

Referência de Biblioteca Padrão

Equipe de documentação de Boca Raton



Referência de Biblioteca Padrão

Equipe de documentação de Boca Raton

Copyright © 2023 HPCC Systems®. All rights reserved

Sua opinião e comentários sobre este documento são muito bem-vindos e podem ser enviados por e-mail para <docfeedback@hpccsystems.com>

Inclua a frase **Feedback sobre documentação** na linha de assunto e indique o nome do documento, o número das páginas e número da versão atual no corpo da mensagem.

LexisNexis e o logotipo Knowledge Burst são marcas comerciais registradas da Reed Elsevier Properties Inc., usadas sob licença.

HPCC Systems® é uma marca registrada da LexisNexis Risk Data Management Inc.

Os demais produtos, logotipos e serviços podem ser marcas comerciais ou registradas de suas respectivas empresas.

Todos os nomes e dados de exemplo usados neste manual são fictícios. Qualquer semelhança com pessoas reais, vivas ou mortas, é mera coincidência.

2023 Version 8.10.70-1

<i>Arquivos lógicos</i>	8
CompareFiles	9
DeleteLogicalFile	10
LogicalFileList	11
FileExists	12
ForeignLogicalFileName	13
GetFileDescription	14
GetLogicalFileAttribute	15
ProtectLogicalFile	16
RenameLogicalFile	17
SetFileDescription	18
SetReadOnly	19
VerifyFile	20
<i>Superfiles</i>	21
CreateSuperFile	22
SuperFileExists	23
DeleteSuperFile	24
GetSuperFileSubCount	25
GetSuperFileSubName	26
LogicalFileSuperOwners	27
LogicalFileSuperSubList	28
SuperFileContents	29
FindSuperFileSubName	30
StartSuperFileTransaction	31
AddSuperFile	32
RemoveSuperFile	33
ClearSuperFile	34
RemoveOwnedSubFiles	35
SwapSuperFile	36
ReplaceSuperFile	37
PromoteSuperFileList	38
FinishSuperFileTransaction	39
<i>Arquivos Externo</i>	40
ExternalLogicalFileName	41
MoveExternalFile	42
DeleteExternalFile	43
CreateExternalDirectory	44
RemoteDirectory	45
<i>Navegação por Arquivos</i>	46
SetColumnMapping	47
GetColumnMapping	49
AddFileRelationship	50
FileRelationshipList	52
RemoveFileRelationship	53
<i>Movimentação de Arquivos</i>	54
DfuPlusExec	55
AbortDfuWorkunit	56
Copy	57
DeSpray	59
RemotePull	60
Replicate	62
SprayFixed	63
SprayDelimited / SprayVariable	65
SprayXML	67

SprayJson	69
WaitDfuWorkunit	71
SetExpireDays	72
GetExpireDays	73
ClearExpireDays	74
<i>Manipulação de String</i>	75
CleanAccents	76
CleanSpaces	77
CompareAtStrength	78
CompareIgnoreCase	79
Contains	80
CountWords	81
DecodeBase64	82
EditDistance	83
EditDistanceWithinRadius	84
EncodeBase64	85
EndsWith	86
EqualIgnoreCase	87
ExcludeFirstWord	88
ExcludeLastWord	89
ExcludeNthWord	90
Extract	91
ExtractMultiple	92
Filter	93
FilterOut	94
Find	95
FindCount	96
FindAtStrength	97
FindAtStrengthReplace	98
FindReplace	99
FindWord	100
FromHexPairs	101
GetNthWord	102
RemoveSuffix	103
Repeat	104
Reverse	105
SplitWords	106
SubstituteExcluded	107
SubstituteIncluded	108
StartsWith	109
ToHexPairs	110
ToLowerCase	111
ToTitleCase	112
ToUpperCase	113
Translate	114
Version	115
WildMatch	116
WordCount	117
Suporte a Metaphone	118
Primary	119
Secondary	120
Double	121
<i>Suporte a Criptografia</i>	122
Visão geral da Biblioteca Criptográfica	123

SupportedHashAlgorithms	124
SupportedSymmetricCipherAlgorithms	125
Algoritmos de Chave Pública Suportados	126
Módulo Hashing	127
Hash	128
Módulo SymmetricEncryption	129
Encrypt (Symmetric)	130
Decrypt (Symmetric)	131
Módulo PublicKeyEncryption	132
Encrypt (PKE)	133
Decrypt (PKE)	134
Sign (PKE)	135
VerifySignature (PKE)	136
Módulo PublicKeyEncryptionFromBuffer	137
Encrypt (PKE From Buffer)	139
Decrypt (PKE From Buffer)	140
Sign (PKE (PKE From Buffer))	141
VerifySignature (PKE From Buffer)	142
PublicKeyEncryptionFromLFN Module	143
Encrypt (PKE From LFN)	145
Decrypt (PKE From LFN)	147
Sign (PKE From LFN)	149
VerifySignature (PKE From LFN)	151
<i>Manipulação de Data e Hora</i>	153
Date Data Types	154
Time Data Types	155
Year	156
Month	157
Day	158
Hour	159
Minute	160
Second	161
DateFromParts	162
TimeFromParts	163
IsLeapYear	164
IsDateLeapYear	165
IsValidDate	166
IsValidTime	167
IsValidGregorianDate	168
FromGregorianYMD	169
ToGregorianYMD	170
FromStringToDate	171
Today	172
CurrentDate	173
CurrentTime	174
DayOfWeek	175
DayOfYear	176
DaysBetween	177
MonthsBetween	178
AdjustDate	179
AdjustCalendar	180
MonthWeekNumFromDate	181
YearWeekNumFromDate	182
UniqueTZAbbreviations	183

UniqueTZLocations	184
TZDataForLocation	185
FindTZData	186
SecondsBetweenTZ	187
AdjustTimeTZ	188
ToLocalTime	189
ToUTCTime	190
AppendTZOffset	191
AppendTZAdjustedTime	193
<i>Manipulação de Cluster</i>	195
Node	196
Nodes	197
LogicalToPhysical	198
DaliServer	199
Group	200
GetExpandLogicalFileName	201
<i>Manipulação de Job</i>	202
WUID	203
Target	204
Name	205
User	206
OS	207
Platform	208
LogString	209
<i>Monitoramento de Arquivo</i>	210
MonitorFile	211
MonitorLogicalFileName	213
<i>Logging</i>	215
dbglog	216
addWorkunitInformation	217
addWorkunitWarning	218
addWorkunitError	219
getGlobalId	220
getLocalId	221
generateGloballyUniqueId	222
<i>Auditoria</i>	223
Audit	224
<i>Utilitários</i>	225
GetHostName	226
ResolveHostName	227
GetUniqueInteger	228
GetEspUrl	229
PlatformVersionCheck	230
<i>Depuração</i>	231
GetParseTree	232
GetXMLParseTree	233
Sleep	234
msTick	235
<i>E-mail</i>	236
SendEmail	237
SendEmailAttachData	238
SendEmailAttachText	239
<i>Serviços da Workunit</i>	240
WorkunitExists	241

WorkunitList	242
SetWorkunitAppValue	244
WUIDonDate	245
WUIDdaysAgo	246
WorkunitTimeStamps	247
WorkunitMessages	248
WorkunitFilesRead	249
WorkunitFilesWritten	250
WorkunitTimings	251
<i>Suporte a BLAS</i>	252
Tipos:	253
ICellFunc	254
Apply2Cells	255
dasum	256
daxpy	257
dgemm	258
dgetf2	259
dpotf2	260
dscal	261
dsyrk	262
dtrsm	263
extract_diag	264
extract_tri	265
make_diag	266
make_vector	267
trace	268
<i>Suporte a Math</i>	269
Infinity	270
NaN	271
isInfinite	272
isNaN	273
isFinite	274
FMod	275
FMatch	276

Arquivos lógicos

CompareFiles

STD.File.CompareFiles(*file1*, *file2* [, *logicalonly*] [, *usecrcs*])

<i>file1</i>	Uma string terminada por nulo que contém o nome lógico do primeiro arquivo.
<i>file2</i>	Uma string terminada por nulo que contém o nome lógico do segundo arquivo.
<i>logicalonly</i>	Opcional. Um indicador booleano TRUE/FALSE que, quando TRUE, não compara informações físicas do disco, mas apenas as informações lógicas no armazenamento de dados do sistema (Dali). Se omitido, o padrão é TRUE.
<i>usecrcs</i>	Opcional. Um indicador booleano TRUE/FALSE que, quando TRUE, indica a comparação de CRCs físicos de todas as partes em disco. Essa comparação pode ser lenta em grandes arquivos. Se omitido, o padrão é FALSE.
Return:	CompareFiles retorna um valor INTEGER4.

A função **CompareFiles** compara *file1* com *file2* e retorna os seguintes valores:

0	<i>O conteúdo de file1 e file2 corresponde exatamente</i>
1	<i>O conteúdo de file1 e file2 corresponde exatamente, mas file1 é mais recente que file2</i>
-1	<i>O conteúdo de file1 e file2 corresponde, mas file2 é mais recente que file1</i>
2	<i>O conteúdo de file1 e file2 não corresponde e file1 é mais recente que file2</i>
-2	<i>O conteúdo de file1 e file2 não corresponde e file2 é mais recente que file1</i>

Exemplo:

```
A := STD.File.CompareFiles('Fred1', 'Fred2');
```

DeleteLogicalFile

STD.File.DeleteLogicalFile(*filename* [, *ifexists*])

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>ifexists</i>	Opcional. Um valor booleano que indica se um erro deverá ser publicado se o <i>filename</i> não existir. Se omitido, o padrão é FALSE.

A função **DeleteLogicalFile** remove o arquivo nomeado do disco.

Exemplo:

```
A := STD.File.DeleteLogicalFile('Fred');
```

LogicalFileList

STD.File.LogicalFileList([*pattern*] [, *includenormal*] [, *includesuper*] [, *unknownszero*] [, *foreigndali*])

<i>pattern</i>	Opcional. Uma string terminada por nulo que contém a máscara dos arquivos a serem listados. Se omitida, o padrão é "*" (todos os arquivos).
<i>includenormal</i>	Opcional. Um indicador booleano que indica se arquivos "normais" devem ser incluídos. Se omitido, o padrão é TRUE.
<i>includesuper</i>	Opcional. Um indicador booleano que indica se superarquivos devem ser incluídos. Se omitido, o padrão é FALSE.
<i>unknownszero</i>	Opcional. Um indicador booleano que indica que tamanhos de arquivos desconhecidos devem ser definidos como zero (0) em vez de um negativo (-1). Se omitido, o padrão é FALSE.
<i>foreigndali</i>	Opcional. O endereço IP do Dali externo usado para resolver o arquivo. Um valor em branco significa que o arquivo é resolvido localmente. Se omitido, o padrão é em branco.
Return:	LogicalFileList retorna um dataset no seguinte formato:

```
EXPORT FsLogicalFileNameRecord := RECORD
  STRING name;
END;

EXPORT FsLogicalFileInfoRecord := RECORD(FsLogicalFileNameRecord)
  BOOLEAN superfile;
  UNSIGNED8 size;
  UNSIGNED8 rowcount;
  STRING19 modified;
  STRING owner;
  STRING cluster;
END;
```

A função **LogicalFileList** retorna uma lista dos arquivos lógicos nos arquivos do ambiente como um dataset no formato listado acima.

Exemplo:

```
OUTPUT(STD.File.LogicalFileList());
//returns all normal files

OUTPUT(STD.File.LogicalFileList(,FALSE,TRUE));
//returns all SuperFiles
```

FileExists

STD.File.FileExists(*filename* [, *physicalcheck*])

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>physicalcheck</i>	Opcional. Um booleano TRUE/FALSE que indica se a existência física do <i>filename</i> no disco deve ser verificada. Se omitido, o padrão é FALSE.
Return:	FileExists retorna um valor BOOLEAN

A função **FileExists** retorna TRUE se o *filename* especificado está presente no utilitário de arquivos distribuídos (DFU). Se *physicalcheck* estiver definido como TRUE, a presença física do arquivo no disco também será verificada.

Exemplo:

```
A := STD.File.FileExists('~CLASS::RT::IN::People');
```

Ver também: SuperFileExists

ForeignLogicalFileName

STD.File.ForeignLogicalFileName(*filename* [, *foreigndali*] [, *absolutePath*])

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>foreigndali</i>	Uma string terminada por nulo que contém o endereço IP da Dali externo. Se omitido, o <i>filename</i> será interpretado como um nome de arquivo lógico externo e convertido em um nome de arquivo lógico local.
<i>absolutePath</i>	Opcional. Um TRUE/FALSE booleano para indicar se um til (~) deve ser acrescentado ao início do nome do arquivo lógico externo. Se omitido, o padrão é FALSE.
Return:	ForeignLogicalFileName retorna um valor VARSTRING (terminado com nulo).

A função **ForeignLogicalFileName** retornará um nome de arquivo lógico externo (se o parâmetro *foreigndali* estiver presente) ou um nome de arquivo lógico local.

Exemplo:

```
sf := '~thor_data400::BASE::Business_Header';  
ff := STD.File.ForeignLogicalFileName(sf, '10.150.29.161', true);  
//results in: ~foreign::10.150.29.161::thor_data400::base::business_header  
lf := STD.File.ForeignLogicalFileName(ff, '', true);  
//results in: ~thor_data400::base::business_header
```

GetFileDescription

STD.File.GetFileDescription(*filename*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
Return:	GetFileDescription retorna um valor VARSTRING (terminado em nulo).

A função **GetFileDescription** retorna uma string que contém as informações da descrição armazenadas pelo DFU sobre o *filename* especificado. Essa descrição é definida usando o ECL Watch ou usando a função STD.File.SetFileDescription.

Exemplo:

```
A := STD.File.GetFileDescription('Fred');
```

GetLogicalFileAttribute

STD.File.GetLogicalFileAttribute(*logicalfilename*, *attrname*)

<i>logicalfilename</i>	Uma string terminada por nulo que contém o nome do arquivo lógico, como conhecido pelo DFU.
<i>attrname</i>	Uma string terminada por nulo que contém o nome do atributo de arquivo a ser retornado. Os valores possíveis são recordSize, recordCount, size, clusterName, directory, owner, description, ECL, partmask, numparts, name, modified, format, job, checkSum, kind, csvSeparate, csvTerminate, headerLength, footerLength, rowTag, workunit, accessed, expireDays, maxRecordSize, csvQuote, blockCompressed, compressedSize, fileCrc, formatCrc ou protected. O valor faz distinção entre maiúsculas e minúsculas.
Return:	GetHostName retorna um valor VARSTRING (terminado por nulo).

A função **GetLogicalFileAttribute** retorna o valor de *attrname* para o *logicalfilename* especificado.

Exemplo:

```
IMPORT STD;
file := '~ certification::full_test_distributed';

OUTPUT(STD.File.GetLogicalFileAttribute(file, 'recordSize'), NAMED('recordSize'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'recordCount'), NAMED('recordCount'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'size'), NAMED('size'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'clusterName'), NAMED('clusterName'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'directory'), NAMED('directory'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'numparts'), NAMED('numparts'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'owner'), NAMED('owner'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'description'), NAMED('description'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'ECL'), NAMED('ECL'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'partmask'), NAMED('partmask'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'name'), NAMED('name'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'modified'), NAMED('modified'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'protected'), NAMED('protected'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'format'), NAMED('format'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'job'), NAMED('job'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'checkSum'), NAMED('checkSum'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'kind'), NAMED('kind'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'csvSeparate'), NAMED('csvSeparate'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'csvTerminate'), NAMED('csvTerminate'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'csvEscape'), NAMED('csvEscape'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'headerLength'), NAMED('headerLength'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'footerLength'), NAMED('footerLength'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'rowtag'), NAMED('rowtag'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'workunit'), NAMED('workunit'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'accessed'), NAMED('accessed'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'expireDays'), NAMED('expireDays'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'maxRecordSize'), NAMED('maxRecordSize'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'csvQuote'), NAMED('csvQuote'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'blockCompressed'), NAMED('blockCompressed'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'compressedSize'), NAMED('compressedSize'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'fileCrc'), NAMED('fileCrc'));
OUTPUT(STD.File.GetLogicalFileAttribute(file, 'formatCrc'), NAMED('formatCrc'));
```

ProtectLogicalFile

STD.File.ProtectLogicalFile(*logicalfilename* [, *value*])

<i>logicalfilename</i>	Uma string terminada por nulo que contém o nome do arquivo lógico, como conhecido pelo DFU.
<i>value</i>	Opcional. Um indicador booleano que indica se o arquivo deve ser protegido ou desprotegido. Se omitido, o padrão é TRUE.

A função **ProtectLogicalFile** ativa e desativa a proteção do *logicalfilename* especificado.

Exemplo:

```
IMPORT STD;
file := '~class::bmf::join::halfkeyed';

STD.File.ProtectLogicalFile(file);           //protect
STD.File.ProtectLogicalFile(file, FALSE);    //unprotect
```


RenameLogicalFile

STD.File.RenameLogicalFile(*filename*, *newname*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico atual do arquivo.
<i>newname</i>	Uma string terminada por nulo que contém o novo nome lógico do arquivo.

A função **RenameLogicalFile** altera o *filename* lógico para *newname*.

Exemplo:

```
A := STD.File.RenameLogicalFile('Fred', 'Freddie');
```

SetFileDescription

STD.File.SetFileDescription(*filename* , *value*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>value</i>	Uma string terminada por nulo que contém a descrição a ser colocada no arquivo.

A função **SetFileDescription** altera as informações da descrição armazenadas pelo DFU sobre o *filename* especificado no *value* especificado. Essa descrição é vista usando o ECL Watch ou a função STD.File.GetFileDescription.

Exemplo:

```
A := STD.File.SetFileDescription('Fred','All the Freds in the world');
```

SetReadOnly

STD.File.SetReadOnly(*filename* , *flag*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>flag</i>	Um valor booleano que indica a forma de definir o atributo somente leitura de <i>filename</i> .

A função **SetReadOnly** alterna o atributo somente leitura do nome de arquivo. Se o indicador for *TRUE* , o atributo somente leitura é ativado.

Exemplo:

```
A := STD.File.SetReadOnly('Fred',TRUE);  
//set read only flag on
```

VerifyFile

STD.File.VerifyFile(*file*, *usecrcs*)

<i>file</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>usecrcs</i>	Um indicador booleano TRUE/FALSE que, quando TRUE, indica a comparação de CRCs físicos de todas as partes em disco. Essa comparação pode ser lenta em grandes arquivos.
Return:	VerifyFile retorna um valor VARSTRING.

A função **VerifyFile** verifica as informações do armazenamento de dados do sistema (Dali) do *arquivo* em relação às partes físicas no disco e retorna os seguintes valores:

OK	As partes dos arquivos correspondem às informações do armazenamento de dados
Could not find file: <i>filename</i>	O <i>nome de arquivo</i> lógico não foi encontrado
Could not find file: <i>partname</i>	O <i>partname</i> não foi encontrado
Modified time differs for: <i>partname</i>	O <i>partname</i> tem uma marcação de horário diferente
File size differs for: <i>partname</i>	O <i>partname</i> tem um tamanho de arquivo
File CRC differs for: <i>partname</i>	O <i>partname</i> tem um CRC diferente

Exemplo:

```
A := STD.File.VerifyFile('Fred1', TRUE);
```

Superfiles

CreateSuperFile

STD.File.CreateSuperFile(*superfile* [, *sequentialparts*] [, *allowExist*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>sequentialparts</i>	Opcional. Um valor booleano que indica se os subarquivos devem ser ordenados sequencialmente. Se omitido, o padrão é FALSE.
<i>allowExist</i>	Opcional. Um valor booleano que indica se um erro deverá ser publicado se o <i>superarquivo</i> já existir. Se TRUE, nenhum erro é publicado. Se omitido, o padrão é FALSE.
Return:	Nulo

A função **CreateSuperFile** cria um *superarquivo* vazio. Essa função não é incluída em uma transação de superarquivo.

O parâmetro *sequentialparts* definido como TRUE controla o caso pouco comum em que a numeração lógica de subarquivos deve ser sequencial (por exemplo, quando todos os subarquivos já estão classificados globalmente). Com *sequentialparts* FALSE (o padrão), as partes dos subarquivos são intercaladas para que possam ser encontradas localmente.

Por exemplo, se em um cluster de 4 nós existirem 3 arquivos (A, B e C), as partes serão as seguintes:

A_1_of_4, B_1_of_4 e C_1_of_4 estão no nó 1

A_1_of_4, B_1_of_4 e C_1_of_4 estão no nó 2

A_1_of_4, B_1_of_4 e C_1_of_4 estão no nó 3

A_1_of_4, B_1_of_4 e C_1_of_4 estão no nó 4

A leitura do superarquivo criado no Thor com *sequentialparts* FALSE se dará na seguinte ordem::

[A1,B1,C1,] [A2,B2,C2,] [A3,B3,C3,] [A4,B4,C4]

para que as leituras sejam todas locais (ou seja, A1, B1 e C1 no nó 1, etc.). Definir *sequentialparts* como TRUE resultará na leitura das partes na ordem do subarquivo, da seguinte forma:

[A1,A2,A3,] [A4,B1,B2] [B3,B4,C1,] [C2,C3,C4]

para que a ordem global de A, B, C e D seja mantida. No entanto, nem todas as partes poderão ser lidas localmente (por exemplo, A2 e A3 serão lidas na parte 1). Por isso, definir *sequentialparts* como verdadeiro é muito menos eficiente. De qualquer forma, como não é comum ter arquivos particionados em ordem, essa opção é raramente definida.

Exemplo:

```
STD.File.CreateSuperFile('~CLASS::RT::IN::SF1',,1);  
//This is the same but uses named parameter  
STD.File.CreateSuperFile('~CLASS::RT::IN::SF1',allowExist := 1);
```

SuperFileExists

STD.File.SuperFileExists(*filename*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
Return:	SuperFileExists retorna um valor BOOLEAN

A função **SuperFileExists** retorna TRUE se o *filename* está presente no utilitário de arquivos distribuídos (DFU) e é um superarquivo. A função retorna FALSE se o *filename* existe, mas não é um superarquivo (em outras palavras, é um DATASET normal. Use a função STD.File.FileExists para detectar sua presença ou ausência).

Essa função não é incluída em uma transação de superarquivo.

Exemplo:

```
A := STD.File.SuperFileExists('~CLASS::RT::IN::SF1');
```

Ver também: FileExists

DeleteSuperFile

STD.File.DeleteSuperFile(*superfile* [, *subdeleteflag*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subdeleteflag</i>	Um valor booleano que indica se os subarquivos devem ser excluídos. Se omitido, o padrão é FALSE. Essa opção não deverá ser usada se o superarquivo contiver qualquer arquivo ou superarquivo estrangeiro.
Return:	Nulo

A função **DeleteSuperFile** remove o *superarquivo*.

Essa função não é incluída em uma transação do superarquivo.

Exemplo:

```
STD.File.DeleteSuperFile( '~CLASS::RT::IN::SF1' );
```


GetSuperFileSubCount

STD.File.GetSuperFileSubCount(*superfile*)

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
Return:	GetUniqueInteger retorna um valor INTEGER4.

A função **GetSuperFileSubCount** retorna o número de subarquivos que compõem o *superfile*.

Essa função não é incluída em uma transação do superarquivo.

Exemplo:

```
A := STD.File.GetSuperFileSubCount ( '~CLASS::RT::IN::SF1' );
```

GetSuperFileSubName

STD.File.GetSuperFileSubName(*superfile*, *subfile* [, *absolutePath*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subfile</i>	Um inteiro no intervalo de 1 (um) até o número total de subarquivos em <i>superfile</i> que especifica a posição ordinal do subarquivo cujo nome será retornado.
<i>absolutePath</i>	Opcional. Um TRUE/FALSE booleano para indicar se um til (~) deve ser acrescentado ao início do nome do arquivo lógico estrangeiro. Se omitido, o padrão é FALSE.
Return:	GetSuperFileSubName retorna um valor VARSTRING.

A função **GetSuperFileSubName** retorna o nome lógico do *subfile* especificado no *superfile*.

Essa função não é incluída em uma transação do superarquivo.

Exemplo:

```
A := STD.File.GetSuperFileSubName('~CLASS::RT::IN::SF1', 1);
    //get name of first sub-file
//this example gets the name of the first sub-file in
// a foreign superfile
sf := '~thor_data400::BASE::Business_Header';
sub := STD.File.GetSuperFileSubName( STD.File.ForeignLogicalFileName (sf,
    '10.150.29.161',
    TRUE),
    1,TRUE);
OUTPUT(STD.File.ForeignLogicalFileName(sub,''));
```

LogicalFileSuperOwners

STD.File.LogicalFileSuperOwners(*filename*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
Return:	LogicalFileSuperOwners retorna um dataset no seguinte formato:

```
EXPORT FsLogicalFileNameRecord := RECORD
  STRING name;
END;
```

A função **LogicalFileSuperOwners** retorna uma lista dos nomes de arquivos lógicos de todos os superarquivos que contêm *filename* como um subarquivo.

Essa função não é incluída em uma transação de superarquivo.

Exemplo:

```
OUTPUT(STD.File.LogicalFileSuperowners('~CLASS::RT::SF::Daily1'));
//returns all SuperFiles that "own" the Daily1 file
```

LogicalFileSuperSubList

STD.File.LogicalFileSuperSubList()

Return: LogicalFileSuperSubList Retorna um dataset no seguinte formato:

```
EXPORT FsLogicalSuperSubRecord := RECORD
  STRING supname{MAXLENGTH(255)};
  STRING subname{MAXLENGTH(255)};
END;
```

A função **LogicalFileSuperSubList** retorna uma lista dos nomes de arquivos lógicos de todos os superarquivos e de seus subarquivos.

Essa função não é incluída em uma transação de superarquivo.

Exemplo:

```
OUTPUT(STD.File.LogicalFileSuperSubList());
//returns all SuperFiles and their sub-files
```

SuperFileContents

STD.File.SuperFileContents(*filename* [, *recurse*])

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do superfile.
<i>recurse</i>	Um indicador booleano que indica se os superarquivos aninhados dentro de <i>filename</i> devem ser expandidos para que apenas arquivos lógicos sejam retornados. Se omitido, o padrão é falso.
Return:	Retorna um dataset no seguinte formato:

```
EXPORT FsLogicalFileNameRecord := RECORD
  STRING name;
END;
```

A função **SuperFileContents** retorna uma lista dos nomes de arquivos lógicos de todos os subarquivos em *filename*.

Essa função não é incluída em uma transação de superarquivo.

Exemplo:

```
OUTPUT(STD.File.SuperFileContents('~CLASS::RT::SF::Daily'));
//returns all files in the SuperFile
```

FindSuperFileSubName

STD.File.FindSuperFileSubName(*superfile*, *subfile*)

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subfile</i>	Uma string terminada por nulo que contém o nome lógico do subfile.
Return:	FindSuperFileSubName retorna um valor INTEGER4.

A função **FindSuperFileSubName** retorna a posição ordinal do subfile especificado no superfile. *subfile* na definição *superfile*.

Essa função não é incluída em uma transação de superarquivo.

Exemplo:

```
A := STD.File.GetSuperFileSubName('~CLASS::RT::IN::SF1', 'Sue');  
//get position of sub-file Sue
```

StartSuperFileTransaction

STD.File.StartSuperFileTransaction()

Return: Nulo

A função **StartSuperFileTransaction** inicia um período de transação para a manutenção do superarquivo. O período de transação é terminado chamando a função **FinishSuperFileTransaction**. Dentro do período de transação, vários superarquivos podem ser mantidos adicionando, removendo, limpando, alternando e substituindo subfiles.

Você deve usar a ação **SEQUENTIAL** para garantir a execução ordenada das chamadas de função dentro do período da transação. Assim, as funções do SuperFile Maintenance são chamadas na ordem em que são listadas entre as funções de início e término do período da transação, mas são confirmadas apenas uma vez (ou seja, realmente executadas) no final da função de transação.

A primeira função de manutenção do SuperFile chamada dentro do período da transação inicia um bloqueio de “leitura” no superfile até o commit. No commit, o superfile é bloqueado para “gravação” para que a transação seja realmente executada, e todos os bloqueios são liberados no final do commit. É importante notar que quaisquer chamadas para funções diferentes das funções de Manutenção do SuperFile dentro do período da transação não fazem parte do período da transação (mesmo que sejam executadas na ordem em que foram escritas). O bloqueio de “leitura” só é gerado quando a primeira função de manutenção do SuperFile é chamada. Enquanto o superfile está bloqueado para “leitura”, nenhum bloqueio simultâneo de “gravação” pode modificar o superfile.

Durante o intervalo de tempo do bloqueio de “gravação” na confirmação (geralmente uma pequena janela de tempo), nenhum bloqueio de “leitura” simultâneo é permitido. Portanto, as funções de manutenção do SuperFile devem ser chamadas dentro de um período da transação para evitar a possibilidade de outro processo tentar modificar o superfile durante a manutenção do subarquivo. Como resultado, o trabalho de manutenção pode ser realizado sem causar problemas com qualquer consulta que possa usar o SuperFile.

A função **FinishSuperFileTransaction** reverte automaticamente a transação se qualquer erro ou falha ocorre durante o período de transação. Se nenhum erro ocorrer, a confirmação ou reversão da transação será controlada pelo parâmetro *rollback* da função **FinishSuperFileTransaction**.

Exemplo:

```
IMPORT STD;

WeeklyRollup:='~Training::Examples::WeeklyRollup';
WeeklySF      := '~Training::Examples::Weekly';
DailySF       := '~Training::Examples::Daily';

DailyDS := DATASET(DailySF,{string Myfield},THOR);

SEQUENTIAL(STD.File.StartSuperFileTransaction(),
            STD.File.ClearSuperFile(DailySF),
            OUTPUT(DailyDS,,WeeklyRollup),
            STD.File.AddSuperFile(WeeklySF,WeeklyRollup),
            STD.File.FinishSuperFileTransaction());
//executes the OUTPUT after a "read" lock on the superfile DailySF
//has been initiated by the ClearSuperFile Maintenance function,
//which in turn executes only at the FinishTransaction
```

AddSuperFile

STD.File.AddSuperFile(*superfile*, *subfile* [, *atpos*] [, *addcontents*] [, *strict*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subfile</i>	Uma string terminada por nulo que contém o nome lógico do subarquivo. Esse subarquivo pode ser outro superarquivo.
<i>atpos</i>	Um inteiro que especifica a posição do subarquivo no superarquivo. <i>subfile</i> na definição <i>superfile</i> . Se omitido, o padrão é zero (0), que posiciona o <i>subfile</i> no final do <i>superfile</i> .
<i>addcontents</i>	Um indicador booleano que, se definido como TRUE, especifica que o <i>subfile</i> é também um superarquivo e que o conteúdo desse superarquivo são adicionados ao superarquivo, em vez de sua referência. Se omitido, o padrão é adicionar por referência (<i>addcontents</i> := FALSE).
<i>strict</i>	Um indicador booleano que especifica, no caso de um <i>subfile</i> que também é um superarquivo, se a existência do superarquivo deve ser verificada e se um erro deve ser gerado se o superarquivo não existe. Além disso, se <i>addcontents</i> estiver definido como TRUE, será verificado se o <i>subfile</i> que é um superarquivo não está vazio. Se omitido, o padrão é falso. Se omitido, o padrão é falso.
Return:	Nulo

A função **AddSuperFile** adiciona o *subfile* à lista de arquivos que compõem o *superfile*. Todos os *subfiles* no *superfile* devem ter exatamente o mesmo tipo e formato da estrutura.

Essa função pode ser incluída em uma transação de superarquivo, mas isso não é obrigatório.

Exemplo:

```
IMPORT STD;
SEQUENTIAL(
  STD.File.StartSuperFileTransaction(),
  STD.File.AddSuperFile('MySuperFile1','MySubFile1'),
  STD.File.AddSuperFile('MySuperFile1','MySubFile2'),
  STD.File.AddSuperFile('MySuperFile2','MySuperFile1'),
  STD.File.AddSuperFile('MySuperFile3','MySuperFile1',addcontents := true),
  STD.File.FinishSuperFileTransaction()
);

// MySuperFile1 contains { MySubFile1, MySubFile2 }
// MySuperFile2 contains { MySuperFile1 }
// MySuperFile3 contains { MySubFile1, MySubFile2 }
```


RemoveSuperFile

STD.File.RemoveSuperFile(*superfile*, *subfile* [, *delete*] [, *removecontents*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subfile</i>	Uma string terminada por nulo que contém o nome lógico do subarquivo. Esse subarquivo pode ser outro superarquivo, bem como um arquivo ou superarquivo estrangeiro.
<i>delete</i>	Um indicador booleano que especifica se o <i>subfile</i> deve ser excluído do disco ou apenas removido da lista de arquivos do <i>superfile</i> . Se omitido, o padrão é apenas remover o arquivo da lista de arquivos do <i>superfile</i> . Essa opção não deve ser usada se o subarquivo for um arquivo ou superarquivo estrangeiro.
<i>removecontents</i>	Um indicador booleano que especifica que o conteúdo de um <i>subfile</i> que um superarquivo será removido recursivamente.
Return:	Nulo

A função **RemoveSuperFile** remove o *subarquivo* da lista de arquivos que compõem o *superfile*.

Essa função pode ser incluída em uma transação de superarquivo.

Exemplo:

```
SEQUENTIAL(  
  STD.File.StartSuperFileTransaction(),  
  STD.File.RemoveSuperFile('MySuperFile', 'MySubFile'),  
  STD.File.FinishSuperFileTransaction()  
);
```

ClearSuperFile

STD.File.ClearSuperFile(*superfile*, [, *delete*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superfile.
<i>delete</i>	Um indicador booleano que especifica se os subarquivos serão excluídos do disco ou apenas removidos da lista de arquivos do <i>superfile</i> . Se omitido, o padrão é apenas remover os <i>superfile</i> da lista de arquivos.
Return:	Nulo

A função **ClearSuperFile** remove todos os subarquivos da lista de arquivos que compõem o *superfile*.

Essa função pode ser incluída em uma transação de superarquivo.

Exemplo:

```
SEQUENTIAL(  
  STD.File.StartSuperFileTransaction(),  
  STD.File.ClearSuperFile('MySuperFile'),  
  STD.File.FinishSuperFileTransaction()  
);
```

RemoveOwnedSubFiles

STD.File.RemoveOwnedSubFiles(*superfile* [, *delete*])

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>delete</i>	Um indicador booleano que especifica se os subarquivos serão excluídos do disco ou apenas removidos da lista de arquivos do <i>superarquivo</i> . Se omitido, o padrão é apenas remover os subarquivos da lista de arquivos do <i>superarquivo</i> .
Return:	Nulo

A função **RemoveOwnedSubFiles** remove do superarquivo especificado todos os subarquivos do proprietário. Os arquivos serão removidos apenas se forem propriedade exclusiva do superarquivo. Se a propriedade do subarquivo for dividida (ou seja, é membro de outro superarquivo), a remoção será ignorada.

Essa função pode ser incluída em uma transação de superarquivo, a menos que o indicador de exclusão seja TRUE.

Exemplo:

```
SEQUENTIAL(  
  STD.File.StartSuperFileTransaction(),  
  STD.File.RemoveOwnedSubFiles('MySuperFile'),  
  STD.File.FinishSuperFileTransaction()  
);
```

SwapSuperFile

STD.File.SwapSuperFile(*superfile1*, *superfile2*)

<i>superfile1</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>superfile2</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
Return:	Null.

A função **SwapSuperFile** move todos os subarquivos de *superfile1* para *superfile2* e vice-versa.

Essa função pode ser incluída em uma transação de superarquivo.

Exemplo:

```
SEQUENTIAL(  
  STD.File.StartSuperFileTransaction(),  
  STD.File.SwapSuperFile('MySuperFile','YourSuperFile'),  
  STD.File.FinishSuperFileTransaction()  
);
```

ReplaceSuperFile

STD.File.ReplaceSuperFile(*superfile*, *subfile1* , *subfile2*)

<i>superfile</i>	Uma string terminada por nulo que contém o nome lógico do superarquivo.
<i>subfile1</i>	Uma string terminada por nulo que contém o nome lógico do subarquivo. Esse subarquivo pode ser outro superarquivo.
<i>subfile2</i>	Uma string terminada por nulo que contém o nome lógico do subarquivo. Esse subarquivo pode ser outro superarquivo.
Return:	Nulo

A função **ReplaceSuperFile** remove o *subfile1* da lista de arquivos que compõem o *superarquivo* e substitui pelo *subarquivo2*.

Essa função pode ser incluída em uma transação de superarquivo.

Exemplo:

```
SEQUENTIAL(  
  STD.File.StartSuperFileTransaction(),  
  STD.File.ReplaceSuperFile('MySuperFile',  
    'MyOldSubFile',  
    'MyNewSubFile'),  
  STD.File.FinishSuperFileTransaction()  
);
```

PromoteSuperFileList

STD.File.PromoteSuperFileList(*supernames* [, *addhead*] [, *deltail*] [, *createjustone*] [, *reverse*])

oldlist := **STD.File.fPromoteSuperFileList**(*supernames* [, *addhead*] [, *deltail*] [, *createjustone*] [, *reverse*]);

<i>supernames</i>	Um conjunto de strings terminadas por nulo que contém os nomes lógicos dos superarquivos a serem processados. Qualquer um não existentes será criado. O conteúdo de cada superarquivo será movido para o próximo da lista (NB: cada superarquivo deve conter subarquivos diferentes).
<i>addhead</i>	Opcional. Uma string terminada por nulo que contém uma lista delimitada por vírgulas de nomes de arquivos lógicos a serem adicionados ao primeiro <i>superfile</i> após a conclusão do processo de promoção.
<i>deltail</i>	Opcional. Um valor booleano que especifica se o conteúdo movido do último superarquivo deve ser fisicamente excluído. Se omitido, o padrão é FALSE.
<i>createjustone</i>	Opcional. Um valor booleano que especifica se apenas um único superarquivo deve ser criado (truncar a lista no primeiro superarquivo não existente). Se omitido, o padrão é FALSE.
<i>reverse</i>	Opcional. Um valor booleano que especifica se a ordem de processamento da lista <i>supernames</i> deve ser invertida, o que corresponde a "rebaixar" os subarquivos, em vez de "promovê-los". Se omitido, o padrão é FALSE.
<i>oldlist</i>	O nome do atributo que recebe a string retornada que contém a lista do conteúdo do subarquivo anterior do superarquivo esvaziado.
Return:	PromoteSuperFileList retorna Null; fPromoteSuperFileList retorna uma string.

A função **PromoteSuperFileList** move os subarquivos da primeira entrada da lista *supernames* para a próxima da lista, repetindo o processo posteriormente em toda a lista *supernames*.

Esta função não usa transações de superarquivos e é uma operação atômica.

Exemplo:

```
STD.File.PrSTD.File.PromoteSuperFileList(['Super1','Super2','Super3'],
    'NewSub1');
//Moves what was in Super1 to Super2,
// what was in Super2 to Super3, replacing what was in Super3,
// and putting NewSub1 in Super1
```

FinishSuperFileTransaction

STD.File.FinishSuperFileTransaction([*rollback*])

<i>rollback</i>	Opcional. Um indicador booleano que indica se a transação deve ser confirmada (FALSE) ou revertida (TRUE). Se omitido, o padrão é FALSE.
Return:	Null.

A função **FinishSuperFileTransaction** encerra o período da transação de manutenção do superfile.

Se o indicador *rollback* é FALSE, a transação é confirmada atômicamente e o período da transação é encerrado. Caso contrário, a transação é revertida e o período da transação é encerrado.

No commit, o superfile é "gravado" bloqueado para a transação que realmente executa, e todos os bloqueios são liberados quando o período de transação encerra. Durante o intervalo de tempo do bloqueio de "gravação" na confirmação (geralmente uma pequena janela de tempo), nenhum bloqueio de "leitura" simultâneo é permitido.

Exemplo:

```
IMPORT STD;

WeeklyRollup:='~Training::Examples::WeeklyRollup';
WeeklySF      := '~Training::Examples::Weekly';
DailySF       := '~Training::Examples::Daily';

DailyDS := DATASET(DailySF,{string Myfield},THOR);

SEQUENTIAL(STD.File.StartSuperFileTransaction(),
  STD.File.ClearSuperFile(DailySF),
  OUTPUT(DailyDS,,WeeklyRollup),
  STD.File.AddSuperFile(WeeklySF,WeeklyRollup),
  STD.File.FinishSuperFileTransaction());
//executes the OUTPUT after a "read" lock on the superfile DailySF
//has been initiated by the ClearSuperFile Maintenance function,
//which in turn executes only at the FinishTransaction
```

Arquivos Externo

ExternalLogicalFileName

STD.File.ExternalLogicalFileName(*machineIP*, *filename*)

<i>machineIP</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
<i>filename</i>	Uma string terminada por nulo que contém caminho/nome do arquivo.
Return:	ExternalLogicalFileName retorna um valor VARSTRING (terminado por nulo).

A função **ExternalLogicalFileName** retorna um nome de arquivo lógico externo codificado adequadamente que pode ser usado para ler um arquivo diretamente de qualquer nó que executa o utilitário dafailesrv (normalmente, uma zona de entrada de arquivos). A função processa caracteres maiúsculos prefixando-os com um caractere de escape na string de retorno.

Exemplo:

```
IP    := '10.150.254.6';
file  := '/c$/training/import/AdvancedECL/people';
DS1   := DATASET(STD.File.ExternalLogicalFileName(IP,file),
                 Training_Advanced.Layout_PeopleFile, FLAT);
OUTPUT(STD.File.ExternalLogicalFileName(IP,file));
//returns:
//~file::10.150.254.6::c$::training::import::^advanced^e^c^l::people
OUTPUT(DS1);
//returns records from the external file
```

MoveExternalFile

STD.File.MoveExternalFile(*location*, *frompath*, *topath*)

<i>location</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
<i>frompath</i>	Uma string terminada por nulo que contém o caminho/nome do arquivo a ser movido.
<i>topath</i>	Uma string terminada por nulo que contém o caminho/nome do arquivo de destino.

A função **MoveExternalFile** move o arquivo físico único especificado por *frompath* para *topath*. Ambos *frompath* e *topath* estão na mesma máquina remota, identificada por *location*. O programa utilitário *dafileserv* deve estar executando na máquina em *location*.

Exemplo:

```
IP      := '10.150.254.6';
infile  := '/c$/training/import/AdvancedECL/people';
outfile := '/c$/training/import/DFUtest/people';
STD.File.MoveExternalFile(IP,infile,outfile);
```

DeleteExternalFile

STD.File.DeleteExternalFile(*location*, *path*)

<i>location</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
<i>path</i>	Uma string terminada por nulo que contém o caminho/nome do arquivo a ser removido.

A função **DeleteExternalFile** remove o arquivo físico único especificado por *path* de *location*. O programa utilitário `dafileserv` deve estar executando na máquina em *location*.

Exemplo:

```
IP      := '10.150.254.6';  
infile := '/c$/training/import/AdvancedECL/people';  
STD.File.DeleteExternalFile(IP,infile);
```

CreateExternalDirectory

STD.File.CreateExternalDirectory(*location*, *path*)

<i>location</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
<i>path</i>	Uma string terminada por nulo que contém o caminho do diretório a ser criado.

A função **CreateExternalDirectory** cria o *path* em *location* (se já não existir). O programa utilitário *dafileserv* deve estar executando na máquina em *location* .

Exemplo:

```
IP    := '10.150.254.6';  
path := '/c$/training/import/NewDir';  
STD.File.CreateExternalDirectory(IP,path);
```

RemoteDirectory

STD.File.RemoteDirectory(*machineIP*, *directory* [, *mask*][, *recurse*])

<i>machineIP</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
<i>directory</i>	Uma string terminada por nulo que contém a path para os diretório para leitura. A string deve estar no formato adequado para o sistema operacional executado na máquina remota.
<i>mask</i>	Opcional. Uma string terminada por nulo que contém a máscara de arquivo que especifica quais arquivos serão incluídos no resultado. Se omitida, o padrão é "*" (todos os arquivos).
<i>recurse</i>	Opcional. Um indicador booleano que sinaliza se é necessário incluir arquivos dos subdiretórios abaixo do <i>diretório</i> . Se omitido, o padrão é FALSE.
Return:	Retorna um dataset no seguinte formato:

```
EXPORT FsFilenameRecord := RECORD
  STRING name;          //filename
  UNSIGNED8 size;       //filesize
  STRING19 modified;    //date-time stamp
END;
```

A função **RemoteDirectory** retorna uma lista de arquivos como um dataset no formato listado acima para o *machineIP* e o *diretório* especificados. Se *recurse* for definido como TRUE, o nome do campo conterá o caminho relativo para o arquivo especificado.

Exemplo:

```
OUTPUT(STD.File.RemoteDirectory('edata12','\in','*.d00'));
OUTPUT(STD.File.RemoteDirectory('10.150.254.6',
  '/var/lib/HPCCSystems/hpcc-data/thor/', 'acc*', TRUE));
```

Navegação por Arquivos

SetColumnMapping

STD.File.SetColumnMapping(*file*, *mapping*);

<i>file</i>	Uma string terminada por nulo que contém o nome do arquivo lógico.
<i>mappings</i>	Uma string terminada por nulo que contém uma lista delimitada por vírgulas dos mapeamentos do campo.

A função **SetColumnMapping** define como os dados nos campos *file* devem ser transformados entre o formato de armazenamento de dados real e o formato de entrada usado para consultar esses dados.

O formato de cada campo na lista *mapping* é:

<field>{set(<transform>(args),...),get(<transform>,...),displayname(<name>)}

<i><field></i>	O nome do campo no arquivo.
set	Opcional. Especifica as transformações aplicadas aos valores fornecidos pelo usuário para convertê-los em valores no arquivo.
<i><transform></i>	Opcional. O nome da função a ser aplicada ao valor. Normalmente, é o nome de uma função de plugin. O valor sendo convertido é sempre fornecido como o primeiro parâmetro da função. No entanto, parâmetros adicionais podem ser especificados em colchetes após o nome da transformação (de forma semelhante à higiene de SALT).
get	Opcional. Especifica as transformações aplicadas aos valores no arquivo para convertê-los nos valores formatados como entendidos pelo usuário.
displayname	Opcional. Permite a associação de um <i>nome</i> diferente do campo entendido naturalmente pelo usuário.

Você pode misturar funções Unicode e de strings, pois o sistema converte automaticamente os parâmetros para os tipos adequados esperados pelas funções.

Exemplo:

```
// A file where the firstname(string) and lastname(unicode) are
//always upper-cased:
// There is no need for a displayname since it isn't really a
// different field as far as the user is concerned, and there is
// obviously no get transformations.
  firstname{set(stringlib.StringToUpperCase)},
             surname{set(unicodelib.UnicodeToUpperCase)}
// A name translated using a phonetic key
// it is worth specifying a display name here, because it will make
// more sense to the user, and the user may want to enter either the
// translated or untranslated names.
  dph_lname{set(metaphonelib.DMetaPhone1),
             displayname(lname)}
// A file where a name is converted to a token using the namelib
// functions. (I don't think we have an example of this)
// (one of the few situations where a get() attribute is useful)
  fnametoken{set(namelib.nameToToken),
              get(namelib.tokenToName),
              displayname(fname)}
// upper case, and only include digits and alphabetic.
  searchname{set(stringlib.StringToUpperCase,
                 stringlib.StringFilter(
                   'ABCDEFGHijklmnopqrstuvwxyz0123456789'))}
// A file with a field that that needs to remove accents and then
```

```
// uppercase:  
lastname{set(unicodeLib.CleanAccents,stringLib.StringToUpperCase)}
```


GetColumnMapping

result := **STD.File.GetColumnMapping**(*file*);

<i>file</i>	Uma string terminada por nulo que contém o nome do arquivo lógico.
Return:	GetColumnMapping retorna uma string terminada por nulo que contém a lista delimitadas por vírgulas dos mapeamentos de campos de <i>file</i> .

A função **GetColumnMapping** retorna os mapeamentos de campo de *file* no mesmo formato especificado para a função **SetColumnMapping**.

Exemplo:

```
Maps := STD.File.GetColumnMapping('Thor::in::SomeFile');
```

AddFileRelationship

STD.File.AddFileRelationship(*primary*, *secondary*, *primaryfields*, *secondaryfields*, [*relationship*], *cardinality*, *payload* [, *description*]);

<i>primary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo primário.
<i>secondary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo secundário.
<i>primaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave primária do arquivo <i>primary</i> . O valor "__fileposition__" indica que <i>secondary</i> é um INDEX que deve usar FETCH para acessar campos sem chave.
<i>secondaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave estrangeira relacionada ao arquivo <i>primary</i> .
<i>relationship</i>	Uma string terminada por nulo que contém "link" ou "view" para indicar o tipo de relação entre os arquivos <i>primary</i> e <i>secondary</i> . Se omitida, o padrão é "link."
<i>cardinality</i>	Uma string terminada por nulo que contém o tipo de relação entre os arquivos <i>primary</i> e <i>secondary</i> . O formato é X:Y, onde X indica o <i>primary</i> e Y indica o <i>secondary</i> . Os valores válidos para X e Y são "1" ou "M".
<i>payload</i>	Um valor BOOLEAN indicando se <i>primary</i> ou <i>secondary</i> são INDEXes da carga útil.
<i>description</i>	Uma string terminada em nulo que contém a descrição da relação.

A função **AddFileRelationship** define a relação entre dois arquivos. Esses arquivos podem ser DATASETs ou INDEXes. Cada registro no arquivo *primary* deve ser definido unicamente por *primaryfields* (*idealmente*), preferivelmente de forma eficiente.

Os parâmetros *primaryfields* e *secondaryfields* podem ter o mesmo formato dos mapeamentos de coluna para um arquivo (consulte a documentação da função SetColumnMappings), embora sejam frequentemente apenas uma lista de campos.

No momento, são usados de duas formas diferentes:

Primeiro, o navegador do roxie precisa de uma forma de determinar quais índices são criados com base em quais arquivos. Uma relação "view" deve ser adicionada a cada vez que um índice é criado de um arquivo, desta forma:

```
STD.File.AddFileRelationship(DG_FlatFileName, DG_IndexFileName,  
                             '', '', 'view', '1:1', false);
```

Para implementar o navegador do roxie, não há necessidade de definir *primaryfields* ou *secondaryfields*. Portanto, é recomendado passar strings em branco.

Segundo, o navegador precisa de uma forma de encontrar todas as informações originais do arquivo de um índice.

Essa fase depende da natureza do índice:

- Se a opção de índice contém todos os dados relevantes do arquivo original, não é necessário fazer nada.
- Se o índice usa um campo fileposition para FETCH dados adicionais do arquivo original, adicione uma relação entre o arquivo original e o índice usando um valor especial de __fileposition__ para indicar que o registro é recuperado usando FETCH.

```
STD.File.AddFileRelationship('fetch_file',  
                             'new_index',  
                             '__fileposition__',  
                             'index_filepos_field',
```

```
'link',  
'1:1',  
true);
```

O arquivo original é o primário, pois as linhas são identificadas unicamente por fileposition (o que também acontece com o índice) e a recuperação é eficiente.

c) Se o índice usar um campo de carga útil que precisa ser pesquisado em outro índice para fornecer as informações, você precisará definir uma relação entre o novo índice e o índice que fornece as informações adicionais. O índice que fornece as informações adicionais é o primário.

```
STD.File.AddFileRelationship('related_index',  
                             'new_index',  
                             'related_key_fields',  
                             'index_filepos_field',  
                             'link',  
                             '1:M',  
                             true);
```

O indicador *payload* está presente, portanto, o navegador do roxie pode distinguir esse link de uma relação mais geral entre dois arquivos.

Você deverá garantir que todos os nomes de superarquivos sejam expandidos se a relação for definida entre os sub-arquivos particulares.

Ao percorrer todos os atributos, pode valer a pena examinar se faz sentido adicionar relações para outras combinações de arquivos. Isso não trará nenhum benefício imediato, mas poderá trazer quando adicionarmos um diagrama ER ao sistema. Alguns exemplos podem ajudar a demonstrar a sintaxe.

Em um exemplo típico, datasets com um arquivo de residências e pessoas, o código abaixo define um linking de relação por ID de residência (hhid):

```
STD.File.AddFileRelationship('HHFile','PersonFile','hhid','hhid','link','1:M',false);
```

Aqui está um exemplo mais hipotético: um query de arquivo com nome e sobrenome relacionado a um possível índice com nomes fonéticos:

```
STD.File.AddFileRelationship('names','inquiries','plastname{set(phonetic)}',  
                             pfirstname{set(phonetic)}',  
                             'lastname{set(fail)},firstname{set(fail)}','link','1:M',false);
```

O mapeamento de falha indica que você pode usar o mapeamento fonético de consultas para nomes, mas não há forma de mapear de nomes para consultas. Também podem existir atributos get(fail) nos campos de índice.

Exemplo:

```
Maps := STD.File.GetColumnMapping('Thor::in::SomeFile');
```

FileRelationshipList

STD.File.FileRelationshipList(*primary*, *secondary* [, *primaryfields*] [, *secondaryfields*] [, *relationship*]);

<i>primary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo primário.
<i>secondary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo secundário.
<i>primaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave primária do arquivo <i>primary</i> . O valor "__fileposition__" indica que <i>secondary</i> é um INDEX que deve usar FETCH para acessar campos sem chave. Se omitida, o padrão é uma string vazia.
<i>secondaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave estrangeira relacionada ao arquivo <i>primary</i> . Se omitida, o padrão é uma string vazia.
<i>relationship</i>	Uma string terminada por nulo que contém "link" ou "view" para indicar o tipo de relação entre os arquivos <i>primary</i> e <i>secondary</i> . Se omitida, o padrão é "link."
Return:	FileRelationshipList retorna um dataset no formato FsFileRelationshipRecord.

A função **FileRelationshipList** retorna uma lista das relações entre arquivos *primário* e *secundário* . Os registros de retorno são estruturados no formato FsFileRelationshipRecord:

```
EXPORT FsFileRelationshipRecord := RECORD
  STRING primaryfile {MAXLENGTH(1023)};
  STRING secondaryfile {MAXLENGTH(1023)};
  STRING primaryflds {MAXLENGTH(1023)};
  STRING secondaryflds {MAXLENGTH(1023)};
  STRING kind {MAXLENGTH(16)};
  STRING cardinality {MAXLENGTH(16)};
  BOOLEAN payload;
  STRING description {MAXLENGTH(1023)};
END;
```

Exemplo:

```
OUTPUT(STD.File.FileRelationshipList('names', 'inquiries'));
```

Ver também: AddFileRelationship

RemoveFileRelationship

STD.File.RemoveFileRelationship(*primary*, *secondary*, [, *primaryfields*] [, *secondaryfields*] [, *relationship*]);

<i>primary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo primário.
<i>secondary</i>	Uma string terminada por nulo que contém o nome do arquivo lógico do arquivo secundário.
<i>primaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave primária do arquivo <i>primary</i> . O valor "__fileposition__" indica que <i>secondary</i> é um INDEX que deve usar FETCH para acessar campos sem chave. Se omitida, o padrão é uma string vazia.
<i>secondaryfields</i>	Uma string terminada por nulo que contém o nome do campo de chave estrangeira relacionada ao arquivo <i>primary</i> . Se omitida, o padrão é uma string vazia.
<i>relationship</i>	Uma string terminada por nulo que contém "link" ou "view" para indicar o tipo de relação entre os arquivos <i>primary</i> e <i>secondary</i> . Se omitida, o padrão é "link."

A função **RemoveFileRelationship** remove uma relação entre os arquivos *primary* e *secondary* .

Exemplo:

```
STD.File.RemoveFileRelationship('names', 'inquiries');
```

Ver também: AddFileRelationship

Movimentação de Arquivos

DfuPlusExec

STD.File.DfuPlusExec(*commandline*])

<i>commandline</i>	Uma string terminada por nulo que contém a linha de comando do DFUPlus.exe a ser executada. Os argumentos válidos estão documentados no manual das Ferramentas de cliente, na seção que descreve o programa DfuPlus.exe.
--------------------	--

A ação **DfuPlusExec** executa a *linha de comando* especificada da mesma forma que o programa DfuPlus.exe. Essa ação simplesmente disponibiliza toda a funcionalidade do programa DfuPlus.exe no código ECL.

Exemplo:

```
IMPORT STD;

serv := 'server=http://10.150.50.12:8010 ';
user := 'username=rlor ';
pswd := 'password=password ';
over := 'overwrite=1 ';
repl := 'replicate=1 ';

action := 'action=despray ';
dstip := 'dstip=10.150.50.14 ';
dstfile := 'dstfile=c:\\import\\despray\\timezones.txt ';
srcname := 'srcname=RTTEMP::timezones.txt ';

cmd := serv + user + pswd + over + repl + action + dstip + dstfile + srcname;
STD.File.DfuPlusExec(cmd);
```

AbortDfuWorkunit

STD.File.AbortDfuWorkunit(*dfuwuid* [,*espserverIPport*])

<i>dfuwuid</i>	Uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID) do job a ser abortado. Esse valor é retornado pelas versões das funções Copy, SprayFixed, SprayVariable, SprayXML e Despray FileServices "com nome começando por f".
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.

A função **AbortDfuWorkunit** aborta a workunit DFU especificada. Normalmente, esse job foi iniciado com o parâmetro *timeout* definido como zero (0).

Exemplo:

```
STD.File.AbortDfuWorkunit( 'D20051108-143758' );
```


Copy

STD.File.Copy(*sourceLogicalName*, *destinationGroup*, *destinationLogicalName*, [*,sourceDali*] [*,timeOut*] [*,espServerIPPort*] [*,maxConnections*] [*,allowOverwrite*] [*,replicate*] [*,asSuperfile*] [*,compress*] [*,forcePush*] [*,transferBufferSize*] [*,preserveCompression*] [*,noSplit*] [*,expireDays*]);

dfuwuid := **STD.File.fCopy**(*sourceLogicalName*, *destinationGroup*, *destinationLogicalName*, [*,sourceDali*] [*,timeOut*] [*,espServerIPPort*] [*,maxConnections*] [*,allowOverwrite*] [*,replicate*] [*,asSuperfile*] [*,compress*] [*,forcePush*] [*,transferBufferSize*] [*,preserveCompression*] [*,noSplit*] [*,expireDays*]);

<i>sourceLogicalName</i>	Uma string terminada em nulo contendo o nome lógico do arquivo.
<i>destinationGroup</i>	Uma string terminada por nulo que contém o cluster de destino do arquivo.
<i>destinationLogicalName</i>	Uma string terminada por nulo que contém o novo nome lógico do arquivo.
<i>sourceDali</i>	Opcional. Uma string terminada por nulo que contém o IP e a porta do Dali que contém o arquivo a ser copiado. Se omitida, o padrão é uma cópia intra-Dali.
<i>timeOut</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espServerIPPort</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowOverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente do mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser replicado automaticamente. Se omitido, o padrão é FALSE.
<i>asSuperfile</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o arquivo deve ser tratado como superarquivo. Se omitido, o padrão é FALSE. Se TRUE e o arquivo a ser copiado for um superarquivo, a operação criará um superarquivo no destino, criando subarquivos conforme a necessidade e sobrescrevendo apenas os subarquivos já existentes cujo conteúdo tenha sido alterado. Se FALSE e o arquivo a ser copiado é um superfile contendo INDEXes, então a operação não é válida e produzirá um erro.
<i>compress</i>	Opcional. Um booleano TRUE ou FALSE indica se LZW compactou o novo arquivo. Se omitido, o padrão é FALSE.
<i>forcePush</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o processo de cópia deve ser executado nos nós de origem enviando para os destinos, em vez de executado nos destinos e recebendo da origem. Se omitido, o padrão é FALSE.
<i>transferBufferSize</i>	Opcional. Um valor inteiro que substitui o valor do tamanho do buffer do servidor DFU (o padrão é 64 k)
<i>preservecompression</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se a compressão do arquivo antigo deve ser preservada durante a cópia. Se omitido, o padrão é TRUE.
<i>noSplit</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando para não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSO

Referência de Biblioteca Padrão
Movimentação de Arquivos

<i>expireDays</i>	Opcional. Um valor inteiro indicando o número de dias antes de remover automaticamente o arquivo. Se omitido, o padrão é -1 (nunca expira).
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerada para o job.
Return:	Copy retorna uma string terminada por nulo contendo o ID da workunit DFU (DFUWUID).

A função **Copy** recebe um arquivo lógico e o copia para outro arquivo lógico. Isso pode ser feito no mesmo cluster, para outro. O destino não pode ser um arquivo externo.

Exemplo:

```
STD.File.Copy( 'OUT::MyFile', STD.System.Thorlib.Group(), 'OUT::MyNewFile' );
```

DeSpray

STD.File.DeSpray(*logicalname*, *destinationIP* , *destinationpath* , [*timeout*] , [*espserverIPport*] , [*maxConnections*] , [*allowoverwrite*])

dfuwuid := **STD.File.fDeSpray**(*logicalname*, *destinationIP* , *destinationpath* , [*timeout*] , [*espserverIPport*] , [*maxConnections*] , [*allowoverwrite*]);

<i>logicalname</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>destinationIP</i>	Uma string terminada por nulo que contém o endereço IP de destino do arquivo.
<i>destinationpath</i>	Uma string terminada por nulo que contém o caminho e o nome do arquivo.
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowoverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente do mesmo nome. Se omitido, o padrão é FALSE.
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerada para a workunit.
Return:	fDeSpray retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **DeSpray** recebe um arquivo lógico e executa um despray (combina todas as partes de cada nó de super-computador em um único arquivo físico) para a zona de entrada de arquivos.

Exemplo:

```
STD.File.DeSpray('OUT::MyFile',  
  '10.150.50.14',  
  'c:\\OutputData\\MyFile.txt',  
  -1,  
  'http://10.150.50.12:8010/FileSpray');
```

RemotePull

STD.File.RemotePull(*remoteURL*, *sourceLogicalname*, *destinationGroup* , *destinationLogicalname*, [*,timeout*] [*,maxConnections*] [*,allowoverwrite*] [*,replicate*] [*,asSuperfile*] [*,forcePush*] [*,transferBufferSize*] [*,wrap*] [*,compress*] [*,noSplit*] [*,expireDays*])

dfuwuid := **STD.File.RemotePull**(*remoteURL*, *sourceLogicalname*, *destinationGroup* , *destinationLogicalname*, [*,timeout*] [*,maxConnections*] [*,allowoverwrite*] [*,replicate*] [*,asSuperfile*] [*,forcePush*] [*,transferBufferSize*] [*,wrap*] [*,compress*] [*,noSplit*] [*,expireDays*]);

<i>remoteURL</i>	Uma string terminada por nulo que contém o protocolo, o IP e o diretório ou DNS equivalente do programa do servidor ESP remoto. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado.
<i>sourceLogicalname</i>	Uma string terminada por nulo que contém o nome lógico local do arquivo.
<i>destinationGroup</i>	Uma string terminada por nulo que contém o nome do cluster de destino.
<i>destinationLogicalname</i>	Uma string terminada por nulo que contém o nome lógico do arquivo no cluster remoto (deve ser especificado totalmente, incluindo o domínio).
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é -1, o que indica que o sistema escolhe um padrão adequado de acordo com o tamanho do cluster.
<i>allowoverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente do mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser replicado automaticamente. Se omitido, o padrão é FALSE.
<i>asSuperfile</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o arquivo deve ser tratado como superarquivo. Se omitido, o padrão é FALSE. Se TRUE e o arquivo a ser copiado for um superarquivo, a operação criará um superarquivo no destino, criando subarquivos conforme a necessidade e sobrescrevendo apenas os subarquivos já existentes cujo conteúdo tenha sido alterado. Se FALSE e o arquivo a ser copiado for um superarquivo, a operação consolidará todo o conteúdo do superarquivo em um único arquivo lógico (e não um superarquivo) no destino.
<i>forcePush</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o processo de cópia deve ser executado nos nós de origem enviando para os destinos, em vez de executado nos destinos e recebendo da origem. Se omitido, o padrão é FALSE.
<i>transferBufferSize</i>	Opcional. Um inteiro que especifica o tamanho do buffer de transferência em bytes. Algumas vezes, valores maiores podem acelerar o processo. Se omitido, é usado o tamanho padrão do buffer, 64K.
<i>wrap</i>	Opcional. Um indicador booleano TRUE ou FALSE que sinaliza se as partes do arquivo devem ser aninhadas automaticamente durante a cópia para clusters de menor tamanho. Por exemplo, na cópia de um cluster de 6 nós para um de 3 nós, cada nó receberá duas partes de arquivo. A diferença está entre o nó 1 receber as partes 1 e 2 ou as partes 1 e 4. Se omitido, o padrão é FALSE.
<i>compress</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser compactado automaticamente usando LZW. Se omitido, o padrão é FALSE.
<i>noSplit</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando para não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSO

Referência de Biblioteca Padrão
Movimentação de Arquivos

<i>expireDays</i>	Opcional. Um valor inteiro indicando o número de dias antes de remover automaticamente o arquivo. Se omitido, o padrão é -1 (nunca expira).
<i>dfuwuid</i>	O nome da definição que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para a workunit.
Return:	fRemotePull retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **RemotePull** executa no *remoteURL*, copiando *sourceLogicalname* do ambiente local que instanciou a operação para o cluster *destinationGroup* do ambiente remoto e atribuindo o nome de *destinationLogicalname*. Isso é bastante semelhante ao uso da função STD.File.Copy com a especificação do parâmetro *espserverIPport*. Como a workunit DFU executa no servidor remoto DFU, a autenticação do nome do usuário deve ser a mesma nos dois sistemas, e o usuário deve ter permissão para copiar arquivos nos dois sistemas.

Exemplo:

```
STD.File.RemotePull('http://10.150.50.14:8010/FileSpray',  
  '~THOR::LOCAL::MyFile',  
  'RemoteThor',  
  '~REMOTETHOR::LOCAL::MyFile');
```

Replicate

STD.File.Replicate (*filename* [, *timeout*] [, *espserverIPport*])

dfuwuid := **STD.File.fReplicate**(*filename* [, *timeout*] [, *espserverIPport*]);

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da tarefa DFU (DFUWUID) gerada para o job.

A função **Replicate** copia as partes individuais de *filename* para os discos espelho do cluster. Normalmente, isso significa que a parte do arquivo na unidade C de um nó é copiada para a unidade D do nó vizinho.

Example:

```
A := STD.File.Replicate('Fred');
```

SprayFixed

STD.File.SprayFixed(*sourceIP* , *sourcepath*, *recordsize*, *destinationgroup*, *destinationlogicalname* , [*timeout*] , [*espserviceIPport*] , [*maxConnections*] , [*allowoverwrite*] , [*replicate*] , [*compress*] , [*failIfNoSourceFile*] , [*expireDays*] , [*dfuServerQueue*] , [*noSplit*])

dfuwuid := **STD.File.fSprayFixed**(*sourceIP* , *sourcepath*, *recordsize*, *destinationgroup*, *destinationlogicalname* , [*timeout*] , [*espserviceIPport*] , [*maxConnections*] , [*allowoverwrite*] , [*replicate*] , [*compress*] , [*failIfNoSourceFile*] , [*expireDays*] , [*dfuServerQueue*] , [*noSplit*])

<i>sourceIP</i>	Uma string terminada por nulo que contém o endereço IP do arquivo.
<i>sourcepath</i>	Uma cadeia terminada por nulo contendo o endereço IP ou nome do host da Dropzone em que o arquivo está localizado.
<i>recordsize</i>	Um inteiro que contém o tamanho dos registros no arquivo.
<i>destinationgroup</i>	Uma string terminada por nulo que contém o nome do supercomputador específico dentro do cluster de destino.
<i>destinationlogicalname</i>	Uma string terminada em nulo contendo o nome lógico do arquivo.
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserviceIPport</i>	Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_f-s_server.
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowoverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente do mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser replicado. Se omitido, o padrão é FALSE.
<i>compress</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser compactado. Se omitido, o padrão é FALSE.
<i>failIfNoSourceFile</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se um arquivo não encontrado gera uma falha. Se omitido, o padrão é FALSE.
<i>expireDays</i>	Opcional. Um valor inteiro indicando o número de dias restantes para o arquivo ser removido automaticamente. Se omitido, o padrão é -1 (sem data de validade).
<i>dfuServerQueue</i>	Nome do Servidor DFU de destino. O padrão é " (empty) para o primeiro na fila de DFU no ambiente.
<i>noSplit</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando para não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSO
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerada para a workunit.
Return:	fSprayFixed retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **SprayFixed** recebe um arquivo de formato fixo da zona de entrada de arquivos e o distribui entre os nós do supercomputador de destino.

Exemplo:

```
STD.File.SprayFixed('10.150.50.14','c:\\InputData\\MyFile.txt',  
    255,'400way','IN::MyFile',-1,  
    'http://10.150.50.12:8010/FileSpray');
```


SprayDelimited / SprayVariable

STD.File.SprayDelimited(*sourceIP* , *sourcePath* , [*sourceMaxRecordSize*] , [*sourceCsvSeparate*] , [*sourceCsvTerminate*] , [*sourceCsvQuote*] , *destinationGroup*, *destinationLogicalName* , [*timeout*] , [*espServerIpPort*] , [*maxConnections*] , [*allowOverwrite*] , [*replicate*] , [*compress*] , [*sourceCsvEscape*] , [*failIfNoSourceFile*] , [*recordStructurePresent*] , [*quotedTerminator*] , [*encoding*] , [*expireDays*] , [*dfuServerQueue*] , [*noSplit*])

dfuwuid := **STD.File.fSprayDelimited**(*sourceIP* , *sourcePath* , [*sourceMaxRecordSize*] , [*sourceCsvSeparate*] , [*sourceCsvTerminate*] , [*sourceCsvQuote*] , *destinationGroup*, *destinationLogicalName* , [*timeout*] , [*espServerIpPort*] , [*maxConnections*] , [*allowOverwrite*] , [*replicate*] , [*compress*] , [*sourceCsvEscape*] , [*failIfNoSourceFile*] , [*recordStructurePresent*] , [*quotedTerminator*] , [*encoding*] , [*expireDays*] , [*dfuServerQueue*] , [*noSplit*])

<i>sourceIP</i>	Uma cadeia terminada por nulo contendo o endereço IP ou nome do host da Dropzone em que o arquivo está localizado.
<i>sourcePath</i>	Uma string terminada por nulo que contém o caminho e o nome do arquivo.
<i>maxrecordsize</i>	Opcional. Um inteiro que contém o tamanho máximo dos registros no arquivo. Se omitido, o padrão é 4096.
<i>srcCSVseparator</i>	Opcional. Uma string terminada por nulo que contém o separador de campos do CSV. Se omitida, o padrão é "\\,"
<i>srcCSVterminator</i>	Opcional. Uma string terminada por nulo que contém o separador de registros do CSV. Se omitida, o padrão é "\\n,\\r\\n"
<i>srcCSVquote</i>	Opcional. Uma string terminada por nulo que contém o delimitador de campos entre aspas do CSV. Se omitida, o padrão é "\"
<i>destinationgroup</i>	Uma string terminada em nulo contendo o nome do supercomputador específico dentro do cluster de destino.
<i>destinationlogical-name</i>	Uma string terminada em nulo contendo o nome lógico do arquivo.
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor no atributo lib_system.ws_f-s_server.
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowoverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente do mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser replicado. Se omitido, o padrão é FALSE.
<i>compress</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser compactado. Se omitido, o padrão é FALSE.
<i>sourceCsvEscape</i>	Opcional. Uma string terminada por nulo que contém os caracteres de escape do CSV. Se omitida, o padrão é nenhum.

Referência de Biblioteca Padrão
Movimentação de Arquivos

<i>failIfNoSourceFile</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o spray deve falhar caso nenhum arquivo de origem seja encontrado. Se omitido, o padrão é FALSE.
<i>recordStructurePresent</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se a estrutura do registro deve ser derivada do cabeçalho do arquivo. Se omitido, o padrão é FALSE.
<i>quotedTerminator</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o caractere terminador pode ser incluído em um campo entre aspas. O Padrão é TRUE para todos. Se FALSE, permite um particionamento mais rápido do arquivo (evitando uma varredura completa do arquivo).
<i>expireDays</i>	Opcional. Um valor inteiro indicando o número de dias restantes para o arquivo ser removido automaticamente. Se omitido, o padrão é -1 (sem data de validade).
<i>encoding</i>	Uma string terminada por nulo que contém a codificação. Pode ser definida como: ascii, utf8, utf8n, utf16, utf16le, utf16be, utf32, utf32le e utf32be. Se omitida, o padrão é ascii.
<i>noSplit</i>	Opcional. A flag booleana TRUE ou FALSE indicando para não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSE.
<i>dfuServerQueue</i>	Nome do Servidor DFU de destino. O padrão é " (empty) para o primeiro na fila de DFU no ambiente.
<i>dfuwuid</i>	O nome da definição que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para a workunit.
Return:	fSprayDelimited retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **SprayDelimited** recebe um arquivo de comprimento variável da zona de entrada de arquivos e o distribui entre os nós do supercomputador de destino.

A função **SprayVariable** agora é chamada **SprayDelimited** e a função **fSprayVariable** é agora chamada **fSprayDelimited**. Os nomes antigos ainda estão disponíveis para compatibilidade com versões anteriores.

Exemplo:

```
STD.File.SprayDelimited('10.150.50.14',
    'c:\\InputData\\MyFile.txt',
    '400way',
    'IN:MyFile',
    -1,
    'http://10.150.50.12:8010/FileSpray');
```

SprayXML

STD.File.SprayXML(*sourceIP* , *sourcepath* , [*maxrecordsize*] , *srcRowTag* , [*srcEncoding*] , *destinationgroup*, *destinationlogicalname* [*timeout*] [*espserverIPport*] [*maxConnections*] [*allowoverwrite*] [*replicate*] [*compress*] , [*failIfNoSourceFile*], [*expireDays*], [*dfuServerQueue*], [*noSplit*])

dfuwuid := **STD.File.fSprayXML**(*sourceIP* , *sourcepath* , [*maxrecordsize*] , *srcRowTag* , [*srcEncoding*] , *destinationgroup*, *destinationlogicalname* , [*timeout*], [*espserverIPport*] , [*maxConnections*], [*allowoverwrite*], [*replicate*], [*compress*] , [*failIfNoSourceFile*], [*expireDays*], [*dfuServerQueue*], [*noSplit*])

<i>sourceIP</i>	Uma cadeia terminada por nulo contendo o endereço IP ou nome do host da Dropzone em que o arquivo está localizado.
<i>sourcepath</i>	Uma string terminada por nulo que contém o caminho e o nome do arquivo.
<i>maxrecordsize</i>	Opcional. Um inteiro que contém o tamanho máximo dos registros no arquivo. Se omitido, o padrão é 8192.
<i>srcRowTag</i>	Uma string terminada por nulo que contém a tag XML delimitadora de linhas. Obrigatório.
<i>srcEncoding</i>	Opcional. Uma string terminada por nulo que contém a codificação. Se omitida, o padrão é "utf8"
<i>destinationgroup</i>	Uma string terminada por nulo que contém o nome do supercomputador específico dentro do cluster de destino.
<i>destinationlogicalname</i>	Uma string terminada em nulo contendo o nome lógico do arquivo.
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo <code>lib_system.ws_fs_server</code> .
<i>maxConnections</i>	Opcional. Um inteiro que especifica o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowoverwrite</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo pode sobrescrever um arquivo existente com o mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser replicado. Se omitido, o padrão é FALSE.
<i>compress</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se o novo arquivo deve ser compactado. Se omitido, o padrão é FALSE.
<i>failIfNoSourceFile</i>	Opcional. Um indicador booleano TRUE ou FALSE que indica se um arquivo não encontrado gera uma falha. Se omitido, o padrão é FALSE.
<i>expireDays</i>	Opcional. Um valor inteiro indicando o número de dias restantes para o arquivo ser removido automaticamente. Se omitido, o padrão é -1 (sem data de validade).
<i>dfuServerQueue</i>	Nome do Servidor DFU de destino. O padrão é " (empty) para o primeiro na fila de DFU no ambiente.
<i>noSplit</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando para não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSO
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para a workunit.

Return:	fSprayXML retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).
---------	--

A função **SprayXML** recebe um arquivo XML formatado corretamente da zona de entrada de arquivos e o distribui entre os nós do supercomputador de destino, gerando um arquivo XML formatado corretamente em cada nó.

Exemplo:

```
STD.File.SprayXML('10.150.50.14','c:\\InputData\\MyFile.txt',,
    'Row',, '400way', 'IN::MyFile', -1,
    'http://10.150.50.12:8010/FileSpray');
```

SprayJson

STD.File.SprayJson(*sourceIP* , *sourcepath* , [*maxrecordsize*] , *srcRowPath* , [*srcEncoding*] , *destinationgroup*, *destinationlogicalname* [*timeout*] [*espserverIPport*] [*maxConnections*] [*allowoverwrite*] [*replicate*] [*compress*] , [*failIfNoSourceFile*], [*expireDays*], [*dfuServerQueue*], [*noSplit*])

dfuwuid := **STD.File.fSprayJson**(*sourceIP* , *sourcepath*, [*maxrecordsize*] , *srcRowPath* , [*srcEncoding*] , *destinationgroup*, *destinationlogicalname* , [*timeout*], [*espserverIPport*] , [*maxConnections*], [*allowoverwrite*], [*replicate*], [*compress*] , [*failIfNoSourceFile*], [*expireDays*], [*dfuServerQueue*], [*noSplit*])

<i>sourceIP</i>	Uma string terminada por nulo contendo o endereço IP ou o nome do host da Dropzone em que o arquivo está localizado.
<i>sourcepath</i>	Uma string terminada por nulo contendo o caminho e o nome do arquivo.
<i>maxrecordsize</i>	Opcional. Um número inteiro contendo o tamanho máximo dos registros no arquivo. Se omitido, o padrão é 8192.
<i>sourceRowPath</i>	O caminho JSON usado para delimitar registros no arquivo de origem. Requerido.
<i>srcEncoding</i>	Opcional. Uma string terminada por nulo contendo a codificação (utf8,utf8n,utf16be,utf16le,utf32be,utf32le). Se omitido, o padrão é 'utf8'
<i>destinationgroup</i>	Uma string terminada por nulo contendo o nome do grupo para distribuir o arquivo.
<i>destinationlogicalname</i>	Uma string terminada por nulo contendo o nome lógico do arquivo a ser criado.
<i>timeout</i>	Opcional. Um valor inteiro indicando a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como zero (0), o controle de execução retornará imediatamente para a workunit ECL sem aguardar a conclusão da workunit do DFU.
<i>espserverIPport</i>	Opcional. Uma sequência terminada por nulo contendo o protocolo IP, porta e diretório, ou o equivalente ao DNS, do programa do servidor ESP. Geralmente, é o mesmo IP e porta que o ECL Watch, com "/FileSpray" anexado. Se omitido, o padrão é o valor contido no atributo lib_system.ws_fs_server.
<i>maxConnections</i>	Opcional. Um número inteiro especificando o número máximo de conexões. Se omitido, o padrão é um (1).
<i>allowoverwrite</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando se o novo arquivo deve sobrescrever um arquivo existente com o mesmo nome. Se omitido, o padrão é FALSE.
<i>replicate</i>	Opcional. Uma flag booleana TRUE ou FALSE, indicando se o novo arquivo deve ser replicado. Se omitido, o padrão é FALSE
<i>compress</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando se deseja compactar o novo arquivo. Se omitido, o padrão é FALSE
<i>failIfNoSourceFile</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando se um arquivo ausente aciona uma falha. Se omitido, o padrão é FALSE.
<i>expireDays</i>	Opcional. Especifica que o arquivo é um arquivo temporário a ser excluído automaticamente após o número especificado de dias desde que o arquivo foi lido. Se omitido, o padrão é -1 (nunca expira). Se definido como 0, o arquivo será excluído automaticamente quando atingir o limite definido na configuração expiryDefault do Servidor Sasha.
<i>dfuServerQueue</i>	Nome da fila do servidor DFU de destino. O padrão é " (vazio) para a primeira fila do DFU no ambiente.
<i>noSplit</i>	Opcional. Uma flag booleana TRUE ou FALSE indicando não dividir uma parte do arquivo em várias partes de destino. O padrão é FALSE.

Referência de Biblioteca Padrão
Movimentação de Arquivos

<i>dfuwuid</i>	O nome do atributo para receber a sting terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para o job..
<i>username</i>	Opcional. String contendo um nome de usuário a ser usado para acesso autenticado ao processo ESP; uma string de valor vazia indica que nenhuma autenticação de usuário é necessária. Se omitido, o padrão é uma string vazia.
<i>userPw:</i>	Opcional. String contendo a senha a ser usada com o usuário citado no argumento <i>username</i> ; se o <i>username</i> estiver vazio, isso será ignorado. Se omitido, o padrão é uma string vazia.
<i>Return:</i>	fSprayJson retorna uma string terminada por nulo contendo o ID da Workunit DFU (DFUWUID).

A função **SprayJson** pega o arquivo JSON preparado da landing zone e os distribui ao longo dos nós dos cluster de destino, produzindo um arquivo JSON bem preparado em cada nó.

Exemplo:

```
STD.File.SprayJson('10.150.50.14','/var/lib/HPCCSystems/mydropzone/colors.json',,
    '/',, 'mythor', 'examples::colors.json', -1,
    'http://10.150.50.12:8010/FileSpray');
```

WaitDfuWorkunit

STD.File.WaitDfuWorkunit(*dfuwuid* [,*timeout*] [,*espserverIPport*])

<i>dfuwuid</i>	Uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID) do job a ser abortado. Esse valor é retornado pelas versões das funções Copy, SprayFixed, SprayVariable, SprayXML e Despray FileServices "com nome começando por f".
<i>timeout</i>	Opcional. Um valor inteiro que indica a configuração de tempo limite. Se omitido, o padrão é -1. Se definido como (0), o controle da execução retorna imediatamente à workunit ECL sem esperar a conclusão da workunit DFU.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
Return:	WaitDfuWorkunit retorna uma string terminada por nulo que contém a string de status final da Workunit DFU (como: scheduled, queued, started, aborted, failed, finished, or monitoring).

A função **WaitDfuWorkunit** aguarda a conclusão da workunit DFU especificada. Normalmente, essa workunit foi iniciada com o parâmetro *timeout* definido como zero(0).

Exemplo:

```
STD.File.WaitDfuWorkunit( 'D20051108-143758' );
```

SetExpireDays

STD.File.SetExpireDays(*lfn*, *expireDays*)

<i>lfn</i>	Uma string contendo o nome lógico do arquivo.
<i>expireDays</i>	Número de dias antes que o arquivo expire. Definir como 0 especifica o uso do valor de expiração padrão do sistema (especificado no atributo <i>ExpiryDefault</i> do servidor Sasha)

A ação **SetExpireDays** configura o critério de expiração (o atributo *expireDays*). O arquivo é excluído pelo servidor Sasha quando não é acessado pelo número de dias especificado.

Exemplo:

```
STD.File.SetExpireDays('~samples::myscope::myfile',30);  
//file expires and is deleted after 30 days w/o access
```

Veja também: GetExpireDays, ClearExpireDays

GetExpireDays

STD.File.GetExpireDays(*lfn*)

<i>lfn</i>	Uma string contendo o nome lógico do arquivo.
------------	---

A função **GetExpireDays** recupera os critérios de expiração de um arquivo lógico (atributo *expireDays*). O retorno de -1 indica que não há configuração de expiração.

Exemplo:

```
A := STD.File.GetExpireDays('~samples::myscope::myfile');  
//returns a file's expireDays
```

Veja também: SetExpireDays, ClearExpireDays

ClearExpireDays

STD.File.ClearExpireDays(*lfn*)

<i>lfn</i>	Uma string contendo o nome lógico do arquivo.
------------	---

A função **ClearExpireDays** limpa os critérios de expiração (atributo *expireDays*).

Exemplo:

```
A := STD.File.ClearExpireDays('~samples::myscope::myfile');  
//clears a file's expireDays
```

Veja também: GetExpireDays, SetExpireDays

Manipulação de String

CleanAccents

STD.Uni.CleanAccents(*source*)

<i>source</i>	Uma string que contém os dados a serem higienizados.
Return:	CleanAccents retorna um valor UNICODE.

A função **CleanAccents** retorna a string *source* com todos os caracteres acentuados substituídos pelos não acentuados.

Exemplo:

```
UNICODE A := STD.Uni.CleanAccents(u'caf\u00E9'); //café - U+00E9 is lowercase e with acute
//a contains 'cafe'
```

CleanSpaces

STD.Str.CleanSpaces(*Source*)

STD.Uni.CleanSpaces(*Source*)

<i>source</i>	Uma string que contém os dados a serem higienizados.
Return:	CleanSpaces retorna um valor STRING ou UNICODE, conforme o caso.

odas as variações da função **CleanSpaces** retornam uma string *source* com todas as instâncias de vários caracteres de espaço adjacentes (2 ou mais espaços juntos, ou um caractere de tabulação) reduzidos a um único caractere de espaço. Os espaços à esquerda e à direita também são eliminados.

Exemplo:

```
A := STD.Str.CleanSpaces('ABCDE ABCDE');  
  //A contains 'ABCDE ABCDE'  
UNICODE C := STD.Uni.CleanSpaces(U'ABCDE ABCDE');  //C contains U'ABCDE ABCDE'
```

CompareAtStrength

STD.Uni.CompareAtStrength(*source1*, *source2*, *strength*)

STD.Uni.LocaleCompareAtStrength(*source1*, *source2*, *locale*, *strength*)

<i>source1</i>	Uma string que contém os dados a serem comparados.
<i>source2</i>	Uma string que contém os dados a serem comparados.
<i>strength</i>	Um valor inteiro que indica a forma de comparação. Os valores válidos são:
	1 ignora acentos e maiúsculas/minúsculas e diferencia apenas as letras.
	2 ignora maiúsculas/minúsculas mas diferencia os acentos.
	3 diferencia entre acentos e maiúsculas/minúsculas, mas ignora diferenças entre Hiragana e Katakana, por exemplo.
	4 diferencia entre acentos, maiúsculas/minúsculas e Hiragana e Katakana, por exemplo, mas ignora marcas de cantilena do hebreu, por exemplo.
	5 diferencia entre todas as strings cujas formas decompostas canonicamente (NFD – Forma de normalização D) não são idênticas
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	CompareAtStrength retorna um valor INTEGER.

As funções **CompareAtStrength** retornam zero (0) se as strings *source1* e *source2* contiverem os mesmos dados, ignorando quaisquer diferenças no caso das letras. Essas funções retornam um negativo (-1) se *source1* < *source2* ou um positivo (1) se *source1* > *source2*.

Exemplo:

```
base := u'caf\u00E9'; // U+00E9 is lowercase e with acute
prim := u'coffee shop'; // 1st difference, different letters
seco := u'cafe'; // 2nd difference, accents (no acute)
tert := u'Caf\u00C9'; // 3rd, caps (U+00C9 is u/c E + acute)

A := STD.Uni.CompareAtStrength(base, prim, 1) != 0;
// base and prim differ at all strengths

A := STD.Uni.CompareAtStrength(base, seco, 1) = 0;
// base and seco same at strength 1 (differ only at strength 2)

A := STD.Uni.CompareAtStrength(base, tert, 1) = 0;
// base and tert same at strength 1 (differ only at strength 3)

A := STD.Uni.CompareAtStrength(base, seco, 2) != 0;
// base and seco differ at strength 2

A := STD.Uni.CompareAtStrength(base, tert, 2) = 0;
// base and tert same at strength 2 (differ only at strength 3)

A := STD.Uni.CompareAtStrength(base, seco, 3) != 0;
// base and seco differ at strength 2

A := STD.Uni.CompareAtStrength(base, tert, 3) != 0;
// base and tert differ at strength 3
```

CompareIgnoreCase

STD.Str.CompareIgnoreCase(*source1*, *source2*)

STD.Uni.CompareIgnoreCase(*source1*, *source2*)

STD.Uni.LocaleCompareIgnoreCase(*source1*, *source2*, *locale*)

<i>source1</i>	Uma string que contém os dados a serem comparados.
<i>source2</i>	Uma string que contém os dados a serem comparados.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	CompareIgnoreCase retorna um valor INTEGER.

A função **CompareIgnoreCase** retorna TRUE se as strings *source1* e *source2* contêm os mesmos dados, sem fazer distinção de maiúsculas e minúsculas. Essas funções retornam (-1) se *source1* < *source2* ou positiva (1) se *source1* > *source2*.

Exemplo:

```
A := STD.Str.CompareIgnoreCase('ABCDE', 'abcde');  
//A contains 0 -- they "match"  
  
B := STD.Str.CompareIgnoreCase('ABCDE', 'edcba');  
//B contains -1 -- they do not "match"
```

Contains

STD.Str.Contains(*source*, *pattern*, *nocase*)

STD.Uni.Contains(*source*, *pattern*, *nocase*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>pattern</i>	Uma string que contém os caracteres a serem comparados. Uma string vazia (") sempre retorna verdadeiro.
<i>nocase</i>	Um valor booleano verdadeiro ou falso que indica se é necessário fazer distinção entre maiúsculas e minúsculas.
Return:	Contém retorna um valor BOOLEAN

As funções **Contains** retornam verdadeiro se todos os caracteres *padrão* aparecem em *source* , caso contrário, retornam falso.

Exemplo:

```
A := STD.Str.Contains(
  'the quick brown fox jumps over the lazy dog',
  'ABCdefghijklmnopqrstuvwxyz', true); //returns TRUE

B:= STD.Str.Contains(
  'the speedy ochre vixen leapt over the indolent retriever',
  'abcdefghijklmnopqrstuvwxyz', false); //returns FALSE -- 'z' is missing
```


CountWords

STD.Str.CountWords(*source*, *separator*)

<i>source</i>	Uma string que contém as palavras a serem contadas.
<i>separator</i>	Uma string que contém o delimitador de palavras a ser usado.
Return:	CountWords retorna um valor inteiro.

A função **CountWords** retorna o número de palavras na string *source* baseado no separador *especificado*.

Exemplo:

```
IMPORT Std;

str1 := 'a word a day keeps the doctor away';
str2 := 'a|word|a|day|keeps|the|doctor|away';

output(LENGTH(TRIM(Str1,LEFT,RIGHT)) - LENGTH(TRIM(Str1,ALL)) + 1);
//finds eight words by removing spaces
STD.Str.CountWords(str1, ' '); //finds eight words based on space delimiter
STD.Str.CountWords(str2, '|'); //finds eight words based on bar delimiter
```

DecodeBase64

STD.Str.DecodeBase64(*value*)

<i>value</i>	Um valor de STRING que contem o dado ser decodificado.
Return:	DecodeBase64 retorna um valor DATA

A função **DecodeBase64** retorna um valor DATA que contém os dados binários decodificados.

Exemplo:

```
IMPORT STD;  
str:='AQIDBAU=';  
DecodedData:= STD.Str.DecodeBase64(str);  
DecodedData;
```

Ver também: EncodeBase64

EditDistance

STD.Str.EditDistance(*string1*, *string2*, *radius*)

STD.Uni.EditDistance(*string1*, *string2*, *locale*, *radius*)

<i>string1</i>	A primeira string de um par de strings a ser comparado.
<i>string2</i>	A segunda string de um par de strings a ser comparado.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
<i>radius</i>	Opcional. O máximo aceitável para distance editável, ou 0 para sem limite. Padrão é 0.
Return:	EditDistance retorna um valor UNSIGNED4.

A função **EditDistance** retorna uma pontuação do algoritmo de distância de Levenshtein padrão para a distância de edição entre *string1* e *string2*. Essa pontuação reflete o número mínimo de operações necessárias para transformar *string1* em *string2*.

Se a distância de edição for maior que o *raio* retornará um valor arbitrário $> \text{raio}$, mas pode não ser exato. Isso permite que a função termine mais rapidamente se as strings são significativamente diferentes.

Exemplo:

```
STD.Str.EditDistance('CAT','CAT'); //returns 0
STD.Str.EditDistance('CAT','BAT'); //returns 1
STD.Str.EditDistance('BAT','BAIT'); //returns 1
STD.Str.EditDistance('CAT','BAIT'); //returns 2
STD.Str.EditDistance('CARTMAN','BATMAN'); //returns 2
STD.Str.EditDistance('CARTMAN','BATMAN',1); //returns arbitrary number > 1
```

EditDistanceWithinRadius

STD.Str.EditDistanceWithinRadius(*string1*, *string2*, *radius*)

STD.Uni.EditDistanceWithinRadius(*string1*, *string2*, *radius*, *locale*)

<i>string1</i>	A primeira string de um par de strings a ser comparado.
<i>string2</i>	A segunda string de um par de strings a ser comparado.
<i>radius</i>	Um inteiro que especifica a distância máxima de edição aceitável.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	EditDistanceWithinRadius retorna um valor BOOLEAN.

A função **EditDistanceWithinRadius** retorna TRUE se a distância de edição entre *string1* e *string2* está dentro de *radius*. Os espaços à direita e à esquerda nas duas strings são eliminados antes da comparação.

Exemplo:

```
IMPORT STD;
STD.Str.EditDistance('CAT','BAIT');           //returns 2

STD.Str.EditDistanceWithinRadius('CAT','BAIT',1); //returns FALSE
STD.Str.EditDistanceWithinRadius('CAT','BAIT',2); //returns TRUE
```

EncodeBase64

STD.Str.EncodeBase64(*value*)

<i>value</i>	Um valor DATA que contém os dados a serem codificados.
Return:	EncodeBase64 returns a STRING value.

A função **EncodeBase64** retorna uma STRING contendo os dados binários codificados em Base64.

Exemplo:

```
IMPORT STD;  
dat:=X'0102030405';  
EncodedStr:= STD.Str.EncodeBase64(dat);  
EncodedStr;
```

Ver também: DecodeBase64

EndsWith

STD.Str.EndsWith(*source*, *suffix*)

<i>source</i>	A string a ser pesquisada.
<i>suffix</i>	A string a ser encontrada.
Return:	EndsWith retorna um valor BOOLEAN

A função **EndsWith** retorna TRUE se o *source* termina com o texto no parâmetro *suffix* .

Exemplo:

```
IMPORT STD;
STD.Str.EndsWith('a word away','away');    //returns TRUE
STD.Str.EndsWith('a word a way','away');    //returns FALSE
```

EqualIgnoreCase

STD.Str.EqualIgnoreCase(*source1*, *source2*)

<i>source1</i>	Uma string que contém os dados a serem comparados.
<i>source2</i>	Uma string que contém os dados a serem comparados.
Return:	EqualIgnoreCase retorna um valor BOOLEAN

A função **EqualIgnoreCase** retorna TRUE se as strings *source1* e *source2* contêm os mesmos dados, sem fazer distinção de maiúsculas e minúsculas.

Exemplo:

```
A := STD.Str.EqualIgnoreCase('ABCDE','abcde');  
//A contains TRUE -- they "match"  
  
B := STD.Str.CompareIgnoreCase('ABCDE','edcba');  
//B contains FALSE -- they do not "match"
```

ExcludeFirstWord

STD.Str.ExcludeFirstWord(*text*)

<i>text</i>	Uma string que contém palavras separadas por espaços em branco.
Return:	ExcludeFirstWord returns um valor STRING

A função **ExcludeFirstWord** retorna a string de *text* com a primeira palavra removida. As palavras são separadas por um ou mais caracteres de espaço em branco. O espaço em branco antes da primeira palavra também é removido.

Exemplo:

```
A := STD.Str.ExcludeFirstWord('The quick brown fox');  
//A contains 'quick brown fox'
```


ExcludeLastWord

STD.Str.ExcludeLastWord(*text*)

<i>text</i>	Uma string que contém palavras separadas por espaços em branco.
Return:	ExcludeLastWord retorna um valor STRING

A função **ExcludeLastWord** retorna a string de *text* com a última palavra removida. As palavras são separadas por um ou mais caracteres de espaço em branco. O espaço em branco após a última palavra também é removido.

Exemplo:

```
A := STD.Str.ExcludeLastWord('The quick brown fox');  
//A contains 'The quick brown'
```

ExcludeNthWord

STD.Str.ExcludeNthWord(*text*, *n*)

<i>text</i>	Uma string que contém palavras separadas por espaços em branco.
<i>n</i>	Um inteiro que contém a posição ordinal da palavra a ser removida.
Return:	ExcludeNthWord retorna um valor STRING

A função **ExcludeNthWord** retorna a string *text* com *n* palavras com th removidas. As palavras são separadas por um ou mais caracteres de espaço em branco. O espaço em branco após a *n-ésima* palavra também será removida (juntamente com o espaço em branco antes, se *n*=1).

Exemplo:

```
A := STD.Str.ExcludeNthWord('The quick brown fox',2);  
//A contains 'The brown fox'
```

Extract

STD.Str.Extract(*source*, *instance*)

STD.Uni.Extract(*source*, *instance*)

<i>source</i>	Uma string que contém uma lista delimitada por vírgulas com dados.
<i>instance</i>	Um inteiro que especifica a posição ordinal do item de dados dentro do <i>source</i> a ser retornado.
Return:	Extract retorna um valor STRING ou UNICODE, conforme o caso.

A função **Extract** retorna os dados da posição ordinal especificada pela *instance* dentro da string *source* delimitada por vírgulas.

Exemplo:

```
//all these examples result in 'Success'

A := IF(STD.Str.Extract('AB,CD,,G,E',0) = ' ',
    'Success',
    'Failure -1');

B := IF(STD.Str.Extract('AB,CD,,G,E',1) = 'AB',
    'Success',
    'Failure -2');

C := IF(STD.Str.Extract('AB,CD,,G,E',2) = 'CD',
    'Success',
    'Failure -3');

D := IF(STD.Str.Extract('AB,CD,,G,E',3) = ' ',
    'Success',
    'Failure -4');

E := IF(STD.Str.Extract('AB,CD,,G,E',4) = 'G',
    'Success',
    'Failure -5');

F := IF(STD.Str.Extract('AB,CD,,G,E',5) = 'E',
    'Success',
    'Failure -6');

G := IF(STD.Str.Extract('AB,CD,,G,E',6) = ' ',
    'Success',
    'Failure -7');
```

ExtractMultiple

STD.Str.ExtractMultiple(*source*, *instance*)

STD.Uni.ExtractMultiple(*source*, *instance*)

<i>source</i>	Uma string que contém uma lista delimitada por vírgulas com dados.
<i>mask</i>	Uma bitmask que especifica a posição ordinal do item de dados dentro de <i>source</i> a ser retornado, onde bit 0 é o item 1, bit 1 é o item 2 e assim por diante.
Return:	ExtractMultiple retorna um valor STRING ou UNICODE, conforme o caso.

A função **ExtractMultiple** retorna os dados nas posições do bitmask especificados por *mask* dentro da string delimitada por vírgulas *source* string, onde bit 0 é o item 1, bit 1 é o item 2 e assim por diante.

Exemplo:

```
IMPORT STD;
MyTestString:= 'You, only, live, twice';
STD.Str.ExtractMultiple(MyTestString, 0b10011 ); //returns 'You, only'
```

Filter

STD.Str.Filter(*source*, *filterstring*)

STD.Uni.Filter(*source*, *filterstring*)

<i>source</i>	Uma string que contém os dados a serem filtrados.
<i>filterstring</i>	Uma string que contém os caracteres a serem usados como filtro.
Return:	Filtro retorna um valor STRING ou UNICODE, conforme o caso.

As funções **StringFilter** retornam a string *source* com todos os caracteres, exceto os caracteres removidos em *filterstring*.

Exemplo:

```
//all these examples result in 'Success'

A := IF(STD.Str.Filter('ADCBE', 'BD') = 'DB',
  'Success',
  'Failure - 1');

B := IF(STD.Str.Filter('ADCBEREED', 'BDG') = 'DBBD',
  'Success',
  'Failure - 2');

C := IF(STD.Str.Filter('ADCBE', '') = '',
  'Success',
  'Failure - 3');

D := IF(STD.Str.Filter('', 'BD') = '',
  'Success',
  'Failure - 4');

E := IF(STD.Str.Filter('ABCDE', 'EDCBA') = 'ABCDE',
  'Success',
  'Failure - 5');
```

FilterOut

STD.Str.FilterOut(*source*, *filterstring*)

STD.Uni.FilterOut(*source*, *filterstring*)

<i>source</i>	Uma string que contém os dados a serem filtrados.
<i>filterstring</i>	Uma string que contém os caracteres a serem usados como filtro.
Return:	FilterOut retorna uma STRING ou valor UNICODE, conforme apropriado

As funções **FilterOut** retornam a string *source* com todos os caracteres em *filterstring* removidos.

Exemplo:

```
//all these examples result in 'Success'

A := IF(STD.Str.FilterOut('ABCDE', 'BD') = 'ACE',
    'Success',
    'Failure - 1');

B := IF(STD.Str.FilterOut('ABCDEABCDE', 'BD') = 'ACEACE',
    'Success',
    'Failure - 2');

C := IF(STD.Str.FilterOut('ABCDEABCDE', '') = 'ABCDEABCDE',
    'Success',
    'Failure - 3');

D := IF(STD.Str.FilterOut('', 'BD') = '',
    'Success',
    'Failure - 4');
```

Find

STD.Str.Find(*source*, *target*, *instance*)

STD.Uni.Find(*source*, *target*, *instance*)

STD.Uni.LocaleFind(*source*, *target*, *instance*, *locale*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>target</i>	Uma string que contém a substring a ser pesquisada.
<i>instance</i>	Um inteiro que especifica qual ocorrência de <i>target</i> deve ser encontrada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	Find retorna um valor INTEGER.

As funções **Find** retornam a posição inicial de índice dentro da string *source* da *instance* especificada da string *target*. Se *target* não for encontrado ou a *instance* especificada for maior que o número de ocorrências do *target* em *source*, **Find** retornará zero (0).

Exemplo:

```
A := IF(STD.Str.Find('ABCDE', 'BC', 1) = 2,
    'Success',
    'Failure - 1'); //success

B := IF(STD.Str.Find('ABCDEABCDE', 'BC', 2) = 7,
    'Success',
    'Failure - 2'); //success

C := IF(STD.Str.Find('ABCDEABCDE', '') = 0,
    'Success',
    'Failure - 3'); //syntax error, missing 3rd parameter

D := IF(STD.Str.Find('', 'BD', 1) = 0,
    'Success',
    'Failure - 4'); //success
```

FindCount

STD.Str.FindCount(*source*, *target*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>target</i>	Uma string que contém a substring a ser pesquisada.
Return:	StringFindCount retorna um valor INTEGER.

A função **FindCount** retorna o número de instâncias não sobrepostas da string *target* dentro da string *source* .

Exemplo:

```
A := IF(STD.Str.FindCount('ABCDE', 'BC') = 1,
    'Success',
    'Failure - 1'); //success

B := IF(STD.Str.FindCount('ABCDEABCDE', 'BC') = 1,
    'Success',
    'Failure - 1'); //failure
```


FindAtStrength

STD.Uni.LocaleFindAtStrength(*source*, *target*, *instance*, *locale*, *strength*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>target</i>	Uma string que contém a substring a ser pesquisada.
<i>instance</i>	Um inteiro que especifica qual ocorrência de <i>target</i> deve ser encontrada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
<i>strength</i>	Um valor inteiro que indica como comparar. Os valores válidos são: 1 ignora acentos e maiúsculas/minúsculas e diferencia apenas as letras. 2 ignora maiúsculas/minúsculas mas diferencia os acentos. 3 diferencia entre acentos e maiúsculas/minúsculas, mas ignora diferenças entre Hiragana e Katakana, por exemplo. 4 diferencia entre acentos, maiúsculas/minúsculas e Hiragana e Katakana, por exemplo, mas ignora marcas de cantilena do hebreu, por exemplo. 5 diferencia entre todas as strings cujas formas decompostas canonicamente (NFD – Forma de normalização D) não são idênticas
Return:	FindAtStrength retorna um valor INTEGER.

A função **FindAtStrength** retorna a posição inicial de índice dentro da string *source* da *instance* especificada da string *target*. *source* string da *instância* especificada da string de *destino*. Se *target* não for encontrado ou a *instance* especificada for maior que o número de ocorrências de *target* em *source*, StringFind retornará zero (0).

Exemplo:

```
base := u'caf\u00E9'; // U+00E9 is lowercase e with acute
prim := u'coffee shop'; // 1st difference, different letters
seco := u'cafe'; // 2nd difference, accents (no acute)
tert := u'Caf\u00C9'; // 3rd, caps (U+00C9 is u/c E + acute)
search := seco + tert + base;
STD.Uni.LocaleFindAtStrength(search, base, 1, 'fr', 1) = 1;
// at strength 1, base matches seco (only secondary diffs)
STD.Uni.LocaleFindAtStrength(search, base, 1, 'fr', 2) = 5;
// at strength 2, base matches tert (only tertiary diffs)
STD.Uni.LocaleFindAtStrength(search, base, 1, 'fr', 3) = 9;
// at strength 3, base doesn't match either seco or tert
STD.Uni.LocaleFindAtStrength(u'le caf\u00E9 vert',
    u'cafe', 1, 'fr', 2) = 4;
// however, an accent on the source,
STD.Uni.LocaleFindAtStrength(u'le caf\u00E9 vert',
    u'cafe', 1, 'fr', 3) = 4;
// rather than on the pattern,
STD.Uni.LocaleFindAtStrength(u'le caf\u00E9 vert',
    u'cafe', 1, 'fr', 4) = 4;
// is ignored at strengths up to 4,
STD.Uni.LocaleFindAtStrength(u'le caf\u00E9 vert',
    u'cafe', 1, 'fr', 5) = 0;
// and only counts at strength 5
```

FindAtStrengthReplace

STD.Uni.LocaleFindAtStrengthReplace(*source*, *target*, *replacement*, *locale*, *strength*)

<i>source</i>	Uma string contendo os dados para pesquisar.
<i>target</i>	Uma string que contém a substring a ser pesquisada.
<i>replacement</i>	Uma string que contém os dados de substituição.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
<i>strength</i>	Um valor inteiro que indica como comparar. Os valores válidos são:
	1 ignora acentos e maiúsculas/minúsculas e diferencia apenas as letras.
	2 ignora maiúsculas/minúsculas mas diferencia os acentos.
	3 diferencia entre acentos e maiúsculas/minúsculas, mas ignora diferenças entre Hiragana e Katakana, por exemplo.
	4 diferencia entre acentos, maiúsculas/minúsculas e Hiragana e Katakana, por exemplo, mas ignora marcas de cantilena do hebreu, por exemplo.
	5 diferencia entre todas as strings cujas formas decompostas canonicamente (NFD – Forma de normalização D) não são idênticas
Return:	FindAtStrengthReplace retorna um valor UNICODE.

As funções **FindAtStrengthReplace** retornam a string *source* com a string *replacement* substituindo todas as instâncias da string *target* . Se a string *target* não estiver na string *source* , a string *source* será retornada sem alterações.

Exemplo:

```
STD.Uni.LocaleFindAtStrengthReplace(u'e\u00E8E\u00C9eE',  
    u'e\u00E9', u'xyz', 'fr', 1) = u'xyzxyzxyz';  
STD.Uni.LocaleFindAtStrengthReplace(u'e\u00E8E\u00C9eE',  
    u'e\u00E9', u'xyz', 'fr', 2) = u'e\u00E8xyzeE';  
STD.Uni.LocaleFindAtStrengthReplace(u'e\u00E8E\u00C9eE',  
    u'e\u00E9', u'xyz', 'fr', 3) = u'e\u00E8E\u00C9eE';
```

FindReplace

STD.Str.FindReplace(*source*, *target*, *replacement*)

STD.Uni.FindReplace(*source*, *target*, *replacement*)

STD.Uni.LocaleFindReplace(*source*, *target*, *replacement*, *locale*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>target</i>	Uma string que contém a substring a ser pesquisada.
<i>replacement</i>	Uma string que contém os dados de substituição.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	FindReplace retorna um valor STRING ou UNICODE, conforme apropriado.

As funções **FindReplace** retornam a string *source* com as strings *replacement* substituídas em todas as instâncias da string *target* . Se a string *target* não estiver na string *source* , a string *source* será retornada sem alterações.

Exemplo:

```
A := STD.Str.FindReplace('ABCDEABCDE', 'BC', 'XY');
//A contains 'AXYDEAXYDE'
A := STD.Uni.FindReplace(u'abcde', u'a', u'AAAAA');
//A contains u'AAAAAbcde'
A := STD.Uni.FindReplace(u'aaaaa', u'aa', u'b');
//A contains u'bba'
A := STD.Uni.FindReplace(u'aaaaaa', u'aa', u'b');
//A contains u'bbb'
A := STD.Uni.LocaleFindReplace(u'gh\u0131klm', u'hyk', u'XxXxX', 'lt');
//A contains u'gXxXxXlm'
A := STD.Uni.LocaleFindReplace(u'gh\u0131klm', u'hyk', u'X', 'lt');
//A contains u'gXlm'
```

FindWord

STD.Str.FindWord(*src*, *word*, *ignore_case*)

STD.Uni.FindWord(*src*, *word*, *ignore_case*)

<i>src</i>	Uma string que contém os dados a serem pesquisados.
<i>word</i>	Uma string que contém a substring a ser pesquisada.
<i>ignore_case</i>	Um valor booleano verdadeiro ou falso que indica se maiúsculas e minúsculas devem ser diferenciadas.
Return:	FindWord retorna um valor BOOLEAN

A função **FindWord** retorna TRUE se a string *word* é encontrada na string *src* .

Exemplo:

```
IMPORT STD;
src := 'Now is the winter of our discontent';
word := 'now';

STD.Str.FindWord(src,word);           // false - case not ignored
STD.Str.FindWord(src,word,TRUE);     // true  - with case ignored word is found
```

FromHexPairs

STD.Str.FromHexPairs(*source*)

<i>source</i>	A string que contém os pares hexadecimais a serem processados.
Retorno:	FromHexPairs retorna um valor de dados, com cada byte criado de um par de dígitos hexadecimais.

A função **FromHexPairs** retorna um valor de dados, com cada byte criado de um par de dígitos hexadecimais.

Exemplo:

```
A := STD.Str.FromHexPairs('0001FF80');
```

GetNthWord

STD.Str.GetNthWord(*source*, *instance*)

STD.Uni.GetNthWord (*source*, *instance* [, *locale*])

<i>source</i>	Uma string que contém as palavras delimitadas por espaço.
<i>instance</i>	Um inteiro que especifica a palavra a ser retornada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	GetNthWord retorna um valor STRING.

A função **GetNthWord** retorna a palavra na posição *instance* na string *source*.

Exemplo:

```
IMPORT Std;

str1 := 'a word a day keeps the doctor away';

STD.Str.GetNthWord(str1,2);    //returns "word"
```

RemoveSuffix

STD.Str.RemoveSuffix(*source*, *suffix*)

<i>source</i>	A string a ser pesquisada.
<i>suffix</i>	O final de string a ser removido.
Return:	RemoveSuffix retorna um valor de string.

A função **RemoveSuffix** retorna a string *source* com o texto do parâmetro *suffix* removido do final dessa string. Se a string *source* não terminar com *suffix*, a string *source* será retornada sem alterações.

Exemplo:

```
IMPORT STD;
STD.Str.RemoveSuffix('a word away','away'); //returns 'a word'
STD.Str.RemoveSuffix('a word a way','away'); //returns 'a word a way'
```

Repeat

STD.Str.Repeat(*text*, *n*)

<i>text</i>	A string a ser repetida (o comprimento máximo é 255 caracteres).
<i>n</i>	O número de repetições.
Return:	Repeat retorna uma STRING que contém <i>n</i> concatenações do texto da string.

A função **Repeat** retorna a string *source* repetida *n* vezes.

Exemplo:

```
A := STD.Str.Repeat('ABC',3); //A contains 'ABCABCABC'
```


Reverse

STD.Str.Reverse(*source*)

STD.Uni.Reverse(*source*)

<i>source</i>	A string que contém os dados a serem revertidos.
Return:	Reverse retorna um valor STRING ou UNICODE, conforme o caso.

As funções **Reverse** retornam a string *source* com todos os caracteres na ordem invertida.

Exemplo:

```
A := STD.Str.Reverse('ABCDE'); //A contains 'EDCBA'
```

SplitWords

STD.Str.SplitWords((*source1*, *source2*, *strength*) [])

<i>source</i>	Uma string que contém as palavras a serem extraídas.
<i>separator</i>	Uma string que contém o delimitador de palavras a ser usado.
<i>allowblank</i>	Opcional. Se TRUE, especifica que itens em branco são permitidos no resultado. Se omitido, o padrão é FALSE.
Return:	SplitWords retorna um valor SET OF STRING.

A função **SplitWords** retorna a lista de palavras na string *source* dividida pelo separador *especificado*.

Exemplo:

```
IMPORT Std;

str1 := 'a word a day keeps the doctor away';
str2 := 'a|word|a|day|keeps|the|doctor|away';

STD.Str.SplitWords(str1, ' ');
//returns ['a', 'word', 'a', 'day', 'keeps', 'the', 'doctor', 'away']

STD.Str.SplitWords(str2, '|');
//returns ['a', 'word', 'a', 'day', 'keeps', 'the', 'doctor', 'away']
```

SubstituteExcluded

STD.Str.SubstituteExcluded(*source*, *target*, *replacement*)

STD.Uni.SubstituteExcluded(*source*, *target*, *replacement*)

<i>source</i>	Uma string contendo os dados para pesquisar.
<i>target</i>	Uma string contendo os caractere para pesquisar.
<i>replacement</i>	Uma string que contém o caractere de substituição como seu primeiro caractere.
Return:	SubstitutoExcluído retorna um valor STRING ou UNICODE, conforme apropriado.

As funções **SubstituteExcluded** retornam a string *source* com o caractere *replacement* substituído em todos os caracteres, exceto os que estão na string *target* . Se a string *target* não estiver na string *source* , a string *source* será retornada com todos os caracteres substituídos pelo caractere *replacement* .

Exemplo:

```
IMPORT STD;
A := STD.Uni.SubstituteExcluded(u'abcdeabcdec', u'cd', u'x');
//A contains u'xxcdxxcdxc';
```

SubstituteIncluded

STD.Str.SubstituteIncluded(*source*, *target*, *replacement*)

STD.Uni.SubstituteIncluded(*source*, *target*, *replacement*)

<i>source</i>	Uma string contendo os dados para pesquisar.
<i>target</i>	Uma string contendo os caractere para pesquisar.
<i>replacement</i>	Uma string que contém o caractere de substituição como seu primeiro caractere.
Return:	SubstituteIncluded retorna um valor STRING ou UNICODE, conforme apropriado.

As funções **SubstituteIncluded** retornam a string *source* com o caractere *replacement* substituído em todos os caracteres que existem em ambas as strings *source* e *target* . Se nenhum caractere da string *target* estiver na string *source* , retornará a string *source* inalterada.

Exemplo:

```
IMPORT STD;
A := STD.Uni.SubstituteIncluded(u'abcde', u'cd', u'x');
//A contains u'abxxe';
B := STD.Str.SubstituteIncluded('abcabc', 'ac', 'yz');
//B contains 'ybyyby'
```

StartsWith

STD.Str.StartsWith(*source*, *prefix*)

<i>source</i>	A string a ser pesquisada.
<i>prefix</i>	A string a ser encontrada.
Return:	StartsWith retorna um valor BOOLEAN

A função **StartsWith** retorna TRUE se o *source* começa com o texto no parâmetro *prefix* .

Exemplo:

```
IMPORT STD;
STD.Str.StartsWith('a word away','a word');    //returns TRUE
STD.Str.StartsWith('a word away','aword');     //returns FALSE
```

ToHexPairs

STD.Str.ToHexPairs(*Source*)

<i>Source</i>	O valor de dados que deve ser expandido como uma sequência de pares hexadecimais.
Retorno:	ToHexPairs retorna uma string que contém uma sequência de pares hexadecimais.

A função **ToHexPairs** converte o valor de dados em uma sequência de pares hexadecimais.

Exemplo:

```
A := STD.Str.ToHexPairs(D'\000\001\377\200');
```

ToLowerCase

STD.Str.ToLowerCase(*source*)

STD.Uni.ToLowerCase(*source*)

STD.Uni.LocaleToLowerCase(*source*, *locale*)

<i>source</i>	Uma string que contém os dados cuja formatação de maiúsculas e minúsculas será alterada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	ToLowerCase retorna um valor STRING ou UNICODE, conforme apropriado.

As funções **ToLowerCase** retornam a string *source* com todos os caracteres maiúsculos convertidos em minúsculos.

Exemplo:

```
A := STD.Str.ToLowerCase('ABCDE'); //A contains 'abcde'
```

ToTitleCase

STD.Str.ToTitleCase(*source*)

STD.Uni.ToTitleCase(*source*)

STD.Uni.LocaleToTitleCase(*source*, *locale*)

<i>source</i>	Uma string que contém os dados cuja formatação de maiúsculas e minúsculas será alterada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Retorno:	ToTitleCase retorna um valor STRING ou UNICODE, conforme apropriado.

As funções **ToTitleCase** retornam a string *source* com a primeira letra de cada palavra em maiúscula e todas as demais letras em minúsculas.

Exemplo:

```
A := STD.Str.ToTitleCase('ABCDE ABCDE '); //A contains 'Abcde Abcde'
B := STD.Str.ToTitleCase('john smith-jones'); //B contains 'John Smith-Jones'
```


ToUpperCase

STD.Str.ToUpperCase(*source*)

STD.Uni.ToUpperCase(*source*)

STD.Uni.LocaleToUpperCase(*source*, *locale*)

<i>source</i>	Uma string que contém os dados cuja formatação de maiúsculas e minúsculas será alterada.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Retorno:	ToUpperCase retorna um valor STRING.

As funções **ToUpperCase** retornam a string *source* com todos os caracteres minúsculos convertidos em maiúsculos.

Exemplo:

```
A := STD.Str.ToUpperCase( 'abcde' );  
//A contains 'ABCDE'
```

Translate

STD.Str.Translate(*source*, *search*, *replacement*)

<i>source</i>	Uma string que contém os caracteres a serem pesquisados.
<i>search</i>	Uma string que contém os caracteres a serem substituídos pelos caracteres na string <i>replacement</i> .
<i>replacement</i>	Uma string que contém os caracteres que serão usados como substituições.
Return:	Translate retorna um valor de STRING.

As funções **Translate** retornam a string *source* com o caractere *replacement* substituído em todos os caracteres da string *source*. string Os caracteres da string *search* são substituídos pelos caracteres na posição equivalente na string *replacement* .

Se nenhum caractere da string *search* estiver na string *source* , será retornada a string *source* sem alterações.

Exemplo:

```
IMPORT STD;  
A := STD.Str.Translate('abcabc', 'ca', 'yz'); //A contains 'zbyzby'
```

Version

STD.Uni.Version()

Retorno:	Version retorna um valor STRING (e.x., '55.1').
----------	---

A função **Version** retorna a versão da biblioteca International Components for Unicode (ICU) instalada.

Exemplo:

```
IMPORT STD;
UniTest:= IF ((INTEGER)std.Uni.Version() < 50, 'Not Supported', (STRING)std.Uni.WordCount(U'我是電腦'));
//Chinese dictionary based iterators added in ICU 50
OUTPUT(UniTest);
```

WildMatch

STD.Str.WildMatch(*source*, *pattern*, *nocase*)

STD.Uni.WildMatch(*source*, *pattern*, *nocase*)

<i>source</i>	Uma string que contém os dados a serem pesquisados.
<i>pattern</i>	Uma string que contém a expressão com curinga a ser correspondida. Os curingas são válidos? (único caractere) e * (vários caracteres).
<i>nocase</i>	Um valor booleano verdadeiro ou falso que indica se é necessário fazer distinção entre maiúsculas e minúsculas.
Return:	WildMatch retorna um valor BOOLEAN.

A função **WildMatch** retorna TRUE se o *padrão* corresponder a *fonte*.

A versão de WildMatch Unicode que não faz distinção entre maiúsculas e minúsculas foi otimizada para priorizar a velocidade em vez da precisão. Para uma operação precisa, você deve usar a função Unicode ToUpperCase explicitamente e, em seguida, a versão de WildMatch Unicode que faz distinção entre maiúsculas e minúsculas, ou usar REGEXFIND.

Exemplo:

```
STD.Str.wildmatch('abcdeabcdec', 'a?c*', false) = TRUE;
```

WordCount

STD.Str.WordCount(*source*)

STD.Uni.WordCount(*source* [, *locale*])

<i>source</i>	Uma string que contém as palavras a serem contadas. As palavras são delimitadas por espaços.
<i>locale</i>	Uma string terminada por nulo que contém o código de idioma e país a ser usado para determinar a ordem correta de classificação e outras operações.
Return:	WordCount retorna um valor inteiro.

A função **WordCount** retorna o número de palavras na string *source* .

Exemplo:

```
IMPORT Std;

str1 := 'a word a day keeps the doctor away';

output(LENGTH(TRIM(Str1,LEFT,RIGHT)) - LENGTH(TRIM(Str1,ALL)) + 1);
//finds eight words by removing spaces

STD.Str.WordCount(str1); //finds eight words based on space delimiter
```

Suporte a Metaphone

Essas funções fornecem uma forma de implementar codificação fonética do Double Metaphone ou do Metaphone 3, ou algoritmos de pesquisa difusa, que retornam um código primário, um código secundário ou os dois códigos para uma string específica.

Primary

STD.Metaphone.Primary(source)

STD.Metaphone3.Primary(source)

<i>source</i>	A string a ser processada.
Return:	Primary retorna um valor de string.

A função **Primary** retorna uma representação textual dos dados da fonte, semelhante a um código Soundex. Essa função retorna o primeiro valor de retorno do algoritmo Double Metaphone.

A função **Metaphone3.Primary** usa as bibliotecas do Metaphone 3 mais recentes, que aprimoram a codificação fonética de palavras em inglês; palavras que não são do idioma inglês, mas são familiares aos americanos; e nomes e sobrenomes encontrados comumente nos Estados Unidos (somente Enterprise Edition).

Exemplo:

```
r := RECORD
  STRING source;
  STRING M1;
  STRING M2;
  STRING Mboth;
END;

r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone.Primary( L.LastName );
  SELF.M2     := STD.Metaphone.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone.Double( L.LastName );
END;

/* Example using Metaphone 3 (available in Enterprise Edition) */
r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone3.Primary( L.LastName );
  SELF.M2     := STD.Metaphone3.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone3.Double( L.LastName );
END;

*/

ds := PROJECT(ProgGuide.Person.File,XF(LEFT));

COUNT(ds);
COUNT(ds(M1 <> M2));
OUTPUT(ds);
OUTPUT(ds(M1 <> M2));
```

Secondary

STD.Metaphone.Secondary(source)

STD.Metaphone3.Secondary(source)

<i>source</i>	A string a ser processada.
Return:	Secondary retorna um valor STRING.

A função **Secondary** retorna uma representação textual dos dados de source, semelhante a um código Soundex. Essa função retorna o segundo valor de retorno do algoritmo Double Metaphone.

A função **Metaphone3.SecondaryPrimary** usa as bibliotecas do Metaphone 3 mais recentes, que aprimoram a codificação fonética de palavras em inglês; palavras que não são do idioma inglês, mas são familiares aos americanos; e nomes e sobrenomes encontrados comumente nos Estados Unidos (somente Enterprise Edition).

Exemplo:

```
r := RECORD
  STRING source;
  STRING M1;
  STRING M2;
  STRING Mboth;
END;

r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone.Primary( L.LastName );
  SELF.M2     := STD.Metaphone.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone.Double( L.LastName );
END;

// Example using Metaphone 3 (available in Enterprise Edition)
/*
r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone3.Primary( L.LastName );
  SELF.M2     := STD.Metaphone3.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone3.Double( L.LastName );
END;
*/

ds := PROJECT(ProgGuide.Person.File,XF(LEFT));

COUNT(ds);
COUNT(ds(M1 <> M2));
OUTPUT(ds);
OUTPUT(ds(M1 <> M2));
```


Double

STD.Metaphone.Double(Source)

STD.Metaphone3.Double(source)

<i>source</i>	A string a ser processada.
Return:	Double retorna um valor STRING

A função **Double** retorna uma representação textual dos dados *source* , semelhante a um código Soundex. Essa função retorna os dois valores de retorno do algoritmo Double Metaphone, concatenados em uma única string de resultado.

A função **Metaphone3.Double** usa as bibliotecas do Metaphone 3 mais recentes, que aprimoram a codificação fonética de palavras em inglês; palavras que não são do idioma inglês, mas são familiares aos americanos; e nomes e sobrenomes encontrados comumente nos Estados Unidos (somente Enterprise Edition).

Exemplo:

```
r := RECORD
  STRING source;
  STRING M1;
  STRING M2;
  STRING Mboth;
END;

r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone.Primary( L.LastName );
  SELF.M2     := STD.Metaphone.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone.Double( L.LastName );
END;

/* Example using Metaphone 3 (available in Enterprise Edition) */
r XF(ProgGuide.Person.File L) := TRANSFORM
  SELF.source := L.LastName;
  SELF.M1     := STD.Metaphone3.Primary( L.LastName );
  SELF.M2     := STD.Metaphone3.Secondary( L.LastName );
  SELF.Mboth  := STD.Metaphone3.Double( L.LastName );
END;
*/

ds := PROJECT(ProgGuide.Person.File,XF(LEFT));

COUNT(ds);
COUNT(ds(M1 <> M2));
OUTPUT(ds);
OUTPUT(ds(M1 <> M2));
```

Suporte a Criptografia

Esta seção fornece suporte para executar funções criptográficas nos dados em ECL.

Visão geral da Biblioteca Criptográfica

Existem três classes de Algoritmos Criptográficos na Biblioteca Criptográfica: Funções de hashing, algoritmos Symmetric-Key e algoritmos Asymmetric-Key.

Funções de Hashing:

- Útil para verificar a integridade dos dados
- Transforma grandes dados de tamanho variáveis em pequenos de tamanho fixo
- Impossível reverter um hash de volta aos dados originais (unidirecional)
- Rápido.

Veja também: SupportedHashAlgorithms

Algoritmos Symmetric-Key:

- Usa uma única chave compartilhada para criptografar/descriptografar dados
- Suporte a algoritmos de bloco
- Rápido.

Veja também: SupportedSymmetricCipherAlgorithms

Algoritmos de Asymmetric-Key (Também conhecido como algoritmos de chave pública ou PKI):

- Par de chaves pública e privada matematicamente associadas
- Usado para criptografar/descriptografar dados
- Usado para criar assinaturas digitais
- Comparativamente mais lento

Veja também: SupportedPublicKeyAlgorithms

SupportedHashAlgorithms

STD.Crypto.SupportedHashAlgorithms();

Retorno:	CONJUNTO DE STRINGs contendo todos os algoritmos de hash suportados
----------	---

A função **SupportedHashAlgorithms** retorna o conjunto de algoritmos de hash suportados

Exemplo:

```
IMPORT STD;  
STD.Crypto.SupportedHashAlgorithms(); //returns SET of STRINGs
```

SupportedSymmetricCipherAlgorithms

STD.Crypto.SupportedSymmetricCipherAlgorithms();

Retorno:	SET OF STRINGs contendo todos os algoritmos Cipher suportados
----------	---

A função **SupportedSymmetricCipherAlgorithms** retorna o conjunto de algoritmos de Cipher suportados

Exemplo:

```
IMPORT STD;  
STD.Crypto.SupportedSymmetricCipherAlgorithms(); //returns SET of STRINGs
```

Algoritmos de Chave Pública Suportados

STD.Crypto.SupportedPublicKeyAlgorithms();

Retorno:	SET OF STRINGs contendo todos os algoritmos de chave pública suportados
----------	---

A função **SupportedPublicKeyAlgorithms** retorna o conjunto de algoritmos de chave pública suportados

Exemplo:

```
IMPORT STD;  
STD.Crypto.SupportedPublicKeyAlgorithms(); //returns SET of STRINGs
```

Módulo Hashing

myHashModule := **STD.Crypto.Hashing**(*hashAlgorithm*);

<i>myHashModule</i>	O nome da estrutura do módulo Hashing
<i>hashAlgorithm</i>	O algoritmo de hashing a ser usado, retornado por SupportedHashAlgorithms()

Um módulo Hashing é definido na ECL. As definições de função subsequentes usam as definições de módulo especificadas na definição do módulo Hashing.

Exemplo:

```
Import STD;

//Hashing module definition
myHashModuleSha512 := Std.Crypto.Hashing('sha512');
myHashModuleSha256 := Std.Crypto.Hashing('sha256');

DATA hash1 := myHashModuleSha512.Hash((DATA)'The quick brown fox jumps over the lazy dog');
DATA hash2 := myHashModuleSha256.Hash((DATA)'The quick brown fox jumps over the lazy dog');

OUTPUT(hash1);
OUTPUT(hash2);
```

Hash

myHashModule.**Hash**(*inputData*);

<i>myHashModule</i>	O nome da estrutura do módulo Hashing
<i>inputData</i>	Os dados para o hash no formato DATA
Retorno:	Conteúdo em hash no formato DATA

A função Hash cria um hash do *inputData* fornecido, usando o algoritmo de hash definido no módulo Hashing.

Exemplo:

```
Import STD;

//Hashing module definition
myHashModuleSha512 := Std.Crypto.Hashing('sha512');
myHashModuleSha256 := Std.Crypto.Hashing('sha256');

DATA hash1 := myHashModuleSha512.Hash((DATA)'The quick brown fox jumps over the lazy dog');
DATA hash2 := myHashModuleSha256.Hash((DATA)'The quick brown fox jumps over the lazy dog');

OUTPUT(hash1);
OUTPUT(hash2);
```


Módulo SymmetricEncryption

mySymEncModule := **STD.Crypto.SymmetricEncryption**(*algorithm*, *passphrase*);

<i>mySymEncModule</i>	O nome da estrutura do módulo Symmetric Encryption
<i>algorithm</i>	O algoritmo a ser usado, retornado por SupportedSymmetricCipherAlgorithms()
<i>passphrase</i>	A senha usada para criptografia/descriptografia

Um módulo Symmetric Encryption é definido na ECL. As definições de função subsequentes usam as opções especificadas na definição do módulo de Symmetric Encryption

Exemplo:

```
IMPORT STD;

//Symmetric Encryption module definition
mySymEncModule := Std.Crypto.SymmetricEncryption('aes-256-cbc',
                                                    '12345678901234567890123456789012');

//encrypt/decrypt string literals
STRING myStr := 'The quick brown fox jumps over the lazy dog';
DATA    encryptedStr := mySymEncModule.Encrypt((DATA)myStr);
STRING  decryptedStr := (STRING)mySymEncModule.Decrypt(encryptedStr);

OUTPUT(myStr);
OUTPUT(decryptedStr);
```

Encrypt (Symmetric)

mySymEncModule.**Encrypt**(*inputData*);

<i>mySymEncModule</i>	O nome da estrutura do módulo Symmetric Encryption
<i>inputData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo criptografado no formato DATA

A função Encrypt criptografa os *inputData* fornecidos, usando as opções definidas no módulo de Symmetric Encryption.

Exemplo:

```
IMPORT STD;

//Symmetric Encryption module definition
mySymEncModule := Std.Crypto.SymmetricEncryption('aes-256-cbc',
                                                    '12345678901234567890123456789012');

//encrypt/decrypt string literals
STRING myStr := 'The quick brown fox jumps over the lazy dog';
DATA    encryptedStr := mySymEncModule.Encrypt((DATA)myStr);
STRING decryptedStr := (STRING)mySymEncModule.Decrypt(encryptedStr);

OUTPUT(myStr);
OUTPUT(decryptedStr);
```

Decrypt (Symmetric)

mySymEncModule.**Decrypt**(*encryptedData*);

<i>mySymEncModule</i>	O nome da estrutura do módulo Symmetric Encryption
<i>encryptedData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo descriptografado no formato DATA

A função Decrypt descriptografa o *encryptedData* fornecido usando as opções definidas no módulo Symmetric Encryption. Você só pode descriptografar dados que foram criptografados pelo método de Criptografia da Biblioteca Padrão.

Exemplo:

```
IMPORT STD;

//Symmetric Encryption module definition
mySymEncModule := Std.Crypto.SymmetricEncryption('aes-256-cbc',
                                                    '12345678901234567890123456789012');

//encrypt/decrypt string literals
STRING myStr := 'The quick brown fox jumps over the lazy dog';
DATA encryptedStr := mySymEncModule.Encrypt((DATA)myStr);
STRING decryptedStr := (STRING)mySymEncModule.Decrypt(encryptedStr);

OUTPUT(myStr);
OUTPUT(decryptedStr);
```

Módulo PublicKeyEncryption

myPKEModule := **STD.Crypto.PublicKeyEncryption**(*pkAlgorithm*, *publicKeyFile*, *privateKeyFile*, *passphrase*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>pkAlgorithm</i>	O algoritmo a ser usado, retornado por SupportedPublicKeyAlgorithms()
<i>publicKeyFile</i>	A especificação de arquivo do arquivo de chave pública formatada pelo PEM
<i>privateKeyFile</i>	A especificação de arquivo do arquivo de chave privada formatada pelo PEM
<i>passphrase</i>	A senha usada para criptografia, descriptografia, assinatura, verificação

Um módulo de Chave Pública Criptografia é definido na ECL. As definições de função subsequentes usam as opções definidas no módulo Public Key Encryption para executar criptografia assimétrica/descriptografia/verificação de assinatura/assinatura digital

Exemplo:

```
IMPORT STD;
privKeyFile := '/var/lib/HPCCSystems/myesp/test.key';
pubKeyFile := '/var/lib/HPCCSystems/myesp/test.key.pub';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryption('RSA', pubKeyFile, privKeyFile, '');

DATA encrypted := myPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');

OUTPUT( (STRING)myPKEModule.Decrypt(encrypted) );
```

Encrypt (PKE)

myPKEModule.**Encrypt**(*inputData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>inputData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo criptografado no formato DATA

A função Criptografar criptografa o *inputData* fornecidos usando as opções especificadas na definição do módulo Public Key Encryption.

Exemplo:

```
IMPORT STD;
privKeyFile := '/var/lib/HPCCSystems/myesp/test.key';
pubKeyFile  := '/var/lib/HPCCSystems/myesp/test.key.pub';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryption('RSA', pubKeyFile, privKeyFile, '');

DATA encrypted := myPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');

OUTPUT( (STRING)myPKEModule.Decrypt(encrypted) );
```

Decrypt (PKE)

myPKEModule.**Decrypt**(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>encryptedData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo descriptografado no formato DATA

A função Decrypt descriptografa o *encryptedData* fornecidos, usando as opções especificadas na definição do módulo Public Key Encryption. Você só pode descriptografar dados que foram criptografados pelo método de Criptografia da Biblioteca Padrão.

Exemplo:

```
IMPORT STD;
privKeyFile := '/var/lib/HPCCSystems/myesp/test.key';
pubKeyFile  := '/var/lib/HPCCSystems/myesp/test.key.pub';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryption('RSA', pubKeyFile, privKeyFile, '');

DATA encrypted := myPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');

OUTPUT( (STRING)myPKEModule.Decrypt(encrypted) );
```

Sign (PKE)

mySymEncModule.**Sign**(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>inputData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Assinatura digital computada

A função Sign cria uma assinatura digital do *inputData* fornecido, usando as opções especificadas na definição do módulo Public Key Encryption.

Exemplo:

```
IMPORT STD;
privKeyFile := '/var/lib/HPCCSystems/myesp/test.key';
pubKeyFile  := '/var/lib/HPCCSystems/myesp/test.key.pub';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryption('RSA', pubKeyFile, privKeyFile, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```

VerifySignature (PKE)

myPKEModule.VerifySignature(*signature*, *signedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>signature</i>	A assinatura digital para verificar
<i>signedData</i>	Dados usados para criar a assinatura no formato DATA
Retorno:	Um valor BOOLEAN para indicar a verificação

A função VerifySignature verifica a *assinatura* digital fornecida usando as opções especificadas na definição do módulo Public Key Encryption.

Exemplo:

```
IMPORT STD;
privKeyFile := '/var/lib/HPCCSystems/myesp/test.key';
pubKeyFile := '/var/lib/HPCCSystems/myesp/test.key.pub';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryption('RSA', pubKeyFile, privKeyFile, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```


Módulo PublicKeyEncryptionFrom-Buffer

myPKEModule := **STD.Crypto.PublicKeyEncryptionFromBuffer**(*pkAlgorithm*, *publicKeyFile*, *privateKeyFile*, *passphrase*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From Buffer
<i>pkAlgorithm</i>	O algoritmo a ser usado, retornado por SupportedPublicKeyAlgorithms()
<i>publicKeyBuff</i>	Buffer de chave pública em formato PEM
<i>privateKeyBuff</i>	Buffer de chave privada em formato PEM
<i>passphrase</i>	A senha usada para criptografia, descriptografia, assinatura, verificação

Um módulo de Public Key Encryption é definido na ECL. As definições de funções subsequentes usam as opções definidas no módulo Public Key Encryption From Buffer para executar a criptografia assimétrica/descriptografia/assinatura digital/verificação de assinatura.

Exemplo:

```
IMPORT STD;

STRING publicKey := '-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftrESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHqLMdydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEIvhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NysWzE/2/ZOWxZdr79XC+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----';

STRING privateKey := '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEA64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftrESmzQ6I/TiUQcmi42soUXmCeEBHqLMdydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEIvhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NysWzE/2/ZOWxZdr79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0bwIDAQABAoIBAQCnGATNYkOOu8wW' + '\n' +
'F5oid3aKwnPytF211WQH3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCpZorcxY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXRpbMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
'70pztRWNOrCzrC+ARlmmDfYUgVftZin53jq606ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAOGBAOKtaVFa6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBef3ds9Fq' + '\n' +
'f0jhhvuV0OcN81YbR/Z1YRJDUs6mHh/2BYSkdeaLkojXTxKR2ba4xQk5dtJCdOPf' + '\n' +
'0c15AlTgOyK2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADbnRSdzzOdo0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCzO1VcEeZtsTHuL04t7NmdHGqNXTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tw67KtYxlr28+CcQoUaQ/Au5kzjE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkd1cNjranHGUrkszbwSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDYIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3LHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuEz+N8rp17VUSeorjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qri/FcSLZ04JLsFrmTPU/Y5y8/dHjY06fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOkgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqlJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQZE/IdkQqifmBlyEjjRHQ81WS+K5NnjNlt0IEscJqOAwv9s' + '\n' +
'iThG5ueb6xoj/N0LuXa8l0UT5aChKWxRHEydegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----';
```

```
//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromBuffer('RSA', PublicKey, PrivateKey, '');

DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```

Encrypt (PKE From Buffer)

myPKEModule.Encrypt(*inputData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>inputData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo criptografado no formato DATA

A função Criptografar criptografa o *inputData* fornecido, usando as opções especificadas na definição do módulo Public Key Encryption From Buffer.

Exemplo:

```
IMPORT STD;

STRING publicKey := '-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAR64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHqLMDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEIvhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NysWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----';

STRING privateKey := '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEAR64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHqLMDydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEIvhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NysWzE/2/ZOWxZdR79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0bwIDAQABaoIBAQCnGatNYkOOu8wW' + '\n' +
'F5oid3aKwnwPytF211WQH3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCPzorcXY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
'70pztRWNorCzrC+ARlmmDfY0UgVFtZin53jq606ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAoGBAOKtaVfa6z2F4Q+koMBXCt4m7dCJNaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhhvuV00cn81YbR/Z1YJRJDUs6mHh/2BYSkdeaLKoJXtXKR2ba4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNRSdzzOdo0JEj4VbNZEtX6nQhB0OortYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCz01VcEeZtsTHuL04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kzE9/4DjXrT09QmVAMciNenc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdIcNJranHGUrkswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rp17VUSeoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qRI/FcSLZ04JLsfRmTPU/Y5y8/dHjY06fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOkgV4YNyphk1UYWHNz8Yy5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnJNlt0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8lOUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----';

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromBuffer('RSA', PublicKey, PrivateKey, '');

DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```

Decrypt (PKE From Buffer)

myPKEModule.Decrypt(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>encryptedData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Conteúdo descriptografado no formato DATA

A função Decrypt descriptografa o *encryptedData* fornecidos, usando as opções especificadas na definição do módulo Public Key Encryption From Buffer. Você só pode descriptografar dados que foram criptografados pelo método de Criptografia da Biblioteca Padrão.

Exemplo:

```
IMPORT STD;

STRING publicKey := '-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEivhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NysWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----';

STRING privateKey := '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEA64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEivhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NysWzE/2/ZOWxZdR79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0bwIDAQABAOIBAQCnGAtNYkOOu8wW' + '\n' +
'F5oid3akwnwPytF211WQh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCPzorcXY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
'70pztrWNORCzrC+ARlmmDfY0UgVftZin53jq606ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAOGBAOKtaVfa6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhhvuV00cNB1YbR/Z1YrJDUs6mHh/2BYSkdeaLKoJkTxKR2bA4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOyK2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADbnRSdzzOdo0JEj4VbNZEtX6nQhB00rtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCz01VcEeZtsTHuL04t7NmdHGqNXTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kjjzE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdlnJranHGUrkszbwSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOX1HDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLM1R' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rpl7VUSeoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qRI/FcSLZ04JLsfRmTPU/Y5y8/dHjY06fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOkgV4YNyphk1UYWHNz8Yy5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnNlt0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8l0UT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----';

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromBuffer('RSA', PublicKey, PrivateKey, '');

DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```

Sign (PKE (PKE From Buffer))

mySymEncModule.Sign(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>inputData</i>	Os dados a serem criptografados no formato DATA
Retorno:	Dados usados para criar a assinatura no formato DATA

A função Sign cria uma assinatura digital do *inputData* fornecido, usando as opções especificadas na definição do módulo Public Key Encryption From Buffer.

Exemplo:

```
IMPORT STD;

STRING publicKey := '-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAR64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKfRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHqLMDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEivhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NysWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----';

STRING privateKey := '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEAR64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKfRESmzQ6I/TiUQcmi42soUXmCeEBHqLMDydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEivhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NysWzE/2/ZOWxZdR79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSzwPMnVuvU0bwIDAQABaoIBAQCnGatNYkOOu8wW' + '\n' +
'F5oid3akwnwPytF211WQH3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCpZorcXY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
'70pztRWNorCzrC+ARlmmDfY0UgVftZin53jq606ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAoGBAOKtaVfa6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhhvuU0OcN81YbR/Z1YJRJDUs6mHh/2BYSkdeaLKoJkTxKR2bA4xQk5dtJCdPf' + '\n' +
'0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNRSdzzOdo0JEj4VbNZEtz6nQhB0OortYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCz01VcEeZtsTHuL04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kzE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdlnJranHGUrkswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rp17VUSeORjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qRI/FcSLZ04JLsfRmTPU/Y5y8/dHjY06fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOkgV4YNyphk1UYWHNz8Yy5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnJNlt0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8lOUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryptionFromBuffer('RSA', publicKey, privateKey, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```

VerifySignature (PKE From Buffer)

myPKEModule.VerifySignature(signature, signedData);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption
<i>signature</i>	A assinatura digital para verificar
<i>signedData</i>	Dados usados para criar a assinatura no formato DATA
Retorno:	Um valor BOOLEAN para indicar a verificação

A função `VerifySignature` verifica a *assinatura* digital fornecida usando as opções especificadas na definição do módulo Public Key Encryption From Buffer.

Exemplo:

```
IMPORT STD;

STRING publicKey := '-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEivhLiGfcm3bpTzWQ66zVz3z/huLbEXEY5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NySWzE/2/ZOWxZdr79XC+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----';

STRING privateKey := '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEA64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEivhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEY5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdr79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
'F5Oid3aKwnwPytF211Wqh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCpZorCXY5zD0mMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
'70pztrWNORCzrC+ARlmmDfYOUgVftZin53jq6O6ullPLzhkm3/+QFRGYwsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAOGBAOKtaVF6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhvhvU0OcN81YbR/ZLYRJdUS6mHh/2BYSKdeaLKOjXtXKR2ba4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNrSdzsOdo0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCzO1VcEeZtsTHuL04t7NmdHGqNxtV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kzE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdLcNJranHGURkzswbXSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOX1hDYIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLM1R' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rpl7VUSeoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qRI/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnJlnt0IEscJqQAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8lOUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----';

//PKE Encryption module definition
myPKEModule := STD.Crypto.PublicKeyEncryptionFromBuffer('RSA', publicKey, privateKey, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```


PublicKeyEncryptionFromLFN Module

myPKEModule := **STD.Crypto.PublicKeyEncryptionFromLFN**(*pkAlgorithm*, *publicKeyFile*, *privateKeyFile*, *passphrase*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From LFN (Logical FileName)
<i>pkAlgorithm</i>	O algoritmo a ser usado, retornado por SupportedPublicKeyAlgorithms()
<i>publicKeyLFN</i>	Arquivo lógico de chave pública formatado PEM
<i>privateKeyLFN</i>	Arquivo lógico de chave pública formatado PEM
<i>passphrase</i>	A senha usada para criptografia, descritografia, assinatura, verificação.

Uma Public Key Encryption From LFN é definida na ECL. As definições de funções subsequentes usam as opções definidas no módulo Public Key Encryption from LFN para executar a verificação assimétrica de criptografia/descriptografia/assinatura digital/assinatura.

Exemplo:

```
IMPORT Std;

PublicKeyFile := '~Examples::certificates::public::pubkey.pem';
PrivateKeyFile := '~Examples::certificates::private::privkey.pem';
//You can restrict access using file scope security
//on the ~Examples::certificates::private scope

pubKey := RECORD
    STRING Key;
END;

dPubKey := DATASET([
    '-----BEGIN PUBLIC KEY-----' + '\n' +
    'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr64RncTp5pV0KMnWRAof' + '\n' +
    'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
    'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
    'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
    '/oKj6q7kInEIvhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
    'J6Tk4NY3NySWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0' + '\n' +
    'bwIDAQAB' + '\n' +
    '-----END PUBLIC KEY-----' + '\n'
], pubKey);

OUTPUT(dPubKey, PublicKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

PrivKey := RECORD
    STRING Key;
END;

dPrivKey := DATASET([
    '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
    'MIIEowIBAAKCAQEAr64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
    '4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
    '8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
    'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEIvhLiGfcm3bpTzWQ6' + '\n' +
    '6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdR79XC' + '\n' +
    '+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
    'F5Oid3aKwnwPytF211WQh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
    'VlyPrdYVZInFjYIJCPzorCXY5zDOMMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
    'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBmN61uMHCxh2tDb2yvl8Zz+El1ADG' + '\n' +
    '70pztRWNoRcZrC+ARlmmDfYOUGVftZin53jq6O6u1PLzhkm3/+QFRGYWsFgQB6J' + '\n' +
    'Z9HJtW5YB47RT5RbLHKXeMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
    'sB0cNeURAoGBAOKtaVfa6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +

```

```
'f0jhhvuV00cN8lYbR/ZlYRJDUs6mHh/2BYSkdeaLKojXTxKR2ba4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNRSdzZ0do0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
'eCzO1VcEeZtsTHuL04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kjkzE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdLcNJranHGUrkszbwSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rpl7VUSeoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qri/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnjNlt0IEscJqOAwv9s' + '\n' +
'iihG5ueb6xoj/N0LuXa8loUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----' + '\n'
}]],PrivKey);

OUTPUT(dPrivKey,,PrivateKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromLFN('RSA', PublicKeyFile, PrivateKeyFile, '');

DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```


Encrypt (PKE From LFN)

myPKEModule.Encrypt(*inputData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From LFN (Logical FileName)
<i>inputData</i>	Os dados para criptografar no formato DATA
Return:	Conteúdo criptografado no formato DATA

A função Encrypt criptografa o *inputData* fornecido, usando as opções especificadas na definição do módulo Public Key Encryption From LFN.

Exemplo:

```
IMPORT Std;

PublicKeyFile := '~Examples::certificates::public::pubkey.pem';
PrivateKeyFile := '~Examples::certificates::private::privkey.pem';
//You can restrict access using file scope security
//on the ~Examples::certificates::private scope

pubKey := RECORD
    STRING Key;
END;

dPubKey := DATASET([
    '-----BEGIN PUBLIC KEY-----' + '\n' +
    'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr64RncTp5pV0KMnWRAof' + '\n' +
    'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
    'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE30cRzH0eBkOryW6X3efWnMoy' + '\n' +
    'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
    '/oKj6q7kInEivhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
    'J6Tk4NY3NySWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0' + '\n' +
    'bwIDAQAB' + '\n' +
    '-----END PUBLIC KEY-----' + '\n'
], pubKey);

OUTPUT(dPubKey, PublicKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

PrivKey := RECORD
    STRING Key;
END;

dPrivKey := DATASET([
    '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
    'MIIEowIBAAKCAQEAr64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
    '4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
    '8iZo7RC7ocQE30cRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
    'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEivhLiGfcm3bpTzWQ6' + '\n' +
    '6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdR79XC' + '\n' +
    '+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
    'F50id3aKwnwPytF211WQh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
    'VlyPrdYVZInFjYIJCpZorCXY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
    'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMM61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
    '70pztrWNOrCzrC+ARlmmDfYOUgVftZin53jq6O6ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
    'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
    'sB0cNeURAoGBAOKtaVFa6z2F4Q+koMBXCt4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
    'f0jhhvuV0OcN81YbR/Z1YRJDUs6mHh/2BYSkdeaLKOjXTxKR2bA4xQk5dtJCdPf' + '\n' +
    '0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
    '1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
    'YEADBNRSdzZ0do0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
    'eCzO1VcEeZtsThU04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpre5D' + '\n' +
    '-----END RSA PRIVATE KEY-----'
], dPrivKey);
```

```
'tW67KtYxlr28+CcQoUaQ/Au5kjlzE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +  
'wZP/bPZMVYKtbsaVkdIcNJranHGUrKzswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +  
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +  
'9mdAjMkszdLTg5uuE+z+N8rpl7VUseoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +  
'spmh9MgBh0JbsbWazK0s9/qrI/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +  
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +  
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnjNlt0IEscJqOAwv9s' + '\n' +  
'iIhG5ueb6xoj/N0LuXa8loUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +  
'-----END RSA PRIVATE KEY-----' + '\n'  
}],PrivKey);  
  
OUTPUT(dPrivKey,,PrivateKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);  
  
//PKE Encryption module definition  
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromLFN('RSA', PublicKeyFile, PrivateKeyFile, '');  
  
DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');  
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```

Decrypt (PKE From LFN)

myPKEModule.Decrypt(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From LFN (Logical FileName)
<i>encryptedData</i>	Os dados para descriptografar no formato DATA
Return:	Conteúdo descriptografado no formato DATA

A função Decrypt descriptografa o *encryptedData* fornecidos, usando as opções especificadas na definição do módulo Public Key Encryption From LFN. Você só pode descriptografar dados que foram criptografados pelo método Criptografar da Biblioteca Padrão.

Exemplo:

```
IMPORT Std;

PublicKeyFile := '~Examples::certificates::public::pubkey.pem';
PrivateKeyFile:= '~Examples::certificates::private::privkey.pem';
//You can restrict access using file scope security
//on the ~Examples::certificates::private scope

pubKey := RECORD
    STRING Key;
END;

dPubKey := DATASET([{
'-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCGKCAQEAr64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEIvhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NySWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----' + '\n'
}],pubKey);

OUTPUT(dPubKey,,PublicKeyFile, CSV(SEPARATOR(' '), TERMINATOR(' ')), OVERWRITE);

PrivKey := RECORD
    STRING Key;
END;

dPrivKey := DATASET([{
'-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEAr64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEIvhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdR79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
'F5Oid3aKwnwPytF211Wqh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCPzorcXY5zD0mMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXrPBMN61uMHCxh2tDb2yv18Zz+El1ADG' + '\n' +
'70pztrWNOrCzrC+ARlmmDfYOUgVFtZin53jq6O6ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXEmc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAOGBAOKtaVF6z2F4Q+koMBXCt4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhvhvU0OcN81YbR/Z1YRJDU56mHh/2BYSKdeaLKOjXTxKR2ba4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOYk2onXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNrSdzZodo0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
}
```

```
'eCz01VcEeZtsTHuL04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kzjE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdLcNJranHGUrkswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rpl7VUseoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qrI/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnJN1t0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8loUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----' + '\n'
}},PrivKey);

OUTPUT(dPrivKey,,PrivateKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromLFN('RSA', PublicKeyFile, PrivateKeyFile, '');

DATA encrypted := MyPKEModule.Encrypt((DATA)'The quick brown fox jumps over the lazy dog');
OUTPUT( (STRING)MyPKEModule.Decrypt(encrypted));
```

Sign (PKE From LFN)

mySymEncModule.**Sign**(*encryptedData*);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From LFN (Logical FileName)
<i>inputData</i>	Os dados para assinar no formato DATA
Return:	Assinatura digital computada no formato DATA

A função **Sign** cria uma assinatura digital do *inputData* fornecido, usando as opções especificadas na definição do módulo Public Key Encryption From LFN.

Exemplo:

```
IMPORT Std;

PublicKeyFile := '~Examples::certificates::public::pubkey.pem';
PrivateKeyFile := '~Examples::certificates::private::privkey.pem';
//You can restrict access using file scope security
//on the ~Examples::certificates::private scope

pubKey := RECORD
    STRING Key;
END;

dPubKey := DATASET([
    '-----BEGIN PUBLIC KEY-----' + '\n' +
    'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBcKCAQEAr64RncTp5pV0KMnWRAof' + '\n' +
    'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
    'BHq1MDydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE30cRzH0eBkOryW6X3efWnMoy' + '\n' +
    'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
    '/oKj6q7kInEivhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
    'J6Tk4NY3NySWzE/2/ZOWxZdR79XC+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0' + '\n' +
    'bwIDAQAB' + '\n' +
    '-----END PUBLIC KEY-----' + '\n'
], pubKey);

OUTPUT(dPubKey, PublicKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

PrivKey := RECORD
    STRING Key;
END;

dPrivKey := DATASET([
    '-----BEGIN RSA PRIVATE KEY-----' + '\n' +
    'MIIEowIBAAKCAQEAr64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
    '4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHq1MDydw9aHOQG17CB30GYsw3Lf' + '\n' +
    '8iZo7RC7ocQE30cRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
    'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEivhLiGfcm3bpTzWQ6' + '\n' +
    '6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdR79XC' + '\n' +
    '+goNL6v/5gPI8B/a3Z8OeM2PfsZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
    'F50id3aKwnwPytF211WQh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
    'VlyPrdYVZInFjYIJCpZorcxY5zDomMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
    'GCK6U18mR9XELBFQ6keeTo2yDu0TQ4oBXRpBMN61uMHCxh2tDb2yvl8Zz+EllADG' + '\n' +
    '70pztrWNORCzrC+ARlmmDfY0UgVftZin53jq6O6ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
    'Z9HJtW5YB47RT5RbLHKXMc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
    'sB0cNeURAoGBAOKtaVF6z2F4Q+koMBXCt4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
    'f0jhhvuV0OcN81YbR/Z1YRJDUs6mHh/2BYSkdeaLKOjXTxKR2bA4xQk5dtJCdPf' + '\n' +
    '0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
    '1wRUXS1dbqemoc+g48wj5r3/qsIG8PsZ2Y8W+oYw7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
    'YEADBNRSdzZ0do0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +
    'eCzO1VcEeZtsThU04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYsPre5D' + '\n' +
    '-----END RSA PRIVATE KEY-----' + '\n'
], dPrivKey);
```

```
'tW67KtYxlr28+CcQoUaQ/Au5kτζE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdIcNJranHGUrKzswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rp17VUseoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qri/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ5lQ+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ8lWS+K5NnjNlt0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8loUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----' + '\n'
}],PrivKey);

OUTPUT(dPrivKey,,PrivateKeyFile, CSV(SEPARATOR(','), TERMINATOR(')'), OVERWRITE);

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromLFN('RSA', PublicKeyFile, PrivateKeyFile, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```

VerifySignature (PKE From LFN)

myPKEModule.VerifySignature(signature, signedData);

<i>myPKEModule</i>	O nome da estrutura do módulo Public Key Encryption From LFN (Logical FileName)
<i>signature</i>	A assinatura digital a ser verificada
<i>signedData</i>	Dados usados para criar a assinatura no formato DATA
Return:	Um valor BOOLEAN para indicar verificação

A função `VerifySignature` verifica a *assinatura* digital fornecida usando as opções especificadas na definição do módulo Public Key Encryption From LFN.

Exemplo:

```
IMPORT Std;

PublicKeyFile := '~Examples::certificates::public::pubkey.pem';
PrivateKeyFile:= '~Examples::certificates::private::privkey.pem';
//You can restrict access using file scope security
//on the ~Examples::certificates::private scope

pubKey := RECORD
    STRING Key;
END;

dPubKey := DATASET([ {
'-----BEGIN PUBLIC KEY-----' + '\n' +
'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAr64RncTp5pV0KMnWRAof' + '\n' +
'od+3AUS/IDngT39j3Iovv9aI2N8g4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeE' + '\n' +
'BHqLMdydw9aHOQG17CB30GYsw3Lf8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoy' + '\n' +
'hIR9MexCldF+3WM/X0IX0ApSs7kuVPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv' + '\n' +
'/oKj6q7kInEIvhLiGfcm3bpTzWQ66zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3' + '\n' +
'J6Tk4NY3NySWzE/2/ZOWxZdr79XC+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0' + '\n' +
'bwIDAQAB' + '\n' +
'-----END PUBLIC KEY-----' + '\n'
}, pubKey);

OUTPUT(dPubKey,,PublicKeyFile, CSV(SEPARATOR(' '), TERMINATOR(' ')), OVERWRITE);

PrivKey := RECORD
    STRING Key;
END;

dPrivKey := DATASET([ {
'-----BEGIN RSA PRIVATE KEY-----' + '\n' +
'MIIEowIBAAKCAQEAr64RncTp5pV0KMnWRAofod+3AUS/IDngT39j3Iovv9aI2N8g' + '\n' +
'4W5ipqhKftRESmzQ6I/TiUQcmi42soUXmCeEBHqLMdydw9aHOQG17CB30GYsw3Lf' + '\n' +
'8iZo7RC7ocQE3OcRzH0eBkOryW6X3efWnMoyhIR9MexCldF+3WM/X0IX0ApSs7ku' + '\n' +
'VPVG4Yj202+1FVO/XNwjMukJG5ASuxpYAQvv/oKj6q7kInEIvhLiGfcm3bpTzWQ6' + '\n' +
'6zVz3z/huLbEXEy5oj2fQaC5E3s5mdpk/CW3J6Tk4NY3NySWzE/2/ZOWxZdr79XC' + '\n' +
'+goNL6v/5gPI8B/a3Z8OeM2PfSZwPMnVuvU0bwIDAQABAoIBAQCnGAtNYkOOu8wW' + '\n' +
'F5Oid3aKwnwPytF211Wqh3v2AcFU17qle+SMRi+ykBL6+u5RU5qH+HSc9Jm31AjW' + '\n' +
'VlyPrdYVZInFjYIJCPzorcXY5zD0mMAuzg5PBVV7VhUA0a5GZck6FC8AilDUcEom' + '\n' +
'GCK6U18mr9XELBFQ6keeTo2yDu0TQ4oBXrPBMN61uMHCxh2tDb2yvl8Zz+El1ADG' + '\n' +
'70pztrWNOrCzrC+ARlmmDfYOUgVFtZin53jq6O6ullPLzhkm3/+QFRGYWsFgQB6J' + '\n' +
'Z9HJtW5YB47RT5RbLHKXEmc6IJW+d+5HrzgTdK79P7wAZk8JCIDyHe2AaNAUzc/G' + '\n' +
'sB0cNeURAOGBAOKtaVF6z2F4Q+koMBXct4m7dCJnaC+qthF249uEOIBeF3ds9Fq' + '\n' +
'f0jhvhvU0OcN8lYbR/ZlYRJdUs6mHh/2BYSKdeaLkojXTxKR2ba4xQk5dtJCdoPf' + '\n' +
'0c15AlTgOYk2oNXP/azDICJYT/cdvIdUL9P4IoZthulFjwG266GacEnNAoGBAMZn' + '\n' +
'1wRUXS1dbqemcc+g48wJ5r3/qsIG8PsZ2Y8W+oYW7diNA5o6acc8YPEWE2RbJDbX' + '\n' +
'YEADBNrSdzZodo0JEj4VbNZEtX6nQhBOOrtYKnnqHVI/XOz3VVu6kedUKdBR87KC' + '\n' +

```

```
'eCz01VcEeZtsTHuL04t7NmdHGqNxTV+jLvzBoQsrAoGAI+fOD+nz6znirYSpRe5D' + '\n' +
'tW67KtYxlr28+CcQoUaQ/Au5kzjE9/4DjXrT09QmVAMciNEnc/sZBjiNzFf525wv' + '\n' +
'wZP/bPZMVYKtbsaVkdLcNJranHGUrkswbxSRzmBQ5/YmCWrdAuYcnhEqmMWcuU9' + '\n' +
'8jiS13JP9hOXlHDyIBYDhV0CgYBV6TznuQgnzp9NpQ/H8ijxilitz3lHTu4mLMlR' + '\n' +
'9mdAjMkszdLTg5uuE+z+N8rpl7VUseoRjb3LvLG4+MXIyDbH/0sDdPm+IjqvCNDR' + '\n' +
'spmh9MgBh0JbsbWazK0s9/qrI/FcSLZ04JLsfRmTPU/Y5y8/dHjYO6fDQhp44RZF' + '\n' +
'iCqNxQKBgHf7KZIOKgV4YNyphk1UYWHNz8YY5o7WtaQ51Q+kIbU8PRd9rqJLZyk2' + '\n' +
'tKf8e6z+wtKjxi8GKQzE/IdkQqiFmBlyEjjRHQ81WS+K5NnJN1t0IEscJqOAwv9s' + '\n' +
'iIhG5ueb6xoj/N0LuXa8loUT5aChKWxRHEYdegqU48f+qxUcJj9R' + '\n' +
'-----END RSA PRIVATE KEY-----' + '\n'
}],PrivKey);

OUTPUT(dPrivKey,,PrivateKeyFile, CSV(SEPARATOR(','), TERMINATOR('\n')), OVERWRITE);

//PKE Encryption module definition
MyPKEModule := STD.Crypto.PublicKeyEncryptionFromLFN('RSA', PublicKeyFile, PrivateKeyFile, '');

DATA signature := myPKEModule.Sign((DATA)'The quick brown fox jumps');
OUTPUT(TRUE = myPKEModule.VerifySignature(signature, (DATA)'The quick brown fox jumps'));
```


Manipulação de Data e Hora

Date Data Types

STD.Date.Date_rec

STD.Date.Date_t

STD.Date.Days_t

Date_rec	Uma estrutura RECORD contendo três campos e INTEGER2 ano, um mês UNSIGNED1 e um dia UNSIGNED1.
Date_t	(Um UNSIGNED4 que contém um valor de data no formato YYYYMMDD.)
Days_t	Um UNSIGNED4 que contém um valor de dados que representa o número de dias decorridos desde uma determinada data base. Esse número pode ser o número de dias da era comum (1 de janeiro de 1 AD = 1) com base no calendário juliano ou gregoriano, ou o número de dias decorridos desde 1 de janeiro de 1900 (1 de janeiro de 1900 = 1) do calendário gregoriano.

Os três tipos de dados de data definidos na biblioteca de padrão de dados são:

```
// A record structure with the different elements separated out.
EXPORT Date_rec := RECORD
  INTEGER2   year;
  UNSIGNED1  month;
  UNSIGNED1  day;
END;

//An unsigned number holding a date in the decimal form YYYYMMDD.
//This type does not support dates prior to 1AD
EXPORT Date_t := UNSIGNED4;

//A number of elapsed days. Value depends on the function called.
EXPORT Days_t := UNSIGNED4;
```

Ver também: Tipos de dados de hora

Time Data Types

STD.Date.Time_rec

STD.Date.Time_t

STD.DateTime_rec

STD.Timestamp_t

Time_rec	Uma estrutura RECORD contendo três campos, um hora INTEGER1 hora, um minute UNSIGNED1 e um segundo UNSIGNED1.
Time_t	Um UNSIGNED3 que mantém a hora do dia no formato decimal HHMMDD.
Seconds_t	Um INTEGER8 que mantém um número de segundos. Pode ser usado para representar uma duração ou o número de segundos desde o início da contagem (1 de janeiro de 1970).
DateTime_rec	Uma estrutura RECORD que contém um Date_rec e um Time_rec
Timestamp_t	Um INTEGER8 que mantém um número de microssegundos. Can be used to represent // either a duration or the number of microseconds since epoch (Jan 1, 1970).

Os três tipos de dados de hora definidos na biblioteca de padrão de dados são:

```
// A record structure with the different time elements separated out.
EXPORT Time_rec := RECORD
    UNSIGNED1    hour;
    UNSIGNED1    minute;
    UNSIGNED1    second;
END;

// An unsigned number holding a time of day in the decimal form HHMMDD.
EXPORT Time_t := UNSIGNED3;
// A signed number holding a number of seconds. Can be used to represent either
// a duration or the number of seconds since epoch (Jan 1, 1970).
EXPORT Seconds_t := INTEGER8;

// A record structure with the different date and time elements separated out.
EXPORT DateTime_rec := RECORD
    Date_rec;
    Time_Rec;
END;

// A signed number holding a number of microseconds. Can be used to represent
// either a duration or the number of microseconds since epoch (Jan 1, 1970).
EXPORT Timestamp_t := INTEGER8;
```

Ver também: Tipos de Dados Data

Year

STD.Date.Year(*date*)

<i>date</i>	Um valor de data no formato Date_t.
Return:	Year retorna um valor INTEGER.

A função **Year** retorna o número do ano a partir do valor de *date* .

Exemplo:

```
IMPORT STD;
UNSIGNED4 MyDate := 20120101;    //January 1, 2012

Y := STD.Date.Year(MyDate);
    //Y contains 2012
```

Month

STD.Date.Month(*date*)

<i>date</i>	Um valor de data no formato Date_t.
Return:	Month retorna um valor INTEGER no intervalo de 1 a 12.

A função **Month** retorna o número do mês do valor *date* .

Exemplo:

```
IMPORT STD;
UNSIGNED4 MyDate := 20120101;    //January 1, 2012

M := STD.Date.Month(MyDate);
    //M contains 1, representing January
```

Day

STD.Date.Day(*date*)

<i>date</i>	Um valor de data no formato Date_t.
Return:	Day retorna um valor INTEGER no intervalo de 1 a 31.

A função **Day** retorna o número do dia do valor *date* .

Exemplo:

```
IMPORT STD;
UNSIGNED4 MyDate := 20120101;    //January 1, 2012

D := STD.Date.Day(MyDate);
    //D contains 1, representing the first of the month
```

Hour

STD.Date.Hour(*time*)

<i>time</i>	Um valor de hora no formato Time_.
Return:	Hour retorna um valor INTEGER que representa as horas no intervalo de 0 a 23.

A função **Hour** retorna as horas do valor de *time* .

Exemplo:

```
IMPORT STD;
MyTime:= STD.Date.CurrentTime(TRUE);    //Local Time

t1 := STD.Date.Hour(MyTime);
    //t1 contains the hour of the current local time
```

Minute

STD.Date.Minute(*time*)

<i>time</i>	Um valor de hora no formato Time_.
Return:	Minute retorna um valor INTEGER que representa os minutos no intervalo de 0 a 59.

A função **Minute** retorna os minutos do valor *time* .

Exemplo:

```
IMPORT STD;
MyTime:= STD.Date.CurrentTime(TRUE);    //Local Time

t1 := STD.Date.Minute(MyTime);
    //t1 contains the minute of the current local time
```


Second

STD.Date.Second(*time*)

<i>time</i>	Um valor de hora no formato Time_.
Return:	Second retorna um valor INTEGER que representa os segundos no intervalo de 0 a 59.

A função **Second** retorna os segundos do valor de *time* .

Exemplo:

```
IMPORT STD;
MyTime:= STD.Date.CurrentTime(TRUE);    //Local Time

t1 := STD.Date.Second(MyTime);
    //t1 contains the second of the current local time
```

DateFromParts

STD.Date.DateFromParts(*year*, *month*, *day*)

<i>year</i>	um valor de ano INTEGER2 no intervalo 0 a 9999.
<i>month</i>	Um valor de mês UNSIGNED1 no intervalo de 1 a 12.
<i>day</i>	Um valor de dia UNSIGNED1 no intervalo de 1 a 31.
Return:	DateFromParts retorna um valor UNSIGNED4.

A função **DateFromParts** retorna um valor Date_t dos parâmetros *year*, *month* e *day* .

Exemplo:

```
IMPORT STD;
INTEGER2 MyYear := 2012;
UNSIGNED1 MyMonth := 1;
UNSIGNED1 MyDay := 1;

D := STD.Date.DateFromParts(MyYear, MyMonth, MyDay);
//D contains 20120101, representing January 1, 2012
```

TimeFromParts

STD.Date.TimeFromParts(*hour*, *minute*, *second*)

<i>hour</i>	Um valor de horas INTEGER1 no intervalo de 0 a 23.
<i>minute</i>	Um valor de minutos UNSIGNED1 no intervalo de 0 a 59.
<i>second</i>	Um valor de segundos UNSIGNED1 no intervalo de 0 a 59.
Return:	TimeReturn retorna um Time_t (Um UNSIGNED3 com a hora do dia no formato HHMMDD.)

A função **TimeFromParts** retorna um valor Time_t dos parâmetros *hour*, *minute* e *second* .

Exemplo:

```
IMPORT STD;
UNSIGNED1  MyHour   := 23;
UNSIGNED1  MyMinute := 59;
UNSIGNED1  MySecond := 50;

T := STD.Date.TimeFromParts(MyHour, MyMinute, MySecond);
//T contains 235950
```

IsLeapYear

STD.Date.IsLeapYear(*ano*)

<i>year</i>	A year value in the INTEGER2 format.
Return:	IsLeapYear retorna um valor BOOLEAN

A função **IsLeapYear** retorna TRUE se *year* é um ano bissexto no calendário gregoriano (ou gregoriano proléptico).

Exemplo:

```
IMPORT STD;
INTEGER2 MyYear := 2012;    //2012

D := STD.Date.IsLeapYear(MyYear);
//D contains TRUE, 2012 is a leap year
```

IsDateLeapYear

STD.Date.IsDateLeapYear(*date*)

<i>date</i>	Uma data no formato Date_t. (Um UNSIGNED4 que contém um valor de data no formato YYYYMMDD.)
Return:	IsDateLeapYear retorna um valor BOOLEAN.

A função **IsDateLeapYear** retorna TRUE se *year* representado em *date* é um ano bissexto no calendário gregoriano (ou gregoriano proléptico).

Exemplo:

```
IMPORT STD;
MyDate := 20120112;    //Jan. 12, 2012

D := STD.Date.IsDateLeapYear(MyDate);
    //D contains TRUE, 2012 is a leap year
```

IsValidDate

STD.Date.IsValidDate(*date* , [*yearLowerBound*],[*yearUpperBound*])

<i>date</i>	Um valor de data no formato Date_t.
<i>yearLowerBound</i>	O ano mínimo aceitável. Opcional. O padrão é 1800.
<i>yearUpperBound</i>	O ano máximo aceitável. Opcional. O padrão é 2100.
Return:	IsValidDateYear retorna um valor BOOLEAN.

A função **sValidDate** retorna TRUE se a data é válida, verificando o se o ano está em um intervalo e validando cada um dos outros componentes individuais.

Exemplo:

```
IMPORT STD;
d1 := 19631122;
d2 := 19990230;
firstTest := STD.Date.IsValidDate(d1); //d1 is valid
secondTest := STD.Date.IsValidDate(d2); //d2 is not valid
```

IsValidTime

STD.Date.IsValidTime(*time*)

<i>time</i>	Um valor de hora no formato Time_t.
Return:	IsValidTime retorna um valor BOOLEAN

A função IsValidDate retorna TRUE se a hora é válida, verificando cada um dos componentes individuais (horas, minutos e segundos).

Exemplo:

```
IMPORT STD;
d1 := 19631122;
d2 := 19990230;
firstTest := STD.Date.IsValidDate(d1); //d1 is valid
secondTest := STD.Date.IsValidDate(d2); //d2 is not valid
```

IsValidGregorianDate

STD.Date.IsValidGregorianDate(*date*)

<i>date</i>	Um valor de data no formato Date_t. (um UNSIGNED4 que contém um valor de data no formato YYYYMMDD).
Return:	IsValidGregorianDateYear retorna um valor BOOLEAN

A função **IsValidGregorianDate** retorna TRUE se a data é válida no calendário gregoriano. O ano deve estar entre 1601 e 30827.

Exemplo:

```
IMPORT STD;
d1 := 19991122;
d2 := 15130230;
firstTest := STD.Date.IsValidGregorianDate(d1); // TRUE
secondTest := STD.Date.IsValidGregorianDate(d2); // FALSE
```


FromGregorianYMD

STD.Date.FromGregorianYMD(*year*, *month*, *day*)

<i>year</i>	um valor de ano INTEGER2 no intervalo 0 a 9999.
<i>month</i>	Um valor de mês UNSIGNED1 no intervalo de 1 a 12.
<i>day</i>	Um valor de dia UNSIGNED1 no intervalo de 1 a 31.
Return:	FromGregorianYMD retorna um valor UNSIGNED4.

A função **FromGregorianYMD** retorna um valor Days_t dos parâmetros *year*, *month* e *day* que representa o número de dias desde 31 de dezembro de 1BC no calendário gregoriano (consulte as FAQ do calendário, por Claus Tondering, em <http://www.tondering.dk/claus/calendar.html>).

Exemplo:

```
IMPORT STD;
INTEGER2 MyYear := 2012;
UNSIGNED1 MyMonth := 1;
UNSIGNED1 MyDay := 1;

D := STD.Date.FromGregorianYMD(MyYear, MyMonth, MyDay);
//D contains 734503
```

ToGregorianYMD

STD.Date.ToGregorianYMD(*days*)

<i>days</i>	Um valor de ano no formato Days_t.
Return:	ToGregorianYMD retorna valores separados de ano, mês e dia.

A função **ToGregorianYMD** converte o número de dias desde 31 de dezembro de 1 BC em uma data no calendário gregoriano. A função retorna um módulo com três valores exportados: Year, Month e Day

Exemplo:

```
IMPORT STD;
INTEGER2 MyYear := 2012;
UNSIGNED1 MyMonth := 1;
UNSIGNED1 MyDay := 1;

J := STD.Date.FromGregorianYMD(MyYear,MyMonth,MyDay);
//J contains 734503

X := STD.Date.ToGregorianYMD(J);
// X is a module with exported values

Y := X.Year; //Y contains 2012
M := X.Month; //M contains 1
D := X.Day; //D contains 1
```

FromStringToDate

STD.Date.FromStringToDate(*date_text*, *format*)

<i>date_text</i>	A string a ser convertida.
<i>format</i>	O formato da string de entrada. Consulte a documentação de strftime para obter detalhes (http://strftime.org/)
return:	A data correspondida na string. Retornará 0 se não houver correspondência ou se houver correspondência, mas o resultado for uma data inválida.

A função **FromStringToDate** converte uma string em Date_t usando o formato de string relevante.

Se a data resultante tiver que ser representável dentro do calendário Gregoriano após o ano 1600, você deve usar a função Std.Date.IsValidGregorianDate() para determinar sua validade.

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'

Exemplo:

```
IMPORT STD;

D1 := STD.Date.FromStringToDate('19720607', '%Y%m%d');
//D1 contains 19720607
D2 := STD.Date.FromStringToDate('19720007', '%Y%m%d');
//D2 contains 0
D3 := STD.Date.FromStringToDate('4/29/1974', '%m/%d/%Y');
//D3 contains 19740429
D4:= STD.Date.FromStringToDate('29/4/1974', '%d/%m/%Y');
//D4 contains 19740429
```

Veja também: IsValidGregorianDate

Today

STD.Date.Today()

Return:	Today retorna date_t (um UNSIGNED4 que contém um valor de data no formato YYYYMMDD) representando a data atual.
---------	---

A função **Today** retorna a data atual no fuso horário local.

Exemplo:

```
IMPORT STD;  
  
D1 := STD.Date.Today();  
    //D1 contains today's date
```

CurrentDate

STD.Date.CurrentDate ([*in_local_time*])

<i>in_local_time</i>	TRUE se o valor retornado deve ser local para o cluster que calcula a hora, FALSE para UTC. Opcional. O padrão é FALSE.
Return:	Today retorna um Date_t que representa a data atual.

A função **CurrentDate** retorna a data atual. Se o parâmetro *in_local_time* for TRUE, os valores retornados serão os locais do cluster que calcula a data. Se FALSE, será retornado valor UTC.

Exemplo:

```
IMPORT STD;
d1 := STD.Date.CurrentDate(True);
//d1 contains the current local date
```

CurrentTime

STD.Date.CurrentTime ([*in_local_time*])

<i>in_local_time</i>	TRUE se o valor retornado deve ser local para o cluster que calcula a hora, FALSE para UTC. Opcional. O padrão é FALSE.
Return:	Today retorna um time_t (Um UNSIGNED3 com a hora do dia no formato HHMMDD.)

A função **CurrentTime** retorna a hora atual. Se o parâmetro *in_local_time* for TRUE, os valores retornados serão os locais do cluster que calcula a hora. Se FALSE, será retornada a hora UTC.

Exemplo:

```
IMPORT STD;
t1 := STD.Date.CurrentTime(True);
//t1 contains the current local time of day
```

DayOfWeek

STD.Date.DayOfWeek(date)

<i>date</i>	Um valor de data no formato Date_t.
Retorno:	DayOfWeek retorna um valor INTEGER representando o dia da semana, em que 1 = domingo.

A função **DayOfWeek** retorna um número que representa o dia da semana para a data especificada. A data deve estar no calendário gregoriano, após o ano 1600.

Exemplo:

```
IMPORT STD;
D1 := STD.Date.DayOfWeek(STD.Date.Today());
// D1 contains the day of the week for today's date
```

DayOfYear

STD.Date.DayOfYear(date)

<i>date</i>	Um valor de data no formato Date_t.
Return:	DayOfYear retorna um valor INTEGER no intervalo de 1 a 366.

A função **DayOfYear** retorna um número que representa o dia do ano da data fornecida. A data deve estar no calendário gregoriano, após o ano 1600.

Exemplo:

```
IMPORT STD;  
D1 := STD.Date.DayOfYear(STD.Date.Today());  
    // D1 contains the day of the year for today's date
```


DaysBetween

STD.Date.DaysBetween(fromDate, toDate)

<i>fromDate</i>	O valor da primeira data, no formato Date_t.
<i>toDate</i>	O valor da última data, no formato Date_t.
Return:	DaysBetween retorna um valor INTEGER com o número de dias entre duas datas.

A função **DaysBetween** calcula o número de dias completos entre duas datas.

Exemplo:

```
IMPORT STD;
StartDate := 19631122;
numDays := STD.Date.DaysBetween(startDate,STD.Date.Today());
// numDays contains the number of days between the startDate and today's date
```

MonthsBetween

STD.Date.MonthsBetween(fromDate, toDate)

<i>fromDate</i>	O valor da primeira data, no formato Date_t.
<i>toDate</i>	O valor da última data, no formato Date_t.
Return:	MonthsBetween retorna um valor INTEGER com o número de meses completos entre duas datas.

A função **MonthsBetween** calcula o número de meses completos entre duas datas.

Exemplo:

```
IMPORT STD;
StartDate := 19631122;
numMonths := STD.Date.MonthsBetween(startDate,STD.Date.Today());
// numMonths contains the number of months between the startDate and today's date
```

AdjustDate

STD.Date.AdjustDate(*date* , [*year_delta*],[*month_delta*] , [*day_delta*])

<i>date</i>	Um valor de data no formato Date_t.
<i>year_delta</i>	O ano mínimo aceitável. Opcional, o valor padrão é zero.
<i>month_delta</i>	O ano mínimo aceitável. Opcional, o valor padrão é zero.
<i>day_delta</i>	O ano máximo aceitável. Opcional, o valor padrão é zero.
Return:	AdjustDate retorna date_t que representa a data ajustada.

A função **AdjustDate** ajusta uma data incrementando ou decrementando valores de ano, mês e/ou dia. A data deve estar no calendário gregoriano, após o ano 1600.

Se a nova data calculada for inválida, será normalizada de acordo com as regras de mktime(). Por exemplo, 20140130 mais 1 mês resulta em 20140302.

Exemplo:

```
IMPORT std;
inDate :=19631123;
Std.Date.AdjustDate(inDate,5,1,3); //returns 19681226
```

Ver também: AdjustCalendar

AdjustCalendar

STD.Date.AdjustCalendar(*date* , [*year_delta*],[*month_delta*] , [*day_delta*])

<i>date</i>	Um valor de data no formato Date_t.
<i>year_delta</i>	O ano mínimo aceitável. Opcional, o valor padrão é zero.
<i>month_delta</i>	O ano mínimo aceitável. Opcional, o valor padrão é zero.
<i>day_delta</i>	O ano máximo aceitável. Opcional, o valor padrão é zero.
Return:	AdjustDate retorna date_t que representa a data ajustada.

A função AdjustCalendar ajusta uma data incrementando ou decrementando meses e/ou anos. A data deve estar no calendário gregoriano, após o ano 1600.

São usadas as regras definidas em McGinn v. State, 46 Neb. 427, 65 N.W. 427, 65 N.W. 46 (1895):

"O termo mês de calendário, usado em decretos ou contratos e não tendo aparentemente sido usado em um sentido diferente, denota um período que termina com o dia do mês posterior numericamente correspondente ao dia do início do período, menos um. Se não houver dia correspondente no mês posterior, o período termina com o seu último dia."

Ajustes de dia são executados após os ajustes de ano e mês usando as regras acima.

Como exemplo, 31 de janeiro de 2014 + 1 mês resulta em 28 de fevereiro de 2014; 31 de janeiro de 2014 + 1 mês + 1 dia resulta em 1 de março de 2014.

Exemplo:

```
IMPORT std;
inDate :=19631123;
Std.Date.AdjustCalendar(inDate,5,1,3); //returns 19681226
```

Ver também: AdjustDate

MonthWeekNumFromDate

STD.Date.MonthWeekNumFromDate(date, startingDayOfWeek)

<i>date</i>	A data (no formato Date_t) para a qual calcular o número da semana.
<i>startingDay-OfWeek</i>	Opcional, o número do índice do primeiro dia da semana, 1 a 7, em que 1 = domingo. O padrão é 1.
Retorno:	O número da semana 1-based da data, relativo ao início do mês da data.

A função **WeekNumFromDate** retorna o número da semana com 1-based no mês da data. A semana 1 sempre contém o primeiro dia do mês e a semana 2 começa no dia da semana seguinte indicado pelo valor de *startingDayOfWeek*.

Esta não é uma implementação ISO-8601 que calcular números de semanas ("week dates").

Exemplo:

```
IMPORT STD;
startDate := STD.Date.Today();
weekNum := STD.Date.MonthWeekNumFromDate(startDate, 2);
weekNum;
```

Ver também: [YearWeekNumFromDate](#)

YearWeekNumFromDate

STD.Date.YearWeekNumFromDate(date, startingDayOfWeek)

<i>date</i>	A data (no formato Date_t) para a qual calcular o número da semana.
<i>startingDay-OfWeek</i>	Opcional, o número do índice do primeiro dia da semana, 1 a 7, em que 1 = domingo. O padrão é 1.
Retorno:	O número da semana 1-based da data, relativo ao início do ano da data.

A função **YearWeekNumFromDate** retorna o número da semana 1-based no ano da data. A semana 1 sempre contém o primeiro dia do ano e a semana 2 começa no dia da semana seguinte indicado pelo valor de *startingDayOfWeek*.

Esta não é uma implementação ISO-8601 que calcular números de semanas ("week dates").

Exemplo:

```
IMPORT STD;
startDate := STD.Date.Today();
weekNum := STD.Date.YearWeekNumFromDate(startDate, 2);
weekNum;
```

Ver também: [MonthWeekNumFromDate](#)

UniqueTZAbbreviations

STD.Date.TimeZone.UniqueTZAbbreviations()

Retorno:	Um novo DATASET({STRING5 tzAbbrev}) contendo as abreviações exclusivas do fuso horário.
----------	---

A função **STD.Date.TimeZone.UniqueTZAbbreviations** retorna uma lista de abreviações de fuso horário exclusivas do dataset codificado no módulo TimeZone. Todas as abreviações estão em maiúsculas.

Exemplo:

```
IMPORT STD;  
STD.Date.TimeZone.UniqueTZAbbreviations();
```

UniqueTZLocations

STD.Date.TimeZone.UniqueTZLocations()

Retorno:	Um novo DATASET ({STRING name}) contendo os nomes de locais exclusivos.
----------	---

A função **STD.Date.TimeZone.UniqueTZLocations** Retorna uma lista de nomes de locais exclusivos do dataset codificado. Todos os nomes estão em maiúsculas.

Exemplo:

```
IMPORT STD;  
STD.Date.TimeZone.UniqueTZLocations();
```


TZDataForLocation

STD.Date.TimeZone.TZDataForLocation(*location*)

<i>location</i>	OBRIGATÓRIO. O nome do local a ser pesquisado; deve ser uma sequência maiúscula não vazia.
Retorno:	Um novo DATASET(STRING5 tzAbbrev, INTEGER4 secondsOffset) contendo os registros encontrados para o local especificado.

A função **STD.Date.TimeZone.TZDataForLocation** retorna os registros de fuso horário para um determinado local.

Exemplo:

```
IMPORT STD;  
STD.Date.TimeZone.TZDataForLocation( 'ASIA' );
```

Ver também: FindTZData

FindTZData

STD.Date.TimeZone.FindTZData(*timeZoneAbbrev*, [*location*])

<i>timeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário a ser pesquisado; deve ser uma string maiúscula não vazia.
<i>location</i>	OPCIONAL. O nome do local a ser pesquisado; se um local não for fornecido ou for uma string vazia, todos os registros correspondentes apenas à abreviação serão retornados.
Retorno:	<p>Um novo DATASET(TZDataLayout) contendo os registros encontrados.</p> <pre>EXPORT TZDataLayout := RECORD STRING5 tzAbbrev; // Time zone abbreviation; always uppercase // may be duplicated between records INTEGER4 secondsOffset; // Number of seconds east (positive) // or west (negative) of UTC SET OF STRING15 locations; // Names of locations that use the given // time zone abbreviation END;</pre>

A função **STD.Date.TimeZone.TZDataForLocation** retorna os registros de fuso horário para uma determinada abreviação e local opcional. Um local deve ser fornecido como um método de diferenciação se a abreviação tiver entradas duplicadas.

Exemplo:

```
IMPORT STD;
STD.Date.TimeZone.FindTZData('CST', 'NORTH AMERICA');
```

Ver também: [TZDataForLocation](#)

SecondsBetweenTZ

STD.Date.TimeZone.SecondsBetweenTZ(*fromTimeZoneAbbrev*, *toTimeZoneAbbrev*, [*fromLocation*,] [*toLocation*])

<i>fromTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário designada como ponto final; deve ser uma sequência maiúscula não vazia.
<i>toTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário designada como ponto final; deve ser uma sequência maiúscula não vazia.
<i>fromLocation</i>	OPCIONAL. O nome do local que acompanha o <i>fromTimeZoneAbbrev</i> se um local não for fornecido ou for uma string vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
<i>toLocation</i>	OPCIONAL. O nome do local que acompanha <i>toTimeZoneAbbrev</i> ; se um local não for fornecido ou for uma sequência vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
Retorna:	O número de segundos entre os dois fusos horários; retorna zero se um dos fusos horários não puder ser encontrado

A função **STD.Date.TimeZone.SecondsBetweenTZ** calcula o deslocamento, em segundos, entre dois fusos horários diferentes. Cada fuso horário é designado por uma abreviação de fuso horário necessária e um nome de local opcional. O resultado é o número de segundos (que podem ser positivos ou negativos) que precisariam ser aplicados a uma hora ao viajar de *fromTimeZoneAbbrev* para *toTimeZoneAbbrev*.

Esteja ciente de que alguns fusos horários representam explicitamente o horário de verão, portanto, é totalmente possível alterar não apenas os fusos horários, mas também a observância do horário de verão em uma única chamada

Exemplo:

```
IMPORT STD;  
STD.Date.TimeZone.SecondsBetweenTZ('CST', 'IST', 'NORTH AMERICA', '');
```

Ver também: [AdjustTimeTZ](#)

AdjustTimeTZ

STD.Date.TimeZone.AdjustTimeTZ(*time*, *fromTimeZoneAbbrev*, *toTimeZoneAbbrev*, [*fromLocation*,] [*toLocation*])

<i>time</i>	OBRIGATÓRIO. O valor da hora (no formato Time_t) para ajustar.
<i>fromTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário em que se supõe que o valor da <i>hora</i> esteja; deve ser uma sequência maiúscula não vazia
<i>toTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário designada como ponto final; deve ser uma sequência maiúscula não vazia.
<i>fromLocation</i>	OPCIONAL. O nome do local que acompanha o <i>fromTimeZoneAbbrev</i> se um local não for fornecido ou for uma string vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
<i>toLocation</i>	OPCIONAL. O nome do local que acompanha <i>toTimeZoneAbbrev</i> ; se um local não for fornecido ou for uma sequência vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
Retorno:	O valor de tempo fornecido (no formato Time_t) ajustado pela diferença entre os dois fusos horários especificados; se um fuso horário não puder ser encontrado, o valor da hora original será retornado inalterado.

A função **STD.Date.TimeZone.AdjustTimeTZ** ajusta um determinado valor de tempo Time_t para outro fuso horário. O horário especificado e o fuso horário de destino são designados por uma abreviação de fuso horário necessário e um nome de local opcional.

Exemplo:

```
IMPORT STD;
STD.Date.TimeZone.AdjustTimeTZ(205246, 'CST', 'IST', 'NORTH AMERICA', '');
```

Ver também: SecondsBetweenTZ

ToLocalTime

STD.Date.TimeZone.ToLocalTime(*utcTime*, *toTimeZoneAbbrev*, [*toLocation*])

<i>utcTime</i>	OBRIGATÓRIO. O valor da hora UTC (no formato Time_t) para ajuste.
<i>toTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário designada como ponto final; deve ser uma sequência maiúscula não vazia.
<i>toLocation</i>	OPCIONAL. O nome do local que acompanha <i>toTimeZoneAbbrev</i> ; se um local não for fornecido ou for uma sequência vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
Retorno:	O valor da hora UTC fornecido (no formato Time_t) ajustado para o fuso horário definido por <i>toTimeZoneAbbrev</i> e <i>toLocation</i> ; se o fuso horário não puder ser encontrado, o valor da hora original será retornado inalterado

A função **STD.Date.TimeZone.ToLocalTime** converte um horário UTC em um horário designado por uma abreviação de fuso horário e local opcional.

Exemplo:

```
IMPORT STD;
STD.Date.TimeZone.ToLocalTime(205246, 'CST', 'NORTH AMERICA');
```

Ver também: [AdjustTimeTZ](#), [ToUTCTime](#)

ToUTCTime

STD.Date.TimeZone.ToUTCTime(*localTime*, *fromTimeZoneAbbrev*, [*fromLocation*])

<i>localTime</i>	OBRIGATÓRIO. O valor da hora (no formato Time_t) para ajustar.
<i>fromTimeZoneAbbrev</i>	OBRIGATÓRIO. A abreviação do fuso horário em que se supõe que o valor <i>localTime</i> esteja; deve ser uma string maiúscula não vazia.
<i>fromLocation</i>	OPCIONAL. O nome do local que acompanha o <i>fromTimeZoneAbbrev</i> se um local não for fornecido ou for uma string vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado.
Retorno:	O valor da hora local fornecido ajustado para a hora UTC; se o fuso horário especificado não puder ser encontrado, o valor da hora UTC original será retornado inalterado.

A função **STD.Date.TimeZone.ToUTCTime** converte um horário local, definido com uma abreviação de fuso horário e local opcional, em um horário UTC.

Exemplo:

```
IMPORT STD;  
STD.Date.TimeZone.ToUTCTime(205246, 'CST', 'NORTH AMERICA');
```

Ver também: [AdjustTimeTZ](#), [ToLocalTime](#)

AppendTZOffset

STD.Date.TimeZone.AppendTZOffset(*infile*, *timeZoneAbbrevField*, *newOffsetField*, [*fromLocationField*,][*toTimeZoneAbbrev*,][*toLocation*])

<i>infile</i>	OBRIGATÓRIO. O dataset a ser processado.
<i>timeZoneAbbrevField</i>	OBRIGATÓRIO. O campo no inFile que contém a abreviação de fuso horário a ser usada para correspondência; os valores nesse campo devem estar em maiúsculas. Não, esta não é uma lei real.
<i>newOffsetField</i>	OBRIGATÓRIO. O campo a ser anexado ao inFile que conterá o número de segundos de deslocamento do UTC. Não, esta não é uma lei real.
<i>fromLocationField</i>	OPCIONAL. O campo no inFile que contém o local do fuso horário para aquele citado por <i>timeZoneAbbrevField</i> . Isso não é uma string. O valor padrão é nulo (indicando que não há campo de localização do fuso horário).
<i>toTimeZoneAbbrev</i>	OPCIONAL. A abreviação do fuso horário para uso em todos os cálculos, como uma string. Padrões para 'UTC'
<i>toLocation</i>	OPCIONAL. O nome do local que acompanha <i>toTimeZoneAbbrev</i> ; se um local não for fornecido ou for uma sequência vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado. O padrão é uma sequência vazia
Retorna:	<p>Um novo dataset com a mesma definição de registro do inFile, mas com quatro novos campos adicionados. Os novos campos são nomeados com base no nome fornecido como o atributo <i>newOffsetField</i>. Os campos anexados são:</p> <pre>INTEGER4 <newOffsetField> // Offset, in seconds, between original // time zone and toTimeZoneAbbrev BOOLEAN <newOffsetField>_is_valid // TRUE if <newOffsetField> contains a // valid value // If <newOffsetField>_is_valid is FALSE // then <newOffsetField> will be zero. STRING5 <newOffsetField>_tz // The value of toTimeZoneAbbrev STRING15 <newOffsetField>_location // The time zone location for // <newOffsetField>_tz.</pre>

O **STD.Date.TimeZone.AppendTZOffset** pega um conjunto de dados que contém uma abreviação de fuso horário e local opcionais e anexa quatro novos atributos ao dataset que contém informações úteis para converter um valor de tempo em outro fuso horário.

Isso pode ser útil como uma etapa de ETL, em que os dados de hora são comuns em relação a um fuso horário específico (por exemplo, UTC). As ações nessa macro de função são conceitualmente semelhantes a `SecondsBetweenTZ()`, mas aplicadas a um conjunto de dados inteiro e um pouco mais eficientemente.

Observação: Para que essa macro de função seja executada corretamente, o código de chamada deve importar a biblioteca STD.

Exemplo:

```
IMPORT STD;
ds := DATASET ([
    {120000, 'CT'},
    {120000, 'ET'}
], {Std.Date.Time_t time, STRING tz});
utcOffsetDS := Std.Date.TimeZone.AppendTZOffset(ds, tz, seconds_to_utc);
OUTPUT(utcOffsetDS, NAMED('offset_to_utc_result'));

ptOffsetDS := Std.Date.TimeZone.AppendTZOffset (ds, tz, seconds_to_pacific_time,
    toTimeZoneAbbrev := 'PT',
    toLocation := 'NORTH AMERICA');
OUTPUT(ptOffsetDS, NAMED('offset_to_pacific_time_result'));
```

Ver também: `AppendTZAdjustedTime` , `SecondsBetweenTZ`

AppendTZAdjustedTime

STD.Date.TimeZone.AppendTZAdjustedTime(*infile*, *timeField*, *timeZoneAbbrevField*, *newTimeField*, [*fromLocationField*,] [*toTimeZoneAbbrev*,] [*toLocation*])

<i>infile</i>	OBRIGATÓRIO. O dataset a ser processado.
<i>timeField</i>	OBRIGATÓRIO. O campo no inFile que contém um horário representado no formato Time_t. Isso não é uma string.
<i>timeZoneAbbrevField</i>	OBRIGATÓRIO. O campo no inFile que contém a abreviação de fuso horário a ser usada para correspondência; os valores nesse campo devem estar em maiúsculas.
<i>newTimeField</i>	OBRIGATÓRIO. O campo a ser anexado ao inFile que conterá o valor ajustado de <i>timeField</i> .
<i>fromLocationField</i>	OPCIONAL. O campo no inFile que contém o local do fuso horário para aquele citado por <i>timeZoneAbbrevField</i> . O padrão é um valor nulo (indicando que não há atributo de local do fuso horário.) Se um local não for fornecido ou for uma sequência vazia, o primeiro registro correspondente a <i>TimeZoneAbbrevField</i> será usado
<i>toTimeZoneAbbrev</i>	OPCIONAL. A abreviação do fuso horário para uso em todos os cálculos, como uma string. O padrão é 'UTC'
<i>toLocation</i>	OPCIONAL. O nome do local que acompanha <i>toTimeZoneAbbrev</i> ; se um local não for fornecido ou for uma string vazia, o primeiro registro correspondente a <i>TimeZoneAbbrev</i> será usado; O padrão é uma sequência vazia
Retorno:	<p>Um novo dataset com a mesma definição de registro do inFile, mas com quatro novos campos adicionados; os novos campos são nomeados com base no nome fornecido como o atributo <i>newOffsetField</i>:</p> <pre>std.Date.Time_t <newOffsetField> // Value of timeField expressed in new // time zone BOOLEAN <newOffsetField>_is_valid // TRUE if <newOffsetField> contains a // valid value // If <newOffsetField>_is_valid is FALSE // then <newOffsetField> will have the same // value as timeField. STRING5 <newOffsetField>_tz // The value of toTimeZoneAbbrev STRING15 <newOffsetField>_location // The time zone location for // <newOffsetField>_tz</pre>

O **STD.Date.TimeZone.AppendTZAdjustedTime** utiliza um determinado dataset que contém um horário (no formato Time_t), uma abreviação de fuso horário e um local opcional para o fuso horário, lê de acrescentar quatro novos campos ao dataset: Um novo atributo Time_t contendo a hora original expressa em um fuso horário diferente e três atributos que fornecem informações sobre esse fuso horário de destino e a validade da tradução.

Isso pode ser útil como uma etapa de ETL, em que os dados de hora são comuns em relação a um fuso horário específico (por exemplo, UTC). As ações nessa macro de função são conceitualmente semelhantes a AdjustTimeTZ(), mas aplicadas a um conjunto de dados inteiro e um pouco mais eficientemente.

Observação: Para que essa macro de função seja executada corretamente, o código de chamada deve importar a biblioteca STD.

Exemplo:

```
IMPORT STD;
ds := DATASET ([
    {120000, 'CT'},
    {120000, 'ET'}
], {Std.Date.Time_t time, STRING tz});

utcRewriteDS := Std.Date.TimeZone.AppendTZAdjustedTime(ds, time, tz, utc_time);
OUTPUT(utcRewriteDS, NAMED('utc_result'));

ptRewriteDS := Std.Date.TimeZone.AppendTZAdjustedTime (ds, time, tz, pacific_time,
    toTimeZoneAbbrev := 'PT',
    toLocation := 'NORTH AMERICA');
OUTPUT(ptRewriteDS, NAMED('pacific_time_result'));
```

Ver também: `AppendTZOffset` , `AdjustTimeTZ`

Manipulação de Cluster

Node

STD.System.Thorlib.Node()

Return:	Node retorna um valor UNSIGNED INTEGER4.
---------	--

A função **Node** retorna o número (baseado em zero) do nó da Refinaria de dados (Thor) ou do Motor de entrega rápida de dados (Roxie).

Example:

```
A := STD.System.Thorlib.Node();
```

Nodes

STD.System.Thorlib.Nodes()

Return:	Node retorna um valor UNSIGNED INTEGER4.
---------	--

A função **Nodes** retorna o número de nós no cluster Thor (para hThor e Roxie, retorna sempre 1). Esse número é o mesmo que a constante de tempo de compilação CLUSTERSIZE. A função Nodes é avaliada a cada vez que é chamada. Portanto, a opção entre usar a função e a constante depende das circunstâncias.

Exemplo:

```
A := STD.System.Thorlib.Nodes();
```

LogicalToPhysical

STD.System.Thorlib.LogicalToPhysical (*filename* [, *createflag*])

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
<i>createflag</i>	Um valor booleano que indica se <i>filename</i> deve ser criado. Se omitido, o padrão é FALSE.
Return:	LogicalToPhysical retorna um valor VARSTRING.

A função **LogicalToPhysical** (lógico para físico) retorna o nome físico do arquivo representado pelo *nome do arquivo lógico*.

Exemplo:

```
A := STD.System.Thorlib.LogicalToPhysical('Fred');
```

DaliServer

STD.System.Thorlib.DaliServer ()

Return:	Daliserver retorna um valor VARSTRING.
---------	--

A função **DaliServer** retorna o IP e a porta do servidor de armazenamento de dados (Dali) do ambiente que está executando a tarefa.

Exemplo:

```
A := Thorlib.Daliserver();
```

Group

STD.System.Thorlib.Group ()

Return:	Group retorna um valor VARSTRING
---------	----------------------------------

A função **Group** retorna o nome do grupo de nós onde a workunit está sendo executada. Essa função não é permitida em clusters Roxie. Esse nome é usado no código ECL para especificar o CLUSTER de destino para uma ação OUTPUT ou um atributo PERSISTed.

Exemplo:

```
IMPORT STD;  
A := STD.System.Thorlib.Group();
```


GetExpandLogicalFileName

ThorLib.GetExpandLogicalFileName (*filename*)

<i>filename</i>	Uma string terminada por nulo que contém o nome lógico do arquivo.
Return:	GetExpandLogicalFileName returns a VARSTRING (null-terminated) value.

A função **GetExpandLogicalFileName** retorna uma string que contém o nome do arquivo lógico expandido (incluindo o escopo padrão, se o nome do arquivo não tiver um til na primeira posição) com todos os caracteres em minúsculas. Esse é o mesmo valor usado internamente em DATASET e OUTPUT.

Exemplo:

```
A := ThorLib.GetExpandLogicalFileName('Fred');
```

Manipulação de Job

WUID

STD.System.Job.WUID ()

Return:	WUID retorna um valor VARSTRING.
---------	----------------------------------

A função **WUID** retorna o identificador de workunit da workunit atual. Esse identificador é o mesmo que a constante de tempo de compilação WORKUNIT.

Exemplo:

```
A := STD.System.Job.WUID();
```

Target

STD.System.Job.Target ()

Return:	Target retorna um valor VARSTRING
---------	-----------------------------------

A função **Target** retorna o nome do cluster onde a workunit está sendo executada. Essa função não é suportada em clusters Roxie. Esse nome é usado por #WORKUNIT, pelo utilitário de linha de comando do ECL e pelo utilitário de linha de comando eclplus para especificar o cluster de destino de um job.

Exemplo:

```
A := STD.System.Job.Target();
```

Name

STD.System.Job.Name ()

Return:	Name retorna uma valor VARSTRING
---------	----------------------------------

A função **Name** retorna o nome da workunit.

Exemplo:

```
A := STD.System.Job.Name();
```

User

STD.System.Job.User ()

Return:	User retorna um valor VARSTRING.
---------	----------------------------------

A função **User** retorna o nome do usuário que está executando a workunit.

Exemplo:

```
A := STD.System.Job.User();
```

OS

STD.System.Job.OS ()

Return:	OS retorna um valor VARSTRING.
---------	--------------------------------

A função **OS** retorna o sistema operacional (Windows ou Linux) do cluster onde a workunit está sendo executada.

Example:

```
A := STD.System.Job.OS();
```

Platform

STD.System.Job.Platform ()

Return:	Platform retorna um valor VARSTRING.
---------	--------------------------------------

A função **Platform** retorna o nome da plataforma (hthor, thor ou roxie) do cluster onde a workunit está sendo executada.

Example:

```
A := STD.System.Job.Platform();
```


LogString

STD.System.Job.LogString (*message*)

<i>message</i>	Uma expressão de string que contém o texto a ser colocado no arquivo de log.
Retorno:	LogString retorna um valor INTEGER.

A função **LogString** gera "USER:" seguido pelo texto de *mensagem* no arquivo de log do eclagent ou do Roxie e retorna o comprimento do texto gravado no arquivo.

Exemplo:

```
A := STD.System.Job.LogString('The text message to log');  
//places USER:The text message to log  
//in the log file
```

Monitoramento de Arquivo

MonitorFile

STD.File.MonitorFile(*event*, [*ip*], *filename*, [*,subdirs*] [*,shotcount*] [*,espserverIPport*])

dfuwuid := **STD.File.fMonitorFile**(*event*, [*ip*], *filename*, [*,subdirs*] [*,shotcount*] [*,espserverIPport*]);

<i>event</i>	Uma string terminada por nulo que contém o nome definido pelo usuário do evento a ser acionado quando <i>filename</i> aparecer. O valor é usado como o primeiro parâmetro da função EVENT.
<i>ip</i>	Opcional. Uma string terminada por nulo que contém o endereço IP do arquivo a ser monitorado. Normalmente, é uma zona de entrada de arquivos. Pode ser omitido apenas se o parâmetro <i>filename</i> contém uma URL completa.
<i>filename</i>	Uma string terminada por nulo que contém o caminho completo do arquivo a monitorar. Pode conter caracteres curinga (* e ?).
<i>subdirs</i>	Opcional. Um valor booleano que indica se devem ser incluídos os arquivos em subdiretórios que correspondem à máscara de curingas, quando <i>filename</i> contém curingas. Se omitido, o padrão é falso.
<i>shotcount</i>	Opcional. Um valor inteiro que indica o número de vezes que o evento deve ser gerado antes da conclusão do job de monitoramento. Um valor de um negativo (-1) indica que o job de monitoramento continuará até ser abortada manualmente. Se omitido, o padrão é 1.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo <code>lib_system.ws_fs_server</code> .
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para o job de monitoramento.
Return:	fMonitorFile retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **MonitorFile** cria uma workunit de monitoramento de arquivos no servidor do DFU. Após ser recebida, a tarefa entra no modo de monitoramento (que pode ser vista na tela da workunit DFU do ECL Watch), que sonda em intervalos fixos. Esse intervalo é especificado na configuração **monitorinterval** do servidor do DFU. O intervalo padrão é 900 segundos (15 minutos). Se um arquivo com nome adequado chegar nesse intervalo, será acionado um *evento* com o nome do objeto acionador como subtipo do evento (consulte a função EVENT).

Esse processo continua até que:

- 1) O número do eventos *shotcount* tenham sido gerados
- 2) O usuário aborte a workunit DFU.

As funções **STD.File.AbortDfuWorkunit** e **STD.File.WaitDfuWorkunit** podem ser usadas para abortar ou aguardar o job DFU passando por elas e retornando ao *dfuwuid*.

Observe as seguintes advertências e restrições:

- 1) Os eventos são gerados apenas no início do job de monitoração ou posteriormente, de acordo com o intervalo de sondagem.
- 2) Observe que *evento* é gerado se o arquivo físico tiver sido criado desde o último intervalo de sondagem. Consequentemente, o *evento* poderá ocorrer antes de o arquivo ser fechado e os dados serem todos gravados. Para garantir que o arquivo não seja lido posteriormente antes de ser concluído, você deve usar uma técnica que elimine essa pos-

sibilidade. Por exemplo, use um arquivo separado de "indicador" em vez do próprio arquivo, ou renomeie o arquivo depois de criado e totalmente gravado.

3) Ao monitorar arquivos físicos, o parâmetro de subtipo da função EVENT (seu segundo parâmetro) corresponde ao URL completo do arquivo com um IP absoluto em vez do nome DNS/netbios do arquivo. Esse parâmetro não pode ser recuperado, mas pode ser usado apenas para corresponder um valor específico.

Exemplo:

```
EventName := 'MyFileEvent';
FileName  := 'c:\\test\\myfile';
LZ       := '10.150.50.14';
STD.File.MonitorFile(EventName,LZ,FileName);
OUTPUT('File Found') : WHEN(EVENT(EventName,'*'),COUNT(1));
```

MonitorLogicalFileName

STD.File.MonitorLogicalFileName(*event*, *filename*, [, *shotcount*] [, *espserverIPport*])

dfuwuid := **STD.File.fMonitorLogicalFileName**(*event*, *filename*, [, *shotcount*] [, *espserverIPport*]);

<i>event</i>	Uma string terminada por nulo que contém o nome definido pelo usuário do evento a ser acionado quando <i>filename</i> aparecer. O valor é usado como o primeiro parâmetro da função EVENT.
<i>filename</i>	Uma string terminada por nulo que contém o nome do arquivo lógico no DFU a ser monitorado.
<i>shotcount</i>	Opcional. Um valor inteiro que indica o número de vezes que o evento deve ser gerado antes da conclusão do job de monitoramento. Um valor de um negativo (-1) indica que o job de monitoramento continuará até ser abortada manualmente. Se omitido, o padrão é 1.
<i>espserverIPport</i>	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
<i>dfuwuid</i>	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para o job de monitoramento.
Return:	fMonitorLogicalFileName retorna uma string terminada com nulo contendo o ID da workunit DFU (DFUWUID).

A função **MonitorLogicalFileName** cria uma workunit de monitoramento de arquivos no servidor do DFU. Após ser recebida, o job entra no modo de monitoramento (que pode ser vista na tela da workunit DFU do eclwatch), que sonda em intervalos fixos (o padrão é 15 minutos). Se um arquivo com nome adequado chegar nesse intervalo, será acionado um *evento* com o nome do objeto acionador como subtipo do evento (consulte a função EVENT).

Esta função não permite caracteres curinga. Para monitorar arquivos físicos ou diretórios usando curingas, use a função MonitorFile .

Esse processo continua até que:

- 1) O número do eventos *shotcount* tenham sido gerados
- 2) O usuário aborte a workunit DFU.

As funções STD.File.AbortDfuWorkunit e STD.File.WaitDfuWorkunit podem ser usadas para abortar ou aguardar o job DFU passando por elas e retornando ao *dfuwuid*.

Observe as seguintes advertências e restrições:

- 1) Se existir um arquivo correspondente no momento em que o job do Monitorador DFU for iniciada, esse arquivo não gerará um evento. Ele só gerará o evento quando o arquivo tiver sido removido e recriado.
- 2) Se um arquivo for criado e depois removido (ou removido e depois recriado) entre os intervalos de sondagem, ele não será visto pelo monitor e não acionará um evento.
- 3) Os eventos são gerados apenas durante o intervalo de sondagem.

Exemplo:

```
EventName := 'MyFileEvent';
FileName  := 'test::myfile';
IF (STD.File.FileExists(FileName),
    STD.File.DeleteLogicalFile(FileName));
```

```
STD.File.MonitorLogicalFileName(EventName,FileName);  
OUTPUT('File Created') : WHEN(EVENT(EventName,'*'),COUNT(1));  
  
rec := RECORD  
    STRING10 key;  
    STRING10 val;  
END;  
afile := DATASET([{'A', '0'}], rec);  
OUTPUT(afile,,FileName);
```

Logging

dbglog

STD.System.Log.dbglog (*text*)

<i>text</i>	Uma string que contém o texto a ser gravado.
Return:	dbglog não retorna um valor.

A ação **dbglog** grava a string de *text* no arquivo eclagent.log da tarefa.

Exemplo:

```
IMPORT STD;  
STD.System.Log.dbglog('Got Here 1');    //write text to log
```


addWorkunitInformation

STD.System.Log.addWorkunitInformation (*text* [*Código*])

<i>text</i>	Uma string que contém o texto a ser gravado.
<i>code</i>	Opcional. O número de código associado ao <i>text</i> . Se omitido, o padrão é zero (0).
Return:	addWorkunitInformation não retorna um valor.

A ação **addWorkunitInformation** grava a string *text* no arquivo eclagent.log da workunit e também exibe o *code* e o *text* da seção Info na página da workunit no ECL Watch.

Exemplo:

```
IMPORT STD;
STD.System.Log.addWorkunitInformation('Got Here',1);
//write text to log and display "1: Got Here" as Info
```

addWorkunitWarning

STD.System.Log.addWorkunitWarning (*text* [, *code*])

<i>text</i>	Uma string que contém o texto a ser gravado.
<i>code</i>	Opcional. O número de código associado ao text. <i>text</i> . Se omitido, o padrão é zero (0).
Return:	addWorkunitWarning não retorna um valor.

A ação **addWorkunitError** grava a string *text* no arquivo eclagent.log da workunit e também exibe o *code* e o *text* na caixa de ferramentas Syntax Errors, juntamente com a seção Erros da página da workunit no ECL Watch.

Exemplo:

```
IMPORT STD;  
STD.System.Log.addWorkunitWarning('Got Here',1);  
//write text to log and display "1: Got Here" in Warnings
```

addWorkunitError

STD.System.Log.addWorkunitError (*text* [, *code*])

<i>text</i>	Uma string que contém o texto a ser gravado.
<i>code</i>	Opcional. O código número a ser associado ao <i>text</i> . Se omitido, o padrão é zero (0).
Return:	addWorkunitError não retorna um valor.

A ação **addWorkunitError** grava a string *text* no arquivo eclagent.log da workunit e também exibe o *code* e o *text* na caixa de ferramentas Syntax Errors, juntamente com a seção Erros da página da workunit no ECL Watch.

Exemplo:

```
IMPORT STD;  
STD.System.Log.addWorkunitError('Got Here',1);  
//write text to log and display "1: Got Here" in Errors
```

getGlobalId

STD.System.Log.getGlobalId ()

Retorno:	getGlobalId retorna o Global Id
----------	---------------------------------

O **getGlobalId** obtém o Global Id associado à consulta ou workunit atual. Exemplo:

```
IMPORT STD;  
STD.System.Log.getGlobalId();
```

getLocalId

STD.System.Log.getLocalId ()

Retorno:	getLocalId retorna o Local Id.
----------	--------------------------------

O **getLocalId** obtém o Local ID associado à consulta ou unidade de trabalho atual. Exemplo:

```
IMPORT STD;  
STD.System.Log.getLocalId();
```

generateGloballyUniqueID

STD.System.Log.generateGloballyUniqueID ()

Retorno:	generateGloballyUniqueID returns a globally unique identifier.
----------	--

generateGloballyUniqueID retorna um identificador global único (GUID) com codificação base58. Condificação base58 é similar a base64 mas evita encoding mas evita caracteres não alfanuméricos e letras visualmente ambíguas. Ele foi projetado para evitar erros por usuários humanos que inserem manualmente os dados copiando de alguma fonte visual. Ele permite copiar/colar facilmente porque um clique duplo geralmente seleciona a string inteira.

```
IMPORT STD;

value1 := std.system.log.generateGloballyUniqueId() : INDEPENDENT;
value2 := NOFOLD(std.system.log.generateGloballyUniqueId()) : INDEPENDENT;

OUTPUT(value1);
OUTPUT(value2);
OUTPUT(IF (value1 = value2, 'Values are not unique', 'Values are unique'));
```

Auditoria

Audit

STD.Audit.Audit(*type*, *message*)

<i>type</i>	Uma constante de string que contém o tipo de entrada de auditoria. No momento, somente INFO é fornecido.
<i>message</i>	Uma string que contém o texto da entrada de auditoria.
Return:	Audit retorna um valor BOOLEAN que indica se foi bem-sucedida ou não.

Audit grava a *message* no log de eventos do Windows ou no log de sistema do Linux no computador do ECL Agent. As entradas podem ser recuperadas dos logs usando ferramentas padrão do sistema operacional.

Exemplo:

```
STD.Audit.Audit('INFO','Audit Message');
```


Utilitários

GetHostName

result := **STD.System.Util.GetHostName** (*ip*);

<i>ip</i>	Uma string terminada por nulo que contém o endereço IP da máquina remota.
Return:	GetHostName retorna um valor VARSTRING (terminado por nulo).

A função **GetHostName** executa uma busca reversa de DNS para retornar o nome do host da máquina no endereço *ip* especificado.

Exemplo:

```
IP := '10.150.254.6';
```

```
OUTPUT(STD.System.Util.GetHostName(IP));
```

ResolveHostName

result := **STD.System.Util.ResolveHostName** (*host*);

<i>host</i>	Uma string terminada por nulo que contém o nome DNS da máquina remota.
Return:	ResolveHostName retorna um valor VARSTRING (terminado por nulo).

A função **ResolveHostName** executa uma busca de DNS para retornar o endereço IP do nome de *host* especificado.

Exemplo:

```
host := 'dataland_dali.br.seisint.com';  
OUTPUT(STD.System.Util.ResolveHostName(host));
```

GetUniqueInteger

result := **STD.System.Util.GetUniqueInteger** ([*dali*]);

<i>dali</i>	Opcional. Uma string terminada por nulo que contém o endereço IP do Dali remoto de onde o número será obtido. Se omitida, o padrão é local.
Return:	GetUniqueInteger retorna um valor UNSIGNED8.

A função **GetUniqueInteger** retorna um número que é único entre todos os nós escravos do *dali* especificado.

Exemplo:

```
IMPORT STD;  
  
OUTPUT(STD.System.Util.GetUniqueInteger());
```

GetEspUrl

result := **STD.File.GetEspUrl** ([*username*, *userPW*]);

<i>username</i>	Opcional. Uma STRING que contém um nome do usuário para uso no acesso autenticado ao processo do ESP. Se omitida, indica que não é necessário autenticar o usuário.
<i>userPW</i>	Opcional. Uma STRING que contém a senha a ser usada com o usuário citado no argumento <i>username</i> . Se o <i>username</i> estiver vazio, isso será ignorado
Retorno:	GetEspUrl retorna um STRING contendo a URL completa (incluindo porta e esquema HTTP) para um processo do servidor ESP. Se mais de um processo ESP estiver definido, será retornado o primeiro processo encontrado. Retorna uma string vazia se não for possível encontrar um processo do servidor ESP no ambiente.

A função **GetEspUrl** retorna a URL completo para um processo do servidor ESP.

Exemplo:

```
EspAddress := STD.File.GetEspUrl
```

PlatformVersionCheck

result := **STD.System.Util.PlatformVersionCheck**(*v*);

<i>v</i>	Obrigatório. A versão mínima da plataforma no formato xx.xx.xx, xx.xx, ou xx (em que xx é um número inteiro e não precisa ser preenchido com zero); caracteres extras à direita (como o '-1' no exemplo abaixo) são ignorados.
Retorno:	TRUE se a versão atual da plataforma for igual ou superior ao argumento, caso contrário, FALSE.

A função **PlatformVersionCheck** testa uma string de versão completa em relação às constantes individuais da versão da plataforma para determinar se a versão da plataforma é pelo menos tão alta quanto o argumento. Esta função é avaliada em tempo de compilação se o argumento for uma constante. Isso o torna útil para incorporar declarações #IF (), como mostrado no exemplo.

Exemplo:

```
IMPORT STD;
#IF(STD.System.Util.PlatformVersionCheck('8.2.0-1'))
  OUTPUT('Platform check TRUE');
#ELSE
  OUTPUT('Platform check FALSE');
#END
```

Depuração

GetParseTree

STD.System.Debug.GetParseTree ()

Return:	GetParseTree retorna um valor STRING.
---------	---------------------------------------

A função **GetParseTree** retorna uma representação textual da correspondência ocorrida usando colchetes (como: a[b[c]d]) para indicar o aninhamento. Essa função é usada apenas dentro da estrutura RECORD ou TRANSFORM que define o resultado de uma operação PARSE. Essa função é útil para depurar operações de PARSE.

Exemplo:

```
IMPORT STD;

r := {string150 line};
d := dataset([
{'Ge 34:2 And when Shechem the son of Hamor the Hivite, '+
'prince of the country, saw her, he took her, and lay with her, '+
'and defiled her.'},
{'Ge 36:10 These are the names of Esaus sons; Eliphaz the son of '+
'Adah the wife of Esau, Reuel the son of Bashemath the wife of '+
'Esau.'}
],r);
PATTERN ws := [' ','\t',' ']*;
PATTERN patStart := FIRST | ws;
PATTERN patEnd := LAST | ws;
PATTERN article := ['A','The','Thou','a','the','thou'];
TOKEN patWord := PATTERN('[a-zA-Z]+');
TOKEN Name := PATTERN('[A-Z][a-zA-Z]+');
RULE Namet := name OPT(ws 'the' ws name);
PATTERN produced_by := OPT(article ws) ['son of','daughter of'];
PATTERN produces_with := OPT(article ws) ['wife of'];
RULE progeny := namet ws ( produced_by | produces_with ) ws namet;
results := RECORD
  STRING LeftName := MATCHTEXT(Namet[1]);
  STRING RightName := MATCHTEXT(Namet[2]);
  STRING LinkPhrase := IF(MATCHTEXT(produced_by[1])<>' ',
    MATCHTEXT(produced_by[1]),
    MATCHTEXT(produces_with[1]));
  STRING Tree := 'Tree: ' + STD.System.Debug.getParseTree();
END;
outfile1 := PARSE(d,line,progeny,results,SCAN ALL);
/* the Tree field output looks like this:
Tree: [namet[name"Shechem"] ws " " produced_by"the son of" ws " " namet[name"Hamor"]]
*/
```


GetXMLParseTree

STD.System.Debug.GetXMLParseTree ()

Return:	GetXMLParseTree retorna um valor STRING.
---------	--

O **GetXMLParseTree** a função GetXMLParseTree retorna uma representação textual da correspondência ocorrida usando tags XML para indicar o aninhamento. Essa função é usada apenas dentro da estrutura RECORD ou TRANSFORM que define o resultado de uma operação PARSE. Essa função é útil para depurar operações de PARSE.

Exemplo:

```
IMPORT STD; r := {stIMPORT STD;

r := {string150 line};
d := dataset([
{'Ge 34:2 And when Shechem the son of Hamor the Hivite, '+
'prince of the country, saw her, he took her, and lay with her, '+
'and defiled her.'},
{'Ge 36:10 These are the names of Esaus sons; Eliphaz the son of '+
'Adah the wife of Esau, Reuel the son of Bashemath the wife of '+
'Esau.'}
],r);

PATTERN ws := [' ','\t',' ',''];
PATTERN patStart := FIRST | ws;
PATTERN patEnd := LAST | ws;
PATTERN article := ['A','The','Thou','a','the','thou'];
TOKEN patWord := PATTERN('[a-zA-Z]+');
TOKEN Name := PATTERN('[A-Z][a-zA-Z]+');
RULE Namet := name OPT(ws 'the' ws name);
PATTERN produced_by := OPT(article ws) ['son of','daughter of'];
PATTERN produces_with := OPT(article ws) ['wife of'];
RULE progeny := namet ws ( produced_by | produces_with ) ws namet;
results := RECORD
    STRING LeftName := MATCHTEXT(Namet[1]);
    STRING RightName := MATCHTEXT(Namet[2]);
    STRING LinkPhrase := IF(MATCHTEXT(produced_by[1])<>'',
        MATCHTEXT(produced_by[1]),
        MATCHTEXT(produces_with[1]));
    STRING Tree := STD.System.Debug.getXMLParseTree();
END;
outfile1 := PARSE(d,line,progeny,results,SCAN ALL);
/* the Tree field output
looks like this:
<namet>
  <name>Shechem</name>
</namet>
<ws> </ws>
<produced_by>the son of</produced_by>
<ws> </ws>
<namet>
  <name>Hamor</name>
</namet>
*/
```

Sleep

STD.System.Debug.Sleep (*duration*)

<i>duration</i>	Um valor inteiro que especifica o comprimento do período da pausa, em milissegundos.
Return:	Sleep não retorna um valor.

A função **Sleep** pausa o processamento pelo número de milissegundos especificado em *duration* .

Exemplo:

```
IMPORT STD;  
STD.System.Debug.Sleep(1000);    //pause for one second before continuing
```

msTick

STD.System.Debug.msTick ()

Return:	msTick retorna um valor inteiro sem sinal de 4 bytes.
---------	---

A função **msTick** retorna o tempo decorrido desde o momento inicial, em milissegundos. O momento inicial é indefinido, o que torna essa função útil apenas para determinar o tempo decorrido entre chamadas à função, subtraindo o valor de retorno mais recente do anterior. Quando o valor de retorno alcançar o valor máximo de um inteiro sem sinal de 4 bytes (2^{32} ou 4 Gb), recomeça em zero (0). Isso ocorre aproximadamente a cada 49,71 dias.

Exemplo:

```
IMPORT STD;
t1 := STD.System.Debug.msTick() : STORED('StartTime'); //get start time

ds1 := DATASET([ {0,0,0,0,0} ],
               {UNSIGNED4 RecID,
                UNSIGNED4 Started,
                UNSIGNED4 ThisOne,
                UNSIGNED4 Elapsed,
                UNSIGNED4 RecsProcessed});

RECORDOF(ds1) XF1(ds1 L, integer C) := TRANSFORM
    SELF.RecID := C;
    SELF := L;
END;
ds2 := NORMALIZE(ds1,100000,XF1(LEFT,COUNTER));

RECORDOF(ds1) XF(ds1 L) := TRANSFORM
    SELF.Started := T1;
    SELF.ThisOne := STD.System.Debug.msTick();
    SELF.Elapsed := SELF.ThisOne - SELF.Started;
    SELF := L;
END;

P := PROJECT(ds2,XF(LEFT)) : PERSIST('~RTTEST::TestTick');
R := ROLLUP(P,
           LEFT.Elapsed=RIGHT.Elapsed,
           TRANSFORM(RECORDOF(ds1),
                     SELF.RecsProcessed := RIGHT.RecID - LEFT.RecID,
                     SELF := LEFT));

paws := STD.System.Debug.Sleep(1000); //pause for one second before continuing
SEQUENTIAL(paws,OUTPUT(P, ALL),OUTPUT(R, ALL));
```

E-mail

SendEmail

STD.System.Email.SendEmail (*sendto*, *subject*, *body*, *server*, *port*, *sender*)

<i>sendto</i>	Uma string terminada por nulo que contém uma lista delimitada por vírgulas dos endereços dos destinatários. A validade dos endereços não é verificada. Portanto, é responsabilidade do programador garantir essa validade.
<i>subject</i>	Uma string terminada por nulo que contém a linha de assunto.
<i>body</i>	Uma string terminada por nulo que contém o texto do e-mail a ser enviado. Essa string deve ter a codificação de caracteres "ISO-8859-1 (latin1)" (o conjunto de caracteres padrão do ECL). Texto em qualquer outro conjunto de caracteres deve ser enviado como anexo (consulte a função <code>STD.System.Email.SendEmailAttachText()</code>).
<i>server</i>	Opcional. Uma string terminada por nulo que contém o nome do servidor de e-mail. Se omitida, assume o valor padrão da variável de ambiente <code>SMTPserver</code> .
<i>port</i>	Opcional. Um valor inteiro UNSIGNED4 que contém o número da porta. Se omitido, assume o valor padrão da variável de ambiente <code>SMTPport</code> .
<i>sender</i>	Opcional. Uma string terminada por nulo que contém o endereço do remetente. Se omitida, assume o valor padrão da variável de ambiente <code>emailSenderAddress</code> .

A função **SendEmail** envia uma mensagem de e-mail.

Exemplo:

```
STD.System.Email.SendEmail( 'me@mydomain.com', 'testing 1,2,3', 'this is a test message');
```

SendEmailAttachData

STD.System.Email.SendEmailAttachData (*sendto*, *subject*, *body*, *attachment*, *mimietype*, *filename*, *server*, *port*, *sender*)

<i>sendto</i>	Uma string terminada por nulo que contém uma lista delimitada por vírgulas dos endereços dos destinatários. A validade dos endereços não é verificada. Portanto, é responsabilidade do programador garantir essa validade.
<i>subject</i>	Uma string terminada por nulo que contém a linha de assunto.
<i>body</i>	Uma string terminada por nulo que contém o texto do e-mail a ser enviado. Essa string deve ter a codificação de caracteres "ISO-8859-1 (latin1)" (o conjunto de caracteres padrão do ECL). Um texto em outro conjunto de caracteres deve ser enviado como <i>attachment</i> .
<i>attachment</i>	Um valor DATA que contém os dados binários a serem anexados.
<i>mimetype</i>	Uma string terminada por nulo que contém o MIME-type do <i>attachment</i> , que pode incluir entrymeters (como "text/plain; charset=ISO-8859-3"). Ao anexar dados binários gerais que não têm um tipo MIME específico, use "application/octet-stream".
<i>filename</i>	Uma string terminada por nulo que contém o nome do <i>attachment</i> a ser exibido pelo leitor de e-mail.
<i>server</i>	Opcional. Uma string terminada por nulo que contém o nome do servidor de e-mail. Se omitida, assume o valor padrão da variável de ambiente SMTPserver.
<i>port</i>	Opcional. Um valor inteiro UNSIGNED4 que contém o número da porta. Se omitido, assume o valor padrão da variável de ambiente SMTPport.
<i>sender</i>	Opcional. Uma string terminada por nulo que contém o endereço do remetente. Se omitida, assume o valor padrão da variável de ambiente emailSenderAddress.

A função **SendEmailAttachData** envia uma mensagem de e-mail com um *anexo* binário.

Exemplo:

```
DATA15 attachment := D'test attachment';
STD.System.Email.SendEmailAttachData( 'me@mydomain.com',
                                       'testing 1,2,3',
                                       'this is a test message',
                                       attachment,
                                       'application/octet-stream',
                                       'attachment.txt' );
```

SendEmailAttachText

STD.System.Email.SendEmailAttachText (*sendto*, *subject*, *body*, *attachment*, *mimietype*, *filename*, *server*, *port*, *sender*)

<i>sendto</i>	Uma string terminada por nulo que contém uma lista delimitada por vírgulas dos endereços dos destinatários. A validade dos endereços não é verificada. Portanto, é responsabilidade do programador garantir essa validade.
<i>subject</i>	Uma string terminada por nulo que contém a linha de assunto.
<i>body</i>	Uma string terminada por nulo que contém o texto do e-mail a ser enviado. Essa string deve ter a codificação de caracteres "ISO-8859-1 (latin1)" (o conjunto de caracteres padrão do ECL). Um texto em outro conjunto de caracteres deve ser enviado como <i>attachment</i> .
<i>attachment</i>	Uma string terminada por nulo que contém o texto a ser anexado.
<i>mimetype</i>	Uma string terminada por nulo que contém o MIME-type do <i>attachment</i> , que pode incluir entymeters (como "text/plain; charset=ISO-8859-3").
<i>filename</i>	Uma string terminada por nulo que contém o nome do <i>attachment</i> a ser exibido pelo leitor de e-mail.
<i>server</i>	Opcional. Uma string terminada por nulo que contém o nome do servidor de e-mail. Se omitida, assume o valor padrão da variável de ambiente SMTPserver.
<i>port</i>	Opcional. Um valor inteiro UNSIGNED4 que contém o número da porta. Se omitido, assume o valor padrão da variável de ambiente SMTPport.
<i>sender</i>	Opcional. Uma string terminada por nulo que contém o endereço do remetente. Se omitida, assume o valor padrão da variável de ambiente emailSenderAddress.

A função **SendEmailAttachText** envia uma mensagem de e-mail com um *anexo* de texto.

Exemplo:

```
STD.System.Email.SendEmailAttachText( 'me@mydomain.com', 'testing 1,2,3',  
                                       'this is a test message', 'this is a test attachment',  
                                       'text/plain; charset=ISO-8859-3', 'attachment.txt');
```

Serviços da Workunit

WorkunitExists

STD.System.Workunit.WorkunitExists(*wuid* [, *online*] [, *archived*])

<i>wuid</i>	Uma string terminada por nulo que contém o WorkUnit IDentifier a ser localizado.
<i>online</i>	Opcional. Um valor booleano true/false que especifica se a pesquisa deve ser executada online. Se omitido, o padrão é TRUE.
<i>archived</i>	Opcional. Um valor booleano true/false que especifica se a pesquisa deve ser executada nos arquivos. Se omitido, o padrão é FALSE.
Return:	WildMatch retorna um valor BOOLEAN.

A função **WorkunitExists** retorna se *wuid* existe.

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitExists('W20070308-164946'));
```

WorkunitList

STD.System.Workunit.WorkunitList (*lowwuid* [, *highwuid*] [, *username*] [, *cluster*] [, *jobname*] [, *state*] [, *priority*] [, *fileread*] [, *filewritten*] [, *roxiecluster*] [, *eclcontains*] [, *online*] [, *archived*] [, *appvalues*])

<i>lowwuid</i>	Uma string terminada por nulo que contém o menor WorkUnit IDentifier a ser listado. Pode ser uma string vazia.
<i>highwuid</i>	Opcional. Uma string terminada por nulo que contém o maior WorkUnit IDentifier a ser listado. Se omitida, o padrão é uma string vazia.
<i>cluster</i>	Opcional. Uma string terminada por nulo que contém o nome do cluster onde a workunit foi executada. Se omitida, o padrão é uma string vazia.
<i>jobname</i>	Opcional. Uma string terminada por nulo que contém o nome da workunit. Pode conter caracteres curinga (* ?) Se omitida, o padrão é uma string vazia.
<i>state</i>	Opcional. Uma string terminada por nulo que contém o estado da workunit. Se omitida, o padrão é uma string vazia.
<i>priority</i>	Opcional. Uma string terminada por nulo que contém a prioridade da workunit. Se omitida, o padrão é uma string vazia.
<i>fileread</i>	Opcional. Uma string terminada por nulo que contém o nome de um arquivo lido pela workunit. Pode conter caracteres curinga (* ?) Se omitida, o padrão é uma string vazia.
<i>filewritten</i>	Opcional. Uma string terminada por nulo que contém o nome de um arquivo gravado pela workunit. Pode conter caracteres curinga (* ?) Se omitida, o padrão é uma string vazia.
<i>roxiecluster</i>	Opcional. Uma string terminada por nulo que contém o nome do cluster Roxie. Se omitida, o padrão é uma string vazia.
<i>eclcontains</i>	Opcional. Uma string terminada por nulo que contém o texto a ser pesquisado no código da ECL da workunit. Pode conter caracteres curinga (* ?) Se omitida, o padrão é uma string vazia.
<i>online</i>	Opcional. Um valor booleano true/false que especifica se a pesquisa deve ser executada online. Se omitido, o padrão é TRUE.
<i>archived</i>	Opcional. Um valor booleano true/false que especifica se a pesquisa deve ser executada nos arquivos. Se omitido, o padrão é FALSE.
<i>appvalues</i>	Opcional. Uma string terminada por nulo que contém os valores do aplicativo a serem pesquisados. Use uma string no formato appname/key=value ou appname/*=value.
Return:	Retorna um dataset no seguinte formato:

A função **WorkunitList** retorna um dataset de todas as workunits que atendem aos critérios de pesquisa especificados pelos parâmetros passados para a função. Todos os parâmetros são parâmetros de pesquisa e todos, exceto o primeiro, podem ser omitidos. Portanto, a forma mais fácil de passar um único parâmetro é usar a técnica de passagem de parâmetros NAMED.

O DATASET resultante está neste formato:

```
WorkunitRecord := RECORD
  STRING24 wuid;
  STRING owner{MAXLENGTH(64)};
  STRING cluster{MAXLENGTH(64)};
  STRING roxiecluster{MAXLENGTH(64)};
  STRING job{MAXLENGTH(256)};
  STRING10 state;
  STRING7 priority;
  STRING20 created;
  STRING20 modified;
```

```
    BOOLEAN online;  
    BOOLEAN protected;  
END;
```

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitList(''));  
    //list all current workunits  
OUTPUT(STD.System.Workunit.WorkunitList('',  
    NAMED eclcontains := 'COUNT'));  
    //list only those where the ECL code contains the word 'COUNT'  
    //this search is case insensitive and does include comments  
  
STD.System.Workunit.SetWorkunitAppValue('MyApp','FirstName','Jim',TRUE);  
OUTPUT(STD.System.Workunit.WorkunitList(appvalues := 'MyApp/FirstName='Jim'));  
    //returns a list of workunits with app values where FirstName='Jim'
```

Ver também: SetWorkunitAppValue

SetWorkunitAppValue

STD.System.Workunit.SetWorkunitAppValue (*app*, *key*, *value*, [*overwrite*])

<i>app</i>	O nome do aplicativo a ser definido.
<i>key</i>	O nome do valor a ser definido.
<i>value</i>	O valor a ser definido.
<i>overwrite</i>	Um indicador booleano TRUE ou FALSE que indica se o valor pode substituir um valor existente. O padrão é true
Return:	SetWorkunitAppValue retorna TRUE se o valor for definido corretamente.

A função **SetWorkunitAppValue** define um valor do aplicativo na workunit atual. Retorna TRUE se o valor for definido corretamente.

Exemplo:

```
IMPORT STD;
STD.System.Workunit.SetWorkunitAppValue('MyApp', 'FirstName', 'Jim', TRUE);
OUTPUT(STD.System.Workunit.WorkunitList(appvalues := 'MyApp/FirstName='Jim'));
//returns a list of workunits with app values where FirstName='Jim'
```

Ver também: WorkunitList

WUIDonDate

STD.System.Workunit.WUIDonDate (*year, month, day, hour, minute*)

<i>year</i>	Um inteiro sem sinal que contém o valor do ano.
<i>month</i>	Um inteiro sem sinal que contém o valor do mês.
<i>day</i>	Um inteiro sem sinal que contém o valor do dia.
<i>hour</i>	Um inteiro sem sinal que contém o valor da hora.
<i>minute</i>	Um inteiro sem sinal que contém o valor do minuto.
Return:	WUIDonDate retorna um valor VASTRING

A função **WUIDonDate** retorna um ID de workunit válido para uma workunit que atende aos parâmetros passados.

Exemplo:

```
lowwuid := STD.System.Workunit.WUIDonDate(2008,02,13,13,00);  
highwuid := STD.System.Workunit.WUIDonDate(2008,02,13,14,00);  
OUTPUT(STD.System.Workunit.WorkunitList(lowwuid,highwuid));  
//returns a list of workunits between 1 & 2 PM on 2/13/08
```

WUIDdaysAgo

STD.System.Workunit.WUIDdaysAgo (*daysago*)

<i>daysago</i>	Um inteiro sem sinal que contém o número de dias a retroceder.
Return:	WUIDdaysAgo retorna um valor VARSTRING.

A função **WUIDdaysAgo** retorna um identificador de tarefa válido de uma tarefa executada no número de dias mais recentes especificado em *daysago* .

Exemplo:

```
daysago := STD.System.Workunit.WUIDdaysAgo(3);  
OUTPUT(STD.System.Workunit.WorkunitList(daysago));  
//returns a list of workunits run in the last 72 hours
```

WorkunitTimeStamps

STD.System.Workunit.WorkunitTimeStamps (*wuid*)

<i>wuid</i>	Uma string terminada por nulo que contém o identificador da workunit.
Return:	WorkunitTimeStamps retorna um valor DATASET.

A função **WorkunitTimeStamps** retorna um DATASET neste formato:

```
EXPORT WsTimeStamp := RECORD
  STRING32 application;
  STRING16 id;
  STRING20 time;
  STRING16 instance;
END;
```

Cada registro no dataset retornado especifica uma etapa no processo de execução da tarefa (criação, compilação, etc.).

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitTimeStamps('W20070308-164946'));
/* produces output like this:
'workunit      ','Created ','2008-02-13T18:28:20Z','
'workunit      ','Modified','2008-02-13T18:32:47Z','
'EclServer     ','Compiled','2008-02-13T18:28:20Z','10.173.9.2:0 '
'EclAgent      ','Started ','2008-02-13T18:32:35Z','training009003'
'Thor - graph1','Finished','2008-02-13T18:32:47Z','training009004'
'Thor - graph1','Started ','2008-02-13T18:32:13Z','training009004'
'EclAgent      ','Finished','2008-02-13T18:33:09Z','training009003'
*/
```

WorkunitMessages

STD.System.Workunit.WorkunitMessages (*wuid*)

<i>wuid</i>	Uma string terminada por nulo que contém o identificador da workunit.
Return:	WorkunitMessages retorna um valor DATASET.

A função **WorkunitTimeStamps** retorna um DATASET neste formato:

```
EXPORT WsMessage := RECORD
  UNSIGNED4 severity;
  INTEGER4 code;
  STRING32 location;
  UNSIGNED4 row;
  UNSIGNED4 col;
  STRING16 source;
  STRING20 time;
  STRING message{MAXLENGTH(1024)};
END;
```

Esta função retorna todas as mensagens na workunit. Cada registro Each record in the retornado no dataset especifica a mensagem na workunit.

O valor de *severidade* pode ser 1 para Alerta, 2 para Erro, ou 3 para informação.

Exemplo:

```
IMPORT STD;
OUTPUT(STD.System.Workunit.WorkunitMessages('W20210602-164946'));
```


WorkunitFilesRead

STD.System.Workunit.WorkunitFilesRead (*wuid*:.)

<i>wuid</i>	Uma string terminada por nulo que contém o identificador de tarefa.
Return:	WorkunitFilesRead retorna um valor DATASET.

A função **WorkunitTimeStamps** retorna um DATASET neste formato:

```
EXPORT WsFileRead := RECORD
  STRING name{MAXLENGTH(256)};
  STRING cluster{MAXLENGTH(64)};
  BOOLEAN isSuper;
  UNSIGNED4 usage;
END;
```

Cada registro no dataset retornado especifica um arquivo lido pela tarefa.

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitFilesRead('W20070308-164946'));
/* produces results that look like this
'rttest::difftest::superfile','thor','true','1'
'rttest::difftest::base1','thor','false','1'
*/
```

WorkunitFilesWritten

STD.System.Workunit.WorkunitFilesWritten (*wuid*)

<i>wuid</i>	Uma string terminada por nulo que contém o identificador da workunit.
Return:	WorkunitFilesWritten retorna um valor DATASET.

A função **WorkunitTimeStamps** retorna um DATASET neste formato:

```
EXPORT WsFileRead := RECORD
  STRING name{MAXLENGTH(256)};
  STRING10 graph;
  STRING cluster{MAXLENGTH(64)};
  UNSIGNED4 kind;
END;
```

Cada registro no dataset retornado especifica um arquivo gravado pela workunit.

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitFilesWritten('W20070308-164946'));
/* produces results that look like this
'rttest::testfetch','graph1','thor','0'
*/
```

WorkunitTimings

STD.System.Workunit.WorkunitTimings (*wuid*)

<i>wuid</i>	Uma string terminada por nulo que contém o identificador da workunit.
Return:	WorkunitTimings retorna um valor DATASET

A função **WorkunitTimeStamps** retorna um DATASET neste formato:

```
EXPORT WsTiming := RECORD
  UNSIGNED4 count;
  UNSIGNED4 duration;
  UNSIGNED4 max;
  STRING name{MAXLENGTH(64)};
END;
```

Cada registro no dataset retornado especifica uma medida de tempo da workunit.

Exemplo:

```
OUTPUT(STD.System.Workunit.WorkunitTimings('W20070308-164946'));
/* produces results that look like this
'1','4','4','EclServer: tree transform'
'1','0','0','EclServer: tree transform: normalize.scope'
'1','1','1','EclServer: tree transform: normalize.initial'
'1','18','18','EclServer: write c++'
'1','40','40','EclServer: generate code'
'1','1010','1010','EclServer: compile code'
'1','33288','33288','Graph graph1 - 1 (1)'
'1','33629','33629','Total thor time: '
'2','1','698000','WorkUnit_lockRemote'
'1','2','2679000','SDS_Initialize'
'1','0','439000','Environment_Initialize'
'1','33775','3710788928','Process'
'1','1','1942000','WorkUnit_unlockRemote'
*/
```

Suporte a BLAS

Esta seção oferece suporte para o subprograma de álgebra linear básica.

As funções BLAS usam o principal mapeamento de coluna para armazenar uma matriz. Este é o mapeamento usado no Fortran, que tem entradas da primeira coluna, seguidas pelas entradas da segunda coluna. Esta é a transposição do formato principal da linha, usado normalmente na linguagem C, em que as entradas da primeira linha são seguidas pelas entradas da segunda linha.

Tipos:

STD.BLAS.Types

<i>value_t</i>	REAL8
<i>dimension_t</i>	UNSIGNED4
<i>matrix_t</i>	SET OF REAL8
<i>Triangle</i>	ENUM(UNSIGNED1, Upper=1, Lower=2)
<i>Diagonal</i>	ENUM(UNSIGNED1, UnitTri=1, NotUnitTri=2)
<i>Side</i>	ENUM(UNSIGNED1, Ax=1, xA=2)

Tipos para suporte a subprogramas de álgebra linear básica de blocos

ICellFunc

STD.BLAS.ICellFunc(*v*, *r*, *c*);

<i>v</i>	O Valor
<i>r</i>	O Ordinal da Linha
<i>c</i>	O Ordinal da Coluna
Return:	O Valor atualizado

ICellFunc é o protótipo de função para Apply2Cells.

Exemplo:

```
IMPORT STD;
REAL8 my_func(STD.BLAS.Types.value_t v, STD.BLAS.Types.dimension_t x,
              STD.BLAS.Types.dimension_t y)
  := 1/v; //set element to the reciprocal value
```

Ver também: Apply2Cells

Apply2Cells

STD.BLAS.Apply2Cells(*m*, *n*, *x*, *f*);

<i>m</i>	Número de Linhas
<i>n</i>	Número de Colunas
<i>x</i>	Matriz
<i>f</i>	Função para aplicar
Return:	O Matriz atualizada

A função **Apply2Cells** itera uma matriz e aplica uma função a cada célula.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.value_t example_1(STD.BLAS.Types.value_t v,
                                STD.BLAS.Types.dimensional_t x,
                                STD.BLAS.Types.dimensional_t y) := FUNCTION
    RETURN IF(x=y, 1.0, 1/v);
END;

init_mat := [1, 2, 4, 4, 5, 10, 2, 5, 2];
new_mat := STD.BLAS.Apply2Cells(3, 3, init_mat, example_1);

// The new_mat matrix will be [1, .5, .25, .25, 1, .1, .5, .2, 1]
```

Ver também: ICellFunc

dasum

STD.BLAS.dasum(*m*, *x*, *incx*, *skipped*);

<i>m</i>	Número de entradas
<i>x</i>	A matriz principal da coluna que mantém o vetor
<i>incxx</i>	O incremento para x, 1 no caso de um vetor real
<i>skipped</i>	O número de entradas ignoradas. O padrão é zero.
Return:	A soma dos valores absolutos

A função **dasum** obtém a soma absoluta, a norma 1 de um vetor.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t test_data := [2, -2, -3, 3, 1, 3, -1, -1, 1];
STD.BLAS.dasum(9, test_data, 1); //sums the absolute values of the matrix, and returns 17
```


daxpy

STD.BLAS.daxpy(*N*, *alpha*, *X*, *incX*, *Y*, *incY*, *x_skipped*, *y_skipped*);

<i>N</i>	Número de entradas
<i>alpha</i>	A coluna principal da matriz que mantém o vetor
<i>X</i>	O incremento para x, 1 no caso de um vetor real
<i>incX</i>	A coluna principal da matriz que mantém o vetor X
<i>Y</i>	A coluna principal da matriz que mantém o vetor Y
<i>incY</i>	O incremento ou passo de Y
<i>x_skipped</i>	O número de entradas ignoradas para chegar ao primeiro X.
<i>y_skipped</i>	O número de entradas ignoradas para chegar ao primeiro Y.
Return:	O Matriz atualizada

A função **daxpy** é usada para somar dois vetores ou matrizes com um multiplicador escalar aplicado durante a operação de soma.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.t_matrix term_1 := [1, 2, 3];
STD.BLAS.Types.t_matrix term_2 := [3, 2, 1].
STD.BLAS.daxpy(3, 2, term_1, 1, term_2, 1); // result is [5, 6, 7]
```

dgemm

STD.BLAS.dgemm(*transposeA*, *transposeB*, *M*, *N*, *K*, *alpha*, *A*, *B*, *beta*, *C*);

<i>transposeA</i>	Verdadeiro quando a transposição de A é usada
<i>transposeB</i>	Verdadeiro quando a transposição de B é usada
<i>M</i>	Número de linhas no produto
<i>N</i>	Número de colunas no produto
<i>K</i>	Número de colunas/linhas para o multiplicador/multiplicado
<i>alpha</i>	Escalar usado em A
<i>A</i>	Matriz A
<i>B</i>	Matriz B
<i>beta</i>	Escala para matriz C
<i>C</i>	Matriz C (ou vazio)
Return:	A Matriz atualizada

A função **dgemm** é usada para multiplicar duas matrizes e opcionalmente adicionar esse produto a outra matriz.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.t_matrix term_a := [2, 4, 8];
STD.BLAS.Types.t_matrix term_c := [2, 1, 1];

STD.BLAS.dgemm(TRUE, FALSE, 3, 3, 1, 1, term_a, term_b);
//the outer product of the term_a and term_b vectors
//result is [4,8, 16, 2, 4, 8, 2, 4, 8]
```

dgetf2

STD.BLAS.dgetf2(*m*, *n*, *a*);

<i>m</i>	Número de linhas da matriz a
<i>n</i>	Número de colunas da matriz a
<i>a</i>	Matriz A
Return:	Matriz composta de fatores, o triângulo inferior tem uma diagonal implícita de uns. O triângulo superior tem a diagonal do composto.

A função **dgetf2** era uma fatoração combinada triangular inferior e superior.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.t_matrix test := [2,4,6,3,10,25, 9,34,100];
STD.BLAS.dgetf2(3, 3, test); //result is [2,2,3,3,4,4,9,16,25];
```

dpotf2

STD.BLAS.dpotf2(*tri*, *r*, *A*, *clear*);

<i>tri</i>	Indica se o triângulo superior ou inferior é usado
<i>r</i>	Número de linhas/colunas na matriz quadrada
<i>A</i>	A matriz quadrada A
<i>clear</i>	Limpa o triângulo não utilizado
Return:	A matriz triangular solicitada

A função **dpotf2** calcula a fatoração de Cholesky de uma matriz definida positiva simétrica real A. O formato da fatoração será $A = U^{**}T^{*}U$ se o parâmetro *tri* for Triangle.Upper ou $A = L^{*}L^{**}T$ se o parâmetro *tri* for Triangle.Lower. Esta é a versão desbloqueada do algoritmo, que chama BLAS nível 2.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t symmetric_pos_def := [4, 6, 8, 6, 13, 18, 8, 18, 29];
Lower_Triangle := BLAS.dpotf2(STD.BLAS.Types.Triangle.lower, 3, symmetric_pos_def);
```

dscal

STD.BLAS.dscal(*N*, *alpha*, *X*, *incX*, *skipped*);

<i>N</i>	Número de elementos no vetor
<i>alpha</i>	O fator de escala
<i>X</i>	A coluna principal da matriz que mantém o vetor
<i>incX</i>	O passo para chegar ao próximo elemento no vetor
<i>skipped</i>	O número de entradas ignoradas para chegar ao primeiro elemento.
Return:	A Matriz atualizada

A função **dscal** escala um vetor alfa.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t test := [1, 1, 1, 2, 2, 2, 3, 3, 3];
result := STD.BLAS.dscal(9, 2.0, test, 1); // multiply each element by
```

dsyrk

STD.BLAS.dsyrk(*tri*, *transposeA*, *N*, *K*, *alpha*, *A*, *beta*, *C*, *clear*);

<i>tri</i>	Indica se o triângulo superior ou inferior é usado
<i>transposeA</i>	Executa a transposição da matriz A para NxK
<i>N</i>	Número de Linhas
<i>K</i>	Número de colunas na matriz de atualização ou transposição
<i>alpha</i>	A escala alfa
<i>A</i>	matriz de atualização, NxK ou KxN
<i>beta</i>	A escala Beta
<i>C</i>	A matriz para atualizar
<i>clear</i>	Limpar o triângulo não atualizado. BLAS supõe que matrizes simétricas têm apenas um dos triângulos e esta opção permite fazer isso ser verdadeiro.
Return:	A Matriz atualizada

A função **dsyrk** implementa uma atualização de classificação simétrica $C \leftarrow \alpha A^*T A + \beta C$ ou $c \leftarrow \alpha A A^*T + \beta C$. C é N x N.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t initC := [1, 1, 1, 2, 2, 2, 3, 3, 3];
STD.BLAS.Types.matrix_t initA := [1, 1, 1];
Test1_mat := STD.BLAS.dsyrk(
    STD.BLAS.Types.Triangle.upper, FALSE, 3, 1, 1, initA, 1, initC, TRUE)
```

dtrsm

STD.BLAS.dtrsm(*side*, *tri*, *transposeA*, *diag*, *M*, *N*, *lda*, *alpha*, *A*, *B*);

<i>side</i>	Lado de A, Side.Ax é $\text{op}(A) X = \alpha B$
<i>tri</i>	Indica se o triângulo superior ou inferior é usado
<i>transposeA</i>	Se $\text{op}(A)$ é a transposição de A
<i>diag</i>	A diagonal (uma diagonal de unidades implícita ou fornecida)
<i>M</i>	Número de Linhas
<i>N</i>	Número de Colunas
<i>lda</i>	A principal dimensão da matriz A, que pode ser M ou N
<i>alpha</i>	O multiplicador escalar de B
<i>A</i>	A matriz triangular
<i>B</i>	A matriz de valores para a solução
Return:	A matriz de coeficientes para obter B

A função **dtrsm** é uma resolvedora de matriz triangular. $\text{op}(A) X = \alpha B$ ou $X \text{op}(A) = \alpha B$ *, onde op é Transpose, X e B são MxN

Exemplo:

```
IMPORT STD;
Side := STD.BLAS.Types.Side;
Diagonal := STD.BLAS.Types.Diagonal;
Triangle := STD.BLAS.Types.Triangle;
STD.BLAS.Types.matrix_t left_a0 := [2, 3, 4, 0, 2, 3, 0, 0, 2];
STD.BLAS.Types.matrix_t mat_b := [4, 6, 8, 6, 13, 18, 8, 18, 29];

Test1_mat := STD.BLAS.dtrsm(Side.Ax, Triangle.Lower, FALSE, Diagonal.NotUnitTri,
                           3, 3, 3, 1.0, left_a0, mat_b)
```

extract_diag

STD.BLAS.extract_diag (*m.n.x*);

<i>m</i>	Número de Linhas
<i>n</i>	Número de Colunas
<i>x</i>	A matriz de onde extrair a diagonal
Return:	Matriz Diagonal

A função **extract_diag** extrai a diagonal da matriz

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t x := [1.0, 2.0, 3.0, 2.0, 2.0, 2.0, 4.0, 4.0, 4.0];
diagonal_only := STD.BLAS.extract_diag(3, 3, x);
```


extract_tri

STD.BLAS.extract_tri (*m*, *n*, *tri*, *dt*, *a*);

<i>m</i>	Número de Linhas
<i>n</i>	Número de Colunas
<i>tri</i>	Indica se o triângulo superior ou inferior é usado
<i>dt</i>	Use Diagonal.NotUnitTri ou Diagonal.UnitTri
<i>a</i>	A matriz, normalmente um composto da fatoração
Return:	Triangle

A função **extract_tri** extrai o triângulo superior ou inferior. A diagonal pode ser a diagonal de unidades real ou implícita.

Exemplo:

```
IMPORT STD;
Diagonal := STD.BLAS.Types.Diagonal;
Triangle := STD.BLAS.Types.Triangle;
STD.BLAS.Types.matrix_t x := [1.0, 2.0, 3.0, 2.0, 2.0, 2.0, 4.0, 4.0, 4.0];
triangle := STD.BLAS.extract_tri(3, 3, Triangle.upper, Diagonal.NotUnitTri, x);
```

make_diag

STD.BLAS.make_diag (m , v , X);

m	Número de entradas na diagonal
v	Valor da opção, o padrão é 1
X	Entrada opcional de valores da diagonal, multiplicados por v
Return:	Matriz Diagonal

A função **make_diag** gera uma matriz diagonal.

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t init1 := [1.0, 2.0, 3.0, 4.0];
Square := STD.BLAS.make_diag(4, 1, init1); //4x4 with diagonal 1, 2, 3, 4
```

make_vector

STD.BLAS.make_vector (m , v);

m	Número de elementos
v	Os valores, o padrão é 1
Return:	O vetor

A função **make_vector** gera um vetor da dimensão n

Exemplo:

```
IMPORT STD;  
twos_vector := STD.BLAS.make_vector(4, 2); // a vector of [2, 2, 2, 2]
```

trace

STD.BLAS.trace ((m , n , x);

m	Número de Linhas
n	Número de Colunas
x	A Matriz
Return:	O trace (soma das entradas da diagonal)

A função **trace** calcula o traço da matriz de entrada

Exemplo:

```
IMPORT STD;
STD.BLAS.Types.matrix_t x := [1.0, 2.0, 3.0, 2.0, 2.0, 2.0, 4.0, 4.0, 4.0];
trace_of_x := STD.BLAS.trace(3,3,x); // the trace is 7
```

Suporte a Math

Esta seção cobre as funções matemáticas comuns na biblioteca padrão.

Infinity

STD.Math.Infinity;

Retorno:	Retorna um valor "infinito" REAL.
----------	-----------------------------------

Infinity retorna um valor "infinito".

Exemplo:

```
IMPORT STD;  
myValue := STD.Math.Infinity;  
myValue;
```

Ver também: [isInfinite](#)

NaN

STD.Math.NaN;

Retorno:	Retorna um valor NaN (Not a Number) sem sinalização.
----------	--

A função **NaN** retorna um valor NaN (Not a Number) sem sinal.

Exemplo:

```
IMPORT STD;  
myValue := STD.Math.NaN;  
myValue;
```

Ver também: IsNan

isInfinite

STD.Math.isInfinite(*val*);

<i>val</i>	O valor a ser testado.
Retorno:	Retorna um BOOLEAN indicando se um valor real é infinito (positivo ou negativo).

A função **isInfinite** retorna se um valor real é infinito (positivo ou negativo).

Exemplo:

```
IMPORT STD;
a := STD.Math.Infinity ;
b := 42.1;
STD.Math.isInfinite(a); //true
STD.Math.isInfinite(b); //false
```

Ver também: Infinity, isFinite

isNaN

STD.Math.isNaN(*val*);

<i>val</i>	O valor a ser testado.
Retorno:	Retorna um BOOLEAN indicando se um valor real é um valor de NaN (not a number).

A função **isNaN** retorna se um valor real é um valor de NaN (não um número).

Exemplo:

```
IMPORT STD;
a := STD.Math.NaN ;
b := 42.1;
STD.Math.isNaN(a); //true
STD.Math.isNaN(b); //false
```

Ver também: NaN, isFinite

isFinite

STD.Math.isFinite(*val*);

<i>val</i>	O valor a ser testado.
Retorno:	Retorna um BOOLEAN indicando se um valor real é um valor válido (nem infinito, nem NaN).

O **isFinite** a função isFinite retorna se um valor real é um valor válido (nem infinito nem NaN).

Exemplo:

```
IMPORT STD;
a := STD.Math.Infinity ;
b := STD.Math.NaN;
c := 42.1;
STD.Math.isFinite(a); //false
STD.Math.isFinite(b); //false
STD.Math.isFinite(c); //true
```

Ver também: `isNaN` , `isInfinite`

FMod

STD.Math.FMod(*numer*, *denom*);

<i>numer</i>	O numerador
<i>denom</i>	O denominador
Retorno:	Retorna o restante em numer/denom de ponto flutuante (arredondado para zero).

A função **FMod** retorna o restante em numer/denom de ponto flutuante (arredondado para zero).

Se denom for zero, o resultado dependerá do indicador `divideByZero`:

- Se definido como zero ou não definido, retorna zero.
- Se definido como "nan", retorna um valor NaN sem sinal.
- Se definido como "fail", gera uma exceção.

Exemplo:

```
#OPTION ('divideByZero', 'nan'); //divide by zero creates a quiet NaN
IMPORT STD;
STD.Math.FMod(5.1, 3.0); // 2.1
STD.Math.FMod(-5.1, 3.0); // -2.1
STD.Math.FMod(5.1, 0); // NaN
```

FMatch

STD.Math.FMatch(*a*, *b*, *epsilon*);

<i>a</i>	O primeiro valor.
<i>b</i>	O segundo valor.
<i>epsilon</i>	A margem de erro permitida.
Retorno:	Retorna se dois valores de ponto flutuante são iguais, dentro da margem do erro epsilon.

A função **FMatch** retorna se dois valores de ponto flutuante são iguais, dentro da margem do erro epsilon.

Exemplo:

```
IMPORT STD;
STD.Math.FMatch(2.6,2.2,0.5); //true
STD.Math.FMatch(2.6,2.2,0.3); //false
```