Root

Table of Contents

Audit

IMPORTS

- lib_auditlib

DESCRIPTIONS

BLAS

IMPORTS

- lib_eclblas

DESCRIPTIONS

MODULE : BLAS

	BLAS

Up

Types | ICellFunc | Apply2Cells | dasum | daxpy | dgemm | dgetf2 | dpotf2 | dscal | dsyrk | dtrsm | extract_diag | extract_tri | make_diag | make_vector | trace |

MODULE : Types

	Types

Up

value_t | dimension_t | matrix_t | Triangle | Diagonal | Side |

ATTRIBUTE : value_t

	value_t

ATTRIBUTE : dimension_t

	dimension_t

ATTRIBUTE : matrix_t

	matrix_t

ATTRIBUTE : Triangle

	Triangle

ATTRIBUTE : Diagonal

	Diagonal

ATTRIBUTE : Side

	Side

FUNCTION : ICellFunc

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

Function prototype for Apply2Cell.

Parameter v ||| the value

Parameter r ||| the row ordinal

Parameter c ||| the column ordinal

Return the updated value

FUNCTION : Apply2Cells

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 m ||| number of rows

Parameter n ||| number of columns

Parameter x ||| matrix

Parameter f ||| function to apply

Return updated matrix

FUNCTION : dasum

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 m ||| the number of entries

Parameter x ||| the column major matrix holding the vector

Parameter incx ||| the increment for x, 1 in the case of an actual vector

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

Return the sum of the absolute values

FUNCTION : daxpy

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 N ||| number of elements in vector

Parameter alpha ||| the scalar multiplier

Parameter X ||| the column major matrix holding the vector X

Parameter incX ||| the increment or stride for the vector

Parameter Y ||| the column major matrix holding the vector Y

Parameter incY ||| the increment or stride of Y

Parameter x_skipped ||| number of entries skipped to get to the first X

Parameter y_skipped ||| number of entries skipped to get to the first Y

Return the updated matrix

FUNCTION : dgemm

Types.matrix_t	dgemm
(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=[])	

[Up](#)

alpha*op(A) op(B) + beta*C where op() is transpose

Parameter transposeA ||| true when transpose of A is used

Parameter transposeB ||| true when transpose of B is used

Parameter M ||| number of rows in product

Parameter N ||| number of columns in product

Parameter K ||| number of columns/rows for the multiplier/multiplicand

Parameter alpha ||| scalar used on A

Parameter A ||| matrix A

Parameter B ||| matrix B

Parameter beta ||| scalar for matrix C

Parameter C ||| matrix C or empty

FUNCTION : dgetf2

Types.matrix_t	dgetf2
(Types.dimension_t m, Types.dimension_t n, Types.matrix_t a)	

Compute LU Factorization of matrix A.

Parameter m ||| number of rows of A

Parameter n ||| number of columns of A

Return composite matrix of factors, lower triangle has an implied diagonal of ones. Upper triangle has the diagonal of the composite.

FUNCTION : dpotf2

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 ||| indicate whether upper or lower triangle is used

Parameter r ||| number of rows/columns in the square matrix

Parameter A ||| the square matrix

Parameter clear ||| clears the unused triangle

Return the triangular matrix requested.

FUNCTION : dscal

Types.matrix_t	dscal
(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 N ||| number of elements in the vector

Parameter alpha ||| the scaling factor

Parameter X ||| the column major matrix holding the vector

Parameter incX ||| the stride to get to the next element in the vector

Parameter skipped ||| the number of elements skipped to get to the first element

Return the updated matrix

FUNCTION : dsyrk

Types.matrix_t	dsyrk
(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 ||| update upper or lower triangle

Parameter transposeA ||| Transpose the A matrix to be NxK

Parameter N ||| number of rows

Parameter K ||| number of columns in the update matrix or transpose

Parameter alpha ||| the alpha scalar

Parameter A ||| the update matrix, either NxK or KxN

Parameter beta ||| the beta scalar

Parameter C ||| the matrix to update

Parameter clear ||| 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.

FUNCTION : dtrsm

Types.matrix_t	dtrsm
(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)	

Triangular matrix solver. op(A) X = alpha B or X op(A) = alpha B where op is Transpose, X and B is MxN

Parameter side ||| side for A, Side.Ax is op(A) X = alpha B

Parameter tri ||| Says whether A is Upper or Lower triangle

Parameter transposeA ||| is op(A) the transpose of A

Parameter diag ||| is the diagonal an implied unit diagonal or supplied

Parameter M ||| number of rows

Parameter N ||| number of columns

Parameter lda ||| the leading dimension of the A matrix, either M or N

Parameter alpha ||| the scalar multiplier for B

Parameter A ||| a triangular matrix

Parameter B ||| the matrix of values for the solve

Return the matrix of coefficients to get B.

FUNCTION : extract_diag

Types.matrix_t	extract_diag
(Types.dimension_t m, Types.dimension_t n, Types.matrix_t x)	

Extract the diagonal of he matrix

Parameter m ||| number of rows

Parameter n ||| number of columns

Parameter x ||| matrix from which to extract the diagonal

Return diagonal matrix

FUNCTION : extract_tri

Types.matrix_t	extract_tri
(Types.dimension_t m, Types.dimension_t n, Types.Triangle tri, Types.Diagonal dt, Types.matrix_t a)	

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

Parameter m ||| number of rows

Parameter n ||| number of columns

Parameter tri ||| Upper or Lower specifier, Triangle.Lower or Triangle.Upper

Parameter dt ||| Use Diagonal.NotUnitTri or Diagonal.UnitTri

Parameter a ||| Matrix, usually a composite from factoring

Return the triangle

FUNCTION : make_diag

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

Generate a diagonal matrix.

Parameter m ||| number of diagonal entries

Parameter v ||| option value, defaults to 1

Parameter X ||| optional input of diagonal values, multiplied by v.

Return a diagonal matrix

FUNCTION : make_vector

Types.matrix_t	make_vector
(Types.dimension_t m, Types.value_t v=1.0)	

Make a vector of dimension m

Parameter m ||| number of elements

Parameter v ||| the values, defaults to 1

Return the vector

FUNCTION : trace

Types.value_t	trace
(Types.dimension_t m, Types.dimension_t n, Types.matrix_t x)	

The trace of the input matrix

13

Parameter m ||| number of rows

Parameter n ||| number of columns

Parameter x ||| the matrix

Return the trace (sum of the diagonal entries)

BundleBase

IMPORTS

DESCRIPTIONS

MODULE : BundleBase

	BundleBase

Up

PropertyRecord | Name | Description | Authors | License | Copyright | DependsOn | Version | Properties | PlatformVersion |

RECORD : PropertyRecord

	PropertyRecord

Up

ATTRIBUTE : Name

STRING	Name

Up

ATTRIBUTE : Description

UTF8	Description

ATTRIBUTE : Authors

SET OF UTF8	Authors

ATTRIBUTE : License

UTF8	License

ATTRIBUTE : Copyright

UTF8	Copyright

ATTRIBUTE : DependsOn

SET OF STRING	DependsOn

ATTRIBUTE : Version

STRING	Version

ATTRIBUTE : Properties

	Properties

ATTRIBUTE : PlatformVersion

STRING	PlatformVersion

Date

IMPORTS

DESCRIPTIONS

MODULE : Date

	Date

Up

Date_rec | Date_t | Days_t | Time_rec | Time_t | Seconds_t | DateTime_rec | Timestamp_t | Year |
Month | Day | Hour | Minute | Second | DateFromParts | TimeFromParts | SecondsFromParts |
SecondsToParts | TimestampToSeconds | IsLeapYear | IsDateLeapYear | FromGregorianYMD |
ToGregorianYMD | FromGregorianDate | ToGregorianDate | DayOfYear | DayOfWeek |
IsJulianLeapYear | FromJulianYMD | ToJulianYMD | FromJulianDate | ToJulianDate | DaysSince1900 |
ToDaysSince1900 | FromDaysSince1900 | YearsBetween | MonthsBetween | DaysBetween |
DateFromDateRec | DateFromRec | TimeFromTimeRec | DateFromDateTimeRec |
TimeFromDateTimeRec | SecondsFromDateTimeRec | FromStringToDate | FromString |
FromStringToTime | MatchDateString | MatchTimeString | DateToString | TimeToString |
SecondsToString | ToString | ConvertDateFormat | ConvertFormat | ConvertTimeFormat |
ConvertDateFormatMultiple | ConvertFormatMultiple | ConvertTimeFormatMultiple | AdjustDate |
AdjustDateBySeconds | AdjustTime | AdjustTimeBySeconds | AdjustSeconds | AdjustCalendar |
IsLocalDaylightSavingsInEffect | LocalTimeZoneOffset | CurrentDate | Today | CurrentTime |
CurrentSeconds | CurrentTimestamp | DatesForMonth | DatesForWeek | IsValidDate |
IsValidGregorianDate | IsValidTime | CreateDate | CreateDateFromSeconds | CreateTime |
CreateTimeFromSeconds | CreateDateTime | CreateDateTimeFromSeconds |

RECORD : Date_rec

	Date_rec

ATTRIBUTE : Date_t

Date_t

ATTRIBUTE : Days_t

Days_t

RECORD : Time_rec

Time_rec

ATTRIBUTE : Time_t

Time_t

ATTRIBUTE : Seconds_t

	Seconds_t

RECORD : DateTime_rec

	DateTime_rec

ATTRIBUTE : Timestamp_t

	Timestamp_t

FUNCTION : Year

INTEGER2	Year
(Date_t date)	

Extracts the year from a date type.

Parameter date ||| The date.

Return An integer representing the year.

FUNCTION : Month

UNSIGNED1	Month
(Date_t date)	

Extracts the month from a date type.

Parameter date ||| The date.

Return An integer representing the year.

FUNCTION : Day

UNSIGNED1	Day
(Date_t date)	

Extracts the day of the month from a date type.

Parameter date ||| The date.

Return An integer representing the year.

FUNCTION : Hour

UNSIGNED1	Hour
(Time_t time)	

Extracts the hour from a time type.

Parameter time ||| The time.

Return An integer representing the hour.

FUNCTION : Minute

UNSIGNED1	Minute
(Time_t time)	

Extracts the minutes from a time type.

Parameter time ||| The time.

Return An integer representing the minutes.

FUNCTION : Second

UNSIGNED1	Second
(Time_t time)	

Extracts the seconds from a time type.

Parameter time ||| The time.

Return An integer representing the seconds.

FUNCTION : DateFromParts

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

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 A date created by combining the fields.

FUNCTION : TimeFromParts

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

Combines hour, minute second to create a time type.

Parameter hour ||| The hour (0-23).

Parameter minute ||| The minute (0-59).

Parameter second ||| The second (0-59).

Return A time created by combining the fields.

FUNCTION : SecondsFromParts

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 ||| The year (1601-30827).

Parameter month ||| The month (1-12).

Parameter day ||| The day (1..daysInMonth).

Parameter hour ||| The hour (0-23).

Parameter minute ||| The minute (0-59).

Parameter second ||| The second (0-59).

Parameter is_local_time ||| 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 A Seconds_t value created by combining the fields.

MODULE : SecondsToParts

	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 ||| The number of seconds since epoch.

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

ATTRIBUTE : Year

INTEGER2	Year

ATTRIBUTE : Month

UNSIGNED1	Month

ATTRIBUTE : Day

UNSIGNED1	Day

ATTRIBUTE : Hour

UNSIGNED1	Hour

ATTRIBUTE : Minute

UNSIGNED1	Minute

ATTRIBUTE : Second

UNSIGNED1	Second

ATTRIBUTE : day_of_week

UNSIGNED1	day_of_week

ATTRIBUTE : date

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 A date created by combining the fields.

ATTRIBUTE : time

Time_t	time

Combines hour, minute second to create a time type.

Parameter hour ||| The hour (0-23).

Parameter minute ||| The minute (0-59).

Parameter second ||| The second (0-59).

Return A time created by combining the fields.

FUNCTION : TimestampToSeconds

Seconds_t	TimestampToSeconds
(Timestamp_t timestamp)	

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

Parameter timestamp ||| The number of microseconds since epoch.

Return The number of seconds since epoch.

FUNCTION : IsLeapYear

BOOLEAN	IsLeapYear
(INTEGER2 year)	

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

Parameter year ||| The year (0-9999).

Return True if the year is a leap year.

FUNCTION : IsDateLeapYear

BOOLEAN	IsDateLeapYear
(Date_t date)	

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

Parameter date ||| The date.

Return True if the year is a leap year.

FUNCTION : FromGregorianYMD

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 ||| The year (-4713..9999).

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

Parameter day ||| The day (1..daysInMonth). A missing value (0) is treated as 1.

Return The number of elapsed days (1 Jan 1AD = 1)

MODULE : ToGregorianYMD

	ToGregorianYMD
(Days_t days)	

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

Parameter days ||| The number of elapsed days (1 Jan 1AD = 1)

Return Module containing Year, Month, Day in the Gregorian calendar

year | month | day |

ATTRIBUTE : year

	year

ATTRIBUTE : month

	month

ATTRIBUTE : day

	day

FUNCTION : FromGregorianDate

Days_t	FromGregorianDate
(Date_t date)	

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

Parameter date ||| The date (using the Gregorian calendar)

Return The number of elapsed days (1 Jan 1AD = 1)

FUNCTION : ToGregorianDate

Date_t	ToGregorianDate
(Days_t days)	

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

Parameter days ||| The number of elapsed days (1 Jan 1AD = 1)

Return A Date_t in the Gregorian calendar

FUNCTION : DayOfYear

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 ||| A Date_t value.

Return A number (1-366) representing the number of days since the beginning of the year.

FUNCTION : DayOfWeek

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 ||| A Date_t value.

Return A number 1-7 representing the day of the week, where 1 = Sunday.

FUNCTION : IsJulianLeapYear

BOOLEAN	IsJulianLeapYear
(INTEGER2 year)	

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

Parameter year ||| The year (0-9999).

Return True if the year is a leap year.

FUNCTION : FromJulianYMD

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

Up

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

Parameter year ||| The year (-4800..9999).

Parameter month ||| The month (1-12).

Parameter day ||| The day (1..daysInMonth).

Return The number of elapsed days (1 Jan 1AD = 1)

MODULE : ToJulianYMD

	ToJulianYMD
(Days_t days)	

Up

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

Parameter days ||| The number of elapsed days (1 Jan 1AD = 1)

Return Module containing Year, Month, Day in the Julian calendar

Day | Month | Year |

ATTRIBUTE : Day

UNSIGNED1	Day

ATTRIBUTE : Month

UNSIGNED1	Month

ATTRIBUTE : Year

INTEGER2	Year

FUNCTION : FromJulianDate

Days_t	FromJulianDate
(Date_t date)	

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

Parameter date ||| The date (using the Julian calendar)

Return The number of elapsed days (1 Jan 1AD = 1)

FUNCTION : ToJulianDate

Date_t	ToJulianDate
(Days_t days)	

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

Parameter days ||| The number of elapsed days (1 Jan 1AD = 1)

Return A Date_t in the Julian calendar

FUNCTION : DaysSince1900

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

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

Parameter year ||| The year (-4713..9999).

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

Parameter day ||| The day (1..daysInMonth). A missing value (0) is treated as 1.

Return The number of elapsed days since 1st January 1900

FUNCTION : ToDaysSince1900

Days_t	ToDaysSince1900
(Date_t date)	

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

Parameter date ||| The date

Return The number of elapsed days since 1st January 1900

FUNCTION : FromDaysSince1900

Date_t	FromDaysSince1900
(Days_t days)	

Up

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

Parameter days ||| The number of elapsed days since 1st Jan 1900

Return A Date_t in the Julian calendar

FUNCTION : YearsBetween

INTEGER	YearsBetween
(Date_t from, Date_t to)	

Up

Calculate the number of whole years between two dates.

Parameter from ||| The first date

Parameter to ||| The last date

Return The number of years between them.

FUNCTION : MonthsBetween

INTEGER	MonthsBetween
(Date_t from, Date_t to)	

Calculate the number of whole months between two dates.

Parameter from ||| The first date

Parameter to ||| The last date

Return The number of months between them.

FUNCTION : DaysBetween

INTEGER	DaysBetween
(Date_t from, Date_t to)	

Calculate the number of days between two dates.

Parameter from ||| The first date

Parameter to ||| The last date

Return The number of days between them.

FUNCTION : DateFromDateRec

Date_t	DateFromDateRec
(Date_rec date)	

Combines the fields from a Date_rec to create a Date_t

Parameter date ||| The row containing the date.

Return A Date_t representing the combined values.

FUNCTION : DateFromRec

Date_t	DateFromRec
(Date_rec date)	

Combines the fields from a Date_rec to create a Date_t

Parameter date ||| The row containing the date.

Return A Date_t representing the combined values.

FUNCTION : TimeFromTimeRec

Time_t	TimeFromTimeRec
(Time_rec time)	

Combines the fields from a Time_rec to create a Time_t

Parameter time ||| The row containing the time.

Return A Time_t representing the combined values.

FUNCTION : DateFromDateTimeRec

Date_t	DateFromDateTimeRec
(DateTime_rec datetime)	

Combines the date fields from a DateTime_rec to create a Date_t

Parameter datetime ||| The row containing the datetime.

Return A Date_t representing the combined values.

FUNCTION : TimeFromDateTimeRec

Time_t	TimeFromDateTimeRec
(DateTime_rec datetime)	

Combines the time fields from a DateTime_rec to create a Time_t

Parameter datetime ||| The row containing the datetime.

Return A Time_t representing the combined values.

FUNCTION : SecondsFromDateTimeRec

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 ||| The row containing the datetime.

Parameter is_local_time ||| 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 A Seconds_t representing the combined values.

FUNCTION : FromStringToDate

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 ||| The string to be converted.

Parameter format ||| The format of the input string. (See documentation for strftime)

Return 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_t	FromString
(STRING date_text, VARSTRING format)	

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

Parameter date_text ||| The string to be converted.

Parameter format ||| The format of the input string. (See documentation for strftime)

Return The date that was matched in the string. Returns 0 if failed to match.

FUNCTION : FromStringToTime

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

Return 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_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 date_text ||| The string to be converted.

Parameter formats ||| A set of formats to check against the string. (See documentation for strftime)

Return The date that was matched in the string. Returns 0 if failed to match.

FUNCTION : MatchTimeString

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 time_text ||| The string to be converted.

Parameter formats ||| A set of formats to check against the string. (See documentation for strftime)

Return The time that was matched in the string. Returns 0 if failed to match.

FUNCTION : DateToString

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

Formats a date as a string.

Parameter date ||| The date to be converted.

Parameter format ||| 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 Blank if date cannot be formatted, or the date in the requested format.

FUNCTION : TimeToString

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

Formats a time as a string.

Parameter time ||| The time to be converted.

Parameter format ||| 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 Blank if the time cannot be formatted, or the time in the requested format.

FUNCTION : SecondsToString

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 seconds ||| The seconds since epoch.

Parameter format ||| 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.

Return The converted seconds as a string.

FUNCTION : ToString

STRING	ToString
(Date_t date, VARSTRING format)	

Formats a date as a string.

Parameter date ||| The date to be converted.

Parameter format ||| The format the date is output in. (See documentation for strftime)

Return Blank if date cannot be formatted, or the date in the requested format.

FUNCTION : ConvertDateFormat

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 date_text ||| The string containing the date to be converted.

Parameter from_format ||| The format the date is to be converted from.

Parameter to_format ||| The format the date is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : ConvertFormat

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 date_text ||| The string containing the date to be converted.

Parameter from_format ||| The format the date is to be converted from.

Parameter to_format ||| The format the date is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : ConvertTimeFormat

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 time_text ||| The string containing the time to be converted.

Parameter from_format ||| The format the time is to be converted from.

Parameter to_format ||| The format the time is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : ConvertDateFormatMultiple

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 date_text ||| The string containing the date to be converted.

Parameter from_formats ||| The list of formats the date is to be converted from.

Parameter to_format ||| The format the date is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : ConvertFormatMultiple

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 date_text ||| The string containing the date to be converted.

Parameter from_formats ||| The list of formats the date is to be converted from.

Parameter to_format ||| The format the date is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : ConvertTimeFormatMultiple

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 time_text ||| The string containing the time to be converted.

Parameter from_formats ||| The list of formats the time is to be converted from.

Parameter to_format ||| The format the time is to be converted to.

Return The converted string, or blank if it failed to match the format.

FUNCTION : AdjustDate

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 ||| The date to adjust.

Parameter year_delta ||| The requested change to the year value; optional, defaults to zero.

Parameter month_delta ||| The requested change to the month value; optional, defaults to zero.

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

Return The adjusted Date_t value.

FUNCTION : AdjustDateBySeconds

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 ||| The date to adjust.

Parameter seconds_delta ||| The requested change to the date, in seconds.

Return The adjusted Date_t value.

FUNCTION : AdjustTime

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 ||| The time to adjust.

Parameter hour_delta ||| The requested change to the hour value; optional, defaults to zero.

Parameter minute_delta ||| The requested change to the minute value; optional, defaults to zero.

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

Return The adjusted Time_t value.

FUNCTION : AdjustTimeBySeconds

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 ||| The time to adjust.

Parameter seconds_delta ||| The requested change to the time, in seconds.

Return The adjusted Time_t value.

FUNCTION : AdjustSeconds

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 seconds ||| The number of seconds to adjust.

Parameter year_delta ||| The requested change to the year value; optional, defaults to zero.

Parameter month_delta ||| The requested change to the month value; optional, defaults to zero.

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

Parameter hour_delta ||| The requested change to the hour value; optional, defaults to zero.

Parameter minute_delta ||| The requested change to the minute value; optional, defaults to zero.

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

Return The adjusted Seconds_t value.

FUNCTION : AdjustCalendar

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

Parameter year_delta ||| The requested change to the year value; optional, defaults to zero.

Parameter month_delta ||| The requested change to the month value; optional, defaults to zero.

Parameter day_delta ||| The requested change to the day value; optional, defaults to zero.

Return The adjusted Date_t value.

FUNCTION : IsLocalDaylightSavingsInEffect

BOOLEAN	IsLocalDaylightSavingsInEffect
()	

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

Return TRUE if daylight savings time is currently in effect, FALSE otherwise.

FUNCTION : LocalTimeZoneOffset

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 The number of seconds offset from UTC.

FUNCTION : CurrentDate

Date_t	CurrentDate
(BOOLEAN in_local_time = FALSE)	

Returns the current date.

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

Return A Date_t representing the current date.

FUNCTION : Today

Date_t	Today
()	

Returns the current date in the local time zone.

Return A Date_t representing the current date.

FUNCTION : CurrentTime

Time_t	CurrentTime
(BOOLEAN in_local_time = FALSE)	

Returns the current time of day

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

Return A Time_t representing the current time of day.

FUNCTION : CurrentSeconds

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

Return A Seconds_t representing the current time in UTC or local time, depending on the argument.

FUNCTION : CurrentTimestamp

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

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

MODULE : DatesForMonth

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 ||| 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.

ATTRIBUTE : startDate

Date_t	startDate

ATTRIBUTE : endDate

Date_t	endDate

MODULE : DatesForWeek

DatesForWeek	
(Date_t as_of_date = CurrentDate(FALSE))	

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

Parameter as_of_date ||| 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.

startDate | endDate |

ATTRIBUTE : startDate

Date_t	startDate

ATTRIBUTE : endDate

Date_t	endDate

FUNCTION : IsValidDate

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 ||| The date to validate.

Parameter yearLowerBound ||| The minimum acceptable year. Optional; defaults to 1800.

Parameter yearUpperBound ||| The maximum acceptable year. Optional; defaults to 2100.

Return TRUE if the date is valid, FALSE otherwise.

FUNCTION : IsValidGregorianDate

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 ||| The Date_t to validate.

Return TRUE if the date is valid, FALSE otherwise.

FUNCTION : IsValidTime

BOOLEAN	IsValidTime
(Time_t time)	

Tests whether a time is valid.

Parameter time ||| The time to validate.

Return TRUE if the time is valid, FALSE otherwise.

TRANSFORM : CreateDate

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

A transform to create a Date_rec from the individual elements

Parameter year ||| The year

Parameter month ||| The month (1-12).

Parameter day ||| The day (1..daysInMonth).

Return A transform that creates a Date_rec containing the date.

TRANSFORM : CreateDateFromSeconds

Date_rec	CreateDateFromSeconds
(Seconds_t seconds)	

A transform to create a Date_rec from a Seconds_t value.

Parameter seconds ||| The number seconds since epoch.

Return A transform that creates a Date_rec containing the date.

TRANSFORM : CreateTime

Time_rec	CreateTime
(UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second)	

A transform to create a Time_rec from the individual elements

Parameter hour ||| The hour (0-23).

Parameter minute ||| The minute (0-59).

Parameter second ||| The second (0-59).

Return A transform that creates a Time_rec containing the time of day.

TRANSFORM : CreateTimeFromSeconds

Time_rec	CreateTimeFromSeconds
(Seconds_t seconds)	

A transform to create a Time_rec from a Seconds_t value.

Parameter seconds ||| The number seconds since epoch.

Return A transform that creates a Time_rec containing the time of day.

TRANSFORM : CreateDateTime

DateTime_rec	CreateDateTime
(INTEGER2 year, UNSIGNED1 month, UNSIGNED1 day, UNSIGNED1 hour, UNSIGNED1 minute, UNSIGNED1 second)	

A transform to create a DateTime_rec from the individual elements

Parameter year ||| The year

Parameter month ||| The month (1-12).

Parameter day ||| The day (1..daysInMonth).

Parameter hour ||| The hour (0-23).

Parameter minute ||| The minute (0-59).

Parameter second ||| The second (0-59).

Return A transform that creates a DateTime_rec containing date and time components.

TRANSFORM : CreateDateTimeFromSeconds

DateTime_rec	CreateDateTimeFromSeconds
(Seconds_t seconds)	

A transform to create a DateTime_rec from a Seconds_t value.

Parameter seconds ||| The number seconds since epoch.

Return A transform that creates a DateTime_rec containing date and time components.

File

IMPORTS

- lib_fileservices

DESCRIPTIONS

MODULE : File

	File

Up

FsFilenameRecord | FsLogicalFileName | FsLogicalFileNameRecord | FsLogicalFileInfoRecord | FsLogicalSuperSubRecord | FsFileRelationshipRecord | RECFMV_RECSIZE | RECFMVB_RECSIZE | PREFIX_VARIABLE_RECSIZE | PREFIX_VARIABLE_BIGENDIAN_RECSIZE | FileExists | DeleteLogicalFile | SetReadOnly | RenameLogicalFile | ForeignLogicalFileName | ExternalLogicalFileName | GetFileDescription | SetFileDescription | RemoteDirectory | LogicalFileList | CompareFiles | VerifyFile | AddFileRelationship | FileRelationshipList | RemoveFileRelationship | GetColumnMapping | SetColumnMapping | EncodeRfsQuery | RfsAction | MoveExternalFile | DeleteExternalFile | CreateExternalDirectory | GetLogicalFileAttribute | ProtectLogicalFile | DfuPlusExec | fSprayFixed | SprayFixed | fSprayVariable | SprayVariable | fSprayDelimited | SprayDelimited | fSprayXml | SprayXml | fDespray | Despray | fCopy | Copy | fReplicate | Replicate | fRemotePull | RemotePull | fMonitorLogicalFileName | MonitorLogicalFileName | fMonitorFile | MonitorFile | WaitDfuWorkunit | AbortDfuWorkunit | CreateSuperFile | SuperFileExists | DeleteSuperFile | GetSuperFileSubCount | GetSuperFileSubName | FindSuperFileSubName | StartSuperFileTransaction | AddSuperFile | RemoveSuperFile | ClearSuperFile | RemoveOwnedSubFiles | DeleteOwnedSubFiles | SwapSuperFile | ReplaceSuperFile | FinishSuperFileTransaction | SuperFileContents | LogicalFileSuperOwners | LogicalFileSuperSubList | fPromoteSuperFileList | PromoteSuperFileList |

RECORD : FsFilenameRecord

FsFilenameRecord

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

ATTRIBUTE : FsLogicalFileName

FsLogicalFileName

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

RECORD : FsLogicalFileNameRecord

FsLogicalFileNameRecord

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

Field name ||| The logical name of the file;

RECORD : FsLogicalFileInfoRecord

FsLogicalFileInfoRecord

A record containing information about a logical file.

Field superfile ||| Is this a superfile?

Field size ||| Number of bytes in the file (before compression)

Field rowcount ||| Number of rows in the file.

RECORD : FsLogicalSuperSubRecord

FsLogicalSuperSubRecord

A record containing information about a superfile and its contents.

Field supername ||| The name of the superfile

Field subname ||| The name of the sub-file

RECORD : FsFileRelationshipRecord

FsFileRelationshipRecord

A record containing information about the relationship between two files.

Field primaryfile ||| The logical filename of the primary file

Field secondaryfile ||| The logical filename of the secondary file.

Field primaryflds ||| 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 secondaryflds ||| The name of the foreign key field relating to the primary file.

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

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

Field payload ||| Indicates whether the primary or secondary are payload INDEXes.

Field description ||| The description of the relationship.

ATTRIBUTE : RECFMV_RECSIZE

	RECFMV_RECSIZE

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

ATTRIBUTE : RECFMVB_RECSIZE

	RECFMVB_RECSIZE

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

ATTRIBUTE : PREFIX_VARIABLE_RECSIZE

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.

ATTRIBUTE : PREFIX_VARIABLE_BIGENDIAN_RECSIZE

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.

FUNCTION : FileExists

boolean	FileExists
(varstring lfn, boolean physical=FALSE)	

Returns whether the file exists.

Parameter lfn ||| The logical name of the file.

Parameter physical ||| Whether to also check for the physical existence on disk. Defaults to FALSE.

Return Whether the file exists.

FUNCTION : DeleteLogicalFile

	DeleteLogicalFile
(varstring lfn, boolean allowMissing=FALSE)	

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

Parameter lfn ||| The logical name of the file.

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

FUNCTION : SetReadOnly

SetReadOnly
(varstring lfn, boolean ro=TRUE)

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

Parameter lfn ||| The logical name of the file.

Parameter ro ||| Whether updates to the file are disallowed. Defaults to TRUE.

FUNCTION : RenameLogicalFile

RenameLogicalFile
(varstring oldname, varstring newname)

Changes the name of a logical file.

Parameter oldname ||| The current name of the file to be renamed.

Parameter newname ||| The new logical name of the file.

FUNCTION : ForeignLogicalFileName

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 name ||| The logical name of the file.

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

Parameter abspath ||| Should a tilde (˜) be prepended to the resulting logical file name. Defaults to FALSE.

FUNCTION : ExternalLogicalFileName

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 ||| The IP address of the remote machine. '' can be used for the local machine.

Parameter path ||| The path/name of the file on the remote machine.

Parameter abspath ||| Should a tilde (˜) be prepended to the resulting logical file name. Defaults to TRUE.

Return The encoded logical filename.

FUNCTION : GetFileDescription

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 ||| The logical name of the file.

FUNCTION : SetFileDescription

	SetFileDescription
(varstring lfn, varstring val)	

Sets the description associated with the specified filename.

Parameter lfn ||| The logical name of the file.

Parameter val ||| The description to be associated with the file.

FUNCTION : RemoteDirectory

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 machineIP ||| The IP address of the remote machine.

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

Parameter recurse ||| Whether to include files from subdirectories under the directory. Defaults to FALSE.

FUNCTION : LogicalFileList

dataset(FsLogicalFileInfoRecord)	LogicalFileList
(varstring namepattern='*', boolean includenormal=TRUE, boolean includesuper=FALSE, boolean unknownszero=FALSE, varstring foreigndali=")	

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

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

Parameter includenormal ||| Whether to include 'normal' files. Defaults to TRUE.

Parameter includesuper ||| Whether to include SuperFiles. Defaults to FALSE.

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

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

FUNCTION : CompareFiles

INTEGER4	CompareFiles
(varstring lfn1, varstring lfn2, boolean logical_only=TRUE, boolean use_crcs=FALSE)	

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

Parameter file1 ||| The logical name of the first file.

Parameter file2 ||| The logical name of the second file.

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

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

Return 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

varstring	VerifyFile
(varstring lfn, boolean usecrcs)	

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

Parameter lfn ||| The name of the file to check.

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

Return '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

	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 primary ||| The logical filename of the primary file.

Parameter secondary ||| The logical filename of the secondary file.

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

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

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

Parameter payload ||| Indicates whether the primary or secondary are payload INDEXes.

Parameter description ||| The description of the relationship.

FUNCTION : FileRelationshipList

dataset(FsFileRelationshipRecord)	FileRelationshipList
(varstring primary, varstring secondary, varstring primflds=", varstring secondaryflds=", varstring kind='link')	

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

Parameter primary ||| The logical filename of the primary file.

Parameter secondary ||| The logical filename of the secondary file.

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

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

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

FUNCTION : RemoveFileRelationship

RemoveFileRelationship
(varstring primary, varstring secondary, varstring primaryflds=", varstring secondaryflds=", varstring kind='link')

Removes a relationship between two files.

Parameter primary ||| The logical filename of the primary file.

Parameter secondary ||| The logical filename of the secondary file.

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

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

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

FUNCTION : GetColumnMapping

varstring	GetColumnMapping
(varstring lfn)	

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

Parameter lfn ||| The logical filename of the primary file.

FUNCTION : SetColumnMapping

	SetColumnMapping
(varstring lfn, varstring mapping)	

Defines how the data in the fields of the file mist 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 lfn ||| The logical filename of the primary file.

Parameter mapping ||| A string containing a comma separated list of field mappings.

FUNCTION : EncodeRfsQuery

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

Parameter query ||| The text of the query to send to the server

---

FUNCTION : RfsAction

	RfsAction
(varstring server, varstring query)	

Sends the query to the rfs server.

Parameter server ||| A string containing the ip:port address for the remote file server.

Parameter query ||| The text of the query to send to the server

---

FUNCTION : MoveExternalFile

MoveExternalFile
(varstring location, varstring frompath, varstring topath)

[Up](#)

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

Parameter location ||| The IP address of the remote machine.

Parameter frompath ||| The path/name of the file to move.

Parameter topath ||| The path/name of the target file.

FUNCTION : DeleteExternalFile

DeleteExternalFile
(varstring location, varstring path)

[Up](#)

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

Parameter location ||| The IP address of the remote machine.

Parameter path ||| The path/name of the file to remove.

FUNCTION : CreateExternalDirectory

	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 ||| The IP address of the remote machine.

Parameter path ||| The path/name of the file to remove.

FUNCTION : GetLogicalFileAttribute

varstring	GetLogicalFileAttribute
(varstring lfn, varstring attrname)	

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

Parameter lfn ||| The name of the logical file.

Parameter attrname ||| The name of the file attribute to return.

FUNCTION : ProtectLogicalFile

	ProtectLogicalFile
(varstring lfn, boolean value=TRUE)	

Toggles protection on and off for the specified logicalfilename.

Parameter lfn ||| The name of the logical file.

Parameter value ||| TRUE to enable protection, FALSE to disable.

FUNCTION : DfuPlusExec

	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.

FUNCTION : fSprayFixed

varstring	fSprayFixed
(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)	

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

Parameter sourceIP ||| The IP address of the file.

Parameter sourcePath ||| The path and name of the file.

Parameter recordsize ||| The size (in bytes) of the records in the file.

Parameter destinationGroup ||| The name of the group to distribute the file across.

Parameter destinationLogicalName ||| The logical name of the file to create.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Parameter maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Parameter replicate ||| Whether to replicate the new file. Defaults to FALSE.

Parameter compress ||| Whether to compress the new file. Defaults to FALSE.

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

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

Return The DFU workunit id for the job.

FUNCTION : SprayFixed

SprayFixed
(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)

Up

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

See fSprayFixed

FUNCTION : fSprayVariable

varstring	fSprayVariable
(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)	

Up

FUNCTION : SprayVariable

	SprayVariable
(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)	

Up

FUNCTION : fSprayDelimited

varstring	fSprayDelimited
(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)	

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

Parameter sourceIP ||| The IP address of the file.

Parameter sourcePath ||| The path and name of the file.

Parameter sourceCsvSeparate ||| The character sequence which separates fields in the file.

Parameter sourceCsvTerminate ||| The character sequence which separates records in the file.

Parameter sourceCsvQuote ||| A string which can be used to delimit fields in the file.

Parameter sourceMaxRecordSize ||| The maximum size (in bytes) of the records in the file.

Parameter destinationGroup ||| The name of the group to distribute the file across.

Parameter destinationLogicalName ||| The logical name of the file to create.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Parameter maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Parameter replicate ||| Whether to replicate the new file. Defaults to FALSE.

Parameter compress ||| Whether to compress the new file. Defaults to FALSE.

Parameter sourceCsvEscape ||| A character that is used to escape quote characters. Defaults to none.

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

Parameter recordStructurePresent ||| If TRUE derives the record structure from the header of the file.

Parameter quotedTerminator ||| 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 expireDays ||| Number of days to auto-remove file. Default is -1, not expire.

Return The DFU workunit id for the job.

FUNCTION : SprayDelimited

SprayDelimited
(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)

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

See fSprayDelimited

FUNCTION : fSprayXml

varstring	fSprayXml
(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)	

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

Parameter sourceIP ||| The IP address of the file.

Parameter sourcePath ||| The path and name of the file.

Parameter sourceMaxRecordSize ||| The maximum size (in bytes) of the records in the file.

Parameter sourceRowTag ||| The xml tag that is used to delimit records in the source file. (This tag cannot recursivly nest.)

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

Parameter destinationGroup ||| The name of the group to distribute the file across.

Parameter destinationLogicalName ||| The logical name of the file to create.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Parameter maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Parameter replicate ||| Whether to replicate the new file. Defaults to FALSE.

Parameter compress ||| Whether to compress the new file. Defaults to FALSE.

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

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

Return The DFU workunit id for the job.

FUNCTION : SprayXml

SprayXml
(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)

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

See fSprayXml

FUNCTION : fDespray

varstring	fDespray
(varstring logicalName, varstring destinationIP, varstring destinationPath, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE)	

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

Parameter logicalName ||| The name of the file to despray.

Parameter destinationIP ||| The IP of the target machine.

Parameter destinationPath ||| The path of the file to create on the destination machine.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Parameter maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Return The DFU workunit id for the job.

FUNCTION : Despray

	Despray
(varstring logicalName, varstring destinationIP, varstring destinationPath, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'), integer4 maxConnections=-1, boolean allowOverwrite=FALSE)	

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

See fDespray

FUNCTION : fCopy

varstring	fCopy
(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)	

Copies a distributed file to another distributed file.

Parameter sourceLogicalName ||| The name of the file to despray.

Parameter destinationGroup ||| The name of the group to distribute the file across.

Parameter destinationLogicalName ||| The logical name of the file to create.

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

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Parameter maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Parameter replicate ||| Should the copied file also be replicated on the destination? Defaults to FALSE

Parameter asSuperfile ||| 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 compress ||| Whether to compress the new file. Defaults to FALSE.

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

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

Return The DFU workunit id for the job.

FUNCTION : Copy

	Copy
	(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)

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

See fCopy

FUNCTION : fReplicate

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

Ensures the specified file is replicated to its mirror copies.

Parameter logicalName ||| The name of the file to replicate.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Return The DFU workunit id for the job.

FUNCTION : Replicate

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

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

See fReplicate

FUNCTION : fRemotePull

varstring	fRemotePull
(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)	

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

Parameter sourceLogicalName ||| The name of the file to despray.

Parameter destinationGroup ||| The name of the group to distribute the file across.

Parameter destinationLogicalName ||| The logical name of the file to create.

Parameter timeOut ||| 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 maxConnections ||| The maximum number of target nodes to write to concurrently. Defaults to 1.

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

Parameter replicate ||| Should the copied file also be replicated on the destination? Defaults to FALSE

Parameter asSuperfile ||| 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 compress ||| Whether to compress the new file. Defaults to FALSE.

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

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

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

Return The DFU workunit id for the job.

FUNCTION : RemotePull

	RemotePull
(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)	

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

See fRemotePull

FUNCTION : fMonitorLogicalFileName

varstring	fMonitorLogicalFileName
(varstring eventToFire, varstring name, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))	

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

Parameter shotCount ||| 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 espServerIpPort ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Return The DFU workunit id for the job.

FUNCTION : MonitorLogicalFileName

	MonitorLogicalFileName
(varstring eventToFire, varstring name, integer4 shotCount=1, varstring espServerIpPort=GETENV('ws_fs_server'))	

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

See fMonitorLogicalFileName

FUNCTION : fMonitorFile

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

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

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

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

Parameter shotCount ||| 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 espServerIpPort ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Return The DFU workunit id for the job.

FUNCTION : MonitorFile

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

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

See fMonitorFile

FUNCTION : WaitDfuWorkunit

varstring	WaitDfuWorkunit
(varstring wuid, integer4 timeOut=-1, varstring espServerIpPort=GETENV('ws_fs_server'))	

Waits for the specified DFU workunit to finish.

Parameter wuid ||| The dfu wfid to wait for.

Parameter timeOut ||| 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 ||| The url of the ESP file copying service. Defaults to the value of ws_fs_server in the environment.

Return A string containing the final status string of the DFU workunit.

FUNCTION : AbortDfuWorkunit

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

Aborts the specified DFU workunit.

Parameter wuid ||| The dfu wfid to abort.

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

FUNCTION : CreateSuperFile

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

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

Parameter superName ||| The logical name of the superfile.

Parameter sequentialParts ||| Whether the sub-files must be sequentially ordered. Default to FALSE.

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

FUNCTION : SuperFileExists

boolean	SuperFileExists
(varstring superName)	

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

Parameter superName ||| The logical name of the superfile.

Return Whether the file exists.

See FileExists

FUNCTION : DeleteSuperFile

	DeleteSuperFile
(varstring superName, boolean deletesub=FALSE)	

Deletes the superfile.

Parameter superName ||| The logical name of the superfile.

See FileExists

FUNCTION : GetSuperFileSubCount

unsigned4	GetSuperFileSubCount
(varstring superName)	

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

Parameter superName ||| The logical name of the superfile.

Return The number of sub-files within the superfile.

FUNCTION : GetSuperFileSubName

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

Up

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

Parameter superName ||| The logical name of the superfile.

Parameter fileNum ||| The 1-based position of the sub-file to return the name of.

Parameter absPath ||| Whether to prepend '~' to the name of the resulting logical file name.

Return The logical name of the selected sub-file.

FUNCTION : FindSuperFileSubName

unsigned4	FindSuperFileSubName
(varstring superName, varstring subName)	

Up

Returns the position of a file within a superfile.

Parameter superName ||| The logical name of the superfile.

Parameter subName ||| The logical name of the sub-file.

Return The 1-based position of the sub-file within the superfile.

FUNCTION : StartSuperFileTransaction

	StartSuperFileTransaction
()	

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

FUNCTION : AddSuperFile

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

Adds a file to a superfile.

Parameter superName ||| The logical name of the superfile.

Parameter subName ||| The name of the logical file to add.

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

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

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

FUNCTION : RemoveSuperFile

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

Removes a sub-file from a superfile.

Parameter superName ||| The logical name of the superfile.

Parameter subName ||| The name of the sub-file to remove.

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

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

FUNCTION : ClearSuperFile

	ClearSuperFile
(varstring superName, boolean del=FALSE)	

Removes all sub-files from a superfile.

Parameter superName ||| The logical name of the superfile.

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

FUNCTION : RemoveOwnedSubFiles

	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 ||| The logical name of the superfile.

FUNCTION : DeleteOwnedSubFiles

DeleteOwnedSubFiles
(varstring superName)

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

See RemoveOwnedSubFIles

FUNCTION : SwapSuperFile

SwapSuperFile
(varstring superName1, varstring superName2)

Swap the contents of two superfiles.

Parameter superName1 ||| The logical name of the first superfile.

Parameter superName2 ||| The logical name of the second superfile.

FUNCTION : ReplaceSuperFile

ReplaceSuperFile
(varstring superName, varstring oldSubFile, varstring newSubFile)

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

Parameter superName ||| The logical name of the superfile.

Parameter oldSubFile ||| The logical name of the sub-file to remove.

Parameter newSubFile ||| The logical name of the sub-file to replace within the superfile.

FUNCTION : FinishSuperFileTransaction

	FinishSuperFileTransaction
(boolean rollback=FALSE)	

Up

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.

FUNCTION : SuperFileContents

dataset(FsLogicalFileNameRecord)	SuperFileContents
(varstring superName, boolean recurse=FALSE)	

Up

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

Parameter superName ||| The logical name of the superfile.

Parameter recurse ||| Should the contents of child-superfiles be expanded. Default is FALSE.

Return A dataset containing the names of the sub-files.

FUNCTION : LogicalFileSuperOwners

dataset(FsLogicalFileNameRecord)	LogicalFileSuperOwners
(varstring name)	

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

Parameter name ||| The name of the logical file.

Return A dataset containing the names of the superfiles.

FUNCTION : LogicalFileSuperSubList

dataset(FsLogicalSuperSubRecord)	LogicalFileSuperSubList
()	

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

Return A dataset containing pairs of superName,subName for each component file.

FUNCTION : fPromoteSuperFileList

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

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 ||| 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 addHead ||| 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 delTail ||| Indicates whether to physically delete the contents moved out of the last superfile. The default is FALSE.

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

Parameter reverse ||| Reverse the order of processing the superfiles list, effectively 'demoting' instead of 'promoting' the sub-files. The default is FALSE.

Return A string containing a comma separated list of the previous sub-file contents of the emptied superfile.

FUNCTION : PromoteSuperFileList

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.

See fPromoteSuperFileList

---

math

IMPORTS

DESCRIPTIONS

MODULE : Math

	Math

Infinity | NaN | isInfinite | isNaN | isFinite | FMod | FMatch |

ATTRIBUTE : Infinity

REAL8	Infinity

Return a real "infinity" value.

ATTRIBUTE : NaN

REAL8	NaN

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

FUNCTION : isInfinite

BOOLEAN	isInfinite
(REAL8 val)	

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

Parameter val ||| The value to test.

FUNCTION : isNaN

BOOLEAN	isNaN
(REAL8 val)	

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

Parameter val ||| The value to test.

FUNCTION : isFinite

BOOLEAN	isFinite
(REAL8 val)	

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

Parameter val ||| The value to test.

FUNCTION : FMod

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 ||| The numerator.

Parameter denom ||| The numerator.

FUNCTION : FMatch

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 a ||| The first value.

Parameter b ||| The second value.

Parameter epsilon ||| The allowable margin of error.

Metaphone

IMPORTS

- lib_metaphone

DESCRIPTIONS

MODULE : Metaphone

	Metaphone

[Up]()

[primary]() | [secondary]() | [double]() |

FUNCTION : primary

String	primary
(STRING src)	

[Up]()

Returns the primary metaphone value

Parameter src ||| The string whose metphone is to be calculated.

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

FUNCTION : secondary

String	secondary
(STRING src)	

Returns the secondary metaphone value

Parameter src ||| The string whose metphone is to be calculated.

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

FUNCTION : double

String	double
(STRING src)	

Returns the double metaphone value (primary and secondary concatenated

Parameter src ||| The string whose metphone is to be calculated.

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

Metaphone3

IMPORTS

DESCRIPTIONS

str

IMPORTS

- lib_stringlib

DESCRIPTIONS

MODULE : Str

	Str

CompareIgnoreCase | EqualIgnoreCase | Find | FindCount | WildMatch | Contains | FilterOut | Filter | SubstituteIncluded | SubstituteExcluded | Translate | ToLowerCase | ToUpperCase | ToCapitalCase | ToTitleCase | Reverse | FindReplace | Extract | CleanSpaces | StartsWith | EndsWith | RemoveSuffix | ExtractMultiple | CountWords | SplitWords | CombineWords | EditDistance | EditDistanceWithinRadius | WordCount | GetNthWord | ExcludeFirstWord | ExcludeLastWord | ExcludeNthWord | FindWord | Repeat | ToHexPairs | FromHexPairs | EncodeBase64 | DecodeBase64 |

FUNCTION : CompareIgnoreCase

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 src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

See Str.EqualIgnoreCase

FUNCTION : EqualIgnoreCase

BOOLEAN	EqualIgnoreCase
(STRING src1, STRING src2)	

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

Parameter src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

See Str.CompareIgnoreCase

FUNCTION : Find

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 src ||| The string that is searched

Parameter sought ||| The string being sought.

Parameter instance ||| Which match instance are we interested in?

FUNCTION : FindCount

UNSIGNED4	FindCount
(STRING src, STRING sought)	

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

Parameter src ||| The string that is searched

Parameter sought ||| The string being sought.

FUNCTION : WildMatch

BOOLEAN	WildMatch
(STRING src, STRING _pattern, BOOLEAN ignore_case)	

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

Parameter src ||| The string that is being tested.

Parameter pattern ||| The pattern to match against.

Parameter ignore_case ||| Whether to ignore differences in case between characters

---

FUNCTION : Contains

BOOLEAN	Contains
(STRING src, STRING _pattern, BOOLEAN ignore_case)	

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 occurence in the pattern.

Parameter src ||| The string that is being tested.

Parameter pattern ||| The pattern to match against.

Parameter ignore_case ||| Whether to ignore differences in case between characters

---

FUNCTION : FilterOut

STRING	FilterOut
(STRING src, STRING filter)	

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

Parameter src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be excluded.

See Str.Filter

FUNCTION : Filter

STRING	Filter
(STRING src, STRING filter)	

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

Parameter src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

See Str.FilterOut

FUNCTION : SubstituteIncluded

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 src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

Parameter replace_char ||| The character to be substituted into the result.

See Std.Str.Translate, Std.Str.SubstituteExcluded

FUNCTION : SubstituteExcluded

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 src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

Parameter replace_char ||| The character to be substituted into the result.

See Std.Str.SubstituteIncluded

FUNCTION : Translate

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 src ||| The string that is being tested.

Parameter search ||| The string containing the set of characters to be included.

Parameter replacement ||| The string containing the characters to act as replacements.

See Std.Str.SubstituteIncluded

FUNCTION : ToLowerCase

STRING	ToLowerCase
(STRING src)	

Returns the argument string with all upper case characters converted to lower case.

Parameter src ||| The string that is being converted.

FUNCTION : ToUpperCase

STRING	ToUpperCase
(STRING src)	

Return the argument string with all lower case characters converted to upper case.

Parameter src ||| The string that is being converted.

FUNCTION : ToCapitalCase

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 ||| The string that is being converted.

FUNCTION : ToTitleCase

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 ||| The string that is being converted.

FUNCTION : Reverse

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 ||| The string that is being reversed.

FUNCTION : FindReplace

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 src ||| The string that is being transformed.

Parameter sought ||| The string to be replaced.

Parameter replacement ||| The string to be substituted into the result.

FUNCTION : Extract

STRING	Extract
(STRING src, UNSIGNED4 instance)	

Returns the nth element from a comma separated string.

Parameter src ||| The string containing the comma separated list.

Parameter instance ||| Which item to select from the list.

FUNCTION : CleanSpaces

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 ||| The string to be cleaned.

FUNCTION : StartsWith

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 ||| The string being searched in.

Parameter prefix ||| The prefix to search for.

FUNCTION : EndsWith

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 src ||| The string being searched in.

Parameter suffix ||| The prefix to search for.

FUNCTION : RemoveSuffix

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 src ||| The string being searched in.

Parameter suffix ||| The prefix to search for.

FUNCTION : ExtractMultiple

STRING	ExtractMultiple
(STRING src, UNSIGNED8 mask)	

Returns a string containing a list of elements from a comma separated string.

Parameter src ||| The string containing the comma separated list.

Parameter mask ||| A bitmask of which elements should be included. Bit 0 is item1, bit1 item 2 etc.

FUNCTION : CountWords

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 src ||| The string being searched in.

Parameter separator ||| The string used to separate words

Parameter allow_blank ||| Indicates if empty/blank string items are included in the results.

FUNCTION : SplitWords

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 src ||| The string being searched in.

Parameter separator ||| The string used to separate words

Parameter allow_blank ||| Indicates if empty/blank string items are included in the results.

FUNCTION : CombineWords

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 ||| The set of strings to be combined.

Parameter separator ||| The string used to separate words.

FUNCTION : EditDistance

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 _left ||| The first string to be compared.

Parameter _right ||| The second string to be compared.

Return The minimum edit distance between the two strings.

FUNCTION : EditDistanceWithinRadius

BOOLEAN	EditDistanceWithinRadius
(STRING _left, STRING _right, UNSIGNED4 radius)	

Returns true if the minimum edit distance between the two strings is with a specific range. The two strings are trimmed before comparing.

Parameter _left ||| The first string to be compared.

Parameter _right ||| The second string to be compared.

Parameter radius ||| The maximum edit distance that is accepable.

Return Whether or not the two strings are within the given specified edit distance.

FUNCTION : WordCount

UNSIGNED4	WordCount
(STRING text)	

Returns the number of words in the string. Words are separated by one or more spaces.

Parameter text ||| The string to be broken into words.

Return The number of words in the string.

FUNCTION : GetNthWord

STRING	GetNthWord
(STRING text, UNSIGNED4 n)	

Returns the n-th word from the string. Words are separated by one or more spaces.

Parameter text ||| The string to be broken into words.

Parameter n ||| Which word should be returned from the function.

Return The number of words in the string.

FUNCTION : ExcludeFirstWord

	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 ||| The string to be broken into words.

Return The string excluding the first word.

FUNCTION : ExcludeLastWord

	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 ||| The string to be broken into words.

Return The string excluding the last word.

FUNCTION : ExcludeNthWord

	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 ||| The string to be broken into words.

Parameter n ||| Which word should be returned from the function.

Return The string excluding the nth word.

FUNCTION : FindWord

BOOLEAN	FindWord
(STRING src, STRING word, BOOLEAN ignore_case=FALSE)	

Tests if the search string contains the supplied word as a whole word.

Parameter src ||| The string that is being tested.

Parameter word ||| The word to be searched for.

Parameter ignore_case ||| Whether to ignore differences in case between characters.

FUNCTION : Repeat

STRING	Repeat
(STRING text, UNSIGNED4 n)	

FUNCTION : ToHexPairs

STRING	ToHexPairs
(DATA value)	

FUNCTION : FromHexPairs

DATA	FromHexPairs
(STRING hex_pairs)	

FUNCTION : EncodeBase64

STRING	EncodeBase64
(DATA value)	

FUNCTION : DecodeBase64

DATA	DecodeBase64
(STRING value)	

Uni

IMPORTS

- lib_unicodelib

DESCRIPTIONS

MODULE : Uni

	Uni

Up

FilterOut | Filter | SubstituteIncluded | SubstituteExcluded | Find | FindWord | LocaleFind | LocaleFindAtStrength | Extract | ToLowerCase | ToUpperCase | ToTitleCase | LocaleToLowerCase | LocaleToUpperCase | LocaleToTitleCase | CompareIgnoreCase | CompareAtStrength | LocaleCompareIgnoreCase | LocaleCompareAtStrength | Reverse | FindReplace | LocaleFindReplace | LocaleFindAtStrengthReplace | CleanAccents | CleanSpaces | WildMatch | Contains | EditDistance | EditDistanceWithinRadius | WordCount | GetNthWord |

FUNCTION : FilterOut

unicode	FilterOut
(unicode src, unicode filter)	

Up

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

Parameter src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be excluded.

See Std.Uni.Filter

FUNCTION : Filter

unicode	Filter
(unicode src, unicode filter)	

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

Parameter src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

See Std.Uni.FilterOut

FUNCTION : SubstituteIncluded

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 src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

Parameter replace_char ||| The character to be substituted into the result.

See Std.Uni.SubstituteOut

FUNCTION : SubstituteExcluded

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 src ||| The string that is being tested.

Parameter filter ||| The string containing the set of characters to be included.

Parameter replace_char ||| The character to be substituted into the result.

See Std.Uni.SubstituteIncluded

FUNCTION : Find

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 src ||| The string that is searched

Parameter sought ||| The string being sought.

Parameter instance ||| Which match instance are we interested in?

FUNCTION : FindWord

BOOLEAN	FindWord
(UNICODE src, UNICODE word, BOOLEAN ignore_case=FALSE)	

Tests if the search string contains the supplied word as a whole word.

Parameter src ||| The string that is being tested.

Parameter word ||| The word to be searched for.

Parameter ignore_case ||| Whether to ignore differences in case between characters.

FUNCTION : LocaleFind

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 src ||| The string that is searched

Parameter sought ||| The string being sought.

Parameter instance ||| Which match instance are we interested in?

Parameter locale_name ||| The locale to use for the comparison

---

FUNCTION : LocaleFindAtStrength

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 src ||| The string that is searched

Parameter sought ||| The string being sought.

Parameter instance ||| Which match instance are we interested in?

Parameter locale_name ||| The locale to use for the comparison

Parameter strength ||| 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 cantellation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

---

FUNCTION : Extract

unicode	Extract
(unicode src, unsigned4 instance)	

Up

Returns the nth element from a comma separated string.

Parameter src ||| The string containing the comma separated list.

Parameter instance ||| Which item to select from the list.

FUNCTION : ToLowerCase

unicode	ToLowerCase
(unicode src)	

Up

Returns the argument string with all upper case characters converted to lower case.

Parameter src ||| The string that is being converted.

FUNCTION : ToUpperCase

unicode	ToUpperCase
(unicode src)	

Return the argument string with all lower case characters converted to upper case.

Parameter src ||| The string that is being converted.

FUNCTION : ToTitleCase

unicode	ToTitleCase
(unicode src)	

Returns the upper case variant of the string using the rules for a particular locale.

Parameter src ||| The string that is being converted.

Parameter locale_name ||| The locale to use for the comparison

---

FUNCTION : LocaleToLowerCase

unicode	LocaleToLowerCase
(unicode src, varstring locale_name)	

Returns the lower case variant of the string using the rules for a particular locale.

Parameter src ||| The string that is being converted.

Parameter locale_name ||| The locale to use for the comparison

---

FUNCTION : LocaleToUpperCase

unicode	LocaleToUpperCase
(unicode src, varstring locale_name)	

Returns the upper case variant of the string using the rules for a particular locale.

Parameter src ||| The string that is being converted.

Parameter locale_name ||| The locale to use for the comparison

---

FUNCTION : LocaleToTitleCase

unicode	LocaleToTitleCase
(unicode src, varstring locale_name)	

Returns the upper case variant of the string using the rules for a particular locale.

Parameter src ||| The string that is being converted.

Parameter locale_name ||| The locale to use for the comparison

---

FUNCTION : CompareIgnoreCase

integer4	CompareIgnoreCase
(unicode src1, unicode src2)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

Parameter src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

See Std.Uni.CompareAtStrength

FUNCTION : CompareAtStrength

integer4	CompareAtStrength
(unicode src1, unicode src2, integer1 strength)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

Parameter src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

Parameter strength ||| 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 cantellation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

See Std.Uni.CompareAtStrength

FUNCTION : LocaleCompareIgnoreCase

integer4	LocaleCompareIgnoreCase
(unicode src1, unicode src2, varstring locale_name)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

Parameter src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

Parameter locale_name ||| The locale to use for the comparison

See Std.Uni.CompareAtStrength

FUNCTION : LocaleCompareAtStrength

integer4	LocaleCompareAtStrength
(unicode src1, unicode src2, varstring locale_name, integer1 strength)	

Compares the two strings case insensitively. Equivalent to comparing at strength 2.

Parameter src1 ||| The first string to be compared.

Parameter src2 ||| The second string to be compared.

Parameter locale_name ||| The locale to use for the comparison

Parameter strength ||| 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 cantellation marks 5 differentiates between all strings whose canonically decomposed forms (NFDNormalization Form D) are non-identical

---

FUNCTION : Reverse

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 ||| The string that is being reversed.

FUNCTION : FindReplace

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 src ||| The string that is being transformed.

Parameter sought ||| The string to be replaced.

Parameter replacement ||| The string to be substituted into the result.

FUNCTION : LocaleFindReplace

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 src ||| The string that is being transformed.

Parameter sought ||| The string to be replaced.

Parameter replacement ||| The string to be substituted into the result.

Parameter locale_name ||| The locale to use for the comparison

FUNCTION : LocaleFindAtStrengthReplace

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 src ||| The string that is being transformed.

Parameter sought ||| The string to be replaced.

Parameter replacement ||| The string to be substituted into the result.

Parameter locale_name ||| The locale to use for the comparison

Parameter strength ||| The strength of the comparison

FUNCTION : CleanAccents

unicode	CleanAccents
(unicode src)	

Returns the source string with all accented characters replaced with unaccented.

Parameter src ||| The string that is being transformed.

FUNCTION : CleanSpaces

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 ||| The string to be cleaned.

FUNCTION : WildMatch

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 src ||| The string that is being tested.

Parameter pattern ||| The pattern to match against.

Parameter ignore_case ||| Whether to ignore differences in case between characters

---

FUNCTION : Contains

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 occurence in the pattern.

Parameter src ||| The string that is being tested.

Parameter pattern ||| The pattern to match against.

Parameter ignore_case ||| Whether to ignore differences in case between characters

FUNCTION : EditDistance

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 _left ||| The first string to be compared.

Parameter _right ||| The second string to be compared.

Parameter localname ||| The locale to use for the comparison. Defaults to ".

Return The minimum edit distance between the two strings.

FUNCTION : EditDistanceWithinRadius

BOOLEAN	EditDistanceWithinRadius
(unicode _left, unicode _right, unsigned4 radius, varstring localename = ")	

Returns true if the minimum edit distance between the two strings is with a specific range. The two strings are trimmed before comparing.

Parameter _left ||| The first string to be compared.

Parameter _right ||| The second string to be compared.

Parameter radius ||| The maximum edit distance that is accepable.

Parameter localname ||| The locale to use for the comparison. Defaults to ".

Return Whether or not the two strings are within the given specified edit distance.

FUNCTION : WordCount

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 ||| The string to be broken into words.

Parameter localname ||| The locale to use for the break semantics. Defaults to ".

Return The number of words in the string.

FUNCTION : GetNthWord

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 ||| The string to be broken into words.

Parameter n ||| Which word should be returned from the function.

Parameter localname ||| The locale to use for the break semantics. Defaults to ".

Return The number of words in the string.

---

system

Table of Contents

system.Debug

<u>**IMPORTS**</u>

- lib_parselib

<u>**DESCRIPTIONS**</u>

system.Email

- lib_fileservices

DESCRIPTIONS

system.Job

DESCRIPTIONS

system.Log

IMPORTS

- lib_logging

DESCRIPTIONS

system.Thorlib

<u>IMPORTS</u>

<u>DESCRIPTIONS</u>

system.Util

IMPORTS

- lib_fileservices

DESCRIPTIONS

system.Workunit

- lib_WORKUNITSERVICES