コールイン API の使用法

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

コールイン API の使用法
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

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 Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

テーブル一覧

このドキュメントについて

このドキュメントでは、InterSystems のコールイン API の使用方法を説明します。これは、C プログラムまたは C++ プログラムから InterSystems IRIS® コマンドの実行と ObjectScript 式の評価ができるようにするためのインタフェースを提供します。

このドキュメントを使用するには、使用するオペレーティング・システムに実用的なレベルで精通し、C、C++、またはオペレーティング・システムの C/C++ の呼び出し標準を使用できるその他の言語について十分な経験があることが必要です。

このドキュメントの構成は、以下のとおりです。

- “コールイン・インタフェース”の章では、C プログラムで InterSystems IRIS コマンドの実行および ObjectScript 式の評価に使用できるコールイン・インタフェースについて説明します。
- “コールイン関数の使用法”の章では、コールイン関数の概要を、その関数で実行するタスク種別に説明します(各関数の詳しい説明については該当のリンクを参照してください)。
- “コールイン関数リファレンス”の章では、すべての InterSystems コールイン関数をアルファベット順に挙げて詳細に説明します。

コールイン関数は、非常に低レベルのプログラミング・インタフェースを提供します。多くの場合、InterSystems IRIS 標準のいずれかの接続オプションを使用することで、はるかに容易に目的を達成できます。詳細は、以下の情報を参照してください。

- ″InterSystems ソフトウェアでの Java の使用法″ の “InterSystems Java 接続オプション”
- InterSystems Managed Provider for .NET の使用法

InterSystems コールアウト・ゲートウェイは、InterSystems IRIS から呼び出すことができる関数を備えた共有ライブラリを作成できるようにするプログラミング・インタフェースです。コールアウト・コードは、通常、C または C++ で記述されますが、C/C++ 呼び出し規則をサポートする任意の言語でも記述できます。

- コールアウト・ゲートウェイの使用法

1

コールイン・インタフェース

InterSystems IRIS® は、C プログラムを使用して InterSystems IRIS コマンドの実行および ObjectScript 式の評価ができるコールイン・インタフェースを提供しています。

コールイン・インタフェースは、広範囲のアプリケーションで使用できます。例えば、コールイン・インタフェースを使用して、総合メニューや GUI から ObjectScript を使用可能にできます。ATM や研究機器など外部デバイスから情報を収集する場合、コールイン・インタフェースを使用して InterSystems IRIS データベースにデータを格納できます。現在 InterSystems IRIS は C と C++ プログラムのみをサポートしますが、プラットフォームの呼び出し標準を使用するあらゆる言語で、コールイン関数を呼び出すことができます。

コールイン関数の概要は、"コールイン関数の使用法"を参照してください。各コールイン関数の詳細な参考資料は、"コールイン関数リファレンス" を参照してください。

1.1 セットアップ

コールイン開発環境には、以下のオプションが含まれている必要があります。

- インストール・パッケージ

  システムに [ ] インストール・オプションによってインストールされたコンポーネントが含まれている必要があります。InterSystems IRIS の既存のインスタンスは、インストーラを再度実行することで更新できます。

  - Windows では、インストール中に [ ]: [ ] オプションを選択します。
  - UNIX® および関連オペレーティング・システムでは、インストール中に `1) Development - Install InterSystems IRIS server and all language bindings` オプションを選択します ("インストール・ガイド" の "UNIX® および Linux" のセクションにある "InterSystems IRIS の標準インストール手順" を参照してください)。

- `%Service_CallIn` サービス

  InterSystems IRIS をセキュリティ・オプション 2 (通常) でインストールした場合、管理ポータルを開いて、System Administration > Security > Services に移動し、`%Service_CallIn` を選択して、[ ] ボックスにチェックが付いていることを確認します。

  InterSystems IRIS をセキュリティ・オプション 1 (最小) でインストールした場合、これには既にチェックが付いているはずです。

1.2 iris-callin.h ヘッダ・ファイル

iris-callin.h ヘッダ・ファイルは、これらの関数のプロトタイプを定義します。これによりC コンパイラは、これらの関数がプログラム内で呼び出されたときにパラメータ・データ型が有効であるかどうかをテストできます。このファイルを、C プログラムの #include 文のリストに追加できます。

```
#include "iris-callin.h"
```

iris-callin.h ファイルには、呼び出しで使用するパラメータ値の定義が含まれ、有効に活用可能な各種の #define が組み込まれています。このような定義には、オペレーティング・システム固有の値、エラー・コード、および InterSystems IRIS がどのように動作するかを決定する値があります。

分散ヘッダ・ファイル iris-callin.h を変換できます。ただし、iris-callin.h は変更されることがあります。変換されたバージョン・ファイルを生成する場合、あらゆる変更を追跡する必要があります。インターシステムズのサポート窓口は、サポートされていない言語に関するお問い合わせにはお答えできません。

返り値とエラー・コード

ほとんどのコールイン関数は、int 型の値を返します。返り値は 16 ビット整数の容量を超えません。この返り値は、IRIS_SUCCESS、InterSystems IRIS エラー、またはコールイン・インタフェース・エラーです。

エラーには、2 種類あります。

- InterSystems IRIS エラー ー InterSystems IRIS エラーの返り値は正の整数です。
- インタフェース・エラー ー インタフェース・エラーの返り値は、0 あるいは負の整数です。

iris-callin.h は、すべてのシステム・エラーとインタフェース・エラーで使用する、IRIS_SUCCESS (0) や IRIS_FAILURE (-1) などの記号を定義します。コールイン関数 IrisErrxlate を呼び出すことにより、InterSystems IRIS エラー (正の整数) を変換できます。

1.3 8 ビットおよび Unicode 文字列の処理

文字列を操作する InterSystems コールイン関数には、8 ビット・バージョンと Unicode バージョンの両方があります。これらの関数は、処理する文字列のタイプを示すために接尾語を使用します。

- 接尾語 “A” の付いた名前、またはまったく接尾語の付かない名前 (例えば、IrisEvalA または IrisPopExStr など) は、8 ビット文字列のバージョンです。
- 接尾語 W の付いた名前 (例えば、IrisEvalW または IrisPopExStrW など) は、2 バイトの Unicode 文字を使用するプラットフォーム上の Unicode 文字列用のバージョンです。
- 接尾語 H の付いた名前 (例えば、IrisEvalH または IrisPopExStrH など) は、4 バイトの Unicode 文字を使用するプラットフォーム上の Unicode 文字列用のバージョンです。

最適なパフォーマンスのために、インストールした InterSystems IRIS のバージョン固有の文字列を使用してください。

1.3.1 8 ビットの文字列データ型

InterSystems IRIS は、ローカルの 8 ビット文字列エンコーディングを使用する以下のデータ型をサポートします。

- IRIS_ASTR ー 8 ビット文字の計算文字列
- IRIS_ASTRP ー 8 ビット計算文字列へのポインタ

タイプ定義は以下のとおりです。

```
#define IRIS_MAXSTRLEN 32767
typedef struct {
   unsigned short  len;
   Callin_char_t   str[IRIS_MAXSTRLEN];
} IRIS_ASTR, *IRIS_ASTRP;
```

IRIS_ASTR 構造体と IRIS_ASTRP 構造体には以下の 2 つの要素が含まれています。

- len ― 整数。入力として使われる場合、この要素は、値が str 要素で供給されている文字列の実際の長さを指定します。出力として使われる場合、この要素は、str 要素に対して許容される最大の長さを指定します。値が返され次第、str の実際の長さと置換されます。
- str ― 入力または出力文字列。

IRIS_MAXSTRLEN は、受け取る、あるいは返される文字列の最大長です。パラメータ文字列は、IRIS_MAXSTRLEN の長さである必要はなく、プログラムにそれほど多くの領域を割り当てる必要もありません。

1.3.2 2 バイト Unicode データ型

InterSystems IRIS は、2 バイトの Unicode 文字を使用するプラットフォーム上で、以下の Unicode 関連データ型をサポートしています。

- IRISWSTR ― Unicode 計算文字列
- IRISWSTRP ― Unicode 計算文字列へのポインタ

タイプ定義は以下のとおりです。

```
typedef struct {
   unsigned short len;
   unsigned short str[IRIS_MAXSTRLEN];
} IRISWSTR, *IRISWSTRP;
```

IRISWSTR 構造体と IRISWSTRP 構造体には以下の 2 つの要素が含まれています。

- len ― 整数。入力として使われる場合、この要素は、値が str 要素で供給されている文字列の実際の長さを指定します。出力として使われる場合、この要素は、str 要素に対して許容される最大の長さを指定します。値が返され次第、str の実際の長さと置換されます。
- str ― 入力または出力文字列。

IRIS_MAXSTRLEN は、受け取る、あるいは返される文字列の最大長です。パラメータ文字列は、IRIS_MAXSTRLEN の長さである必要はなく、プログラムにそれほど多くの領域を割り当てる必要もありません。

InterSystems IRIS の Unicode 対応バージョンでは、データ型 IRIS_WSTRING もあります。これは、2 バイトのプラットフォーム上でネイティブな文字列型を表します。IrisType はこの型を返します。さらに、IrisConvert では、返り値のデータ型として IRIS_WSTRING を指定できます。この型が要求された場合、結果は IRISWSTR バッファ内の計算 Unicode 文字列として渡されます。

1.3.3 4 バイト Unicode データ型

InterSystems IRIS は、4 バイトの Unicode 文字を使用するプラットフォーム上で、以下の Unicode 関連データ型をサポートしています。

- IRISHSTR ― 拡張 Unicode 計算文字列
- IRISHSTRP ― 拡張 Unicode 計算文字列へのポインタ

タイプ定義は以下のとおりです。

```
typedef struct {
   unsigned int len;
   wchar_t str[IRIS_MAXSTRLEN];
} IRISHSTR, *IRISHSTRP;
```

IRISHSTR 構造体と IRISHSTRP 構造体には以下の 2 つの要素が含まれています。

- len ― 整数。入力として使われる場合、この要素は、値が str 要素で供給されている文字列の実際の長さを指定します。出力として使われる場合、この要素は、str 要素に対して許容される最大の長さを指定します。値が返され次第、str の実際の長さと置換されます。
- str ― 入力または出力文字列。

IRIS_MAXSTRLEN は、受け取る、あるいは返される文字列の最大長です。パラメータ文字列は、IRIS_MAXSTRLEN の長さである必要はなく、プログラムにそれほど多くの領域を割り当てる必要もありません。

InterSystems IRIS の Unicode 対応バージョンでは、データ型 IRIS_HSTRING もあります。これは、4 バイトのプラットフォーム上でネイティブな文字列型を表します。IrisType はこの型を返します。さらに、IrisConvert では、返り値のデータ型として IRIS_HSTRING を指定できます。この型が要求された場合、結果は IRISHSTR バッファ内の計算 Unicode 文字列として渡されます。

Unicode 対応の InterSystems IRIS では、2 バイトの文字のみを使用するため、これらの文字列は InterSystems IRIS での受信時に UTF-16 に変換され、InterSystems IRIS からの送信時に UTF-16 から 4 バイトの Unicode に変換されます。$W 関数ファミリ (例えば、$WASCII() や $WCHAR()) を InterSystems IRIS コードで使用して、これらの文字列を処理できます。

1.3.4 システム中立の記号定義

一部の関数では、8 ビット・システムまたは Unicode システムで実行されるかどうかによって、許容される入力と出力が異なります。“A” (ASCII) 関数の多くでは、IRISSTR、IRIS_STR、IRISSTRP、または IRIS_STRP の各型を受け入れるものとして引数が定義されています。“A”、“W”、および “H” のいずれも使用しない記号の定義は、記号 IRIS_UNICODE および IRIS_WCHART がコンパイル時に定義されるかどうかによって、8 ビット名または Unicode 名のいずれかと条件付きで関連付けることができます。この方法により、ローカルの 8 ビット・エンコーディングあるいは Unicode エンコーディングのいずれかで動作する中立なシンボルを持つソース・コードを記述できます。

以下の iris-callin.h からの引用は概念を示しています。

```
#if defined(IRIS_UNICODE) /* Unicode character strings */
#define   IRISSTR      IRISWSTR
#define   IRIS_STR     IRISWSTR
#define   IRISSTRP     IRISWSTRP
#define   IRIS_STRP    IRISWSTRP
#define   IRIS_STRING  IRIS_WSTRING

#elif defined(IRIS_WCHART)  /* wchar_t character strings */
#define   IRISSTR      IRISHSTR
#define   IRIS_STR     IRISHSTR
#define   IRISSTRP     IRISHSTRP
#define   IRIS_STRP    IRISHSTRP
#define   IRIS_STRING  IRIS_HSTRING

#else                  /* 8-bit character strings */
#define   IRISSTR      IRIS_ASTR
#define   IRIS_STR     IRIS_ASTR
#define   IRISSTRP     IRIS_ASTRP
#define   IRIS_STRP    IRIS_ASTRP
#define   IRIS_STRING  IRIS_ASTRING
#endif
```

1.4 InterSystems セキュリティ関数の使用法

InterSystems IRIS パスワードを使用する作業のために、2 つの関数が提供されています。

- IrisSecureStart – IrisStart に類似していますが、パスワード認証のために追加のパラメータが提供されます。IrisStart 関数は、今では非推奨となっています。使用すると、Username、Password、および ExeName に NULL を指定して IrisSecureStart を呼び出したかのように動作します。何らかの形式のパスワード認証を使用する必要がある場合、IrisStart は使用できません。
- IrisChangePassword – この関数は、ユーザが InterSystems 認証を使用している場合にユーザのパスワードを変更します (LDAP/DELEGATED/Kerberos などに対しては有効ではありません)。コールイン・セッションが初期化される前に呼び出される必要があります。

ASCII "A"、Unicode "W"、および Unicode "H" の各インストール用に、IrisSecureStart 関数と IrisChangePassword 関数があります。これらの新しい関数では、渡されたパラメータをナロー化、ワイド化、またはそのまま使用し、それらを新規コールイン・データ領域に保管した上で、最終的に IrisStart エントリ・ポイントを呼び出します。

IrisStart と IrisSecureStart の pin パラメータおよび pout パラメータは、NULL として渡すことができます。NULL は、プラットフォーム既定の入力および出力デバイスを使用することを意味します。

1.5 マルチスレッドでのコールインの使用

InterSystems IRIS は、Windows および UNIX® の一部のバージョンで実行されているスレッド・プログラムでコールインが使用できるように機能拡張されています (一覧については、このリリース用のオンライン・ドキュメント "インターシステムズのサポート対象プラットフォーム" の "その他のサポート対象機能" を参照してください)。スレッド化されたアプリケーションは、libirisdbt.so または irisdbt.lib に対してリンクする必要があります。

1 つのプログラムで複数のスレッド (UNIX® 環境では pthreads) を生成でき、各スレッドでは IrisSecureStart を呼び出すことで InterSystems IRIS との独立した接続を確立できます。スレッドは、InterSystems IRIS に対する単一の接続を共有できません。InterSystems IRIS を使用する必要があるスレッドは、それぞれが IrisSecureStart を呼び出さなければなりません。スレッドがコールイン関数の使用を試みるときに、IrisSecureStart を呼び出していない場合は、`IRIS_NOCON` エラーが返されます。

ログインの一部としての認証情報を指定するために IrisSecureStart が使用されている場合、認証情報はスレッド間で共有されていないので、各スレッドは IrisSecureStart を呼び出して、接続のために適正なユーザ名/パスワードを提供しなければなりません。C コンパイラは (非スレッド化ビルドで直接メモリ参照を使用する) スレッド・ローカル・ストレージにアクセスするために追加のコードを生成する必要があるため、InterSystems IRIS でスレッドを使用すると、パフォーマンスが劣化します。

1.5.1 スレッドおよび UNIX® のシグナル処理

UNIX® では、InterSystems IRIS は多くのシグナルを使用します。InterSystems IRIS で使用するシグナルと同じシグナルをアプリケーションでも使用する場合は、InterSystems IRIS でこれらがどのように処理されるのかを認識しておく必要があります。すべてのシグナルには、OS で指定された既定のアクションがあります。アプリケーションでは、既定のアクションをそのまま実行できるほか、シグナルを処理または無視することもできます。シグナルを処理する場合は、そのシグナルをブロックするスレッドと受信するスレッドをアプリケーションで選択できます。シグナルの中には、ブロック、無視、処理のいずれも不可能なものがあります。多くのシグナルの既定のアクションはプロセスの停止なので、この既定のアクションをそのままにしておくことはできません。以下のシグナルは捕捉も無視もできず、これらのシグナルが発生すればプロセスが終了します。

シグナル	処理
SIGKILL	直ちにプロセスを終了します。
SIGSTOP	後で再開できるようにプロセスを停止します。

アプリケーションでシグナルごとに確立するアクションはプロセス全体で有効です。シグナルを各スレッドに送信できるかどうかは、スレッドに固有です。各スレッドでは、他のスレッドとは無関係にシグナルの処理方法を指定できます。あるスレッドではすべてのシグナルをブロックし、別のスレッドではすべてのシグナルを受信できるといった指定が可能です。シグナルをスレッドに送信したときの動作は、そのシグナルに対してプロセス規模で設定した処理によって決まります。

1.5.1.1 シグナル処理

InterSystems IRIS は、アプリケーション・ハンドラとシグナル・マスクを保存し、適切な時点でこれらをリストアすることによって、シグナル処理をアプリケーションによる処理と統合しています。シグナルは、以下の方法で処理されます。

生成されたシグナル

InterSystems IRIS では、生成されるすべてのシグナルで使用するために独自のシグナル・ハンドラをインストールします。また、現在のアプリケーションのシグナル・ハンドラを保存します。生成されたシグナルをスレッドで捕捉すると、シグナル・ハンドラはそのスレッドを InterSystems IRIS から切断し、アプリケーションのシグナル処理関数が存在する場合はそれを呼び出して、`pthread_exit` を実行します。

シグナル・ハンドラはプロセス全体で有効なので、InterSystems IRIS に接続されていないスレッドもシグナル・ハンドラに記録されます。接続されていないスレッドを検出すると、InterSystems IRIS はアプリケーションのハンドラを呼び出し、`pthread_exit` を実行します。

同期シグナル

InterSystems IRIS は、すべての同期シグナルに対してシグナル・ハンドラを確立し、スレッドが InterSystems IRIS に接続されるたびにそのスレッドでのこれらのシグナルに対するブロックを解除します (詳細は “同期シグナル” を参照してください)。

非同期シグナル

InterSystems IRIS は、プロセスを終了するすべての非同期シグナルを処理します (詳細は “非同期シグナル” を参照してください)。

保存/リストア・ハンドラ

最初のスレッドがシステムに接続したとき、システムはそのシグナルの状態を保存します。最後のスレッドを切断するときには、処理したシグナルごとにそのシグナルの状態をリストアします。

スレッド・シグナル・マスクの保存/リストア

スレッド・シグナル・マスクは、接続時に保存され、スレッドの切断時にリストアされます。

1.5.1.2 同期シグナル

同期シグナル (`SIGSEGV` など) は、アプリケーション自体で生成されます。InterSystems IRIS は、すべての同期シグナルに対してシグナル・ハンドラを確立し、スレッドが InterSystems IRIS に接続されるたびにそのスレッドでのこれらのシグナルに対するブロックを解除します。

同期シグナルは、そのシグナルを生成したスレッドにより捕捉されます。アプリケーションで生成したシグナル (`SIGSEGV` など) のハンドラをそのアプリケーションで指定していない場合、またはスレッドでシグナルをブロックしている場合、OS はプロセス全体を停止します。シグナル・ハンドラに記録されたスレッドは、他のスレッドに影響を与えることなく、

(pthread_exit により) 正常に終了できます。スレッドがハンドラから戻ろうとすると、OS はプロセス全体を停止します。以下のシグナルが発生するとスレッドが終了します。

シグナル	処理
SIGABRT	中止シグナルを処理します。
SIGBUS	バスのエラー
SIGEMT	EMT 命令
SIGFPE	浮動少数点の例外
SIGILL	不正な命令
SIGSEGV	アクセス違反
SIGSYS	システム呼び出しへの不正な引数
SIGTRAP	トレース・トラップ
SIGXCPU	CPU の時間制限を超えています (setrlimit)。

1.5.1.3 非同期シグナル

非同期シグナル (SIGALRM、SIGINT、SIGTERM など) はアプリケーション外部で生成されます。InterSystems IRIS は、プロセスを終了するすべての非同期シグナルを処理します。

非同期シグナルは、そのシグナルをブロックしていないスレッドであれば捕捉できます。使用するスレッドはシステム側で選択します。プロセスの終了を既定のアクションとするあらゆるシグナルは、それを受信できる 1 つ以上のスレッドで処理する必要があります。処理しない場合は、明確に無視する必要があります。

アプリケーションでは、処理するシグナルのシグナル・ハンドラを確立し、これらのシグナルをブロックしないスレッドを開始する必要があります。そのスレッドは、そのシグナルを受信して処理できる唯一のスレッドとなります。アプリケーションから IrisStart を最初に呼び出す前に、ハンドラとシグナル処理可能なスレッドの両方が存在している必要があります。IrisStart への最初の呼び出しで、プロセスを終了するすべての非同期シグナルに対して以下のアクションが実行されます。

- InterSystems IRIS は、これらのシグナルのハンドラを探します。ハンドラが見つかると、InterSystems IRIS はそれをそのままにします。見つからない場合、InterSystems IRIS はシグナルを SIG_IGN (シグナルを無視) に設定します。
- InterSystems IRIS は、シグナルにハンドラがあるかどうかに関係なく、接続しているスレッドでこれらのシグナルをすべてブロックします。したがって、シグナルにハンドラがある場合は、InterSystems IRIS に接続していないスレッドのみがそのシグナルを捕捉できます。

以下のシグナルはこのプロセスの影響を受けます。

シグナル	処理
SIGALRM	タイマ
SIGCHLD	スレッドでブロックされます。
SIGDANGER	処理されない場合は無視します。
SIGHUP	処理されない場合は無視します。
SIGINT	処理されない場合は無視します。
SIGPIPE	処理されない場合は無視します。

シグナル	処理
SIGQUIT	処理されない場合は無視します。
SIGTERM	SIGTERM が処理されない場合は InterSystems IRIS で処理します。SIGTERM シグナルを受信した InterSystems IRIS ハンドラはすべてのスレッドを切断し、新しい接続を許可しません。SIGTERM のハンドラはスタックされません。
SIGUSR1	プロセス間通信
SIGUSR2	プロセス間通信
SIGVTALRM	仮想タイマ
SIGXFSZ	InterSystems IRIS 非同期スレッド停止

1.6 コールイン・プログラミングのヒント

このセクションは、以下のトピックで構成されています。

- すべてのコールイン・プログラム用のヒント
- Windows 用のヒント
- UNIX® および Linux 用のヒント

1.6.1 すべてのコールイン・プログラム用のヒント

外部プログラムは、InterSystems IRIS のデータ構造を破損しないように、特定の規則に従う必要があります。データ構造が破損すると、システムが停止する可能性があります。

- 開いているファイル数の制限

  InterSystems IRIS で必要なデータベースや他のファイルを開くことができるように、プログラムで大量のファイルを開かないように注意する必要があります。通常、InterSystems IRIS は使用中の開いているファイルの割り当てを検索し、データベース用に一定のファイル数を確保し、それ以外を Open コマンドに割り当てます。割り当てによって異なりますが、InterSystems IRIS には、同時に開く InterSystems IRIS データベース・ファイルが 6 ～ 30 個、また、Open コマンドで開くファイルが 0 ～ 36 個あると考えられます。

- コールイン・アプリケーションのディレクトリ名の最大文字列長

  コールイン・アプリケーションを持つディレクトリには、232 文字未満のフル・パスが必要です。 例えば、アプリケーションが `C:\IrisApps\Accounting\AccountsPayable\` ディレクトリにある場合、このディレクトリの文字数は 40 文字であるため有効です。

- 停止する前に、IrisStart の後に IrisEnd を呼び出す

  IrisStart の呼び出しにより接続が確立した場合、接続完了後、IrisEnd を呼び出す必要があります。必要な数だけコールイン関数を呼び出すことが可能です。

  接続が切断されている場合でも、IrisEnd を呼び出す必要があります。接続は、RESJOB パラメータを持つ IrisAbort を呼び出すことで切断できます。

  IrisEnd は、別の IrisStart 呼び出しの準備に必要なクリーンアップ処理を実行します。(接続が切断された場合) IrisEnd を呼び出さずに再度 IrisStart を呼び出すと、IRIS_CONBROKEN コードが返されます。

- 終了する前に、ObjectScript の実行が完了するまで待機する

プログラムを終了する場合、ObjectScript があらゆる未処理の要求を完了していることを確認する必要があります。コールイン関数 IrisContext を使用して、ObjectScript 内にいるかどうかを判断します。この呼び出しは、終了ハンドラと Ctrl-C、あるいは Ctrl-Y ハンドラで特に重要です。IrisContext がゼロ以外の値を返す場合、IrisAbort を呼び出すことができます。

- コールイン・セッションでのマージンの維持

  コールイン・セッション内ではマージンの設定が可能ですが、そのマージン設定は現在のコマンド行の残りに対してのみ保持されます。プログラム (ダイレクト・モードと同様) に、以下の行が含まれる場合、

  ```
  :Use 0:10 Write x
  ```

  コマンド行の有効期間に対してマージン 10 が確立されます。

  一部の呼び出しはコマンド行に影響を与え、それに伴ってマージンも影響を受けます。これらは、関数の説明で "InterSystems IRIS にコールインする" という注釈が付いている呼び出しです。

- IrisStart() を使用する場合はシグナル処理を実行しない

  IrisStart によってさまざまなシグナルにハンドラが設定されるので、呼び出し元のアプリケーションで設定したシグナル・ハンドラとこれらのハンドラが競合する可能性があります。

1.6.2 Windows 用のヒント

これらのヒントは、Windows にのみ適用されます。

- iris 共有ライブラリ (irisdb.dll) を使用してコールイン・アプリケーションを構築する際の制約

  共有ライブラリ (irisdb.dll) を使用してコールイン・アプリケーションを構築する場合、グローバル・バッファ・プールが大きいと、コールインの初期化 (IrisStart 内) が失敗して下記のエラーが発生する可能性があります。

  ```
  <InterSystems IRIS Startup Error: Mapping shared memory (203)>
  ```

  この原因は、Windows でロードするシステム DLL の動作にあります。Win 32 API または Microsoft Foundation Class (Microsoft Visual C++ 開発をサポートする主要ライブラリ) を使用してコード作成したアプリケーションの場合、初期化した後、直ちにその Windows コード用の DLL を OS でロードする必要があります。これらの DLL は仮想ストレージの先頭 (上位アドレス) からロードされるので、ヒープとして利用できる領域が減少します。ほとんどのシステムでは、他の DLL (例えば、表示グラフィックをサポートする DLL など) も多数あり、Windows プロセスごとに、仮想ストレージのかなり上位のアドレスに自動的にロードされます。これらの DLL は、主に 0X10000000 (256 MB) などの特定のアドレス領域を要求する傾向があるので、仮想ストレージの下位アドレスにある数百メガバイトの連続メモリが分断されます。その結果、コールイン実行可能プログラムでは、InterSystems IRIS 共有メモリ・セグメントをマッピングする仮想メモリ領域が不足します。

1.6.3 UNIX® および Linux 用のヒント

これらのヒントは、UNIX® および Linux にのみ適用されます。

- UNIX® 上では割り込みを無効にしない

  UNIX® は割り込みを使用します。割り込みの受け渡しを妨げないでください。

- 予約済みのシグナルの使用を避ける

  UNIX® では、InterSystems IRIS は多くのシグナルを使用します。可能であれば、InterSystems IRIS にリンクしたアプリケーション・プログラムでは、以下の予約済みシグナルの使用を避けてください。

SIGABRT	SIGDANGER	SIGILL	SIGQUIT	SIGTERM	SIGVTALRM
SIGALRM	SIGEMT	SIGINT	SIGSTOP	SIGTRAP	SIGXCPU
SIGBUS	SIGFPE	SIGKILL	SIGSEGV	SIGUSR1	SIGXFSZ
SIGCHLD	SIGHUP	SIGPIPE	SIGSYS	SIGUSR2	

アプリケーションでこれらのシグナルを使用する場合は、InterSystems IRIS でこれらがどのように処理されるのかを認識しておく必要があります。詳細は、″スレッドおよび UNIX® のシグナル処理″ を参照してください。

1.6.3.1 UNIX® 上でのコールイン実行可能プログラムの許可の設定

InterSystems IRIS 実行可能プログラム、ファイル、および共有メモリやオペレーティング・システム・メッセージなどのリソースの所有者は、インストール時に選択されたユーザ (インストール所有者)、および既定名 irisusr (インストール時に別の名前を選択できます) を持つグループです。これらのファイルやリソースには、このユーザ ID を持つプロセス、またはこのグループに属するプロセスのみアクセスできます。それ以外の場合、InterSystems IRIS に接続しようとすると、接続の確立前に、オペレーティング・システムから保護エラー (通常、アクセス拒否) が表示されます。

コールイン・プログラムは、実効グループ ID が irisusr である場合にのみ実行できます。この条件を満たすには、以下の条件のうち 1 つを満たしている必要があります。

- プログラムは irisusr グループ (または、irisusr からその他へ変更された場合は代替実行グループ) に属するユーザにより実行される。
- プログラムでは、(UNIX® の chgrp コマンドと chmod コマンドを使用して) uid または gid のファイル・アクセス権を操作することにより、有効なユーザまたはグループを設定している。

2
コールイン関数の使用法

ここでは、コールイン関数の概要を説明します。各関数の詳しい説明については、それぞれのリンクを参照してください。ここで説明するカテゴリは以下のとおりです。

- プロセス・コントロール

  これらの関数は、コールイン・セッションを開始し、停止します。また、このセッションに関連する各種設定を制御します。

- 関数とルーチン

  これらの関数は、関数呼び出し、またはルーチン呼び出しを実行します。スタック関数は、関数参照またはルーチン参照をプッシュするために用意されています。

- トランザクションとロック

  これらの関数は、標準の InterSystems IRIS® トランザクション・コマンド (TSTART、TCOMMIT、および TROLLBACK) と LOCK コマンドを実行します。

- オブジェクトの管理

  これらの関数は、Oref カウンタを操作し、メソッド呼び出しを実行して、プロパティの取得、または設定を行います。スタック関数は、Oref、メソッド参照、プロパティ名に対しても含まれています。

- グローバルの管理

  これらの関数は、InterSystems IRIS で呼び出され、グローバルを操作します。関数は、グローバルを引数スタックにプッシュするために用意されています。

- 文字列の管理

  これらの関数は文字列をある形式から別の形式に変換します。また、文字列引数のプッシュ、またはポップを行います。

- 単純なデータ型の管理

  これらのスタック関数は、int、double、$list、または pointer 値を持つ引数のプッシュとポップに使用されます。

この後のセクションでは、個々の関数について詳しく説明します。

2.1 プロセス・コントロール

これらの関数は、コールイン・セッションの開始と停止、セッションに関連する各種設定の制御、ObjectScript コマンドや式を実行するための高レベルのインタフェースを提供します。

2.1.1 セッション制御

これらの関数は、コールイン・セッションを開始し、停止します。また、このセッションに関連する各種設定を制御します。

テーブル 2-1: セッション制御関数

IrisAbort	InterSystems IRIS に現在の要求を終了するように命令します。
IrisChangePasswordA[W][H]	InterSystems 認証が使用されている場合、ユーザのパスワードを変更します。コールイン・セッションが初期化される前に呼び出される必要があります。
IrisContext	$ZF コールバック・セッションにあるか、コールイン呼び出しの InterSystems IRIS 側にあるか、またはユーザ・プログラム側にあるかを示す整数を返します。
IrisCtrl	InterSystems IRIS が CTRL-C を無視するかどうかを決定します。
IrisEnd	InterSystems IRIS セッションを終了し、必要であれば、切断された接続を削除します (InterSystems IRIS 内での呼び出し)。
IrisEndAll	すべてのコールイン・スレッドを切断し、そのスレッドが終了するまで待機します。
IrisOflush	保留中の出力をすべてフラッシュします。
IrisPromptA[W][H]	ターミナルである文字列を返します。
IrisSetDir	実行時に、マネージャのディレクトリ (IrisSys¥Mgr) の名前を動的に設定します。Windows では、InterSystems IRIS の共有ライブラリ・バージョンでこの関数が必要です。
IrisSignal	ユーザ・プログラムによって検出されるシグナルを処理するために InterSystems IRIS に報告します。
IrisSecureStartA[W][H]	InterSystems IRIS プロセスを開始します。
IrisStartA[W][H]	(お勧めしません。代わりに、IrisSecureStart を使用してください) InterSystems IRIS プロセスを開始します。

2.1.2 ObjectScript の実行

これらの関数は、ObjectScript コマンドや式の実行に使用される高レベルなインタフェースを提供します。

テーブル 2-2: ObjectScript コマンド関数

IrisExecuteA[W][H]	ObjectScript コマンドを実行します (InterSystems IRIS 内での呼び出し)。
IrisEvalA[W][H]	ObjectScript 式を評価します (InterSystems IRIS 内での呼び出し)。
IrisConvert	IrisEval によって返された InterSystems IRIS 式の値を返します。
IrisType	IrisEval で返される項目のデータ型を返します。
IrisErrorA[W][H]	最新のエラー・メッセージ、関連するソース文字列、ソース文字列内でのエラーの発生箇所のオフセットを返します。
IrisErrxlateA[W][H]	コールイン関数から返されたエラー番号に関連する InterSystems IRIS エラー文字列を返します。

2.2 関数とルーチン

これらの関数は InterSystems IRIS で呼び出され、関数呼び出し、またはルーチン呼び出しを実行します。関数は、関数参照またはルーチン参照を引数スタックにプッシュするために用意されています。

テーブル 2-3: 関数呼び出しおよびルーチン呼び出しを実行する関数

IrisDoFun	ルーチン呼び出しを実行します (特別な場合) (InterSystems IRIS 内での呼び出し)。
IrisDoRtn	ルーチン呼び出しを実行します (InterSystems IRIS 内での呼び出し)。
IrisExtFun	外部関数呼び出しを実行します(InterSystems IRIS 内での呼び出し)。
IrisPop	引数スタックから値をポップ・オフします。
IrisUnPop	以下からスタック・エントリをリストアします。IrisPop
IrisPushFunc[W][H]	外部関数参照を引数スタックにプッシュします。
IrisPushFuncX[W][H]	拡張関数参照を引数スタックにプッシュします
IrisPushRtn[W][H]	ルーチン参照を引数スタックにプッシュします
IrisPushRtnX[W][H]	拡張ルーチン参照を引数スタックにプッシュします

2.3 トランザクションとロック

これらの関数は、標準の InterSystems IRIS トランザクション・コマンド (TSTART、TCOMMIT、および TROLLBACK) と LOCK コマンドを実行します。

2.3.1 トランザクション

以下の関数は、標準の InterSystems IRIS トランザクション・コマンドを実行します。

テーブル 2-4: トランザクション関数

IrisTCommit	TCommit コマンドを実行します。
IrisTLevel	トランザクション処理に対する、現在の入れ子レベル ($TLEVEL) を返します。
IrisTRollback	TRollback コマンドを実行します。
IrisTStart	TStart コマンドを実行します。

2.3.2 ロック

これらの関数は、さまざまな形式で InterSystems IRIS LOCK コマンドを実行します。関数は、IrisAcquireLock 関数で使用する目的で引数スタックにロック名をプッシュするために用意されています。

テーブル 2-5: ロック関数

IrisAcquireLock	LOCK コマンドを実行します。
IrisReleaseAllLocks	引数なしの InterSystems IRIS LOCK コマンドを実行して、プロセスによって現在保持されているロックをすべて削除します。
IrisReleaseLock	InterSystems IRIS LOCK – コマンドを実行して、指定したロック名のロック・カウントをデクリメントします。
IrisPushLock[W][H]	ロック名を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。
IrisPushLockX[W][H]	ロック名と環境文字列を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

2.4 オブジェクトの管理

これらの関数は、Oref カウンタを操作し、メソッド呼び出しを実行して、プロパティ値の取得、または設定を行うために、InterSystems IRIS 内で呼び出されます。スタック関数は、Oref、メソッド参照、プロパティ名に対しても含まれています。

2.4.1 Orefs

テーブル 2-6: Oref 関数

IrisCloseOref	OREF の参照カウンタをデクリメントします (InterSystems IRIS 内での呼び出し)。
IrisIncrementCountOref	OREF の参照カウンタをインクリメントします
IrisPopOref	引数スタックから OREF をポップ・オフします
IrisPushOref	OREF を引数スタックにプッシュします

2.4.2 メソッド

テーブル 2-7: メソッド関数

IrisInvokeMethod	インスタンス・メソッド呼び出しを実行します(InterSystems IRIS 内での呼び出し)。
IrisPushMethod[W][H]	インスタンス・メソッド参照を引数スタックにプッシュします
IrisInvokeClassMethod	メソッド呼び出しを実行します(InterSystems IRIS 内での呼び出し)。
IrisPushClassMethod[W][H]	クラス・メソッド参照を引数スタックにプッシュします

2.4.3 プロパティ

テーブル 2-8: プロパティ関数

IrisGetProperty	プロパティの値を取得します(InterSystems IRIS 内での呼び出し)。
IrisSetProperty	プロパティの値を格納します(InterSystems IRIS 内での呼び出し)。
IrisPushProperty[W][H]	プロパティ名を引数スタックにプッシュします

2.5 グローバルの管理

これらの関数は、InterSystems IRIS で呼び出され、グローバルを操作します。関数は、グローバルを引数スタックにプッシュするために用意されています。

テーブル 2-9: グローバルを管理するための関数

IrisGlobalGet	IrisPushGlobal[W][H] および任意の添え字で定義されたグローバル参照の値を取得します。このノード値は、引数スタックにプッシュされます。
IrisGlobalGetBinary	IrisGlobalGet のようにグローバル参照の値を取得します。また、その結果がバイナリ文字列であり、指定されたバッファに収まるかどうかを確認するためにテストを実行します。
IrisGlobalSet	グローバル参照の値を格納します。この呼び出しを行う前に、ノード値を引数スタックにプッシュしておく必要があります。
IrisGlobalData	指定したグローバルに対して $Data を実行します。
IrisGlobalIncrement	$Increment を実行し、スタックの一番上の結果を返します。
IrisGlobalKill	グローバル・ノードまたはツリーに対して ZKILL を実行します。
IrisGlobalOrder	指定したグローバルに対して $Order を実行します。
IrisGlobalQuery	指定したグローバルに対して $Query を実行します。
IrisGlobalRelease	保持されたグローバル・バッファの所有権 (存在する場合) を解放します。
IrisPushGlobal[W][H]	グローバル名を引数スタックにプッシュします
IrisPushGlobalX[W][H]	拡張グローバル名を引数スタックにプッシュします

2.6 文字列の管理

これらの関数は文字列をある形式から別の形式に変換します。また、文字列引数のプッシュ、またはポップを行います。これらの文字列関数は、標準的な文字列および従来の短い文字列の両方で使用できます。ローカルの 8 ビット・エンコーディング、2 バイト Unicode、および 4 バイト Unicode 向きの関数が用意されています。

テーブル 2-10: 文字列関数

IrisCvtExStrInA[W][H]	指定の外部文字セット・エンコーディングを持つ文字列を、InterSystems IRIS で内部的に使用される文字列エンコーディングに変換します。
IrisCvtExStrOutA[W][H]	InterSystems IRIS で内部的に使用される文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します。
IrisExStrKill	文字列に関連付けられているストレージを解放します。
IrisExStrNew[W][H]	文字列に対して要求されたストレージの容量を割り当て、EXSTR 構造に、構造の長さおよび構造の値フィールドへのポインタを埋め込みます。
IrisPopExStr[W][H]	引数スタックから値をポップ・オフし、必要なタイプの文字列に変換します。
IrisPushExStr[W][H]	文字列を引数スタックにプッシュします。

2.7 その他のデータ型の管理

以下の関数は、int、double、$list、または pointer などのデータ型の値を持つ引数をプッシュおよびポップして、ビット文字列内の指定されたビット値の位置を返すために使用されます。

テーブル 2-11: その他のデータ型の関数

IrisPushInt	integer 型を引数スタックにプッシュします
IrisPopInt	引数スタックから値をポップ・オフし、integer 型に変換します
IrisPushInt64	64 ビット (long long) 値を引数スタックにプッシュします
IrisPopInt64	引数スタックから値をポップ・オフし、64 ビット (long long) 値に変換します
IrisPushDbl	double 型を引数スタックにプッシュします
IrisPushIEEEDbl	IEEE の double 型を引数スタックにプッシュします
IrisPopDbl	引数スタックから値をポップ・オフし、double 型に変換します。
IrisPushList	$LIST オブジェクトを変換して、引数スタックにプッシュします。
IrisPopList	引数スタックから $LIST オブジェクトをポップ・オフして変換します。
IrisPushPtr	ポインタ値を引数スタックにプッシュします。
IrisPopPtr	引数スタックからポインタ値をポップ・オフします。
IrisPushUndef	省略された関数引数として解釈される未定義値をプッシュします。
IrisBitFind[B]	ビット文字列内の指定されたビット値の位置を返します。InterSystems IRIS $BITFIND と似ています。

3

コールイン関数リファレンス

このリファレンスの章では、すべての InterSystems コールイン関数をアルファベット順に挙げて、詳しく説明します。機能別のコールイン関数の概要については、"コールイン関数の使用法" を参照してください。

注釈　文字列を操作する InterSystems コールイン関数には、8 ビット・バージョンと Unicode バージョンの両方があります。これらの関数は、処理する文字列のタイプを示すために接尾語を使用します。

- 接尾語 "A" の付いた名前、またはまったく接尾語の付かない名前 (例えば、IrisEvalA または IrisPopExStr など) は、8 ビット文字列のバージョンです。
- 接尾語 W の付いた名前 (例えば、IrisEvalW または IrisPopExStrW など) は、2 バイトの Unicode 文字を使用するプラットフォーム上の Unicode 文字列用のバージョンです。
- 接尾語 H の付いた名前 (例えば、IrisEvalH または IrisPopExStrH など) は、4 バイトの Unicode 文字を使用するプラットフォーム上の Unicode 文字列用のバージョンです。

いくつかのバージョンがある関数であっても、ここではわかりやすいように各関数の説明でまとめて扱っています。例えば、IrisEvalA[W][H]、IrisPopExStr[W][H] のように記載しています。

3.1 アルファベット順の関数リスト

ここでは、すべてのコールイン関数をアルファベット順に挙げ、各関数の簡単な説明と詳細な説明へのリンクを示します。

- IrisAbort ー 必要に応じて、InterSystems IRIS® 側で処理中の現在の要求をキャンセルするように InterSystems IRIS に命令します。
- IrisAcquireLock ー InterSystems IRIS LOCK コマンドを実行します。IrisPushLockX[W][H] を使用してロック参照を設定しておく必要があります。
- IrisCallExecuteFunc ー コマンド文字列が IrisPushExecuteFuncA[W][H] によってスタックにプッシュされた後、$Xecute() 関数を実行します。
- IrisChangePasswordA[W][H] ー InterSystems 認証が使用されている場合に、ユーザのパスワードを変更します (他の形式の認証では無効)。
- IrisBitFind[B] ー ビット文字列内の指定されたビット値の位置を返します (InterSystems IRIS $BITFIND に類似)。
- IrisCloseOref ー OREF に対するシステム参照カウンタをデクリメントします。
- IrisContext ー 外部コールイン・プログラムを使用しているときに、接続の InterSystems IRIS 側で現在処理中の要求がある場合、`True` を返します。

- IrisConvert ー IrisEvalA[W][H] で返された値を適切な形式に変換し、その戻り値で指定されたアドレスに置きます。
- IrisCtrl ー InterSystems IRIS が CTRL-C を無視するかどうかを決定します。
- IrisCvtExStrInA[W][H] ー 指定の外部文字セット・エンコーディングを持つ文字列を、8 ビット・バージョンの InterSystems IRIS でのみ内部的に使用されるローカルの 8 ビット文字列エンコーディングに変換します。
- IrisCvtExStrOutA[W][H] ー InterSystems IRIS の 8 ビット製品で内部的に使用されるローカルの 8 ビット文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します (これは、8 ビット・バージョンの InterSystems IRIS でのみ使用できます)。
- IrisCvtInA[W][H] ー 指定の外部文字セット・エンコーディングを持つ文字列を、ローカルの 8 ビット文字列エンコーディング (8 ビット・バージョンの InterSystems IRIS でのみ内部的に使用される) または Unicode 文字列エンコーディング (Unicode バージョンの InterSystems IRIS で内部的に使用される) に変換します。
- IrisCvtOutA[W][H] ー InterSystems IRIS の 8 ビット製品で内部的に使用されるローカルの 8 ビット文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します (これは、8 ビット・バージョンの InterSystems IRIS でのみ使用できます)。
- IrisDoFun ー ルーチン呼び出しを実行します (特殊な場合)。
- IrisDoRtn ー ルーチン呼び出しを実行します。
- IrisEnd ー InterSystems IRIS プロセスを終了します。切断された接続が存在する場合は、削除操作も行います。
- IrisEndAll ー すべてのコールイン・スレッドを切断し、そのスレッドが終了するまで待機します。
- IrisErrorA[W][H] ー 最新のエラー・メッセージ、関連するソース文字列、ソース文字列内でのエラーの発生箇所のオフセットを返します。
- IrisErrxlateA[W][H] ー 整数のエラー・コードを InterSystems IRIS エラー文字列に変換します。
- IrisEvalA[W][H] ー 文字列を、それが InterSystems IRIS の式であるかのように評価し、戻り値をメモリに格納して、IrisType と IrisConvert による処理がさらに進められるようにします。
- IrisExecuteA[W][H] ー ターミナルで入力されたかのように、コマンド文字列を実行します。
- IrisExecuteArgs ー 引数を指定してコマンド文字列を実行します。
- IrisExStrKill ー EXSTR 文字列に関連付けられているストレージを解放します。
- IrisExStrNew[W][H] ー 文字列に対して要求されたストレージの容量を割り当て、EXSTR 構造に、構造の長さおよび構造の値フィールドへのポインタを埋め込みます。
- IrisExtFun ー 戻り値を引数スタックにプッシュする外部関数呼び出しを実行します。
- IrisGetProperty ー IrisPushProperty[W][H] により定義されたプロパティの値を取得します。この値は、引数スタックにプッシュされます。
- IrisGlobalData ー 指定したグローバルに対して $Data を実行します。
- IrisGlobalGet ー IrisPushGlobal[W][H] および任意の添え字で定義されたグローバル参照の値を取得します。このノード値は、引数スタックにプッシュされます。
- IrisGlobalIncrement ー ー$INCREMENT を実行し、スタックの一番上の結果を返します。
- IrisGlobalKill ー グローバル・ノードまたはツリーに対して ZKILL を実行します。
- IrisGlobalOrder ー 指定したグローバルに対して $Order を実行します。
- IrisGlobalQuery ー 指定したグローバルに対して $Query を実行します。
- IrisGlobalRelease ー 保持されたグローバル・バッファの所有権 (存在する場合) を解放します。
- IrisGlobalSet ー IrisPushGlobal[W][H] および任意の添え字により定義されたグローバル参照の値を格納します。この呼び出しを行う前に、ノード値を引数スタックにプッシュしておく必要があります。

- IrisIncrementCountOref ー OREF に対するシステム参照カウンタをインクリメントします。
- IrisInvokeClassMethod ー IrisPushClassMethod[W][H] および任意の引数によって定義されたクラス・メソッド呼び出しを実行します。返り値は、引数スタックにプッシュされます。
- IrisInvokeMethod ー IrisPushMethod[W][H]、および引数スタックにプッシュされた任意の引数によって定義されるインスタンス・メソッド呼び出しを実行します。
- IrisOflush ー 保留中の出力をすべてフラッシュします。
- IrisPop ー 引数スタックから値をポップ・オフします。
- IrisPopCvtW[H] ー 引数スタックからローカルの 8 ビット文字列をポップ・オフし、Unicode に変換します。Unicode バージョンの IrisPopStr[W][H] と同じです。
- IrisPopDbl ー 引数スタックから値をポップ・オフし、double 型に変換します。
- IrisPopExStr[W][H] ー 引数スタックから値をポップ・オフし、string 型に変換します。
- IrisPopExStrCvtW[H] ー 引数スタックから値をポップ・オフし、long 型の Unicode 文字列に変換します。
- IrisPopInt ー 引数スタックから値をポップ・オフし、integer 型に変換します。
- IrisPopInt64 ー 引数スタックから値をポップ・オフし、64 ビット (long long) 数値に変換します。
- IrisPopList ー 引数スタックから $LIST オブジェクトをポップ・オフして変換します。
- IrisPopOref ー 引数スタックから OREF をポップ・オフします。
- IrisPopPtr ー 引数スタックから、ポインタを内部形式でポップ・オフします。
- IrisPopStr[W][H] ー 引数スタックから値をポップ・オフし、string 型に変換します。
- IrisPromptA[W][H] ー ターミナルである文字列を返します。
- IrisPushClassMethod[W][H] ー クラス・メソッド参照を引数スタックにプッシュします。
- IrisPushCvtW[H] ー Unicode 文字列をローカルの 8 ビットに変換して、引数スタックにプッシュします。Unicode バージョンの IrisPushStr[W][H] と同じです。
- IrisPushDbl ー double 型を引数スタックにプッシュします。
- IrisPushExecuteFuncA[W][H] ー IrisCallExecuteFunc による呼び出しの準備として $Xecute() コマンドをスタックにプッシュします。
- IrisPushExStr[W][H] ー 文字列を引数スタックにプッシュします。
- IrisPushExStrCvtW[H] ー Unicode 文字列をローカルの 8 ビット・エンコーディングに変換して、引数スタックにプッシュします。
- IrisPushFunc[W][H] ー 外部関数参照を引数スタックにプッシュします。
- IrisPushFuncX[W][H] ー 拡張外部関数参照を引数スタックにプッシュします。
- IrisPushGlobal[W][H] ー グローバル参照を引数スタックにプッシュします。
- IrisPushGlobalX[W][H] ー 拡張グローバル参照を引数スタックにプッシュします。
- IrisPushIEEEDbl ー IEEE の double 型を引数スタックにプッシュします。
- IrisPushInt ー integer 型を引数スタックにプッシュします。
- IrisPushInt64 ー 64 ビット (long long) 数値を引数スタックにプッシュします。
- IrisPushList ー $LIST オブジェクトを変換し、引数スタックにプッシュします。
- IrisPushLock[W][H] ー ロック名を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

- IrisPushLockX[W][H] ー ロック名と環境文字列を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。
- IrisPushMethod[W][H] ー インスタンス・メソッド参照を引数スタックにプッシュします。
- IrisPushOref ー OREF を引数スタックにプッシュします。
- IrisPushProperty[W][H] ー プロパティ参照を引数スタックにプッシュします。
- IrisPushPtr ー ポインタを内部形式で引数スタックにプッシュします。
- IrisPushRtn[W][H] ー ルーチン参照を引数スタックにプッシュします。
- IrisPushRtnX[W][H] ー 拡張ルーチン参照を引数スタックにプッシュします。
- IrisPushStr[W][H] ー バイト文字列を引数スタックにプッシュします。
- IrisPushUndef ー 省略された関数引数として解釈される未定義値をプッシュします。
- IrisReleaseAllLocks ー 引数なしの InterSystems IRIS LOCK コマンドを実行して、プロセスによって現在保持されているロックをすべて削除します。
- IrisReleaseLock ー InterSystems IRIS LOCK コマンドを実行して、指定したロック名のロック・カウントをデクリメントします。このコマンドは、一度に 1 つずつ増分ロックを解放するだけです。
- IrisSecureStartA[W][H] ー プロセスをセットアップするために、InterSystems IRIS を呼び出します。
- IrisSetDir ー 実行時に、マネージャのディレクトリの名前を動的に設定します。
- IrisSetProperty ー IrisPushProperty[W][H] により定義されたプロパティの値を格納します。
- IrisSignal ー ユーザ・プログラムにより補足された信号を InterSystems IRIS に渡します。
- IrisSPCReceive ー シングル・プロセス通信メッセージを受信します。
- IrisSPCSend ー シングル・プロセス通信メッセージを送信します。
- IrisStartA[W][H] ー InterSystems IRIS プロセスをセットアップするために、InterSystems IRIS を呼び出します。
- IrisTCommit ー InterSystems IRIS TCommit コマンドを実行します。
- IrisTLevel ー トランザクション処理に対する、現在の入れ子レベル ($TLEVEL) を返します。
- IrisTRollback ー InterSystems IRIS TRollback コマンドを実行します。
- IrisTStart ー InterSystems IRIS TStart コマンドを実行します。
- IrisType ー IrisEvalA[W][H] により、関数値として返される項目のネイティブ・タイプを返します。
- IrisUnPop ー IrisPop からスタック・エントリをリストアします。

3.2 IrisAbort

```
int IrisAbort(unsigned long type)
```

引数

type	接続の切断方法を指定する次の事前定義値のどちらかになります。 ・ IRIS_CTRLC – (CTRL-C が IrisCtrl で有効になっているかどうかに関係なく) CTRL-C が処理されたかのように、InterSystems IRIS の処理に割り込みます。InterSystems IRIS への接続は維持されます。 ・ IRIS_RESJOB – コールイン接続を切断します。その後、最初に IrisEnd、次に IrisStart を呼び出し、InterSystems IRIS に再接続する必要があります。

説明

必要に応じて、InterSystems IRIS 側で処理中の現在の要求をキャンセルするように InterSystems IRIS に命令します。この関数は、AST (非同期システム・トラップ)、またはコールイン側で実行されているスレッドで重大なイベントが検知された場合に使用します (現在、処理中の InterSystems IRIS 要求があるかどうかを判断するには、IrisContext を使用します)。ただし、これは AST または独立したスレッドを使用するコールイン・プログラムにのみ当てはまります。

IrisAbort の返り値

IRIS_BADARG	終了タイプは無効です。
IRIS_CONBROKEN	接続は切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_NOTINCACHE	現在コールイン・パートナーが InterSystems IRIS に存在しません。
IRIS_SUCCESS	接続が確立されました。

例

```
rc = IrisAbort(IRIS_CTRLC);
```

3.3 IrisAcquireLock

```
int IrisAcquireLock(int nsub, int flg, int tout, int * rval)
```

引数

nsub	ロック参照にある添え字の数。
flg	ロック・コマンドの修飾語。有効な値は、IRIS_INCREMENTAL_LOCK と IRIS_SHARED_LOCK のいずれか、または両方です。
tout	ロック・コマンドが完了するのを待機する時間 (秒)。タイムアウトを指定しない場合、マイナスの値。0 は、ロックが利用できない場合に即座に返すという意味です。ただし、ロックがリモート・システムにマップされている場合には最小のタイムアウトが適用される場合があります。
rval	成功を 1、失敗を 0 で示す int 返り値への、オプションのポインタ。

説明

InterSystems IRIS LOCK コマンドを実行します。ロック参照は、IrisPushLock で設定済みである必要があります。

IrisAcquireLock の返り値

IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	LOCK コマンドを正常に呼び出しました (ただし、rval パラメータを調べて、ロックが成功したかどうかを判断する必要があります)。
IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。

3.4 IrisBitFind

```
int IrisBitFind(int strlen, unsigned short *bitstr, int newlen, int srch, int revflg)
```

引数

strlen	ビット文字列のデータ長。
bitstr	Unicode ビット文字列へのポインタ。
newlen	文字列の最初から検索を開始する場合は 0 を指定し、それ以外は 1 から始まる開始位置を指定します。
srch	ビット文字列内を検索するビット値 (0 または 1)。
revflg	検索の方向を指定します。 1 ― newlen で指定された位置から順方向 (左から右) に検索します。 0 ― newlen で指定された位置から逆方向に検索します。

説明

ビット文字列 bitstr 内の srch で指定された値を持つビットの次のビット位置 (先頭は 1) を返します。検索の方向は revflg で指定します。指定された方向に指定された値のビットがそれ以上見つからない場合は 0 が返されます。

この関数は、InterSystems IRIS $BITFIND に類似しています (“ビット文字列関数の概要” も参照してください)。

IrisBitFind の返り値

IRIS_SUCCESS	操作は正常に終了しました。

3.5 IrisBitFindB

```
int IrisBitFindB(int strlen, unsigned char *bitstr, int newlen, int srch, int revflg)
```

引数

strlen	ビット文字列のデータ長。
bitstr	ビット文字列へのポインタ。
newlen	文字列の最初から検索を開始する場合は 0 を指定し、それ以外は 1 から始まる開始位置を指定します。
srch	ビット文字列内を検索するビット値 (0 または 1)。
revflg	検索の方向を指定します。 1 ― newlen で指定された位置から順方向 (左から右) に検索します。 0 ― newlen で指定された位置から逆方向に検索します。

説明

ビット文字列 bitstr 内の srch で指定された値を持つビットの次のビット位置 (先頭は 1) を返します。検索の方向は revflg で指定します。指定された方向に指定された値のビットがそれ以上見つからない場合は 0 が返されます。

この関数は、InterSystems IRIS $BITFIND に類似しています (“ビット文字列関数の概要” も参照してください)。

IrisBitFindB の返り値

IRIS_SUCCESS	操作は正常に終了しました。

3.6 IrisCallExecuteFunc

```
int IrisCallExecuteFunc(int numargs)
```

引数

numargs	スタックにプッシュされた引数の数。

説明

引数を指定して $Xecute() 関数を呼び出します。この関数のコマンド文字列を IrisPushExecuteFuncA[W][H] で引数スタックにプッシュした後、引数をプッシュする必要があります。引数の数は 0 に設定できます。関数の結果は、引数スタック上に残ります。

IrisCallExecuteFunc の戻り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_ERPARAMETER	Xecute 文字列は引数を予期していません。
IRIS_SUCCESS	操作は正常に終了しました。
IRIS_FAILURE	予期しないエラーが発生しました。

3.7 IrisChangePasswordA

バリアント：IrisChangePasswordW、IrisChangePasswordH

```
int IrisChangePasswordA(IRIS_ASTRP username, IRIS_ASTRP oldpassword, IRIS_ASTRP newpassword)
```

引数

username	変更が必要なパスワードを使用しているユーザのユーザ名。
oldpassword	ユーザの古いパスワード。
newpassword	新しいパスワード。

説明

この関数は、InterSystems 認証または代行認証が使用されている場合、ユーザのパスワードを変更できます。LDAP、Kerberos、またはその他の形式の認証では有効ではありません。コールイン・セッションが初期化される前に呼び出される必要があります。典型的な使用法は、IrisSecureStart からの **IRIS_CHANGEPASSWORD** エラーを処理することです。この場合、パスワードを変更するために IrisChangePassword が呼び出され、それから IrisSecureStart が再度呼び出されます。

IrisChangePasswordA の返り値

IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	パスワードが変更されました。

3.8 IrisChangePasswordH

バリアント：IrisChangePasswordA、IrisChangePasswordW

```
int IrisChangePasswordH(IRISHSTRP username, IRISHSTRP oldpassword, IRISHSTRP newpassword)
```

引数

username	変更が必要なパスワードを使用しているユーザのユーザ名。
oldpassword	ユーザの古いパスワード。
newpassword	新しいパスワード。

説明

この関数は、InterSystems 認証または代行認証が使用されている場合、ユーザのパスワードを変更できます。LDAP、Kerberos、またはその他の形式の認証では有効ではありません。コールイン・セッションが初期化される前に呼び出される必要があります。典型的な使用法は、IrisSecureStart からの **IRIS_CHANGEPASSWORD** エラーを処理することです。この場合、パスワードを変更するために IrisChangePassword が呼び出され、それから IrisSecureStart が再度呼び出されます。

IrisChangePasswordH の返り値

IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	パスワードが変更されました。

3.9 IrisChangePasswordW

バリアント：IrisChangePasswordA、IrisChangePasswordH

```
int IrisChangePasswordW(IRISWSTRP username, IRISWSTRP oldpassword, IRISWSTRP newpassword)
```

引数

username	変更が必要なパスワードを使用しているユーザのユーザ名。
oldpassword	ユーザの古いパスワード。
newpassword	新しいパスワード。

説明

この関数は、InterSystems 認証または代行認証が使用されている場合、ユーザのパスワードを変更できます。LDAP、Kerberos、またはその他の形式の認証では有効ではありません。コールイン・セッションが初期化される前に呼び出される必要があります。典型的な使用法は、IrisSecureStart からの IRIS_CHANGEPASSWORD エラーを処理することです。この場合、パスワードを変更するために IrisChangePassword が呼び出され、それから IrisSecureStart が再度呼び出されます。

IrisChangePasswordW の返り値

IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	パスワードが変更されました。

3.10 IrisCloseOref

```
int IrisCloseOref(unsigned int oref)
```

引数

oref	オブジェクト参照。

説明

OREF に対するシステム参照カウンタをデクリメントします。

IrisCloseOref の返り値

IRIS_ERBADOREF	OREF が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.11 IrisContext

```
int IrisContext()
```

説明

関数値として整数を返します。

($ZF 関数から呼び出されたモジュールとは対照的に) 外部コールイン・プログラムを使用していて、このプログラムが AST または独立したスレッドを使用している場合、IrisContext は、接続の InterSystems IRIS 側に現在処理中の要求が存在するかどうかを返します。これは、処理を完了させるには、InterSystems IRIS に戻る必要があるかどうかを判断するために必要な情報です。

IrisContext の返り値

-1	$ZF コールバック経由で InterSystems IRIS に作成されました。
0	現在、接続がありません。あるいは、InterSystems IRIS に入っていません。
1	外部接続 (例えば、$ZF ではない接続) から InterSystems IRIS に入りました。終了ハンドラなどの AST (非同期システム・トラップ) は、処理を完了できるように InterSystems IRIS に返す必要があります。

注釈　$ZF 関数にプログラムと AST のどちらから入ったかという情報が必要なのは、AST からの場合、処理を完了させるには InterSystems IRIS に戻る必要があるからです。

例

```
rc = IrisContext();
```

3.12 IrisConvert

```
int IrisConvert(unsigned long type, void * rbuf)
```

引数

type	#define で定義された型。以下の有効な値を持ちます。
rbuf	このデータ型に適したサイズのデータ領域のアドレス。型が IRIS_ASTRING の場合、rbuf は、結果を含む IRIS_ASTR 構造のアドレスになります。また、この構造の len 要素には、返される文字列の最大サイズ (文字数) を表す値を入力する必要があります。同様に、型が IRIS_WSTRING である場合、rbuf は、IRISWSTR 構造のアドレスになります。この構造の len 要素には、最大サイズ (文字数) を表す値を入力する必要があります。

説明

IrisEval で返された値を適切な形式に変換し、その返り値 (以下の rbuf を参照) を指定されたアドレスに置きます。

有効な type の値は以下のとおりです。

- IRIS_ASTRING ー 8 ビット文字列
- IRIS_CHAR ー 8 ビット符号付整数
- IRIS_DOUBLE ー 64 ビット浮動小数点
- IRIS_FLOAT ー 32 ビット浮動小数点

- IRIS_INT ー 32 ビット符号付整数
- IRIS_INT2 ー 16 ビット符号付整数
- IRIS_INT4 ー 32 ビット符号付整数
- IRIS_INT8 ー 64 ビット符号付整数
- IRIS_UCHAR ー 8 ビット符号なし整数
- IRIS_UNIT ー 32 ビット符号なし整数
- IRIS_UNIT2 ー 16 ビット符号なし整数
- IRIS_UNIT4 ー 32 ビット符号なし整数
- IRIS_UNIT8 ー 64 ビット符号なし整数
- IRIS_WSTRING ー Unicode 文字列

IrisConvert の返り値

IRIS_BADARG	型が無効です。
IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_NOCON	接続が確立されていません。
IRIS_NORES	返すことができるタイプを持つ結果がありません（この呼び出しの前に、IrisEvalA が呼び出されていません）。
IRIS_RETTRUNC	成功しました。しかし、タイプ IRIS_ASTRING、IRIS_INT8、IRIS_UINT8、および IRIS_WSTRING は、retval に割り当てられた領域に収まらない値になりました。つまり、IRIS_INT8 と IRIS_UINT8 に対し、式の浮動小数点数が、64 ビットに適応するように正規化されなかったことを意味します。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	最後の IrisEval で返された値は、適切に変換されました。

注釈　InterSystems IRIS は、（負の指数を含む）小数部を持つ浮動小数点タイプ IRIS_FLOAT と IRIS_DOUBLE、および 64 ビット整数タイプ（IRIS_INT8 と IRIS_UINT8）の返り値の算出時、除算を実行します。したがって、元の値とは異なる値が返される可能性があります。IRIS_ASTRING、IRIS_INT8、IRIS_UINT8、および IRIS_WSTRING は、状態 IRIS_RETTRUNC を返すことができます。

例

```
IRIS_ASTR retval;
/* define variable retval */

retval.len = 20;
/* maximum return length of string */

rc = IrisConvert(IRIS_ASTRING,&retval);
```

3.13 IrisCtrl

```
int IrisCtrl(unsigned long flags)
```

引数

flags	特定のキーストロークが、InterSystems IRIS でどのように処理されるかを表す 2 種類の #define 値のどちらかになります。

説明

InterSystems IRIS が CTRL-C を無視するかどうかを決定します。flags は、以下に関するビット状態値を持つことができます。

- IRIS_DISACTRLC — InterSystems IRIS は CTRL-C を無視します。
- IRIS_ENABCTRLC — BREAK コマンドまたは OPEN コマンドでオーバーライドされていない限り、関数が呼び出されない場合の既定値として使用されます。InterSystems IRIS では、CTRL-C は <INTERRUPT> を生成します。

IrisCtrl の返り値

IRIS_FAILURE	(コールイン実行可能プログラムの内部からではなく) $ZF 関数から呼び出された場合に返されます。
IRIS_SUCCESS	制御関数が実行されました。

例

```
rc = IrisCtrl(IRIS_ENABCTRLC);
```

3.14 IrisCvtExStrInA

バリアント：IrisCvtExStrInW、IrisCvtExStrInH

```
int IrisCvtExStrInA(IRIS_EXSTRP src, IRIS_ASTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、内部的に使用されるローカルの 8 ビット文字列エンコーディングに変換します。

IrisCvtExStrInA の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	Unicode では使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.15 IrisCvtExStrInW

バリアント：IrisCvtExStrInA、IrisCvtExStrInH

```
int IrisCvtExStrInW(IRIS_EXSTRP src, IRISWSTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、InterSystems IRIS で内部的に使用される 2 バイトの Unicode 文字列エンコーディングに変換します。

IrisCvtExStrInW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.16 IrisCvtExStrInH

バリアント：IrisCvtExStrInA、IrisCvtExStrInW

```
int IrisCvtExStrInH(IRIS_EXSTRP src, IRISWSTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、InterSystems IRIS で内部的に使用される 4 バイトの Unicode 文字列エンコーディングに変換します。

IrisCvtExStrInH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.17 IrisCvtExStrOutA

バリアント：IrisCvtExStrOutW、IrisCvtExStrOutH

```
int IrisCvtExStrOutA(IRIS_EXSTRP src, IRIS_ASTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

従来の InterSystems の 8 ビット製品で内部的に使用される 8 ビット文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します。

IrisCvtExStrOutA の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	Unicode では使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.18 IrisCvtExStrOutW

バリアント：IrisCvtExStrOutA、IrisCvtExStrOutH

```
int IrisCvtExStrOutW(IRIS_EXSTRP src, IRISWSTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

InterSystems IRIS で内部的に使用される 2 バイトの Unicode 文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します。

IrisCvtExStrOutW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.19 IrisCvtExStrOutH

バリアント：IrisCvtExStrOutA、IrisCvtExStrOutW

```
int IrisCvtExStrOutH(IRIS_EXSTRP src, IRISWSTRP tbl, IRIS_EXSTRP res)
```

引数

src	変換する文字列を含む IRIS_EXSTRP 変数のアドレスです。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	結果を含む IRIS_EXSTRP 変数のアドレスです。

説明

InterSystems IRIS で内部的に使用される 4 バイトの Unicode 文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します。

IrisCvtExStrOutH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.20 IrisCvtInA

バリアント：IrisCvtInW、IrisCvtInH

```
int IrisCvtInA(IRIS_ASTRP src, IRIS_ASTRP tbl, IRIS_ASTRP res)
```

引数

src	変換される外部文字セット・エンコーディングの文字列 (計算文字列バッファを使用して記述されます)。この文字列は初期化する必要があります。例えば、値を、出力として期待される最大文字数を表す空白の数に設定します。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	8 ビットの計算文字列の結果を含む IRIS_ASTR 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、8 ビット・バージョンの InterSystems IRIS でのみ内部的に使用されるローカルの 8 ビット文字列エンコーディングに変換します。

IrisCvtInA の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	Unicode では使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.21 IrisCvtInW

バリアント：IrisCvtInA、IrisCvtInH

```
int IrisCvtInW(IRIS_ASTRP src, IRISWSTRP tbl, IRISWSTRP res)
```

引数

src	変換される外部文字セット・エンコーディングの文字列 (Unicode 文字列を格納するために必要なバイト数を使用して記述されます)。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	Unicode の計算文字列の結果を含む IRISWSTR 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、Unicode バージョンの InterSystems IRIS で内部的に使用される Unicode 文字列エンコーディングに変換します。

IrisCvtInW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.22 IrisCvtInH

バリアント：IrisCvtInA、IrisCvtInW

```
int IrisCvtInH(IRIS_ASTRP src, IRISHSTRP tbl, IRISHSTRP res)
```

引数

src	変換される外部文字セット・エンコーディングの文字列 (Unicode 文字列を格納するために必要なバイト数を使用して記述されます)。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	Unicode の計算文字列の結果を含む IRISHSTRP 変数のアドレスです。

説明

指定の外部文字セット・エンコーディングを持つ文字列を、Unicode バージョンの InterSystems IRIS で内部的に使用される Unicode 文字列エンコーディングに変換します。

IrisCvtInH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.23 IrisCvtOutA

バリアント：IrisCvtOutW、IrisCvtOutH

```
int IrisCvtOutA(IRIS_ASTRP src, IRIS_ASTRP tbl, IRIS_ASTRP res)
```

引数

src	InterSystems IRIS の 8 ビット製品で内部的に使用されるローカルの 8 ビット文字列エンコーディングの文字列 (NULL ポインタが渡される場合、InterSystems IRIS は IrisEvalA または IrisEvalW への前回の呼び出しの結果を使用します)。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	ターゲットの外部文字セット・エンコーディングの結果を含む IRIS_ASTR 変数のアドレス (8 ビットの計算文字列バッファを使用して記述されます)。

説明

InterSystems IRIS の 8 ビット製品で内部的に使用されるローカルの 8 ビット文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します (これは、8 ビット・バージョンの InterSystems IRIS でのみ使用できます)。

IrisCvtOutA の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	Unicode では使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.24 IrisCvtOutW

バリアント：IrisCvtOutA、IrisCvtOutH

```
int IrisCvtOutW(IRISWSTRP src, IRISWSTRP tbl, IRIS_ASTRP res)
```

引数

src	InterSystems IRIS の Unicode 製品で内部的に使用される Unicode 文字列エンコーディングの文字列 (NULL ポインタが渡される場合、InterSystems IRIS は IrisEvalA または IrisEvalW への前回の呼び出しの結果を使用します)。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	ターゲットの外部文字セット・エンコーディングの結果を含む IRIS_ASTR 変数のアドレス (8 ビットの計算文字列バッファを使用して記述されます)。

説明

Unicode バージョンの InterSystems IRIS で内部的に使用される Unicode 文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します (これは、Unicode バージョンの InterSystems IRIS でのみ使用できます)。

IrisCvtOutW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.25 IrisCvtOutH

バリアント：IrisCvtOutA、IrisCvtOutW

```
int IrisCvtOutH(IRISHSTRP src, IRISHSTRP tbl, IRIS_ASTRP res)
```

引数

src	InterSystems IRIS の Unicode 製品で内部的に使用される Unicode 文字列エンコーディングの文字列 (NULL ポインタが渡される場合、InterSystems IRIS は IrisEvalA または IrisEvalW への前回の呼び出しの結果を使用します)。
tbl	変換を実行するために使用する入出力変換テーブル名です (NULL 文字列の場合、既定処理で入出力変換テーブル名を使用する必要があることを示します)。
res	ターゲットの外部文字セット・エンコーディングの結果を含む IRIS_ASTR 変数のアドレス (8 ビットの計算文字列バッファを使用して記述されます)。

説明

Unicode バージョンの InterSystems IRIS で内部的に使用される Unicode 文字列エンコーディングからの文字列を、指定の外部文字セット・エンコーディングを持つ文字列に変換します (これは、Unicode バージョンの InterSystems IRIS でのみ使用できます)。

IrisCvtOutH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_ERRUNIMPLEMENTED	8 ビット・システムでは使用できません。
IRIS_ERVALUE	指定の入出力変換テーブル名は未定義であるか、あるいは入力コンポーネントがありませんでした。
IRIS_ERXLATE	入力文字列は、指定の入出力変換テーブルを使用して変換できませんでした。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTRUNC	結果用のバッファが小さすぎたため、結果が切り捨てられました。
IRIS_FAILURE	変換データ構造の構築中にエラーが発生しました (パーティション・メモリ不足の場合があります)。
IRIS_SUCCESS	変換が正常に完了しました。

3.26 IrisDoFun

```
int IrisDoFun(unsigned int flags, int narg)
```

引数

flags	IrisPushRtn[XW] からのルーチン・フラグ。
narg	引数スタックにプッシュされた呼び出し引数の数。ターゲットには、(できるだけ空の) 仮パラメータ・リストが必要です。

説明

ルーチン呼び出しを実行します (特殊な場合)。

IrisDoFun の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_FAILURE	内部的な整合性にエラーがあります。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.27 IrisDoRtn

```
int IrisDoRtn(unsigned int flags, int narg)
```

引数

flags	IrisPushRtn[XW] からのルーチン・フラグ。
narg	引数スタックにプッシュされた呼び出し引数の数。0 の場合、ターゲットは仮パラメータ・リストを持つことができません。

説明

ルーチン呼び出しを実行します。

IrisDoRtn の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_FAILURE	内部的な整合性にエラーがあります。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.28 IrisEnd

```
int IrisEnd()
```

説明

InterSystems IRIS プロセスを終了します。切断された接続が存在する場合は、削除操作も行います。

IrisEnd の返り値

IRIS_FAILURE	(コールイン実行可能プログラムの内部からではなく) $ZF 関数から呼び出された場合に返されます。
IRIS_NOCON	接続が確立されていません。
IRIS_SUCCESS	InterSystems IRIS セッションが終了、あるいは削除されました。

IrisEnd は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
rc = IrisEnd();
```

3.29 IrisEndAll

```
int IrisEndAll()
```

説明

スレッド・コールイン環境のすべてのスレッドを切断してから、スレッドが終了するようにスケジュールし、それらのスレッドが終了するまで待機します。

IrisEndAll の返り値

IRIS_SUCCESS	InterSystems IRIS セッションが終了、あるいは削除されました。

例

```
rc = IrisEndAll();
```

3.30 IrisErrorA

バリアント：IrisErrorW、IrisErrorH

```
int IrisErrorA(IRIS_ASTRP msg, IRIS_ASTRP src, int * offp)
```

引数

msg	エラー・メッセージ、またはエラー・メッセージを受信する変数のアドレス。
src	エラーのソース文字列、またはエラー・メッセージのソース文字列を受信する変数のアドレス。
offp	errsrc の位置に対するオフセットを指定する整数、またはエラー・メッセージのソース文字列に対するオフセットを受信する整数のアドレス。

説明

最新のエラー・メッセージ、それに関連付けられたソース文字列、およびそのソース文字列の中でエラーが発生した箇所までのオフセットを返します。

IrisErrorA の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	errmsg または errsrc の返り値の長さが有効なサイズではありませんでした。
IRIS_SUCCESS	接続が確立されました。

例

```
IRIS_ASTR errmsg;
IRIS_ASTR srcline;
int offset;
errmsg.len = 50;
srcline.len = 100;
if ((rc = IrisErrorA(&errmsg, &srcline, &offset)) != IRIS_SUCCESS)
printf("\r\nfailed to display error - rc = %d",rc);
```

3.31 IrisErrorH

バリアント：IrisErrorA、IrisErrorW

```
int IrisErrorH(IRISHSTRP msg, IRISHSTRP src, int * offp)
```

引数

msg	エラー・メッセージ、またはエラー・メッセージを受信する変数のアドレス。
src	エラーのソース文字列、またはエラー・メッセージのソース文字列を受信する変数のアドレス。
offp	errsrc の位置に対するオフセット、またはエラー・メッセージのソース文字列に対するオフセットを受信する整数のアドレス。

説明

最新のエラー・メッセージ、それに関連付けられたソース文字列、およびそのソース文字列の中でエラーが発生した箇所までのオフセットを返します。

IrisErrorH の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	errmsg または errsrc の返り値の長さが有効なサイズではありませんでした。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISHSTRP errmsg;
IRISHSTRP srcline;
int offset;
errmsg.len = 50;
srcline.len = 100;
if ((rc = IrisErrorH(&errmsg, &srcline, &offset)) != IRIS_SUCCESS)
printf("\r\nfailed to display error - rc = %d",rc);
```

3.32 IrisErrorW

バリアント：IrisErrorA、IrisErrorH

```
int IrisErrorW(IRISWSTRP msg, IRISWSTRP src, int * offp)
```

引数

msg	エラー・メッセージ、またはエラー・メッセージを受信する変数のアドレス。
src	エラーのソース文字列、またはエラー・メッセージのソース文字列を受信する変数のアドレス。
offp	errsrc の位置に対するオフセット、またはエラー・メッセージのソース文字列に対するオフセットを受信する整数のアドレス。

説明

最新のエラー・メッセージ、それに関連付けられたソース文字列、およびそのソース文字列の中でエラーが発生した箇所までのオフセットを返します。

IrisErrorW の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	errmsg または errsrc の返り値の長さが有効なサイズではありませんでした。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISWSTRP errmsg;
IRISWSTRP srcline;
int offset;
errmsg.len = 50;
srcline.len = 100;
if ((rc = IrisErrorW(&errmsg, &srcline, &offset)) != IRIS_SUCCESS)
printf("\r\nfailed to display error - rc = %d",rc);
```

3.33 IrisErrxlateA

バリアント：IrisErrxlateW、IrisErrxlateH

```
int IrisErrxlateA(int code, IRIS_ASTRP rbuf)
```

引数

code	エラー・コードです。
rbuf	InterSystems IRIS エラー文字列を含む IRIS_ASTR 変数のアドレスです。len フィールドには、返される文字列の最大サイズをロードする必要があります。

説明

エラー・コード code を InterSystems IRIS エラー文字列に変換し、その文字列を rbuf で示す構造に書き込みます。

IrisErrxlateA の返り値

IRIS_ERUNKNOWN	指定のコードは、(コールイン・インタフェース・エラー範囲で) 1 より小さいか、最大の InterSystems IRIS エラー番号より大きくなっています。
IRIS_RETTRUNC	関連するエラー文字列は、割り当てられる場所に収まるように切り捨てられます。
IRIS_SUCCESS	接続が確立されました。

例

```
IRIS_ASTR retval; /* define variable retval */
retval.len = 30; /* maximum return length of string */
rc = IrisErrxlateA(IRIS_ERSTORE,&retval);
```

3.34 IrisErrxlateH

バリアント：IrisErrxlateA、IrisErrxlateW

```
int IrisErrxlateH(int code, IRISHSTRP rbuf)
```

引数

code	エラー・コードです。
rbuf	InterSystems IRIS エラー文字列を含む IRISHSTRP 変数のアドレスです。len フィールドには、返される文字列の最大サイズをロードする必要があります。

説明

エラー・コード code を InterSystems IRIS エラー文字列に変換し、その文字列を rbuf で示す構造に書き込みます。

IrisErrxlateH の返り値

IRIS_ERUNKNOWN	指定のコードは、(コールイン・インタフェース・エラー範囲で) 1 より小さいか、最大の InterSystems IRIS エラー番号より大きくなっています。
IRIS_RETTRUNC	関連するエラー文字列は、割り当てられる場所に収まるように切り捨てられます。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISHSTR retval; /* define variable retval */
retval.len = 30; /* maximum return length of string */
rc = IrisErrxlateH(IRIS_ERSTORE,&retval);
```

3.35 IrisErrxlateW

バリアント：IrisErrxlateA、IrisErrxlateH

```
int IrisErrxlateW(int code, IRISWSTRP rbuf)
```

引数

code	エラー・コードです。
rbuf	InterSystems IRIS エラー文字列を含む IRISWSTR 変数のアドレスです。len フィールドには、返される文字列の最大サイズをロードする必要があります。

説明

エラー・コード code を InterSystems IRIS エラー文字列に変換し、その文字列を rbuf で示す構造に書き込みます。

IrisErrxlateW の返り値

IRIS_ERUNKNOWN	指定のコードは、(コールイン・インタフェース・エラー範囲で) 1 より小さいか、最大の InterSystems IRIS エラー番号より大きくなっています。
IRIS_RETTRUNC	関連するエラー文字列は、割り当てられる場所に収まるように切り捨てられます。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISWSTR retval; /* define variable retval */
retval.len = 30; /* maximum return length of string */
rc = IrisErrxlateW(IRIS_ERSTORE,&retval);
```

3.36 IrisEvalA

バリアント：IrisEvalW、IrisEvalH

```
int IrisEvalA(IRIS_ASTRP volatile expr)
```

引数

expr	IRIS_ASTR 変数のアドレス。

説明

文字列を、それが InterSystems IRIS の式であるかのように評価し、返り値をメモリに格納して、IrisType と IrisConvert による処理がさらに進められるようにします。

IrisEvalA が正常に完了した場合、IrisType と IrisConvert への呼び出しが完了するようにフラグがセットされます。これらの関数は、IrisEvalA から返される項目の処理に使用されます。

注意　　次に、IrisEvalA、IrisExecuteA、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisEvalA の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRIS_ERSYSTEM	InterSystems IRIS が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に評価されました。

IrisEvalA は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
IRIS_ASTR retval;
IRIS_ASTR expr;

strcpy(expr.str, "\"Record\"_^Recnum_\" = \"_$$^GetRec(^Recnum)");
expr.len = strlen(expr.str);
rc = IrisEvalA(&expr);
if (rc == IRIS_SUCCESS)
    rc = IrisConvert(IRIS_ASTRING,&retval);
```

3.37 IrisEvalH

バリアント：IrisEvalA、IrisEvalW

```
int IrisEvalH(IRISHSTRP volatile expr)
```

引数

expr	IRISHSTRP 変数のアドレス。

説明

文字列を、それが InterSystems IRIS の式であるかのように評価し、返り値をメモリに格納して、IrisType と IrisConvert による処理がさらに進められるようにします。

IrisEvalH が正常に完了した場合、IrisType と IrisConvert への呼び出しが完了するようにフラグがセットされます。これらの関数は、IrisEvalA から返される項目の処理に使用されます。

注意　　次に、IrisEvalH、IrisExecuteH、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisEvalH の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRISW_ERSYSTEM	InterSystems IRIS が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に評価されました。

IrisEvalH は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
IRISHSTRP retval;
IRISHSTRP expr;

strcpy(expr.str, "\"Record\"_^Recnum_\" = \"_$$^GetRec(^Recnum)");
expr.len = strlen(expr.str);
rc = IrisEvalH(&expr);
if (rc == IRIS_SUCCESS)
    rc = IrisConvert(ING,&retval);
```

3.38 IrisEvalW

バリアント：IrisEvalA、IrisEvalH

```
int IrisEvalW(IRISWSTRP volatile expr)
```

引数

expr	IRISWSTR 変数のアドレス。

説明

文字列を、それが InterSystems IRIS の式であるかのように評価し、返り値をメモリに格納して、IrisType と IrisConvert による処理がさらに進められるようにします。

IrisEvalW が正常に完了した場合、IrisType と IrisConvert への呼び出しが完了するようにフラグがセットされます。これらの関数は、IrisEvalA から返される項目の処理に使用されます。

注意　　次に、IrisEvalW、IrisExecuteW、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisEvalW の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRISHW_ERSYSTEM	InterSystems IRIS が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に評価されました。

IrisEvalW は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
IRISWSTR retval;
IRISWSTR expr;

strcpy(expr.str, "\"Record\"_^Recnum_\" = \"_$$^GetRec(^Recnum)");
expr.len = strlen(expr.str);
rc = IrisEvalW(&expr);
if (rc == IRIS_SUCCESS)
    rc = IrisConvert(ING,&retval);
```

3.39 IrisExecuteA

バリアント：IrisExecuteW、IrisExecuteH

```
int IrisExecuteA(IRIS_ASTRP volatile cmd)
```

引数

cmd	IRIS_ASTR 変数のアドレス。

説明

ターミナルで入力されたかのように、コマンド文字列を実行します。

注意　　次に、IrisEvalA、IrisExecuteA、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisExecuteA の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に実行されました。

IrisExecuteA は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
IRIS_ASTR command;
sprintf(command.str,"set $namespace = \"USER\""); /* changes namespace */
command.len = strlen(command.str);
rc = IrisExecuteA(&command);
```

3.40 IrisExecuteH

バリアント：IrisExecuteA、IrisExecuteW

```
int IrisExecuteH(IRISHSTRP volatile cmd)
```

引数

cmd	IRIS_ASTR 変数のアドレス。

説明

ターミナルで入力されたかのように、コマンド文字列を実行します。

IrisExecuteH が正常に完了した場合、IrisType と IrisConvert への呼び出しが完了するようにフラグがセットされます。これらの関数は、IrisEvalH から返される項目の処理に使用されます。

注意　　次に、IrisEvalH、IrisExecuteH、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisExecuteH の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に実行されました。

IrisExecuteH は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
unsigned short zname[] = {'Z','N',' ','"','U','S','E','R','"'};
IRISHSTRP pcommand;
pcommand.str = zname;
pcommand.len = sizeof(zname) / sizeof(unsigned short);
rc = IrisExecuteH(pcommand);
```

3.41 IrisExecuteW

バリアント：IrisExecuteA、IrisExecuteH

```
int IrisExecuteW(IRISWSTRP volatile cmd)
```

引数

cmd	IRIS_ASTR 変数のアドレス。

説明

ターミナルで入力されたかのように、コマンド文字列を実行します。

IrisExecuteW が正常に完了した場合、IrisType と IrisConvert への呼び出しが完了するようにフラグがセットされます。これらの関数は、IrisEvalW から返される項目の処理に使用されます。

注意　　次に、IrisEvalW、IrisExecuteW、または IrisEnd を呼び出すと、既存の返り値が上書きされます。

IrisExecuteW の返り値

IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_NOCON	接続が確立されていません。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_SUCCESS	文字列は、正常に実行されました。

IrisExecuteW は、任意の InterSystems IRIS エラー・コードを返すこともできます。

例

```
int rc;
unsigned short zname[] = {'Z','N',' ','"','U','S','E','R','"'};
IRISWSTRP pcommand;
pcommand.str = zname;
pcommand.len = sizeof(zname) / sizeof(unsigned short);
rc = IrisExecuteW(pcommand);
```

3.42 IrisExecuteArgs

```
int IrisExecuteArgs(int numargs)
```

引数

numargs	スタックにプッシュされた引数の数。

説明

引数を指定してコマンド文字列を実行します。コマンド文字列を通常のプッシュ文字列関数で引数スタックにプッシュした後、引数をプッシュする必要があります。引数の数は 0 に設定できます。

IrisExecuteArgs の戻り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	操作は正常に終了しました。

3.43 IrisExStrKill

```
int IrisExStrKill(IRIS_EXSTRP obj)
```

引数

obj	文字列へのポインタ。

説明

EXSTR 文字列に関連付けられているストレージを解放します。

IrisExStrKill の返り値

IRIS_ERUNIMPLEMENTED	文字列が未定義です。
IRIS_SUCCESS	文字列のストレージが解放されました。

3.44 IrisExStrNew

バリアント：IrisExStrNewW、IrisExStrNewH

```
unsigned char * IrisExStrNew(IRIS_EXSTRP zstr, int size)
```

引数

zstr	IRIS_EXSTR 文字列記述子へのポインタ。
size	割り当てる 8 ビット文字の数

説明

文字列に対して要求されたストレージの容量を割り当て、EXSTR 構造に、構造の長さおよび構造の値フィールドへのポインタを埋め込みます。

IrisExStrNew の返り値

割り当てられた文字列へのポインタ、または、文字列が割り当てられていない場合は NULL を返します。

3.45 IrisExStrNewW

バリアント：IrisExStrNew、IrisExStrNewH

```
unsigned short * IrisExStrNewW(IRIS_EXSTRP zstr, int size)
```

引数

zstr	IRIS_EXSTR 文字列記述子へのポインタ。
size	割り当てる 2 バイト文字の数

説明

文字列に対して要求されたストレージの容量を割り当て、EXSTR 構造に、構造の長さおよび構造の値フィールドへのポインタを埋め込みます。

IrisExStrNewW の返り値

割り当てられた文字列へのポインタ、または、文字列が割り当てられていない場合は NULL を返します。

3.46 IrisExStrNewH

バリアント：IrisExStrNew、IrisExStrNewW

```
unsigned short * IrisExStrNewH(IRIS_EXSTRP zstr, int size)
```

引数

zstr	IRIS_EXSTR 文字列記述子へのポインタ。
size	割り当てる 4 バイト文字の数

説明

文字列に対して要求されたストレージの容量を割り当て、EXSTR 構造に、構造の長さおよび構造の値フィールドへのポインタを埋め込みます。

IrisExStrNewH の返り値

割り当てられた文字列へのポインタ、または、文字列が割り当てられていない場合は NULL を返します。

3.47 IrisExtFun

```
int IrisExtFun(unsigned int flags, int narg)
```

引数

flags	IrisPushFunc[XW] からのルーチン・フラグ。
narg	引数スタックにプッシュされた呼び出し引数の数。

説明

返り値を引数スタックにプッシュする外部関数呼び出しを実行します。

IrisExtFun の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_FAILURE	内部的な整合性にエラーがあります。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.48 IrisGetProperty

```
int IrisGetProperty()
```

説明

IrisPushProperty により定義されたプロパティの値を取得します。この値は、引数スタックにプッシュされます。

IrisGetProperty の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.49 IrisGlobalData

```
int IrisGlobalData(int narg, int valueflag)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。
valueflag	データ値がある場合、データ値を返すかどうかを示します。

説明

指定したグローバルに対して $Data を実行します。

IrisGlobalData の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.50 IrisGlobalGet

```
int IrisGlobalGet(int narg, int flag)
```

引数

narg	引数スタックにプッシュされた添え字式の数。
flag	グローバル参照が定義されていない場合の動作を指定します。 ・ 0 ― IRIS_ERUNDEF を返します。 ・ 1 ― IRIS_SUCCESS を返しますが、返り値は空の文字列になります。

説明

IrisPushGlobal および任意の添え字により定義されたグローバル参照の値を取得します。このノード値は、引数スタックにプッシュされます。

IrisGlobalGet の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.51 IrisGlobalGetBinary

```
int IrisGlobalGetBinary(int numsub, int flag, int *plen, Callin_char_t **pbuf)
```

引数

numsub	引数スタックにプッシュされた添え字式の数。
flag	グローバル参照が定義されていない場合の動作を指定します。 ・ 0 ― IRIS_ERUNDEF を返します。 ・ 1 ― IRIS_SUCCESS を返しますが、返り値は空の文字列になります。
plen	バッファの長さへのポインタ。
pbuf	バッファ・ポインタへのポインタ。

説明

IrisPushGlobal[W][H] および任意の添え字で定義されたグローバル参照の値を取得します。また、その結果が指定されたバッファに収まるバイナリ文字列かどうかを確認するためにテストを実行します。このノード値は、引数スタックにプッシュされます。

IrisGlobalGetBinary の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.52 IrisGlobalIncrement

```
int IrisGlobalIncrement(int narg)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。

説明

$INCREMENT を実行し、スタックの一番上の結果を返します。

IrisGlobalIncrement の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_ERMAXINCR	MAXINCREMENT システム・エラー
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.53 IrisGlobalKill

```
int IrisGlobalKill(int narg, int nodeonly)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。
nodeonly	1 の値は、指定されたノードのみを強制停止することを示します。値が 0 の場合、指定されたグローバル・ツリー全体が強制停止されます。

説明

グローバル・ノードまたはツリーに対して ZKILL を実行します。

IrisGlobalKill の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.54 IrisGlobalOrder

```
int IrisGlobalOrder(int narg, int dir, int valueflag)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。
dir	$Order の方向は、1 の場合は前向き、-1 は後ろ向きです。
valueflag	データ値がある場合、データ値を返すかどうかを示します。

説明

指定したグローバルに対して $Order を実行します。

IrisGlobalOrder の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.55 IrisGlobalQuery

```
int IrisGlobalQuery(int narg, int dir, int valueflag)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。
dir	$Query の方向は、1 の場合は前向き、-1 は後ろ向きです。
valueflag	データ値がある場合、データ値を返すかどうかを示します。

説明

指定したグローバルに対して $Query を実行します。

IrisGlobalQuery の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_ERPROTECT	保護違反です。
IRIS_ERUNDEF	ノードに関連付けられた値はありません。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.56 IrisGlobalRelease

```
int IrisGlobalRelease( )
```

説明

保持されたグローバル・バッファの所有権 (存在する場合) を解放します。

IrisGlobalRelease の返り値

IRIS_SUCCESS	操作は正常に終了しました。

3.57 IrisGlobalSet

```
int IrisGlobalSet(int narg)
```

引数

narg	引数スタックにプッシュされた添え字式の数。

説明

IrisPushGlobal および任意の添え字により定義されたグローバル参照の値を格納します。この呼び出しを行う前に、ノード値を引数スタックにプッシュしておく必要があります。

IrisGlobalSet の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.58 IrisIncrementCountOref

```
int IrisIncrementCountOref(unsigned int oref)
```

引数

oref	オブジェクト参照。

説明

OREF に対するシステム参照カウンタをインクリメントします。

IrisIncrementCountOref の返り値

IRIS_ERBADOREF	OREF が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.59 IrisInvokeClassMethod

```
int IrisInvokeClassMethod(int narg)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。

説明

IrisPushClassMethod[W] および任意の引数によって定義されたクラス・メソッド呼び出しを実行します。返り値は、引数スタックにプッシュされます。

IrisInvokeClassMethod の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.60 IrisInvokeMethod

```
int IrisInvokeMethod(int narg)
```

引数

narg	引数スタックにプッシュされた呼び出し引数の数。

説明

IrisPushMethod[W]、および引数スタックにプッシュされた任意の引数によって定義されるインスタンス・メソッド呼び出しを実行します。

IrisInvokeMethod の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.61 IrisOflush

```
int IrisOflush()
```

説明

保留中の出力をすべてフラッシュします。

IrisOflush の返り値

IRIS_FAILURE	(コールイン実行可能プログラムの内部からではなく) $ZF 関数から呼び出された場合に返されます。
IRIS_SUCCESS	制御関数が実行されました。

3.62 IrisPop

```
int IrisPop(void ** arg)
```

引数

arg	引数スタック・エントリへのポインタ。

説明

引数スタックから値をポップ・オフします。

IrisPop の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.63 IrisPopCvtW

バリアント：IrisPopCvtH

```
int IrisPopCvtW(int * lenp, unsigned short ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

非推奨：すべての文字列に Long 型の文字列関数 IrisPopExStrCvtW を使用する必要があります。

引数スタックからローカルの 8 ビット文字列をポップ・オフして、2 バイトの Unicode に変換します。Unicode 環境での IrisPopStrW と同じです。

IrisPopCvtW の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.64 IrisPopCvtH

バリアント：IrisPopCvtW

```
int IrisPopCvtH(int * lenp, wchar_t ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

引数スタックからローカルの 8 ビット文字列をポップ・オフして、4 バイトの Unicode に変換します。Unicode 環境での IrisPopStrH と同じです。

IrisPopCvtH の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.65 IrisPopDbl

```
int IrisPopDbl(double * nump)
```

引数

nump	ダブル値へのポインタ。

説明

引数スタックから値をポップ・オフし、double 型に変換します。

IrisPopDbl の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.66 IrisPopExStr

バリアント：IrisPopExStrW、IrisPopExStrH

```
int IrisPopExStr(IRIS_EXSTRP sstrp)
```

引数

sstrp	標準的な文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、ローカルの 8 ビット・エンコーディングの文字列に変換します。

IrisPopExStr の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。
IRIS_EXSTR_INUSE	sstrp が NULL に初期化されていなかった場合に返されます。

3.67 IrisPopExStrW

バリアント：IrisPopExStr、IrisPopExStrH

```
int IrisPopExStrW(IRIS_EXSTRP sstrp)
```

引数

sstrp	標準的な文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、2 バイト Unicode 文字列に変換します。

IrisPopExStrW の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
IRIS_EXSTR_INUSE	sstrp が NULL に初期化されていなかった場合に返されます。

3.68 IrisPopExStrH

バリアント：IrisPopExStr、IrisPopExStrW

```
int IrisPopExStrH(IRIS_EXSTRP sstrp)
```

引数

sstrp	標準的な文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、4 バイト Unicode 文字列に変換します。

IrisPopExStrH の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
IRIS_EXSTR_INUSE	sstrp が NULL に初期化されていなかった場合に返されます。

3.69 IrisPopExStrCvtW

バリアント：IrisPopExStrCvtH

```
int IrisPopExStrCvtW(IRIS_EXSTRP sstr)
```

引数

sstr	Long 型の文字列ポインタへのポインタ。

説明

引数スタックからローカルの 8 ビット文字列をポップ・オフして、2 バイトの Unicode 文字列に変換します。Unicode システムでは、これは IrisPopExStrW と同じです。

IrisPopExStrCvtW の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.70 IrisPopExStrCvtH

バリアント：IrisPopExStrCvtW

```
int IrisPopExStrCvtW(IRIS_EXSTRP sstr)
```

引数

sstr	Long 型の文字列ポインタへのポインタ。

説明

引数スタックからローカルの 8 ビット文字列をポップ・オフして、4 バイトの Unicode 文字列に変換します。Unicode システムでは、これは IrisPopExStrH と同じです。

IrisPopExStrCvtH の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.71 IrisPopInt

```
int IrisPopInt(int* nump)
```

引数

nump	整数値へのポインタ。

説明

引数スタックから値をポップ・オフし、integer 型に変換します。

IrisPopInt の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.72 IrisPopInt64

```
int IrisPopInt64(long long * nump)
```

引数

nump	long long 値へのポインタ。

説明

引数スタックから値をポップ・オフし、64 ビット (long long) 値に変換します

IrisPopInt64 の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.73 IrisPopList

```
int IrisPopList(int * lenp, Callin_char_t ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

引数スタックから $LIST オブジェクトをポップ・オフして変換します。

IrisPopList の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.74 IrisPopOref

```
int IrisPopOref(unsigned int * orefp)
```

引数

orefp	OREF 値へのポインタ。

説明

引数スタックから OREF をポップ・オフします。

IrisPopOref の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERNOOREF	結果は OREF ではありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.75 IrisPopPtr

```
int IrisPopPtr(void ** ptrp)
```

引数

ptrp	汎用ポインタへのポインタ。

説明

引数スタックから、ポインタを内部形式でポップ・オフします。

IrisPopPtr の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_BADARG	このエントリは、有効なポインタではありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.76 IrisPopStr

バリアント：IrisPopStrW、IrisPopStrH

```
int IrisPopStr(int * lenp, Callin_char_t ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、string 型に変換します。

IrisPopStr の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。

3.77 IrisPopStrW

バリアント：IrisPopStr、IrisPopStrH

```
int IrisPopStrW(int * lenp, unsigned short ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、2 バイトの Unicode 文字列に変換します。

IrisPopStrW の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.78 IrisPopStrH

バリアント：IrisPopStr、IrisPopStrW

```
int IrisPopStrH(int * lenp, wchar_t ** strp)
```

引数

lenp	文字列の長さへのポインタ。
strp	文字列ポインタへのポインタ。

説明

引数スタックから値をポップ・オフし、4 バイトの Unicode 文字列に変換します。

IrisPopStrH の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.79 IrisPromptA

バリアント：IrisPromptW、IrisPromptH

```
int IrisPromptA(IRIS_ASTRP rbuf)
```

引数

rbuf	プロンプト文字列。返される文字列の最小長は 5 文字です。

説明

(“>” のない) ターミナルである文字列を返します。

IrisPromptA の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	rbuf は、5 文字以上の長さにする必要があります。
IRIS_SUCCESS	接続が確立されました。

例

```
IRIS_ASTR retval;      /* define variable retval */
retval.len = 5;         /* maximum return length of string */
rc = IrisPromptA(&retval);
```

3.80 IrisPromptH

バリアント：IrisPromptA、IrisPromptW

```
int IrisPromptH(IRISHSTRP rbuf)
```

引数

rbuf	プロンプト文字列。返される文字列の最小長は 5 文字です。

説明

(“>” のない) ターミナルである文字列を返します。

IrisPromptH の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_FAILURE	要求は失敗しました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	rbuf は、5 文字以上の長さにする必要があります。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISHSTRP retval; /* define variable retval */
retval.len = 5; /* maximum return length of string */
rc = IrisPromptH( &retval);
```

3.81 IrisPromptW

バリアント：IrisPromptA、IrisPromptH

```
int IrisPromptW(IRISWSTRP rbuf)
```

引数

rbuf	プロンプト文字列。返される文字列の最小長は 5 文字です。

説明

(“>” のない) ターミナルである文字列を返します。

IrisPromptW の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_FAILURE	要求は失敗しました。
IRIS_NOCON	接続が確立されていません。
IRIS_RETTOOSMALL	rbuf は、5 文字以上の長さにする必要があります。
IRIS_SUCCESS	接続が確立されました。

例

```
IRISWSTR retval; /* define variable retval */
retval.len = 5; /* maximum return length of string */
rc = IrisConvertW( &retval);
```

3.82 IrisPushClassMethod

バリアント：IrisPushClassMethodW、IrisPushClassMethodH

```
int IrisPushClassMethod(int clen, const Callin_char_t * cptr,
                        int mlen, const Callin_char_t * mptr, int flg)
```

引数

clen	クラス名の長さ（文字数）。
cptr	クラス名へのポインタ。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

クラス・メソッド参照を引数スタックにプッシュします。

IrisPushClassMethod の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.83 IrisPushClassMethodH

バリアント：IrisPushClassMethod、IrisPushClassMethodW

```
int IrisPushClassMethodH(int clen, const wchar_t * cptr,
                         int mlen, const wchar_t * mptr, int flg)
```

引数

clen	クラス名の長さ（文字数）。
cptr	クラス名へのポインタ。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

4 バイト Unicode クラス・メソッド参照を引数スタックにプッシュします。

IrisPushClassMethodH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.84 IrisPushClassMethodW

バリアント：IrisPushClassMethod、IrisPushClassMethodH

```
int IrisPushClassMethodW(int clen, const unsigned short * cptr,
                         int mlen, const unsigned short * mptr, int flg)
```

引数

clen	クラス名の長さ（文字数）。
cptr	クラス名へのポインタ。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

2 バイト Unicode クラス・メソッド参照を引数スタックにプッシュします。

IrisPushClassMethodW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.85 IrisPushCvtW

バリアント：IrisPushCvtH

```
int IrisPushCvtW(int len, const unsigned short * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

非推奨：すべての文字列に Long 型の文字列関数 IrisPushExStrCvtW を使用する必要があります。

Unicode 文字列をローカルの 8 ビットに変換して、引数スタックにプッシュします。Unicode バージョンの IrisPushStrW と同じです。

IrisPushCvtW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	文字列の変換によるものです。

3.86 IrisPushCvtH

バリアント：IrisPushCvtW

```
int IrisPushCvtH(int len, const wchar_t * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

Unicode 文字列をローカルの 8 ビットに変換して、引数スタックにプッシュします。Unicode バージョンの IrisPushStrH と同じです。

IrisPushCvtH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	文字列の変換によるものです。

3.87 IrisPushDbl

```
int IrisPushDbl(double num)
```

引数

num	double 値。

説明

double 型を引数スタックにプッシュします。

IrisPushDbl の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.88 IrisPushExecuteFuncA

バリアント：IrisPushExecuteFuncW、IrisPushExecuteFuncH

```
int IrisPushExecuteFuncA(int len, const unsigned char *ptr)
```

引数

len	コマンド文字列の長さ
ptr	コマンド文字列へのポインタ

説明

IrisCallExecuteFunc() による呼び出しの準備として $Xecute() コマンドを引数スタックにプッシュします。

IrisPushExecuteFuncA の戻り値

IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.89 IrisPushExecuteFuncW

バリアント：IrisPushExecuteFuncA、IrisPushExecuteFuncH

```
int IrisPushExecuteFuncW(int len, const unsigned short *ptr)
```

引数

len	コマンド文字列の長さ
ptr	コマンド文字列へのポインタ

説明

IrisCallExecuteFunc() による呼び出しの準備として $Xecute() コマンドを引数スタックにプッシュします。

IrisPushExecuteFuncW の戻り値

IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.90 IrisPushExecuteFuncH

バリアント：IrisPushExecuteFuncA、IrisPushExecuteFuncW

```
int IrisPushExecuteFuncH(int len, const wchar_t *ptr)
```

引数

len	コマンド文字列の長さ
ptr	コマンド文字列へのポインタ

説明

IrisCallExecuteFunc() による呼び出しの準備として $Xecute() コマンドを引数スタックにプッシュします。

IrisPushExecuteFuncH の戻り値

IRIS_STRTOOLONG	文字列が長すぎます。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.91 IrisPushExStr

バリアント：IrisPushExStrW、IrisPushExStrH

```
int IrisPushExStr(IRIS_EXSTRP sptr)
```

引数

sptr	引数値へのポインタ。

説明

文字列を引数スタックにプッシュします。

IrisPushExStr の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.92 IrisPushExStrW

バリアント：IrisPushExStr、IrisPushExStrH

```
int IrisPushExStrW(IRIS_EXSTRP sptr)
```

引数

sptr	引数値へのポインタ。

説明

Unicode 文字列を引数スタックにプッシュします。

IrisPushExStrW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.93 IrisPushExStrH

バリアント：IrisPushExStr、IrisPushExStrW

```
int IrisPushExStrH(IRIS_EXSTRP sptr)
```

引数

sptr	引数値へのポインタ。

説明

4 バイト Unicode 文字列を引数スタックにプッシュします。

IrisPushExStrH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.94 IrisPushExStrCvtW

バリアント：IrisPushExStrCvtH

```
int IrisPushExStrCvtW(IRIS_EXSTRP sptr)
```

引数

sptr	引数値へのポインタ。

説明

Unicode 文字列をローカルの 8 ビットに変換して、引数スタックにプッシュします。

IrisPushExStrCvtW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	文字列の変換によるものです。

3.95 IrisPushExStrCvtH

バリアント：IrisPushExStrCvtW

```
int IrisPushExStrCvtH(IRIS_EXSTRP sptr)
```

引数

sptr	引数値へのポインタ。

説明

4 バイトの Unicode 文字列をローカルの 8 ビットに変換して、引数スタックにプッシュします。

IrisPushExStrCvtH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	文字列の変換によるものです。

3.96 IrisPushFunc

バリアント：IrisPushFuncW、IrisPushFuncH

```
int IrisPushFunc(unsigned int * rflag, int tlen, const Callin_char_t * tptr,
                 int nlen, const Callin_char_t * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL (″″) であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tagptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL (″″) で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

外部関数参照を引数スタックにプッシュします。

IrisPushFunc の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.97 IrisPushFuncH

バリアント：IrisPushFunc、IrisPushFuncW

```
int IrisPushFuncH(unsigned int * rflag, int tlen, const wchar_t * tptr,
                  int nlen, const wchar_t * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL ("") であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、`(void *) 0` がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL ("") で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、`(void *) 0` がポインタ値として使用される場合があります。

説明

4 バイト Unicode 外部関数参照を引数スタックにプッシュします。

IrisPushFuncH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.98 IrisPushFuncW

バリアント：IrisPushFunc、IrisPushFuncH

```
int IrisPushFuncW(unsigned int * rflag, int tlen, const unsigned short * tptr,
                  int nlen, const unsigned short * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ（文字数）。ここで、0 はタグ名が NULL（""）であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ（文字数）。ここで、0 はルーチン名が NULL（""）で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

2 バイト Unicode 外部関数参照を引数スタックにプッシュします。

IrisPushFuncW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.99 IrisPushFuncX

バリアント：IrisPushFuncXW、IrisPushFuncXH

```
int IrisPushFuncX(unsigned int * rflag, int tlen, const Callin_char_t * tptr, int off,
                  int elen, const Callin_char_t * eptr,
                  int nlen, const Callin_char_t * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL ("") であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ (文字数)。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL ("") で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

拡張外部関数参照を引数スタックにプッシュします。

IrisPushFuncX の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.100 IrisPushFuncXH

バリアント：IrisPushFuncX、IrisPushFuncXW

```
int IrisPushFuncXH(unsigned int * rflag, int tlen, const wchar_t * tptr, int off,
                   int elen, const wchar_t * eptr, int nlen, const wchar_t * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL (″″) であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ (文字数)。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL (″″) で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

4 バイト Unicode 拡張関数ルーチン参照を引数スタックにプッシュします。

IrisPushFuncXH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.101 IrisPushFuncXW

バリアント：IrisPushFuncX、IrisPushFuncXH

```
int IrisPushFuncXW(unsigned int * rflag, int tlen, const unsigned short * tptr, int off,
                   int elen, const unsigned short * eptr,
                   int nlen, const unsigned short * nptr)
```

引数

rflag	IrisExtFun で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL (″″) であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(`void *) 0` がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ (文字数)。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(`void *) 0` がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL (″″) で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(`void *) 0` がポインタ値として使用される場合があります。

説明

2 バイト Unicode 拡張関数ルーチン参照を引数スタックにプッシュします。

IrisPushFuncXW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.102 IrisPushGlobal

バリアント：IrisPushGlobalW、IrisPushGlobalH

```
int IrisPushGlobal(int nlen, const Callin_char_t * nptr)
```

引数

nlen	グローバル名の長さ (文字数)。
nptr	グローバル名へのポインタ。

説明

グローバル参照を引数スタックにプッシュします。

IrisPushGlobal の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.103 IrisPushGlobalH

バリアント：IrisPushGlobal、IrisPushGlobalW

```
intIrisPushGlobalH(int nlen, const wchar_t * nptr)
```

引数

nlen	グローバル名の長さ（文字数）。
nptr	グローバル名へのポインタ。

説明

4 バイト Unicode グローバル参照を引数スタックにプッシュします。

IrisPushGlobalH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.104 IrisPushGlobalW

バリアント：IrisPushGlobal、IrisPushGlobalH

```
int IrisPushGlobalW(int nlen, const unsigned short * nptr)
```

引数

nlen	グローバル名の長さ（文字数）。
nptr	グローバル名へのポインタ。

説明

2 バイト Unicode グローバル参照を引数スタックにプッシュします。

IrisPushGlobalW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.105 IrisPushGlobalX

バリアント：IrisPushGlobalXW、IrisPushGlobalXH

```
int IrisPushGlobalX(int nlen, const Callin_char_t * nptr,
                    int elen, const Callin_char_t * eptr)
```

引数

nlen	グローバル名の長さ（文字数）。
nptr	グローバル名へのポインタ。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、`(void *) 0` がポインタ値として使用される場合があります。

説明

拡張グローバル参照を引数スタックにプッシュします。

IrisPushGlobalX の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.106 IrisPushGlobalXH

バリアント：IrisPushGlobalX、IrisPushGlobalXW

```
int IrisPushGlobalXH(int nlen, const wchar_t * nptr, int elen, const wchar_t * eptr)
```

引数

nlen	グローバル名の長さ（文字数）。
nptr	グローバル名へのポインタ。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、`(void *) 0` がポインタ値として使用される場合があります。

説明

4 バイト Unicode 拡張グローバル参照を引数スタックにプッシュします。

IrisPushGlobalXH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTAC	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.107 IrisPushGlobalXW

バリアント：IrisPushGlobalX、IrisPushGlobalXH

```
int IrisPushGlobalXW(int nlen, const unsigned short * nptr,
                     int elen, const unsigned short * eptr)
```

引数

nlen	グローバル名の長さ（文字数）。
nptr	グローバル名へのポインタ。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、`(void *) 0` がポインタ値として使用される場合があります。

説明

2 バイト Unicode 拡張グローバル参照を引数スタックにプッシュします。

IrisPushGlobalXW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTAC	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.108 IrisPushIEEEDbl

```
int IrisPushIEEEDbl(double num)
```

引数

num	double 値。

説明

IEEE の double 型を引数スタックにプッシュします

IrisPushIEEEDbl の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.109 IrisPushInt

```
int IrisPushInt(int num)
```

引数

num	整数値。

説明

integer 型を引数スタックにプッシュします

IrisPushInt の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.110 IrisPushInt64

```
int IrisPushInt64(long long num)
```

引数

num	long long 値。

説明

64 ビット (long long) 値を引数スタックにプッシュします。

IrisPushInt64 の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.111 IrisPushList

```
int IrisPushList(int len, const Callin_char_t * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

$LIST オブジェクトを変換し、引数スタックにプッシュします。

IrisPushList の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	文字列要素の変換によります。

3.112 IrisPushLock

バリアント：IrisPushLockW、IrisPushLockH

```
int IrisPushLock(int nlen, const Callin_char_t * nptr)
```

引数

nlen	ロック名の長さ（バイト単位）
nptr	ロック名へのポインタ。

説明

ロック名を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLock の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.113 IrisPushLockH

バリアント：IrisPushLock、IrisPushLockW

```
int IrisPushLockH(int nlen, const wchar_t * nptr)
```

引数

nlen	ロック名の長さ (2 バイトまたは 4 バイト文字の数)
nptr	ロック名へのポインタ。

説明

ロック名を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLockH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.114 IrisPushLockW

バリアント：IrisPushLock、IrisPushLockH

```
int IrisPushLockW(int nlen, const unsigned short * nptr)
```

引数

nlen	ロック名の長さ（2 バイト文字の数）
nptr	ロック名へのポインタ。

説明

ロック名を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLockW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.115 IrisPushLockX

バリアント：IrisPushLockXW、IrisPushLockXH

```
int IrisPushLockX(int nlen, const Callin_char_t * nptr, int elen, const Callin_char_t * eptr)
```

引数

nlen	ロック名の長さ（8 ビット文字の数）
nptr	ロック名へのポインタ。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。名前の形式は < >^[< >]^< > にする必要があります。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

ロック名と環境文字列を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLockX の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.116 IrisPushLockXH

バリアント：IrisPushLockX、IrisPushLockXW

```
int IrisPushLockXH(int nlen, const wchar_t * nptr, int elen, const wchar_t * eptr)
```

引数

nlen	ロック名の長さ (2 バイトまたは 4 バイト文字の数)
nptr	ロック名へのポインタ。
elen	環境名の長さ (文字数)。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。名前の形式は < >^[< >]^< > にする必要があります。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

ロック名と環境文字列を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLockXH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.117 IrisPushLockXW

バリアント：IrisPushLockX、IrisPushLockXH

```
int IrisPushLockXW(int nlen, const unsigned short * nptr, int elen, const unsigned short * eptr)
```

引数

nlen	ロック名の長さ（2 バイト文字の数）
nptr	ロック名へのポインタ。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。名前の形式は < >^[< >]^< > にする必要があります。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

ロック名と環境文字列を引数スタックにプッシュすることにより、IrisAcquireLock コマンドを初期化します。

IrisPushLockXW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.118 IrisPushMethod

バリアント：IrisPushMethodW、IrisPushMethodH

```
int IrisPushMethod(unsigned int oref, int mlen, const Callin_char_t * mptr, int flg)
```

引数

oref	オブジェクト参照。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

インスタンス・メソッド参照を引数スタックにプッシュします。

IrisPushMethod の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.119 IrisPushMethodH

バリアント：IrisPushMethod、IrisPushMethodW

```
int IrisPushMethodH(unsigned int oref, int mlen, const wchar_t * mptr, int flg)
```

引数

oref	オブジェクト参照。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

4 バイト Unicode インスタンス・メソッド参照を引数スタックにプッシュします。

IrisPushMethodH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.120 IrisPushMethodW

バリアント：IrisPushMethod、IrisPushMethodH

```
int IrisPushMethodW(unsigned int oref, int mlen, const unsigned short * mptr, int flg)
```

引数

oref	オブジェクト参照。
mlen	メソッド名の長さ（文字数）。
mptr	メソッド名へのポインタ。
flg	メソッドが値を返すかどうかを指定します。メソッドが値を返す場合、値を取得するにはこのフラグを 1 に設定する必要があります。このメソッドは、引数付きの Quit を使用して値を返す必要があります。値を返さない場合は、このパラメータを 0 に設定します。

説明

2 バイト Unicode インスタンス・メソッド参照を引数スタックにプッシュします。

IrisPushMethodW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.121 IrisPushOref

```
int IrisPushOref(unsigned int oref)
```

引数

oref	オブジェクト参照。

説明

OREF を引数スタックにプッシュします。

IrisPushOref の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERBADOREF	OREF が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.122 IrisPushProperty

バリアント：IrisPushPropertyW、IrisPushPropertyH

```
int IrisPushProperty(unsigned int oref, int plen, const Callin_char_t * pptr)
```

引数

oref	オブジェクト参照。
plen	プロパティ名の長さ（文字数）。
pptr	プロパティ名へのポインタ。

説明

プロパティ参照を引数スタックにプッシュします。

IrisPushProperty の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。

3.123 IrisPushPropertyH

バリアント：IrisPushProperty、IrisPushPropertyW

```
int IrisPushPropertyH(unsigned int oref, int plen, const wchar_t * pptr)
```

引数

oref	オブジェクト参照。
plen	プロパティ名の長さ（文字数）。
pptr	プロパティ名へのポインタ。

説明

4 バイト Unicode プロパティ参照を引数スタックにプッシュします。

IrisPushPropertyH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.124 IrisPushPropertyW

バリアント：IrisPushProperty、IrisPushPropertyH

```
int IrisPushPropertyW(unsigned int oref, int plen, const unsigned short * pptr)
```

引数

oref	オブジェクト参照。
plen	プロパティ名の長さ（文字数）。
pptr	プロパティ名へのポインタ。

説明

2 バイト Unicode プロパティ参照を引数スタックにプッシュします。

IrisPushPropertyW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_BADARG	呼び出し引数が不正です。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.125 IrisPushPtr

```
int IrisPushPtr(void * ptr)
```

引数

ptr	汎用ポインタ。

説明

ポインタを内部形式で引数スタックにプッシュします。

IrisPushPtr の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.126 IrisPushRtn

バリアント：IrisPushRtnW、IrisPushRtnH

```
int IrisPushRtn(unsigned int * rflag, int tlen, const Callin_char_t * tptr,
                int nlen, const Callin_char_t * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL ("") であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL ("") で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

ルーチン参照を引数スタックにプッシュします。すべての引数を受け付けるバージョンについては、"IrisPushRtnX" を参照してください。これは、タグ名とルーチン名のみを受け付ける短い形式の関数です。

IrisPushRtn の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.127 IrisPushRtnH

バリアント：IrisPushRtn、IrisPushRtnW

```
int IrisPushRtnH(unsigned int * rflag, int tlen, const wchar_t * tptr,
                 int nlen, const wchar_t * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL ("") であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL ("") で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

4 バイト Unicode ルーチン参照を引数スタックにプッシュします。すべての引数を受け付けるバージョンについては、"IrisPushRtnXH" を参照してください。これは、タグ名とルーチン名のみを受け付ける短い形式の関数です。

IrisPushRtnH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.128 IrisPushRtnW

バリアント：IrisPushRtn、IrisPushRtnH

```
int IrisPushRtnW(unsigned int * rflag, int tlen, const unsigned short * tptr,
                 int nlen, const unsigned short * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ (文字数)。ここで、0 はタグ名が NULL ("") であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ (文字数)。ここで、0 はルーチン名が NULL ("") で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

2 バイト Unicode ルーチン参照を引数スタックにプッシュします。すべての引数を受け付けるバージョンについては、"IrisPushRtnXW" を参照してください。これは、タグ名とルーチン名のみを受け付ける短い形式の関数です。

IrisPushRtnW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.129 IrisPushRtnX

バリアント：IrisPushRtnXW、IrisPushRtnXH

```
int IrisPushRtnX(unsigned int * rflag, int tlen, const Callin_char_t * tptr,
                 int off, int elen, const Callin_char_t * eptr,
                 int nlen, const Callin_char_t * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ（文字数）。ここで、0 はタグ名が NULL（""）であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ（文字数）。ここで、0 はルーチン名が NULL（""）で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

拡張ルーチン参照を引数スタックにプッシュします。タグ名とルーチン名のみを受け付ける短い形式については、"IrisPushRtn" を参照してください。

IrisPushRtnX の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.130 IrisPushRtnXH

バリアント：IrisPushRtnX、IrisPushRtnXW

```
int IrisPushRtnXH(unsigned int * rflag, int tlen, const wchar_t * tptr,
                  int off, int elen, const wchar_t * eptr,
                  int nlen, const wchar_t * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ（文字数）。ここで、0 はタグ名が NULL（""）であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ（文字数）。ここで、0 はルーチン名が NULL（""）で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

4 バイト Unicode 拡張ルーチン参照を引数スタックにプッシュします。タグ名とルーチン名のみを受け付ける短い形式については、"IrisPushRtnH" を参照してください。

IrisPushRtnXH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.131 IrisPushRtnXW

バリアント：IrisPushRtnX、IrisPushRtnXH

```
int IrisPushRtnXW(unsigned int * rflag, int tlen, const unsigned short * tptr,
                  int off, int elen, const unsigned short * eptr,
                  int nlen, const unsigned short * nptr)
```

引数

rflag	IrisDoRtn で使用するためのルーチン・フラグ。
tlen	タグ名の長さ（文字数）。ここで、0 はタグ名が NULL（""）であることを表します。
tptr	タグ名へのポインタ。tlen == 0 の場合、tptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
off	指定されたタグからの行オフセット。ここで、0 はオフセットがないことを表します。
elen	環境名の長さ（文字数）。ここで、0 は指定された環境がなく、この関数では現在の環境が使用されることを表します。
eptr	環境名へのポインタ。elen == 0 の場合、eptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。
nlen	ルーチン名の長さ（文字数）。ここで、0 はルーチン名が NULL（""）で、現在のルーチン名が使用されることを表します。
nptr	ルーチン名へのポインタ。nlen == 0 の場合、nptr は使用されず、(void *) 0 がポインタ値として使用される場合があります。

説明

2 バイト Unicode 拡張ルーチン参照を引数スタックにプッシュします。タグ名とルーチン名のみを受け付ける短い形式については、"IrisPushRtnW" を参照してください。

IrisPushRtnXW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.132 IrisPushStr

バリアント：IrisPushStrW、IrisPushStrH

```
int IrisPushStr(int len, const Callin_char_t * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

バイト文字列を引数スタックにプッシュします。

IrisPushStr の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.133 IrisPushStrW

バリアント：IrisPushStr、IrisPushStrH

```
int IrisPushStrW(int len, const unsigned short * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

2 バイトの Unicode 文字列を引数スタックにプッシュします。

IrisPushStrW の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.134 IrisPushStrH

バリアント：IrisPushStr、IrisPushStrW

```
int IrisPushStrH(int len, const wchar_t * ptr)
```

引数

len	文字列内の文字数。
ptr	文字列へのポインタ。

説明

4 バイトの Unicode 文字列を引数スタックにプッシュします。

IrisPushStrH の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	InterSystems IRIS エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.135 IrisPushUndef

```
int IrisPushUndef()
```

説明

未定義値を引数スタックにプッシュします。この値は省略された関数引数として解釈されます。

IrisPushUndef の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。

3.136 IrisReleaseAllLocks

```
int IrisReleaseAllLocks( )
```

説明

引数なしの InterSystems IRIS LOCK コマンドを実行して、プロセスによって現在保持されているロックをすべて削除します。

IrisReleaseAllLocks の返り値

IRIS_SUCCESS	操作は正常に終了しました。

3.137 IrisReleaseLock

```
int IrisReleaseLock(int nsub, int flg)
```

引数

nsub	ロック参照にある添え字の数。
flg	ロック・コマンドの修飾語。有効な値は、IRIS_IMMEDIATE_RELEASE と IRIS_SHARED_LOCK のいずれか、または両方です。

説明

InterSystems IRIS LOCK コマンドを実行して、指定したロック名のロック・カウントをデクリメントします。このコマンドは、一度に 1 つずつ増分ロックを解放するだけです。

IrisReleaseLock の返り値

IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_SUCCESS	ロックに成功しました。

3.138 IrisSecureStartA

バリアント：IrisSecureStartW、IrisSecureStartH

```
int IrisSecureStartA(IRIS_ASTRP username, IRIS_ASTRP password, IRIS_ASTRP exename,
                     unsigned long flags, int tout, IRIS_ASTRP prinp, IRIS_ASTRP prout)
```

引数

username	認証するユーザ名です。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
password	認証するパスワードです。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
exename	コールイン実行可能プログラム名（または他のプロセス識別子です）。このユーザ定義文字列は、JOBEXAM および監査記録に表示されます。NULL は有効な値です。
flags	1 つまたは複数のターミナル設定を以下に示します。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列（prinp.len == 0）はプロセスの標準入力デバイスの使用を意味しています。NULL ポインタ（`((void *) 0)`）は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列（prout.len == 0）はプロセスの標準出力デバイスの使用を意味しています。NULL ポインタ（`((void *) 0)`）は NULL デバイスの使用を意味しています。

説明

プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス（prinp）、および出力デバイス（prout）が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ― InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります（既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS により実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます）。
- IRIS_TTALL ― 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。

- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisSecureStartA の返り値

IRIS_ACCESSDENIED	認証が失敗しました。実際の認証エラーについて、監査ログを確認してください。
IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisSecureStartH を呼び出した場合に返されます。
IRIS_CHANGEPASSWORD	パスワードの変更が必要です。この返り値は、InterSystems 認証を使用している場合にのみ返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEnd は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

3.139 IrisSecureStartH

バリアント：IrisSecureStartA、IrisSecureStartW

```
int IrisSecureStartH(IRISHSTRP username, IRISHSTRP password, IRISHSTRP exename,
                     unsigned long flags, int tout, IRISHSTRP prinp, IRISHSTRP prout)
```

引数

username	認証するユーザ名です。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
password	認証するパスワードです。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
exename	コールイン実行可能プログラム名 (または他のプロセス識別子です)。このユーザ定義文字列は、JOBEXAM および監査記録に表示されます。NULL は有効な値です。
flags	1 つまたは複数のターミナル設定を以下に示します。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列 (prinp.len == 0) はプロセスの標準入力デバイスの使用を意味しています。NULL ポインタ (`((void *) 0)`) は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列 (prout.len == 0) はプロセスの標準出力デバイスの使用を意味しています。NULL ポインタ (`((void *) 0)`) は NULL デバイスの使用を意味しています。

説明

プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス (prinp)、および出力デバイス (prout) が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ー InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります (既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS により実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます)。
- IRIS_TTALL ー 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。
- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisSecureStartH の返り値

IRIS_ACCESSDENIED	認証が失敗しました。実際の認証エラーについて、監査ログを確認してください。
IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisSecureStartH を呼び出した場合に返されます。
IRIS_CHANGEPASSWORD	パスワードの変更が必要です。この返り値は、InterSystems 認証を使用している場合にのみ返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEnd は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

3.140 IrisSecureStartW

バリアント：IrisSecureStartA、IrisSecureStartH

```
int IrisSecureStartW(IRISWSTRP username, IRISWSTRP password, IRISWSTRP exename,
                     unsigned long flags, int tout, IRISWSTRP prinp, IRISWSTRP prout)
```

引数

username	認証するユーザ名です。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
password	認証するパスワードです。UnknownUser 認証または OS 認証として認証するには NULL を使用し、それ以外の場合は Kerberos 資格情報キャッシュの情報を使用します。
exename	コールイン実行可能プログラム名 (または他のプロセス識別子です)。このユーザ定義文字列は、JOBEXAM および監査記録に表示されます。NULL は有効な値です。
flags	1 つまたは複数のターミナル設定を以下に示します。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列 (prinp.len == 0) はプロセスの標準入力デバイスの使用を意味しています。NULL ポインタ ((void *) 0) は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列 (prout.len == 0) はプロセスの標準出力デバイスの使用を意味しています。NULL ポインタ ((void *) 0) は NULL デバイスの使用を意味しています。

説明

プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス(prinp)、および出力デバイス(prout)が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ー InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります(既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS により実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます)。
- IRIS_TTALL ー 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。
- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisSecureStartW の返り値

IRIS_ACCESSDENIED	認証が失敗しました。実際の認証エラーについて、監査ログを確認してください。
IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisSecureStartH を呼び出した場合に返されます。
IRIS_CHANGEPASSWORD	パスワードの変更が必要です。この返り値は、InterSystems 認証を使用している場合にのみ返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEnd は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

3.141 IrisSetDir

```
int IrisSetDir(char * dir)
```

引数

dir	ディレクトリ名文字列へのポインタ。

説明

実行時に、マネージャのディレクトリ (IrisSys¥Mgr) の名前を動的に設定します。Windows では、InterSystems IRIS の共有ライブラリ・バージョンで、インストールのマネージャ・ディレクトリを識別するためにこの関数を使用する必要があります。

IrisSetDir の返り値

IRIS_FAILURE	(コールイン実行可能プログラムの内部からではなく) $ZF 関数から呼び出された場合に返されます。
IRIS_SUCCESS	制御関数が実行されました。

3.142 IrisSetProperty

```
int IrisSetProperty( )
```

説明

IrisPushProperty により定義されたプロパティの値を格納します。この呼び出しを行う前に、値を引数スタックにプッシュしておく必要があります。

IrisSetProperty の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_SUCCESS	操作は正常に終了しました。

3.143 IrisSignal

```
int IrisSignal(int signal)
```

引数

signal	オペレーティング・システムの信号値です。

説明

ユーザ・プログラムにより補足された信号を InterSystems IRIS に渡します。

この関数は、IrisAbort に非常によく似ていますが、適切と思われる任意のアクションに対して、接続のスレッドまたはユーザ側から、InterSystems IRIS 側に、任意の既知の信号値を渡すことができます。例えば、InterSystems IRIS に対するユーザ定義の信号ハンドラで傍受された信号を渡すために使用できます。

例

```
rc = IrisSignal(CTRL_C_EVENT); // Windows response to Ctrl-C
rc = IrisSignal(CTRL_C_EVENT); // UNIX response to Ctrl-C
```

IrisSignal の返り値

IRIS_CONBROKEN	接続は切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_NOTINIRIS	現在コールイン・パートナーが InterSystems IRIS に存在しません。
IRIS_SUCCESS	接続が確立されました。

3.144 IrisSPCReceive

```
int IrisSPCReceive(int * lenp, Callin_char_t * ptr)
```

引数

lenp	受信する最大長です。返されるときに、実際に受信されたバイト数を示すように修正されます。
ptr	メッセージを受信するバッファへのポインタです。lenp バイト以上にする必要があります。

説明

シングル・プロセス通信メッセージを受信します。現在のデバイスは、SPC モードで開かれた TCP でなければなりません。そうでない場合、IRIS_ERFUNCTION が返されます。

IrisSPCReceive の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERFUNCTION	現在のデバイスが TCP デバイスでないか、接続されていません。
IRIS_SUCCESS	操作は正常に終了しました。

3.145 IrisSPCSend

```
int IrisSPCSend(int len, const Callin_char_t * ptr)
```

引数

len	メッセージの長さ（単位はバイト）
ptr	メッセージを含む文字列へのポインタ。

説明

シングル・プロセス通信メッセージを送信します。現在のデバイスは、SPC モードで開かれた TCP でなければなりません。そうでない場合、IRIS_ERFUNCTION が返されます。

IrisSPCSend の返り値

IRIS_CONBROKEN	接続は、深刻なエラーによって切断されました。
IRIS_NOCON	接続が確立されていません。
IRIS_ERSYSTEM	データベース・エンジンが <SYSTEM> エラーを生成した、またはコールインが内部データの不整合を検出しました。
IRIS_ERFUNCTION	現在のデバイスが TCP デバイスでないか、接続されていません。
IRIS_ERARGSTACK	引数スタックがオーバーフローしています。
IRIS_ERSTRINGSTACK	文字列スタックがオーバーフローしています。
IRIS_SUCCESS	操作は正常に終了しました。
任意の InterSystems IRIS エラー	名前の変換によります。

3.146 IrisStartA

バリアント：IrisStartW、IrisStartH

```
int IrisStartA(unsigned long flags, int tout, IRIS_ASTRP prinp, IRIS_ASTRP prout)
```

引数

flags	以下に説明する値の 1 つまたは複数を取ります。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列 (prinp.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列 (prout.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。

説明

InterSystems IRIS プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス (prinp)、および出力デバイス (prout) が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ― InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります (既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS によ

り実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます)。

- IRIS_TTALL ー 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。
- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisStartA の返り値

IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisStartA を呼び出した場合に返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEndA は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

例

InterSystems IRIS プロセスが開始されます。ターミナルは、各インタフェースのコールイン関数の後にリセットされます。20 秒以内にパーティションが割り当てられない場合、起動は失敗します。ファイル dobackup は、入力用に使用します。このファイルには、InterSystems IRIS バックアップ用の ObjectScript スクリプトが含まれます。出力は、ターミナルに表示されます。

```
IRIS_ASTR inpdev;
IRIS_ASTR outdev;
int rc;

strcpy(inpdev.str, "[BATCHDIR]dobackup");
inpdev.len = strlen(inpdev.str);
strcpy(outdev.str,"");
outdev.len = strlen(outdev.str);
rc = IrisStartA(IRIS_TTALL|IRIS_TTNOUSE,0,inpdev,outdev);
```

3.147 IrisStartH

バリアント：IrisStartA、IrisStartW

```
int IrisStartH(unsigned long flags,int tout,IRISHSTRP prinp,IRISHSTRP prout)
```

引数

flags	以下に説明する値の 1 つまたは複数を取ります。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列 (prinp.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列 (prout.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。

説明

InterSystems IRIS プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス (prinp)、および出力デバイス (prout) が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ー InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります (既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS により実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます)。
- IRIS_TTALL ー 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。
- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisStartH の返り値

IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisStartH を呼び出した場合に返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEndH は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

例

InterSystems IRIS プロセスが開始されます。ターミナルは、各インタフェースのコールイン関数の後にリセットされます。20 秒以内にパーティションが割り当てられない場合、起動は失敗します。ファイル dobackup は、入力用に使用します。このファイルには、InterSystems IRIS バックアップ用の ObjectScript スクリプトが含まれます。出力は、ターミナルに表示されます。

```
inpdev;
outdev;
int rc;

strcpy(inpdev.str, "[BATCHDIR]dobackup");
inpdev.len = strlen(inpdev.str);
strcpy(outdev.str,"");
outdev.len = strlen(outdev.str);
rc = IrisStartH(IRIS_TTALL|IRIS_TTNOUSE,0,inpdev,outdev);
```

3.148 IrisStartW

バリアント：IrisStartA、IrisStartH

```
int IrisStartW(unsigned long flags,int tout,IRISWSTRP prinp,IRISWSTRP prout)
```

引数

flags	以下に説明する値の 1 つまたは複数を取ります。
tout	秒単位で指定されたタイムアウト値です。既定値は 0 です。0 を指定すると、タイムアウトにはなりません。タイムアウトは、パーティションが使用可能になるまで待機する場合にのみ適用されます。パーティションの初期化、内部リソースの待機、主入出力デバイスを開く処理などには適用されません。
prinp	InterSystems IRIS の主入力デバイスを定義する文字列です。空の文字列 (prinp.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。
prout	InterSystems IRIS の主出力デバイスを定義する文字列です。空の文字列 (prout.len == 0) はプロセスの主デバイスの使用を意味しています。NULL ポインタ (`(void *) 0`) は NULL デバイスの使用を意味しています。

説明

InterSystems IRIS プロセスをセットアップするために、InterSystems IRIS を呼び出します。

このコマンドを実行すると、最初の入出力操作を待たずに、入力デバイス(prinp)、および出力デバイス(prout)が開きます。これに対し、接続を開始した場合は通常、主入力デバイスまたは主出力デバイスは、最初に使用されるまで開きません。

flags 変数の有効な値は以下のとおりです。

- IRIS_PROGMODE ー InterSystems IRIS により、接続は、アプリケーション・モードではなく、プログラマ・モードの接続として扱われます。つまり、呼び出し側関数に対して明確なエラーが報告され、接続は有効なままになります(既定では、コールイン接続は、アプリケーション・モードでのルーチンの実行のようなものです。InterSystems IRIS により実行時のエラーが検知されると、接続が閉じられ、現在の操作に対してエラー IRIS_CONBROKEN が返されるだけでなく、それ以降に、新しい接続を確立せずにコールインを使用しようとしても、同じエラーが返されます)。
- IRIS_TTALL ー 既定値。InterSystems IRIS はターミナルの設定を初期化し、インタフェースを呼び出して戻るたびに、その設定をリストアする必要があります。
- IRIS_TTCALLIN ー InterSystems IRIS は、ターミナルが呼び出されるたびにそのターミナルを初期化しますが、ターミナルのリストアは IrisEnd が呼び出されたときまたは接続が切断されたときにのみ実行します。
- IRIS_TTSTART ー InterSystems IRIS は、接続の確立時にターミナルを初期化し、切断時にターミナルをリセットする必要があります。
- IRIS_TTNEVER ー InterSystems IRIS は、ターミナルの設定を変更できません。
- IRIS_TTNONE ー InterSystems IRIS は、主入出力デバイスからの入出力はできません。これは主入力、および主出力に対して NULL デバイスを指定することと同等です。主入力からの Read コマンドは <ENDOFFILE> エラーを生成し、主出力への Write コマンドは無視されます。
- IRIS_TTNOUSE ー このフラグは、IRIS_TTALL、IRIS_TTCALLIN、および IRIS_TTSTART で使用可能です。これは、IRIS_TTNEVER フラグと IRIS_TTNONE フラグにより暗黙的に設定されます。最初に主入力と主出力を開いた後は、InterSystems IRIS の Open コマンドと Use コマンドを使用してもターミナルの変更はできなくなります。

IrisStartW の返り値

IRIS_ALREADYCON	既に接続されています。$ZF 関数から IrisStartW を呼び出した場合に返されます。
IRIS_CONBROKEN	接続が確立され、その後切断されました。また、IrisEndW は、削除するために呼び出されませんでした。
IRIS_FAILURE	予期しないエラーが発生しました。
IRIS_STRTOOLONG	prinp または prout が長すぎます。
IRIS_SUCCESS	接続が確立されました。

フラグ・パラメータは、C プログラムの振る舞いと、InterSystems IRIS ターミナル属性の設定方法を通知します。時間はかかりますが最も安全な方法は、ObjectScript の呼び出しごとに InterSystems IRIS からターミナルを設定し、その設定をリストアすることです。ただし、ユーザ自身が設定を処理し、プログラムに関連する情報のみ収集すれば、ObjectScript のオーバーヘッドを軽減できます。パラメータ IRIS_TTNEVER により、オーバーヘッドを最小限にできます。

例

InterSystems IRIS プロセスが開始されます。ターミナルは、各インタフェースのコールイン関数の後にリセットされます。20 秒以内にパーティションが割り当てられない場合、起動は失敗します。ファイル dobackup は、入力用に使用します。

このファイルには、InterSystems IRIS バックアップ用の ObjectScript スクリプトが含まれます。出力は、ターミナルに表示されます。

```
inpdev;
outdev;
int rc;

strcpy(inpdev.str, "[BATCHDIR]dobackup");
inpdev.len = strlen(inpdev.str);
strcpy(outdev.str,"");
outdev.len = strlen(outdev.str);
rc = IrisStartW(IRIS_TTALL|IRIS_TTNOUSE,0,inpdev,outdev);
```

3.149 IrisTCommit

```
int IrisTCommit( )
```

説明

InterSystems IRIS TCommit コマンドを実行します。

IrisTCommit の返り値

IRIS_SUCCESS	TCommit は正常に終了しました。

3.150 IrisTLevel

```
int IrisTLevel( )
```

説明

トランザクション処理に対する、現在の入れ子レベル (\$TLEVEL) を返します。

IrisTLevel の返り値

IRIS_SUCCESS	TLevel は正常に終了しました。

3.151 IrisTRollback

```
int IrisTRollback(int nlev)
```

引数

nlev	ロールバックするレベル数を決定します (0 の場合はすべてのレベル、1 の場合は 1 つのレベル)。

説明

InterSystems IRIS TRollback コマンドを実行します。nlev が 0 の場合、TSTART が発行されているレベル数に関係なく、処理中のすべてのトランザクションをロールバックし、\$TLEVEL を 0 にリセットします。nlev が 1 の場合、入れ子になったトランザクションの現在のレベル (最新の TSTART によって開始されたレベル) をロールバックし、\$TLEVEL を 1 デクリメントします。

IrisTRollback の返り値

IRIS_SUCCESS	TStart は正常に終了しました。

3.152 IrisTStart

```
int IrisTStart( )
```

説明

InterSystems IRIS TStart コマンドを実行します。

IrisTStart の返り値

IRIS_SUCCESS	TStart は正常に終了しました。

3.153 IrisType

```
int IrisType( )
```

説明

IrisEvalA、IrisEvalW、または IrisEvalH により関数値として返される項目の、ネイティブ・タイプを返します。

IrisType の返り値

IRIS_ASTRING	8 ビットの文字列。
IRIS_CONBROKEN	接続は、深刻なエラー状態、または RESJOB によって切断されました。
IRIS_DOUBLE	64 ビットの浮動小数点。
IRIS_ERSYSTEM	ObjectScript が <SYSTEM> エラーを生成した場合または $ZF 関数から呼び出された場合は、内部カウンタが同期していない可能性があります。
IRIS_IEEE_DBL	IEEE の 64 ビットの浮動小数点。
IRIS_INT	32 ビットの整数。
IRIS_NOCON	接続が確立されていません。
IRIS_NORES	返すことができるタイプを持つ結果がありません（この呼び出しの前に、IrisEvalA または IrisEvalW が呼び出されていません）。
IRIS_OREF	InterSystems IRIS オブジェクト参照。
IRIS_WSTRING	Unicode 文字列。

例

```
rc = IrisType();
```

3.154 IrisUnPop

```
int IrisUnPop( )
```

説明

IrisPop からスタック・エントリをリストアします。

IrisUnPop の返り値

IRIS_NORES	返すことができるタイプを持つ結果が、この呼び出しの前にありません。
IRIS_SUCCESS	操作は正常に終了しました。