING — No Doc

PARAMETER secondaryflds ||| VARSTRING — No Doc

RETURN —

FUNCTION FileRelationshipList

File \

<code>dataset(FsFileRelationshipRecord)</code>	FileRelationshipList
<pre>(varstring primary, varstring secondary, varstring primflds=", varstring secondaryflds=", varstring kind='link')</pre>	

Returns a dataset of relationships. The return records are structured in the FsFileRelationshipRecord format.

PARAMETER primaryfields ||| — The name of the primary key field for the primary file.

PARAMETER relationship ||| — The type of relationship between the primary and secondary files. Containing either 'link' or 'view'. Default is "link".

PARAMETER secondaryfields ||| — The name of the foreign key field relating to the primary file.

PARAMETER primary ||| VARSTRING — The logical filename of the primary file.

PARAMETER secondary ||| VARSTRING — The logical filename of the secondary file.

PARAMETER secondaryflds ||| VARSTRING — No Doc

PARAMETER primflds ||| VARSTRING — No Doc

PARAMETER kind ||| VARSTRING — No Doc

RETURN TABLE (FsFileRelationshipRecord) —

FUNCTION RemoveFileRelationship

File \

RemoveFileRelationship
<pre>(varstring primary, varstring secondary, varstring primaryflds=", varstring secondaryflds=", varstring kind='link')</pre>

Removes a relationship between two files.

PARAMETER primaryfields ||| — The name of the primary key field for the primary file.

PARAMETER relationship ||| — The type of relationship between the primary and secondary files.
Containing either 'link' or 'view'. Default is "link".

PARAMETER secondaryfields ||| — The name of the foreign key field relating to the primary file.

PARAMETER primary ||| VARSTRING — The logical filename of the primary file.

PARAMETER secondary ||| VARSTRING — The logical filename of the secondary file.

PARAMETER secondaryflds ||| VARSTRING — No Doc

PARAMETER kind ||| VARSTRING — No Doc

PARAMETER primaryflds ||| VARSTRING — No Doc

RETURN —

FUNCTION GetColumnMapping

File \

varstring	GetColumnMapping
(varstring lfn)	

Returns the field mappings for the file, in the same format specified for the SetColumnMapping function.

PARAMETER lfn ||| VARSTRING — The logical filename of the primary file.

RETURN VARSTRING —

FUNCTION SetColumnMapping

File \

SetColumnMapping
(varstring lfn, varstring mapping)

Defines how the data in the fields of the file must be transformed between the actual data storage format and the input format used to query that data. This is used by the user interface of the roxie browser.

PARAMETER mapping ||| VARSTRING — A string containing a comma separated list of field mappings.

PARAMETER lfn ||| VARSTRING — The logical filename of the primary file.

RETURN —

FUNCTION EncodeRfsQuery

File \

varstring	EncodeRfsQuery
(varstring server, varstring query)	

Returns a string that can be used in a DATASET declaration to read data from an RFS (Remote File Server) instance (e.g. rfsmysql) on another node.

PARAMETER query ||| VARSTRING — The text of the query to send to the server

PARAMETER server ||| VARSTRING — A string containing the ip:port address for the remote file server.

RETURN VARSTRING —

FUNCTION RfsAction

File \

RfsAction
(varstring server, varstring query)

Sends the query to the rfs server.

PARAMETER query ||| VARSTRING — The text of the query to send to the server

PARAMETER server ||| VARSTRING — A string containing the ip:port address for the remote file server.

RETURN —

FUNCTION MoveExternalFile

File \

MoveExternalFile
(varstring location, varstring frompath, varstring topath)

Moves the single physical file between two locations on the same remote machine. The dfileserv utility program must be running on the location machine.

PARAMETER location ||| VARSTRING — The IP address of the remote machine.

PARAMETER frompath ||| VARSTRING — The path/name of the file to move.

PARAMETER topath ||| VARSTRING — The path/name of the target file.

RETURN —

FUNCTION DeleteExternalFile

File \

DeleteExternalFile
(varstring location, varstring path)

Removes a single physical file from a remote machine. The dafileserv utility program must be running on the location machine.

PARAMETER location ||| VARSTRING — The IP address of the remote machine.

PARAMETER path ||| VARSTRING — The path/name of the file to remove.

RETURN —

FUNCTION CreateExternalDirectory

File \

CreateExternalDirectory
(varstring location, varstring path)

Creates the path on the location (if it does not already exist). The dafileserv utility program must be running on the location machine.

PARAMETER location ||| VARSTRING — The IP address of the remote machine.

PARAMETER path ||| VARSTRING — The path/name of the file to remove.

RETURN —

FUNCTION GetLogicalFileAttribute

File \

varstring	GetLogicalFileAttribute
(varstring lfn, varstring attrname)	

Returns the value of the given attribute for the specified logicalfilename.

PARAMETER attrname ||| VARSTRING — The name of the file attribute to return.

PARAMETER lfn ||| VARSTRING — The name of the logical file.

RETURN VARSTRING —

FUNCTION ProtectLogicalFile

File \

ProtectLogicalFile
(varstring lfn, boolean value=TRUE)

Toggles protection on and off for the specified logicalfilename.

PARAMETER lfn ||| VARSTRING — The name of the logical file.

PARAMETER value ||| BOOLEAN — TRUE to enable protection, FALSE to disable.

RETURN —

FUNCTION DfuPlusExec

File \

DfuPlusExec
(varstring cmdline)

The DfuPlusExec action executes the specified command line just as the DfuPLus.exe program would do. This allows you to have all the functionality of the DfuPLUS.exe program available within your ECL code. param cmdline The DFUPlus.exe command line to execute. The valid arguments are documented in the Client Tools manual, in the section describing the DfuPlus.exe program.

PARAMETER cmdline ||| VARSTRING — No Doc

RETURN —

FUNCTION fSprayFixed

File \

varstring	fSprayFixed
<pre>(varstring sourceIP, varstring sourcePath, integer4 recordSize, varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, boolean failIfNoSourceFile=FALSE, integer4 expireDays=-1)</pre>	

Sprays a file of fixed length records from a single machine and distributes it across the nodes of the destination group.

PARAMETER destinationLogicalName ||| VARSTRING — The logical name of the file to create.

PARAMETER expireDays ||| INTEGER4 — Number of days to auto-remove file. Default is -1, not expire.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER recordsize ||| INTEGER4 — The size (in bytes) of the records in the file.

PARAMETER sourcePath ||| VARSTRING — The path and name of the file.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER compress ||| BOOLEAN — Whether to compress the new file. Defaults to FALSE.

PARAMETER sourceIP ||| VARSTRING — The IP address of the file.

PARAMETER replicate ||| BOOLEAN — Whether to replicate the new file. Defaults to FALSE.

PARAMETER destinationGroup ||| VARSTRING — The name of the group to distribute the file across.

PARAMETER maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.

PARAMETER failIfNoSourceFile ||| BOOLEAN — If TRUE it causes a missing source file to trigger a failure. Defaults to FALSE.

PARAMETER allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION SprayFixed

File \

SprayFixed
<pre>(varstring sourceIP, varstring sourcePath, integer4 recordSize, varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, boolean failIfNoSourceFile=FALSE, integer4 expireDays=-1)</pre>

Same as fSprayFixed, but does not return the DFU Workunit ID.

PARAMETER sourceip ||| VARSTRING — No Doc

PARAMETER recordsize ||| INTEGER4 — No Doc

PARAMETER expiredays ||| INTEGER4 — No Doc

PARAMETER failifnosourcefile ||| BOOLEAN — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER sourcepath ||| VARSTRING — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

RETURN —

SEE fSprayFixed

FUNCTION fSprayVariable

File \

varstring	fSprayVariable
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceCsvSeparate='\\', varstring sourceCsvTerminate='\\n,\\r\\n', varstring sourceCsvQuote='\"', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, varstring sourceCsvEscape=", boolean failIfNoSourceFile=FALSE, boolean recordStructurePresent=FALSE, boolean quotedTerminator=TRUE, varstring encoding='ascii', integer4 expireDays=-1)</pre>	

No Documentation Found

PARAMETER sourceip ||| VARSTRING — No Doc

PARAMETER quotedterminator ||| BOOLEAN — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

PARAMETER failifnosourcefile ||| BOOLEAN — No Doc

PARAMETER sourcemaxrecordsize ||| INTEGER4 — No Doc

PARAMETER sourcecsvseparate ||| VARSTRING — No Doc

PARAMETER encoding ||| VARSTRING — No Doc

PARAMETER sourcecsvquote ||| VARSTRING — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER sourcecsvescape ||| VARSTRING — No Doc

PARAMETER recordstructurepresent ||| BOOLEAN — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER sourcepath ||| VARSTRING — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER expiredays ||| INTEGER4 — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER sourcecsvterminate ||| VARSTRING — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

RETURN VARSTRING —

FUNCTION SprayVariable

File \

SprayVariable
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceCsvSeparate='\,', varstring sourceCsvTerminate='\n,\r\n', varstring sourceCsvQuote='"', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, varstring sourceCsvEscape="", boolean failIfNoSourceFile=FALSE, boolean recordStructurePresent=FALSE, boolean quotedTerminator=TRUE, varstring encoding='ascii', integer4 expireDays=-1)</pre>

No Documentation Found

PARAMETER sourceip ||| VARSTRING — No Doc

PARAMETER quotedterminator ||| BOOLEAN — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

PARAMETER	<u>failifnosourcefile</u> BOOLEAN — No Doc
PARAMETER	<u>sourcemaxrecordsize</u> INTEGER4 — No Doc
PARAMETER	<u>sourcecsvseparate</u> VARSTRING — No Doc
PARAMETER	<u>encoding</u> VARSTRING — No Doc
PARAMETER	<u>sourcecsvquote</u> VARSTRING — No Doc
PARAMETER	<u>maxconnections</u> INTEGER4 — No Doc
PARAMETER	<u>sourcecsvescape</u> VARSTRING — No Doc
PARAMETER	<u>recordstructurepresent</u> BOOLEAN — No Doc
PARAMETER	<u>compress</u> BOOLEAN — No Doc
PARAMETER	<u>sourcepath</u> VARSTRING — No Doc
PARAMETER	<u>allowoverwrite</u> BOOLEAN — No Doc
PARAMETER	<u>replicate</u> BOOLEAN — No Doc
PARAMETER	<u>expiredays</u> INTEGER4 — No Doc
PARAMETER	<u>espserveripport</u> VARSTRING — No Doc
PARAMETER	<u>sourcecsvterminate</u> VARSTRING — No Doc
PARAMETER	<u>destinationlogicalname</u> VARSTRING — No Doc
PARAMETER	<u>timeout</u> INTEGER4 — No Doc
RETURN	—

FUNCTION fSprayDelimited

File \

varstring	fSprayDelimited
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceCsvSeparate='\\', varstring sourceCsvTerminate='\\n,\\r\\n', varstring sourceCsvQuote='\"', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, varstring sourceCsvEscape=", boolean failIfNoSourceFile=FALSE, boolean recordStructurePresent=FALSE, boolean quotedTerminator=TRUE, varstring encoding='ascii', integer4 expireDays=-1)</pre>	

Sprays a file of fixed delimited records from a single machine and distributes it across the nodes of the destination group.

PARAMETER destinationLogicalName ||| VARSTRING — The logical name of the file to create.

PARAMETER expireDays ||| INTEGER4 — Number of days to auto-remove file. Default is -1, not expire.

PARAMETER sourceMaxRecordSize ||| INTEGER4 — The maximum size (in bytes) of the records in the file.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER quotedTerminator ||| BOOLEAN — Can the terminator character be included in a quoted field. Defaults to TRUE. If FALSE it allows quicker partitioning of the file (avoiding a complete file scan).

PARAMETER sourcePath ||| VARSTRING — The path and name of the file.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER compress ||| BOOLEAN — Whether to compress the new file. Defaults to FALSE.

PARAMETER sourceIP ||| VARSTRING — The IP address of the file.

PARAMETER sourceCsvSeparate ||| VARSTRING — The character sequence which separates fields in the file.

PARAMETER replicate ||| BOOLEAN — Whether to replicate the new file. Defaults to FALSE.

PARAMETER sourceCsvTerminate ||| VARSTRING — The character sequence which separates records in the file.

PARAMETER destinationGroup ||| VARSTRING — The name of the group to distribute the file across.

PARAMETER recordStructurePresent ||| BOOLEAN — If TRUE derives the record structure from the header of the file.

PARAMETER sourceCsvEscape ||| VARSTRING — A character that is used to escape quote characters. Defaults to none.

PARAMETER maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.

PARAMETER failIfNoSourceFile ||| BOOLEAN — If TRUE it causes a missing source file to trigger a failure. Defaults to FALSE.

PARAMETER allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE

PARAMETER sourceCsvQuote ||| VARSTRING — A string which can be used to delimit fields in the file.

PARAMETER encoding ||| VARSTRING — No Doc

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION SprayDelimited

File \

SprayDelimited
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceCsvSeparate='\,', varstring sourceCsvTerminate='\n,\\r\\n', varstring sourceCsvQuote='"', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, varstring sourceCsvEscape="", boolean failIfNoSourceFile=FALSE, boolean recordStructurePresent=FALSE, boolean quotedTerminator=TRUE, const varstring encoding='ascii', integer4 expireDays=-1)</pre>

Same as fSprayDelimited, but does not return the DFU Workunit ID.

PARAMETER sourceip ||| VARSTRING — No Doc

PARAMETER quotedterminator ||| BOOLEAN — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

PARAMETER failifnosourcefile ||| BOOLEAN — No Doc

PARAMETER sourcemaxrecordsize ||| INTEGER4 — No Doc

PARAMETER sourcecsvseparate ||| VARSTRING — No Doc

PARAMETER encoding ||| VARSTRING — No Doc

PARAMETER sourcecsvquote ||| VARSTRING — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER sourcecsvescape ||| VARSTRING — No Doc

PARAMETER recordstructurepresent ||| BOOLEAN — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER sourcepath ||| VARSTRING — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER expiredays ||| INTEGER4 — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER sourcecsvterminate ||| VARSTRING — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

RETURN —

SEE fSprayDelimited

FUNCTION fSprayXml

File \

varstring	fSprayXml
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceRowTag, varstring sourceEncoding='utf8', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, boolean failIfNoSourceFile=FALSE, integer4 expireDays=-1)</pre>	

Sprays an xml file from a single machine and distributes it across the nodes of the destination group.

PARAMETER destinationLogicalName ||| VARSTRING — The logical name of the file to create.

PARAMETER expireDays ||| INTEGER4 — Number of days to auto-remove file. Default is -1, not expire.

PARAMETER sourceMaxRecordSize ||| INTEGER4 — The maximum size (in bytes) of the records in the file.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER sourceRowTag ||| VARSTRING — The xml tag that is used to delimit records in the source file. (This tag cannot recursively nest.)

PARAMETER sourcePath ||| VARSTRING — The path and name of the file.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER compress ||| BOOLEAN — Whether to compress the new file. Defaults to FALSE.

PARAMETER sourceIP ||| VARSTRING — The IP address of the file.

PARAMETER replicate ||| BOOLEAN — Whether to replicate the new file. Defaults to FALSE.

PARAMETER destinationGroup ||| VARSTRING — The name of the group to distribute the file across.

PARAMETER sourceEncoding ||| VARSTRING — The unicode encoding of the file. (utf8,utf8n,utf16be,utf16le,utf32be,utf32le)

PARAMETER maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.

PARAMETER failIfNoSourceFile ||| BOOLEAN — If TRUE it causes a missing source file to trigger a failure. Defaults to FALSE.

PARAMETER allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION SprayXml

File \

SprayXml
<pre>(varstring sourceIP, varstring sourcePath, integer4 sourceMaxRecordSize=8192, varstring sourceRowTag, varstring sourceEncoding='utf8', varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean compress=FALSE, boolean failIfNoSourceFile=FALSE, integer4 expireDays=-1)</pre>

Same as fSprayXml, but does not return the DFU Workunit ID.

PARAMETER sourceip ||| VARSTRING — No Doc

PARAMETER expiredays ||| INTEGER4 — No Doc

PARAMETER failifnosourcefile ||| BOOLEAN — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

PARAMETER sourceencoding ||| VARSTRING — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER sourcepath ||| VARSTRING — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER sourcerowtag ||| VARSTRING — No Doc

PARAMETER sourcemaxrecordsize ||| INTEGER4 — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

RETURN —

SEE fSprayXml

FUNCTION fDespray

File \

varstring	fDespray
<pre>(varstring logicalName, varstring destinationIP, varstring destinationPath, integer4 timeout=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE)</pre>	

Copies a distributed file from multiple machines, and desprays it to a single file on a single machine.

PARAMETER maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER destinationPath ||| VARSTRING — The path of the file to create on the destination machine.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER logicalName ||| VARSTRING — The name of the file to despray.

PARAMETER allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE

PARAMETER destinationIP ||| VARSTRING — The IP of the target machine.

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION Despray

File \

	Despray
<pre>(varstring logicalName, varstring destinationIP, varstring destinationPath, integer4 timeout=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE)</pre>	

Same as fDespray, but does not return the DFU Workunit ID.

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER destinationpath ||| VARSTRING — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER logicalname ||| VARSTRING — No Doc

PARAMETER destinationip ||| VARSTRING — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

RETURN —

SEE fDespray

FUNCTION fCopy

File \

varstring	fCopy
<pre>(varstring sourceLogicalName, varstring destinationGroup, varstring destinationLogicalName, varstring sourceDali=", integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean asSuperfile=FALSE, boolean compress=FALSE, boolean forcePush=FALSE, integer4 transferBufferSize=0, boolean preserveCompression=TRUE)</pre>	

Copies a distributed file to another distributed file.

PARAMETER sourceDali ||| VARSTRING — The dali that contains the source file (blank implies same dali). Defaults to same dali.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER compress ||| BOOLEAN — Whether to compress the new file. Defaults to FALSE.

- PARAMETER** asSuperfile ||| BOOLEAN — Should the file be copied as a superfile? If TRUE and source is a superfile, then the operation creates a superfile on the target, creating sub-files as needed and only overwriting existing sub-files whose content has changed. If FALSE, a single file is created. Defaults to FALSE.
- PARAMETER** destinationLogicalName ||| VARSTRING — The logical name of the file to create.
- PARAMETER** sourceLogicalName ||| VARSTRING — The name of the file to despray.
- PARAMETER** transferBufferSize ||| INTEGER4 — Overrides the size (in bytes) of the internal buffer used to copy the file. Default is 64k.
- PARAMETER** replicate ||| BOOLEAN — Should the copied file also be replicated on the destination? Defaults to FALSE
- PARAMETER** destinationGroup ||| VARSTRING — The name of the group to distribute the file across.
- PARAMETER** maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.
- PARAMETER** allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE
- PARAMETER** forcePush ||| BOOLEAN — Should the copy process be executed on the source nodes (push) or on the destination nodes (pull)? Default is to pull.
- PARAMETER** preservecompression ||| BOOLEAN — No Doc
- RETURN** VARSTRING — The DFU workunit id for the job.

FUNCTION Copy

File \

Copy
<pre>(varstring sourceLogicalName, varstring destinationGroup, varstring destinationLogicalName, varstring sourceDali=", integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean asSuperfile=FALSE, boolean compress=FALSE, boolean forcePush=FALSE, integer4 transferBufferSize=0, boolean preserveCompression=TRUE)</pre>

Same as fCopy, but does not return the DFU Workunit ID.

PARAMETER forcepush ||| BOOLEAN — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

PARAMETER transferbuffersize ||| INTEGER4 — No Doc

PARAMETER sourcelogicalname ||| VARSTRING — No Doc

PARAMETER assuperfile ||| BOOLEAN — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER sourcedali ||| VARSTRING — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER preservecompression ||| BOOLEAN — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

RETURN —

SEE fCopy

FUNCTION fReplicate

File \

varstring	fReplicate
(varstring logicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'))	

Ensures the specified file is replicated to its mirror copies.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER logicalName ||| VARSTRING — The name of the file to replicate.

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION Replicate

File \

Replicate
<pre>(varstring logicalName, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>

Same as fReplicated, but does not return the DFU Workunit ID.

PARAMETER logicalname ||| VARSTRING — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

RETURN —

SEE fReplicate

FUNCTION fRemotePull

File \

varstring	fRemotePull
<pre>(varstring remoteEspFsURL, varstring sourceLogicalName, varstring destinationGroup, varstring destinationLogicalName, integer4 timeOut=-1, integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean asSuperfile=FALSE, boolean forcePush=FALSE, integer4 transferBufferSize=0, boolean wrap=FALSE, boolean compress=FALSE)</pre>	

Copies a distributed file to a distributed file on remote system. Similar to fCopy, except the copy executes remotely. Since the DFU workunit executes on the remote DFU server, the user name authentication must be the same on both systems, and the user must have rights to copy files on both systems.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER wrap ||| BOOLEAN — Should the fileparts be wrapped when copying to a smaller sized cluster? The default is FALSE.

PARAMETER compress ||| BOOLEAN — Whether to compress the new file. Defaults to FALSE.

PARAMETER asSuperfile ||| BOOLEAN — Should the file be copied as a superfile? If TRUE and source is a superfile, then the operation creates a superfile on the target, creating sub-files as needed and only overwriting existing sub-files whose content has changed. If FALSE a single file is created. Defaults to FALSE.

PARAMETER destinationLogicalName ||| VARSTRING — The logical name of the file to create.

PARAMETER sourceLogicalName ||| VARSTRING — The name of the file to despray.

PARAMETER remoteEspFsURL ||| VARSTRING — The url of the remote ESP file copying service.

PARAMETER transferBufferSize ||| INTEGER4 — Overrides the size (in bytes) of the internal buffer used to copy the file. Default is 64k.

PARAMETER replicate ||| BOOLEAN — Should the copied file also be replicated on the destination? Defaults to FALSE

PARAMETER destinationGroup ||| VARSTRING — The name of the group to distribute the file across.

PARAMETER maxConnections ||| INTEGER4 — The maximum number of target nodes to write to concurrently. Defaults to 1.

PARAMETER allowOverwrite ||| BOOLEAN — Is it valid to overwrite an existing file of the same name? Defaults to FALSE

PARAMETER forcePush ||| BOOLEAN — Should the copy process should be executed on the source nodes (push) or on the destination nodes (pull)? Default is to pull.

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION RemotePull

File \

RemotePull
<pre>(varstring remoteEspFsURL, varstring sourceLogicalName, varstring destinationGroup, varstring destinationLogicalName, integer4 timeout=-1, integer4 maxConnections=-1, boolean allowOverwrite=FALSE, boolean replicate=FALSE, boolean asSuperfile=FALSE, boolean forcePush=FALSE, integer4 transferBufferSize=0, boolean wrap=FALSE, boolean compress=FALSE)</pre>

Same as fRemotePull, but does not return the DFU Workunit ID.

PARAMETER forcepush ||| BOOLEAN — No Doc

PARAMETER timeout ||| INTEGER4 — No Doc

PARAMETER transferbuffersize ||| INTEGER4 — No Doc

PARAMETER sourcelogicalname ||| VARSTRING — No Doc

PARAMETER assuperfile ||| BOOLEAN — No Doc

PARAMETER maxconnections ||| INTEGER4 — No Doc

PARAMETER wrap ||| BOOLEAN — No Doc

PARAMETER compress ||| BOOLEAN — No Doc

PARAMETER allowoverwrite ||| BOOLEAN — No Doc

PARAMETER replicate ||| BOOLEAN — No Doc

PARAMETER remoteespfsurl ||| VARSTRING — No Doc

PARAMETER destinationlogicalname ||| VARSTRING — No Doc

PARAMETER destinationgroup ||| VARSTRING — No Doc

RETURN —

SEE fRemotePull

FUNCTION fMonitorLogicalFileName

File \

varstring	fMonitorLogicalFileName
<pre>(varstring eventToFire, varstring name, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>	

Creates a file monitor job in the DFU Server. If an appropriately named file arrives in this interval it will fire the event with the name of the triggering object as the event subtype (see the EVENT function).

PARAMETER **espServerIpPort** ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER **eventToFire** ||| VARSTRING — The user-defined name of the event to fire when the filename appears. This value is used as the first parameter to the EVENT function.

PARAMETER **name** ||| VARSTRING — The name of the logical file to monitor. This may contain wildcard characters (* and ?)

PARAMETER **shotCount** ||| INTEGER4 — The number of times to generate the event before the monitoring job completes. A value of -1 indicates the monitoring job continues until manually aborted. The default is 1.

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION MonitorLogicalFileName

File \

	MonitorLogicalFileName
<pre>(varstring eventToFire, varstring name, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>	

Same as fMonitorLogicalFileName, but does not return the DFU Workunit ID.

PARAMETER **shotcount** ||| INTEGER4 — No Doc

PARAMETER **eventtofire** ||| VARSTRING — No Doc

PARAMETER **name** ||| VARSTRING — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

RETURN —

SEE fMonitorLogicalFileName

FUNCTION fMonitorFile

File \

varstring	fMonitorFile
<pre>(varstring eventToFire, varstring ip, varstring filename, boolean subDirs=FALSE, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>	

Creates a file monitor job in the DFU Server. If an appropriately named file arrives in this interval it will fire the event with the name of the triggering object as the event subtype (see the EVENT function).

PARAMETER filename ||| VARSTRING — The full path of the file(s) to monitor. This may contain wildcard characters (* and ?)

PARAMETER subDirs ||| BOOLEAN — Whether to include files in sub-directories (when the filename contains wildcards). Defaults to FALSE.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

PARAMETER shotCount ||| INTEGER4 — The number of times to generate the event before the monitoring job completes. A value of -1 indicates the monitoring job continues until manually aborted. The default is 1.

PARAMETER ip ||| VARSTRING — The the IP address for the file to monitor. This may be omitted if the filename parameter contains a complete URL.

PARAMETER eventToFire ||| VARSTRING — The user-defined name of the event to fire when the filename appears. This value is used as the first parameter to the EVENT function.

RETURN VARSTRING — The DFU workunit id for the job.

FUNCTION MonitorFile

File \

	MonitorFile
<pre>(varstring eventToFire, varstring ip, varstring filename, boolean subdirs=FALSE, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>	

Same as fMonitorFile, but does not return the DFU Workunit ID.

PARAMETER filename ||| VARSTRING — No Doc

PARAMETER shotcount ||| INTEGER4 — No Doc

PARAMETER subdirs ||| BOOLEAN — No Doc

PARAMETER ip ||| VARSTRING — No Doc

PARAMETER espserveripport ||| VARSTRING — No Doc

PARAMETER eventtofire ||| VARSTRING — No Doc

RETURN —

SEE fMonitorFile

FUNCTION WaitDfuWorkunit

File \

varstring	WaitDfuWorkunit
<pre>(varstring wuid, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'))</pre>	

Waits for the specified DFU workunit to finish.

PARAMETER timeOut ||| INTEGER4 — The time in ms to wait for the operation to complete. A value of 0 causes the call to return immediately. Defaults to no timeout (-1).

PARAMETER wuid ||| VARSTRING — The dfu wfid to wait for.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

RETURN VARSTRING — A string containing the final status string of the DFU workunit.

FUNCTION AbortDfuWorkunit

File \

AbortDfuWorkunit
(varstring wuid, varstring espServerIpPort=GETENV('ws_fs_server'))

Aborts the specified DFU workunit.

PARAMETER wuid ||| VARSTRING — The dfu wfid to abort.

PARAMETER espServerIpPort ||| VARSTRING — The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

RETURN —

FUNCTION CreateSuperFile

File \

CreateSuperFile
(varstring superName, boolean sequentialParts=FALSE, boolean allowExist=FALSE)

Creates an empty superfile. This function is not included in a superfile transaction.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER sequentialParts ||| BOOLEAN — Whether the sub-files must be sequentially ordered. Default to FALSE.

PARAMETER allowExist ||| BOOLEAN — Indicating whether to post an error if the superfile already exists. If TRUE, no error is posted. Defaults to FALSE.

RETURN —

FUNCTION SuperFileExists

File \

boolean	SuperFileExists
(varstring superName)	

Checks if the specified filename is present in the Distributed File Utility (DFU) and is a SuperFile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

RETURN BOOLEAN — Whether the file exists.

SEE FileExists

FUNCTION DeleteSuperFile

File \

	DeleteSuperFile
(varstring superName, boolean deletesub=FALSE)	

Deletes the superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER deletesub ||| BOOLEAN — No Doc

RETURN —

SEE FileExists

FUNCTION GetSuperFileSubCount

File \

unsigned4	GetSuperFileSubCount
(varstring superName)	

Returns the number of sub-files contained within a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

RETURN UNSIGNED4 — The number of sub-files within the superfile.

FUNCTION GetSuperFileSubName

File \

varstring	GetSuperFileSubName
(varstring superName, unsigned4 fileNum, boolean absPath=FALSE)	

Returns the name of the Nth sub-file within a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER absPath ||| BOOLEAN — Whether to prepend '~' to the name of the resulting logical file name.

PARAMETER fileNum ||| UNSIGNED4 — The 1-based position of the sub-file to return the name of.

RETURN VARSTRING — The logical name of the selected sub-file.

FUNCTION FindSuperFileSubName

File \

unsigned4	FindSuperFileSubName
(varstring superName, varstring subName)	

Returns the position of a file within a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER subName ||| VARSTRING — The logical name of the sub-file.

RETURN UNSIGNED4 — The 1-based position of the sub-file within the superfile.

FUNCTION StartSuperFileTransaction

File \

	StartSuperFileTransaction
()	

Starts a superfile transaction. All superfile operations within the transaction will either be executed atomically or rolled back when the transaction is finished.

RETURN —

FUNCTION AddSuperFile

File \

	AddSuperFile
(varstring superName, varstring subName, unsigned4 atPos=0, boolean addContents=FALSE, boolean strict=FALSE)	

Adds a file to a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER strict ||| BOOLEAN — Check addContents only if subName is a superfile, and ensure superfiles exist.

PARAMETER addContents ||| BOOLEAN — Controls whether adding a superfile adds the superfile, or its contents. Defaults to FALSE (do not expand).

PARAMETER subName ||| VARSTRING — The name of the logical file to add.

PARAMETER atPos ||| UNSIGNED4 — The position to add the sub-file, or 0 to append. Defaults to 0.

RETURN —

FUNCTION RemoveSuperFile

File \

RemoveSuperFile
(varstring superName, varstring subName, boolean del=FALSE, boolean removeContents=FALSE)

Removes a sub-file from a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER removeContents ||| BOOLEAN — Controls whether the contents of a sub-file which is a superfile should be recursively removed. Defaults to FALSE.

PARAMETER del ||| BOOLEAN — Indicates whether the sub-file should also be removed from the disk. Defaults to FALSE.

PARAMETER subName ||| VARSTRING — The name of the sub-file to remove.

RETURN —

FUNCTION ClearSuperFile

File \

ClearSuperFile
(varstring superName, boolean del=FALSE)

Removes all sub-files from a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER del ||| BOOLEAN — Indicates whether the sub-files should also be removed from the disk. Defaults to FALSE.

RETURN —

FUNCTION RemoveOwnedSubFiles

File \

RemoveOwnedSubFiles
(varstring superName, boolean del=FALSE)

Removes all soley-owned sub-files from a superfile. If a sub-file is also contained within another superfile then it is retained.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER del ||| BOOLEAN — No Doc

RETURN —

FUNCTION DeleteOwnedSubFiles

File \

DeleteOwnedSubFiles
(varstring superName)

Legacy version of RemoveOwnedSubFiles which was incorrectly named in a previous version.

PARAMETER supername ||| VARSTRING — No Doc

RETURN —

SEE RemoveOwnedSubFiles

FUNCTION SwapSuperFile

File \

SwapSuperFile
(varstring superName1, varstring superName2)

Swap the contents of two superfiles.

PARAMETER superName1 ||| VARSTRING — The logical name of the first superfile.

PARAMETER superName2 ||| VARSTRING — The logical name of the second superfile.

RETURN —

FUNCTION ReplaceSuperFile

File \

ReplaceSuperFile
(varstring superName, varstring oldSubFile, varstring newSubFile)

Removes a sub-file from a superfile and replaces it with another.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER oldSubFile ||| VARSTRING — The logical name of the sub-file to remove.

PARAMETER newSubFile ||| VARSTRING — The logical name of the sub-file to replace within the superfile.

RETURN —

FUNCTION FinishSuperFileTransaction

File \

FinishSuperFileTransaction
(boolean rollback=FALSE)

Finishes a superfile transaction. This executes all the operations since the matching StartSuperFileTransaction(). If there are any errors, then all of the operations are rolled back.

PARAMETER rollback ||| BOOLEAN — No Doc

RETURN —

FUNCTION SuperFileContents

File \

<code>dataset(FsLogicalFileNameRecord)</code>	SuperFileContents
<code>(varstring superName, boolean recurse=FALSE)</code>	

Returns the list of sub-files contained within a superfile.

PARAMETER superName ||| VARSTRING — The logical name of the superfile.

PARAMETER recurse ||| BOOLEAN — Should the contents of child-superfiles be expanded. Default is FALSE.

RETURN TABLE (FsLogicalFileNameRecord) — A dataset containing the names of the sub-files.

FUNCTION LogicalFileSuperOwners

File \

<code>dataset(FsLogicalFileNameRecord)</code>	LogicalFileSuperOwners
<code>(varstring name)</code>	

Returns the list of superfiles that a logical file is contained within.

PARAMETER name ||| VARSTRING — The name of the logical file.

RETURN TABLE (FsLogicalFileNameRecord) — A dataset containing the names of the superfiles.

FUNCTION LogicalFileSuperSubList

File \

<code>dataset(FsLogicalSuperSubRecord)</code>	LogicalFileSuperSubList
<code>()</code>	

Returns the list of all the superfiles in the system and their component sub-files.

RETURN **TABLE** (**FsLogicalSuperSubRecord**) — A dataset containing pairs of superName,subName for each component file.

FUNCTION **fPromoteSuperFileList**

File \

varstring	fPromoteSuperFileList
<code>(set of varstring superNames, varstring addHead="", boolean delTail=FALSE, boolean createOnlyOne=FALSE, boolean reverse=FALSE)</code>	

Moves the sub-files from the first entry in the list of superfiles to the next in the list, repeating the process through the list of superfiles.

PARAMETER **superNames** ||| SET (VARSTRING) — A set of the names of the superfiles to act on. Any that do not exist will be created. The contents of each superfile will be moved to the next in the list.

PARAMETER **createOnlyOne** ||| BOOLEAN — Specifies whether to only create a single superfile (truncate the list at the first non-existent superfile). The default is FALSE.

PARAMETER **delTail** ||| BOOLEAN — Indicates whether to physically delete the contents moved out of the last superfile. The default is FALSE.

PARAMETER **addHead** ||| VARSTRING — A string containing a comma-delimited list of logical file names to add to the first superfile after the promotion process is complete. Defaults to ”.

PARAMETER **reverse** ||| BOOLEAN — Reverse the order of processing the superfiles list, effectively ‘demoting’ instead of ‘promoting’ the sub-files. The default is FALSE.

RETURN **VARSTRING** — A string containing a comma separated list of the previous sub-file contents of the emptied superfile.

FUNCTION PromoteSuperFileList

File \

PromoteSuperFileList
(set of varstring superNames, varstring addHead="", boolean delTail=FALSE, boolean createOnlyOne=FALSE, boolean reverse=FALSE)

Same as fPromoteSuperFileList, but does not return the DFU Workunit ID.

PARAMETER deltail ||| BOOLEAN — No Doc

PARAMETER reverse ||| BOOLEAN — No Doc

PARAMETER supernames ||| SET (VARSTRING) — No Doc

PARAMETER createonlyone ||| BOOLEAN — No Doc

PARAMETER addhead ||| VARSTRING — No Doc

RETURN —

SEE fPromoteSuperFileList

math

[Go Up](#)

DESCRIPTIONS

MODULE Math

	Math
--	------

No Documentation Found

Children

1. [Infinity](#) : Return a real "infinity" value
2. [NaN](#) : Return a non-signalling NaN (Not a Number)value
3. [isInfinite](#) : Return whether a real value is infinite (positive or negative)
4. [isNaN](#) : Return whether a real value is a NaN (not a number) value
5. [isFinite](#) : Return whether a real value is a valid value (neither infinite not NaN)
6. [FMod](#) : Returns the floating-point remainder of numer/denom (rounded towards zero)
7. [FMatch](#) : Returns whether two floating point values are the same, within margin of error epsilon

ATTRIBUTE Infinity

[Math](#) \

REAL8	Infinity
--------------	----------

Return a real "infinity" value.

RETURN REAL8 —

ATTRIBUTE NaN

Math \

REAL8	NaN
-------	-----

Return a non-signalling NaN (Not a Number) value.

RETURN REAL8 —

FUNCTION isInfinite

Math \

BOOLEAN	isInfinite
(REAL8 val)	

Return whether a real value is infinite (positive or negative).

PARAMETER val ||| REAL8 — The value to test.

RETURN BOOLEAN —

FUNCTION isNaN

Math \

BOOLEAN	isNaN
(REAL8 val)	

Return whether a real value is a NaN (not a number) value.

PARAMETER val ||| REAL8 — The value to test.

RETURN BOOLEAN —

FUNCTION isFinite

Math \

BOOLEAN	isFinite
(REAL8 val)	

Return whether a real value is a valid value (neither infinite not NaN).

PARAMETER val ||| REAL8 — The value to test.

RETURN BOOLEAN —

FUNCTION FMod

Math \

REAL8	FMod
(REAL8 numer, REAL8 denom)	

Returns the floating-point remainder of numer/denom (rounded towards zero). If denom is zero, the result depends on the -fdivideByZero flag: 'zero' or unset: return zero. 'nan': return a non-signalling NaN value 'fail': throw an exception

PARAMETER numer ||| REAL8 — The numerator.

PARAMETER denom ||| REAL8 — The denominator.

RETURN REAL8 —

FUNCTION FMatch

Math \

BOOLEAN	FMatch
(REAL8 a, REAL8 b, REAL8 epsilon=0.0)	

Returns whether two floating point values are the same, within margin of error epsilon.

PARAMETER b ||| REAL8 — The second value.

PARAMETER a ||| REAL8 — The first value.

PARAMETER epsilon ||| REAL8 — The allowable margin of error.

RETURN BOOLEAN —

Metaphone

[Go Up](#)

IMPORTS

`lib__metaphone` |

DESCRIPTIONS

MODULE Metaphone

	Metaphone
--	-----------

No Documentation Found

Children

1. [primary](#) : Returns the primary metaphone value
2. [secondary](#) : Returns the secondary metaphone value
3. [double](#) : Returns the double metaphone value (primary and secondary concatenated)

FUNCTION primary

[Metaphone](#) \

String	primary
(STRING src)	

Returns the primary metaphone value

PARAMETER src ||| STRING — The string whose metaphone is to be calculated.

RETURN STRING —

SEE http://en.wikipedia.org/wiki/Metaphone#Double_Metaphone

FUNCTION secondary

Metaphone \

String	secondary
(STRING src)	

Returns the secondary metaphone value

PARAMETER src ||| STRING — The string whose metaphone is to be calculated.

RETURN STRING —

SEE http://en.wikipedia.org/wiki/Metaphone#Double_Metaphone

FUNCTION double

Metaphone \

String	double
(STRING src)	

Returns the double metaphone value (primary and secondary concatenated)

PARAMETER src ||| STRING — The string whose metphone is to be calculated.

RETURN STRING —

SEE http://en.wikipedia.org/wiki/Metaphone#Double__Metaphone

str

[Go Up](#)

IMPORTS

lib_stringlib |

DESCRIPTIONS

MODULE Str

Str

No Documentation Found

Children

1. [CompareIgnoreCase](#) : Compares the two strings case insensitively
2. [EqualIgnoreCase](#) : Tests whether the two strings are identical ignoring differences in case
3. [Find](#) : Returns the character position of the nth match of the search string with the first string
4. [FindCount](#) : Returns the number of occurrences of the second string within the first string
5. [WildMatch](#) : Tests if the search string matches the pattern
6. [Contains](#) : Tests if the search string contains each of the characters in the pattern
7. [FilterOut](#) : Returns the first string with all characters within the second string removed
8. [Filter](#) : Returns the first string with all characters not within the second string removed
9. [SubstituteIncluded](#) : Returns the source string with the replacement character substituted for all characters included in the filter string

10. [SubstituteExcluded](#) : Returns the source string with the replacement character substituted for all characters not included in the filter string
11. [Translate](#) : Returns the source string with the all characters that match characters in the search string replaced with the character at the corresponding position in the replacement string
12. [ToLowerCase](#) : Returns the argument string with all upper case characters converted to lower case
13. [ToUpperCase](#) : Return the argument string with all lower case characters converted to upper case
14. [ToCapitalCase](#) : Returns the argument string with the first letter of each word in upper case and all other letters left as-is
15. [ToTitleCase](#) : Returns the argument string with the first letter of each word in upper case and all other letters lower case
16. [Reverse](#) : Returns the argument string with all characters in reverse order
17. [FindReplace](#) : Returns the source string with the replacement string substituted for all instances of the search string
18. [Extract](#) : Returns the nth element from a comma separated string
19. [CleanSpaces](#) : Returns the source string with all instances of multiple adjacent space characters (2 or more spaces together) reduced to a single space character
20. [StartsWith](#) : Returns true if the prefix string matches the leading characters in the source string
21. [EndsWith](#) : Returns true if the suffix string matches the trailing characters in the source string
22. [RemoveSuffix](#) : Removes the suffix from the search string, if present, and returns the result
23. [ExtractMultiple](#) : Returns a string containing a list of elements from a comma separated string
24. [CountWords](#) : Returns the number of words that the string contains
25. [SplitWords](#) : Returns the list of words extracted from the string
26. [CombineWords](#) : Returns the list of words extracted from the string
27. [EditDistance](#) : Returns the minimum edit distance between the two strings
28. [EditDistanceWithinRadius](#) : Returns true if the minimum edit distance between the two strings is within a specific range
29. [WordCount](#) : Returns the number of words in the string
30. [GetNthWord](#) : Returns the n-th word from the string
31. [ExcludeFirstWord](#) : Returns everything except the first word from the string
32. [ExcludeLastWord](#) : Returns everything except the last word from the string
33. [ExcludeNthWord](#) : Returns everything except the nth word from the string
34. [FindWord](#) : Tests if the search string contains the supplied word as a whole word
35. [Repeat](#) : No Documentation Found

- 36. [ToHexPairs](#) : No Documentation Found
 - 37. [FromHexPairs](#) : No Documentation Found
 - 38. [EncodeBase64](#) : No Documentation Found
 - 39. [DecodeBase64](#) : No Documentation Found
-

FUNCTION CompareIgnoreCase

Str \

INTEGER4	CompareIgnoreCase
(STRING src1, STRING src2)	

Compares the two strings case insensitively. Returns a negative integer, zero, or a positive integer according to whether the first string is less than, equal to, or greater than the second.

PARAMETER src2 ||| STRING — The second string to be compared.

PARAMETER src1 ||| STRING — The first string to be compared.

RETURN INTEGER4 —

SEE Str.EqualIgnoreCase

FUNCTION EqualIgnoreCase

Str \

BOOLEAN	EqualIgnoreCase
(STRING src1, STRING src2)	

Tests whether the two strings are identical ignoring differences in case.

PARAMETER src2 ||| STRING — The second string to be compared.

PARAMETER src1 ||| STRING — The first string to be compared.

RETURN BOOLEAN —

SEE Str.CompareIgnoreCase

FUNCTION Find

Str \

UNSIGNED4	Find
(STRING src, STRING sought, UNSIGNED4 instance = 1)	

Returns the character position of the nth match of the search string with the first string. If no match is found the attribute returns 0. If an instance is omitted the position of the first instance is returned.

PARAMETER sought ||| STRING — The string being sought.

PARAMETER src ||| STRING — The string that is searched

PARAMETER instance ||| UNSIGNED4 — Which match instance are we interested in?

RETURN UNSIGNED4 —

FUNCTION FindCount

Str \

UNSIGNED4	FindCount
(STRING src, STRING sought)	

Returns the number of occurrences of the second string within the first string.

PARAMETER sought ||| STRING — The string being sought.

PARAMETER src ||| STRING — The string that is searched

RETURN UNSIGNED4 —

FUNCTION WildMatch

Str \

BOOLEAN	WildMatch
(STRING <u>src</u> , STRING <u>_pattern</u> , BOOLEAN <u>ignore_case</u>)	

Tests if the search string matches the pattern. The pattern can contain wildcards '?' (single character) and '*' (multiple character).

PARAMETER pattern ||| — The pattern to match against.

PARAMETER src ||| STRING — The string that is being tested.

PARAMETER ignore_case ||| BOOLEAN — Whether to ignore differences in case between characters

PARAMETER _pattern ||| STRING — No Doc

RETURN BOOLEAN —

FUNCTION Contains

Str \

BOOLEAN	Contains
(STRING <u>src</u> , STRING <u>_pattern</u> , BOOLEAN <u>ignore_case</u>)	

Tests if the search string contains each of the characters in the pattern. If the pattern contains duplicate characters those characters will match once for each occurrence in the pattern.

PARAMETER pattern ||| — The pattern to match against.

PARAMETER src ||| STRING — The string that is being tested.

PARAMETER ignore__case ||| BOOLEAN — Whether to ignore differences in case between characters

PARAMETER __pattern ||| STRING — No Doc

RETURN BOOLEAN —

FUNCTION FilterOut

Str \

STRING	FilterOut
(STRING src, STRING filter)	

Returns the first string with all characters within the second string removed.

PARAMETER filter ||| STRING — The string containing the set of characters to be excluded.

PARAMETER src ||| STRING — The string that is being tested.

RETURN STRING —

SEE Str.Filter

FUNCTION Filter

Str \

STRING	Filter
(STRING src, STRING filter)	

Returns the first string with all characters not within the second string removed.

PARAMETER filter ||| STRING — The string containing the set of characters to be included.

PARAMETER src ||| STRING — The string that is being tested.

RETURN STRING —

SEE Str.FilterOut

FUNCTION SubstituteIncluded

Str \

STRING	SubstituteIncluded
(STRING src, STRING filter, STRING1 replace_char)	

Returns the source string with the replacement character substituted for all characters included in the filter string. MORE: Should this be a general string substitution?

PARAMETER filter ||| STRING — The string containing the set of characters to be included.

PARAMETER src ||| STRING — The string that is being tested.

PARAMETER replace__char ||| STRING1 — The character to be substituted into the result.

RETURN STRING —

SEE Std.Str.Translate, Std.Str.SubstituteExcluded

FUNCTION SubstituteExcluded

Str \

STRING	SubstituteExcluded
(STRING src, STRING filter, STRING1 replace_char)	

Returns the source string with the replacement character substituted for all characters not included in the filter string. MORE: Should this be a general string substitution?

PARAMETER filter ||| STRING — The string containing the set of characters to be included.

PARAMETER src ||| STRING — The string that is being tested.

PARAMETER replace_char ||| STRING1 — The character to be substituted into the result.

RETURN STRING —

SEE Std.Str.SubstituteIncluded

FUNCTION Translate

Str \

STRING	Translate
(STRING src, STRING search, STRING replacement)	

Returns the source string with the all characters that match characters in the search string replaced with the character at the corresponding position in the replacement string.

PARAMETER search ||| STRING — The string containing the set of characters to be included.

PARAMETER replacement ||| STRING — The string containing the characters to act as replacements.

PARAMETER src ||| STRING — The string that is being tested.

RETURN STRING —

SEE Std.Str.SubstituteIncluded

FUNCTION ToLowerCase

Str \

STRING	ToLowerCase
(STRING src)	

Returns the argument string with all upper case characters converted to lower case.

PARAMETER src ||| STRING — The string that is being converted.

RETURN STRING —

FUNCTION ToUpperCase

Str \

STRING	ToUpperCase
(STRING src)	

Return the argument string with all lower case characters converted to upper case.

PARAMETER src ||| STRING — The string that is being converted.

RETURN STRING —

FUNCTION ToCapitalCase

Str \

STRING	ToCapitalCase
(STRING src)	

Returns the argument string with the first letter of each word in upper case and all other letters left as-is. A contiguous sequence of alphanumeric characters is treated as a word.

PARAMETER src ||| STRING — The string that is being converted.

RETURN STRING —

FUNCTION ToTitleCase

Str \

STRING	ToTitleCase
(STRING src)	

Returns the argument string with the first letter of each word in upper case and all other letters lower case. A contiguous sequence of alphanumeric characters is treated as a word.

PARAMETER src ||| STRING — The string that is being converted.

RETURN STRING —

FUNCTION Reverse

Str \

STRING	Reverse
(STRING src)	

Returns the argument string with all characters in reverse order. Note the argument is not TRIMMED before it is reversed.

PARAMETER src ||| STRING — The string that is being reversed.

RETURN STRING —

FUNCTION FindReplace

Str \

STRING	FindReplace
(STRING src, STRING sought, STRING replacement)	

Returns the source string with the replacement string substituted for all instances of the search string.

PARAMETER sought ||| STRING — The string to be replaced.

PARAMETER replacement ||| STRING — The string to be substituted into the result.

PARAMETER src ||| STRING — The string that is being transformed.

RETURN STRING —

FUNCTION Extract

Str \

STRING	Extract
(STRING src, UNSIGNED4 instance)	

Returns the nth element from a comma separated string.

PARAMETER src ||| STRING — The string containing the comma separated list.

PARAMETER instance ||| UNSIGNED4 — Which item to select from the list.

RETURN STRING —

FUNCTION CleanSpaces

Str \

STRING	CleanSpaces
(STRING src)	

Returns the source string with all instances of multiple adjacent space characters (2 or more spaces together) reduced to a single space character. Leading and trailing spaces are removed, and tab characters are converted to spaces.

PARAMETER src ||| STRING — The string to be cleaned.

RETURN STRING —

FUNCTION StartsWith

Str \

BOOLEAN	StartsWith
(STRING src, STRING prefix)	

Returns true if the prefix string matches the leading characters in the source string. Trailing spaces are stripped from the prefix before matching. // x.myString.StartsWith('x') as an alternative syntax would be even better

PARAMETER src ||| STRING — The string being searched in.

PARAMETER prefix ||| STRING — The prefix to search for.

RETURN BOOLEAN —

FUNCTION EndsWith

Str \

BOOLEAN	EndsWith
(STRING src, STRING suffix)	

Returns true if the suffix string matches the trailing characters in the source string. Trailing spaces are stripped from both strings before matching.

PARAMETER suffix ||| STRING — The prefix to search for.

PARAMETER src ||| STRING — The string being searched in.

RETURN BOOLEAN —

FUNCTION RemoveSuffix

Str \

STRING	RemoveSuffix
(STRING src, STRING suffix)	

Removes the suffix from the search string, if present, and returns the result. Trailing spaces are stripped from both strings before matching.

PARAMETER suffix ||| STRING — The prefix to search for.

PARAMETER src ||| STRING — The string being searched in.

RETURN STRING —

FUNCTION ExtractMultiple

Str \

STRING	ExtractMultiple
(STRING src, UNSIGNED8 mask)	

Returns a string containing a list of elements from a comma separated string.

PARAMETER mask ||| UNSIGNED8 — A bitmask of which elements should be included. Bit 0 is item1, bit1 item 2 etc.

PARAMETER src ||| STRING — The string containing the comma separated list.

RETURN STRING —

FUNCTION CountWords

Str \

UNSIGNED4	CountWords
(STRING src, STRING separator, BOOLEAN allow_blank = FALSE)	

Returns the number of words that the string contains. Words are separated by one or more separator strings. No spaces are stripped from either string before matching.

PARAMETER allow_blank ||| BOOLEAN — Indicates if empty/blank string items are included in the results.

PARAMETER src ||| STRING — The string being searched in.

PARAMETER separator ||| STRING — The string used to separate words

RETURN UNSIGNED4 —

FUNCTION SplitWords

Str \

SET OF STRING	SplitWords
(STRING src, STRING separator, BOOLEAN allow_blank = FALSE)	

Returns the list of words extracted from the string. Words are separated by one or more separator strings. No spaces are stripped from either string before matching.

PARAMETER allow_blank ||| BOOLEAN — Indicates if empty/blank string items are included in the results.

PARAMETER src ||| STRING — The string being searched in.

PARAMETER separator ||| STRING — The string used to separate words

RETURN SET (STRING) —

FUNCTION CombineWords

Str \

STRING	CombineWords
(SET OF STRING words, STRING separator)	

Returns the list of words extracted from the string. Words are separated by one or more separator strings. No spaces are stripped from either string before matching.

PARAMETER words ||| SET (STRING) — The set of strings to be combined.

PARAMETER separator ||| STRING — The string used to separate words.

RETURN STRING —

FUNCTION EditDistance

Str \

UNSIGNED4	EditDistance
(STRING _left, STRING _right)	

Returns the minimum edit distance between the two strings. An insert change or delete counts as a single edit. The two strings are trimmed before comparing.

PARAMETER __right ||| STRING — The second string to be compared.

PARAMETER __left ||| STRING — The first string to be compared.

RETURN UNSIGNED4 — The minimum edit distance between the two strings.

FUNCTION EditDistanceWithinRadius

Str \

BOOLEAN	EditDistanceWithinRadius
(STRING _left, STRING _right, UNSIGNED4 radius)	

Returns true if the minimum edit distance between the two strings is within a specific range. The two strings are trimmed before comparing.

PARAMETER radius ||| UNSIGNED4 — The maximum edit distance that is acceptable.

PARAMETER __right ||| STRING — The second string to be compared.

PARAMETER __left ||| STRING — The first string to be compared.

RETURN BOOLEAN — Whether or not the two strings are within the given specified edit distance.

FUNCTION WordCount

Str \

UNSIGNED4	WordCount
(STRING text)	

Returns the number of words in the string. Words are separated by one or more spaces.

PARAMETER text ||| STRING — The string to be broken into words.

RETURN UNSIGNED4 — The number of words in the string.

FUNCTION GetNthWord

Str \

STRING	GetNthWord
(STRING text, UNSIGNED4 n)	

Returns the n-th word from the string. Words are separated by one or more spaces.

PARAMETER text ||| STRING — The string to be broken into words.

PARAMETER n ||| UNSIGNED4 — Which word should be returned from the function.

RETURN STRING — The number of words in the string.

FUNCTION ExcludeFirstWord

Str \

	ExcludeFirstWord
(STRING text)	

Returns everything except the first word from the string. Words are separated by one or more whitespace characters. Whitespace before and after the first word is also removed.

PARAMETER text ||| STRING — The string to be broken into words.

RETURN STRING — The string excluding the first word.

FUNCTION ExcludeLastWord

Str \

	ExcludeLastWord
(STRING text)	

Returns everything except the last word from the string. Words are separated by one or more whitespace characters. Whitespace after a word is removed with the word and leading whitespace is removed with the first word.

PARAMETER text ||| STRING — The string to be broken into words.

RETURN STRING — The string excluding the last word.

FUNCTION ExcludeNthWord

Str \

	ExcludeNthWord
(STRING text, UNSIGNED2 n)	

Returns everything except the nth word from the string. Words are separated by one or more whitespace characters. Whitespace after a word is removed with the word and leading whitespace is removed with the first word.

PARAMETER text ||| STRING — The string to be broken into words.

PARAMETER n ||| UNSIGNED2 — Which word should be returned from the function.

RETURN **STRING** — The string excluding the nth word.

FUNCTION FindWord

Str \

BOOLEAN	FindWord
(STRING src, STRING word, BOOLEAN ignore_case=FALSE)	

Tests if the search string contains the supplied word as a whole word.

PARAMETER word ||| STRING — The word to be searched for.

PARAMETER src ||| STRING — The string that is being tested.

PARAMETER ignore_case ||| BOOLEAN — Whether to ignore differences in case between characters.

RETURN **BOOLEAN** —

FUNCTION Repeat

Str \

STRING	Repeat
(STRING text, UNSIGNED4 n)	

No Documentation Found

PARAMETER text ||| STRING — No Doc

PARAMETER n ||| UNSIGNED4 — No Doc

RETURN **STRING** —

FUNCTION ToHexPairs

Str \

STRING	ToHexPairs
(DATA value)	

No Documentation Found

PARAMETER value ||| DATA — No Doc

RETURN STRING —

FUNCTION FromHexPairs

Str \

DATA	FromHexPairs
(STRING hex_pairs)	

No Documentation Found

PARAMETER hex_pairs ||| STRING — No Doc

RETURN DATA —

FUNCTION EncodeBase64

Str \

STRING	EncodeBase64
(DATA value)	

No Documentation Found

PARAMETER value ||| DATA — No Doc

RETURN STRING —

FUNCTION DecodeBase64

Str \

DATA	DecodeBase64
(STRING value)	

No Documentation Found

PARAMETER value ||| STRING — No Doc

RETURN DATA —

Uni

[Go Up](#)

IMPORTS

lib__unicodelib |

DESCRIPTIONS

MODULE Uni

	Uni
--	-----

No Documentation Found

Children

1. [FilterOut](#) : Returns the first string with all characters within the second string removed
2. [Filter](#) : Returns the first string with all characters not within the second string removed
3. [SubstituteIncluded](#) : Returns the source string with the replacement character substituted for all characters included in the filter string
4. [SubstituteExcluded](#) : Returns the source string with the replacement character substituted for all characters not included in the filter string
5. [Find](#) : Returns the character position of the nth match of the search string with the first string
6. [FindWord](#) : Tests if the search string contains the supplied word as a whole word
7. [LocaleFind](#) : Returns the character position of the nth match of the search string with the first string

8. [LocaleFindAtStrength](#) : Returns the character position of the nth match of the search string with the first string
9. [Extract](#) : Returns the nth element from a comma separated string
10. [ToLowerCase](#) : Returns the argument string with all upper case characters converted to lower case
11. [ToUpperCase](#) : Return the argument string with all lower case characters converted to upper case
12. [ToTitleCase](#) : Returns the upper case variant of the string using the rules for a particular locale
13. [LocaleToLowerCase](#) : Returns the lower case variant of the string using the rules for a particular locale
14. [LocaleToUpperCase](#) : Returns the upper case variant of the string using the rules for a particular locale
15. [LocaleToTitleCase](#) : Returns the upper case variant of the string using the rules for a particular locale
16. [CompareIgnoreCase](#) : Compares the two strings case insensitively
17. [CompareAtStrength](#) : Compares the two strings case insensitively
18. [LocaleCompareIgnoreCase](#) : Compares the two strings case insensitively
19. [LocaleCompareAtStrength](#) : Compares the two strings case insensitively
20. [Reverse](#) : Returns the argument string with all characters in reverse order
21. [FindReplace](#) : Returns the source string with the replacement string substituted for all instances of the search string
22. [LocaleFindReplace](#) : Returns the source string with the replacement string substituted for all instances of the search string
23. [LocaleFindAtStrengthReplace](#) : Returns the source string with the replacement string substituted for all instances of the search string
24. [CleanAccents](#) : Returns the source string with all accented characters replaced with unaccented
25. [CleanSpaces](#) : Returns the source string with all instances of multiple adjacent space characters (2 or more spaces together) reduced to a single space character
26. [WildMatch](#) : Tests if the search string matches the pattern
27. [Contains](#) : Tests if the search string contains each of the characters in the pattern
28. [EditDistance](#) : Returns the minimum edit distance between the two strings
29. [EditDistanceWithinRadius](#) : Returns true if the minimum edit distance between the two strings is within a specific range
30. [WordCount](#) : Returns the number of words in the string
31. [GetNthWord](#) : Returns the n-th word from the string

FUNCTION FilterOut

Uni \

unicode	FilterOut
(unicode src, unicode filter)	

Returns the first string with all characters within the second string removed.

PARAMETER filter ||| UNICODE — The string containing the set of characters to be excluded.

PARAMETER src ||| UNICODE — The string that is being tested.

RETURN UNICODE —

SEE Std.Uni.Filter

FUNCTION Filter

Uni \

unicode	Filter
(unicode src, unicode filter)	

Returns the first string with all characters not within the second string removed.

PARAMETER filter ||| UNICODE — The string containing the set of characters to be included.

PARAMETER src ||| UNICODE — The string that is being tested.

RETURN UNICODE —

SEE Std.Uni.FilterOut

FUNCTION **SubstituteIncluded**

Uni \

unicode	SubstituteIncluded
(unicode src, unicode filter, unicode replace_char)	

Returns the source string with the replacement character substituted for all characters included in the filter string. MORE: Should this be a general string substitution?

PARAMETER filter ||| UNICODE — The string containing the set of characters to be included.

PARAMETER src ||| UNICODE — The string that is being tested.

PARAMETER replace__char ||| UNICODE — The character to be substituted into the result.

RETURN UNICODE —

SEE Std.Uni.SubstituteOut

FUNCTION **SubstituteExcluded**

Uni \

unicode	SubstituteExcluded
(unicode src, unicode filter, unicode replace_char)	

Returns the source string with the replacement character substituted for all characters not included in the filter string. MORE: Should this be a general string substitution?

PARAMETER filter ||| UNICODE — The string containing the set of characters to be included.

PARAMETER src ||| UNICODE — The string that is being tested.

PARAMETER replace__char ||| UNICODE — The character to be substituted into the result.

RETURN UNICODE —

SEE Std.Uni.SubstituteIncluded

FUNCTION Find

Uni \

UNSIGNED4	Find
(unicode src, unicode sought, unsigned4 instance)	

Returns the character position of the nth match of the search string with the first string. If no match is found the attribute returns 0. If an instance is omitted the position of the first instance is returned.

PARAMETER sought ||| UNICODE — The string being sought.

PARAMETER src ||| UNICODE — The string that is searched

PARAMETER instance ||| UNSIGNED4 — Which match instance are we interested in?

RETURN UNSIGNED4 —

FUNCTION FindWord

Uni \

BOOLEAN	FindWord
(UNICODE src, UNICODE word, BOOLEAN ignore_case=FALSE)	

Tests if the search string contains the supplied word as a whole word.

PARAMETER word ||| UNICODE — The word to be searched for.

PARAMETER src ||| UNICODE — The string that is being tested.

PARAMETER ignore_case ||| BOOLEAN — Whether to ignore differences in case between characters.

RETURN BOOLEAN —

FUNCTION LocaleFind

Uni \

UNSIGNED4	LocaleFind
(unicode src, unicode sought, unsigned4 instance, varstring locale_name)	

Returns the character position of the nth match of the search string with the first string. If no match is found the attribute returns 0. If an instance is omitted the position of the first instance is returned.

PARAMETER sought ||| UNICODE — The string being sought.

PARAMETER src ||| UNICODE — The string that is searched

PARAMETER instance ||| UNSIGNED4 — Which match instance are we interested in?

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN UNSIGNED4 —

FUNCTION LocaleFindAtStrength

Uni \

UNSIGNED4	LocaleFindAtStrength
(unicode src, unicode tofind, unsigned4 instance, varstring locale_name, integer1 strength)	

Returns the character position of the nth match of the search string with the first string. If no match is found the attribute returns 0. If an instance is omitted the position of the first instance is returned.

PARAMETER sought ||| — The string being sought.

PARAMETER strength ||| INTEGER1 — The strength of the comparison 1 ignores accents and case, differentiating only between letters 2 ignores case but differentiates between accents. 3 differentiates between accents and case but ignores e.g. differences between Hiragana and Katakana 4 differentiates between accents and case and e.g. Hiragana/Katakana, but ignores e.g. Hebrew cantillation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

PARAMETER src ||| UNICODE — The string that is searched

PARAMETER instance ||| UNSIGNED4 — Which match instance are we interested in?

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

PARAMETER tofind ||| UNICODE — No Doc

RETURN UNSIGNED4 —

FUNCTION Extract

Uni \

unicode	Extract
(unicode src, unsigned4 instance)	

Returns the nth element from a comma separated string.

PARAMETER src ||| UNICODE — The string containing the comma separated list.

PARAMETER instance ||| UNSIGNED4 — Which item to select from the list.

RETURN UNICODE —

FUNCTION ToLowerCase

Uni \

unicode	ToLowerCase
(unicode src)	

Returns the argument string with all upper case characters converted to lower case.

PARAMETER src ||| UNICODE — The string that is being converted.

RETURN UNICODE —

FUNCTION ToUpperCase

Uni \

unicode	ToUpperCase
(unicode src)	

Return the argument string with all lower case characters converted to upper case.

PARAMETER src ||| UNICODE — The string that is being converted.

RETURN UNICODE —

FUNCTION ToTitleCase

Uni \

unicode	ToTitleCase
(unicode src)	

Returns the upper case variant of the string using the rules for a particular locale.

PARAMETER src ||| UNICODE — The string that is being converted.

PARAMETER locale__name ||| — The locale to use for the comparison

RETURN UNICODE —

FUNCTION LocaleToLowerCase

Uni \

unicode	LocaleToLowerCase
(unicode src, varstring locale_name)	

Returns the lower case variant of the string using the rules for a particular locale.

PARAMETER src ||| UNICODE — The string that is being converted.

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN UNICODE —

FUNCTION LocaleToUpperCase

Uni \

unicode	LocaleToUpperCase
(unicode src, varstring locale_name)	

Returns the upper case variant of the string using the rules for a particular locale.

PARAMETER src ||| UNICODE — The string that is being converted.

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN UNICODE —

FUNCTION LocaleToTitleCase

Uni \

unicode	LocaleToTitleCase
(unicode src, varstring locale_name)	

Returns the upper case variant of the string using the rules for a particular locale.

PARAMETER src ||| UNICODE — The string that is being converted.

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN UNICODE —

FUNCTION CompareIgnoreCase

Uni \

integer4	CompareIgnoreCase
(unicode src1, unicode src2)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

PARAMETER src2 ||| UNICODE — The second string to be compared.

PARAMETER src1 ||| UNICODE — The first string to be compared.

RETURN INTEGER4 —

SEE Std.Uni.CompareAtStrength

FUNCTION CompareAtStrength

Uni \

integer4	CompareAtStrength
(unicode src1, unicode src2, integer1 strength)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

PARAMETER src2 ||| UNICODE — The second string to be compared.

PARAMETER src1 ||| UNICODE — The first string to be compared.

PARAMETER strength ||| INTEGER1 — The strength of the comparison 1 ignores accents and case, differentiating only between letters 2 ignores case but differentiates between accents. 3 differentiates between accents and case but ignores e.g. differences between Hiragana and Katakana 4 differentiates between accents and case and e.g. Hiragana/Katakana, but ignores e.g. Hebrew cantillation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

RETURN INTEGER4 —

SEE Std.Uni.CompareAtStrength

FUNCTION LocaleCompareIgnoreCase

Uni \

<code>integer4</code>	<code>LocaleCompareIgnoreCase</code>
<code>(unicode src1, unicode src2, varstring locale_name)</code>	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

PARAMETER src2 ||| UNICODE — The second string to be compared.

PARAMETER src1 ||| UNICODE — The first string to be compared.

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN INTEGER4 —

SEE Std.Uni.CompareAtStrength

FUNCTION LocaleCompareAtStrength

Uni \

integer4	LocaleCompareAtStrength
(unicode src1, unicode src2, varstring locale_name, integer1 strength)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

PARAMETER src2 ||| UNICODE — The second string to be compared.

PARAMETER src1 ||| UNICODE — The first string to be compared.

PARAMETER **strength** ||| INTEGER1 — The strength of the comparison 1 ignores accents and case, differentiating only between letters 2 ignores case but differentiates between accents. 3 differentiates between accents and case but ignores e.g. differences between Hiragana and Katakana 4 differentiates between accents and case and e.g. Hiragana/Katakana, but ignores e.g. Hebrew cantillation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN INTEGER4 —

FUNCTION Reverse

Uni \

unicode	Reverse
(unicode src)	

Returns the argument string with all characters in reverse order. Note the argument is not TRIMMED before it is reversed.

PARAMETER src ||| UNICODE — The string that is being reversed.

RETURN UNICODE —

FUNCTION FindReplace

Uni \

unicode	FindReplace
(unicode src, unicode sought, unicode replacement)	

Returns the source string with the replacement string substituted for all instances of the search string.

PARAMETER sought ||| UNICODE — The string to be replaced.

PARAMETER replacement ||| UNICODE — The string to be substituted into the result.

PARAMETER src ||| UNICODE — The string that is being transformed.

RETURN UNICODE —

FUNCTION LocaleFindReplace

Uni \

unicode	LocaleFindReplace
(unicode src, unicode sought, unicode replacement, varstring locale_name)	

Returns the source string with the replacement string substituted for all instances of the search string.

PARAMETER sought ||| UNICODE — The string to be replaced.

PARAMETER replacement ||| UNICODE — The string to be substituted into the result.

PARAMETER src ||| UNICODE — The string that is being transformed.

PARAMETER locale__name ||| VARSTRING — The locale to use for the comparison

RETURN UNICODE —

FUNCTION LocaleFindAtStrengthReplace

Uni \

unicode	LocaleFindAtStrengthReplace
(unicode src, unicode sought, unicode replacement, varstring locale_name, integer1 strength)	

Returns the source string with the replacement string substituted for all instances of the search string.

PARAMETER sought ||| UNICODE — The string to be replaced.

PARAMETER strength ||| INTEGER1 — The strength of the comparison

PARAMETER replacement ||| UNICODE — The string to be substituted into the result.

PARAMETER src ||| UNICODE — The string that is being transformed.

PARAMETER locale_name ||| VARSTRING — The locale to use for the comparison

RETURN UNICODE —

FUNCTION CleanAccents

Uni \

unicode	CleanAccents
(unicode src)	

Returns the source string with all accented characters replaced with unaccented.

PARAMETER src ||| UNICODE — The string that is being transformed.

RETURN UNICODE —

FUNCTION CleanSpaces

Uni \

unicode	CleanSpaces
(unicode src)	

Returns the source string with all instances of multiple adjacent space characters (2 or more spaces together) reduced to a single space character. Leading and trailing spaces are removed, and tab characters are converted to spaces.

PARAMETER src ||| UNICODE — The string to be cleaned.

RETURN UNICODE —

FUNCTION WildMatch

Uni \

boolean	WildMatch
(unicode src, unicode _pattern, boolean _noCase)	

Tests if the search string matches the pattern. The pattern can contain wildcards '?' (single character) and '*' (multiple character).

PARAMETER pattern ||| — The pattern to match against.

PARAMETER src ||| UNICODE — The string that is being tested.

PARAMETER ignore_case ||| — Whether to ignore differences in case between characters

PARAMETER __pattern ||| UNICODE — No Doc

PARAMETER __nocase ||| BOOLEAN — No Doc

RETURN BOOLEAN —

FUNCTION Contains

Uni \

BOOLEAN	Contains
(unicode src, unicode _pattern, boolean _noCase)	

Tests if the search string contains each of the characters in the pattern. If the pattern contains duplicate characters those characters will match once for each occurrence in the pattern.

PARAMETER pattern ||| — The pattern to match against.

PARAMETER src ||| UNICODE — The string that is being tested.

PARAMETER ignore_case ||| — Whether to ignore differences in case between characters

PARAMETER _pattern ||| UNICODE — No Doc

PARAMETER _nocase ||| BOOLEAN — No Doc

RETURN BOOLEAN —

FUNCTION EditDistance

Uni \

UNSIGNED4	EditDistance
(unicode _left, unicode _right, varstring localename = "")	

Returns the minimum edit distance between the two strings. An insert change or delete counts as a single edit. The two strings are trimmed before comparing.

PARAMETER _right ||| UNICODE — The second string to be compared.

PARAMETER localename ||| — The locale to use for the comparison. Defaults to ”.

PARAMETER _left ||| UNICODE — The first string to be compared.

PARAMETER localename ||| VARSTRING — No Doc

RETURN UNSIGNED4 — The minimum edit distance between the two strings.

FUNCTION EditDistanceWithinRadius

Uni \

BOOLEAN	EditDistanceWithinRadius
(unicode _left, unicode _right, unsigned4 radius, varstring localename = ")	

Returns true if the minimum edit distance between the two strings is within a specific range. The two strings are trimmed before comparing.

PARAMETER radius ||| UNSIGNED4 — The maximum edit distance that is acceptable.

PARAMETER _right ||| UNICODE — The second string to be compared.

PARAMETER localename ||| — The locale to use for the comparison. Defaults to "".

PARAMETER _left ||| UNICODE — The first string to be compared.

PARAMETER localename ||| VARSTRING — No Doc

RETURN BOOLEAN — Whether or not the two strings are within the given specified edit distance.

FUNCTION WordCount

Uni \

unsigned4	WordCount
(unicode text, varstring localename = ")	

Returns the number of words in the string. Word boundaries are marked by the unicode break semantics.

PARAMETER text ||| UNICODE — The string to be broken into words.

PARAMETER localename ||| — The locale to use for the break semantics. Defaults to "".

PARAMETER localename ||| VARSTRING — No Doc

RETURN UNSIGNED4 — The number of words in the string.

FUNCTION GetNthWord

Uni \

unicode	GetNthWord
(unicode text, unsigned4 n, varstring localename = ")	

Returns the n-th word from the string. Word boundaries are marked by the unicode break semantics.

PARAMETER text ||| UNICODE — The string to be broken into words.

PARAMETER n ||| UNSIGNED4 — Which word should be returned from the function.

PARAMETER localname ||| — The locale to use for the break semantics. Defaults to ”.

PARAMETER localename ||| VARSTRING — No Doc

RETURN UNICODE — The number of words in the string.
