



FormCalc ユーザーリファレンス

2009年10月

Adobe® LiveCycle™ Designer ES2

バージョン 9.0

© 2009 Adobe Systems Incorporated. All rights reserved.

Adobe® LiveCycle™ Designer ES2 9.0 FormCalc ユーザーリファレンス (Microsoft® Windows® 版)

第 4.0 版、2009 年 10 月

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names, company logos and user names in sample material or sample forms included in this documentation and/or software are for demonstration purposes only and are not intended to refer to any actual organization or persons.

Adobe, Adobe logo, Adobe Reader, Acrobat, and LiveCycle are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Microsoft and Windows are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are the property of their respective owners.

This product contains either BSAFE and/or TIPEM software by RSA Security, Inc.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by the IronSmith Project (<http://www.ironsmith.org/>).

This product includes software developed by the OpenSymphony Group (<http://www.opensymphony.com/>).

This product includes software developed by the Indiana University Extreme! Lab (<http://www.extreme.indiana.edu/>).

This product includes copyrighted software developed by E. Wray Johnson for use and distribution by the Object Data Management Group (<http://www.odmg.org/>).

Portions © Eastman Kodak Company, 199- and used under license. All rights reserved. Kodak is a registered trademark and Photo CD is a trademark of Eastman Kodak Company.

Powered by Celequest. Copyright 2005-2008 Adobe Systems Incorporated. All rights reserved. Contains technology distributed under license from Celequest Corporation. Copyright 2005 Celequest Corporation. All rights reserved.

Single sign-on, extending Active Directory to Adobe LiveCycle ES provided by Quest Software “www.quest.com/identity-management” in a subsequent minor release that is not a bug fix (i.e., version 1.1 to 1.2 but not 1.1.1 to 1.1.2) of the Licensee Product that incorporates the Licensed Product.

The Spelling portion of this product is based on Proximity Linguistic Technology. © Copyright 1989, 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1990 Merriam-Webster Inc. © Copyright 1990 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2003 Franklin Electronic Publishers Inc. © Copyright 2003 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2004 Franklin Electronic Publishers, Inc. © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA.

© Copyright 1991 Dr.Lluis de Yzaguirre I Maura © Copyright 1991 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1990 Munksgaard International Publishers Ltd. © Copyright 1990 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1995 Van Dale Lexicografie bv © Copyright 1996 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1990 IDE a.s. © Copyright 1990 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2004 Franklin Electronics Publishers, Inc. © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA.

© Copyright 1992 Hachette/Franklin Electronic Publishers, Inc. © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2004 Bertelsmann Lexikon Verlag © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2004 MorphoLogic Inc. © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1990 Williams Collins Sons & Co. Ltd. © Copyright 1990 All Rights Reserved Proximity

Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 1993-95 Russicon Company Ltd. © Copyright 1995 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA. © Copyright 2004 IDE a.s. © Copyright 2004 All Rights Reserved Proximity Technology A Division of Franklin Electronic Publishers, Inc. Burlington, New Jersey USA.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

はじめに	
本マニュアルの内容	5
対象読者	5
関連マニュアル	5
1. FormCalc の概要	
2. 構築ブロック	
数式	12
変数	20
参照構文	20
プロパティとメソッド呼び出し	24
組み込み関数の呼び出し	24
3. アルファベット順関数リスト	
4. 演算関数	
5. 日付と時間関数	
6. 会計関数	
7. 論理関数	
8. その他の関数	
9. 文字列関数	
10. URL 関数	
索引	82

はじめに

Adobe® LiveCycle™ Designer ES にはツールセットが用意されており、フォーム開発者はそれらを使用してインテリジェントなビジネス文書を構築できます。演算やスクリプティングを組み込むことで、フォームを充実させることができます。例えば、単純な演算を使用して発注書の費用を自動更新したり、より高度なスクリプティングを使用してユーザーのロケールに合った外観にフォームを修正したりできます。

Designer ES2 では演算を作成しやすくするため、FormCalc が提供されています。FormCalc は Adobe が開発したわかりやすい演算言語で、一般的な表計算アプリケーションをモデルとしています。FormCalc は、単純でスクリプティングをあまり経験したことのない開発者にとっても使用しやすいものです。また、その他のスクリプティング言語にも共通するルールや表記規則に準拠しているため、経験豊富なフォーム開発者もそれまでのスキルを生かして FormCalc を使用できます。

本マニュアルの内容

これは、Designer ES2 を使用し、FormCalc 演算をフォームに組み込みたいと考えているフォーム開発者向けのガイドです。このガイドは FormCalc 関数のリファレンスであり、関数の分類に従って章別に編成されています。また、FormCalc 言語と、FormCalc の式を構成する構築ブロックの概要について説明しています。

対象読者

本マニュアルには、Designer ES2 で作成したフォームデザインの機能を高める演算を作成するために、FormCalc 言語を使用したいフォーム開発者の一助となる情報が含まれています。

関連マニュアル

フォームでの FormCalc 演算の使用について詳しくは、Designer ES2 ヘルプの「計算とスクリプトの作成」を参照してください。

FormCalc の技術情報については、http://partners.adobe.com/public/developer/xml/index_arch.html の『Adobe XML Forms Architecture (XFA) Specification, version 2.4』を参照してください。

1. FormCalc の概要

FormCalc は、一般的な表計算ソフトウェアをモデルとした簡単かつ強力な演算言語です。FormCalc の目的は、従来のスクリプティングテクニックや言語の知識がなくても、すばやくかつ効果的なフォームデザインを容易にすることです。エンドユーザーに時間のかかる演算、検証、その他の認証の手間を取らせないフォームを、FormCalc に馴染みのないユーザーでもいくつかの組み込み関数を使用してすばやく作成することが可能です。FormCalc を使用すると、フォーム開発者がフォームのデザイン時に周辺の基本的な情報を作成できるため、入力されるデータに合わせて処理を行うインタラクティブフォームを作成できます。

FormCalc を構成する組み込み関数が扱うデータは、数学、日時、文字列、会計、論理、Web など、広範囲な領域にわたります。これらはフォームでよく使用されるデータの種類であり、FormCalc の関数を使用すると実用的な方法でこれらのデータを迅速かつ簡単に処理することができます。

[6 ページ「Designer ES2 でのスクリプティングについて」](#)

[26 ページ「アルファベット順関数リスト」](#)

Designer ES2 でのスクリプティングについて

Designer ES2 では、すべてのスクリプト記述場所で FormCalc がデフォルトのスクリプティング言語となっています。ただし、JavaScript™ を選択することもできます。スクリプティングは各フォームオブジェクトに付随するさまざまなイベントで起こり、インタラクティブフォームで FormCalc と JavaScript を混合して使用できます。ただし、サーバーベースの処理（Forms ES2 など）を使用してインターネットブラウザで表示するフォームを作成している場合、特定のフォームオブジェクトイベントにある FormCalc スクリプトは HTML フォームにレンダリングされません。これは、ユーザーが完成フォームで作業するとき、インターネットブラウザのエラーが発生しないようにするための機能です。

2. 構築ブロック

FormCalc 言語は、FormCalc の式を構成する多数の構築ブロックから成り立っています。FormCalc の各式は、この構築ブロックを組み合わせたシーケンスです。

- [7 ページ「リテラル」](#)
- [9 ページ「演算子」](#)
- [9 ページ「コメント」](#)
- [10 ページ「キーワード」](#)
- [11 ページ「識別子」](#)
- [11 ページ「行終端子」](#)
- [11 ページ「空白」](#)

リテラル

リテラルは、処理のために FormCalc に渡されるすべての値の基となる定数値です。リテラルの 2 つの一般的な型は数値と文字列です。

数値リテラル

数値リテラルは、整数、小数点、小数部、指数（「e」または「E」）およびオプションの符号付き指数値のうち 1 つまたは複数から成る、数字中心のシーケンスです。以下はすべてリテラル数の例です。

- -12
- 1.5362
- 0.875
- 5.56e-2
- 1.234E10

リテラル数では、整数部または小数部のいずれかを省略できますが、両方を省略することはできません。また、小数部内では、小数点または指数値のいずれかを省略できますが、両方を省略することはできません。

数値リテラルはすべて、内部的に Institute of Electrical and Electronics Engineers (IEEE) の 64 ビットバイナリ値に変換されます。ただし、IEEE 値では有限数しか表現できないので、一部の値は 2 進小数としての表現を持ちません。これは、1/3 などの値を小数として正確に表現できないことに似ています（10 進値で完全に正確に表現するには、無限の小数桁数が必要です）。

このような、相当する 2 進小数を持たない値のセットにあたるのは、一般的に、指数の前に 16 を超える有効数字を持つ数値リテラルです。FormCalc は、IEEE 標準に従って、これらの値を表現可能な最も近い IEEE 64 ビット値に四捨五入します。例えば、次の値。

```
123456789.012345678
```

は、（最も近い）次の値に四捨五入されます。

```
123456789.01234567
```

一方、次の数値リテラル。

```
9999999999999999
```

は、（最も近い）次の値に四捨五入されます。

```
10000000000000000
```


演算子

FormCalc には、単項、乗法、加法、大小関係、等価、論理の各演算子や、割り当て演算子など、多数の演算子があります。

FormCalc の演算子のいくつかは、等価なニーモニックキーワードを持っています。このキーワード演算子は、HTML および XML ソーステキストに FormCalc の式を埋め込む際、そのソーステキストでより小さい (<)、より大きい (>) およびアンパサンド (&) の各記号が事前定義された意味を持っていて、それらをエスケープしなければならない場合に便利です。次の表に、FormCalc のすべての演算子を、シンボル形式と、必要に応じてニーモニック形式で示します。

演算子のタイプ	表現
加算	+
除算	/
等価	== eq <> ne
論理積	& and
論理和	or
乗算	*
関係	< lt (より小さい) > gt (より大きい) <= le (より小さいまたは等しい) >= ge (より大きいまたは等しい)
減算	-
単項	- + not

コメント

コメントは、FormCalc によって実行されないコードセクションです。通常、コメントには、特定のコードフラグメントの用途を説明した情報や指示が含まれています。コメントに記述した情報は、実行時にはすべて無視されます。

コメントは、セミコロン (;)、またはスラッシュのペア (//) を使用して指定できます。FormCalc では、コメントはその先頭から次の行終端子まで続きます。

キャラクタ名	表現
コメント	; //

例：

```
// これはコメントのタイプの 1 つです
First_Name="Tony"
Initial="C" ; これは別のタイプのコメントです
Last_Name="Blue"
```

イベント上のすべての FormCalc 演算のコメント化

特定のイベントについてすべての FormCalc の演算をコメント化していると、フォームを「PDF プレビュー」タブでプレビューしたとき、または最終の PDF を表示したときに、エラーが発生します。値を返すには、それぞれ FormCalc の演算が必要ですが、FormCalc ではコメントは値とは見なされません。

コメントした FormCalc コードがエラーを返さないようにするには、次のいずれかの操作を行います。

- イベントからコメント化されたコードを削除します。
- イベント上で FormCalc コードに値を返す式を追加します。
式の値がフォーム上で望ましくない結果とならないようにするには、次のいずれかの形式の式を使用します。
- 次の例のように、1 文字から成る単純式。

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// 以下の単純式はイベントの値を 0 に設定します。
0
```

- オブジェクトの値を保持する割り当て式。この形式の式は、次の例のように、コメント化された FormCalc コードが calculate イベントにある場合に実際のオブジェクトの値が置き換えられるのを防ぐために使用します。

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// 以下の割り当て式では、現在のフィールドの値を
// それ自体に設定します。
$.rawValue = $.rawValue
```

キーワード

FormCalc 内のキーワードは予約済みの語であり、大文字と小文字が区別されます。キーワードは、式の一部、特殊な数値リテラルおよび演算子として使用されます。

次の表に、FormCalc キーワードを示します。フォームデザインのオブジェクトに名前を付けるときは、これらの語を使用しないでください。

and	endif	in	step
break	endwhile	infinity	then
continue	eq	le	this
do	exit	lt	throw
downto	for	nan	upto
else	foreach	ne	var
elseif	func	not	while
end	ge	null	
endfor	gt	または	
endfunc	if	return	

識別子

識別子は、関数名またはメソッド名を表す、長さが無制限の文字シーケンスです。各識別子は、次のいずれかの文字から始まっている必要があります。

- (ユニコード文字の分類に基づく) アルファベット文字
- アンダースコア (_)
- ドル記号 (\$)
- 感嘆符 (!)

FormCalc の識別子は、大文字と小文字が区別されます。つまり、文字の大文字小文字だけが違っている識別子も、別の識別子と見なされます。

キャラクタ名	表現
識別子	A..Z、a..z \$! _

以下は有効な識別子の例です。

```
GetAddr  
$primary  
_item  
!dbresult
```

行終端子

行終端子は、行を分けて読みやすくするために使用されます。

次の表に、FormCalc の有効な行終端子を示します。

キャラクタ名	ユニコード文字
復帰	#xD U+000D
改行	#xA  &#D;

空白

空白文字は、各種のオブジェクト、算術演算子および算術オブジェクトを相互に区切る文字です。空白文字は読みやすさを向上させるためのもので、FormCalc の処理には関係がありません。

キャラクタ名	ユニコード文字
改ページ	#xC
水平タブ	#x9
Space	#x20
垂直タブ	#xB

数式

リテラル、演算子、コメント、キーワード、識別子、行終端子および空白が一緒になって、式のリストが形成されます。式のリストには 1 つの式しか含まれていなくても構いません。一般的に、リストのそれぞれの式は 1 つの値にリゾルブされます。また、リストの最後の式の値がリスト全体の値になります。

例えば、フォームデザインにある 2 つのフィールドの以下のシナリオを考えてください。

フィールド名	演算	戻り値
Field1	5 + Abs(Price) "Hello World" 10 * 3 + 5 * 4	50
Field2	10 * 3 + 5 * 4	50

Field1 と Field2 の式のリストを評価した後のフィールドの値は、両方とも 50 です。

FormCalc では、式のリストを構成する各種の式を次のカテゴリに分けています。

- [12 ページ「単純」](#)
- [14 ページ「割り当て」](#)
- [14 ページ「論理和」](#)
- [14 ページ「論理積」](#)
- [15 ページ「単項」](#)
- [15 ページ「等価および不等価」](#)
- [16 ページ「関係」](#)
- [17 ページ「if 式」](#)
- [17 ページ「While 式」](#)
- [18 ページ「For 式」](#)
- [18 ページ「Foreach 式」](#)
- [19 ページ「Break 式」](#)
- [19 ページ「Continue 式」](#)

単純

最も基本的なフォームでは、FormCalc の式は、論理的な方法で文字列化された演算子、キーワードおよびリテラルの集まりになっています。例えば、以下はすべて単純式です。

```
2
"abc"
2 - 3 * 10 / 2 + 7
```

FormCalc の各式は、演算子の順序が式の構文から必ずしも明らかでなくても、演算子の従来の順序に従って単一の値にリゾルブされます。例えば、以下の式のセットは、フォームデザインのオブジェクトに適用されたときに等価な結果になります。

式	等価な式	戻り値
"abc"	"abc"	abc
2 - 3 * 10 / 2 + 7	2 - (3 * (10 / 2)) + 7	-6
10 * 3 + 5 * 4	(10 * 3) + (5 * 4)	50
0 and 1 or 2 > 1	(0 and 1) or (2 > 1)	1 (true)
2 < 3 not 1 == 1	(2 < 3) not (1 == 1)	0 (false)

上の表から分かるように、FormCalc のすべての演算子には、式での優先順位があります。次の表に演算子の階層を示します。

優先順位	演算子
高	=
	(単項) -, +, not
	*, /
	+, -
	<, <=, >, >=, lt, le, gt, ge
	==, <>, eq, ne
	&, and
低	, or

演算値の型変換

指定された処理内の演算値がその処理の期待された型と一致しない場合、FormCalc は、必要な型と一致するように演算値を型変換します。この型変換がどのように行われるかは、処理で必要とされる演算値の型によって異なります。

数値の処理

数値以外の演算値が関係する数値処理を実行する場合、まず、数値以外の演算値が、相当する数値に型変換されます。数値でない演算値が正しく数値に変換されない場合、その値は 0 です。null 値の演算値を数値に型変換するとき、その値は常に 0 です。

次の表に、数値以外の演算値の型変換の例を示します。

式	等価な式	戻り値
(5 - "abc") * 3	(5 - 0) * 3	15
"100" / 10e1	100 / 10e1	1
5 + null + 3	5 + 0 + 3	8

Boolean 値の処理

Boolean 値以外の演算値に Boolean 処理を実行する場合、まず、Boolean 値以外の演算値が、相当する Boolean 値に型変換されます。Boolean 値以外の演算値がゼロ以外の値に正しく変換されない場合、演算値の値は true (1) です。正しく変換される場合は、演算値の値は false (0) です。null 値の演算値を Boolean 値に型変換させる場合、演算値の値は常に false (0) です。例えば、次の式

```
"abc" | 2
```

は、1 と評価されます。つまり、false | true = true で、一方、

```
if ("abc") then
  10
else
  20
endif
```

は、20 と評価されます。

文字列の処理

文字列以外の演算値に文字列処理を実行する場合、まず、文字列以外の演算値の値を文字列として使用することにより演算値が文字列に型変換されます。`null` 値の演算値を文字列に型変換させる場合、演算値の値は常に空文字列です。例えば、次の式。

```
concat("The total is ", 2, " dollars and ", 57, " cents.")
```

は、"The total is 2 dollars and 57 cents." と評価されます。

注意：数式の評価中、中間ステップとして NaN、+Inf、-Inf のいずれかが出た場合、FormCalc ではエラー例外が生成され、数式の残りの部分にそのエラーを伝えます。したがって、式の値は常に 0 になります。例えば、

```
3 / 0 + 1
```

は、0 と評価されます。

割り当て

割り当て式は、指定された参照構文によって識別されるプロパティを単純式の値に設定します。例：

```
$template.purchase_order.name.first = "Tony"
```

これは、フォームデザインのオブジェクト「first」の値を Tony に設定します。

参照構文の使用について詳しくは、[20 ページ「参照構文」](#)を参照してください。

論理和

論理和式は、少なくとも 1 つの演算値が true (1) の場合は true (1) を、両方の演算値が false (0) の場合は false (0) を返します。オペランドが両方とも null の場合、式は null を返します。

式	文字表現
論理和	 or

以下に、論理和式の使用例を示します。

式	戻り値
1 or 0	1 (true)
0 0	0 (false)
0 or 1 0 or 0	1 (true)

論理積

論理積式は、両方の演算値が true (1) の場合は true (1) を、少なくとも 1 つの演算値が false (0) の場合は false (0) を返します。オペランドが両方とも null の場合、式は null を返します。

式	文字表現
論理積	& and

構築ブロック

以下に、論理積式の使用例を示します。

式	戻り値
1 and 0	0 (false)
0 & 0	1 (true)
0 and 1 & 0 and 0	0 (false)

単項

単項式は、どの単項演算子を使用されるかによって異なる結果を返します。

式	文字表現	戻り値
単項	-	演算値の算術否定、または演算値が null の場合は null。
	+	演算値の算術値（変更なし）、または演算値が null の場合は null。
	not	演算値の論理否定。

注意： null 演算値の算術否定は null を返し、null 演算値の論理否定は Boolean 値の true を返します。これは、次の常識的なステートメントによって正当化されます。つまり、null が何もないことを意味する場合、「何もないことはない」とは何かがあるという意味になります。

以下に、単項式の使用例を示します。

式	戻り値
-(17)	-17
-(-17)	17
+(17)	17
+(-17)	-17
not("true")	1 (true)
not(1)	0 (false)

等価および不等価

等価および不等価式は、演算値の等価比較結果を返します。

式	文字表現	戻り値
等価	== eq	両方の演算値の比較結果が等しくない場合は true (1)、等しい場合は false (0)。
不等価	<> ne	両方の演算値の比較結果が等しくない場合は true (1)、等しい場合は false (0)。

以下のような特別な場合にも等価演算子を使用できます。

- いずれかの演算値が null の場合。この場合、null の比較が実行されます。null 値の演算値は、両方の演算値が null のときは等しい結果になり、一方の演算値が null でないときは等しくないという結果になります。
- 両方の演算値が参照の場合。この場合、両方の演算値が同じオブジェクトを参照するときは等しい結果になり、両方の演算値が同じオブジェクトを参照しないときは等しくないという結果になります。
- 両方の演算値が文字列値の場合。この場合、ロケールに依存する辞書式の文字列比較が演算値で実行されます。文字列の比較が実行されない場合、両方もが null でなければ、演算値は数値に型変換され、数値比較が実行されます。

構築ブロック

以下に、等価式と不等価式の使用例を示します。

式	戻り値
3 == 3	1 (true)
3 <> 4	1 (true)
"abc" eq "def"	0 (false)
"def" ne "abc"	1 (true)
5 + 5 == 10	1 (true)
5 + 5 <> "10"	0 (false)

関係

関係式は、演算値の比較結果を Boolean 値で返します。

式	文字表現	戻り値
関係	< lt	1つ目の演算値が2つ目の演算値より小さいときは true (1)、1つ目の演算値が2つ目の演算値より大きいときは false (0)。
	> gt	1つ目の演算値が2つ目の演算値より大きいときは true (1)、1つ目の演算値が2つ目の演算値より小さいときは false (0)。
	<= le	1つ目の演算値が2つ目の演算値より小さいまたは等しいときは true (1)、1つ目の演算値が2つ目の演算値より大きいときは false (0)。
	>= ge	1つ目の演算値が2つ目の演算値より大きいまたは等しいときは true (1)、1つ目の演算値が2つ目の演算値より小さいときは false (0)。

以下のような特別な場合にも関係演算子を使用できます。

- いずれかの演算値が null 値の場合。この場合、null の比較が実行されます。null 値の演算値は、両方の演算値が null で、関係演算子がより小さいか等しい、またはより大きい等しいときは等しい結果になり、それ以外のときは等しくないという結果になります。
- 両方の演算値が文字列値の場合。この場合、ロケールに依存する辞書式の文字列比較が演算値で実行されます。文字列の比較が実行されない場合、両方もが null でなければ、演算値は数値に型変換され、数値比較が実行されます。

以下に、関係式の使用例を示します。

式	戻り値
3 < 3	0 (false)
3 > 4	0 (false)
"abc" <= "def"	1 (true)
"def" > "abc"	1 (true)
12 >= 12	1 (true)
"true" < "false"	0 (false)

if 式

if 式は、与えられた単純式が真かどうか評価し、真の値に対応する式のリストの結果を返す条件ステートメントです。最初の単純式が false (0) と評価される場合、FormCalc は elseif および else 条件が真かどうか調べ、適切な場合は式のリストの結果を返します。

式	構文	戻り値
if	<pre>if (単純式) then 式のリスト elseif (単純式) then 式のリスト else 式のリスト endif</pre>	<p>if 式に記述されている有効条件と関連付けられた式のリストの結果。</p> <p>elseif(...) ステートメントや else ステートメントを if 式の中で使用することは必須ではありませんが、endif を使用して式の終わりを示すことは必須です。</p>

以下に、if 式の使用例を示します。

式	戻り値
<pre>if (1 < 2) then 1 endif</pre>	1
<pre>if ("abc" > "def") then 1 and 0 else 0 endif</pre>	0
<pre>if (Field1 < Field2) then Field3 = 0 elseif (Field1 > Field2) then Field3 = 40 elseif (Field1 == Field2) then Field3 = 10 endif</pre>	Field1 と Field2 の値によって変わります。例えば、Field1 が 20、Field2 が 10 の場合、この数式は Field3 を 40 に設定します。

While 式

while 式は、与えられた単純式を評価するインタラクティブなステートメントまたはループです。評価の結果が真 (1) の場合、FormCalc は do 条件を繰り返し調べて、式のリストの結果を返します。結果が false (0) の場合、制御は次のステートメントに渡されます。

while 式は、条件の反復が必要な場合に特に適しています。逆に、無条件の反復が必要な状況ではたいていの場合、for 条件を使用するのが最適です。

式	構文	戻り値
While	<pre>while (単純式) do 式のリスト endwhile</pre>	do 条件に関連付けられた式のリストの結果です。

次の例では、3 ではない list1 にリストされた XML 要素のすべてに addItem メソッドを使用して、XML ファイルからコンボボックスへ要素の値が追加されます。

```
var List = ref(xfa.record.lists.list1)
var i = 0
while (List.nodes.item(i+1).value ne "3") do
$.addItem (List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
endwhile
```

For 式

for 式は、条件付きのインタラクティブなステートメントまたはループです。

for 式は、無条件の反復が必要なループの場合に特に適しています。逆に、条件の反復が必要な状況では `while` 条件を使用するのが最適です。

for 式の値は、評価された最後の評価リストの値、または `false` (0) です。

for 条件は、ループ操作を制御する FormCalc 変数を初期化します。

`upto` 変数では `step` 式を増加させながら、`start` 式から `end` 式までをループ変数の値だけ繰り返します。`step` 式を省略すると、`step` は 1 をデフォルトにして増加します。

`downto` 変数では `step` 式を減少させながら、`start` 式から `end` 式までをループ変数の値だけ繰り返します。`step` 式を省略すると、`step` は -1 をデフォルトにして減少します。

ループの反復は、`end` 式の値によって制御されます。各反復の前に `end` 式が評価され、ループ変数と比較されます。結果が真 (1) の場合、式のリストは評価されます。各評価が終わると `step` 式が評価され、ループ変数に追加されます。

各反復の前に `end` 式が評価され、ループ変数と比較されます。さらに、`do` 式の各評価が終わると `step` 式が評価され、ループ変数に追加されます。

`start` 式が `end` 式を超えると for のループは終了します。`start` 式が `end` 式を超えることも可能です。`upto` を使用してプラス方向に、`downto` を使用してマイナス方向に大きくすることができます。

式	構文	戻り値
For	<pre>for variable = start 式 (upto downto) end 式 (step step 式) do 式のリスト endfor</pre> <p><code>start</code>、<code>end</code> および <code>step</code> 式は、すべて単純式の必要があります。</p>	do 条件に関連付けられた式のリストの結果です。

次の例では、`list1` にリストされた XML 要素のすべてに `addItem` メソッドを使用し、XML ファイルからコンボボックスへ要素の値が追加されます。

```
var List = ref(xfa.record.lists.list1)
for i=0 upto List.nodes.length - 1 step 2 do
$.addItem (List.nodes.item(i).value, "")
endfor
```

Foreach 式

foreach 式は、引数リストにある各値に式のリストを繰り返し実行します。

ループに何も入力しないと、foreach 式の値は、評価された最後の式のリストにある値、またはゼロ (0) です。

(ループ変数が宣言された後) 1 度だけ実行される `in` 条件によって、ループの反復が制御されます。各反復を行う前に、ループ変数に引数のリストから一連の値が割り当てられます。引数のリストには必ず値が必要です。

式	構文	戻り値
Foreach	<pre>foreach variable in(引数のリスト)do 式のリスト endfor</pre> <p>引数のリストに複数の単純式がある場合は、コンマ (,) を使用して区切ります。</p>	ループが入力されなかった場合は、評価された最後のリストの値、またはゼロ (0) です。

構築ブロック

次の例では、「display」XML 要素の値だけが foreach コンボボックスに追加されます。

```
foreach Item in (xfa.record.lists.list1.display[*]) do
$.addItem(Item, "")
endfor
```

Break 式

break 式により、while、for または foreach 式のループを含む最も深い部分から、即座に終了が実行されます。制御は、ループの終端の後にある式に渡されます。

break 式の値は常にゼロ (0) 値です。

式	構文	戻り値
Break	break	制御を、ループ終端の後にある式に渡します。

次の例では、現在の値が「Display data for 2」に等しいかどうか調べるために、while ループ内に if 条件があります。真の場合、break が実行されてループの継続が停止します。

```
var List = ref(xfa.record.lists.list1)
var i=0
while (List.nodes.item(i+1).value ne "3") do
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
if (List.nodes.item(i) eq "Display data for 2" then
break
endif
endwhile
```

Continue 式

continue 式は、while、for または foreach のループを含む最も深い部分の次の反復を継続させます。

continue 式の値は常にゼロ (0) 値です。

式	構文	戻り値
Continue	continue	while 式で使用されると、制御は while 条件に渡されます。for 式で使用されると、制御は step 式に渡されます。

次の例の目的は、コンボボックスを XML ファイルの値と共に埋め込むことです。現在の XML 要素の値が「Display data for 3」の場合、while のループは break 式を経て終了します。現在の XML 要素の値が「Display data for 2」の場合、スクリプトは変数 i (カウンターです) に 2 を加え、ループは即座に次のサイクルへ移動します。最後の 2 行は、現在の XML 要素の値が「Display data for 2」の場合は無視されます。

```
var List = ref(xfa.record.lists.list1)
var i = 0
while (List.nodes.item(i+1).value ne "5") do
if (List.nodes.item(i) eq "Display data for 3") then
break
endif
if (List.nodes.item(i) eq "Display data for 2" then
i=i+2
continue
endif
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i=i+2
endwhile
```

変数

FormCalc の演算では、データを保存するための変数を作成したり処理したりできます。作成する各変数に割り当てる名前は、それぞれ異なる識別子である必要があります。

例えば、次の FormCalc 数式は `userName` 変数を定義し、`userName` の値となるテキストフィールドの値を設定します。

```
var userName = "Tony Blue"
TextField1.rawValue = userName
```

フォームのプロパティダイアログボックスの「変数」タブで定義する変数も同様に参照できます。次の FormCalc 数式では、Concat 関数を使用してフォームの変数である `salutation` と `name` を使用したテキストフィールドの値を設定します。

```
TextField1.rawValue = Concat(salutation, name)
```

注意：FormCalc を使用して作成する変数と同様の名前の変数が、フォームのプロパティダイアログボックスの「変数」タブで定義されている場合、FormCalc で作成した変数が使用されます。

参照構文

FormCalc では、参照構文を使用して、フォームデザインのオブジェクトのプロパティと値にアクセスすることができます。次の例は、オブジェクトの値の割り当てと取得を示しています。

```
Invoice_Total.rawValue = Invoice_SubTotal.rawValue * (8 / 100)
```

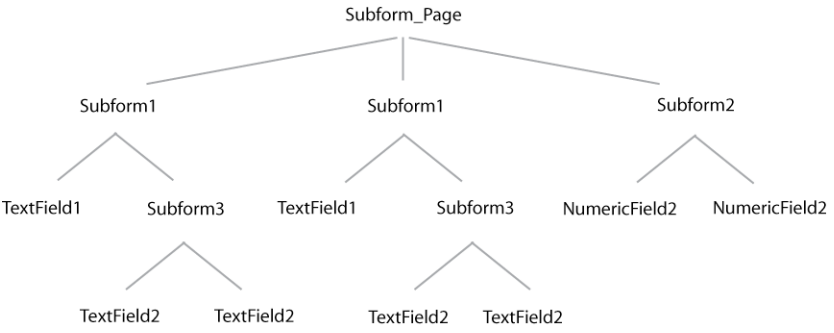
この場合、参照構文 `Invoice_Total` により、`Invoice_SubTotal * (8 / 100)` の値がフィールド `Invoice_Total` に割り当てられます。

フォームデザインのコンテキストでは、完全な参照構文がフォームデザインのすべてのオブジェクトにアクセスできます。

オブジェクトのプロパティに簡単にアクセスできるように、参照を作成する場合に煩雑さを省くためのショートカットが FormCalc に含まれています。FormCalc の参照構文のショートカットを次の表にまとめています。

表記法	説明
\$	次の例に示すように現在のフィールドまたはオブジェクトを参照します。 \$ = "Tony Blue" 上の例では現在のフィールドまたはオブジェクトの値を Tony Blue に設定します。
\$data	データモデル <code>xfa.datasets.data</code> のルートを表します。以下に例を示します。 \$data.purchaseOrder.total は、以下と等価です。 <code>xfa.datasets.data.purchaseOrder.total</code>
\$event	現在のフォームオブジェクトのイベントを表します。以下に例を示します。 \$event.name は、以下と等価です。 <code>xfa.event.name</code>
\$form	フォームモデル <code>xfa.form</code> のルートを表します。以下に例を示します。 \$form.purchaseOrder.tax は、次のステートメントと等価です。 <code>xfa.form.purchaseOrder.tax</code>

表記法	説明
\$host	ホストのオブジェクトを表します。以下に例を示します。 <pre>\$host.messageBox("Hello world")</pre> は、以下と等価です。 <pre>xfa.host.messageBox("Hello world")</pre>
\$layout	レイアウトモデル <code>xfa.layout</code> のルートを表します。以下に例を示します。 <pre>\$layout.ready</pre> は、次のステートメントと等価です。 <pre>xfa.layout.ready</pre>
\$record	XML ファイルといったデータコレクション内の現在のデータを表します。以下に例を示します。 <pre>\$record.header.txtOrderedByCity</pre> は、 <code>txtOrderedByCity</code> ノード（現在の XML データの <code>header</code> ノード内）を参照します。
\$template	テンプレートモデル <code>xfa.template</code> のルートを表します。以下に例を示します。 <pre>\$template.purchaseOrder.item</pre> は、以下と等価です。 <pre>xfa.template.purchaseOrder.item</pre>
!	データモデル <code>xfa.datasets</code> のルートを表します。以下に例を示します。 <pre>!data</pre> は、以下と等価です。 <pre>xfa.datasets.data</pre>
*	名前と無関係にサブフォームなどの任意のコンテナ内にあるフォームオブジェクトをすべて選択したり、類似の名前を持つオブジェクトをすべて選択したりします。 以下の式はフォームに <code>item</code> という名前を持つすべてのオブジェクトを選択します。 <pre>xfa.form.form1.item[*]</pre> <code>resolveNode</code> メソッドでは、 <code>*</code> （アスタリスク）構文を JavaScript と共に使用できます。 <code>resolveNode</code> メソッドについて詳しくは、Designer ES2 ヘルプ、または『 LiveCycle Designer ES スクリプティングリファレンス 』を参照してください。

表記法	説明
..	<p>参照構文の任意の場所に 2 つのピリオドを入力すると、現在のコンテナオブジェクトのサブコンテナを構成するオブジェクト (サブフォームなど) を検索できます。例えば、Subform_Page..Subform2 という式は、(通常どおりに) ノード Subform_Page を検索し、Subform_Page の子孫 Subform2 を検出することを意味します。</p>  <pre> graph TD SP[Subform_Page] --> S1_1[Subform1] SP --> S1_2[Subform1] SP --> S1_3[Subform1] SP --> S2[Subform2] S1_1 --> T1_1[TextField1] S1_1 --> S3_1[Subform3] S1_2 --> T1_2[TextField1] S1_2 --> S3_2[Subform3] S2 --> NF2_1[NumericField2] S2 --> NF2_2[NumericField2] S3_1 --> TF2_1[TextField2] S3_1 --> TF2_2[TextField2] S3_2 --> TF2_3[TextField2] S3_2 --> TF2_4[TextField2] </pre> <p>上記のサンプルツリーを使用すると、 <code>Subform_Page..TextField2</code> は、以下と等価です。 <code>Subform_Page.Subform1[0].Subform3.TextField2[0]</code> これは、<code>TextField2[0]</code> が、FormCalc が検索で遭遇する最初の <code>Subform1</code> ノードにあるからです。2 つ目の例として、 <code>Subform_Page..Subform3[*]</code> 4 つの <code>TextField2</code> オブジェクトをすべて返します。 resolveNode メソッドでは、<code>..</code> (二重ピリオド) 構文を JavaScript と共に使用できます。resolveNode メソッドについて詳しくは、Designer ES2 ヘルプ、または『LiveCycle Designer ES スクリプティングリファレンス』を参照してください。</p>
#	<p>番号記号 (#) 表記は、参照構文で次のいずれかの項目を表します。</p> <ul style="list-style-type: none"> • 名称未設定オブジェクト。例えば、次の参照構文は名称未設定のサブフォームにアクセスします。 <code>xfa.form.form1.#subform</code> • プロパティとオブジェクトが同じ名前の場合、参照構文でプロパティを指定します。例えば、サブフォームに <code>name</code> という名称のフィールドも含まれている場合、次の参照構文はサブフォームの <code>name</code> プロパティにアクセスします。 <code>xfa.form.form1.#subform.#name</code> <p>resolveNode メソッドでは、'#' (シャープ記号) 構文を JavaScript と共に使用できます。resolveNode メソッドについて詳しくは、Designer ES2 ヘルプ、または『LiveCycle Designer ES スクリプティングリファレンス』を参照してください。</p>

表記法	説明
[]	<p>角括弧 ([]) 表記はオブジェクトのオカレンス値を表します。オカレンス値の参照を作成するには、オブジェクト名の後ろに角括弧 ([]) を置き、括弧内に以下のいずれかの値を入れます。</p> <ul style="list-style-type: none"> [n] の n は、0 から始まる、オカレンスの絶対インデックス番号です。オカレンス値が範囲外の場合は何も値を返しません。以下に例を示します。 <pre>xfa.form.form1.#subform.Quantity[3]</pre> は、Quantity オブジェクトの 4 番目のオカレンスを参照します。 [+/- n] の n は、参照を作成するオブジェクトのオカレンスと相対的なオカレンスを示します。正の値を指定するとオカレンス値は大きくなり、負の値を指定するとオカレンス値は小さくなります。以下に例を示します。 <pre>xfa.form.form1.#subform.Quantity[+2]</pre> <p>この参照では、参照を作成するコンテナのオカレンス値よりもオカレンス値が 2 だけ大きい Quantity の値が出力されます。例えば、この参照を Quantity[2] オブジェクトに追加すると、参照は以下と同じになります。</p> <pre>xfa.template.Quantity[4]</pre> <p>計算されたインデックス番号が範囲外の場合は、エラーを返します。</p> <p>この構文は、特定のオブジェクトの前または次のオカレンスを検索するときに最も一般的に使用します。例えば、Quantity オブジェクトのオカレンス (1 つ目は除く) はすべて、Quantity[-1] を利用して前の Quantity オブジェクトの値を得ます。</p> <ul style="list-style-type: none"> [*] はオブジェクトの複数のオカレンスを示します。名前が付いた最初のオブジェクトが検出され、そのオブジェクトの兄弟で同名のオブジェクトが返されます。この表記法を使用すると、オブジェクトのコレクションが返されます。以下に例を示します。 <pre>xfa.form.form1.#subform.Quantity[*]</pre> この式は、参照で検出された最初の Quantity の兄弟で、Quantity という名前を持つすべてのオブジェクトを参照します。 <p>アラビア語、ヘブライ語、タイ語およびベトナム語に特化したフォームでは、(右から左に表記する言語であっても) 参照構文が常に右側にあります。</p>

表記法	説明
[] (続き)	<div style="text-align: center;"> <pre> graph TD SP[Subform_Page] --> S1[Subform1] SP --> S2[Subform1] SP --> S3[Subform2] S1 --> T1[TextField1] S1 --> S3_1[Subform3] S2 --> T2[TextField1] S2 --> S3_2[Subform3] S3 --> NF1[NumericField2] S3 --> NF2[NumericField2] S3_1 --> TF2_1[TextField2] S3_1 --> TF2_2[TextField2] S3_2 --> TF2_3[TextField2] S3_2 --> TF2_4[TextField2] </pre> </div> <p>上記のツリーを参照した場合、以下の式が返すオブジェクトはそれぞれ次のとおりです。</p> <ul style="list-style-type: none"> • <code>Subform_Page.Subform1[*]</code> は両方の <code>Subform1</code> オブジェクトを返します。 • <code>Subform_Page.Subform1.Subform3.TextField2[*]</code> は、2つの <code>TextField2</code> オブジェクトを返します。 <code>Subform_Page.Subform1</code> は左側の最初の <code>Subform1</code> オブジェクトに解決され、<code>TextField2[*]</code> は <code>Subform3</code> オブジェクトを基準にして評価されます。 • <code>Subform_Page.Subform1[*].TextField1</code> は両方の <code>TextField1</code> インスタンスを返します。 <code>Subform_Page.Subform1[*]</code> は両方の <code>Subform1</code> オブジェクトに解決され、<code>TextField1</code> は <code>Subform1</code> オブジェクトを基準にして評価されます。 • <code>Subform_Page.Subform1[*].Subform3.TextField2[1]</code> は、左から2番目と4番目の <code>TextField2</code> オブジェクトを返します。 <code>Subform_Page.Subform1[*]</code> は両方の <code>Subform1</code> オブジェクトに解決され、<code>TextField2[1]</code> は <code>Subform3</code> オブジェクトを基準にして評価されます。 • <code>Subform_Page.Subform1[*].Subform3[*]</code> は両方の <code>Subform3</code> オブジェクトのインスタンスを返します。 • <code>Subform_Page.*</code> は <code>Subform1</code> オブジェクトおよび <code>Subform2</code> オブジェクトを返します。 • <code>Subform_Page.Subform2.*</code> は <code>NumericField2</code> オブジェクトの2つのインスタンスを返します。 <p><code>resolveNode</code> メソッドでは、'[]' (角括弧) 構文を JavaScript と共に使用できます。<code>resolveNode</code> メソッドについては詳しくは、LiveCycle Designer ES ヘルプ、または『LiveCycle Designer ES スクリプティングリファレンス』を参照してください。</p>

プロパティとメソッド呼び出し

Designer ES2には、フォームデザインのすべてのオブジェクトのためのさまざまなプロパティとメソッドが定義されています。FormCalcでは、これらのプロパティとメソッドにアクセスすることができ、プロパティやメソッドを使用してフォーム上のオブジェクトの外観と動作を変更することができます。関数呼び出しと同様に、プロパティとメソッドは、特定の順序でプロパティとメソッドに引数を渡すことによって起動します。各プロパティと各メソッドの引数の数と型は、オブジェクトのタイプによって異なります。

注意: フォームデザインのオブジェクトが異なると、サポートされるプロパティとメソッドも異なります。サポートされるプロパティとメソッドオブジェクトの一覧については、『[Designer ES2 スクリプティングリファレンス](#)』を参照してください。

組み込み関数の呼び出し

FormCalcは、幅広い機能を持つ組み込み関数の大きなセットをサポートしています。関数名は大文字小文字は区別されませんが、キーワードと異なり、FormCalcによって予約されている関数名はありません。つまり、FormCalcの関数名と名前が一致するオブジェクトを持ったフォーム上で計算を行っても衝突は起こりません。

関数に応じて、値を実行して返すのに引数のセットが必要な場合もあれば、必要ない場合もあります。関数の多くは、オプションの引数を持っています。つまり、特定の状況で引数が必要かどうかはユーザーが決定します。

FormCalcは、先頭の引数から始めて、関数のすべての引数を順番に評価します。必要な数よりも少ない数の引数を関数に渡そうとすると、関数からエラー例外が生成されます。

各関数は、それぞれの引数が特定の形式（数値リテラルまたは文字列リテラル）になっていることを期待します。引数の値が関数が期待する形式と一致しない場合、FormCalc によって値の変換が行われます。次に例を示します。

```
Len (35)
```

Len 関数では、リテラル文字列が前提となっています。この場合、FormCalc は引数を数値の 35 から文字列の「35」に変換し、関数は 2 と評価されます。

ただし、文字列リテラルを数値リテラルに変換する場合は、それほど単純ではありません。次に例を示します。

```
Abs ("abc")
```

Abs 関数では、数値リテラルが前提となっています。FormCalc は、すべての文字列リテラルの値を 0 として変換します。これは、**Apr** 関数の場合など、0 の値を入力すると強制的にエラーとなる関数で問題を引き起こす可能性があります。

関数の引数の中には整数値のみを必要とするものもあります。そのような場合、渡された引数は小数部が切り捨てられて常に整数値に変換されます。

組み込み関数の重要な特性をまとめます。

- 組み込み関数の名前は大文字小文字は区別されません。
- 組み込み関数は事前に定義されているものですが、その名前は予約された語ではありません。つまり、組み込み関数 **Max** は、Max という名前のオブジェクト、オブジェクトプロパティまたはオブジェクトメソッドと矛盾しないことを意味します。
- 組み込み関数の多くは必要な引数の数が決められていますが、その引数に続けて任意の数の引数を指定することができます。
- 組み込み関数 **Avg**、**Count**、**Max**、**Min**、**Sum** および **Concat** は、引数をいくつでも受け取ることができます。

FormCalc のすべての関数の完全なリストについては、[26 ページ「アルファベット順関数リスト」](#)を参照してください。

3. アルファベット順関数リスト

下記のテーブルには、使用可能なすべての FormCalc 関数、個々の関数の説明、およびそれぞれの関数が属するカテゴリの種類が記載されています。

関数	説明	種類
29 ページ 「Abs」	数値または数式の絶対値を返します。	演算
52 ページ 「Apr」	年間のローン利率を百分率で返します。	会計
67 ページ 「At」	別の文字列中に存在する、任意の文字列の開始文字位置を特定します。	文字列
29 ページ 「Avg」	セットの数値や数式の値を求め、そのセット内の null でない要素の平均を返します。	演算
30 ページ 「Ceil」	渡された数値以上の自然数を返します。	演算
60 ページ 「Choose」	渡されたパラメーターセットから値を選択します。	論理
68 ページ 「Concat」	複数の文字列を連結して返します。	文字列
30 ページ 「Count」	セットの値や数式の値を求め、そのセット内の null でない要素の数を返します。	演算
53 ページ 「CTerm」	将来増額する固定された複利率を補うための投資に必要な期間の値を返します。	会計
43 ページ 「Date」	現在のシステム日付を、 36 ページ 「エポック」からの日数として返します。	日付と時間
44 ページ 「Date2Num」	日付文字列を渡すと、 36 ページ 「エポック」以降の日数を返します。	日付と時間
44 ページ 「DateFmt」	日付形式スタイルが渡されると、日付形式文字列を返します。	日付と時間
68 ページ 「Decode」	渡された文字列をデコードして返します。	文字列
69 ページ 「Encode」	渡された文字列をエンコードして返します。	文字列
63 ページ 「Eval」	渡されたフォームの演算後の値を返します。	その他
60 ページ 「Exists」	渡されたパラメーターが既存オブジェクトへの参照構文かどうかを判定します。	論理
31 ページ 「Floor」	渡された値以下の最大整数値を返します。	演算
69 ページ 「Format」	指定したピクチャ形式文字列に従って、渡されたデータをフォーマットします。	文字列
53 ページ 「FV」	一定の利率で一定期間に返済されるときの将来の金額を返します。	会計
79 ページ 「Get」	渡された URL のコンテンツをダウンロードします。	URL
61 ページ 「HasValue」	渡されたパラメーターが null、空、空文字列でない値を持つアクセッサーかどうか判定します。	論理
54 ページ 「IPmt」	一定期間のローンに対して支払われる利子の額を返します。	会計
45 ページ 「IsoDate2Num」	有効な日付文字列が渡されると、 エポック 以降の日数を返します。	日付と時間
46 ページ 「IsoTime2Num」	有効な時間文字列が渡されると、 エポック 以降のミリ秒数を返します。	日付と時間
70 ページ 「Left」	左側の最初の文字から指定された文字数を抽出します。	文字列
71 ページ 「Len」	渡された文字列の文字数を返します。	文字列
46 ページ 「LocalDateFmt」	日付形式スタイルが渡されると、ローカライズされた日付形式文字列を返します。	日付と時間
47 ページ 「LocalTimeFmt」	時間形式スタイルに対してローカライズされた時間形式文字列を返します。	日付と時間
71 ページ 「Lower」	指定された文字列内の大文字をすべて小文字に変換します。	文字列
72 ページ 「Ltrim」	文字の前にある空白文字をすべて削除した文字列を返します。	文字列

関数	説明	種類
31 ページ「Max」	渡された数値セットの null でない要素の最大値を返します。	演算
32 ページ「Min」	渡された数値セットの null でない要素の最小値を返します。	演算
33 ページ「Mod」	数値を別の数値で割り算したときの係数を返します。	演算
55 ページ「NPV」	割引率と各支払いの将来のキャッシュフローに基づいた、投資の正味現在価値を返します。	会計
63 ページ「Null」	null 値を返します。null 値は値がないことを意味します。	その他
47 ページ「Num2Date」	エポック 以降の日数を渡すと、日付文字列を返します。	日付と時間
48 ページ「Num2GMTime」	エポック 以降のミリ秒数に対して GMT 時間文字列を返します。	日付と時間
48 ページ「Num2Time」	エポック 以降のミリ秒数を渡すと、時間文字列を返します。	日付と時間
61 ページ「Oneof」	値が渡されたセット内にある場合は true (1) を返し、ない場合は false (0) を返します。	論理
72 ページ「Parse」	指定されたピクチャ形式に従って、渡されたデータを分析します。	文字列
55 ページ「Pmt」	一定の返済額および一定の利率に基づき、ローン返済額を返します。	会計
79 ページ「Post」	渡されたデータを指定した URL にポストします。	URL
56 ページ「PPmt」	一定期間のローンに対して返済された元本の額を返します。	会計
80 ページ「Put」	渡されたデータを指定した URL にアップロードします。	URL
57 ページ「PV」	一定の利率で定期的に定額返済される融資の現価を返します。	会計
57 ページ「Rate」	渡された期間内で、現在から将来にかけて増加する融資額に対して要求される期間における複利を返します。	会計
64 ページ「Ref」	既存のオブジェクトの参照を返します。	その他
73 ページ「Replace」	指定された文字列内で、任意の文字列を別の文字列ですべて置換します。	文字列
73 ページ「Right」	右側の最後の文字から渡された文字数を抽出します。	文字列
33 ページ「Round」	渡された数値または数式の値を求め、渡された小数点の位で四捨五入した値を返します。	演算
74 ページ「Rtrim」	文字の後ろにある空白文字をすべて削除した文字列を返します。	文字列
74 ページ「Space」	渡された数の空スペースで構成される文字列を返します。	文字列
75 ページ「Str」	数字を文字列へ変換します。FormCalc は変換結果を指定された幅にフォーマットし、指定された小数位に数を四捨五入します。	文字列
75 ページ「Stuff」	文字列を別の文字列へ挿入します。	文字列
76 ページ「Substr」	渡された文字列の一部を抽出します。	文字列
34 ページ「Sum」	渡された数値セットの null でない要素の合計を返します。	演算
58 ページ「Term」	有利子負債に対して定期的に定額返済した場合、渡された将来の額に達するのに必要な期間数を返します。	会計
49 ページ「Time」	エポック 以降のミリ秒数として現在のシステム時刻を返します。	日付と時間
49 ページ「Time2Num」	時間文字列が渡されると、エポック 以降のミリ秒数を返します。	日付と時間
50 ページ「TimeFmt」	時間形式スタイルに対して時間形式を返します。	日付と時間
64 ページ「UnitType」	ユニットスパンの単位を返します。ユニットスパンは、数値とそれに続く単位名を構成する文字列です。	その他
65 ページ「UnitValue」	オプションの単位に変換した後、測定の数値とそれに関連するユニットスパンを返します。	その他

アルファベット順関数リスト

関数	説明	種類
77 ページ 「Upper」	文字列内の小文字をすべて大文字に変換します。	文字列
77 ページ 「Uuid」	識別方法として使用される UUID (Universally Unique Identifier) 文字列を返します。	文字列
62 ページ 「Within」	テスト値が渡された範囲内にある場合は true (1) を返し、ない場合は false (0) を返します。	論理
78 ページ 「WordNum」	渡された数を表す英語テキストを返します。	文字列

4. 演算関数

ここで説明する関数を使用すると、数学的演算を一通り実行できます。

関数

- [29 ページ](#) 「Abs」
- [29 ページ](#) 「Avg」
- [30 ページ](#) 「Ceil」
- [30 ページ](#) 「Count」
- [31 ページ](#) 「Floor」
- [31 ページ](#) 「Max」
- [32 ページ](#) 「Min」
- [33 ページ](#) 「Mod」
- [33 ページ](#) 「Round」
- [34 ページ](#) 「Sum」

Abs

数値または数式の絶対値を返します。値または数式が null の場合は null を返します。

構文

Abs (n1)

パラメーター

パラメーター	説明
n1	値を求める数値または数式です。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ](#)「数値リテラル」を参照してください。

例

次に示す式は Abs 関数の使用例です。

式	戻り値
Abs (1.03)	1.03
Abs (-1.03)	1.03
Abs (0)	0

Avg

セットの数値や数式の値を求め、そのセット内の null でない要素の平均を返します。

構文

Avg (n1 [, n2 ...])

演算関数

パラメーター

パラメーター	説明
n1	セットの最初の数値または数式です。
n2 (オプション)	追加の数値または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Avg 関数の使用例です。

式	戻り値
Avg(0, 32, 16)	16
Avg(2.5, 17, null)	9.75
Avg(Price[0], Price[1], Price[2], Price[3])	Price の null ではない 1 ~ 4 番目までの値の平均値。
Avg(Quantity[*])	Quantity の null ではないすべての値の平均値。

Ceil

渡された数値以上の整数を返します。または、パラメーターが null の場合は null を返します。

構文

Ceil(n)

パラメーター

パラメーター	説明
n	任意の数値または数式です。 n が数値または数式ではない場合は、0 を返します。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Ceil 関数の使用例です。

式	戻り値
Ceil(2.5875)	3
Ceil(-5.9)	-5
Ceil("abc")	0
Ceil(A)	A の値が 99.999 のときは 100 を返します。

Count

数値および／または数式のセットを求め、そのセット内の null でない要素の数を返します。

演算関数

構文

Count(*n1* [, *n2* ...])

パラメーター

パラメーター	説明
n1	任意の数値または数式です。
n2 (オプション)	追加の数値および/または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Count 関数の使用例です。

式	戻り値
Count("Tony", "Blue", 41)	3
Count(Customers[*])	Customers の null ではない値の数。
Count(Coverage[2], "Home", "Auto")	Coverage の 3 番目の値が null ではない場合、3。

Floor

渡された値以下の最大整数値を返します。

構文

Floor(*n*)

パラメーター

パラメーター	説明
n	任意の数値または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Floor 関数の使用例です。

式	戻り値
Floor(21.3409873)	21
Floor(5.999965342)	5
Floor(3.2 * 15)	48

Max

渡された数値セットの null でない要素の最大値を返します。

構文

Max(*n1* [, *n2* ...])

演算関数

パラメーター

パラメーター	説明
n1	任意の数値または数式です。
n2 (オプション)	追加の数値および/または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Max 関数の使用例です。

式	戻り値
Max(234, 15, 107)	234
Max("abc", 15, "Tony Blue")	15
Max("abc")	0
Max(Field1[*], Field2[0])	Field1 の null ではない値と Field2 の最初の値を評価し、高いほうの値を返します。
Max(Min(Field1[*], Field2[0]), Field3, Field4)	最初の数式では、Field1 の null ではない値と Field2 の最初の値を評価し、低いほうの値を返します。この値を Field3 と Field4 の値と比較し、最も大きい値が最終結果となります。 32 ページ「Min」 も参照してください。

Min

渡された数値セットの null でない要素の最小値を返します。

構文

Min(n1 [, n2 ...])

パラメーター

パラメーター	説明
n1	任意の数値または数式です。
n2 (オプション)	追加の数値および/または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

演算関数

例

次に示す式は Min 関数の使用例です。

式	戻り値
Min(234, 15, 107)	15
Min("abc", 15, "Tony Blue")	15
Min("abc")	0
Min(Field1[*], Field2[0])	Sales_July の null ではない値と Sales_August の最初の値を評価し、低いほうの値を返します。
Min(Max(Field1[*], Field2[0]), Field3, Field4)	最初の数式では、Field1 の null ではない値と Field2 の最初の値を評価し、高いほうの値を返します。この値を Field3 と Field4 の値と比較し、最も小さい値が最終結果となります。 31 ページ「Max」 も参照してください。

Mod

数値を別の数値で割り算したときの係数を返します。係数は、被除数を除数で割り算したときの剰余です。剰余の符号は、被除数の符号と常に一致します。

構文

Mod(n1, n2)

パラメーター

パラメーター	説明
n1	被除数である数値または数式です。
n2	除数である数値または数式です。

If n1 や n2 が数値または数式ではない場合は、0 を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#) を参照してください。

例

次に示す式は Mod 関数の使用例です。

式	戻り値
Mod(64, -3)	1
Mod(-13, 3)	-1
Mod("abc", 2)	0
Mod(X[0], Y[9])	X の最初の値を被除数として、Y の 10 番目の値を除数として使用します。
Mod(Round(Value[4], 2), Max(Value[*]))	Value の最初から 5 番目の値を小数点第 3 位で四捨五入して被除数とし、Value の null ではないすべての値の最大値を除数とします。 31 ページ「Max」 および 33 ページ「Round」 も参照してください。

Round

渡された数値または数式の値を求め、渡された小数点の位で四捨五入した値を返します。

演算関数

構文

Round(*n1* [, *n2*])

パラメーター

パラメーター	説明
n1	値を求める数値または数式です。
n2 (オプション)	n1 の値を求める場合の小数点の位の数で、最大 12 です。 n2 の値がない場合、または n2 が無効な場合、関数の小数点の位は 0 と仮定します。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Round 関数の使用例です。

式	戻り値
Round(12.389764537, 4)	12.3898
Round(20/3, 2)	6.67
Round(8.9897, "abc")	9
Round(FV(400, 0.10/12, 30*12), 2)	904195.17. FV 関数を使用して値を求め、小数点第 3 位で四捨五入します。 53 ページ「FV」 も参照してください。
Round(Total_Price, 2)	Total_Price の値を小数点第 3 位で四捨五入します。

Sum

渡された数値セットの null でない要素の合計を返します。

構文

Sum(*n1* [, *n2* ...])

パラメーター

パラメーター	説明
n1	任意の数値または数式です。
n2 (オプション)	追加の数値および/または数式です。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

演算関数

例

次に示す式は Sum 関数の使用例です。

式	戻り値
Sum(2, 4, 6, 8)	20
Sum(-2, 4, -6, 8)	4
Sum(4, 16, "abc", 19)	39
Sum(Amount[2], Amount[5])	Amount の 3 番目と 6 番目の値の合計。
Sum(Round(20/3, 2), Max(Amount[*]), Min(Amount[*]))	20/3 を小数点第 3 位で四捨五入した値と、Amount の null ではない値の最大値と最小値との合計。 31 ページ「Max」、32 ページ「Min」および 33 ページ「Round」も参照してください。

5. 日付と時間関数

この節の関数では、特に日時値を作成したり操作したりします。

関数

- [43 ページ](#) 「Date」
- [44 ページ](#) 「Date2Num」
- [44 ページ](#) 「DateFmt」
- [45 ページ](#) 「IsoDate2Num」
- [46 ページ](#) 「IsoTime2Num」
- [46 ページ](#) 「LocalDateFmt」
- [47 ページ](#) 「LocalTimeFmt」
- [47 ページ](#) 「Num2Date」
- [48 ページ](#) 「Num2GMTime」
- [48 ページ](#) 「Num2Time」
- [49 ページ](#) 「Time」
- [49 ページ](#) 「Time2Num」
- [50 ページ](#) 「TimeFmt」

日付と時間の構造

エポック

日付値と時間値の両方には、時間が始まった瞬間である、原点またはエポックが関連付けられています。このエポックよりも前の日付の値や時間の値はすべて無効です。

すべての日付関数の値は、エポックからの日数で表されます。すべての時間関数の値は、エポックからのミリ秒数で表されます。

Designer ES2 では、すべての日付関数のエポックについては、第 1 日目を 1900 年 1 月 1 日とし、すべての時間関数のエポックについては、第 1 ミリ秒目を夜中の 00:00:00、グリニッジ標準時 (GMT) とすると定義しています。この定義では、GMT より東の時間帯にいるユーザーに負の時間値が返されることがあります。

日付形式

日付形式は、日付の表示方法を簡単に指定したもので、日付が使用する形式を表す各種の句読点や記号で構成されています。次に示す表は日付形式の例です。

日付形式	例
MM/DD/YY	11/11/78
DD/MM/YY	25/07/85
MMMM DD, YYYY	March 10, 1964

日付と時間関数

日付形式は ISO 標準で管理されています。各国または地域によって独自の形式が指定されます。一般的に、日付形式には、ショート形式、標準形式、ロング形式、フル形式の 4 種類があります。次の表に、4 つのカテゴリ別に異なるロケールの異なる時間形式の例を示します。

ロケール識別子と説明	日付形式 (カテゴリ)	例
en_GB 英語 (英国)	DD/MM/YY (ショート形式)	08/12/92 08/04/05
fr_CA フランス語 (カナダ)	YY-MM-DD (標準形式)	92-08-18
de_DE ドイツ語 (ドイツ)	D. MMMM YYYY (ロング形式)	17. Juni 1989
fr_FR フランス語 (フランス)	EEEE, 'le' D MMMM YYYY (フル形式)	Lundi 29 Octobre 1990

時間形式

時間形式は、時間の形式を簡単に指定したもので、句読点、リテラル、パターン記号によって構成されています。次に示す表は時間形式の例です。

時間形式	例
h:MM A	7:15 PM
HH:MM:SS	21:35:26
HH:MM:SS 'o'clock' A Z	14:20:10 o'clock PM EDT

時間形式は ISO 標準で管理されています。時間形式についても、デフォルト形式、ショート形式、標準形式、ロング形式、フル形式を各国が指定しています。ロケールは、その国の標準に合った時間形式を識別する役割を担っています。

次の表に、4 つのカテゴリ別に異なるロケールの異なる時間形式の例を示します。

ロケール識別子と説明	時間形式 (カテゴリ)	例
en_GB 英語 (英国)	HH:MM (ショート形式)	14:13
fr_CA フランス語 (カナダ)	HH:MM:SS (標準形式)	12:15:50
de_DE ドイツ語 (ドイツ)	HH:MM:SS z (ロング形式)	14:13:13 -0400
fr_FR フランス語 (フランス)	HH 'h' MM Z (フル形式)	14 h 13 GMT-04:00

日付と時間関数

日付と時間のパターン形式

日付 / 時間フィールドに日時パターンを作成するには、次の記号を使用する必要があります。一部の日付記号は、中国語、日本語および韓国語ロケールでのみ使用します。それらの記号についても下で説明します。

注意：コンマ (,)、ダッシュ (-)、コロン (:)、スラッシュ (/)、ピリオド (.) およびスペース () はリテラル値として扱われ、パターンのどこでも使用できます。パターンにテキスト文字列を含めるには、文字列を一重引用符 (') で囲みます。例えば、表示パターンとして 'お客様の支払期限: ' MM-DD-YY を指定できます。

日付記号	説明	ロケールに依存する入力値の場合、ロケールが英語（米国）の形式設定された値は 1/1/08（2008 年 1 月 1 日）です
D	月の日付を、1 桁または 2 桁の数値（1～31）で表します。	1
DD	月の日付を 2 桁の数値（01～31）で表します。	01
J	1 年の何日目かを、1 桁、2 桁または 3 桁の数値（1～366）で表します。	1
JJJ	1 年の何日目かを 3 桁の数値（001～366）で表します。	001
M	月を 1 桁または 2 桁の数値（1～12）で表します。	1
MM	月を 2 桁（01～12）の数値で表します。	01
MMM	月名を略称で表示します。	Jan
MMMM	月名を略さずに表示します。	January
E	曜日を 1 桁の数字（1～7）の数値で表します。ここで 1 は日曜日です。	3（2008 年 1 月 1 日は火曜日のため）
EEE	曜日を略称で表示します。	Tue（2008 年 1 月 1 日は火曜日のため）
EEEE	曜日を略さずに表示します。	火曜日（2008 年 1 月 1 日は火曜日のため）
YY	年（西暦）を 2 桁で表します。30 未満の数値は 2000 年より後、30 を超える数値は 2000 年より前を表します。例えば、00=2000、29=2029、30=1930、99=1999 となります。	08
YYYY	年（西暦）を 4 桁で表します。	2008
G	紀元名（BC または AD）を表示します。	AD
w	月の第何週かを 1 桁の数値（0～5）で表します。その月で最も早く、土曜日で終わる連続した 4 日間は第 1 週になります。	1
WW	年の第何週かを表す 2 桁の数値（01～53）で、ISO-8601 に規定されています。1 月 4 日を含む週が第 1 週になります。	01

中国語、日本語および韓国語ロケールの日付パターン指定用に、追加の日付パターンがいくつかあります。

日付と時間関数

日本語の元号は数種類の記号で表されます。次のうち最後の5つは元号記号で、日本の元号を表す代替記号として使用します。

中国語、日本語および韓国語の日付シンボル	説明
DDD	当該ロケールで月の日を表す漢数字
DDDD	当該ロケールで月の日を表す漢数字（「十」表記）
YYY	当該ロケールで年を表す漢数字
YYYYY	当該ロケールで年を表す漢数字（「十」表記）
g	当該ロケールの元号。このパターンでは、現在の日本の元号「平成」に対しては ASCII 文字の「H」（U+48）が表示されます。
gg	当該ロケールの元号。このパターンでは、現在の日本の元号に対しては Unicode の 1 文字（U+5E73）が表示されます。
ggg	当該ロケールの元号。このパターンでは、現在の日本の元号に対しては Unicode の複数文字（U+5E73 U+6210）が表示されます。
ᄀ	当該ロケールの元号。このパターンでは、現在の日本の元号に対しては 全角文字の「H」（U+FF28）が表示されます。
ᄁ ᄂ	当該ロケールの元号。このパターンでは、現在の日本の元号に対しては Unicode の 1 文字（U+337B）が表示されます。

時間シンボル	説明	ロケールに依存する入力値	ロケールが英語（米国）の形式設定された値
h	時間（12 時間制）を 1 桁または 2 桁の数値（1～12）で表します。	12:08 AM または 2:08 PM	12 または 2
hh	時間（12 時間制）を 2 桁の数値（01～12）で表します。	12:08 AM または 2:08 PM	12 または 02
k	時間（12 時間制）を 1 桁または 2 桁の数値（0～11）で表します。	12:08 AM または 2:08 PM	0 または 2
kk	時間（12 時間制）を 2 桁の数値（00～11）で表します。	12:08 AM または 2:08 PM	00 または 02
H	時間（24 時間制）を 1 桁または 2 桁の数値（0～23）で表します。	12:08 AM または 2:08 PM	0 または 14
HH	時間（24 時間制）を 2 桁の数値（00～23）で表します。	12:08 AM または 2:08 PM	00 または 14
K	時間（24 時間制）を 1 桁または 2 桁の数値（1～24）で表します。	12:08 AM または 2:08 PM	24 または 14
KK	時間（24 時間制）を 2 桁の数値（01～24）で表します。	12:08 AM または 2:08 PM	24 または 14
M	分を 1 桁または 2 桁の数値（0～59）で表します。 注意：このシンボルを時間シンボルで使用する必要があります。	2:08 PM	8
MM	分を 2 桁の数値（00～59）で表します。 注意：このシンボルを時間シンボルで使用する必要があります。	2:08 PM	08
S	秒を 1 桁または 2 桁の数値（0～59）で表します。 注意：このシンボルを時間と分のシンボルで使用する必要があります。	2:08:09 PM	9
SS	秒を 2 桁の数値（00～59）で表します。 注意：このシンボルを時間と分のシンボルで使用する必要があります。	2:08:09 PM	09
FFF	1000 分の 1 秒を 3 桁の数値（000～999）で表します。 注意：このシンボルを時間、分、秒のシンボルで使用する必要があります。	2:08:09 PM	09
A	午前零時から正午まで（AM）または正午から午前零時まで（PM）で表します。	2:08:09 PM	PM
z	ISO-8601 タイムゾーン形式（例えば、Z、+0500、-0030、-01、+0100）。 注意：このシンボルを時間シンボルで使用する必要があります。	2:08:09 PM	-0400

日付と時間関数

時間シンボル	説明	ロケールに依存する入力値	ロケールが英語（米国）の形式設定された値
zz	別の ISO-8601 タイムゾーン形式（例えば、Z、+05:00、-00:30、-01、+01:00）。 注意：このシンボルを時間シンボルで使用する必要があります。	2:08:09 PM	-04:00
Z	タイムゾーン名の略称（例：GMT、GMT+05:00、GMT-00:30、EST、PDT） 注意：このシンボルを時間シンボルで使用する必要があります。	2:08:09 PM	EDT

予約済みの記号

以下の記号は特別な意味を持つため、リテラルテキストとしては使用できません。

記号	説明
?	送信する場合、この記号は何らかの 1 文字に対応します。表示のためにマージする場合はスペースになります。
*	送信する場合、この記号は 0 または Unicode の空白文字に対応します。表示のためにマージする場合はスペースになります。
+	送信する場合、この記号は 1 つ以上の Unicode の空白文字に対応します。表示のためにマージする場合はスペースになります。

ロケール

ロケールは、国際標準の整備にあたって特定の国（言語、国または地域）を識別するために使用される標準的な用語です。FormCalc の場合、ロケールは、特定の国や地域で使用される日付、時間、数値および通貨の形式を定義し、エンドユーザーが自分達の慣習に合った形式を使用できるようにします。

各ロケールは、ロケール識別子と呼ばれる固有の文字列で構成されています。これらの文字列の構成は、インターネットエンジニアリングタスクフォース（Internet Engineering Task Force : IETF）として知られる国際標準化機構（International Standards Organization : ISO）によって管理されています。IETF は、インターネット協会（www.isoc.org）の活動団体です。

ロケール識別子は、言語の部分、国または地域の部分、またはその両方から構成されます。次の表に Designer ES2 の本リリースで有効なロケールを示します。

言語	国または地域	ISO コード
アラビア語	アルジェリア	ar_DZ
アラビア語	バーレーン	ar_BH
アラビア語	エジプト	ar_EG
アラビア語	イラク	ar_IQ
アラビア語	ヨルダン	ar_JO
アラビア語	クウェート	ar_KW
アラビア語	レバノン	ar_LB
アラビア語	リビア	ar_LY
アラビア語	モロッコ	ar_MA
アラビア語	オマーン	ar_OM
アラビア語	カタール	ar_QA
アラビア語	サウジアラビア	ar_SA
アラビア語	スーダン	ar_SD
アラビア語	シリア	ar_SY

日付と時間関数

言語	国または地域	ISO コード
アラビア語	チュニジア	ar_TN
アラビア語	アラブ首長国連邦	ar_AE
アラビア語	イエメン	ar_YE
アルメニア語	アルメニア	hy_AM
アゼルバイジャン語 (キリル)	アゼルバイジャン	az_Cyrl_AZ
アゼルバイジャン語 (ラテン)	アゼルバイジャン	az_Latn_AZ
バスク語	スペイン	eu_ES
ボスニア語	ボスニア・ヘルツェゴビナ	bs_BA
ブルガリア語	ブルガリア	bg_BG
カタロニア語	スペイン	ca_ES
中国語	中華人民共和国 (簡体字)	zh_CN
中国語	香港、中国	zh_HK
中国語	台湾 (繁体字)	zh_TW
クロアチア語	クロアチア	hr_HR
チェコ語	チェコ共和国	cs_CZ
デンマーク語	デンマーク	da_DK
オランダ語	ベルギー	nl_BE
オランダ語	オランダ	nl_NL
英語	オーストラリア	en_AU
英語	ベルギー	en_BE
英語	カナダ	en_CA
英語	香港、中国	en_HK
英語	インド	en_IN
英語	アイルランド	en_IE
英語	ニュージーランド	en_NZ
英語	フィリピン	en_PH
英語	シンガポール	en_SG
英語	南アフリカ	en_ZA
英語	英国	en_GB
英語	英国ユーロ	en_GB_EURO
英語	アメリカ合衆国	en_US
英語	米領バージン諸島	en_VI
エストニア語	エストニア	et_EE
フィンランド語	フィンランド	fi_FI
フランス語	ベルギー	fr_BE
フランス語	カナダ	fr_CA

日付と時間関数

言語	国または地域	ISO コード
フランス語	フランス	fr_FR
フランス語	ルクセンブルク	fr_LU
フランス語	スイス	fr_CH
ドイツ語	オーストリア	de_AT
ドイツ語	ドイツ	de_DE
ドイツ語	ルクセンブルク	de_LU
ドイツ語	スイス	de_CH
ギリシャ語	ギリシャ	el_GR
ヘブライ語	イスラエル	he_IL
ハンガリー語	ハンガリー	hu_HU
インドネシア語	インドネシア	id_ID
イタリア語	イタリア	it_IT
イタリア語	スイス	it_CH
日本語	日本	ja_JP
カザフ語	カザフスタン	kk_KZ
クメール語	カンボジア	km_KH
韓国語	韓国	ko_KR
韓国語	韓国 (Hanja)	ko_KR_HANI
ラオ語	ラオス	lo_LA
ラトビア語	ラトビア	lv_LV
リトアニア語	リトアニア	lt_LT
マレー語	マレーシア	ms_MY
ノルウェー語 - ブークモール	ノルウェー	nb_NO
ノルウェー語 - ニーノシク	ノルウェー	nn_NO
ペルシャ語	イラン	fa_IR
ポーランド語	ポーランド	pl_PL
ポルトガル語	ブラジル	pt_BR
ポルトガル語	ポルトガル	pt_PT
ルーマニア語	ルーマニア	ro_RO
ロシア語	ロシア	ru_RU
セルビア語 (キリル)	セルビア・モンテネグロ	sr_Cyrl_CS
セルビア語 (ラテン)	セルビア・モンテネグロ	sr_Latn_CS
スロバキア語	スロバキア	sk_SK
スロベニア語	スロベニア	sl_SI
スペイン語	アルゼンチン	es_AR
スペイン語	ボリビア	es_BO

日付と時間関数

言語	国または地域	ISO コード
スペイン語	チリ	es_CL
スペイン語	コロンビア	es_CO
スペイン語	コスタリカ	es_CR
スペイン語	ドミニカ共和国	es_DO
スペイン語	エクアドル	es_EC
スペイン語	エルサルバドル	es_SV
スペイン語	グアテマラ	es_GT
スペイン語	ホンジュラス	es_HN
スペイン語	メキシコ	es_MX
スペイン語	ニカラグア	es_NI
スペイン語	パナマ	es_PA
スペイン語	パラグアイ	es_PY
スペイン語	ペルー	es_PE
スペイン語	プエルトリコ	es_PR
スペイン語	スペイン	es_ES
スペイン語	アメリカ合衆国	es_US
スペイン語	ウルグアイ	es_UY
スペイン語	ベネズエラ	es_VE
スウェーデン語	スウェーデン	sv_SE
タガログ語	フィリピン	tl_PH
タイ語	タイ	th_TH
タイ語	タイ (トラディショナル)	th_TH_TH
トルコ語	トルコ	tr_TR
ウクライナ語	ウクライナ	uk_UA
ベトナム語	ベトナム	vi_VN

ロケールの要素は通常、両方とも重要です。例えば、カナダ英語とイギリス英語では曜日名と月名の形式は同じですが、日付の形式が異なります。したがって、英語という言語ロケールを指定しただけでは不十分です。逆に、ロケールとして国を指定するだけでも十分ではありません。例えば、カナダでは英語とフランス語で異なる日付形式が使用されています。

一般的に、すべてのアプリケーションはロケールが存在する環境で動作しています。このロケールは、環境ロケールとして知られています。場合によっては、ロケールが存在しないシステムまたは環境でアプリケーションが動作していることもあります。このようなことは稀ですが、この場合環境ロケールはデフォルトのアメリカ英語 (en_US) に設定されます。このロケールは、デフォルトロケールとして知られています。

Date

現在のシステム日付を、エポックからの日数として返します。

構文

Date ()

日付と時間関数**パラメーター**

なし

例

次に示す式は Date 関数の使用例です。

式	戻り値
Date ()	37875 (エポックから 2003 年 9 月 12 日までの日数)

Date2Num

日付文字列を渡すと、エポック以降の日数を返します。

構文Date2Num(*d* [, *f* [, *k*]])**パラメーター**

パラメーター	説明
<i>d</i>	<i>k</i> が指定するロケールにも準拠する、 <i>f</i> が規定する形式の日付文字列。
<i>f</i> (オプション)	日付形式文字列。 <i>f</i> を省略すると、デフォルトの日付形式である MMM D, YYYY が使用されます。
<i>k</i> (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。 <i>k</i> を省略した場合 (または無効な場合) は、環境ロケールが使用されます。

次のいずれかの条件が真の場合、関数は値 0 を返します。

- 指定された日付形式が、関数で指定された形式と一致しない。
- 関数のロケールまたは日付形式が無効である。

情報が不十分でエポック以降の一意の日付を決められない (つまり、日付に関する情報が不足しているか不完全である)。

例

次に示す式は Date2Num 関数の使用例です。

式	戻り値
Date2Num("Mar 15, 1996")	35138
Date2Num("1/1/1900", "D/M/YYYY")	1
Date2Num("03/15/96", "MM/DD/YY")	35138
Date2Num("Aug 1, 1996", "MMM D, YYYY")	35277
Date2Num("96-08-20", "YY-MM-DD", "fr_FR")	35296
Date2Num("1/3/00", "D/M/YY") - Date2Num("1/2/00", "D/M/YY")	29

DateFmt

日付形式スタイルが渡されると、日付形式文字列を返します。

構文DateFmt([*n* [, *k*]])

日付と時間関数

パラメーター

パラメーター	説明
n (オプション)	ロケール固有の時間形式スタイルを次のように識別する整数値。 <ul style="list-style-type: none"> • 1 (ショートスタイル) • 2 (標準スタイル) • 3 (ロングスタイル) • 4 (フルスタイル) n を省略した場合 (または無効な場合) は、デフォルトのスタイル値である 0 が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k を省略した場合 (または無効な場合) は、環境ロケールが使用されます。

例

次に示す式は DateFmt 関数の使用例です。

式	戻り値
DateFmt (1)	M/D/YY (ロケールに en_US を設定している場合)
DateFmt (2, "fr_CA")	YY-MM-DD
DateFmt (3, "de_DE")	D. MMMM YYYY
DateFmt (4, "fr_FR")	EEEE D' MMMM YYYY

IsoDate2Num

有効な日付文字列が渡されると、エポック開始以降の日数を返します。

構文

IsoDate2Num (d)

パラメーター

パラメーター	説明
d	有効な日付文字列

例

次に示す式は IsoDate2Num 関数の使用例です。

式	戻り値
IsoDate2Num ("1900")	1
IsoDate2Num ("1900-01")	1
IsoDate2Num ("1900-01-01")	1
IsoDate2Num ("19960315T20:20:20")	35138
IsoDate2Num ("2000-03-01") - IsoDate2Num ("20000201")	29

日付と時間関数

IsoTime2Num

有効な時間文字列が渡されると、エポック以降のミリ秒数を返します。

構文

```
IsoTime2Num(d)
```

パラメーター

パラメーター	説明
d	有効な時間文字列

例

次に示す式は IsoTime2Num 関数の使用例です。

式	戻り値
IsoTime2Num("00:00:00Z")	東部標準時 (ET) ゾーンにいるユーザーの場合は 1
IsoTime2Num("13")	アメリカのボストンにいるユーザーの場合は 64800001
IsoTime2Num("13:13:13")	カリフォルニアにいるユーザーの場合は 76393001
IsoTime2Num("19111111T131313+01")	東部標準時 (ET) ゾーンにいるユーザーの場合は 43993001

LocalDateFmt

日付形式スタイルが渡されると、ローカライズされた日付形式文字列を返します。

構文

```
LocalDateFmt([n [, k]])
```

パラメーター

パラメーター	説明
n (オプション)	ロケール固有の日付形式スタイルを次のように識別する整数値。 <ul style="list-style-type: none"> 1 (ショートスタイル) 2 (標準スタイル) 3 (ロングスタイル) 4 (フルスタイル) n を省略した場合 (または無効な場合) は、デフォルトのスタイル値である 0 が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k を省略した場合 (または無効な場合) は、環境ロケールが使用されます。

例

次に示す式は LocalDateFmt 関数の使用例です。

式	戻り値
LocalDateFmt(1, "de_DE")	tt.MM.uu
LocalDateFmt(2, "fr_CA")	aa-MM-jj
LocalDateFmt(3, "de_CH")	t. MMMM jjjj
LocalDateFmt(4, "fr_FR")	EEEE j MMMM aaaa

日付と時間関数

LocalTimeFmt

時間形式スタイルに対してローカライズされた時間形式文字列を返します。

構文

```
LocalTimeFmt([n [, k]])
```

パラメーター

パラメーター	説明
n (オプション)	ロケール固有の時間形式スタイルを次のように識別する整数値。 <ul style="list-style-type: none"> 1 (ショートスタイル) 2 (標準スタイル) 3 (ロングスタイル) 4 (フルスタイル) n を省略した場合 (または無効な場合) は、デフォルトのスタイル値である 0 が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k を省略した場合 (または無効な場合) は、環境ロケールが使用されます。

例

次に示す式は LocalTimeFmt 関数の使用例です。

式	戻り値
LocalTimeFmt(1, "de_DE")	HH:mm
LocalTimeFmt(2, "fr_CA")	HH:mm:ss
LocalTimeFmt(3, "de_CH")	HH:mm:ss z
LocalTimeFmt(4, "fr_FR")	HH' h 'mm z

Num2Date

エポック以降の日数を渡すと、日付文字列を返します。

構文

```
Num2Date(n [, f [, k]])
```

パラメーター

パラメーター	説明
n	日数を表す整数値。 n が無効な場合、関数はエラーを返します。
f (オプション)	日付形式文字列。f に対して値を指定しなかった場合、関数ではデフォルトの日付形式である MMM D, YYYY が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k に対して値を指定しなかった場合、または k が無効な場合は、環境ロケールが使用されます。

次のいずれかの条件が真の場合、関数は値 0 を返します。

- 指定された日付形式が、関数で指定された形式と一致しない。
- 関数のロケールまたは日付形式が無効である。

情報が不十分でエポック以降の一意の日付を決められない (つまり、日付に関する情報が不足しているか不完全である)。

日付と時間関数

例

次に示す式は Num2Date 関数の使用例です。

式	戻り値
Num2Date(1, "DD/MM/YYYY")	01/01/1900
Num2Date(35139, "DD-MMM-YYYY", "de_DE")	16-Mrz-1996
Num2Date(Date2Num("Mar 15, 2000") - Date2Num("98-03-15", "YY-MM-DD", "fr_CA"))	Jan 1, 1902

Num2GMTime

エポック以降のミリ秒数に対して GMT 時間文字列を返します。

構文

Num2GMTime(*n* [, *f* [, *k*]])

パラメーター

パラメーター	説明
<i>n</i>	ミリ秒数を表す整数値。 <i>n</i> が無効な場合、関数はエラーを返します。
<i>f</i> (オプション)	時間形式文字列。 <i>f</i> に対して値を指定しなかった場合、関数ではデフォルトの時間形式である H:MM:SS A が使用されます。
<i>k</i> (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。 <i>k</i> に対して値を指定しなかった場合、または <i>k</i> が無効な場合は、環境ロケールが使用されます。

次のいずれかの条件が真の場合、関数は値 0 を返します。

- 指定された時間形式が、関数で指定された形式と一致しない。
- 関数のロケールまたは時間形式が無効である。

情報が不十分でエポック以降の一意の時間を決められない（つまり、時間に関する情報が不足しているか不完全である）。

例

次に示す式は Num2GMTime 関数の使用例です。

式	戻り値
Num2GMTime(1, "HH:MM:SS")	00:00:00
Num2GMTime(65593001, "HH:MM:SS Z")	18:13:13 GMT
Num2GMTime(43993001, TimeFmt(4, "de_DE"), "de_DE")	12.13 Uhr GMT

Num2Time

エポック以降のミリ秒数を渡すと、時間文字列を返します。

構文

Num2Time(*n* [, *f* [, *k*]])

日付と時間関数

パラメーター

パラメーター	説明
n	ミリ秒数を表す整数値。 n が無効な場合、関数はエラーを返します。
f (オプション)	時間形式文字列。f に対して値を指定しなかった場合、関数ではデフォルトの時間形式である H:MM:SS A が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k に対して値を指定しなかった場合、または k が無効な場合は、環境ロケールが使用されます。

次のいずれかの条件が真の場合、関数は値 0 を返します。

- 指定された時間形式が、関数で指定された形式と一致しない。
- 関数のロケールまたは時間形式が無効である。

情報が不十分でエポック以降の一意の時間を決められない（つまり、時間に関する情報が不足しているか不完全である）。

例

次に示す式は Num2Time 関数の使用例です。

式	戻り値
Num2Time(1, "HH:MM:SS")	イギリスのグリニッジでは 00:00:00、東京では 09:00:00
Num2Time(65593001, "HH:MM:SS Z")	アメリカのボストンでは、東部標準時 (EST) 13:13:13
Num2Time(65593001, "HH:MM:SS Z", "de_DE")	アメリカのボストンにいるスイスドイツ語ユーザーの場合は、13:13:13 GMT-05:00
Num2Time(43993001, TimeFmt(4, "de_DE"), "de_DE")	スイスのチューリッヒにいるユーザーの場合は、13.13 Uhr GMT+01:00
Num2Time(43993001, "HH:MM:SSzz")	スイスのチューリッヒにいるユーザーの場合は、13:13+01:00

Time

エポック以降のミリ秒数として現在のシステム時刻を返します。

構文

Time()

パラメーター

なし

例

次に示す式は Time 関数の使用例です。

式	戻り値
Time()	2003年9月15日のちょうど 3:52:15pm には、東部標準時 (EST) ゾーンのユーザーに対して 71533235 が返されます。

Time2Num

時間文字列が渡されると、エポック以降のミリ秒数を返します。

構文

Time2Num(d [, f [, k]])

日付と時間関数

パラメーター

パラメーター	説明
d	k が指定するロケールにも準拠する f が規定する形式の時間文字列。
f (オプション)	時間形式文字列。f に対して値を指定しなかった場合、関数ではデフォルトの時間形式である H:MM:SS A が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k に対して値を指定しなかった場合、または k が無効な場合は、環境ロケールが使用されます。

次のいずれかの条件が真の場合、関数は値 0 を返します。

- 指定された時間形式が、関数で指定された形式と一致しない。
- 関数のロケールまたは時間形式が無効である。

情報が不十分でエポック以降の一意の時間を決められない（つまり、時間に関する情報が不足しているか不完全である）。

例

次に示す式は Time2Num 関数の使用例です。

式	戻り値
Time2Num("00:00:00 GMT", "HH:MM:SS Z")	1
Time2Num("1:13:13 PM")	太平洋標準時のカリフォルニアのユーザーに対しては、76393001。 サマータイム時の同じユーザーに対しては、76033001。
Time2Num("13:13:13", "HH:MM:SS") - Time2Num("13:13:13 GMT", "HH:MM:SS Z") / (60 * 60 * 1000)	標準時では、バンクーバーのユーザーに対しては 8、オタワのユーザーに対しては 5。サマータイム時の戻り値はそれぞれ 7 と 4 になります。
Time2Num("13:13:13 GMT", "HH:MM:SS Z", "fr_FR")	47593001

TimeFmt

時間形式スタイルに対して時間形式を返します。

構文

TimeFmt([n [, k]])

パラメーター

パラメーター	説明
n (オプション)	ロケール固有の時間形式スタイルを次のように識別する整数値。 <ul style="list-style-type: none"> 1 (ショートスタイル) 2 (標準スタイル) 3 (ロングスタイル) 4 (フルスタイル) n に対して値を指定しなかった場合、または n が無効な場合、関数ではデフォルトスタイル値が使用されます。
k (オプション)	ロケールの命名標準に準拠するロケール識別子文字列。k を省略した場合（または無効な場合）は、環境ロケールが使用されます。

日付と時間関数

例

次に示す式は TimeFmt 関数の使用例です。

式	戻り値
TimeFmt (1)	h:MM A (ロケールに en_US が設定されている場合)
TimeFmt (2, "fr_CA")	HH:MM:SS
TimeFmt (3, "fr_FR")	HH:MM:SS Z
TimeFmt (4, "de_DE")	H.MM' Uhr 'Z

6. 会計関数

これらの関数は、利息、元本および評価の会計関係の様々な計算を実行します。

関数

- [52 ページ「Apr」](#)
- [53 ページ「CTerm」](#)
- [53 ページ「FV」](#)
- [54 ページ「IPmt」](#)
- [55 ページ「NPV」](#)
- [55 ページ「Pmt」](#)
- [56 ページ「PPmt」](#)
- [57 ページ「PV」](#)
- [57 ページ「Rate」](#)
- [58 ページ「Term」](#)

Apr

年間のローン利率を百分率で返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

Apr (n1, n2, n3)

パラメーター

パラメーター	説明
n1	ローンの元本を表す数値または数式です。
n2	ローンの支払い金額を表す数値または数式です。
n3	ローン期間中の期間数を表す数値または数式です。

いずれかのパラメーターが null の場合、null を返します。いずれかのパラメーターが負の値または 0 の場合、エラーを返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Apr 関数の使用例です。

式	戻り値
Apr(35000, 269.50, 360)	\$35,000 のローンを月 \$269.50 ずつ 30 年で返済した場合、0.08515404566。
Apr(210000 * 0.75, 850 + 110, 25 * 26)	0.07161332404
Apr(-20000, 250, 120)	エラー
Apr(P_Value, Payment, Time)	この例では、実際の数値または数式の代わりに、変数を使用します。

会計関数

CTerm

将来増額する固定された複利率を補うための投資に必要な期間の値を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

CTerm(*n1*, *n2*, *n3*)

パラメーター

パラメーター	説明
n1	期ごとの利率を表す数値または数式です。
n2	投資の将来価値を表す数値または数式です。
n3	投資開始額を表す数値または数式です。

いずれかのパラメーターが null の場合、null を返します。いずれかのパラメーターが負の値または 0 の場合、エラーを返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は CTerm 関数の使用例です。

式	戻り値
CTerm(0.02, 1000, 100)	116.2767474515
CTerm(0.10, 500000, 12000)	39.13224648502
CTerm(0.0275 + 0.0025, 1000000, 55000 * 0.10)	176.02226044975
CTerm(Int_Rate, Target_Amount, P_Value)	この例では、実際の数値または数式の代わりに、変数を使用します。

FV

一定の利率で一定期間に返済されるときの将来の金額を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

FV(*n1*, *n2*, *n3*)

パラメーター

パラメーター	説明
n1	支払い金額を表す数値または数式です。
n2	投資の期ごとの利息を表す数値または数式です。
n3	支払い期数の合計を表す数値または数式です。

会計関数

次のいずれかの条件を満たす場合、エラーを返します。

- $n1$ または $n3$ のいずれかが負の値または 0 の場合。
- $n2$ が負の値の場合。

いずれかのパラメーターが `null` の場合、`null` を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は `FV` 関数の使用例です。

式	戻り値
<code>FV(400, 0.10 / 12, 30 * 12)</code>	904195.16991842445。これは、月に \$400 の投資が、年 10 % で成長した場合の 30 年後の値です。
<code>FV(1000, 0.075 / 4, 10 * 4)</code>	58791.96145535981。これは、1 か月ごとに \$1,000 の投資をし、四半期ごとに 7.5 % 成長した場合の、10 年後の価値です。
<code>FV(Payment [0], Int_Rate / 4, Time)</code>	この例では、実際の数値または数式の代わりに、変数を使用します。

IPmt

一定期間のローンに対して支払われる利子の額を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

`IPmt(n1, n2, n3, n4, n5)`

パラメーター

パラメーター	説明
$n1$	ローンの元本を表す数値または数式です。
$n2$	投資の年利を表す数値または数式です。
$n3$	月ごとの支払い額を表す数値または数式です。
$n4$	支払いが行われる最初の月を表す数値または数式です。
$n5$	計算するべき月の数を表す数値または数式です。

次のいずれかの条件を満たす場合、エラーを返します。

- $n1$ 、 $n2$ または $n3$ が負の値か 0 の場合。
- $n4$ または $n5$ が負の値の場合。

いずれかのパラメーターが `null` の場合、`null` を返します。支払い金額 ($n3$) が月々の利息負担より少ない場合、0 を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

会計関数

例

次に示す式は IPmt 関数の使用例です。

式	戻り値
IPmt(30000, 0.085, 295.50, 7, 3)	624.8839283142。利率 8.5 % の \$30,000 ローンで、ローン期間中の 7 か月目から 10 か月目の 3 か月間に返済した利息の額です。
IPmt(160000, 0.0475, 980, 24, 12)	7103.80833569485。ローン期間中の 3 年目に返済する利息の返済額です。
IPmt(15000, 0.065, 65.50, 15, 1)	0。これは、この月に発生する利息より月の支払いが少ないからです。

NPV

割引率と各支払いの将来のキャッシュフローに基づいた、投資の正味現在価値を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

NPV(n1, n2 [, ...])

パラメーター

パラメーター	説明
n1	一期の割引率を表す数字または数値です。
n2	期末に必ず発生するキャッシュフローの価値を表す数値または数式です。n2 以降で指定される値が正しい順序になっていることが重要です。

n1 が負の値または 0 の場合はエラーを返します。いずれかのパラメーターが null の場合は null を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は NPV 関数の使用例です。

式	戻り値
NPV(0.065, 5000)	4694.83568075117。これは、年利 6.5 % で、将来 \$5,000 になる投資の正味現在価値です。
NPV(0.10, 500, 1500, 4000, 10000)	11529.60863329007。年利 10 % で、今後 4 年間の各年に \$500、\$1,500、\$4,000 および \$10,000 を生む投資の正味現在価値です。
NPV(0.0275 / 12, 50, 60, 40, 100, 25)	273.14193838457。年利 2.75 % で、今後 5 か月間にそれぞれ \$50、\$60、\$40、\$100 および \$25 を生む投資の正味現在価値です。

Pmt

一定の返済額および一定の利率に基づき、ローン返済額を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

Pmt(n1, n2, n3)

会計関数

パラメーター

パラメーター	説明
n1	ローンの元本を表す数値または数式です。
n2	投資の期ごとの利率を表す数値または数式です。
n3	支払い期数の合計を表す数値または数式です。

いずれかのパラメーターが負の値または 0 の場合はエラーを返します。いずれかのパラメーターが null の場合は null を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Pmt 関数の使用例です。

式	戻り値
Pmt (150000, 0.0475 / 12, 25 * 12)	855.17604207164。これは、\$150,000 のローンを、年利 4.75 % で 25 年返済する場合の各月の返済額です。
Pmt (25000, 0.085, 12)	3403.82145169876。これは、\$25,000 のローンを、年利 8.5 % で 12 年返済する場合の年間の返済額です。

PPmt

一定期間のローンに対して返済された元本の額を返します。

注意：利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

PPmt (n1, n2, n3, n4, n5)

パラメーター

パラメーター	説明
n1	ローンの元本を表す数値または数式です。
n2	年利を表す数値または数式です。
n3	月ごとの支払い額を表す数値または数式です。
n4	支払いが行われる最初の月を表す数値または数式です。
n5	計算するべき月の数を表す数値または数式です。

次のいずれかの条件を満たす場合、エラーを返します。

- n1、n2 または n3 が負の値か 0 の場合。
- n4 または n5 が負の数の場合。

いずれかのパラメーターが null の場合、null を返します。支払い金額 (n3) が月々の利息負担より少ない場合、0 を返します。

注意：FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

会計関数

例

次に示す式は PPmt 関数の使用例です。

式	戻り値
PPmt(30000, 0.085, 295.50, 7, 3)	261.6160716858。これは、利率 8.5 % の \$30,000 ローンで、ローン期間中の 7 か月目から 10 か月目の 3 か月間に返済した元本の額です。
PPmt(160000, 0.0475, 980, 24, 12)	4656.19166430515。ローン期間中の 3 年目に返済された元本の年額です。
PPmt(15000, 0.065, 65.50, 15, 1)	0。これは、この月に発生する利息より月の支払いが少なく、元本部分の返済がないからです。

PV

一定の利率で定期的に定額返済される融資の現価を返します。

注意: 利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

PV(n1, n2, n3)

パラメーター

パラメーター	説明
n1	支払い金額を表す数値または数式です。
n2	投資の期ごとの利息を表す数値または数式です。
n3	支払い期数の合計を表す数値または数式です。

n1 または n3 のいずれかが負の値または 0 の場合はエラーを返します。いずれかのパラメーターが null の場合は null を返します。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は PV 関数の使用例です。

式	戻り値
PV(400, 0.10 / 12, 30 * 12)	45580.32799074439。これは、月に \$400 の投資が、年 10 % で成長した場合の 30 年後の値です。
PV(1000, 0.075 / 4, 10 * 4)	58791.96145535981。これは、1 か月ごとに \$1,000 の投資をし、四半期ごとに 7.5 % 成長した場合の、10 年後の価値です。
PV(Payment[0], Int_Rate / 4, Time)	この例では、実際の数値または数式の代わりに、変数を使用します。

Rate

渡された期間内で、現在から将来にかけて増加する融資額に対して要求される期間における複利を返します。

注意: 利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

Rate(n1, n2, n3)

会計関数

パラメーター

パラメーター	説明
n1	投資の将来価値を表す数値または数式です。
n2	投資の現在価値を表す数値または数式です。
n3	投資期数の合計を表す数値または数式です。

いずれかのパラメーターが負の値または 0 の場合はエラーを返します。いずれかのパラメーターが null の場合は null を返します。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

例

次に示す式は Rate 関数の使用例です。

式	戻り値
Rate(12000, 8000, 5)	0.0844717712 (8.45%)。これは、\$8,000 の現在価値が 5 期の投資で \$12,000 に成長するために必要な期あたりの利率です。
Rate(10000, 0.25 * 5000, 4 * 12)	0.04427378243 (4.43%)。これは、現在価値が 4 年間で \$10,000 に成長するために必要な月あたりの利率です。
Rate(Target_Value, Pres_Value[*], Term * 12)	この例では、実際の数値または数式の代わりに、変数を使用します。

Term

利付きの口座に定期的に定額を払い込んだ場合に、渡された将来価値になるために必要な期数を返します。

注意: 利息の計算方法は、国ごとに異なります。この関数は、米国の利息計算基準に基づいて計算します。

構文

Term(n1, n2, n3)

パラメーター

パラメーター	説明
n1	それぞれの期末の支払額を表す数値または数式です。
n2	投資の期ごとの利率を表す数値または数式です。
n3	投資の将来価値を表す数値または数式です。

いずれかのパラメーターが負の値または 0 の場合はエラーを返します。いずれかのパラメーターが null の場合は null を返します。

注意: FormCalc は、浮動小数点数値を処理するとき、IEEE-754 世界標準に準拠します。詳しくは、[7 ページ「数値リテラル」](#)を参照してください。

会計関数

例

次に示す式は Term 関数の使用例です。

式	戻り値
Term(475, .05, 1500)	3.00477517728 (約 3)。各期 5% の利率で、\$475 の投資が \$1,500 に成長するのに必要な期数です。
Term(2500, 0.0275 + 0.0025, 5000)	1.97128786369。各期 3% の利率で、\$2,500 の投資が \$5,000 になるのに必要な期数です。
Rate(Inv_Value[0], Int_Rate + 0.0050, Target_Value)	この例では、実際の数値または数式の代わりに、変数を使用します。この場合、1 番目の変数 Inv_Value が支払額として使用され、変数 Int_Rate に 0.5% を加えたものが利率として使用され、変数 Target_Value が投資の将来価値として使用されます。

7. 論理関数

ここで説明する関数は、情報のテストや分析を行って、`true` または `false` の結果を得る場合に便利です。

関数

- [60 ページ「Choose」](#)
- [60 ページ「Exists」](#)
- [61 ページ「HasValue」](#)
- [61 ページ「Oneof」](#)
- [62 ページ「Within」](#)

Choose

渡されたパラメーターセットから値を選択します。

構文

```
Choose(n, s1 [, s2 ...])
```

パラメーター

パラメーター	説明
n	セット内で選択する値の位置です。この値が整数でない場合は、n の端数を切り捨てて最も近い自然数にします。 次のいずれかの条件を満たす場合は、空文字列を返します。 <ul style="list-style-type: none"> • n が 1 より小さい。 • n がセット内のアイテム数よりも大きい。 n が null の場合、関数は null を返します。
s1	値セット内の 1 番目の値です。
s2 (オプション)	セット内の追加の値です。

例

次に示す式は Choose 関数の使用例です。

式	戻り値
Choose(3, "Taxes", "Price", "Person", "Teller")	Person
Choose(2, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)	9
Choose(Item_Num[0], Items[*])	Item_Num の 1 番目の値で定義される位置に対応する、Items セット内の値を返します。
Choose(20/3, "A", "B", "C", "D", "E", "F", "G", "H")	F

Exists

渡されたパラメーターが既存オブジェクトへの参照構文かどうかを判定します。

構文

```
Exists(v)
```

論理関数

パラメーター

パラメーター	説明
v	有効な参照構文式 v が参照構文でない場合は、false (0) を返します。

例

次に示す式は Exists 関数の使用例です。

式	戻り値
Exists(Item)	Item オブジェクトが存在する場合は true (1)。その他の場合は false (0)。
Exists("hello world")	false (0)。文字列が参照構文ではありません。
Exists(Invoice.Border.Edge[1].Color)	Invoice オブジェクトが存在し、そのオブジェクトに Border プロパティが存在し、その Border プロパティに 1 つ以上の Edge プロパティが存在し、その Edge プロパティに Color プロパティが存在する場合は true (1)、それ以外の場合は、false (0) を返します。

HasValue

渡されたパラメーターが null、空、空文字列でない値を持つ参照構文かどうかを判定します。

構文

HasValue(v)

パラメーター

パラメーター	説明
v	有効な参照構文式 v が参照構文でない場合は、false (0) を返します。

例

次に示す式は HasValue 関数の使用例です。

式	戻り値
HasValue(2)	true (1)
HasValue(" ")	false (0)
HasValue(Amount[*])	エラー
HasValue(Amount[0])	Amount の 1 番目の値を求め、null、空、空白のいずれの値でもない場合は true (1) を返します。

Oneof

渡された値がセット内にあるかどうか判断します。

構文

Oneof(s1, s2 [, s3 ...])

論理関数

パラメーター

パラメーター	説明
s1	セット内で選択する値の位置です。この値が整数でない場合は、s1 の端数を切り捨てて最も近い自然数にします。
s2	値セット内の 1 番目の値です。
s3 (オプション)	セット内の追加の値です。

例

次に示す式は Oneof 関数の使用例です。

式	戻り値
Oneof(3, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)	true (1)
Oneof("John", "Bill", "Gary", "Joan", "John", "Lisa")	true (1)
Oneof(3, 1, 25)	false (0)
Oneof("loan", Fields[*])	Fields の任意の値が loan という値を持つかどうかを検証します。

Within

渡された値が渡された範囲内にあるかどうか判断します。

構文

Within(s1, s2, s3)

パラメーター

パラメーター	説明
s1	テストする値。 s1 が数値の場合、順序付け比較は数値です。 s1 が数値以外の場合、順序付け比較は現在のロケールの照合順序を使用します。詳しくは、 40 ページ「ロケール」 を参照してください。 s1 が null の場合、関数は null を返します。
s2	テスト範囲の下界です。
s3	テスト範囲の上界です。

例

次に示す式は Within 関数の使用例です。

式	戻り値
Within("C", "A", "D")	true (1)
Within(1.5, 0, 2)	true (1)
Within(-1, 0, 2)	false (0)
Within(\$, 1, 10)	現在の値が 1 から 10 までにある場合は、true (1)

8. その他の関数

この節では、これ以外のどの関数カテゴリにも当てはまらない関数について説明します。これらの関数は様々な用途に役立ちます。

関数

- [63 ページ「Eval」](#)
- [63 ページ「Null」](#)
- [64 ページ「Ref」](#)
- [64 ページ「UnitType」](#)
- [65 ページ「UnitValue」](#)

Eval

渡されたフォームの演算後の値を返します。

構文

Eval(*s*)

パラメーター

パラメーター	説明
<i>s</i>	<p>数式または数式のリストを表す有効な文字列です。</p> <p>Eval 関数は、ユーザー定義の変数と関数を参照できません。次に例を示します。</p> <pre>var s = "var t = concat(s, "hello")" eval(s)</pre> <p>この場合、Eval 関数は <i>s</i> を認識しないため、エラーを返します。変数 <i>s</i> を参照する後続の関数も失敗します。</p>

例

次に示す式は Eval 関数の使用例です。

式	戻り値
eval("10*3+5*4")	50
eval("hello")	error

Null

null 値を返します。null 値は値がないことを意味します。

定義

Null()

パラメーター

なし

その他の関数

例

次に示す式は Null 関数の使用例です。

式	戻り値
Null()	null
Null() + 5	5
Quantity = Null()	null を Quantity オブジェクトに割り当てます。
Concat("ABC", Null(), "DEF")	ABCDEF 68 ページ「Concat」 も参照してください。

Ref

既存のオブジェクトの参照を返します。

定義

Ref(*v*)

パラメーター

パラメーター	説明
<i>v</i>	参照構文、プロパティ、メソッド、関数のいずれかを表す有効な文字列です。 渡されたパラメーターが null の場合は、null 参照を返します。その他すべての渡されたパラメーターの場合、エラー例外を生成します。

例

次に示す式は Ref 関数の使用例です。

数式	戻り値
Ref("10*3+5*4")	10*3+5*4
Ref("hello")	hello

UnitType

ユニットスパンの単位を返します。ユニットスパンは、数値とそれに続く単位名を構成する文字列です。

構文

UnitType(*s*)

その他の関数

パラメーター

パラメーター	説明
s	<p>数値と有効な単位名（ユニットスパン）を含む有効な文字列です。次の測定単位が認められています。</p> <ul style="list-style-type: none"> • in、inches • mm、millimeters • cm、centimeters • pt、points • pc、picas • mp、millipoints <p>s が無効の場合は、in を返します。</p>

例

次に示す式は UnitType 関数の使用例です。

式	結果
UnitType("36 in")	in
UnitType("2.54centimeters")	cm
UnitType("picas")	pc
UnitType("2.cm")	cm
UnitType("2.zero cm")	in
UnitType("kilometers")	in
UnitType(Size[0])	Size の最初の値の測定値を返します。

UnitValue

オプションの単位に変換した後、測定の数値とそれに関連するユニットスパンを返します。ユニットスパンは、数値とそれに続く有効な測定単位を構成する文字列です。

構文

UnitValue(s1 [, s2])

パラメーター

パラメーター	説明
s1	<p>数値と有効な単位名（ユニットスパン）を含む有効な文字列です。次の測定単位が認められています。</p> <ul style="list-style-type: none"> • in、inches • mm、millimeters • cm、centimeters • pt、picas、points • mp、millipoints
s2 (オプション)	<p>有効な測定単位を含む文字列です。s1 で指定したユニットスパンをこの新しい測定単位に変換します。</p> <p>s2 の値を指定しない場合は、s1 で指定した測定単位を使用します。s2 が無効な場合は、s1 をインチに変換します。</p>

その他の関数

例

次に示す式は UnitValue 関数の使用例です。

式	戻り値
UnitValue("2in")	2
UnitValue("2in", "cm")	5.08
UnitValue("6", "pt")	432
UnitValue("A", "cm")	0
UnitValue(Size[2], "mp")	Size の 3 番目の値をミリポイントに変換した測定値を返します。
UnitValue("5.08cm", "kilograms")	2

9. 文字列関数

この節の関数は、文字列の値の操作、評価および作成を行います。

関数

- [67 ページ](#) 「At」
- [68 ページ](#) 「Concat」
- [68 ページ](#) 「Decode」
- [69 ページ](#) 「Encode」
- [69 ページ](#) 「Format」
- [70 ページ](#) 「Left」
- [71 ページ](#) 「Len」
- [71 ページ](#) 「Lower」
- [72 ページ](#) 「Ltrim」
- [72 ページ](#) 「Parse」
- [73 ページ](#) 「Replace」
- [73 ページ](#) 「Right」
- [74 ページ](#) 「Rtrim」
- [74 ページ](#) 「Space」
- [75 ページ](#) 「Str」
- [75 ページ](#) 「Stuff」
- [76 ページ](#) 「Substr」
- [77 ページ](#) 「Uuid」
- [77 ページ](#) 「Upper」
- [78 ページ](#) 「WordNum」

At

別の文字列中に存在する、任意の文字列の開始文字位置を特定します。

構文

At (*s1*, *s2*)

パラメーター

パラメーター	説明
s1	ソース文字列です。
s2	検索文字列です。 s2 が s1 の一部ではない場合、0 を返します。 s2 が空の場合、1 を返します。

文字列関数

例

次に示す式は At 関数の使用例です。

式	戻り値
At("ABCDEFGH", "AB")	1
At("ABCDEFGH", "F")	6
At(23412931298471, 29)	5 (ソース文字列内での 29 の最初の候補の位置)
At(Ltrim(Cust_Info[0]), "555")	Cust_Info の 1 番目の値での文字列 555 の位置。 72 ページ「Ltrim」 も参照してください。

Concat

複数の文字列を連結して返します。

構文

Concat(*s1* [, *s2* ...])

パラメーター

パラメーター	説明
<i>s1</i>	集合内の最初の文字列
<i>s2</i> (オプション)	集合に加えられる追加文字列

例

次に示す式は Concat 関数の使用例です。

式	戻り値
Concat("ABC", "DEF")	ABCDEF
Concat("Tony", Space(1), "Blue")	Tony Blue 74 ページ「Space」 も参照してください。
Concat("You owe ", WordNum(1154.67, 2), ".")	You owe One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents. 78 ページ「WordNum」 も参照してください。

Decode

渡された文字列をデコードして返します。

構文

Decode(*s1* [, *s2*])

文字列関数

パラメーター

パラメーター	説明
s1	デコードする文字列です。
s2 (オプション)	<p>実行するデコードの種類を識別する文字列です。有効なデコード文字列は次のとおりです。</p> <ul style="list-style-type: none"> • url (URL デコード) • html (HTML デコード) • xml (XML デコード) <p>s2 に値を指定しないと、URL デコードが使用されます。</p>

例

次に示す式は Decode 関数の使用例です。

式	戻り値
Decode("ÆÁ&Aacirc;Á&Aacirc;", "html")	€ÁÂÃ
Decode("~!@#%&^&*()_+ `{ } [] <>?, . / ; ' : <>; , . / ; ' ; : ", "xml")	~!@#%&^&*()_+ `{ } [] <>?, . / ; ' :

Encode

渡された文字列をエンコードして返します。

構文

Encode(s1 [, s2])

パラメーター

パラメーター	説明
s1	エンコードする文字列です。
s2 (オプション)	<p>実行するエンコードの種類を識別する文字列です。有効なエンコード文字列は次のとおりです。</p> <ul style="list-style-type: none"> • url (URL エンコード) • html (HTML エンコード) • xml (XML エンコード) <p>s2 に値を指定しないと、URL エンコードが使用されます。</p>

例

次に示す式は Encode 関数の使用例です。

式	戻り値
Encode(" "hello, world! " ", "url")	%22hello,%20world!%22
Encode("ÁÂÃÄÅ", "html")	ÁÂÃÄÅÆ

Format

指定したピクチャ形式文字列に従って、渡されたデータをフォーマットします。

文字列関数

構文

Format (s1, s2 [, s3 ...])

パラメーター

パラメーター	説明
s1	パターン形式文字列です。ロケールに依存する日付形式または時間形式場合があります。40 ページ「ロケール」を参照してください。
s2	<p>フォーマットするソースデータです。</p> <p>日付パターン形式の場合、ISO 日時文字列または次のいずれかの形式の ISO 日付文字列のソースデータを使用してください。</p> <ul style="list-style-type: none"> YYYY[MM[DD]] YYYY[-MM[-DD]] <p>時間パターン形式の場合、ISO 日時文字列または次のいずれかの形式の ISO 時間文字列のソースデータを使用してください。</p> <ul style="list-style-type: none"> HH[MM[SS[.FFF][z]]] HH[MM[SS[.FFF][+HH[MM]]]] HH[MM[SS[.FFF][-HH[MM]]]] HH[:MM[:SS[:FFF][z]]] HH[:MM[:SS[:FFF][-HH[:MM]]]] HH[:MM[:SS[:FFF][+HH[:MM]]]] <p>日付パターン形式の場合、ISO 日時文字列のソースデータを使用してください。</p> <p>数値パターン形式の場合、数値のソースデータを使用してください。</p> <p>テキストパターン形式の場合、テキストのソースデータを使用してください。</p> <p>複合パターン形式の場合、ソースデータ引数の数とパターンのサブ要素の数を一致させてください。</p>
s3 (オプション)	フォーマットする追加ソースデータです。

例

次に示す式は Format 関数の使用例です。

式	戻り値
Format ("MMM D, YYYY", "20020901")	Sep 1, 2002
Format ("\$\$9,999,999.99", 1234567.89)	アメリカでは \$1,234,567.89、フランスでは 1 234 567,89 ユーロ。

Left

左側の最初の文字から指定された文字数を抽出します。

構文

Left (s, n)

パラメーター

パラメーター	説明
s	抽出する文字列です。
n	<p>抽出する文字の数です。</p> <p>抽出する文字の数が文字列より長い場合は、文字列全体を返します。</p> <p>抽出する文字の数が 0 以下の場合は、空文字列を返します。</p>

文字列関数

例

次に示す式は Left 関数の使用例です。

式	戻り値
Left("ABCDEFGH", 3)	ABC
Left("Tony Blue", 5)	"Tony "
Left(Telephone[0], 3)	Telephone の 1 番目の値の最初の 3 文字。
Left(Rtrim>Last_Name), 3)	Last_Name の最初の 3 文字。 74 ページ「Rtrim」 も参照してください。

Len

渡された文字列の文字数を返します。

構文

Len(*s*)

パラメーター

パラメーター	説明
<i>s</i>	調べる文字列です。

例

次に示す式は Len 関数の使用例です。

式	戻り値
Len("ABDCEFGH")	8
Len(4)	1
Len(Str(4.532, 6, 4))	6 75 ページ「Str」 も参照してください。
Len(Amount[*])	Amount の 1 番目の値の文字数。

Lower

指定された文字列内の大文字をすべて小文字に変換します。

構文

Lower(*s*, [, *k*])

パラメーター

パラメーター	説明
<i>s</i>	変換する文字列です。
<i>k</i> (オプション)	有効なロケールを表す文字列です。 <i>k</i> に値を指定しないと、環境ロケールが使用されます。 40 ページ「ロケール」 も参照してください。 この関数で変換されるのは、ユニコード文字の U+41 ~ U+5A (ASCII 文字) および U+FF21 ~ U+FF3A (全角文字) のみです。

文字列関数

例

次に示す式は Lower 関数の使用例です。

式	戻り値
Lower("ABC")	abc
Lower("21 Main St.")	21 main st.
Lower(15)	15
Lower(Address[0])	Address の 1 番目の値の文字をすべて小文字に変換します。

Ltrim

文字の前にある空白文字をすべて削除した文字列を返します。

空白文字には、ASCII スペース、水平タブ、改行、垂直タブ、用紙送り、復帰、Unicode 空白文字 (Unicode カテゴリが Z のもの) が含まれます。

構文

Ltrim(*s*)

パラメーター

パラメーター	説明
<i>s</i>	トリミングする文字列です。

例

次に示す式は Ltrim 関数の使用例です。

式	戻り値
Ltrim(" ABCD")	"ABCD"
Ltrim(Rtrim(" Tony Blue "))	"Tony Blue" 74 ページ「Rtrim」も参照してください。
Ltrim(Address[0])	Address の 1 番目の値から、文字の前にある空白文字をすべて削除します。

Parse

指定されたピクチャ形式に従って、渡されたデータを分析します。

データは次のいずれかの値に解析されます。

- 日付パターン形式：YYYY-MM-DD という形式の ISO 日付文字列
- 時間パターン形式：HH:MM:SS という形式の ISO 時間文字列
- 日時パターン形式：YYYY-MM-DDTHH:MM:SS という形式の ISO 日時文字列
- 数字パターン形式：数字
- テキストパターン：テキスト

構文

Parse(*s1*, *s2*)

文字列関数

パラメーター

パラメーター	説明
s1	有効な日付または時間パターン形式文字列です。 日付と時間形式について詳しくは、 36 ページ「日付と時間の構造」 を参照してください。
s2	解析する文字列データです。

例

次に示す式は Parse 関数の使用例です。

式	戻り値
Parse("MMM D, YYYY", "Sep 1, 2002")	2002-09-01
Parse("\$9,999,999.99", "\$1,234,567.89")	米国では 1234567.89。

Replace

指定された文字列内で、任意の文字列を別の文字列ですべて置換します。

構文

Replace(s1, s2 [, s3])

パラメーター

パラメーター	説明
s1	ソース文字列です。
s2	置換する文字列です。
s3 (オプション)	置換後の文字列。 s3 に値を指定しないと、または s3 が null の場合、空文字列が使用されます。

例

次に示す式は Replace 関数の使用例です。

式	戻り値
Replace("Tony Blue", "Tony", "Chris")	Chris Blue
Replace("ABCDEFGH", "D")	ABCEFGH
Replace("ABCDEFGH", "D")	ABCDEFGH
Replace(Comments[0], "recieve", "receive")	Comments の 1 番目の値にある単語を正しいスペル receive に置換します。

Right

右側の最後の文字から渡された文字数を抽出します。

構文

Right(s, n)

文字列関数

パラメーター

パラメーター	説明
s	抽出する文字列です。
n	抽出する文字の数です。 n が文字列より長い場合は、文字列全体を返します。 n が 0 以下の場合は、空文字列を返します。

例

次に示す式は Right 関数の使用例です。

式	戻り値
Right("ABCDEFGH", 3)	FGH
Right("Tony Blue", 5)	" Blue"
Right(Telephone[0], 7)	Telephone の 1 番目の値の最後の 7 文字。
Right(Rtrim(CreditCard_Num), 4)	CreditCard_Num の最後の 4 文字。 74 ページ「Rtrim」 も参照してください。

Rtrim

文字の後にある空白文字をすべて削除した文字列を返します。

空白文字には、ASCII スペース、水平タブ、改行、垂直タブ、用紙送り、復帰、Unicode 空白文字 (Unicode カテゴリが Z のもの) が含まれます。

構文

Rtrim(s)

パラメーター

パラメーター	説明
s	トリミングする文字列です。

例

次に示す式は Rtrim 関数の使用例です。

式	戻り値
Rtrim("ABCD ")	"ABCD"
Rtrim("Tony Blue ")	"Tony Blue"
Rtrim(Address[0])	Address の 1 番目の値から、文字の後にある空白文字をすべて削除します。

Space

渡された数の空スペースで構成される文字列を返します。

構文

Space(n)

文字列関数

パラメーター

パラメーター	説明
n	空スペースの数です。

例

次に示す式は Space 関数の使用例です。

式	戻り値
Space(5)	" "
Space(Max(Amount[*]))	Amount の最大値と同じ文字数の空白文字列。 31 ページ「Max」も参照してください。
Concat("Tony", Space(1), "Blue")	Tony Blue

Str

数字を文字列へ変換します。FormCalc は変換結果を指定された幅にフォーマットし、指定された小数位に数を四捨五入します。

構文

Str(n1 [, n2 [, n3]])

パラメーター

パラメーター	説明
n1	変換する数です。
n2 (オプション)	文字列の最大幅です。n2 に値を指定しなかった場合、デフォルト幅の 10 が使用されます。 変換結果の文字列が n2 より長い場合は、n2 で指定した幅のアスタリスク (*) 文字列を返します。
n3 (オプション)	小数点以下の桁数です。n3 に値を指定しなかった場合、デフォルト精度の 0 が使用されます。

例

次に示す式は Str 関数の使用例です。

式	戻り値
Str(2.456)	" 2"
Str(4.532, 6, 4)	4.5320
Str(234.458, 4)	" 234"
Str(31.2345, 4, 2)	****
Str(Max(Amount[*]), 6, 2)	Amount の最大値を小数点以下 2 位までの 6 文字の文字列に変換します。 31 ページ「Max」も参照してください。

Stuff

文字列を別の文字列へ挿入します。

構文

Stuff(s1, n1, n2 [, s2])

文字列関数

パラメーター

パラメーター	説明
s1	ソース文字列です。
n1	新しい文字列 s2 が挿入される s1 の位置です。 n1 が 1 未満の場合は、最初の文字位置を指すとみなします。n1 が s1 の長さを超える場合は、最後の文字位置を指すとみなします。
n2	文字列 s1 の文字位置 n1 から削除する文字の数です。 n2 が 0 以下の場合は、文字数を 0 とみなします。
s2 (オプション)	s1 に挿入する文字列です。 s2 に値を指定しないと、空文字列が使用されます。

例

次に示す式は Stuff 関数の使用例です。

式	戻り値
Stuff("TonyBlue", 5, 0, " ")	Tony Blue
Stuff("ABCDEFGH", 4, 2)	ABCFGH
Stuff(Address[0], Len(Address[0]), 0, "Street")	単語 Street の 1 番目の値を Address の最後に追加します。 71 ページ「Len」も参照してください。
Stuff("members-list@myweb.com", 0, 0, "cc:")	cc:members-list@myweb.com

Substr

渡された文字列の一部を抽出します。

構文

Substr(s1, n1, n2)

パラメーター

パラメーター	説明
s1	ソース文字列です。
n1	抽出を開始する文字列 s1 内の位置です。 n1 が 1 未満の場合は、最初の文字位置を指すとみなします。n1 が s1 の長さを超える場合は、最後の文字位置を指すとみなします。
n2	抽出する文字の数です。 n2 が 0 以下の場合、FormCalc は空文字列を返します。n1 + n2 が s1 の長さを超える場合は、n1 から s1 の最後までまでのサブ文字列を返します。

例

次に示す式は Substr 関数の使用例です。

式	戻り値
Substr("ABCDEFG", 3, 4)	CDEF
Substr(3214, 2, 1)	2
Substr>Last_Name[0], 1, 3)	Last_Name の 1 番目の値から、最初の 3 文字を返します。

文字列関数

式	戻り値
Substr("ABCDEFGH", 5, 0)	" "
Substr("21 Waterloo St.", 4, 5)	Water

Uuid

識別方法として使用される UUID (Universally Unique Identifier) 文字列を返します。

構文

Uuid([n])

パラメーター

パラメーター	説明
n	<p>UUID 文字列の形式を識別する数です。有効な数は次のとおりです。</p> <ul style="list-style-type: none"> 0 (デフォルト値) : UUID 文字列には、16 進数のオクテットのみが含まれます。 1 : UUID 文字列には、決められた位置で 16 進数オクテットのシーケンスを区切るダッシュ文字が含まれます。 <p>n に値を指定しないと、デフォルト値が使用されます。</p>

例

次に示す式は Uuid 関数の使用例です。

式	戻り値
Uuid()	3c3400001037be8996c400a0c9c86dd5 などの値
Uuid(0)	3c3400001037be8996c400a0c9c86dd5 などの値
Uuid(1)	1a3ac000-3dde-f352-96c4-00a0c9c86dd5 などの値
Uuid(7)	1a3ac000-3dde-f352-96c4-00a0c9c86dd5 などの値

Upper

文字列内の小文字をすべて大文字に変換します。

構文

Upper(s [, k])

パラメーター

パラメーター	説明
s	変換する文字列です。
k (オプション)	<p>有効なローケルを表す文字列です。k に値を指定しないと、環境ローケルが使用されます。</p> <p>40 ページ「ローケル」 も参照してください。</p> <p>この関数で変換されるのは、ユニコード文字の U+61 ~ U+7A (ASCII 文字) および U+FF41 ~ U+FF5A (全角文字) のみです。</p>

文字列関数

例

次に示す式は Upper 関数の使用例です。

式	戻り値
Upper("abc")	ABC
Upper("21 Main St.")	21 MAIN ST.
Upper(15)	15
Upper(Address[0])	Address の 1 番目の値の文字をすべて大文字に変換します。

WordNum

渡された数を表す英語テキストを返します。

構文

WordNum(n1 [, n2 [, k]])

パラメーター

パラメーター	説明
n1	変換する数です。 以下の文のいずれかが true の場合、エラーを示す「*」(アスタリスク)を返します。 <ul style="list-style-type: none"> n1 が数字ではない。 n1 の整数値が負の値である。 n1 の整数値が 922,337,203,685,477,550 より大きい。
n2 (オプション)	形式オプションを識別する数です。有効な数は次のとおりです。 <ul style="list-style-type: none"> 0 (デフォルト値)：数字は単純な数を表すテキストに変換されます。 1：数字は小数点のない金額を表すテキストに変換されます。 2：数字は小数点のある金額を表すテキストに変換されます。 n2 に値を指定しないと、デフォルト値 (0) が使用されます。
k (オプション)	有効なロケールを表す文字列です。k に値を指定しないと、環境ロケールが使用されます。 40 ページ「ロケール」 も参照してください。 本リリースの時点では、英語以外のロケール識別子をこの関数で指定することはできません。

例

次に示す式は WordNum 関数の使用例です。

式	戻り値
WordNum(123.45)	One Hundred and Twenty-three Dollars
WordNum(123.45, 1)	One Hundred and Twenty-three Dollars
WordNum(1154.67, 2)	One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents
WordNum(43, 2)	Forty-three Dollars And Zero Cents
WordNum(Amount[0], 2)	Amount の 1 番目の値を変換します。

10. URL 関数

ここで説明する関数では、アクセス可能な URL との間で、コンテンツタイプやエンコーディングデータなどの情報をやり取りします。

関数

- [79 ページ「Get」](#)
- [79 ページ「Post」](#)
- [80 ページ「Put」](#)

Get

渡された URL のコンテンツをダウンロードします。

注意： Adobe Acrobat® と Adobe Reader® では、*initialize* イベントが開始するまでは、フォームが承認済みかどうかを確認できません。フォームのレンダリングの前に承認済みのフォームで *Get* 関数を使用する場合は、*docReady* イベントを使用します。

構文

Get (*s*)

パラメーター

パラメーター	説明
<i>s</i>	ダウンロードする URL です。 URL からダウンロードできない場合は、エラーを返します。

例

次に示す式は *Get* 関数の使用例です。

式	戻り値
Get ("http://www.myweb.com/data/mydata.xml")	指定のファイルから XML データを取得します。
Get ("ftp://ftp.gnu.org/gnu/GPL")	GNU Public License のコンテンツです。
Get ("http://intranet?sql=SELECT**+FROM+projects+FOR+XML+AUTO,+ELEMENTS")	指定の Web サイトへの SQL クエリの結果です。

Post

渡されたデータを指定した URL にポストします。

注意： Acrobat と Adobe Reader では、*initialize* イベントが開始するまでは、フォームが承認済みかどうかを確認できません。フォームのレンダリングの前に承認済みのフォームで *Post* 関数を使用する場合は、*docReady* イベントを使用します。

構文

Post (*s1*, *s2* [, *s3* [, *s4* [, *s5*]]])

URL 関数

パラメーター

パラメーター	説明
s1	ポストする URL です。
s2	ポストするデータです。 データをポストできない場合は、エラーを返します。
s3 (オプション)	<p>ポストするデータのコンテンツタイプを含む文字列です。有効なコンテンツタイプは、以下のとおりです。</p> <ul style="list-style-type: none"> • application/octet-stream (デフォルト値) • text/html • text/xml • text/plain • multipart/form-data • application/x-www-form-urlencoded • その他の有効な MIME タイプ <p>s3 の値を指定しないと、コンテンツタイプはデフォルト値に設定されます。アプリケーションでは、指定したコンテンツタイプに沿った正しい形式のデータがポストされます。</p>
s4 (オプション)	<p>データをエンコードするために使用するコードページの名前を含む文字列です。有効なコードページ名は、以下のとおりです。</p> <ul style="list-style-type: none"> • UTF-8 (デフォルト値) • UTF-16 • ISO-8859-1 • Internet Assigned Numbers Authority (IANA) がリスト表示している任意の文字エンコーディング <p>s4 の値を指定しないと、コードページはデフォルト値に設定されます。アプリケーションでは、指定したコードページに一致するエンコーディングのデータがポストされます。</p>
s5 (オプション)	<p>データのポストに付属する HTTP ヘッダーを含む文字列です。</p> <p>s5 の値を指定しないと、ポスト時に HTTP ヘッダーが含まれません。</p> <p>SOAP サーバーの場合は通常、ポスト時に SOAPAction ヘッダーが必要です。</p>

例

次に示す式は Post 関数の使用例です。

式	戻り値
<pre>Post("http://tools_build/scripts/jfecho.cgi", "user=joe&passwd=xxxxx&date=27/08/2002", "application/x-www-form-urlencoded")</pre>	URL エンコードされたいくつかのログインデータをサーバーにポストし、サーバーが認識したページを返します。
<pre>Post("http://www.nanonull.com/TimeService/ TimeService.asmx/getLocalTime", "<?xml version='1.0' encoding='UTF-8'?'><soap:Envelope><soap:Body> <getLocalTime/></soap:Body> </soap:Envelope>", "text/xml", "utf-8", "http://www.Nanonull.com/TimeService/getLocalTime")</pre>	ローカル時間の SOAP 要求をいくつかのサーバーにポストし、XML 応答が戻ってくることを期待します。

Put

渡されたデータを指定した URL にアップロードします。

注意: Acrobat と Adobe Reader では、*initialize* イベントが開始するまでは、フォームが承認済みかどうかを確認できません。フォームのレンダリングの前に承認済みのフォームで *Put* 関数を使用する場合は、*docReady* イベントを使用します。

URL 関数

構文

```
Put (s1, s2 [, s3 ])
```

パラメーター

パラメーター	説明
s1	アップロードする URL です。
s2	アップロードするデータです。 データをアップロードできない場合は、エラーを返します。
s3 (オプション)	データをエンコードするために使用するコードページの名前を含む文字列です。有効なコードページ名は、以下のとおりです。 <ul style="list-style-type: none">• UTF-8 (デフォルト値)• UTF-16• ISO8859-1• Internet Assigned Numbers Authority (IANA) がリスト表示している任意の文字エンコーディング s3 の値を指定しないと、コードページはデフォルト値に設定されます。アプリケーションでは、指定したコードページに一致するエンコーディングのデータがアップロードされます。

例

次に示す式は Put 関数の使用例です。

式	戻り値
<pre>Put ("ftp://www.example.com/pub/fubu.xml", "<?xml version='1.0' encoding='UTF-8'?><msg>hello world!</msg>")</pre>	XML データを pub/fubu.xml ファイルにアップロードすることを FTP サーバーがユーザーに許可している場合は、何も返しません。それ以外の場合は、エラーを返します。

注意: この例はサーバー環境でのみ使用でき、Acrobat または Adobe Reader では使用できません。Acrobat および Adobe Reader で表示されるフォームでは、HTTP、HTTPS および FILE プロトコルを使用します。

索引

- A**
- Abs (演算関数) 29
 - Apr (会計関数) 52
 - At (文字列関数) 67
 - Avg (演算関数) 29
- B**
- Boolean 値の処理 13
 - break 式、FormCalc 19
- C**
- Ceil (演算関数) 30
 - Choose (論理関数) 60
 - Concat (文字列関数) 68
 - continue 式、FormCalc 19
 - Count (演算関数) 30
 - CTerm (会計関数) 53
- D**
- Date 関数 43
 - Date2Num 関数 44
 - DateFmt 関数 44
 - Decode (文字列関数) 68
- E**
- Encode (文字列関数) 69
 - Eval (その他の関数) 63
 - Exists (論理関数) 60
- F**
- Floor (演算関数) 31
 - for 式、FormCalc 18
 - foreach 式、FormCalc 18
 - Format (文字列関数) 69
 - FormCalc
 - 演算子 9
 - 関数呼び出し、FormCalc 24
 - 行終端子 11
 - 空白文字 11
 - 組み込み関数 24
 - コメント 9
 - 参照構文のショートカット 20
 - 式 12
 - 識別子 11
 - 制限されているキーワード 10
 - 説明 6
 - 変数 20
 - リテラル 7
 - 論理関数 60
 - 論理式 14
 - FormCalc 関数
 - URL 79
 - アルファベット順リスト 26
 - 演算 29
 - 会計 52
 - 日付/時間 36
 - FV (会計関数) 53
- G**
- Get (URL 関数) 79
- H**
- HasValue (論理関数) 61
- I**
- if 式、FormCalc 17
 - IPmt (会計関数) 54
 - IsoDate2Num 関数 45
 - IsoTime2Num 関数 46
- L**
- Left (文字列関数) 70
 - Len (文字列関数) 71
 - LocalDateFmt 関数 46
 - LocalTimeFmt 関数 47
 - Lower (文字列関数) 71
 - Ltrim (文字列関数) 72
- M**
- Max (演算関数) 31
 - Min (演算関数) 32
 - Mod (演算関数) 33
- N**
- NPV (会計関数) 55
 - null 値 30
 - Null (その他の関数) 63
 - Num2Date 関数 47
 - Num2GMTIME 関数 48
 - Num2Time 関数 48
- O**
- Oneof (論理関数) 61
- P**
- Parse (文字列関数) 72
 - Pmt (会計関数) 55
 - Post (URL 関数) 79
 - PPmt (会計関数) 56
 - Put (URL 関数) 80
 - PV (会計関数) 57
- R**
- Rate (会計関数) 57
 - Ref (その他の関数) 64
 - Replace (文字列関数) 73
 - Right (文字列関数) 73
 - Round (演算関数) 33
 - Rtrim (文字列関数) 74
- S**
- Space (文字列関数) 74
 - Str (文字列関数) 75
 - Stuff (文字列関数) 75
 - Substr (文字列関数) 76
 - Sum (演算関数) 34
- T**
- Term (会計関数) 58
 - Time 関数 49
 - Time2Num 関数 49
 - TimeFmt 関数 50

U

UnitType (その他の関数) 64
 UnitValue (その他の関数) 65
 Upper (文字列関数) 77
 URL FormCalc 関数 79
 URL へのデータのアップロード 80
 URL へのデータのポスト 79
 UUID (Universally Unique Identifier) 77
 Uuid (文字列関数) 77

W

while 式、FormCalc 17
 Within (論理関数) 62
 WordNum (文字列関数) 78

あ

アルファベット順関数リスト、FormCalc 26

え

エスケープシーケンス、ユニコード 8
 エポック、FormCalc 36
 演算関数、FormCalc 29
 演算子、FormCalc 9
 演算値、型変換 13

か

会計関数、FormCalc 52
 空スペース、文字列 74
 環境ロケール 43
 関係式、FormCalc 16
 関数呼び出し、FormCalc 24

き

キーワード、制限 10
 行終端子、FormCalc 11

く

空白
 説明 11
 文字列から削除 72, 74
 空白文字の削除 72, 74
 空文字列 8

け

係数 33

こ

構文、参照する 20
 コメント、FormCalc 9

さ

参照構文
 FormCalc のショートカット 20
 ショートカット 20
 説明 20

し

式
 FormCalc 12
 識別、一意 77
 識別子、FormCalc 11
 終端子、行、FormCalc 11
 ショートカット、参照構文 20
 時間関数、「FormCalc 関数」を参照
 時間形式
 FormCalc 38
 string 47, 50
 説明 37
 条件ステートメント、FormCalc 17, 18, 19

す

数学関数、FormCalc 29
 数字
 テキストへの変換 78
 文字列への変換 75
 数値の処理 13
 数値リテラル、FormCalc 7
 スクリプティングの説明 6

た

単項式、FormCalc 15
 単純式、FormCalc 12
 ダウンロード、URL コンテンツ 79

て

デフォルトロケール 43

と

等価式、FormCalc 15

は

配列の参照 23
 パターン
 日付/時間 38
 パターン形式
 従って解析 72
 適用 69
 日付/時間 38

ひ

日付/時間フィールドオブジェクト
 FormCalc 関数 36
 パターン作成用の記号 38
 日付形式
 FormCalc 38
 string 44, 46
 説明 36
 日付と時間のパターンの記号 38

ふ

不等価式、FormCalc 15

へ

変換
 大文字と小文字 71, 77
 時間文字列と数値 49
 数字からテキストへ 78
 数字から文字列へ 75

変数

FormCalc 20

も

文字
 大文字と小文字の変換 71, 77
 開始位置 67
 文字列から抽出 70, 73
 文字列の空白文字の削除 72, 74
 文字列関数 67
 文字列の結合 68
 文字列の処理 13
 文字列リテラル、FormCalc 8

ゆ

ユニコードエスケープシーケンス 8

リ

リテラル、FormCalc 7

ろ

ロケール

説明 40

論理関数、FormCalc 60

論理式、FormCalc 14

わ

割り当て式、FormCalc 14