



# Caché ^%R ルーチンの使 用法

Version 5.1  
2006-03-14

Caché ^%R ルーチンの使用法

Caché Version 5.1 2006-03-14

Copyright © 2006 InterSystems Corporation.

All rights reserved.

このドキュメントは、Sun Microsystems、RenderX Inc.、アドビ システムズ および ワールドワイド・ウェブ・コンソーシアム (www.w3c.org) のツールと情報を使用して、Adobe Portable Document Format (PDF) で作成およびフォーマットされました。主要ドキュメント開発ツールは、InterSystemsが構築したCaché と Javaを使用した特別目的のXML処理アプリケーションです。



Caché 製品とロゴは InterSystems Corporation の登録商標です。



Ensemble 製品とロゴは InterSystems Corporation の登録商標です。



InterSystems という名前とロゴは InterSystems Corporation の登録商標です

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を稼動および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

Caché および InterSystems Caché、Caché SQL、Caché ObjectScript および Caché Object は、インターシステムズ社の商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems ワールドワイド カスタマサポート

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

# 目次

Caché ^%R ルーチンの使用法.....	1
ParseRoutineName ^%R .....	5
ROUTINE ^%R .....	7
FMTERR ^%R .....	10
CHECK ^%R .....	12
DATE ^%R .....	13
DEL ^%R .....	15
EXISTS ^%R .....	16
LANG ^%R .....	17
LANGSET ^%R .....	19
LENGTH ^%R .....	19
SIZE ^%R .....	21
LINE ^%R .....	23
LINESET ^%R .....	25
LOCK ^%R .....	29
UNLOCK ^%R .....	29
VERMAX ^%R .....	30
VERMAXSET ^%R .....	33
VERSION1 ^%R .....	34
VERSION ^%R .....	35
テーブル一覧	
名前の解析 .....	4
一般的なルーチン操作 .....	4
バージョン制御 .....	4



# Caché ^%R ルーチンの使用法

## 背景

Caché の以前のリリースにおいて、別のルーチンによるルーチンの生成および操作は、付加的に行なわれていました。つまり、ルーチンの制御は ZLOAD、ZINSERT、ZREMOVE および関連するコマンドを使用して、1 行ずつルーチンを構築、操作、コンパイル、保存していました。

この方法はエラーを分離するには有効ですが、効率的ではありません。ObjectScript コンパイラの進化に伴い、コンパイラの開始と終了に関連するコストが、コードの 1 行をコンパイルするコストに比べて、大幅に増大しました。その結果、プログラミングにより生成されたルーチンのコンパイル速度を改善するために、Caché バージョン 4 で %R ルーチンが導入されました。%R ルーチンにより、生成されたルーチンを 1 行ごとではなくユニットとして処理することができるようになりました。

1 行ごとにルーチンを操作することも可能ですが、%R を使用する方法を推奨します。

注釈: 警告に対し、2 つの対処法があります。アプリケーションが ZINSERT への各呼び出しの後に構文エラーの有無を判断するのに、\$ZUTIL(62, 0) を使用する場合、常にエラーは発生していないと報告します。以前は、ZINSERT は挿入を実行するたびにコンパイルを実行していましたが、現在は ZSAVE コマンドが実行されるまでコンパイルしません。しかし、ZSAVE は \$ZUTIL(62, 0) 機能経由ではエラーを報告しません。

## ルーチン名

ルーチン名を供給する文字列は、%R のエントリ・ポイントの多くで使用されています。以下では、この名前をコンポーネントの一部に評価するための一般的な方法について説明します。

通常、ルーチン名は 3 つの部分 (ベース名、ルーチン拡張子、この 2 つの要素が指定するソースのバージョン) から構成され、各部分はピリオドで区切られています。しかし、以下のような場合は、名前が正確に解釈できない可能性があります。

- ・ ルーチン名の一部として、ネームスペースが含まれている場合
- ・ 各拡張子やバージョンの要素がオプションで、任意の呼び出しの中に提供されていない場合
- ・ 特定の条件下で、ワイルドカード文字 (\*) が各コンポーネントのすべて、または一部として許可される場合
- ・ Caché バージョン 4 に関し、ベース名の一部としてピリオドが有効である場合 (ルーチンを含むパッケージを指定します)

従来のプログラムが ^%R 機能と互換を持つようにするため、以下の方法により、文字列をベース名、拡張子、バージョン、ネームスペースの構成部分に解析します。

1. ネームスペース構成要素が空文字列に初期化されます。

---

2. 文字列に “[” または "]" の文字が含まれている場合は、フォームの形式であるとみなします。

・ [“<NAMESPACE>”]<REMAINDER>

あるいは

・ |“<NAMESPACE>”|<REMAINDER>

ネームスペースの要素を、“(” の間から取り出します。文字列は <REMAINDER> に設定され、処理を継続します。

3. 文字列がワイルドカード文字 (\*) のみから構成されている場合は、拡張子とバージョンの要素は空文字列に設定されます。ベース名はワイルドカード文字に設定され、名前の処理が終了します。

4. 文字列が “<TEXT>.\*.\*” という形式であれば、拡張子とバージョンは “\*” に、ベース名は <TEXT> に設定され、処理が完了します。

5. 文字列が “<TEXT>.\*” という形式であれば、バージョンは 0、拡張子は “\*”、ベース名は <TEXT> に設定され、処理が完了します。

6. 文字列が “<TEXT>.<EXT>.\*” という形式で、<EXT> が有効な拡張子 (以下のうちのいずれか) であれば、

- ・ MAC
- ・ INT
- ・ INC
- ・ OBJ
- ・ BAS
- ・ COS

(この場合、大文字・小文字は関係ありません) バージョンは “\*” に、拡張子は <EXT> に、ベース名は <TEXT> に設定されます。

そうでない場合は、バージョンは 0、拡張子は “\*”、ベース名は <TEXT>.<EXT> に設定され、処理が完了します。

7. 文字列が “<TEXT>.<EXT>.<VER>” という形式で、EXT が上記の有効な拡張子のうちの 1 つであり、VER が整数であれば (符号付きまたは符号なし)、バージョンは VER、拡張子は <EXT>、ベース名は <TEXT> に設定されます。

そうでない場合は、バージョンは 0、拡張子は “\*”、ベース名は <TEXT>.<EXT>.<VER> に設定され、処理が完了します。

8. そうでない場合は、バージョンは 0、拡張子は “\*”、ベース名は文字列の残りの文字に設定され、処理が完了します。

---

上記は、名前の構文解析のアルゴリズムの一般的な説明です。名前の構文解析の例は、以下の [ParseRoutineName](#) エントリ・ポイントを参照してください。これは、`^%R` のその他のエントリ・ポイントによって使用されるネーム・パーサーです。

### 言語のエンコード

いくつかのエントリ・ポイントでは、整数値を使用してサブジェクト・ルーチンが記述されている言語を識別します。言語を指定するエンコードは以下のとおりです。

- ・ 0 - Caché ObjectScript (デフォルト)
- ・ 1 - DSM-11
- ・ 2 - DTM
- ・ 3 - 未使用 - 廃止
- ・ 4 - 未使用 - 廃止
- ・ 5 - DSM-VMS
- ・ 6 - DSM-J
- ・ 7 - DTM-J
- ・ 8 - MSM
- ・ 9 - Caché Basic

1 から 8 までの値を使用して、古いシステム間のソース・コードの解釈の微妙な相違点に対し、コンパイラをカスタマイズします。

注釈: Caché の現在のユーザは、0 または 9 を使用するようになしてください。その他は、古いアプリケーションの維持および更新に使用してください。

### バージョン

`%R` 内のエントリ・ポイントには、ROUTINE エントリ・ポイントへの "B" オプションと一緒に使用するとき、制約付きのバージョン制御形式を提供するものがあります。これらは、使用可能な特定のルーチンの "古い" バージョンを指定数維持し、必要に応じて再生できるように準備します。詳細は、"[VERMAX](#)"、"[VERMAXSET](#)"、"[VERSION1](#)"、"[VERSION](#)" を参照してください。

注釈: バージョンをいっさい使用する必要がない場合は、最大をゼロに設定できます。別の方法として、システム・グローバルの `^rBACKUP`、`^rMACSAVE` および `^rINCSAVE` を停止することができます。

### エントリ・ポイント

`^%R` で公式にサポートされているエントリ・ポイントを機能別に分類して、以下に示します。

## 名前の解析

エントリー・ポイント	目的
ParseRoutineName	ルーチン名を、構成部分に分離します

## 一般的なルーチン操作

エントリー・ポイント	目的
ROUTINE	データベースのソース・コードをコンパイルして保存、またはロードします
FMterr	エラーを表示します
CHECK	ソース・コードの構文を確認します
DATE	保存されたルーチンの日付を取得します
DEL	データベースからルーチンを削除します
EXISTS	ルーチンの有無を確認します
LANG	ルーチンのソース言語インジケータを取得します
LANGSET	ルーチンのソース言語インジケータを設定します
LENGTH	ルーチンのソース行をカウントします
SIZE	ルーチンのソース文字をカウントします
LINE	保存されたルーチンから 1 行を返します
LINESET	保存されたルーチンに、新しいソース行を挿入/置換します。
LOCK	保存されたルーチンの排他的使用を試みます
UNLOCK	保存されたルーチンの排他的使用を中止します

## バージョン制御

エントリー・ポイント	目的
VERMAX	バックアップとして保持するバージョンの最大数を返します
VERMAXSET	バックアップとして保持するバージョンの最大数を設定します
VERSION1	最も古いバックアップのバージョン番号を返します
VERSION	関連するバージョン番号に一致するバージョン番号を返します

Caché ObjectScript 関数についての詳細は、“Caché ObjectScript の使用法”の“関数”を参照してください。

## ParseRoutineName^%R

```
ParseRoutineName^%R (rtn, .extent, .version, .namesp)
```

### パラメータ

rtn	ターゲット・ルーチンの名前
extent	パースされた拡張子を持つ文字列
version	バージョン番号を表す文字列
namesp	(ルーチン名にネームスペースがあれば)ネームスペースを表す文字列

### 概要

この関数は、^%R のその他のエントリ・ポイントが使用するネーム・パーサーへのアクセスを提供します。

### パラメータ

#### rtn

データベースから取得する、またはデータベースに保存するルーチンの名前を指定する文字列。ルーチンの名前全部、あるいは一部を表します。この名前は、大文字と小文字を区別します。拡張子は、大文字と小文字を区別しません。

#### extent

出力引数。パースされた拡張子を保持します。

#### version

出力引数。パースされたバージョン番号を保持します。

#### namesp

rtn が明示的なネームスペース構成要素を含む場合、この出力引数がそれを含みます。含まない場合は、空文字列に設定されます。

## 備考

この関数は、^%Rのその他のエントリ・ポイントが使用する内部ルーチンです。これはパブリック・エントリ・ポイントですが、パース・ルールが実際どのように適用されるのかを知りたい開発者の方のために、ここで説明します。

この関数は、結果として、出力引数とベース・ルーチン名を返します。

## 例

以下の例は、さまざまなルーチン名について ParseRoutineName^%R の使用方法を示します。

```
; Build a list of names
Set samples = $LISTBUILD("foo",
    "foo.bar",
    "foo.mac",
    "foo*.bar",
    "foo*.*.13",
    "foo.mac.-234",
    "^|" "DeltaQuadrant" |Voyager.int.1",
    "^[" "^AlphaQuadrant" ]NCC.1701.MAC.4",
    "^[" "twilight" ,"zone" ]Somewhere.INT.19")

; show the results
For i = 1 : 1 : $LISTLENGTH(samples)
{
    Set (Ext, Ver, Nsp) = "???"
    Set Input = $LIST(samples, i)
    Set BaseName = $$ParseRoutineName^%R(Input, .Ext, .Ver, .Nsp)
    Write Input, !
    Write ?3, "Base:", ?15, BaseName, !
    Write ?3, "Extension:", ?15, Ext, !
    Write ?3, "Version:", ?15, Ver, !
    Write ?3, "Namespace:", ?15, Nsp, !, !
}
}
```

# ROUTINE^%R

```
ROUTINE^%R (rtn, .code, .errs, options, langmode, filedate, namesp, iunlock)
```

## パラメータ

rtn	ターゲット・ルーチンの名前
code	ソース・コード配列への参照
errs	処理中に検出されたエラーのリストへの参照
options	希望する処理の選択肢を含む文字列
langmode	ソースの言語を指定するインジケータ
filedate	結果として作成されたファイルで使用するタイムスタンプ
namesp	ルーチンが処理されるネームスペース
iunlock	処理の完了後に直ちにルーチンのロックを解除することを示すスイッチ

## 概要

この関数は、Caché ObjectScript プログラムがプログラムの的にソース・コードを操作することを許可します。関数に渡されるパラメータが実行を指定するプログラムは以下のとおりです。

- ・ 後で使用するためにソース・コードを保存します。
- ・ 保存されたソース・コードを検索します。
- ・ ソース・コードをコンパイルし、オプションでコンパイルされたオブジェクト・コードを保存します。

## パラメータ

### rtn

有効な文字列として評価する式。文字列の内容は、データベースから取得する、またはデータベースに保存するルーチンの名前を指定します。“SomeRoutine.MAC”などのように、名前と拡張子がピリオドで区切られた完全なルーチン名を含む必要があります。この名前は、大文字と小文字を区別します。拡張子は大文字と小文字を区別しません。

### code

ソース・コードの配列。この配列は、ルーチンのコンパイル時、あるいは保存時に、ソース・コードを提供します。ルーチンがフェッチされたときに、ソースを受け取ります。

配列の形式は以下のとおりです。

- ・ code(0) は、このオペレーションに関連するソース・コードの行数を含みます。
- ・ code(1) から code(N) までは、ソース・コードの各行を含みます。N は、Code(0) に含まれている値です。

## errs

options が指定する処理の実行を試行する間に検出されたエラーのリスト。値の \$LIST として引数 err が返されます。値は、`FMterr^%R` 関数を呼び出して表示できます。

## options

ルーチンで試行される処理を示す一連の文字からなる文字列。許可される値およびその意味は、以下のとおりです。

- ・ L - 事前に保存されたルーチンを code パラメータにロードします。
- ・ C - code に含まれるルーチンをコンパイルします。
- ・ D - ターゲット・ルーチンの保存されたソースを削除します。
- ・ S - ルーチン・ソースを保存し、コンパイルされたバージョンも (あれば) 保存します。
- ・ B - タイプ "MAC" または "INT" のルーチンを保存する際、保存を実行する前にバックアップ・バージョンを作成します。

文字列は、大文字と小文字を区別しません。処理が試行される順番は、文字列に配置されている順番通りです。

## langmode

code の文が記述されている言語を示す整数。

## filedate

Caché のルーチンが修正された日付として使用するタイムスタンプ。filedate の値は、\$HOROLOG 形式で指定します (詳細は、"Caché ObjectScript リファレンス" を参照)。filedate が指定されていない場合は、現在の日付と時間を使用します。

## namesp

指定された処理を実行するネームスペースを値に持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## iunlock

ブーリアン値。真 (1) は、処理の完了後に直ちにルーチンに対するすべてのロックが解放されることを示します。偽 (既定) の場合は、保持されたすべてのロックが以前のバージョンの Caché と同じように扱われます。

## 備考

このルーチンは、ルーチンで最初の引数として指定された処理の実行を試行し、結果を、以下の形式の文字列で返します。

・ N<sup>^</sup>Status1,Status2,...

N の値は、この試行が最終的に成功したか、あるいは失敗したかを示します。N が 1 であれば、すべての処理が成功したことを表し、N が 0 であれば、失敗した処理があることを表します。

Status1、Status2、Status3 などは、対応するオプションの結果の要約を表します。

## 例

以下の例は、ROUTINE<sup>^</sup>%R を使用して 1 つのルーチンをコンパイル、保存、実行する方法を示しています。これは、ルーチンを直接実行します。

この例は、このドキュメントで説明されている、<sup>^</sup>%R 内の他のエントリ・ポイントを利用しています。

**警告!** この例は、**AnExample.INT** という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

## FMterr^%R

```
; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
; Note leading spaces on the following entries"
Set code($INCREMENT(N)) = "    Write "Starting AnExample"", !"
Set code($INCREMENT(N)) = "    Set Y = 1"
Set code($INCREMENT(N)) = "    Set Z = 3"
Set code($INCREMENT(N)) = "    Write Y, "" + "", Z, "" = "", (Y + Z), !"
Set code($INCREMENT(N)) = "    Write "Finished AnExample"", !"
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set name = "AnExample"
Set ext = "INT"
Set routine = name _ "." _ ext
Set options = "CS"          ; Compile and Save
Set errors = ""            ; empty list

; do it
Set return = $$ROUTINE^%R(routine, .code, .errors, options)

; show the simple result
Write "Compilation result: ", return, !
If (+return = "")
{
    ; format and display the errors
    Write $$FMterr^%R(.errors, .code), !
}
Else
{
    ; run it
    Write "Calling AnExample", !
    Set entrypoint = name _ "^" _ name
    Do @entrypoint
    ; remove the source and object
    for suffix = "int", "OBJ"
    {
        Set component = name _ "." _ suffix
        Write "Removing ", component, ": ", $$DEL^%R(component), !
    }
}
}
```

## FMterr^%R

```
FMterr^%R (.errs, .code, .lines)
```

### パラメータ

errs	処理中に検出されたエラーのリストへの参照
code	ソース・コード配列への参照
lines	見つかったエラーの解釈を含むテキスト行の配列への参照

## 概要

この関数は、ROUTINE^R あるいは CHECK^R 呼び出しにより生成されたエラーのデータを、ユーザが見やすい表示形式に変換します。

## パラメータ

### errs

ROUTINE^R または CHECK^R の呼び出しにより返された、エラーのリスト。

### code

ソース・コードを含む配列。この配列は、ルーチンのソース・コードを供給するので、表示された情報には、テキストの一部として、エラーを起こしたソース行が含まれます。このフォーマットは、ROUTINE^R エントリ・ポイントが使用するものと同じです。提供されていない場合は、エラー・メッセージには、エラーの内容ではなくエラーが発生した位置のみが表示されます。

### lines

エラー・メッセージを生成するテキスト行が書き込まれる配列への参照。このフォーマットは、ソース・コード配列と同じで、lines(0) は、配列内のエラー・メッセージの行数を含みます。

このパラメータが提供されていない場合、結果として生じるテキストは、関数呼び出しの値として返されます。\$CHAR(13, 10) がある場所には、返される文字列で強制改行が入ります。

lines が提供されている場合、各配列項目は、別の行を表します。内部の強制改行も、返される結果もありません。つまり、ルーチンは、(DO コマンド経由で) サブルーチンとして呼び出すようにしてください。

## 備考

このルーチンは、ROUTINE^R に返されたエラー・リストを、ユーザが使いやすい形式に再フォーマットします。

## 例

以下は、不正なプログラム上で CHECK^R を使用して、エラー配列を生成します。この配列は、その後テキストに変換され、表示されます。

## CHECK^%R

---

```
; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
Set code($INCREMENT(N)) = "    Write "Starting AnExample", !"
Set code($INCREMENT(N)) = "    Write "A closing quote is missing here^, !"
Set code($INCREMENT(N)) = "    Quiet"
Set code(0) = N

Set errors = ""                ; empty list

; do it
Set return = $$CHECK^%R(.code, .errors)

; show the simple result
Write return, !
If (return = 0)
{
    ; format and display the errors
    Write !, "Errors as one text string", !
    Write $$FMTEERR^%R(.errors, .code), !
    Write !, "Errors as lines from an array", !
    Set lines = ""
    Do FMTEERR^%R(.errors, .code, .lines)
    for i = 1 : 1 : lines(0)
    {
        Write "errors(", i, ") = ", lines(i), !
    }
}
Else
{
    ; should never get here
    Write "Something is radically wrong", !
}
}
```

## CHECK^%R

---

CHECK^%R (.code, .errs, langmode)

### パラメータ

code	ソース・コード配列への参照
errs	処理中に検出されたエラーのリストへの参照
langmode	ソースの言語を指定するインジケータ

### 概要

この関数は、提供されたソース・コードの構文チェックを実行し、見つかったエラーを返します。

## パラメータ

### code

ソース・コードを含む配列。この配列は、正しい構文をチェックするルーチンのソース・コードを提供します。このフォーマットは、ROUTINE<sup>^</sup>%R エントリ・ポイントが使用するものと同じです。

### errs

構文チェッカーによって報告されたエラーのリスト。値の \$LIST として返されます。FMTERR<sup>^</sup>%R ルーチンを呼び出すことにより、その値を表示できます。

### langmode

code の文が記述されている言語を指定する整数。許可されている値は、ROUTINE<sup>^</sup>%R で許可されているものと同じです。

## 備考

このルーチンは関数として呼び出され、結果として 0 または 1 を返します。結果の意味は以下のとおりです。

- ・ 1 - 成功しました。構文エラーはありません。
- ・ 0 - 失敗しました。構文エラーが見つかり、errs 配列に返されました。

注釈: 返り値 1 は、提供されたソースがコンパイル時にエラーを生じないことを示します。ソース・コードの実行時にエラーが発生しない、あるいは意図した結果を得られることを意味しているわけではありません。

## 例

以下の例は、CHECK<sup>^</sup>%R のインスタンスに対する FMTERR<sup>^</sup>%R ルーチンの例です。

## DATE<sup>^</sup>%R

```
DATE^%R (rtn, format, namesp)
```

### パラメータ

rtn	ターゲット・ルーチンの名前
format	希望するタイム・スタンプの形式
namesp	ルーチンが保存されているネームスペース

## 概要

この関数は、指定されたルーチンが Caché に保存された日時を返します。

## パラメータ

### rtn

データベースから取得する、またはデータベースに保存するルーチンの名前を指定している文字列。“SomeRoutine.MAC” などのように、名前と拡張子がピリオドで区切られた完全なルーチン名を含む必要があります。この名前は、大文字と小文字を区別します。拡張子は、大文字と小文字を区別しません。

指定されたルーチンが見つからなければ、この関数は NULL 文字列を返します。

### format

返される日付と時間の形式を指定する整数。使用できる値は、\$ZDATETIME 関数と同じです。format の値を使用して、日付部分をフォーマットします。時間は常に “HH:MM:SS” の形で表されます。

\$ZDATETIME で許可されている値以外では、0 を指定すると、タイム・スタンプを \$HOROLOG 形式で返します。

与えられた format 値が許可されたものでなければ、関数の値としてエラー・ステータスが返されます。以下はその例です。

### namesp

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、ルーチンに関連するタイム・スタンプを、ルーチンが保存された時間で返します。

注釈: ROUTINE^%R の呼び出し元が filedate パラメータで異なる値を提供している場合は、この値はルーチンが保存された実際の日付や時間と異なる値になります。

## 例

以下の例は、許可されているすべての形式で %R が保存された日付と時間を表示します。

```

; set up the data
Set name = "%R"
Set ext = "OBJ"
Set routine = name _ "." _ ext
Set namespace = "%CACHELIB"

; do it
Write "Timestamp for ", routine, !
for fmt = 0 : 1 : 12
{
    Set filedate = $$DATE^%R(routine, fmt, namespace)
    Write "Format ", fmt, ": ", ?10, filedate, !
}

```

以下の例は、format の値が不正である場合に発生するエラー・ステータスを表示します。

```

; set up the data
Set name = "%R"
Set ext = "OBJ"
Set routine = name _ "." _ ext
Set namespace = "%CACHELIB"

; do it
Write "Timestamp for ", routine, !
Set filedate = $$DATE^%R(routine, 100, namespace)
Write "Value: ", ?10, filedate, !

```

指定された名前のルーチンがない場合は、以下の例のような結果になります。関数の値として NULL 文字列が返されます。

```

; set up the data
Set name = "%NONEXISTENT"
Set ext = "OBJ"
Set routine = name _ "." _ ext
Set namespace = "%CACHELIB"

; do it
Write "Timestamp for ", routine, !
Set filedate = $$DATE^%R(routine, 1, namespace)
Write "Return length: ", $LENGTH(filedate), !

```

## DEL^%R

DEL^%R (rtn, namesp)

### パラメータ

rtn	ターゲット・ルーチンの名前
namesp	ルーチンが保存されているネームスペース

### 概要

この関数は、指定されたルーチンを Caché データベースから削除します。

## パラメータ

### rtn

データベースから取得する、またはデータベースに保存するルーチンの名前を提供する文字列。この名前は、大文字と小文字を区別します。拡張子は、大文字と小文字を区別しません。

この名前と拡張子には、ワイルドカードが含まれます。例えば、“Foo\*.\*” は、拡張子にかかわらず最初の 3 文字が “Foo” であるすべてのルーチンを削除します。

### namesp

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、指定されたルーチンをデータベースから削除します。少なくとも 1 つのルーチンが見つかり削除された場合は 1 を返し、それ以外の場合は 0 を返します。

## 例

DEL<sup>^</sup>%R のインスタンスに対する ROUTINE<sup>^</sup>%R ルーチンの例を参照してください。

## EXISTS<sup>^</sup>%R

```
EXISTS^%R (rtn, namesp)
```

## パラメータ

rtn	ターゲット・ルーチンの名前
namesp	ルーチンが保存されているネームスペース

## 概要

この関数は、Cache データベース内に指定されたルーチンが存在するか否かを判断します。

## パラメータ

### rtn

検索されるルーチンの名前を指定する文字列。

この名前と拡張子には、ワイルドカードを含むことができます。

## namesp

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、指定されたネームスペースに rtn と一致するルーチンがあれば 1 を返し、なければ 0 を返します。

## 例

以下は、この資料の件名の有無を確認します。

```

; set up the data
Set ext = "OBJ"
Set namespace = "%CACHELIB"

; do it
for basename = "%R", "ArbitraryName"
{
    Set routine = basename _ "." _ ext
    Set present = $$EXIST^%R(routine, namespace)
    Write routine, $SELECT(present:" exists", 1:" is missing"), "!", !
}

```

# LANG<sup>^</sup>%R

LANG<sup>^</sup>%R (rtn, namesp)

## パラメータ

rtn	ターゲット・ルーチンの名前
namesp	ルーチンが保存されているネームスペース

## 概要

この関数は、指定されたルーチンの言語コードの値を返します。

## パラメータ

### rtn

有効な文字列として評価する式。文字列の内容により、検索するルーチンの名前を指定します。

## namespace

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、指定されたルーチンの言語モードをエンコードする整数値を返します。エンコードのリストは、このドキュメントの最初の部分に記載されています。

ルーチンの拡張子が無効である場合は、関数は空文字列を返します。拡張子が“MAC”であれば、ルーチンの有無にかかわらず、返される値は 0 (ObjectScript) です。

## 例

以下の例では、いくつかのルーチンについて、記述される言語を確認します。

```
; set up the data
Set namespace = "%CACHELIB"

set languages = $LISTBUILD("ObjectScript",
                           "DSM-11",
                           "DTM",
                           "",
                           "",
                           "DSM-VMS",
                           "DSM-J",
                           "DTM-J",
                           "MSM",
                           "Basic")

; do it
for routine = "%R.OBJ", "ArbitraryName.MAC", "SomeOther.RTN"
{
  Set code = $$LANG^%R(routine, namespace)
  If ($ISVALIDNUM(code))
  {
    Set lang = $LISTGET(languages, (code + 1), "")
    Set:(lang = "") lang = "Unknown"
    Write routine, " language: ", lang, !
  }
  Else
  {
    Write routine, ": invalid extension", !
  }
}
```

# LANGSET<sup>^</sup>%R

```
LANGSET^%R ( rtn, langmode )
```

## パラメータ

rtn	ターゲット・ルーチンの名前
langmode	ソースの新規の言語を指定するインジケータ

## 概要

この関数は、指定されたルーチンの言語コードの値を返します。

## パラメータ

### rtn

検索されるルーチンの名前を指定する文字列。

### langmode

rtn の文が記述されている言語を指定する整数。

## 備考

この関数は、データベース内のルーチンのソースが保存された言語エンコードを変更します。これは、従来のアプリケーションのための関数です。従来のアプリケーションは、複数のプラットフォームで動作し、実行時に環境に適応する必要があります。

この関数は、保存された言語モードの変更に成功した場合は 1 を返し、失敗した場合は 0 を返します。

## 例

# LENGTH<sup>^</sup>%R

```
LENGTH^%R ( rtn, namesp )
```

## パラメータ

rtn	ターゲット・ルーチンの名前
namesp	ルーチンが保存されているネームスペース

## 概要

この関数は、データベースに保存されたルーチンの行数を返します。

## パラメータ

**rtn**

検索されるルーチン (文字列として) の名前。

**namesp**

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、データベースに保存されたルーチンの行数を整数値で返します。ルーチンが存在しない場合は、0 を返します。

## 例

以下はルーチンを生成し、それをデータベースに保存します。これは、ルーチンのメモリ内の情報を消去し、その情報をデータベースに問い合わせます。

この例は、このドキュメントで説明されている、<sup>^</sup>%R 内の他のエントリ・ポイントを利用しています。

### 警告!

この例は、**AnExample.INT** という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

```

; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
; Note leading spaces on the following entries"
Set code($INCREMENT(N)) = "    Write "Starting AnExample", !"
Set code($INCREMENT(N)) = ""
Set code($INCREMENT(N)) = "    Write "Some text", !"
Set code($INCREMENT(N)) = ""
Set code($INCREMENT(N)) = "    Write "Finished AnExample", !"
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set routine = "AnExample.INT"
Set options = "S"           ; Save
Set errors = ""           ; empty list

; do it
Set return = $$ROUTINE^%R(routine, .code, .errors, options)

; show the simple result
Write "Save result: ", return, !
If (+return = "")
{
    ; format and display the errors
    Write $$FMTErr^%R(.errors, .code), !
}
Else
{
    ; remove local info
    Kill code

    ; find out about it
    Write "Lines in ", routine, ": ", $$LENGTH^%R(routine), !

    ; remove the saved source
    Write "Removing ", routine, ": ", $$DEL^%R(routine), !
}

```

## SIZE^%R

SIZE^%R (rtn, namesp)

### パラメータ

rtn	ターゲット・ルーチンの名前
namesp	ルーチンが保存されているネームスペース

### 概要

この関数は、データベースに保存されたルーチンの文字数を返します。

## パラメータ

### rtn

検索されるルーチンの名前を提供する文字列。

### namesp

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、データベースに保存されたルーチンの文字数を整数値で返します。ルーチンが存在しない場合は、0 を返します。

## 例

以下はルーチンを生成し、それをデータベースに保存します。これは、ルーチンのメモリ内の情報を消去し、その情報をデータベースに問い合わせます。

この例は、このドキュメントで説明されている、<sup>^</sup>%R 内の他のエントリ・ポイントを利用しています。

### 警告!

この例は、**AnExample.INT** という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

```

; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
; Note leading spaces on the following entries"
Set code($INCREMENT(N)) = "    Write "AnExample", !"
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set routine = "AnExample.INT"
Set options = "S"           ; Save
Set errors = ""            ; empty list

; do it
Set return = $$ROUTINE^%R(routine, .code, .errors, options)

; show the simple result
Write "Save result: ", return, !
If (+return = "")
{
    ; format and display the errors
    Write $$FMTErr^%R(.errors, .code), !
}
Else
{
    ; remove local info
    Kill code

    ; find out about it
    Write "Characters in ", routine, ": ", $$SIZE^%R(routine), !

    ; remove the saved source
    Write "Removing ", routine, ": ", $$DEL^%R(routine), !
}

```

## LINE^%R

LINE^%R (rtn, linenum, namesp)

### パラメータ

rtn	ターゲット・ルーチンの名前
linenum	希望する行数
namesp	ルーチンが保存されているネームスペース

### 概要

この関数は、データベースに保存されたルーチンからソースの行を返します。

## パラメータ

### rtn

検索されるルーチン名。

### linenum

希望するコンテンツを持つルーチンの行の番号。ルーチンの最初の行の番号は常に 1 です。

### namesp

指定されたルーチンが保存されているネームスペースを表す値を持つ文字列。このパラメータはオプションです。指定されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、希望する行の内容を文字列として返します。linenum が以下の制約を満たしていない場合、

- ・ `1 <= linenum <= $$LENGTH^%R(rtn, namesp)`

その行は存在せず、関数は結果として空文字列を返します。この関数だけでは、存在しない行なのか、行に文字が含まれていないのかを見分けることはできません。

## 例

以下はルーチンを生成し、それをデータベースに保存します。これは、ルーチンのメモリ内の情報を消去し、その情報をデータベースに問い合わせます。

この例は、このドキュメントで説明されている、<sup>^</sup>%R 内の他のエントリ・ポイントを利用しています。

### 警告!

この例は、**AnExample.INT** という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

```

; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
; Note leading spaces on the following entries"
Set code($INCREMENT(N)) = "    Write "AnExample", !"
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set routine = "AnExample.INT"
Set options = "S"           ; Save
Set errors = ""            ; empty list

; do it
Set return = $$ROUTINE^%R(routine, .code, .errors, options)

; show the simple result
Write "Save result: ", return, !
If (+return = "")
{
    ; format and display the errors
    Write $$FMTERR^%R(.errors, .code), !
}
Else
{
    ; remove local info
    Kill code

    ; list the routine backwards
    For i = $$LENGTH^%R(routine) : -1 : 1
    {
        Write "Line ", i, ":: ", $$LINE^%R(routine, i), !
    }

    ; remove the saved source
    Write "Removing ", routine, ": ", $$DEL^%R(routine), !
}
}

```

## LINESET<sup>^</sup>%R

```
LINESET^%R (rtn, linenum, linetext)
```

### パラメータ

rtn	ターゲット・ルーチンの名前
linenum	置換する、または追加する行数
linetext	新規の行の内容

### 概要

この関数は、指定されたルーチンが示す位置にソース・テキストを挿入します。

## パラメータ

### rtn

修正されるルーチンの名前を指定する文字列。

### linenum

希望するコンテンツを持つルーチンの行の番号。ルーチンの最初の行の番号は 1 です。

### linetext

ルーチンに挿入されるソース・テキストの文字列。

## 備考

この関数は、ルーチンの linenum の位置に linetext を挿入します。条件

- ・ linenum > \$\$LENGTH<sup>^</sup>%R(rtn, namesp)

上記の条件に該当する場合、ルーチン・ソースは空の行をソースに追加することにより効率的に拡張され、

- ・ linenum - \$\$LENGTH<sup>^</sup>%R(rtn namesp)

その結果、linenum の範囲は以下のようになります。

- ・ 1 <= linenum <= \$\$LENGTH<sup>^</sup>%R(rtn. namesp)

そして、その行の既存のテキストが置換されます。

この関数は、置換に成功すると結果として 1 を、それ以外の場合は 0 を返します。

注釈: 行を 1 行置換しても、対応するオブジェクト・ルーチンに影響はありません。その変更を有効にするには、ソース・ルーチンをリコンパイルする必要があります。

## 例

以下はルーチンを生成し、それをコンパイルしてデータベースに保存します。これは、ルーチンのメモリ内の情報を消去し、新しく作成されたルーチンを呼び出します。

そして、ソースの行を変更し、ルーチンを呼び出して、オブジェクトにまだ変更がないことを示します。

最後に、データベースから新しいソースをロードして、リコンパイル、保存し、再度このルーチンを呼び出して変更が有効になっていることを表します。

この例は、このドキュメントで説明されている、<sup>^</sup>%R 内の他のエントリ・ポイントを利用しています。

**警告!**

この例は、**AnExample.INT** という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

```

; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
; Note leading spaces on the following entries"
Set code($INCREMENT(N)) = "    Write ""User: """"Hello, world""""", !"
Set code($INCREMENT(N)) = "    ; A dummy line to be replaced later"
Set ReplaceLoc = N
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set name = "AnExample"
Set ext = "INT"
Set routine = name _ "." _ ext
Set options = "CS"           ; Compile and Save
Set errors = ""             ; empty list

; do it
Set return = $$ROUTINE^%R(routine, .code, .errors, options)
If (+return = "")
{
    ; format and display the errors
    Write $$FMTErr^%R(.errors, .code), !
    Quit
}

; run it
Set entrypoint = name _ "^" _ name
Do @entrypoint

Set NewLine = "    Write ""World: """"Hello yourself""""", !"
If ($$LINESET^%R(routine, ReplaceLoc, NewLine) '= 1)
{
    Write "Line replacement failed", !
    Quit
}

; show what is in the local array and in database
Write "Local: ", code(ReplaceLoc), !
Write "Saved: ", $$LINE^%R(routine, ReplaceLoc), !

; run it again to show same result
Do @entrypoint

; load, re-compile and save
Write "Updating...", !
Set return = $$ROUTINE^%R(routine, .code, .errors, "LCS")
If (+return = "")
{
    ; format and display the errors
    Write $$FMTErr^%R(.errors, .code), !
    Quit
}

; run it one last time to show change
Do @entrypoint

; remove the source and object
for suffix = "int", "OBJ"
{
    Set component = name _ "." _ suffix
    Write "Removing ", component, ": ", $$DEL^%R(component), !
}

```

## LOCK^%R

```
LOCK^%R ( rtn, timeout )
```

### パラメータ

rtn	ターゲット・ルーチンの名前
timeout	試行を中止する前に待機する秒数

### 概要

この関数は、データベースに保存されたルーチンへの排他的なアクセスを試みます。

### パラメータ

#### rtn

文字列としてロックされるルーチンの名前です。

#### timeout

ルーチンが、そのルーチンのロックの取得を中止する前に待機する最大秒数を指定する整数。このパラメータはオプションです。これが提供されていない場合は、デフォルトの timeout は 2 です。

### 備考

これは、データベースに保存されたルーチンへの排他的なアクセスを試みます。この試行が成功すれば 1 を、ロックが取得できなければ 0 を返します。

### 例

## UNLOCK^%R

```
UNLOCK^%R ( rtn, iunlock )
```

### パラメータ

rtn	ターゲット・ルーチンの名前
iunlock	処理の完了後に直ちにルーチンのロックを解除することを示すスイッチ

## 概要

この関数は、**UNLOCK** のアクションを取り消します。

## パラメータ

**rtn**

文字列として (ロックされるであろう) ルーチンの名前です。

**iunlock**

ブーリアン値。真 (1) は、処理の完了後に直ちにルーチンに対するすべてのロックが解放されることを示します。偽 (既定) の場合は、保持されたすべてのロックが以前のバージョンの Cache と同じように扱われます。

## 備考

この関数は、**UNLOCK** 経由で取得したルーチンのロックを中止します。この試行が成功すれば 1 を返します。また例えば、ルーチンが以前にロックされていなかったなど、オペレーションが失敗した場合は 0 を返します。

## 例

# VERMAX<sup>^</sup>%R

---

VERMAX<sup>^</sup>%R (extent, namesp)

## パラメータ

extent	希望するバージョン情報を持つエクステント
namesp	このエクステントが保存されているネームスペース

## 概要

この関数は、ルーチンのバージョン番号を、このエクステントと一緒に返します。このエクステントは維持されます。

## パラメータ

**extent**

希望するバージョン情報に対するエクステントを指定する文字列。

## namesp

文字列の内容は、ネームスペースを指定して、エクステントのバージョン情報を調査します。これが提供されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、特定のエクステントについてのバージョン情報を、指定されたネームスペースで調査します。この呼び出しの前に、アプリケーションによって明示的にバージョン情報が設定されていなければ、システムのデフォルト値 (現在は 4) を使用します。

N は関数が返す値です。その他に現在のバージョン (最新またはマスタ・バージョン) が 1 つと N-1 バックアップ・バージョンがあります。

注釈: バックアップとして利用できるバージョン番号は、連番の数列である必要はありません。中間のバージョンは、DEL<sup>®</sup>R への呼び出しにより削除されている可能性もあります。

## 例

以下の例は、バージョン管理に関する多くのエントリ・ポイントの使用法を示します。

**警告!** この例は、AnExample.INT という SAMPLES ネームスペース内にルーチンを作成します。その名前のルーチンがそのネームスペースにすでにある場合は、オーバーライドします。

```

; Change the namespace we will use
Znspace "SAMPLES"

; fill in the source code array
Set N = 0
Set code($INCREMENT(N)) = "AnExample ; An example routine"
Set code($INCREMENT(N)) = "    Write "Starting AnExample", !"
Set code($INCREMENT(N)) = "    Write "Line to be replaced", !"
Set lininx = N
Set code($INCREMENT(N)) = "    Write "Finished AnExample", !"
Set code($INCREMENT(N)) = "    Quit"
Set code(0) = N

Set name = "AnExample"
Set ext = "MAC"
Set routine = name _ "." _ ext
Set options = "BCS" ; Backup, Compile and Save
Set errors = "" ; empty list
Set abort = 0

; get the number of backups
Set maxvers = $$VERMAX^%R(ext)
Write "Default versions: ", maxvers, !
; increase it by one
Do VERMAXSET^%R(ext, (maxvers + 1))
; confirm it
set maxvers = $$VERMAX^%R(ext)
Write "Explicit versions: ", maxvers, !

; process the base version
Set return = $$ROUTINE^%R(routine, .code, .errors, options)
; show the simple result
Write "Compilation result: ", return, !
If (+return = "")
{
    ; format and display the errors
    Write $$FMterr^%R(.errors, .code), !
    set abort = 1
}
Else
{
    ; run it
    Write "Calling AnExample", !
    Set entrypoint = name _ "^" _ name
    Do @entrypoint
}
Quit:(abort)

; generate a bunch of versions
for i = 1 : 1 : 10
{
    ; process the new version
    Write !, "Loop ", i, !
    Set code(lininx) = "    Write "Iteration " _ i _ "", !"
    Set return = $$ROUTINE^%R(routine, .code, .errors, options)
    ; show the simple result
    Write "Compilation result: ", return, !
    If (+return = "")
    {
        ; format and display the errors
        Write $$FMterr^%R(.errors, .code), !
        set abort = 1
    }
    Else
    {
        ; run it
        Write "Calling AnExample", !
        Set entrypoint = name _ "^" _ name
    }
}

```

```

    Do @entrypoint
  }

  ; get the version boundaries
  Set oldest = $$VERSION1^%R(routine)
  Set vername = routine _ "." _ "-1"
  Set youngest = $$VERSION^%R(vername)
  Write "Oldest: ", oldest, "; ", "Youngest: ", youngest, !
}
; remove the source and object
for suffix = "int", "OBJ"
{
  Set component = name _ "." _ suffix
  Write "Removing ", component, ": ", $$DEL^%R(component), !
}
; reset the maximum versions
Kill ^rBACKUP(0,ext)
Quit

```

## VERMAXSET^%R

VERMAX^%R (extent, max, namesp)

### パラメータ

extent	希望するバージョン情報を持つエクステント
max	保存されるバージョンの最大数
namesp	このエクステントが保存されているネームスペース

### 概要

この関数は、ルーチンのバージョン番号を、このエクステントと一緒に設定します。このエクステントは、指定されたネームスペースで保持されます。

### パラメータ

#### extent

文字列の内容は、希望するバージョン情報に対するエクステントを指定します。

#### max

正のゼロ以外の整数に評価する式。バージョンの番号は、バックアップとして保持されます。

#### namesp

文字列の内容は、ネームスペースを指定して、エクステントのバージョン情報を調査します。これが提供されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、指定されたネームスペースの特定のエクステントについての、バージョン情報を設定します。max を 1 に設定すると、最新のバージョンのみが保存されます。

新規の最大数が設定されると 1 を返し、そうでない場合は 0 を返します。

## 例

VERMAX^%R エントリ・ポイントと提供されている例を参照してください。

# VERSION1^%R

---

```
VERSION1^%R ( rtn, namesp )
```

## パラメータ

rtn	ルーチン名を与えて文字列に評価される式
namesp	このエクステントが保存されているネームスペース

## 概要

これは、バックアップとして保持されているもののうち、最も古い番号を返します。

## パラメータ

### rtn

ルーチンの名前を提供する文字列。アプリケーションは、このルーチンの最も古いバージョンに動作します。これは、名前の一部としてエクステントを含みます。この名前は、大文字と小文字を区別します。エクステントは区別しません。rtn の一部として提供される任意のバージョン番号は、無視されます。

### namesp

有効な文字列として評価する式。文字列の内容は、ネームスペースを指定して、エクステントのバージョン情報を調査します。これが提供されていない場合は、現在のネームスペースであるとみなします。

## 備考

この関数は、指定されたネームスペースを調査して、バックアップされた特定のルーチンのバージョンを検索します。最も古いルーチンのバージョン番号が返されます。

## 例

VERMAX^%R エントリ・ポイントと提供されている例を参照してください。

# VERSION^%R

VERSION^%R (rtn, namesp)

## パラメータ

rtn	ルーチン名を与えて文字列に評価される式
namesp	このエクステントが保存されているネームスペース

## 概要

これは、最新のバージョンに関連するバージョン番号を返します。

## パラメータ

### rtn

ルーチンの名前を指定する文字列。これは、名前の一部としてエクステントを含みます。この名前は、大文字と小文字を区別します。エクステントは区別しません。バージョン番号は、符号付きの値で表し、最新のバックアップとの関係を示します。例えば、“SomeRoutine.MAC.-1” は、最新のものの 1 つ前のルーチンのバージョン番号を要求します。

### namesp

有効な文字列として評価する式。文字列の内容は、ネームスペースを指定して、エクステントのバージョン情報を調査します。これが提供されていない場合は、現在のネームスペースであるとみなします。

## 備考

名前のバージョン部分が関係ない場合は、指定されたバージョン番号を返します。関係がある場合は、最新バージョンに関連するバックアップのバージョン番号を返します。

指定されたバージョンが存在しない場合は、このルーチンは 0 を返します。

## 例

VERMAX^%R エントリ・ポイントと提供されている例を参照してください。

