関数

@から始まる文字列は関数呼び出しを表します。たとえば、

```
@Seen()
```

は、@Seenという関数の呼び出しを表します。関数を呼び出すときに引数を渡すときには、()内に引数を指定します。例えば、ひとつの引数を渡すには、

```
@MessageBox('Test')
```

のようにします。また、

```
@Concat('String1', 'String2')
```

のように複数の引数を渡すこともできます。

定義されている関数は以下のリストを参照してください。@Defunを使用すると独自の関数を定義することもできます。

関数のリスト

- @Account
- @AccountClass
- @AccountDirectory
- @Add
- @Address
- @AddressBook
- @And
- @Attachment
- @BeginWith
- @Body
- @BodyCharset
- @Catch
- @Clipboard
- @ComputerName
- @Concat
- @Contain
- @Copy
- @Date
- @Decode
- @Defun
- @Delete
- @DeleteAttachment
- @Deleted
- @Download

- @DownloadText
- @Draft
- @Equal
- @Eval
- @Execute
- @Exist
- @Exit
- @False
- @Field
- @FieldParameter
- @Find
- @FindEach
- @Flag
- @Folder
- @FolderFlag
- @FolderParameter
- @ForEach
- @FormatAddress
- @FormatDate
- @Forwarded
- @Greater
- @Header
- @HtmlEscape
- @I
- @Id
- @Identity
- @If
- @Include
- @InputBox
- @InvokeAction
- @Junk
- @Label
- @Length
- @Less
- @Load
- @LookupAddressBook
- @Marked
- @MessageBox
- @MessageCount
- @MessageId
- @Messages
- @Move
- @Multipart
- @Name
- @New
- @Not
- @Or
- @OSVersion
- @Param
- @ParseURL
- @Part
- @Partial
- @Passed
- @ProcessId

@Profile
- @ProfileName
- @Progn
- @Quote
- @References
- @RegexFind
- @RegexMatch
- @RegexReplace
- @Remove
- @Replied
- @Save
- @SaveAttachment
- @Script
- @Seen
- @SelectBox
- @Selected
- @Sent
- @Set
- @Size
- @SpecialFolder
- @SubAccount
- @Subject
- @Substring
- @SubstringAfter
- @SubstringBefore
- @Subtract
- @Thread
- @True
- @UnseenMessageCount
- @URI
- @User1
- @User2
- @User3
- @User4
- @Variable
- @While

呼び出し可能な関数

各関数は、状況に応じて呼び出し可能であるかどうかが決まっています。例えば、@AddressBookはUIが存在しない場合には呼び出せません。呼び出し可能であるかどうかを決定する条件には、以下のようなものがあります。

- UIが必要
- UIスレッドからのみ呼び出し可能
- メッセージに対する変更が可能

マクロを呼び出すことのできる各場面において、上記の条件は以下のように設定されます。

- 作成用のテンプレート

    - UIあり
    - UIスレッド
    - 変更可能
- 表示用のテンプレート
    - UIあり
    - UIスレッド
    - 変更不可
- エクスポート・印刷・引用符付き貼り付けのテンプレート
    - UIあり
    - UIスレッド
    - 変更不可
- MessageMacroアクション
    - UIあり
    - UIスレッド
    - 変更可能
- X-QMAIL-Macro, X-QMAIL-DraftMacro
    - UIあり
    - UIスレッド
    - 変更可能
- 手動振り分け・アクティブ振り分け（手動でのメッセージ移動や手動振り分けから引き起こされた場合）の条件
    - UIあり
    - UIスレッド
    - 変更不可
- 自動振り分け・アクティブ振り分け（自動振り分けから引き起こされた場合）の条件
    - UIなし
    - バックグラウンドスレッド
    - 変更不可
- フィルタ
    - UIなし
    - UIスレッド
    - 変更不可
- 同期フィルタ
    - UIなし
    - バックグラウンドスレッド
    - 変更不可
- 基本検索のマクロ
    - UIあり
    - UIスレッド
    - 変更不可
- ヘッダビュー
    - UIあり
    - UIスレッド
    - 変更不可
- リストビューの色
    - UIなし
    - UIスレッド
    - 変更不可
- ヘッダカラム
    - UIなし
    - UIスレッド

    - 変更不可
- 動的メニュー
    - UIなし
    - UIスレッド
    - 変更不可
- フォント
    - UIあり
    - UIスレッド
    - 変更不可

各関数がどの条件で呼び出し可能かは各関数の説明を参照してください。

@AccountClass

```
String @AccountClass(String account?)
```

説明

accountで指定されたアカウントのアカウントクラスを返します。指定された名前のアカウントが見つからない場合にはエラーになります。引数が省略された場合には、コンテキストアカウントのアカウントクラスを返します。コンテキストアカウントがない場合にはエラーになります。

引数

String account
　　アカウント名

エラー

- 引数の数が合っていない場合
- 指定された名前のアカウントが見つからない場合
- コンテキストアカウントがない場合

条件

なし

例

```
# コンテキストアカウントのアカウントクラスを取得
# -> mail
@AccountClass()

# RSSという名前のアカウントのアカウントクラスを取得
# -> rss
@AccountClass('RSS')
```

@AccountDirectory

```
String @AccountDirectory(String account?)
```

説明

accountで指定されたアカウントのディレクトリを返します。指定された名前のアカウントが見つからない場合にはエラーになります。引数が省略された場合には、コンテキストアカウントのディレクトリを返します。コンテキストアカウントがない場合にはエラーになります。

アカウントのディレクトリとは、そのアカウントに関するファイルが保存されるディレクトリです。

引数

String account
　　アカウント名

エラー

- 引数の数が合っていない場合
- 指定された名前のアカウントが見つからない場合
- コンテキストアカウントがない場合

条件

なし

例

```
# コンテキストアカウントのディレクトリを取得
# -> C:\Documents and Settings\username\Application Data\QMAIL3\accounts\Sub
@AccountDirectory()

# Mainという名前のアカウントのディレクトリを取得
# -> C:\Documents and Settings\username\Application Data\QMAIL3\accounts\Main
```

```
@AccountDirectory('Main')
```

@Account

```
String @Account()
```

説明

コンテキストアカウントの名前を返します。コンテキストアカウントがない場合にはエラーになります。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストアカウントがない場合

条件

なし

例

```
# アカウント名を取得
# -> Main
@Account()
```

@Add

```
Number @Add(Number arg1, Number arg2)
```

説明

arg1とarg2の和を返します。

引数

Number arg1
　　数値
Number arg2
　　数値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 1 + 2
# -> 3
@Add(1, 2)
```

@AddressBook

```
String @AddressBook(String to?, String cc?, String bcc?)
```

説明

[アドレスの選択]ダイアログを開きます。to, cc, bccの各引数にRFC2822形式のアドレスリストを指定するとそれらのアドレスが選択された状態でダイアログを開きます。アドレス選択ダイアログで[OK]をクリックして閉じると、選択されたアドレスがRFC2822のヘッダ形式で返されます。

引数

String to
　　Toのアドレス
String cc
　　Ccのアドレス
String bcc
　　Bccのアドレス

エラー

- 引数の数が合っていない場合
- UIスレッド以外から呼び出した場合
- UIがない場合

条件

- UIスレッドからのみ呼び出し可能
- UIが必要

例

```
# [アドレスの選択]ダイアログを開く
@AddressBook()
```

```
# Bccに自分のアドレスを選択した状態で[アドレスの選択]ダイアログを開く
@AddressBook('', '', @I())
```

@Address

```
Address @Address(Field field)
```

説明

引数で指定されたフィールドをアドレスリストとしてパースし、含まれているアドレスのリストを返します。フィールドがアドレスを指定するフィールドでなかった場合やパースに失敗した場合には空のリストを返します。

パースされた各アドレスに対してアドレスを取得します。グループアドレスの場合には含まれるアドレスを全て取り出します。

たとえば、

```
To: test1@example.org,
 Test2 <test2@example.org>,
 Test3: test3@example.org, Test4 <test4@example.org>;
```

というヘッダを含むメッセージに対して、@Address(To)を適用すると、test1@example.org, test2@example.org, test3@example.org, test4@example.orgを含むアドレスのリストが返されます。

引数

Field field
　　アドレスのリストを取得するフィールド

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合

条件

なし

例

```
# Toで指定されたアドレスのリストを取得
@Address(To)
```

@And

```
Boolean @And(Boolean arg+)
```

説明

argで指定された真偽値の論理積を返します。つまり、全ての値がTrueならばTrueを返し、それ以外の場合にはFalseを返します。1個以上の任意の数の引数を渡せます。

引数

Boolean arg
　　真偽値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# TrueとTrueの論理積
# -> True
@And(@True(), @True())

# 既読かつマークされている
@And(@Seen(), @Marked())
```

振り分け

振り分け機能を使用すると指定した条件に合致したメッセージに対して、まとめて指定したフォルダに移動・コピーしたり、フラグやラベルを変更したりすることができます。

振り分けには以下の三種類の振り分けがあり、それぞれ実行されるタイミングが異なります。

手動振り分け
　　メニューやショートカットキーなどを使用して手動で振り分けを実行します。
自動振り分け
　　メッセージをサーバから受信したときに自動で振り分けを実行します。
アクティブ振り分け
　　メッセージを移動・コピーしたときに振り分けを実行します。

振り分け条件の指定は、マクロを使用して指定します。ただし、よく使われる条件に関してはマクロを記述することなく指定することができます。振り分け対象とするメッセージの条件の指定や、合致したメッセージにどのような操作をするかの指定は、振り分けルールの設定で行います。

手動振り分け

メニューやショートカットキーなどを使用して手動で振り分けを実行します。

メニューから[メッセージ]-[振り分け]を選択すると現在選択しているフォルダ内の表示されているメッセージを振り分けます。フィルタによって表示されないようになっているメッセージは振り分けられません。[メッセージ]-[すべてのフォルダを振り分け]を選択すると、アカウント内の全ての通常フォルダ内にあるメッセージを振り分けます。ただし、隠されているフォルダは振り分け対象になりません。選択されているメッセージだけ振り分けるには、[メッセージ]-[選択されたメッセージを振り分け]を選択します。

さらに、MessageApplyRuleBackgroundアクションやMessageApplyRuleBackgroundAllアクションを使うと、振り分けをバックグラウンドで実行することができます。この場合、自動振り分けで既存のメッセージを振り分けるのと同じように振り分けられます。

自動振り分け

メッセージをサーバから受信したときに自動で振り分けを実行します。

自動振り分けを有効にするには、アカウントのプロパティの高度の設定で[自動で振り分ける]にチェックを入れます。新規にサーバからダウンロードされたメッセージが振り分け対象になります。同じタブで[既存のメッセージも振り分ける]にもチェックを入れると、それに加えてフォルダ内にすでにあるメッセージも振り分け対象になります。振り分け対象になる既存のメッセージは受信しているフォルダ内のメッセージだけです。たとえ

ば、POP3アカウントであれば受信箱にあるメッセージ、IMAP4やNNTPでは同期しているフォルダ内のメッセージになります。

アクティブ振り分け

メッセージを移動・コピーしたときに振り分けを実行します。

たとえば、あるフォルダにアクティブ振り分けを設定しておくと、そのフォルダに移動されたメッセージに対してその場で振り分けを適用することができます。アクティブ振り分けは振り分けによって移動されたメッセージに対しても実行されます。このため、ループするようなルールを設定すると[*1]止まらなくなりますので注意してください。

アクティブ振り分けはそのフォルダにメッセージが追加されるときに動作します。たとえば、メッセージをそのフォルダに移動・コピーしたときや、インポートしたときに動作します。また、添付ファイルを削除したときや、分割されたメッセージを結合に、処理後のメッセージに対しても適用されます。ただし、たとえばUndoで元に戻されたときなどには実行されません。また、メッセージを受信したときに受信したメッセージに対しても実行されません。このときには自動振り分けが実行されます。ただし、自動振り分けによって移動された先のフォルダにアクティブ振り分けが設定されているとそちらは実行されます。

IMAP4のリモートフォルダに対してはアクティブ振り分けは無効です。これは、リモートフォルダへのメッセージの追加は先にサーバ上でメッセージが移動されてからそれを同期するという形で動作するためです。サーバ上で移動したメッセージを同期するときには自動振り分けが機能しますのでそちらで振り分けてください。

[*1] Folder1に対して「Folder2に対して移動する」というルールを設定し、Folder2に対して「Folder1に移動する」というルールを設定するなど

@Attachment

```
String @Attachment(String separator?, Boolean uri?)
```

説明

コンテキストメッセージの添付ファイルの名前を返します。複数の添付ファイルがある場合には、separatorで指定した文字列で区切られます。separatorを指定しなかった場合には、「, 」で区切られます。uriにTrueを指定すると名前の代わりにURIが返されます。

引数

String separator
　　セパレータ
Boolean uri
　　URIを返すかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージホルダがない場合（URIを取得する場合）
- コンテキストメッセージホルダが一時的な場合（URIを取得する場合）
- メッセージの取得に失敗した場合

条件

なし

例

```
# 添付ファイルの名前を取得
@Attachment()

# 添付ファイルのURIを「,」で区切って取得
@Attachment(',', @True())
```

@BeginWith

```
Boolean @BeginWith(String s1, String s2, Boolean case?)
```

説明

s1がs2から始まる場合にはTrue、それ以外の場合にはFalseを返します。caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

String s1
　　文字列
String s2
　　文字列
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# %Subjectに[Qs:が含まれる（大文字と小文字を区別）
@Begin(%Subject, '[Qs:', @True())
```

@BodyCharset

```
String @BodyCharset(Number type?, Part part?)
```

説明

コンテキストメッセージの本文のエンコーディングを返します。partが指定された場合にはそのパートの本文のエンコーディングを返します。この関数は、@Bodyを使用したときに使用されるエンコーディングを返します。エンコーディングを直接指定している場合にはそのエンコーディングが、それ以外の場合には自動判定されたエンコーディングを返します。マルチパートの場合に各パートのエンコーディングが異なる場合にはutf-8を返します。

typeには本文のフォーマット方法を指定します。@Bodyのtype引数と同じ値が指定できます。ただし、引数が省略された場合には:BODY-INLINEを指定したのと同じになります。

引数

Number type
　　フォーマット方法
Part part
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# 本文を全て取得
@Body()

# インラインの本文を「> 」で引用して取得
@Body('> ', :BODY-INLINE)

# インラインの本文を取得（ただし、message/rfc822のパートは必ず取得）
@Body('', :BODY-RFC822INLINE)

# マルチパートメッセージではじめのパートの本文を取得
@Body('', :BODY-ALL, @Part(0))
```

@Body

```
String @Body(String quote?, Number type?, Part part?)
```

説明

コンテキストメッセージの本文を返します。partが指定された場合にはそのパートの本文を返します。

quoteには引用符を指定します。空文字列以外を指定すると指定された引用符で引用されます。

typeには本文のフォーマット方法を指定します。指定できるのは以下のいずれかです。

:BODY-ALL
: 本文全てを返します。マルチパートの場合にはパートを展開して返します。

:BODY-RFC822INLINE
: BODY-INLINEと同じですが、message/rfc822のパートはContent-Dispositionに関わらず返されます。

:BODY-INLINE
: インラインの本文を返します。インラインの本文とは、Content-Dispositionが指定されていないかまたはinlineかつ、Content-Typeがtext/*またはmessage/rfc822のものです。

引数が省略された場合には:BODY-ALLを指定したのと同じになります。

引数

String quote
: 引用符

Number type
: フォーマット方法

Part part
: パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合

メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# 本文を全て取得
@Body()

# インラインの本文を取得
@Body(:BODY-INLINE)
```

@Catch

```
Value @Catch(Expr expr, Expr handler)
```

説明

exprを評価しその結果を返します。その間にエラーが発生した場合にはhandlerを評価してその結果を返します。

引数

Expr expr
　　評価する式
Expr handler
　　エラーが起きたときに評価される式

エラー

- 引数の数が合っていない場合
- handlerを評価中にエラーが発生した場合

条件

なし

例

```
# ファイルを読み込んで返すが、エラーが起きたら空文字列を返す
@Catch(@Load('C:\\Temp\\test.txt'), '')

# コンテキストメッセージに既読フラグを設定するが、
# コンテキストメッセージがなかったら何もしない
@Catch(@Seen(@True()), @True())
```

@Clipboard

```
String @Clipboard(String s?)
```

説明

引数が指定されていない場合にはクリップボードから文字列を取得して返します。取得できない場合には空文字列を返します。

引数が指定された場合にはsをクリップボードにセットします。

引数

String s
　　クリップボードにセットする文字列

エラー

- 引数の数が合っていない場合
- クリップボードにセットするのに失敗した場合

条件

なし

例

```
# クリップボードから文字列を取得
@Clipboard()

# クリップボードに本文をコピー
@Clipboard(@Body())
```

リストビューの色

リストビューの各行には条件に応じて文字色を指定することができます。たとえば、デフォルトでは未読のメッセージは赤で、それ以外のメッセージは黒で表示されるようになっています。どの条件でどの色を使用するかは、色の設定で行います。条件はマクロで指定しますが、よく使われる条件に関してはマクロを記述することなく指定することができます。

色の指定の詳細

色の指定は以下のような構造をしています。まず、QMAIL3内部では色セットのリストを持っています。色セットは、その色セットが使用されるアカウント・フォルダと、色エントリのリストを持っています。色エントリでは、そのエントリが使用される条件と、文字色・背景色・フォント（太字）の設定を持っています。

使用される色は以下のように決定されます。

1. まず、現在のアカウントとフォルダ名を使って、使用可能な全ての色セットの中から全ての色エントリをリストを作ります。
2. (1)で作った色エントリのリストの上から順に条件をチェックしていきます。条件にあう色エントリがあれば、そのエントリで指定されている文字色・背景色・フォント（太字）を取得します。
3. エントリによっては文字色のみ指定されていたり、文字色とフォント（太字）のみ指定されているので、文字色・背景色・フォント（太字）の全てが確定するまで次の色エントリを調べます。
4. 三つ全てが決まるか、最後の色エントリまで調べ終わったら終了します。
5. 最後まで調べても決まらなかった項目に関してはデフォルトの値（リストビューで指定された文字色・背景色と、通常のフォント）が使用されます。

@ComputerName

```
String @ComputerName()
```

説明

実行しているPCの名前を返します。取得できなかった場合には空文字列を返します。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# PC名を取得
@ComputerName()
```

@Concat

```
String @Concat(String arg*)
```

説明

argで指定された文字列を連結した文字列を返します。0個以上の任意の数の引数を渡せます。引数が指定されない場合には空文字列を返します。

引数

String arg
　　連結する文字列

エラー

なし

条件

なし

例

```
# FooとBarを連結
# -> FooBar
@Concat('Foo', 'Bar')
```

@Contain

```
Boolean @Contain(String s1, String s2, Boolean case?)
```

説明

s1にs2が含まれている場合にはTrue、それ以外の場合にはFalseを返します。caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

String s1
　　文字列
String s2
　　文字列
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# %SubjectにQsが含まれる（大文字と小文字を区別）
@Contain(%Subject, 'Qs', @True())

# Fromに@example.orgが含まれる
@Contain(From, '@example.org')
```

@Copy

```
Boolean @Copy(String folder)
```

説明

コンテキストメッセージを指定されたフォルダにコピーします。常にTrueを返します。

引数でフォルダを指定するときには、以下の形式のいずれかが使用できます。

フォルダの完全名
: フォルダの完全名を指定した場合には、現在選択されているアカウントの指定された名前のフォルダを意味します。例えば、「受信箱/テスト」のように指定すると、「受信箱」の子フォルダの「テスト」フォルダを意味します。

アカウント名+フォルダの完全名
: アカウントまで含めて指定する場合には、フォルダの完全名の前に「//アカウント名/」を置きます。例えば、「//Main/受信箱」と指定すると、現在選択されているアカウントに関係なく「Main」アカウントの「受信箱」を意味します。

指定されたフォルダが見つからない場合やコピーできない場合にはエラーになります。

引数

String folder
: フォルダ名

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- 指定されたフォルダが見つからない場合
- 指定されたフォルダにコピーできない場合

条件

- メッセージに対する変更が可能

例

```
# Testにコピー
@Copy('Test')
```

作成用のテンプレート

作成用のテンプレートはメッセージを作成するときに使用します。作成用のテンプレートを評価した結果はRFC2822で規定されている形式になっている必要があります。つまり、ヘッダ部と本文が空行で区切られ、ヘッダ部は名前と値が「:」で区切られた形式になっている必要があります。ただし、日本語などのASCII文字列以外をそのまま含むことができます。このとき、特殊なヘッダを使用することで、アカウントや署名などを指定することもできます。

たとえば、返信用のテンプレートを評価すると、Toに返信元のFromのアドレスがセットされ、本文に返信元の本文を引用されたメッセージになります。このメッセージをエディットウィンドウで開いて返信用のメッセージを作成します。

デフォルトでは以下の作成用のテンプレートが作成されます。

- new.template
- reply.template
- reply_all.template
- forward.template
- edit.template

これらはそれぞれ、[メッセージ]メニューの[新規], [返信], [全員に返信], [転送], [編集]を実行したときに使用されます。また、[メッセージ]-[テンプレート]メニューの下にはデフォルト以外テンプレートがリストされます。ここにリストするためには、テンプレートのファイル名を「create_」から始めます。たとえば、create_Test.templateというファイル名でテンプレートを作成すると、[メッセージ]-[テンプレート]-[Test]として表示されます。

また、メニューやツールバーをカスタマイズすることで、独自のテンプレートをメニューやツールバーの好きなところに割り当てることができます。詳細は、メニューのカスタマイズ、ツールバーのカスタマイズ、キーボードショートカットのカスタマイズと、MessageCreateアクションを参照してください。

ヘッダビューのカスタマイズ

header.xmlを編集することでヘッダビューをカスタマイズできます。

リストビューのカスタマイズ

リストビューにどのような情報を表示するかはカスタマイズすることができます。カスタマイズした設定は各フォルダごとに保存されますので、フォルダごとに表示する情報を指定することができます。リストビューをカスタマイズするには、[表示]–[カラムをカスタマイズ]を選択します。選択すると、[カラムのカスタマイズ]ダイアログが開きます。

メニューのカスタマイズ

menus.xmlを編集することでメニューをカスタマイズできます。

Windows Mobile 5.0以降では、トップレベルのポップアップメニューの数を2個以下にするとソフトキーが使用できます。

@Date

```
Time @Date(String date?)
```

説明

引数が指定されない場合には現在の時間を返します。指定された場合には、dateをRFC2822形式のDateの文字列としてパースし、その時間を返します。パースに失敗すると現在の時間を返します。

RFC2822形式の時間は、メールのヘッダで使われている形式で、以下のようなフォーマットになります。

```
Mon, 3 Apr 2006 15:23:05 +0900
```

引数

String date
　　RFC2822形式の時間

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 現在の時間を取得
@Date()

# コンテキストメッセージのDateヘッダの時間を取得
@Date(Date)
```

@Decode

```
String @Decode(String s)
```

説明

指定された文字列にASCII文字のみが含まれているときに、RFC2047に基づいてデコードします。非ASCII文字が含まれているときには何もしません。

たとえば、

```
@Decode('=?iso-2022-jp?B?GyRCJUYlOSVIGyhC?=')
```

は「テスト」を返します。

通常ヘッダから取得した文字列はデコードされているため、この関数を使う必要はありません。

引数

String s
　　文字列

エラー

- 引数の数が合っていない場合

条件

なし

例

@Defun

```
Boolean @Defun(String name, Expr expr)
```

説明

関数を定義します。既に同じ名前の関数が定義されていた場合にはFalseを、それ以外の場合にはTrueを返します。

二番目の引数は関数の定義時には評価されず、その関数が呼び出されたときに評価されます。二番目の引数中で関数に渡された引数にアクセスするには$<数字>を使用します。$0は関数名をあらわし、$1からは渡された引数をあらわします。渡されていない引数を参照するとエラーになります。

引数

String name
　　関数名
Expr expr
　　関数本体

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 掛け算する関数Multiplyを定義
@Defun('Multiply',
       @If(@Equal($1, 0),
           0,
           @Equal($1, 1),
           $2,
```

```
            @Add($2, @Multiply(@Subtract($1, 1), $2))))
```

@DeleteAttachment

```
Boolean @DeleteAttachment()
```

説明

コンテキストメッセージの添付ファイルを削除します。常にTrueを返します。

添付ファイルを削除すると、コンテキストメッセージが存在しなくなります。このため、以降のマクロでコンテキストメッセージが必要な関数などを呼び出すとエラーになります。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- 添付ファイルが削除できない場合

条件

- メッセージに対する変更が可能

例

```
# 添付ファイルを削除
@DeleteAttachment()
```

@Deleted

```
Boolean @Deleted(Boolean deleted?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが削除済みの場合にはTrue、それ以外の場合にはFalseを返します。deletedにTrueを指定されて呼び出された場合にはコンテキストメッセージを削除済みにし、Falseを指定された場合には削除済みではなくします。フラグを参照してください。

引数

Boolean deleted
　削除済みにするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 削除済み、または削除済みではなくするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 削除済みかどうか調べる
@Deleted()

# 削除済みにする
@Deleted(@True())
```

@Delete

```
Boolean @Delete(Boolean direct?)
```

説明

コンテキストメッセージを削除します。常にTrueを返します。

directにTrueを指定するとゴミ箱に入らず直接削除されます。Falseを指定した場合、または指定しなかった場合には、削除されたメッセージがゴミ箱に移動されるかは、EditDeleteアクションと同様に動作します。

ゴミ箱に入れずに直接削除された場合、コンテキストメッセージが存在しなくなります。このため、以降のマクロでコンテキストメッセージが必要な関数などを呼び出すとエラーになります。

引数

Boolean direct
　　ゴミ箱を使わずに直接削除するかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- 削除できない場合

条件

- メッセージに対する変更が可能

例

```
# メッセージを削除
@Delete()
```

```
# メッセージを直接削除
@Delete(@True())
```

@Download

```
Boolean @Download(Boolean download?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージがダウンロード予約されている場合にはTrue、それ以外の場合にはFalseを返します。downloadにTrueを指定されて呼び出された場合にはコンテキストメッセージをダウンロード予約し、Falseを指定された場合にはダウンロード予約をキャンセルします。フラグを参照してください。

引数

Boolean download
　　ダウンロード予約するかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- ダウンロード予約、またはキャンセルに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ダウンロード予約されているかどうか調べる
@Download()

# ダウンロード予約する
@Download(@True())
```

@DownloadText

```
Boolean @DownloadText(Boolean download?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが本文ダウンロード予約されている場合にはTrue、それ以外の場合にはFalseを返します。downloadにTrueを指定されて呼び出された場合にはコンテキストメッセージを本文ダウンロード予約し、Falseを指定された場合には本文ダウンロード予約をキャンセルします。フラグを参照してください。

引数

Boolean download
　　本文ダウンロード予約するかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 本文ダウンロード予約、またはキャンセルに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 本文ダウンロード予約されているかどうか調べる
@DownloadText()

# 本文ダウンロード予約する
@DownloadText(@True())
```

@Draft

```
Boolean @Draft(Boolean draft?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが草稿の場合にはTrue、それ以外の場合にはFalseを返します。draftにTrueを指定されて呼び出された場合にはコンテキストメッセージを草稿にし、Falseを指定された場合には草稿ではなくします。フラグを参照してください。

引数

Boolean draft
　　草稿にするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 草稿、または草稿ではなくするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 草稿かどうか調べる
@Draft()

# 草稿ではなくする
@Draft(@False())
```

@Equal

```
Boolean @Equal(Value v1, Value v2, Boolean case?)
```

説明

v1とv2が等しい場合にはTrue、それ以外の場合にはFalseを返します。v1とv2がともに真偽値の場合には真偽値として、v1とv2がともに数値の場合には数値として、それ以外の場合には文字列として比較します。文字列として比較する場合には、caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

Value v1
: 値

Value v2
: 文字列

Boolean case
: 大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# List-Idが<ml@example.org>
@Equal(List-Id, '<ml@example.org>')

# IDが1234
@Equal(@Id(), 1234)
```

@Eval

```
Value @Eval(String expr)
```

説明

指定されたマクロを実行します。

引数

String expr
　　実行するマクロ

エラー

- 引数の数が合っていない場合
- マクロとして不正な場合

条件

なし

例

```
# 文字列で指定されたマクロを実行
@Eval('@If(Reply-To, Reply-To, From)')
```

@Execute

```
String @Execute(String command, String input?)
```

説明

commandで指定されたコマンドを実行します。

commandが関連付けられたファイルだった場合には関連付けでファイルを開きます。

inputが指定された場合には、指定された文字列をシステムのエンコーディングでバイト列に変換した結果をコマンドの標準入力に渡します。この場合、標準出力から出力されたバイト列をシステムのエンコーディングで文字列に変換した結果を返します。inputの指定はWindows版でのみ可能です。

inputが指定されなかった場合には、コマンドを実行して空文字列を返します。

引数

String command
　　実行するコマンド
String input
　　コマンドの標準入力に渡す文字列

エラー

- 引数の数が合っていない場合
- コマンドの実行に失敗した場合

条件

なし

例

```
# メモ帳を起動
@Execute('notepad.exe')

# 空白文字を含むときには""で括る
@Execute('"C:\\Program Files\\Test\\test.exe"')

# 引数を渡す
@Execute('notepad.exe "C:\\Temp\\test.txt"')

# 関連付け
@Execute('C:/Temp/Test.doc')

# フィルタ
@Execute('sed.exe -e "s/foo/bar/"', 'foo')
```

@Exist

```
Boolean @Exist(String name)
```

説明

ヘッダに指定された名前のフィールドがあるかどうかを返します。

フィールドをBooleanに変換するとフィールドの存在が調べられるので以下の三つは同じ意味になります。

```
@If(@Exist('X-ML-Name'), 1, 2)
@If(X-ML-Name, 1, 2)
@If(@Field('X-ML-Name'), 1, 2)
```

引数

String name
　　フィールド名

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合

条件

なし

例

```
# X-ML-Nameがあるかどうか調べる
@Exist('X-ML-Name')
```

@Exit

```
Value @Exit()
```

説明

マクロの実行を中断します。返り値は不定になります。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 条件によって中断
@If(@Seen(), @Exit(), @True())
```

@False

```
Boolean @False()
```

説明

Falseを返します。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# False
# -> False
@False()
```

@Field

```
Field @Field(String name, Part part?)
```

説明

コンテキストメッセージの指定された名前のヘッダを返します。partが指定された場合にはそのパートの指定した名前のヘッダを返します。

引数

String name
　　ヘッダ名
Part part
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# Toを取得
@Field('To')

# マルチパートメッセージではじめのパートのContent-IDを取得
@Field('Content-ID', @Part(0))
```

@FieldParameter

```
String @FieldParameter(String name, String paramName?, Part part?)
```

説明

コンテキストメッセージの指定された名前のヘッダの指定された名前のパラメータの値を返します。partが指定された場合にはそのパートの指定した名前のヘッダの指定された名前のパラメータの値を返します。指定された名前のヘッダやパラメータが見つからない場合には空文字列を返します。指定された名前のヘッダが複数ある場合には先頭のヘッダが使用されます。

paramNameが空文字列か省略された場合には、パラメータでない部分を返します。たとえば、

```
Content-Type: text/plain; charset=iso-2022-jp
```

というようなヘッダの場合、@FieldParameter('Content-Type')は「text/plain」を、@FieldParameter('Content-Type', 'charset')は「iso-2022-jp」を返します。

引数

String name
　　ヘッダ名
String paramName
　　パラメータ名
Part part
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

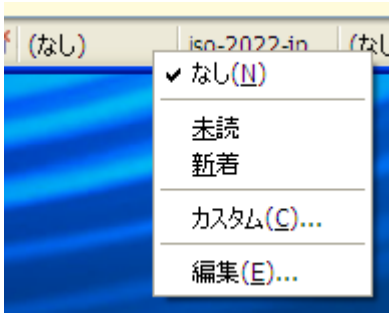
例

```
# Content-Typeの値を取得
@FieldParameter('Content-Type')

# Content-Typeのcharset値を取得
@FieldParameter('Content-Type', 'charset')

# マルチパートメッセージではじめのパートのContent-Dispositionのfilenameパラメータを取得
@Field('Content-Disposition', 'filename', @Part(0))
```

フィルタ

フィルタを使用すると、リストビューに表示するメッセージをフィルタすることができます。例えば、未読のメッセージのみ、マークされたメッセージのみを表示するなど、特定の条件にマッチするメッセージのみを表示することができます。フィルタはマクロを使用して任意の条件で作成することができます。

適用するフィルタは、[表示]-[フィルタ]で指定します。現在適用されているフィルタはステータスバーに表示されます。ここを右クリックすることでもフィルタを指定することができます。



[なし]を選択するとフィルタを使用しません。[未読], [新着]などを選択すると対応するフィルタが適用されます。[カスタム]を選択するとその場だけで使用するフィルタを作成することができます。[編集]を選択するとフィルタを編集することができます。

@FindEach

```
Value @FindEach(MessageList messages, Expr condition, Expr expr?)
```

説明

messagesで指定された各メッセージをコンテキストメッセージとして、conditionで指定された式を評価し、結果がTrueになったメッセージをコンテキストメッセージとしてexprを評価して返します。exprが指定されなかった場合には最後に評価してconditionの結果をそのまま返します。全てのメッセージに対してconditionを評価した値がFalseになった場合には、Falseを返します。

引数

MessageList messages
　　対象のメッセージリスト
Expr condition
　　条件式
Expr expr
　　評価する式

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合
- 式を評価中にエラーが発生した場合

条件

なし

例

```
# コンテキストメッセージが所属するスレッドの先頭のメッセージのMessage-Idを取得
@FindEach(@Thread(), @True(), Message-Id)
```

```
# コンテキストメッセージが所属するスレッドに自分が送信したメッセージがあるかどうかを調べる
@FindEach(@Thread(), @Equal(@Address(From), @Address(@I())))

# コンテキストメッセージが所属するスレッドの中で先頭の未読のメッセージのSubjectを取得
@FindEach(@Thread(), @Not(@Seen()), Subject)
```

@Find

```
Number @Find(String s1, String s2, Number index?, Boolean case?)
```

説明

s1の中で最初にs2が現れるインデックスを返します。indexが指定された場合には、指定された文字数目から検索を始めます。caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。インデックスは0ベースです。見つからなかった場合には-1を返します。

引数

String s1
　　文字列
String s2
　　検索する文字列
Number index
　　検索を始めるインデックス
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 文字を検索
# -> 2
@Find('abcabcabc', 'c')
```

```
# インデックスを指定
# -> 5
@Find('abcabcabc', 'Ca', 3)
```

@Flag

```
Boolean @Flag(Number flag, Boolean value?)
```

説明

valueが指定されていない場合には、コンテキストメッセージのフラグを取得します。指定されている場合には、フラグを設定します。

この関数の代わりに以下の関数を使用してください。

- @Seen
- @Replied
- @Forwarded
- @Sent
- @Draft
- @Marked
- @Deleted
- @Download
- @DownloadText
- @Multipart
- @Partial
- @User1
- @User2
- @User3
- @User4

引数

Number flag
: フラグ

Boolean value
: 設定する場合にフラグを立てるか倒すか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- コンテキストメッセージが一時的な場合（フラグを設定する場合）
- フラグの設定に失敗した場合（フラグを設定する場合）

条件

なし

例

```
# 0x00000001のフラグを取得
@Flag(1)
```

@FolderFlag

```
Boolean @FolderFlag(String folder, Number flag)
```

説明

folderで指定されたフォルダがflagで指定したフラグを持っているかどうかを返します。

folderにはフラグを調べたいフォルダの完全名を指定します。指定されたフォルダが見つからないとエラーになります。

flagには調べたいフラグを指定します。指定できるのは以下のいずれかです。

:FF-NOSELECT
: 選択可能かどうか。選択可能な場合にはFalseが、それ以外の場合にはTrueが返されます。IMAP4アカウントではメッセージを入れられないフォルダではこのフラグがTrueになります。

:FF-NOINFERIORS
: 子フォルダを持つことができるかどうか。子フォルダを持つことができる場合にはFalseが、それ以外の場合にはTrueが返されます。IMAP4アカウントでは子フォルダを作成することができないフォルダではこのフラグがTrueになります。

:FF-CUSTOMFLAGS
: カスタムフラグをサポートするかどうか。IMAP4アカウントではサーバ上に任意のフラグを保存できる場合にはTrueが、それ以外の場合にはFalseが返されます。

:FF-NORENAME
: 名前を変更することができるかどうか。名前を変更できる場合にはFalseが、それ以外の場合にはTrueが返されます。

:FF-LOCAL
: ローカルフォルダかどうか。ローカルフォルダの場合にはTrueが、それ以外の場合にはFalseが返されます。

:FF-HIDE
: 隠されているかどうか。隠されている場合にはTrueが、それ以外の場合にはFalseが返されます。

:FF-CHILDOFROOT
: IMAP4アカウントで指定されたルートフォルダの子フォルダの場合にはTrueが、それ以外の場合にはFalseが返されます。

:FF-IGNOREUNSEEN
: 未読を無視するかどうか。未読を無視する場合にはTrueが、それ以外の場合にはFalseが返されます。

:FF-INBOX
: 受信箱かどうか。

:FF-OUTBOX
: 送信箱かどうか。

:FF-SENTBOX
: 送信控えかどうか。

:FF-TRASHBOX

ゴミ箱かどうか。
:FF-DRAFTBOX
草稿箱かどうか。
:FF-SEARCHBOX
検索かどうか。
:FF-JUNKBOX
スパムかどうか。
:FF-SYNCABLE
同期可能かどうか。
:FF-SYNCWHENOPEN
開いたときに同期するかどうか。
:FF-CACHEWHENREAD
本文を読んだときにキャッシュするかどうか。

引数

String folder
フォルダ名
Number flag
調べるフラグ

エラー

- 引数の数が合っていない場合
- 指定されたフォルダが見つからない場合

条件

なし

例

```
# 選択されているメッセージが格納されているフォルダが受信箱かどうかを調べる
@FolderFlag(@Folder(), :FF-INBOX)

# 現在のフォルダが同期可能かどうか調べる
@FolderFlag(@Folder(:FN-FULLNAME, :FT-CURRENT), :FF-SYNCABLE)
```

@Folder

```
String @Folder(Boolean full?, Boolean current?)
```

説明

コンテキストメッセージが保存されているフォルダ、またはコンテキストフォルダの名前を返します。

fullには完全名を返すか単一名を返すかを指定します。指定できるのは以下のいずれかです。

:FN-FULLNAME
　　完全名を返します。
:FN-NAME
　　単一名を返します。

引数が省略された場合には:FN-FULLNAMEを指定したのと同じになります。

例えば、「受信箱」の下にある「テスト」フォルダにメッセージが保存されている場合には、完全名は「受信箱/テスト」になり、単一名は「テスト」になります。

currentにはフォルダ名の取得方法を指定します。指定できるのは以下のいずれかです。

:FT-MESSAGE
　　コンテキストメッセージが格納されているフォルダを返します。
:FT-CURRENT
　　コンテキストフォルダを返します。

引数が省略された場合には:FT-MESSAGEを指定したのと同じになります。

例えば、検索フォルダを開いていた場合、:FT-MESSAGEを指定するとメッセージが格納されているフォルダの名前が返され、:FT-CURRENTを指定すると検索フォルダの名前が返されます。

FT-MESSAGEを指定した場合にコンテキストメッセージホルダがない、または:FT-CURRENTを指定した場合にコンテキストフォルダがない（アカウントが選択されている）ときには空文字列を返します。

引数

Boolean full
　　完全名を取得するかどうか
Boolean current

フォルダ名の取得方法

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合

条件

なし

例

```
# フォルダ名を完全名で取得
@Folder()

# 単一名で取得
@Folder(:FN-NAME)

# 現在のフォルダ名を完全名で取得
@Folder(:FN-FULLNAME, :FT-CURRENT)
```

@FolderParameter

```
String @FolderParameter(String folder, String name)
```

説明

folderで指定されたフォルダのnameで指定されたパラメータの値を返します。

folderにはフラグを調べたいフォルダの完全名を指定します。指定されたフォルダが見つからないとエラーになります。nameには取得するパラメータの名前を指定します。指定されたパラメータが設定されていなかった場合には空文字列を返します。

引数

String folder
　　フォルダ名
String name
　　パラメータ名

エラー

- 引数の数が合っていない場合
- 指定されたフォルダが見つからない場合

条件

なし

例

```
# 現在のフォルダのToパラメータを取得
@FolderParameter(@Folder(:FN-FULLNAME, :FT-CURRENT), 'To')
```

フォント

各ビューのフォントはオプションの設定で個別に設定することができます。

フォントグループ

さらにメッセージビューとプレビューはメッセージに応じてフォントを切り替えることができます。例えば、英語のメッセージはTahomaで、メールマガジンは等幅のＭＳ ゴシックで、それ以外はＭＳ Ｐゴシックで、というような設定ができます。

この機能はUIからは設定することができませんので、手動で設定する必要があります。まず、fonts.xmlでフォントグループを定義します。そして、qmail.xmlでメッセージビュー、またはプレビューで使用するフォントグループを指定します。指定するのはそれぞれMessageWindow/FontGroupと、PreviewWindow/FontGroupです。

例えば、Content-Typeのcharsetがus-asciiまたはiso-8859-xの場合にはTahomaで、「メルマガ」フォルダのメッセージはＭＳ ゴシックで、それ以外はＭＳ Ｐゴシックで表示するには以下のようなfonts.xmlを作成します。

```
<?xml version="1.0" encoding="utf-8"?>
<fonts>
 <group name="main">
  <fontSet match="@Progn(@Set('charset', @BodyCharset()),
                         @Or(@BeginWith($charset, 'iso-8859-'),
                             @Equal($charset, 'us-ascii')))">
   <font face="Tahoma" size="9"/>
  </fontSet>
  <fontSet match="@Equal(@Folder(), 'メルマガ')">
   <font face="ＭＳ ゴシック" size="9"/>
  </fontSet>
  <fontSet>
   <font face="ＭＳ Ｐゴシック" size="9"/>
  </fontSet>
 </group>
</fonts>
```

そして、qmail.xmlのPreviewWindow/FontGroupにmainと指定します。また、ViewFontGroupアクションを使用してフォントグループを切り替えることもできます。

フォントグループの定義の仕方の詳細は、fonts.xmlを参照してください。

@ForEach

```
Boolean @ForEach(MessageList messages, Expr expr)
```

説明

messagesで指定された各メッセージをコンテキストメッセージとして、exprで指定された式を繰り返し評価します。常にTrueを返します。

引数

MessageList messages
　　対象のメッセージリスト
Expr expr
　　評価する式

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合
- 式を評価中にエラーが発生した場合

条件

なし

例

```
# 受信箱内のすべてのメッセージを既読にする
@ForEach(@Messages('受信箱'), @Seen(@True()))

# コンテキストメッセージが属するスレッドのメッセージ全てをマークする
@ForEach(@Thread(), @Marked(@True()))
```

@FormatAddress

```
String @FormatAddress(Field field, Number type?, Number lookup?)
```

説明

引数で指定されたフィールドをアドレスリストとしてパースし、その結果をフォーマットした文字列を返します。パースに失敗した場合には空文字列を返します。

typeに指定した値で各アドレスのフォーマット方法が代わります。以下のいずれかが指定できます。

:FORMAT-ALL
　　名前 <アドレス>
:FORMAT-ADDRESS
　　アドレス
:FORMAT-NAME
　　名前（名前が指定されたいない場合にはアドレス）
:FORMAT-VIEW
　　名前 <アドレス>（ただし、名前部分をエスケープしない）

指定しなかった場合には:FORMAT-ALLを指定したのと同じになります。:FORMAT-ALLと:FORMAT-VIEWは似ていますが、必要なエスケープをするかどうかが異なります。たとえば、

```
To: "Test (Test)" <test@example.org>, "Yamada, Taro" <test2@example.org>
```

というヘッダを持つメッセージに、@FormatAddress(To, :FORMAT-ALL)を適用すると「"Test (Test)" <test@example.org>, "Yamada, Taro" <test2@example.org>」が返されますが、@FormatAddress(To, :FORMAT-VIEW)を適用すると「Test (Test) <test@example.org>, Yamada, Taro <test2@example.org>」が返されます。このように、:FORMAT-VIEWを指定すると見た目重視でフォーマットされるため、その文字列をパースすることができなくなる可能性があります。

lookupに指定した値で使用する名前をアドレス帳から逆引きするかどうかを指定します。以下のいずれかが指定できます。

:LOOKUP-NONE
　　逆引きをしません
:LOOKUP-EMPTY
　　名前が指定されていない場合のみ逆引きをします
:LOOKUP-FORCE
　　常に逆引きをします

逆引きをした場合には、フォーマット時に名前の部分をアドレス帳から逆引きした名前で置き換えます。アドレス帳に逆引きしたアドレスが含まれていない場合には置き換えは行

われません。また、同じアドレスが複数のエントリに表れる場合には始めに見つかったエントリが使われます。lookupを指定しなかった場合には:LOOKUP-NONEを指定したのと同じになります。

引数

Field field
　　フォーマットするフィールド
Number type
　　フォーマットの形式
Number lookup
　　アドレス帳の逆引き方法

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合

条件

なし

例

```
# Fromを表示用に逆引きしてフォーマット
@FormatAddress(From, :FORMAT-VIEW, :LOOKUP-FORCE)

# Toをフォーマット
@FormatAddress(To)
```

@FormatDate

```
String @FormatDate(Time date, String format, Number timezone?)
```

説明

dateで指定された時間をformatで指定された書式でフォーマットします。

書式には任意の文字列を指定できますが、%に続く文字列が時間の一部に置き換えられます。

%Y2
: 二桁の年（例:06）

%Y4
: 四桁の年（例:2006）

%M0
: 二桁の月（例:05）

%M1
: 3文字の月の名前（例:Mar）

%M2
: 長い月の名前（例:January）

%M3
: ユーザのロケールでローカライズされた月の名前

%M4
: ユーザのロケールでローカライズされた長い月の名前

%D
: 二桁の日（例:29）

%W
: 3文字の曜日（例:Mon）（この指定は古い形式です。代わりに%W0を使用してください）

%W0
: 3文字の曜日（例:Mon）

%W1
: 長い曜日（例:Monday）

%W2
: ユーザのロケールでローカライズされた曜日名

%W3
: ユーザのロケールでローカライズされた長い曜日名

%h
: 24時間制での二桁の時（例:19）

%m
: 二桁の分（例:34）

%s
: 二桁の秒（例:02）

%z
: +か-の後に4桁のタイムゾーン（例:+0900）

%Z
　　+か-の後に:で区切られた4桁のタイムゾーン（例:-07:00）
%%
　　%自体

たとえば、「%Y4/%M0/%D %h:%m:%s」を指定すると、「2006/05/07 21:56:32」のようにフォーマットされます。

timezoneには以下の定数を指定できます。

:TZ-UTC
　　UTCでフォーマット
:TZ-LOCAL
　　ローカルのタイムゾーンでフォーマット
:TZ-ORIGINAL
　　オリジナルのタイムゾーンでフォーマット

指定しない場合には、:TZ-LOCALを指定したのと同じになります。

引数

Time date
　　時間
String format
　　書式
Number timezone
　　どのタイムゾーンでフォーマットするか

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合
- formatの文字列が不正な場合
- timezoneの値が不正な場合

条件

なし

例

```
# Dateヘッダの時間をローカルタイムゾーンでフォーマット
@FormatDate(@Date(Date), '%Y4/%M0/%D %h:%m:%s')
```

```
# 現在の時間をUTCでフォーマット
@FormatDate(@Date(), '%Y4/%M0/%d', :TZ-UTC)
```

@Forwarded

```
Boolean @Forwarded(Boolean forwarded?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが転送済みの場合にはTrue、それ以外の場合にはFalseを返します。forwardedにTrueを指定されて呼び出された場合にはコンテキストメッセージを転送済みにし、Falseを指定された場合には転送済みではなくします。フラグを参照してください。

引数

Boolean forwarded
　　転送済みにするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 転送済み、または転送済みではなくするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 転送済みかどうか調べる
@Forwarded()

# 転送済みにする
@Forwarded(@True())
```

@Greater

```
Boolean @Greater(Value v1, Value v2, Boolean case?)
```

説明

v1がv2よりも大きい場合にはTrue、それ以外の場合にはFalseを返します。v1とv2がともに真偽値の場合には真偽値として、v1とv2がともに数値の場合には数値として、それ以外の場合には文字列として比較します。文字列として比較する場合には、caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

Value v1
: 値

Value v2
: 文字列

Boolean case
: 大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# サイズが100KBより大きい
@Greater(@Size(), 102400)

# aはBよりも大きい（大文字と小文字を区別）
# -> True
```

```
@Greater('a', 'B', @True())
```

@Header

```
String @Header(String remove?, Part part?)
```

説明

コンテキストメッセージのヘッダを返します。partが指定された場合にはそのパートのヘッダを返します。

removeには返されるヘッダから除外したいヘッダの名前を「,」区切りで指定します。

引数

String remove
　　除外するヘッダのリスト
Part part
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# ヘッダを全て取得
@Header()

# ToとMessage-Idを除外してヘッダを取得
@Header('To,Message-Id')
```

```
# マルチパートメッセージではじめのパートのヘッダを取得
@Header('', @Part(0))
```

@HtmlEscape

```
String @HtmlEscape(String s)
```

説明

sをHTMLとしてエスケープした文字列を返します。

以下の置き換えを行います。

- <を<
- >を>
- &を&
- "を"

引数

String s
　　エスケープする文字列

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 本文をHTMLとしてエスケープ
@HtmlEscape(@Body())
```

@Identity

```
String @Identity()
```

説明

コンテキストアカウントの現在のサブアカウントの同一性値を返します。指定されていない場合には空文字列を返します。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストアカウントがない場合

条件

なし

例

```
# 同一性値を取得
@Identity()
```

@Id

```
Number @Id()
```

説明

コンテキストメッセージのIDを返します。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合

条件

なし

例

```
# IDを取得
@Id()
```

@If

```
Value @If(Boolean condition, Value value, Value otherwise)
```

説明

conditionを評価してTrueになったときにはvalueを返します。Falseになったときにはotherwiseを返します。3個以上の任意の奇数個の引数を渡せます。この場合、1番目の引数がTrueになった場合には2番目の引数を、3番目の引数がTrueになった場合には4番目の引数を、というように評価していき、どの条件にも合致しなかった場合には最後の引数を返します。

引数

Boolean condition
　　条件
Value value
　　値
Value otherwise
　　どの条件にも合致しなかったときの値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# Toがexample.orgを含んでいたら1、それ以外なら0
@If(@Contain(To, 'example.org'), 1, 0)

# Toがexample.orgを含んでいたら1、exampleを含んでいたら2、それ以外なら0
@If(@Contain(To, 'example.org'), 1,
```

```
    @Contain(To, 'example'), 2, 0)
```

@I

```
Field @I(String account?, String subaccount?)
```

説明

自分をあらわす仮想的なヘッダを返します。このヘッダはアカウントで指定した名前とメールアドレスを使用して、

```
I: 名前 <メールアドレス>
```

というようなヘッダがあると仮定して取得されたヘッダです。

accountやsubaccountが指定されると指定されたアカウントやサブアカウントの自分の名前やアドレスを取得します。指定されない場合には、コンテキストアカウントと現在のサブアカウントを使用します。

引数

String account
　　アカウント名
String subaccount
　　サブアカウント名

エラー

- 引数の数が合っていない場合
- コンテキストアカウントがない場合（アカウントを指定しなかった場合）
- 指定されたアカウントが見つからない場合（アカウントを指定した場合）
- UIスレッド以外から呼び出した場合（アカウントだけ指定した場合）

条件

- UIスレッドからのみ呼び出し可能（アカウントだけ指定した場合）

例

```
# 自分の名前とアドレスからヘッダを取得
@I()

# X-QMAIL-SubAccountで指定されたサブアカウントのアドレスを取得
@I(@Account(), X-QMAIL-SubAccount)
```

@Include

```
Value @Include(String path)
```

説明

指定されたパスのファイルから読み込んだマクロを評価して結果を返します。つまり、@Eval(@Load(path))と同じです。パスに相対パスが指定された場合にはメールボックスディレクトリからの相対パスになります。ファイルはシステムのエンコーディングでエンコードされている必要があります。

引数

String path
　　ファイルパス

エラー

- 引数の数が合っていない場合
- 読み込みに失敗した場合
- マクロのパース・実行に失敗した場合

条件

なし

例

```
# 外部ファイルにあるマクロを実行
@Include('test.mac');
```

@InputBox

```
String @InputBox(String message, Boolean multiline?, String default?)
```

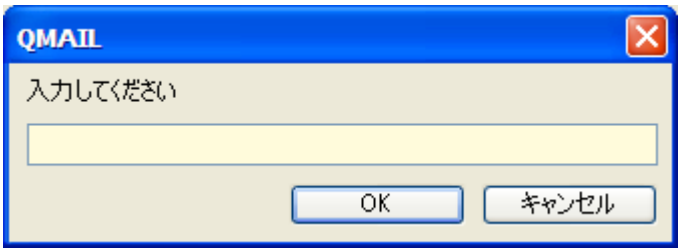
説明

ユーザに入力を求めるためのダイアログを表示し、入力された文字列を返します。messageには表示するメッセージを指定します。defaultを指定するとダイアログを開いたときにその文字列が入力欄に表示されます。

multilineにはダイアログのタイプを指定します。指定できるのは以下のとおりです。

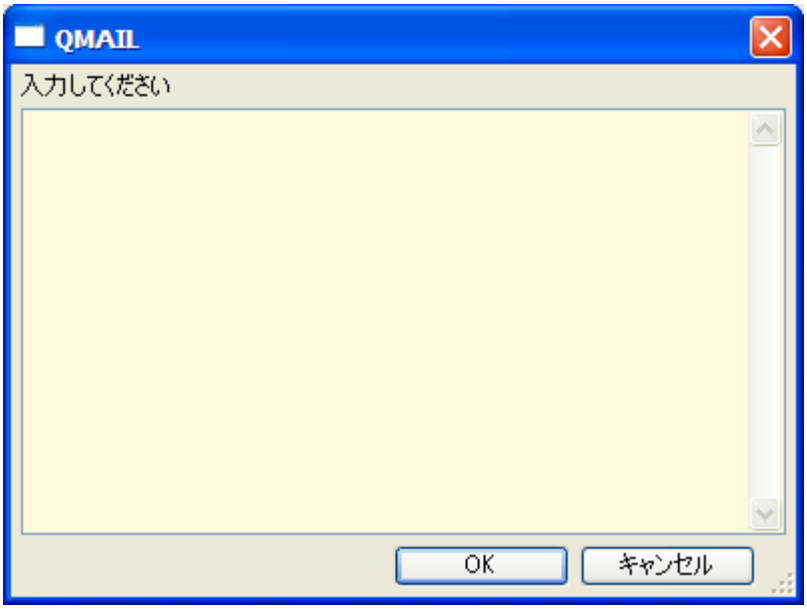
:INPUT-SINGLELINE

単数行の入力を求めるダイアログ



:INPUT-MULTILINE

複数行の入力を求めるダイアログ



指定しない場合には、:INPUT-SINGLELINEを指定したのと同じになります。

引数

String message
　　表示するメッセージ
Boolean multiline
　　ダイアログのタイプ
String default
　　デフォルトの入力文字列

エラー

- 引数の数が合っていない場合
- UIがない場合

条件

- UIが必要

例

```
# 単数行
@InputBox('入力してください')

# 単数行でデフォルトを指定
@InputBox('入力してください', :INPUT-SINGLELINE, 'テスト')

# 複数行
@InputBox('入力してください', :INPUT-MULTILINE)
```

@InvokeAction

```
Boolean @InvokeAction(String name, String arg+)
```

説明

nameで指定されたアクションを起動します。それ以降の引数はアクションの引数としてアクションに渡されます。常にTrueを返します。

この関数は任意のアクションを起動することができるため、使用するには注意が必要です。たとえば、EditDeleteアクションを起動して処理中のメッセージを削除するとクラッシュすることがあります。

引数

String name
　　アクションの名前
arg
　　アクションに渡す引数

エラー

- 引数の数が合っていない場合
- UIスレッド以外から呼び出した場合
- UIがない場合
- アクションを起動できない場合

条件

- UIスレッドからのみ呼び出し可能
- UIが必要

例

```
# MessageCreateアクションでnew.templateを使って新規メッセージを作成する
```

```
@InvokeAction('MessageCreate', 'new')
```

@Junk

```
Boolean @Junk(Boolean white?, Boolean black?)
```

説明

スパムフィルタを使用して、スパムかどうかを判定して返します。

whiteが指定された場合、その値を評価してTrueになったらクリーンだと判定します。blackが指定された場合、その値を評価してTrueになったらスパムだと判定します。ただし、whiteを評価した値がTrueになった場合には、blackは評価されません。whiteもblackもFalseに評価された場合（指定されなかった場合も含む）にはスパムフィルタを使用してスパムかどうか判定します。

whiteやblackを評価してスパムかどうか判定した場合でも、スパムフィルタの設定で[フィルタした結果を学習する]にチェックが入っている場合には、判定結果を使用してスパムフィルタに学習させます。

引数

Boolean white
　　クリーンだと判断される条件
Boolean black
　　スパムだと判断される条件

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合

条件

なし

例

```
# スパムかどうか判定する
@Junk()

# スパムかどうか判定する（ただし、既読のメッセージはすべてクリーン）
@Junk(@Seen())
```

@Label

```
String @Label(String label?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージのラベルを返します。コンテキストメッセージホルダがない場合と、ラベルが指定されていない場合には空文字列を返します。

labelが指定された場合には、指定された文字列をコンテキストメッセージのラベルとして設定し、そのラベルを返します。

ラベルを参照してください。

引数

String label
　　設定するラベル

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合（引数付きで呼び出された場合）
- コンテキストメッセージホルダが一時的な場合（引数付きで呼び出された場合）
- ラベルを設定するのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ラベルを取得する
@Label()
```

```
# ラベルを設定する
@Label('qmail')
```

@Length

```
Number @Length(String s, Boolean byte?)
```

説明

指定された文字列の長さを返します。byteにTrueを指定すると、プラットフォームのエンコーディング[*1]に変換したときのバイト数を返します。

引数

String s
　　文字列
Boolean byte
　　バイト列の長さを返すかどうか

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 文字列の長さを取得
# -> 3
@Length('abc')

# バイト数を取得
# -> 6（日本語環境の場合）
@Length('日本語', @True())
```

[*1]日本語プラットフォームの場合にはCP932

@Less

```
Boolean @Less(Value v1, Value v2, Boolean case?)
```

説明

v1がv2よりも小さい場合にはTrue、それ以外の場合にはFalseを返します。v1とv2がともに真偽値の場合には真偽値として、v1とv2がともに数値の場合には数値として、それ以外の場合には文字列として比較します。文字列として比較する場合には、caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

Value v1
　　値
Value v2
　　文字列
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# サイズが10KBより小さい
@Less(@Size(), 10240)

# abcはdよりも小さい
# -> True
```

```
@Less('abc', 'd')
```

@Load

```
String @Load(String path, Boolean template?, String encoding?)
```

説明

指定されたパスのファイルから読み込んだ内容を返します。パスに相対パスが指定された場合にはメールボックスディレクトリからの相対パスになります。

templateにTrueを指定すると読み込んだ内容をテンプレートとして処理し、処理結果を返します。

encodingを指定すると指定されたエンコーディングで読み込みます。指定されない場合にはシステムのデフォルトのエンコーディングで読み込みます。

引数

String path
　　ファイルパス
Boolean template
　　テンプレートとして読み込むかどうか
String encoding
　　エンコーディング

エラー

- 引数の数が合っていない場合
- 読み込みに失敗した場合
- テンプレートの処理に失敗した場合（テンプレートとして読み込んだ場合）

条件

なし

例

```
# ファイルを読み込む
@Load('C:\\Temp\\Test.txt')

# 相対パスでエンコーディングを指定
@Load('profiles/qmail.xml', @False(), 'utf-8')

# テンプレートを処理する
@Load('templates/mail/test.template', @True())
```

@LookupAddressBook

```
String @LookupAddressBook(String address, Number type)
```

説明

指定されたアドレスをアドレス帳から逆引きします。

typeには逆引きする値を指定します。以下のいずれかが指定できます。

:ADDRESS-NAME
　　名前
:ADDRESS-SORTKEY
　　ソートキー
:ADDRESS-ADDRESS
　　アドレス
:ADDRESS-ALIAS
　　エイリアス
:ADDRESS-CATEGORY
　　カテゴリ
:ADDRESS-COMMENT
　　コメント

指定しなかった場合には、:ADDRESS-NAMEを指定したのと同じになります。指定したアドレスのエントリが見つからなかった場合には空文字列を返します。

ADDRESS-ADDRESSは指定したアドレスと同じ値を返します（大文字・小文字は変わる可能性があります）。このため、:ADDRESS-ADDRESSを指定して逆引きして空文字列かどうか調べることでアドレス帳にエントリがあるかどうかを調べることができます。

引数

Number ltype
　　逆引きタイプ

エラー

- 引数の数が合っていない場合

条件

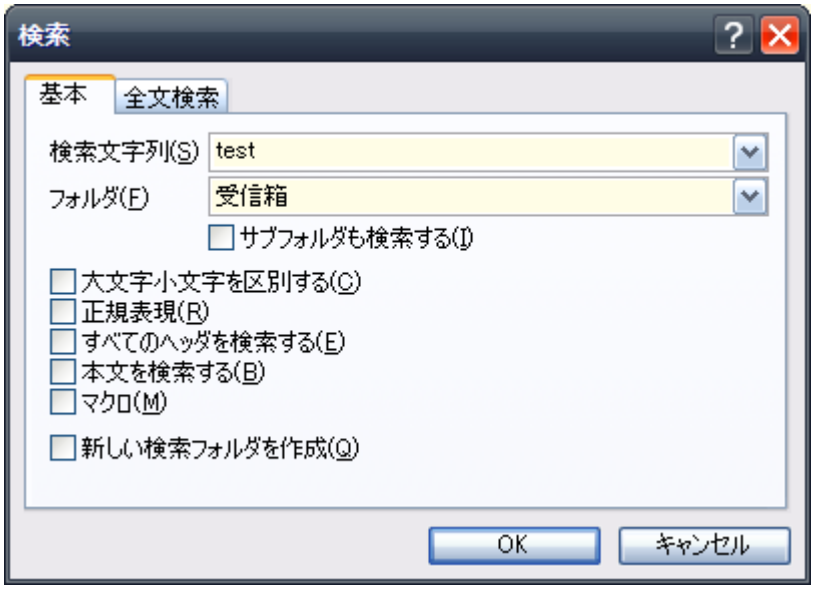
なし

例

```
# 名前を逆引き
@LookupAddressBook($address)

# コメントを逆引き
@LookupAddressBook($address, :ADDRESS-COMMENT)

# アドレス帳にエントリがあるかどうかを調べる
@If(@LookupAddressBook($address, :ADDRESS-ADDRESS),
    'アドレス帳にエントリあり',
    'アドレス帳にエントリなし')
```

基本検索

基本検索ではマクロを使用した検索を行います。



[検索文字列]

検索文字列を指定します。

[フォルダ]

検索対象のフォルダを指定します。「(すべてのフォルダ)」を選択すると全てのフォルダから検索します。

[サブフォルダも検索]

チェックすると[フォルダ]で指定したフォルダの全てのサブフォルダも検索対象になります。

[大文字小文字を区別する]

検索時に大文字と小文字を区別するかどうかを指定します。

[正規表現]

検索文字列が正規表現かどうかを指定します。

[すべてのヘッダを検索する]

すべてのヘッダを検索するかどうかを指定します。

[本文を検索する]

本文を検索するかどうかを指定します。

[マクロ]

検索文字列で指定された文字列をマクロとして処理し、そのマクロにマッチするメッセージを検索します。

[新しい検索フォルダを作成する]

通常、指定した条件がデフォルトの検索フォルダに設定され、その検索フォルダを開くことで検索結果を表示します。このチェックボックスにチェックを入れると、指定した条件で新たに検索フォルダを作成し、そのフォルダを開きます。検索フォルダについては、フォルダを参照してください。

マクロの生成方法

基本検索では指定された条件からマクロを生成し、そのマクロにマッチするメッセージを検索します。生成されるマクロは以下のように決定されます。

まず、[マクロ]にチェックを入れた場合には、[検索文字列]がそのままマクロになります。それ以外の場合、検索の設定の[マクロ]で指定したマクロを使用します。デフォルトでは、「@Or(@F(%Subject, $Search, $Case), @F(%From, $Search, $Case), @F(%To, $Search, $Case), @F(@Label(), $Search, $Case))」です。

$Searchは[検索文字列]で指定された文字列を以下のように扱った値になります。まず、[正規表現]がチェックされていない場合には、マクロの文字列リテラルになります。チェックされた場合には、正規表現リテラルになり[大文字小文字を区別する]にチェックが入っていない場合には、iオプションが指定されます。たとえば入力された文字が「foo*」だった場合には、

[正規表現]にチェックなし
　　"foo*"
[正規表現]にチェックあり

　　[大文字小文字を区別する]にチェックなし
　　　　/foo*/
　　[大文字小文字を区別する]にチェックあり
　　　　/foo*/i

のようになります。

$Caseは[大文字小文字を区別する]にチェックを入れた場合には真に、それ以外の場合には偽になります。また、$Regexは[正規表現]にチェックを入れた場合には真に、それ以

外の場合には偽になります。

@Fは[正規表現]にチェックを入れた場合には@Defun('F', @RegexMatch($1, $2))、入れない場合には@Defun('F', @Contain($1, $2, $3)のように定義される関数です。

ここで指定されたマクロを@X()と考えると、検索時のマクロは以下のようになります。

```
@Progn(@Set('Search', <検索文字列>),
       @Set('Case', <大文字と小文字を区別する>),
       @Set('Regex', <正規表現>),
       @Defun('F', <検索関数>),
       @X())
```

さらに、[すべてのヘッダを検索する]や[本文を検索する]にチェックを入れた場合には、X()の代わりに以下のようなマクロが使われます。

- [すべてのヘッダを検索する]にチェックを入れた場合

  ```
  @Or(@X(), @F(@Decode(@Header()), $Search, $Case))
  ```

- [本文を検索する]にチェックを入れた場合

  ```
  @Or(@X(), @F(@Body(), $Search, $Case))
  ```

- 上記の両方の場合

  ```
  @Or(@X(),
      @F(@Decode(@Header()), $Search, $Case),
      @F(@Body(), $Search, $Case))
  ```

@Marked

```
Boolean @Marked(Boolean marked?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージがマークされている場合にはTrue、それ以外の場合にはFalseを返します。markedにTrueを指定されて呼び出された場合にはコンテキストメッセージにマークを付け、Falseを指定された場合にはマークをはずします。フラグを参照してください。

引数

Boolean marked
　　マークを付けるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- マークを付けたりはずしたりするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# マークされているかどうか調べる
@Marked()

# マークを付ける
@Marked(@True())
```

@MessageBox

```
Number @MessageBox(String message, Number type?)
```

説明

messageに指定されたメッセージを表示するメッセージボックスを表示します。typeにはメッセージボックスのタイプを指定します。メッセージボックスのタイプは、MessageBox APIのuTypeに指定できる値を指定できます。指定されなかった場合には、MB_OK | MB_ICONINFORMATIONが指定されたのと同じになります。メッセージボックスを表示した後で、MessageBox APIの返り値を返します。

引数

String message
　　メッセージ
Number type?
　　タイプ

エラー

- 引数の数が合っていない場合
- UIがない場合

条件

- UIが必要

例

```
# メッセージボックスを表示
@MessageBox('Test')

# メッセージボックスに本文を表示
@MessageBox(@Body(:BODY-INLINE))

# Yes/Noを尋ねるメッセージボックスを表示して処理を分ける
```

```
@If(@Equal(@MessageBox('処理を続けますか?', 292), 6),
    @MessageBox('ごにょごにょ'),
    @Exit())
```

@MessageCount

```
Number @MessageCount(String folder?)
```

説明

メッセージ数を返します。folderが指定された場合にはそのフォルダ内のメッセージ数を返します。指定されなかった場合にはアカウント中のすべてのメッセージ数を返します。

引数

String folder
　　フォルダの完全名

エラー

- 引数の数が合っていない場合
- 指定されたフォルダが存在しない場合

条件

なし

例

```
# アカウント中のメッセージ数を取得する
@MessageCount()

# カレントフォルダのメッセージ数を取得する
@MessageCount(@Folder(:FN-FULLNAME, :FT-CURRENT))
```

@MessageId

```
String @MessageId()
```

説明

コンテキストメッセージのMessage-Idを返します。存在しない場合やパースに失敗した場合には空文字列を返します。`Message-Id`を使用すると、コンテキストメッセージのMessage-Idが不正な場合でも文字列として取得できてしまいますが、`@MessageId()`を使用すれば、そのような場合には空文字列になります。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# Message-Idを取得
@MessageId()
```

MessageMacroアクション

引数で指定されたマクロを実行します。引数を指定しなかった場合には、[マクロの実行]ダイアログを開くので、実行するマクロを指定します。

リストビューにフォーカスがある場合には、選択されている各メッセージをコンテキストメッセージとしてマクロが評価されます。つまり、選択されているメッセージの回数だけマクロが実行されます。選択されているメッセージがない場合には何もしません。

プレビューやメッセージビューにフォーカスがある場合には、表示しているメッセージをコンテキストメッセージとして一回だけマクロが評価されます。メッセージが表示されていない場合には何もしません。

フォルダビューやフォルダリストビュー、フォルダコンボボックスにフォーカスがあるときには、選択されているフォルダ内にある各メッセージをコンテキストメッセージとしてマクロが評価されます。つまり、選択されているフォルダ内のメッセージ数の回数だけマクロが実行されます。特に、フォルダリストビューで複数のフォルダを選択している場合には、選択されているフォルダ内の全てのメッセージの回数だけマクロが実行されます。

マクロを@Exitで終了したり、@InputBoxで表示した入力ダイアログでキャンセルしたり、エラーが発生した場合にはその時点でマクロの実行を中断します。

これを利用して、選択されている、またはフォルダ内にあるメッセージ数に関わらず一回だけマクロを実行したい場合には以下のようにマクロを指定します。

```
@Progn(<ここに実行したいマクロを指定する>,
       @Exit())
```

引数

1
: 実行するマクロ

有効なウィンドウ・ビュー

- メインウィンドウ
- メッセージウィンドウ

@Messages

```
MessageList @Messages(String folder?, Number id?)
```

説明

メッセージのリストを返します。folderが指定された場合にはそのフォルダ内のメッセージをすべて返します。さらにidが指定された場合にはそのIDのメッセージを返します。何も指定されなかった場合にはアカウント中のすべてのメッセージのリストを返します。

返されたメッセージリストは@ForEachや@FindEachなどを使用して処理することができます。

引数

String folder
　　フォルダの完全名
Number id
　　メッセージのID

エラー

- 引数の数が合っていない場合
- 指定されたフォルダが存在しない場合
- コンテキストアカウントがない場合（フォルダを指定しなかったか、フォルダの指定にアカウント名が含まれなかった場合）
- 指定されたフォルダが通常フォルダではない場合（idが指定された場合）
- UIスレッド以外から呼び出した場合（指定したフォルダのアカウントがコンテキストアカウントと異なる場合）

条件

- UIスレッドからのみ呼び出し可能（指定したフォルダのアカウントがコンテキストアカウントと異なる場合）

例

```
# 受信箱のすべてのメッセージを既読にする
@ForEach(@Messages('受信箱'), @Seen(@True()))

# 受信箱のIDが1000のメッセージを削除する
@ForEach(@Messages('受信箱', 1000), @Delete())
```

@Move

```
Boolean @Move(String folder)
```

説明

コンテキストメッセージを指定されたフォルダに移動します。常にTrueを返します。

引数でフォルダを指定するときには、以下の形式のいずれかが使用できます。

フォルダの完全名
: フォルダの完全名を指定した場合には、現在選択されているアカウントの指定された名前のフォルダを意味します。例えば、「受信箱/テスト」のように指定すると、「受信箱」の子フォルダの「テスト」フォルダを意味します。

アカウント名+フォルダの完全名
: アカウントまで含めて指定する場合には、フォルダの完全名の前に「//アカウント名/」を置きます。例えば、「//Main/受信箱」と指定すると、現在選択されているアカウントに関係なく「Main」アカウントの「受信箱」を意味します。

指定されたフォルダが見つからない場合や移動できない場合にはエラーになります。

他のアカウントに移動した場合や、ローカルフォルダからリモートフォルダ、またはその逆に移動した場合、コンテキストメッセージが存在しなくなります。このため、以降のマクロでコンテキストメッセージが必要な関数などを呼び出すとエラーになります。

引数

String folder
: フォルダ名

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- 指定されたフォルダが見つからない場合
- 指定されたフォルダに移動できない場合

条件

- メッセージに対する変更が可能

例

```
# Testに移動
@Move('Test')
```

@Multipart

```
Boolean @Multipart()
```

説明

コンテキストメッセージがマルチパートの場合にはTrue、それ以外の場合にはFalseを返します。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合

条件

なし

例

```
# マルチパートかどうか調べる
@Multipart()
```

@Name

```
Address @Name(Field field)
```

説明

引数で指定されたフィールドをアドレスリストとしてパースし、含まれている名前のリストを返します。フィールドがアドレスを指定するフィールドでなかった場合やパースに失敗した場合には空のリストを返します。

パースされた各アドレスに対して名前が指定されている場合にはその名前を、指定されていない場合にはアドレスを返します。グループアドレスの場合には、グループの名前を返します。

たとえば、

```
To: test1@example.org,
 Test2 <test2@example.org>,
 Test3: test3@example.org, Test4 <test4@example.org>;
```

というヘッダを含むメッセージに対して、@Name(To)を適用すると、test1@example.org, Test2, Test3を含む名前のリストが返されます。

引数

Field field
　　名前のリストを取得するフィールド

エラー

- 引数の数が合っていない場合
- 引数の型が合っていない場合

条件

なし

例

```
# Toで指定された名前のリストを取得
@Name(To)
```

@New

```
Boolean @New()
```

説明

自動振り分けを使用して振り分け中に、コンテキストメッセージが新着メッセージの場合にTrueを、それ以外の場合にはFalseを返します。それ以外の時に使用された場合には常にFalseを返します。自動振り分けで既存のメッセージを振り分ける設定の場合に、この関数を使用することで新着メッセージかどうかを判断することができます。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 新着メッセージ
@New()

# 既存のメッセージ
@Not(@New())
```

@Not

```
Boolean @Not(Boolean arg)
```

説明

argで指定された真偽値の反対の値を返します。つまり、Trueの場合にはFalseを、Falseの場合にはTrueを返します。

引数

Boolean arg
　　真偽値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# Trueの反対
# -> False
@Not(@True())

# 未読
@Not(@Seen())
```

@Or

```
Boolean @Or(Boolean arg+)
```

説明

argで指定された真偽値の論理和を返します。つまり、全ての値がFalseならばFalseを返し、それ以外の場合にはTrueを返します。1個以上の任意の数の引数を渡せます。

引数

Boolean arg
　　真偽値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# TrueとFalseの論理和
# -> True
@And(@True(), @False())

# 未読またはマークされている
@Or(@Not(@Seen()), @Marked())
```

@OSVersion

```
String @OSVersion()
```

説明

OSの名前とバージョンを返します。例えば、「Windows XP Service Pack 2」のような文字列が返されます。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# OSの名前とバージョンを取得
@OSVersion()
```

その他のテンプレート

作成用と表示用以外に以下のテンプレートがあります。

印刷用のテンプレート

print.templateは印刷するときに使用するテンプレートです。QMAIL3で印刷するときには、対象のメッセージをprint.templateで文字列に変換してからそれを関連付けによって印刷します。デフォルトのprint.templateはメッセージをHTMLにフォーマットします。フォーマットを変更したい場合や、その他の形式で印刷したい場合にはテンプレートを編集する必要があります。詳細は、印刷を参照してください。

引用用のテンプレート

quote.templateはエディットビューで[編集]-[引用符付きで貼り付け]を実行したときに使用されるテンプレートです。引用元の表示を変更したい場合などには、このテンプレートを編集する必要があります。

エクスポート用のテンプレート

メッセージをエクスポートするときに使用されるテンプレートです。エクスポート時に使用するテンプレートのファイル名は「export_」から始まっている必要があります。たとえば、「export_Simple.template」という名前のテンプレートを作成すると、[書き出し]ダイアログの[テンプレート]に「Simple」がリストされます。エクスポート時のテンプレートの使用方法については、FileExportアクションを参照してください。

mailto URL処理用のテンプレート

url.templateは、mailto URLに関連付けされた場合など、コマンドラインで-sオプションを使用してmailto URLを渡された場合に使用されるテンプレートです。mailto URLから宛先やSubject、本文などを取り出してメッセージを作成し、エディットウィンドウで開きます。この処理をするときにはurlという名前のテンプレートの引数に指定されたURLが渡されます。

@Param

```
String @Param(String name)
```

説明

コンテキストメッセージのパラメータを返します。指定された名前のパラメータが設定されていない場合には空文字列を返します。

以下のパラメータ名が取得できます。

SignedBy
: S/MIMEの署名を検証したときに取得した署名したDN、またはPGPの署名を検証したときに取得した署名したユーザ名。

Certificate
: S/MIMEの署名を検証したときに取得した証明書、またはPGPの署名を検証したときに取得した情報。

Verify
: S/MIMEやPGPで署名を検証したときや復号した結果。以下の文字列を空白文字で区切った文字列になります。

  Verified
  : 検証に成功した場合

  AddressMatch
  : アドレスがマッチした場合（検証に成功した場合のみ）

  AddressMismatch
  : アドレスがマッチしなかった場合（検証に成功した場合のみ）

  VerifyFailed
  : 検証に失敗した場合

  Decrypted
  : 復号した場合

引数

String name
: パラメータ名

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合

- メッセージの取得に失敗した場合

条件

なし

例

```
# SignedByパラメータを取得
@Param('SignedBy')

# 署名が検証されたかどうかを調べる
@Contain(@Param('Verify'), 'Verified')
```

@ParseURL

```
String @ParseURL(String url)
```

説明

urlで指定されたmailto URLをパースしてヘッダ形式の文字列（とあれば本文）を返します。

たとえば、<mailto:test@example.org>からは、

```
To: test@example.org
```

を、<mailto:test@example.org?Cc=test2@example.org?Subject=Test&Body=Test%20Body>からは、

```
To: test@example.org
Cc: test2@example.org
Subject: Test

Test Body
```

を返します。処理されるヘッダは、To, Cc, Subject, In-Reply-To, Referencesとbodyです。

URL中の非ASCII文字列（%xxでエスケープされたものも含む）はUTF-8として正しいバイト列であるときにはUTF-8として、正しくないバイト列であるときにはプラットフォームのデフォルトエンコーディングとして処理されます。

urlで指定された文字列がmailto:から始まっていなかった場合には、Toとして指定された文字列を持ったヘッダ形式の文字列を返します。たとえば、test3@example.orgを引数として渡すと、

```
To: test3@example.org
```

を返します。

引数

String url
: パースするmailto URL

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# urlという変数に設定されたURLをパース
@ParseUrl($url)
```

@Part

```
Part @Part(Number index, Part part?)
```

説明

コンテキストメッセージの指定されたインデックスのパートを返します。partが指定された場合にはそのパートの子パートの指定されたインデックス番目を返します。インデックスは0ベースです。

パートが存在しないインデックスを指定した場合にもエラーになりませんが、返されたPartを@Bodyなどのパートを引数に指定できる関数に渡すとエラーになります。パートが存在するかどうかは返されたPartをBooleanに変換することで調べられます。

引数

Number index
　　パートのインデックス
Part part
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# マルチパートなら最初のパートの本文を取得
```

```
@If(@Multipart(),
    @If(@Part(0),
        @Body('', :BODY-INLINE, @Part(0)),
        ''),
    @Body('', :BODY-INLINE))
```

@Partial

```
Boolean @Partial()
```

説明

コンテキストメッセージが全てダウンロードされているかどうかを返します。メッセージ全体がダウンロードされている場合にはFalse、それ以外の場合にはTrueを返します。IMAP4やNNTPでメッセージをキャッシュしていない場合にもTrueを返します。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合

条件

なし

例

```
# メッセージ全体がダウンロードされているかどうかを調べる
@Not(@Partial())
```

@Passed

```
Boolean @Passed(Number value, Number unit?)
```

説明

コンテキストメッセージの日付から指定された時間が経過したかどうかを返します。

unitにはvalueで指定した数値の単位を指定します。指定できるのは以下のいずれかです。

:PASSED-DAY
　　日
:PASSED-HOUR
　　時間
:PASSED-MINUTE
　　分
:PASSED-SECOND
　　秒

指定しなかった場合には:PASSED-DAYを指定したのと同じになります。

引数

Number value
　　経過した時間
Number unit
　　経過した時間の単位

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合

条件

なし

例

```
# 7日経っているかどうか調べる
@Passed(7)

# 36時間経っているかどうか調べる
@Passed(36, :PASSED-HOUR)
```

@ProcessId

```
Number @ProcessId()
```

説明

現在のプロセスのIDを返します。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# プロセスIDを取得
@ProcessId()
```

@Profile

```
String @Profile(String path, String section, String key, String default?)
```

説明

プロファイル形式のXMLファイルから値を取り出します。プロファイル形式はqmail.xmlと同様の形式です。詳細は、qmail.xmlを参照してください。

pathに空文字列を指定するとデフォルトのqmail.xmlから値を取得します。この場合、すでにロードされているqmail.xmlから値をロードするため高速にロードできますが、内部的に書き換えられている場合には現在のファイルの内容と異なる可能性があります。それ以外のパスを指定すると指定されたファイルをプロファイル形式のXMLファイルとして読み込み、値を返します。相対パスを指定するとメールボックスディレクトリからの相対パスとして解釈します。

sectionとkeyにはセクションの名前とキーの名前を指定します。defaultには値が存在しなかった場合に返す値を指定します。指定しなかった場合に値が存在しないと空文字列を返します。

引数

String path
　　ファイルのパス
String section
　　セクションの名前
String key
　　キーの名前
String default
　　値が存在しなかったときに返す値

エラー

- 引数の数が合っていない場合
- ファイルの読み込みに失敗した場合（pathに空文字列以外を指定した場合）

条件

なし

例

```
# qmail.xmlのGlobalセクションのBccの値を取得（デフォルトは1）
@Profile('', 'Global', 'Bcc', '1')

# C:\test.xmlからTestセクションのFooの値を取得
@Profile('C:\\test.xml', 'Test', 'Foo')
```

@ProfileName

```
String @ProcessName()
```

説明

プロファイル名を返します。プロファイルを参照してください。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# プロファイル名を取得
@ProfileName()
```

@Progn

```
Value @Progn(Value value+)
```

説明

引数で指定された値を順に評価し最後に評価した値を返します。副作用のある関数を順次呼び出したい場合に使用します。

引数

Value value
　　値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 一度変数に値を代入してからその値を使って計算した結果を取得
@Progn(@Set('x', 1),
       @Set('y', 2),
       @Add($x, $y))
```

@Quote

```
String @Quote(String s, String quote)
```

説明

指定された文字列を指定された引用符で引用します。各行の先頭に引用符が挿入されます。

引数

String s
　　文字列
String quote
　　引用符

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 本文を引用してクリップボードにコピー
# @Clipboard(@Quote(@Body(), '> '))
```

@References

```
String @References(Number size?)
```

説明

コンテキストメッセージのReferencesを返します。sizeが指定された場合には、後から最大でその個数分だけを返します。

引数

Number size
　　最大取得数

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合

条件

なし

例

```
# 最大4個のReferencesを取得
@References(4)
```

@RegexFind

```
Number @RegexFind(String s, Regex regex, Number index?)
```

説明

sで指定した文字列の中で最初にregexで指定した正規表現にマッチするインデックスを返します。indexが指定された場合には、指定された文字数目から検索を始めます。インデックスは0ベースです。見つからなかった場合には-1を返します。

正規表現でキャプチャされた文字列は、以降のマクロで$_<数字>でアクセスできます。$_0はマッチした文字列全体をあらわし、$_1以降は各キャプチャされた文字列をあらわします。

regexには正規表現以外が指定された場合には、文字列に変換した上で正規表現としてコンパイルされます。

引数

String s
: 文字列

Regex regex
: 正規表現

Number index
: 検索を始めるインデックス

エラー

- 引数の数が合っていない場合
- 正規表現が不正な場合（文字列で指定した場合）

条件

なし

例

```
# 正規表現で検索
# -> 1
@RegexFind('abcabc', /b./)

# 正規表現を文字列で指定
# -> 2
@RegexFind('abcabc', 'b.*')

# インデックスを指定
# -> 6
@RegexFind('ABCabcABC', /[A-Z]/, 3)
```

@RegexMatch

```
Boolean @RegexMatch(String s, Regex regex)
```

説明

sで指定した文字列がregexで指定した正規表現にマッチするかどうかを返します。指定した文字列の一部がマッチする場合にもTrueを返します。

正規表現でキャプチャされた文字列は、以降のマクロで$_<数字>でアクセスできます。$_0はマッチした文字列全体をあらわし、$_1以降は各キャプチャされた文字列をあらわします。

regexには正規表現以外が指定された場合には、文字列に変換した上で正規表現としてコンパイルされます。

引数

String s
　　文字列
Regex regex
　　正規表現

エラー

- 引数の数が合っていない場合
- 正規表現が不正な場合（文字列で指定した場合）

条件

なし

例

```
# 本文中に正規表現にマッチする文字列があるかどうかを調べる
```

```
@RegexMatch(@Body(), /http:\/\//)

# マッチした文字列を返す
@Progn(@RegexMatch(@Body(), /^http:(.*)$/), $_1)
```

@RegexReplace

```
String @RegexReplace(String s, Regex regex, String replace, Boolean global?)
```

説明

sで指定した文字列中のregexで指定した正規表現にマッチする文字列をreplaceで指定した文字列に置換します。マッチしなかった場合には何もしません。globalがTrueの場合には文字列中の全てのマッチした文字列が置換されます。Flaseの場合や指定されなかった場合には始めに見つかった文字列のみが置換されます。

置換文字列中で正規表現によってキャプチャされた文字列を使用したい場合には、$<数値>を使用します。$0はマッチした文字列全体をあらわし、$1以降は各キャプチャされた文字列をあらわします。

正規表現でキャプチャされた文字列は、以降のマクロで$_<数字>でアクセスできます。$_0はマッチした文字列全体をあらわし、$_1以降は各キャプチャされた文字列をあらわします。

regexには正規表現以外が指定された場合には、文字列に変換した上で正規表現としてコンパイルされます。

引数

String s
: 文字列

Regex regex
: 正規表現

String replace
: 置換文字列

Boolean global
: 全てを置換するかどうか

エラー

- 引数の数が合っていない場合
- 正規表現が不正な場合（文字列で指定した場合）

条件

なし

例

```
# 正規表現で置換
# -> abcz23
@RegexReplace('abc123', /[0-9]/, 'z')

# 全体を置換
# -> abczzz
@RegexReplace('abc123', /[0-9]/, 'z', @True())

# Subjectの先頭からRe:を削除
@RegexReplace(Subject, /^Re(?:^[0-9]+)?: *(.*)$/i, '$1')
```

@Remove

```
String @Remove(String address, String remove+)
```

説明

addressで指定されたアドレスのリストから、removeで指定されたアドレスを削除した結果のアドレスのリストを返します。

引数

String address
　　アドレスのリスト
String remove
　　削除するアドレス

エラー

- 引数の数が合っていない場合
- アドレスのパースに失敗した場合

条件

なし

例

```
# Ccに指定されたアドレスから自分のアドレスを削除
@Remove(Cc, @Address(@I()))

# Toに指定されたアドレスから自分のアドレスとFromに指定されたアドレスを削除
@Remove(To, @Address(@I()), @Address(From))
```

@Replied

```
Boolean @Replied(Boolean replied?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが返信済みの場合にはTrue、それ以外の場合にはFalseを返します。repliedにTrueを指定されて呼び出された場合にはコンテキストメッセージを返信済みにし、Falseを指定された場合には返信済みではなくします。フラグを参照してください。

引数

Boolean replied
　　返信済みにするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 返信済み、または返信済みではなくするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 返信済みかどうか調べる
@Replied()

# 返信済みにする
@Replied(@True())
```

@SaveAttachment

```
String @SaveAttachment(String path, Expr expr?, Part part?)
```

説明

コンテキストメッセージの添付ファイルをpathで指定されたディレクトリに保存します。partが指定された場合には、そのパートの添付ファイルを保存します。保存先のファイルが既に存在する場合は上書きします。

exprが指定された場合には、各添付ファイルを保存する前に指定した式が呼び出されます。このとき一番目の引数に添付ファイル名が、二番目の引数に添付ファイルのパートが渡されます。この式が返した文字列がファイル名になります。exprが指定されない場合には、添付ファイル名がそのままファイル名になります。

最後に保存した添付ファイルのパスを返します。

引数

String path
　　保存先のディレクトリのパス
Expr expr
　　ファイル名を決める式
Part part
　　パート

エラー

- 引数の数が合っていない場合
- メッセージの取得に失敗した場合
- 指定したパートがない場合（partを指定した場合）
- 保存に失敗した場合

条件

なし

例

```
# すべての添付ファイルをC:\Tempに保存する
@SaveAttachment('C:\\Temp')

# ファイル名の先頭に日付をつけて保存する
@SaveAttachment('C:\\Temp',
                @Concat(@FormatDate(@Date(), '%Y4%M0%D'), '_', $1))
```

@Save

```
Boolean @Save(String path, String content, String encoding?)
```

説明

指定されたパスに指定された内容を書き込み、書き込んだ内容を返します。パスに相対パスが指定された場合にはメールボックスディレクトリからの相対パスになります。

encodingを指定すると指定されたエンコーディングで書き込みます。指定されない場合にはシステムのデフォルトのエンコーディングで書き込みます。

引数

String path
　　ファイルパス
String content
　　内容
String encoding
　　エンコーディング

エラー

- 引数の数が合っていない場合
- 書き込みに失敗した場合

条件

なし

例

```
# 本文をファイルに書き込む
@Save('C:\\Temp\\Test.txt', @Body())
```

@Script

```
Value @Script(String script, String lang, Value args+)
```

説明

指定されたスクリプトを実行します。

引数

String script
　　実行するスクリプト
String lang
　　スクリプト言語の名前
Value args
　　スクリプトに渡す引数

エラー

- 引数の数が合っていない場合
- スクリプトのパース・実行に失敗した場合

条件

なし

例

```
# スクリプトを実行
@Script('result.value = 1 + 2', 'JScript')

# スクリプトに引数を渡して実行
@Script('result.value = arguments(0) + arguments(1)', 'JScript', 1, 2)
```

@Seen

```
Boolean @Seen(Boolean seen?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが既読の場合にはTrue、それ以外の場合にはFalseを返します。seenにTrueを指定されて呼び出された場合にはコンテキストメッセージを既読にし、Falseを指定された場合には未読にします。フラグを参照してください。

引数

Boolean seen
　　既読にするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 既読、または未読にするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 既読かどうか調べる
@Seen()

# 未読にする
@Seen(@False())
```

@SelectBox

```
String @SelectBox(String message, String candidates, Number type?, String
default?)
```

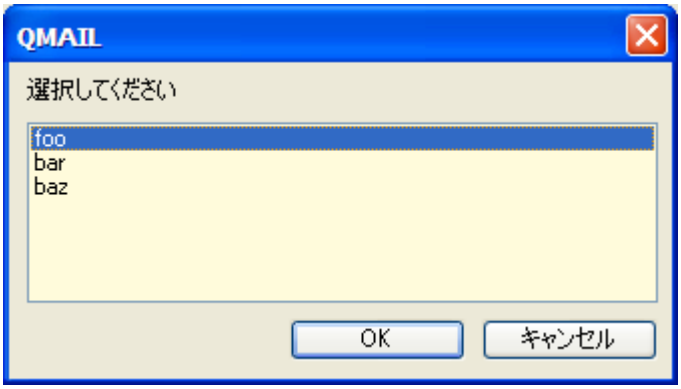
説明

ユーザに選択を求めるためのダイアログを表示し、選択された文字列を返します。messageには表示するメッセージを指定します。candidatesにはリストに候補として表示する文字列を改行区切りで指定します。たとえば、foo, bar, bazを候補にする場合には、'foo\nbar\nbaz'のように指定します。

typeにはリストのタイプを指定します。指定できるのは以下のとおりです。

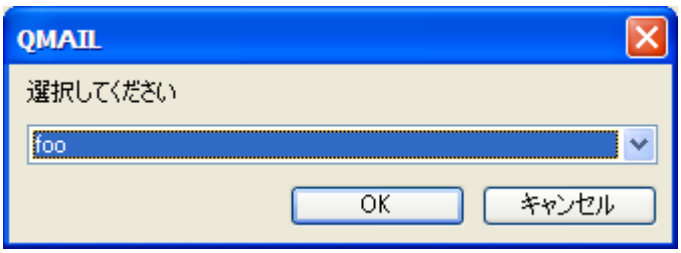
:SELECT-LIST

　　リスト表示



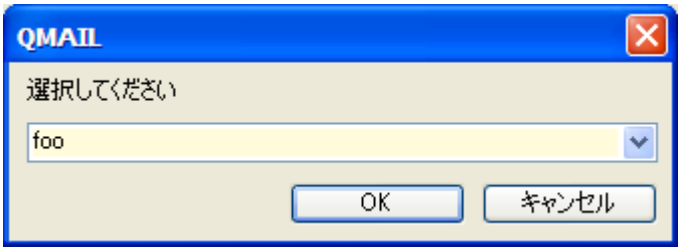
:SELECT-DROPDOWNLIST

　　ドロップダウンリスト表示



:SELECT-DROPDOWN

　　ドロップダウン表示



指定しない場合には、SELECT-LISTを指定したのと同じになります。

defaultを指定するとデフォルトでその候補が選択されます。候補にない値を指定すると、指定しないのと同じになりますが、typeが:SELECT-DROPDOWNの場合にはその値が入力された状態になります。指定しない場合には先頭の候補が選択されます。

引数

String message
　　表示するメッセージ
String candidates
　　リストに表示する候補
Number type
　　リストのタイプ
String default
　　デフォルトの入力文字列

エラー

- 引数の数が合っていない場合
- UIがない場合

条件

- UIが必要

例

```
# foo, bar, bazからリストで選択
@SelectBox('選択してください', 'foo\nbar\nbaz')

# us-ascii, iso-2022-jp, utf-8からドロップダウンで選択（デフォルトはiso-2022-jp）
@SelectBox('エンコーディング', 'us-ascii\niso-2022-jp\nutf-8', :SELECT-DROPDOWN,
'iso-2022-jp')
```

@Selected

```
MessageList @Selected()
```

説明

選択されているメッセージのリストを返します。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 選択されているメッセージ全てにラベルを設定する
@ForEach(@Selected(), @Label('qmail'))
```

@Sent

```
Boolean @Sent(Boolean sent?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージが送信済みの場合にはTrue、それ以外の場合にはFalseを返します。sentにTrueを指定されて呼び出された場合にはコンテキストメッセージを送信済みにし、Falseを指定された場合には送信済みではなくします。フラグを参照してください。

引数

Boolean sent
　　送信済みにするかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- 送信済み、または送信済みではなくするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# 送信済みかどうか調べる
@Sent()

# 送信済みにする
@Sent(@True())
```

@Set

```
Value @Set(String name, Value value, Boolean global?)
```

説明

nameで指定された名前の変数にvalueを代入します。globalに:GLOBALを指定するとグローバル変数になります。指定されなかった場合にはローカル変数になります。valueをそのまま返します。

引数

String name
　　変数名
Value value
　　値
Boolean global
　　グローバル変数かどうか

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# testという名前のローカル変数に@Address(To)の結果を代入
@Set('test', @Address(To))

# bccという名前のグローバル変数に@Profile('', 'Global', 'Bcc', '1')の結果を代入
@Set('bcc', @Profile('', 'Global', 'Bcc', '1'), :GLOBAL)
```

@Size

```
Number @Size(Boolean text?)
```

説明

コンテキストメッセージのサイズを返します。textにTrueが指定された場合には、取得できれば本文部分のみサイズを返します。取得できない場合にはメッセージ全体のサイズを返します。textにFalseが指定された場合や引数が省略された場合にはメッセージ全体のサイズを返します。

引数

Boolean text
　　本文部分のみのサイズを返すかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合

条件

なし

例

```
# サイズを取得
@Size()

# 本文部分のサイズを取得
@Size(@True())
```

@SpecialFolder

```
String @SpecialFolder(Number type, String account?)
```

説明

特殊フォルダの完全名を取得します。

typeには取得したい特殊フォルダのタイプを指定します。以下のいずれかが指定できます。

:SF-INBOX
　　受信箱
:SF-OUTBOX
　　送信箱
:SF-SENTBOX
　　送信済み
:SF-TRASHBOX
　　ゴミ箱
:SF-DRAFTBOX
　　草稿箱
:SF-SEARCHBOX
　　検索
:SF-JUNKBOX
　　スパム

accountにアカウント名を指定すると指定したアカウントの特殊フォルダを取得します。それ以外の場合にはコンテキストアカウントの特殊フォルダを取得します。

引数

Number type
　　取得する特殊フォルダのタイプ
String account
　　アカウント

エラー

- 引数の数が合っていない場合
- コンテキストアカウントがない場合（アカウントを指定しなかった場合）
- 指定されたアカウントが見つからない場合（アカウントを指定した場合）

- 指定されたタイプが不正な場合

条件

なし

例

```
# 受信箱のフォルダ名を取得
@Folder(:SF-INBOX)

# Testという名前のアカウントのスパムフォルダの名前を取得
@Folder(:SF-JUNKBOX, 'Test')
```

特殊なヘッダ

QMAIL3内部で使用される特殊なヘッダです。一部のヘッダは作成用のテンプレートで使用することができます。また、外部エディタを使用してメッセージを作成する場合には、付加的な情報を外部エディタからQMAIL3に渡すためにこれらのヘッダの一部を使用することができます。

ここで定義されているヘッダを含め、X-QMAIL-から始まるヘッダは送信直前に削除されます。独自にヘッダに保存しておきたい情報がある場合には、X-QMAIL-から始まる名前を使用することで、その情報が送信されてしまうことを防ぐことができます。

X-QMAIL-Account

アカウント名を指定します。

テンプレートでこのヘッダを指定すると、エディットビューで開いたときに指定されたアカウントが選択されます。

外部エディタからメッセージを作成するときに指定すると、指定されたアカウントの送信箱にメッセージが作成されます。

X-QMAIL-SubAccount

サブアカウント名を指定します。

テンプレートでこのヘッダを指定すると、エディットビューを開いたときに指定されたサブアカウントが選択されます。

メッセージ作成時に指定すると、指定したサブアカウントと同じ同一性を持つサブアカウント以外から送信されなくなります。詳しくは、マルチアカウントを参照してください。

X-QMAIL-Signature

署名の名前を指定します。

テンプレートでこのヘッダを指定すると、エディットビューを開いたときに指定された署名が選択されます。このとき空文字列を指定すると、署名として「(なし)」が選択されます。メッセージを作成したときには、再度メッセージを編集したときに二重に署名が付かないように、自動的に空文字列が指定されます。

外部エディタからメッセージを作成するときに指定すると、指定された署名が自動的に挿入されます。

X-QMAIL-Attachment

添付するファイルのパスまたはメッセージのURIを指定します。複数のパスまたはURIを指定するときには「,」で区切ります。パスに「,」や「\」を含めるには「\」でエスケープします。例えば、

```
X-QMAIL-Attachment: C:\\Temp\\test.png, C:\\Program Files\\QMAIL3\\q3u.exe
```

のように指定します。

テンプレートでこのヘッダを指定すると、エディットビューを開いたときに指定したファイルが添付ファイルとして指定された状態になります。

外部エディタからメッセージを作成するときに指定すると、指定されたファイルが添付されます。

X-QMAIL-ArchiveAttachment

添付ファイルを圧縮するときのファイル名を指定します。

外部エディタからメッセージを作成するときに指定すると、指定されたファイル名で添付ファイルが圧縮されます。

X-QMAIL-AttachmentDeleted

添付ファイルを削除したときに自動的に付加されます。10進で数値を指定して値が0以外だと添付ファイルが削除されたとみなされます。

X-QMAIL-OriginalCharset

返信元のメッセージの文字コードを指定します。

外部エディタからメッセージを作成するときに指定すると、その文字コードがデフォルトの文字コードと互換性がない場合には、作成されるメッセージがutf-8でエンコードされます。

X-QMAIL-Flags

メッセージのフラグを10進数で指定します。

メッセージをエクスポートするときに[フラグを書き出す]にチェックを入れるとフラグがこのヘッダに設定されます。

メッセージをインポートするときに、このヘッダがあると読み込んだメッセージにフラグを反映します。

X-QMAIL-Link

URLを指定します。

MessageOpenLinkアクションが実行されたときに開くURLを指定します。RSSアカウントでフィードを取り込んだときに自動的に設定されます。

X-QMAIL-Macro

メッセージが送信箱に入れられたときに実行されるマクロを指定します。このマクロはコンテキストメッセージが無い状態で実行されます。

例えば、標準のreply.templateでは、メッセージが送信箱に入れられたときに返信元のメッセージに返信フラグを立てるためのマクロを設定します。

X-QMAIL-DraftMacro

メッセージが草稿として草稿箱に保存されたときに実行されるマクロを指定します。このマクロはコンテキストメッセージが無い状態で実行されます。

例えば、標準のedit.templateでは、メッセージが草稿として保存されたときに元のメッセージを削除するためのマクロを設定します。

X-QMAIL-EditMacro

メッセージがエディットウィンドウで開かれたときに実行されるマクロを指定します。このマクロはコンテキストメッセージ、コンテキストフォルダとも無い状態で実行されます。指定したマクロはエディットウィンドウが表示される直前に実行されます。@InvokeActionを使用してエディットウィンドウを閉じるようなアクション（FileSendアクションなど）を使用した場合、エディットウィンドウは表示されること無く破棄されます。

X-QMAIL-EnvelopeFrom

SMTPサーバに送信するEnvelope Fromを指定します。通常、Fromヘッダで指定したアドレスが使用されますが、それとは異なるアドレスをEnvelope Fromとして使用したい場合には指定することができます。

アカウント単位でEnvelope Fromを指定したい場合には、account.xmlのSmtp/EnvelopeFromで指定することもできます。

X-QMAIL-Sentbox

メッセージが送信された後に移動されるフォルダを指定します。通常は送信されたメッセージは送信箱から送信控えに移動されますが、このヘッダが指定されている場合には指定されたフォルダに移動されます。ただし、指定されたフォルダが見つからなかったり、通常フォルダでなかった場合には、送信控えに移動されます。

たとえば、reply.templateで返信元のメッセージのフォルダ名をこのヘッダに設定することで、送信控えを返信元のメッセージと同じフォルダに保存することができます。

X-QMAIL-Outbox

メッセージを作成したときに保存されるフォルダを指定します。通常は作成されたメッセージは送信箱（ドラフトの場合には草稿箱）に保存されますが、このヘッダが指定されている場合には指定されたフォルダに保存されます。ただし、指定されたフォルダが見つからなかったり、通常フォルダでなかった場合には、送信箱（または草稿箱）に保存されます。

指定されたフォルダが送信箱でなかった場合には、保存されたメッセージは送信されませんので注意してください。このヘッダは、たとえばメモを保存したい場合などに使用することができます。

@SubAccount

```
String @SubAccount()
```

説明

コンテキストアカウントの現在のサブアカウントの名前を返します。コンテキストアカウントがない場合にはエラーになります。現在のサブアカウントがデフォルトのサブアカウントだった場合には空文字列が返されます。

引数

なし

エラー

- 引数の数が合っていない場合
- コンテキストアカウントがない場合

条件

なし

例

```
# サブアカウント名を取得
# ->
@SubAccount()
```

@Subject

```
String @Subject(Boolean removeRe?, Boolean removeMl?)
```

説明

コンテキストメッセージのSubjectを返します。取得できなかった場合には空文字列を返します。

removeReにTrueを指定すると先頭の「Re:」に類するものを取り除きます。removeMlにTrueを指定すると先頭のML番号（[qs:00123]のような文字列）を取り除きます。これらを指定しない場合には、Falseを指定したのと同じになります。

@Subject()はSubjectと同じ結果を返します。

引数

Boolean removeRe
　　先頭のRe:を取り除くかどうか
Boolean removeMl
　　先頭のML番号を取り除くかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- メッセージの取得に失敗した場合

条件

なし

例

```
# Re:とML番号を取り除いてSubjectを取得
```

```
@Subject(@True(), @True())
```

@SubstringAfter

```
String @SubstringAfter(String s, String separator, Boolean case?)
```

説明

sで指定された文字列内のseparatorで指定した文字列より後の文字列を返します。そのような文字列が見つからない場合には空文字列を返します。caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

String s
　　文字列
String separator
　　検索する文字列
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# cより後を取得
# -> def
@SubstringAfter('abcdef', 'c')

# 大文字小文字を区別してXYZより後を取得
# -> ZZ
@SubstringAfter('wxyzXYZZZ', 'XYZ', @True())
```

@SubstringBefore

```
String @SubstringBefore(String s, String separator, Boolean case?)
```

説明

sで指定された文字列内のseparatorで指定した文字列より前の文字列を返します。そのような文字列が見つからない場合には空文字列を返します。caseにTrueを指定すると大文字と小文字を区別し、Falseの場合や省略された場合には区別しません。

引数

String s
　　文字列
String separator
　　検索する文字列
Boolean case
　　大文字と小文字を区別する場合にはTrue、それ以外の場合にはFalse

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# cより前を取得
# -> ab
@SubstringBefore('abcdef', 'c')

# 大文字小文字を区別してXYZより前を取得
# -> wxyz
@SubstringBefore('wxyzXYZZZ', 'XYZ', @True())
```

@Substring

```
String @Substring(String s, Number offset, Number length?)
```

説明

sで指定された文字列の、offsetで指定された文字目からlengthで指定された長さの部分文字列を取得します。offsetは0ベースで指定します。lengthを指定しない場合や長さが文字列の最後を超える場合には文字列の最後までを取得します。offsetがsの長さよりも大きい場合には空文字列を返します。

引数

String s
: 文字列

Number offset
: 0ベースのオフセット

Number length
: 長さ

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 3文字目から4文字取得
# -> defg
@Substring('abcdefghij', 3, 4)

# 5文字目以降を取得
# -> fghij
@Substring('abcdefghij', 5)
```

@Subtract

```
Number @Subtract(Number arg1, Number arg2)
```

説明

arg1とarg2の差を返します。

引数

Number arg1
　　数値
Number arg2
　　数値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 10 - 7
# -> 3
@Subtract(10, 7)
```

同期フィルタ

同期フィルタはサーバからメッセージを受信するときの動作をコントロールするために使用します。例えば、以下のようなことができます。

- POP3であるサイズ以上の大きさのメッセージは最初の100行だけ取り込む
- IMAP4で特定の相手から来たメッセージはテキスト部分だけキャッシュする
- NNTPで件名に特定の文字が含まれているメッセージだけダウンロードする

デフォルトでは以下のような同期フィルタが用意されています。

ヘッダのみ (POP3)
: POP3でヘッダのみダウンロードします。

最大100行 (POP3)
: POP3で本文の先頭100行までをダウンロードします。

ヘッダのみ (IMAP4)
: IMAP4でヘッダのみキャッシュします。

テキストのみ (IMAP4)
: IMAP4でテキスト部分のみキャッシュします。

すべて (IMAP4)
: IMAP4でメッセージ全体をキャッシュします。

すべて (NNTP)
: NNTPでメッセージ全体をキャッシュします。

これらの動作をカスタマイズしたり、特定の条件にマッチするメッセージに対してのみ動作を適用するには、同期フィルタを作成します。同期フィルタを作成するには、[ツール]–[オプション]を選択して[オプション]ダイアログを開き、[同期フィルタ]パネルを選択します。設定方法については、同期フィルタの設定を参照してください。

受信時にどの同期フィルタを使用するかは、アカウントの設定の高度の設定の[同期フィルタ]で指定します。また、巡回時には、ここで指定したのとは別の同期フィルタを指定することができます。詳細は、巡回の設定を参照してください。

同期フィルタを指定しないと、POP3の場合にはメッセージ全体をダウンロードし、IMAP4とNNTPではインデックスのみ作成します。

@Thread

```
MessageList @Thread(Boolean all?)
```

説明

コンテキストメッセージと同じスレッドに属するメッセージのリストを返します。allにTrueを指定すると同じアカウントの全てのメッセージからスレッドを生成します。Falseを指定した場合や指定されなかった場合には同じフォルダ内のメッセージからスレッドを生成します。

引数

Boolean all
　　全てのフォルダのメッセージを調べるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- コンテキストメッセージホルダが一時的な場合

条件

なし

例

```
# 現在のメッセージと同じスレッドに属するメッセージを全て既読にする
@ForEach(@Thread(), @Seen(@True()))
```

@True

```
Boolean @True()
```

説明

Trueを返します。

引数

なし

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# True
# -> True
@True()
```

@UnseenMessageCount

```
Number @UnseenMessageCount(String folder?)
```

説明

未読メッセージ数を返します。folderが指定された場合にはそのフォルダ内の未読メッセージ数を返します。指定されなかった場合にはアカウント中のすべての未読メッセージ数を返します。

引数

String folder
　　フォルダの完全名

エラー

- 引数の数が合っていない場合
- 指定されたフォルダが存在しない場合

条件

なし

例

```
# アカウント中の未読メッセージ数を取得する
@UnseenMessageCount()

# カレントフォルダの未読メッセージ数を取得する
@UnseenMessageCount(@Folder(:FN-FULLNAME, :FT-CURRENT))
```

@URI

```
String @URI(Part part?)
```

説明

コンテキストメッセージを指す内部的に使用されるURIを返します。

引数

Part part?
　　パート

エラー

- 引数の数が合っていない場合
- コンテキストメッセージホルダがない場合
- コンテキストメッセージホルダが一時的な場合
- メッセージの取得に失敗した場合（partを指定した場合）
- 指定したパートがない場合（partを指定した場合）

条件

なし

例

```
# URIを取得する
@URI()
```

@User1

```
Boolean @User1(Boolean user?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージにユーザ1フラグが付いている場合にはTrue、それ以外の場合にはFalseを返します。userにTrueを指定されて呼び出された場合にはコンテキストメッセージにユーザ1フラグを付け、Falseを指定された場合にはユーザ1フラグをはずします。フラグを参照してください。

引数

Boolean user
　　ユーザ1フラグを付けるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- フラグを付けたりはずしたりするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ユーザ1フラグが付いているかどうか調べる
@User1()

# ユーザ1フラグをつける
@User1(@True())
```

@User2

```
Boolean @User2(Boolean user?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージにユーザ2フラグが付いている場合にはTrue、それ以外の場合にはFalseを返します。userにTrueを指定されて呼び出された場合にはコンテキストメッセージにユーザ2フラグを付け、Falseを指定された場合にはユーザ2フラグをはずします。フラグを参照してください。

引数

Boolean user
　　ユーザ2フラグを付けるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- フラグを付けたりはずしたりするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ユーザ2フラグが付いているかどうか調べる
@User2()

# ユーザ2フラグをつける
@User2(@True())
```

@User3

```
Boolean @User3(Boolean user?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージにユーザ3フラグが付いている場合にはTrue、それ以外の場合にはFalseを返します。userにTrueを指定されて呼び出された場合にはコンテキストメッセージにユーザ3フラグを付け、Falseを指定された場合にはユーザ3フラグをはずします。フラグを参照してください。

引数

Boolean user
　　ユーザ3フラグを付けるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- フラグを付けたりはずしたりするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ユーザ3フラグが付いているかどうか調べる
@User3()

# ユーザ3フラグをつける
@User3(@True())
```

@User4

```
Boolean @User4(Boolean user?)
```

説明

引数なしで呼び出された場合には、コンテキストメッセージにユーザ4フラグが付いている場合にはTrue、それ以外の場合にはFalseを返します。userにTrueを指定されて呼び出された場合にはコンテキストメッセージにユーザ4フラグを付け、Falseを指定された場合にはユーザ4フラグをはずします。フラグを参照してください。

引数

Boolean user
　　ユーザ4フラグを付けるかどうか

エラー

- 引数の数が合っていない場合
- コンテキストメッセージがない場合
- コンテキストメッセージが一時的な場合（引数付きで呼び出された場合）
- フラグを付けたりはずしたりするのに失敗した場合（引数付きで呼び出された場合）

条件

なし

例

```
# ユーザ4フラグが付いているかどうか調べる
@User4()

# ユーザ4フラグをつける
@User4(@True())
```

@Variable

```
Value @Variable(String name, Value value?, Boolean global?)
```

説明

nameで指定された名前の変数の値を返します。変数が見つかった場合にはvalueとglobalは無視されます。指定した名前の変数が見つからない場合には、@Setと同様に振舞います。

引数

String name
　　変数名
Value value
　　値
Boolean global
　　グローバル変数かどうか

エラー

- 引数の数が合っていない場合

条件

なし

例

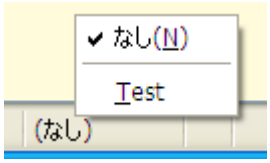
```
# testという変数に値が入っていたらそれを返す。入っていなかったら0をセットしてそれを返す
@Variable('test', 0)
```

表示用のテンプレート

表示用のテンプレートはメッセージを表示するときに使用することができます。表示用のテンプレートを評価した結果は単に文字列として画面上に表示されるだけなので、評価結果の形式は特にありません。

デフォルトでは表示用のテンプレートは含まれていませんが、「view_」から始まるファイル名でテンプレートを作成すると[表示]-[テンプレート]メニューの下にリストされます。たとえば、view_Test.templateというファイル名でテンプレートを作成すると、[表示]-[テンプレート]-[Test]として表示されます。

現在使用しているテンプレートは、ステータスバーに表示されます。ここを右クリックすることでも使用するテンプレートを指定することができます。



[なし]を選択するとテンプレートを使用しません。その他のテンプレートを指定すると指定したテンプレートが使用されます。

@While

```
Value @While(Boolean condition, Value value)
```

説明

conditionの値がTrueの間、valueの値を繰り返し評価します。conditionも毎回評価されます。最後に評価されたconditionが返されます。

使い方を誤ると無限ループしますので注意してください。

引数

Boolean condition
　　繰り返しの条件
Value value
　　評価する値

エラー

- 引数の数が合っていない場合

条件

なし

例

```
# 変数に値を代入しながら10回ループを回す
@While(@Less(@Variable('n', 0), 10),
       @Progn(@MessageBox($n),
              @Set('n', @Add($n, 1))))
```