入出力デバイス・ガイド

Version 2023.1
2024-01-02

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

入出力デバイス・ガイド
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

図一覧

テーブル一覧

1

入出力デバイス

InterSystems IRIS® は、物理デバイスと論理デバイスの両方でさまざまな種類のデバイスをサポートしています。この章では、以下について説明します。

- 入出力デバイス
- ターミナル入出力
- ローカル・プロセス間通信
- TCP クライアント/サーバ通信
- シーケンシャル・ファイルの入出力
- スプーラ
- プリンタ

InterSystems IRIS は、物理入出力デバイスおよび論理入出力デバイスの両方をサポートします。サポートされる物理デバイスは、以下のとおりです。

- ターミナル
- プリンタ
- ディスク・ドライブ

サポートされる論理デバイスは、以下のとおりです。

- 主デバイス
- スプーラ
- シーケンシャル・ファイル
- フラット・ファイル
- インタジョブ・コミュニケーション・デバイス (IJC デバイス)
- ルーチン・インタロック・デバイス

1.1 デバイス制御ユーティリティ

デバイスを管理するための ObjectScript ユーティリティがいくつか用意されています。以下のテーブルは、これらのユーティリティの要約、およびこれらにアクセスするための別の方法を示しています。

テーブル 1-1: InterSystems IRIS デバイス・ユーティリティ

動作	ユーティリティ	説明
デバイスの定義	管理ポータルのデバイス構成	ユーザが %IS ユーティリティでアクセスできるデバイスを定義します。そのデバイスは、%IS グローバルに格納されています。これらのデバイス定義を、編集、削除することができます。[デバイス] のサブセクションでは、ニーモニックおよびエイリアスを含むデバイスを定義しますが、既定のデバイスも提供されています。デバイスのサブタイプは [デバイスサブタイプ] のサブセクションで定義しますが、既定のサブタイプも提供されています。
既定のニーモニック空間の定義	管理ポータルの IO 構成オプション	WRITE /mnemonic コマンドを使用してデバイスを制御できます。このパネルでは、ニーモニック空間指定の引数付きの OPEN コマンドまたは USE コマンドが先行しない WRITE /mnemonic コマンドを実行するときに、InterSystems IRIS が使用する既定のニーモニック空間の名前を入力します。
文字ベースのアプリケーションで、インタラクティブにデバイスを選択する	%IS	プログラマがこのユーティリティを文字ベースのアプリケーションで呼び出すと、ユーザは "Device:" プロンプトで InterSystems IRIS デバイス・ニーモニックを指定してデバイスを選択できます。InterSystems IRIS デバイスとそのニーモニックは、管理ポータルの [デバイス] オプションを使用して構成する必要があります。
スプール・ファイルで、印刷出力を保存する	%SPOOL	詳細は、“スプール・デバイス” の章を参照してください。

1.2 既定デバイス

1.2.1 デバイス

InterSystems IRIS をインストールする際に、既定のデバイスが定義されます。これらは、管理ポータルの **[デバイス]** 構成サブセクションに表示されます。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[デバイス]** の順に選択します。定義済みのデバイスのリストが表示されます。

1.2.2 デバイス・サブタイプ

InterSystems IRIS には、多数の既定のデバイス・サブタイプが含まれています。各デバイス・サブタイプは、画面のサイズや改ページなどのデバイス特性を定義します。

サブタイプの一覧は、管理ポータルの **[デバイスサブタイプ]** 構成オプションにあります。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[デバイスサブタイプ]** の順に選択します。定義済みのサブタイプのリストが表示されます。

1.3 デバイスの識別

管理ポータルの **[デバイス]** 構成セクションでデバイスを定義するときは、次の 3 つのデバイス識別子を入力して、デバイスを指定します。

- ニーモニック ― %IS デバイス・プロンプトで使用します。

- デバイス ID ― OPEN コマンドで使用します。
- エイリアス ― OPEN コマンドのデバイス ID の場所に使用します。

これらのデバイス識別子には、次のように物理デバイス名より優れた性質があります。

- ユーザが持つ物理デバイスの数にかかわらず、一意に論理デバイスを識別します。
- 各論理デバイスに異なる特性を割り当てます。
- ユーザ・アプリケーションは、プラットフォームごとに異なる実際の物理デバイス名を知らなくても、デバイスを参照できます。

デバイス識別子の使用法の詳細は、"デバイスへのアクセス" を参照してください。

1.3.1 デバイス・ニーモニック

特定のデバイスと 1 つ以上の ニーモニック を対応させることができます。%IS 文字ベース・ユーティリティにより発行された “Device:” プロンプトに応答して、ニーモニックを使用します。

ニーモニックには以下のような利点があります。

- 柔軟性が高く、ニーモニック・ポイントの位置をユーザが変更でき、開発者がアプリケーションを変更する必要がありません。
- ユーザにわかりやすく、開発者が覚えやすいようにできています。例えば、ニーモニック・プリンタでプリンタ・デバイスをセットアップしたり、あるいはファイル名のデバイス ID をセットアップし、ニーモニック・ファイルにその名を付けることができます。

1.3.2 デバイス ID

番号または他のオペレーティング・システム名により、デバイスを識別できます。この識別子を OPEN コマンドで使用します。

1.3.3 デバイス・エイリアス

ユーザが定義した InterSystems IRIS デバイスごとに、1 つ以上の エイリアス を定義できます。ユーザが OPEN コマンドでエイリアスを指定すると、InterSystems IRIS はこれをデバイス ID に変換します。

InterSystems IRIS が提供する既定のデバイス ID は、ほとんどのユーザに対し適切です。ただし、ユーザがこれらの既定を変更することが必要な場合もあります。これは、管理ポータルでデバイスの構成設定の一部としてエイリアスを入力することによって行えます。

1.3.4 既定のデバイス ID およびニーモニック

以下のテーブルは、InterSystems IRIS をインストールする場合の、各デバイス・タイプに対する既定デバイス番号およびニーモニックです。

テーブル 1-2: 既定のデバイス ID およびニーモニック

デバイス	デバイス ID	ニーモニック	メモ
主デバイス	0	TERM	このデバイスのデバイス ID を変更することはできません
スプーラ	2	SPOOL	

ただし、InterSystems IRIS は、ユーザがデバイスの定義に使用する他のデバイス番号も認識します。以下のテーブルは、認識されるデバイス番号のリストを示します。

テーブル 1-3: InterSystems IRIS 既定のデバイス番号

デバイス番号	タイプ	定義
0	主デバイス	インタラクティブなプロセスの場合、これはユーザがログインするターミナルです。InterSystems IRIS ジョブ起動プロセスの場合は NULL デバイス (既定)、ジョブ起動プロセスを生成するジョブ・コマンドの場合は引数リストにあるデバイスです。
1	messages.log	このデバイス番号を使用して、エラー・メッセージや他の特別なメッセージをシステム・メッセージ・ログに送信します。例えば、ターミナルから `OPEN 1 USE 1 WRITE "This is a test" CLOSE 1` を発行すると、指定された文字列がメッセージ・ログに書き込まれます。WriteToConsoleLog() メソッドも参照してください。
2	InterSystems IRIS システム・スプーラ	これは、出力を保存するグローバルです。これにより、別の時に物理入出力デバイスに出力を送信できます。
63	表示バッファ	VIEW コマンドおよび $VIEW 関数を併用し、メモリとディスクの間でデータを移動します。
20-46, 200-223	ルーチン・インタロック・デバイス	DSM ロック・アプリケーションとの互換性を提供します。
4-19, 64-199, 224-255, 2048-2375	IJC デバイス	インタジョブ・コミュニケーション論理デバイス (IJC デバイス)。InterSystems IRIS プロセス間での情報の移動に使用します。これらのデバイスの可用性を制御できます。詳細は、"InterSystems IRIS プロセス間の通信" を参照してください。
なし	NULL デバイス	/dev/ または NL:。表示しない出力を廃棄するために使用する NULL デバイス。
256-2047	ターミナル、プリンタ、フラット・ファイル	

注釈　＊ デバイス 50 には、2048 のハードコードされたブロックサイズがあります。

1.3.5 デバイス・タイプ

ニーモニックやデバイス番号に加え、InterSystems IRIS は入出力デバイス・タイプをサポートします。各内部デバイス番号は、以下のいずれかのタイプに属します。以下のテーブルは、デバイス・タイプを示します。

タイプ	意味
TRM	ターミナル
SPL	スプール・デバイス
IPC	インタープロセス通信デバイス
OTH	その他のデバイス (プリンタなど)

1.4 デバイスの定義

管理ポータルの [**デバイス**] 構成設定で、デバイスを定義、編集、削除します。入力した情報は、^%IS グローバルに格納されています。このグローバルについての詳細は、"^%IS グローバルの構造" のセクションを参照してください。

InterSystems IRIS の稼動中にデバイスを変更した場合、InterSystems IRIS を再起動せずに変更を有効にするかどうかを尋ねられます。変更を有効にすることに同意した場合、すぐに新規の定義が利用できるようになります。

1.5 デバイスへのアクセス

Windows システムでは、インタジョブ・コミュニケーション・デバイスおよびルーチン・インタロック・デバイスに対して、必ずデバイス番号を使用します。ターミナルやプリンタに対しては、ユーザが割り当てたデバイスのニーモニックやデバイス番号を使用できます。

UNIX® システムでは、UNIX® ファイル仕様を使用してファイルを参照したり、ファイルを参照するデバイス番号を設定したりできます。

デバイスにアクセスするには、以下の 2 つのうちのいずれかの方法を使用します。

- %IS ユーティリティの “Device:” プロンプトで、デバイスのニーモニックを入力します。
- OPEN コマンドを呼び出し、デバイス ID あるいはエイリアスを入力します。

1.5.1 %IS ユーティリティでユーザがデバイスを選択できるようにする

文字ベースのアプリケーションを使用するユーザが、デバイスをインタラクティブに選択できるようにするには、アプリケーションの %IS ユーティリティを呼び出します。%IS ユーティリティの詳細は、"ユーザによるデバイスの指定" を参照してください。

%IS ユーティリティを使用して、デバイスを選択するには、次の手順に従います。

1. “Device:” プロンプトで、デバイスのニーモニックを入力します。

   テーブル 1-4: 事前定義のニーモニック空間

ニーモニック	対応するデバイス
<ENTER>	ターミナル画面
SPOOL	スプーラ
2	スプーラ
|PRN|	既定の Windows のプリンタ
ファイル名 : MYFILE.TXT DEV$:[TEST]MYFILE.TXT C:¥MGR¥MYFILE.TXT	指定したパスのファイル。パス指定がない場合は、現在のディレクトリ。

2. デバイスのタイプによって、別のプロンプトが表示される場合もあります。

テーブル 1-5: デバイス・ユーティリティ

デバイス	プロンプト	有効な応答
ターミナル	Right Margin	行ごとの文字の数を表す数字
プリンタ	Right Margin	行ごとの文字の数を表す数字
スプーラ	Name (of file)	プラットフォームの有効なファイル名。パスはオプション。
ファイル名	Parameters	デバイス・タイプに対する OPEN コマンドの有効なパラメータ・リスト

1.5.2 OPEN コマンドによるデバイスへのアクセス

ターミナルまたは ObjectScript アプリケーションで OPEN コマンドを使用して、読み取るあるいは書き込む特定のデバイスを開きます。デバイスの指定には、デバイス ID またはそのエイリアスを使用できます。

1.5.3 デバイスとの相互運用レベル

%IS またはOPEN コマンドと併用するデバイス識別子は、相互運用のレベルで 3 段階に分けられます。したがって、%IS グローバルの“Device:”プロンプトでニーモニック 47 を入力しても、最終的に使用されるデバイス ID は異なる場合があります。以下で、その 3 段階について説明します。

1.5.3.1 レベル 1 : %IS ユーティリティ・レベル

デバイスが %IS ユーティリティで選択されている場合、この最初のレベルが使用されます。^%IS グローバルのニーモニックは、デバイス番号と対応付けられます。その後、%IS ユーティリティは、そのデバイス番号に OPEN コマンドを発行します。

1.5.3.2 レベル 2 : OPEN コマンド・レベル

OPEN コマンド内で、InterSystems IRIS はデバイス・パネル・テーブルのエイリアス列にこの番号があるかどうかを確認します。番号があれば、これを実際のデバイス番号、あるいはそのデバイスの名前に変換します。

注釈　^%IS のニーモニックを使用してデバイスにアクセスする場合は、デバイス ID が一致していても、関連付けられたデバイスが異なるエイリアスを定義しないようにしてください。

1.6 既定のニーモニック空間の定義

プログラマは、アプリケーションで WRITE /mnemonic コマンドを使用して、デバイスを制御できます。例えば、%X364 ニーモニック空間を使用するときに、ターミナル・デバイスの現在の行の特定の列に、カーソルを移動することができます。

ObjectScript

```
WRITE /CHA(column)
```

ニーモニックの特定の値によって発生するアクションは、WRITE コマンドが使用しているニーモニック空間によって決まります。ニーモニック空間とは、デバイスのアクションおよび属性を定義するエントリ・ポイント (ニーモニック) を持つルーチンです。

WRITE コマンドは、デバイスに対する OPEN コマンドや USE コマンドで定義されたニーモニック空間を使用します。OPEN コマンドまたは USE コマンドにニーモニック空間の引数が含まれない場合、InterSystems IRIS はデバイス・タイプに対する既定のニーモニック空間を使用します。

1.6.1 事前定義のニーモニック空間

InterSystems IRIS には、事前に定義された (既定の) ニーモニック空間 ^%X364 が用意されています。これは、X3.64 (ANSI) ターミナル用の既定のニーモニック空間です。ターミナル、シーケンシャル・ファイル、およびその他のデバイス起動時の既定です。

これらの既定値は、管理ポータルで定義します。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[IO設定]** の順に選択します。

ユーザ独自のニーモニック空間ルーチンを作成するには、これらのデバイス・タイプのうち 1 つ以上のタイプに対して InterSystems IRIS が使用している、既定のニーモニック空間を変更します。

2
入出力デバイスとコマンド

この章では、InterSystems IRIS® Data Platform アプリケーションと InterSystems IRIS プロンプトにおける入出力デバイスの操作方法を説明します。"入出力デバイス" の章では、ユーザのデバイスは適切にセットアップが完了しているものとして説明します。特定のデバイスに関する追加情報には、このドキュメントの他の章を参照してください。

2.1 入出力コマンドの概要

入出力コマンドにより、デバイスの所有、使用、読み取り、書き込み、クローズができます。入出力処理をデバイスに指示するには、以下のコマンドを最初に発行します。

- デバイスが主デバイスではない場合、OPEN コマンドを発行し、所有権を構築します。
- USE コマンドを発行して、デバイスを現在のデバイスにします。
- 次に、READ と WRITE コマンドを発行して、そのデバイスからの読み取り、デバイスへの書き込みを行います。
- CLOSE コマンドを発行してデバイスの所有権を解放し、他のプロセスがそのデバイスを使用できるようにします。

以下のセクションでは、InterSystems IRIS 入出力コマンドの概要を説明します。

2.1.1 一般的な入出力構文

以下の一般構文は、ObjectScript で入出力コマンド・キーワードをサポートする入出力コマンドを適用しています。

```
OPEN device:paramlist:timeout:"mnespace"
USE device:paramlist:"mnespace"
CLOSE device:paramlist
```

paramlist は、単一のパラメータ、または括弧で囲まれコロンで区切られたパラメータのリストのいずれかです。

```
 parameter  (parameter:parameter[:...])
```

parameter は、位置パラメータあるいはキーワード・パラメータのいずれかです。キーワード・パラメータは以下の構文になります。

```
/keyword[=value]
```

最初にスラッシュがあるものがキーワード・パラメータで、そのスラッシュの有無により位置パラメータ値と区別します。位置パラメータ値は、コロンで区切られたリストからその位置を取得します。キーワード・パラメータ値は、指定したキーワードから取得します。

位置パラメータとキーワード・パラメータのどちらも、同じ paramlist で指定できます。例えば、以下の例は位置パラメータとキーワード・パラメータを混在して使用し、JIS の入出力変換による書き込みモードあるいはシーケンシャル・モードで test.dat というファイルを新規に開きます。

ObjectScript

```
OPEN "test.dat":("NWS":/IOTABLE="JIS")
```

2.1.2 OPEN コマンド

OPEN コマンドは、指定されたデバイスの所有権を構築し、入出力チャネルを開きます。この所有権は、CLOSE コマンドを発行するか、プロセスが終了するか、物理的処理がデバイスを閉じるまで保持されます。物理入出力デバイスや、TCP 接続などのプロセス間通信では、この所有権により、他のプロセスすべてがデバイスにアクセスできないようにしています。シーケンシャル・ファイルなどの論理入出力デバイスでは、この所有権が設定されていても、共有アクセスの形式で他のプロセスからデバイスにアクセスできることがあります。同じシーケンシャル・ファイルを複数のプロセスで開けるかどうかは、プラットフォームに依存しています。LOCK コマンドを使用してシーケンシャル・ファイルへのアクセスを制限してください。

2.1.2.1 構文

```
OPEN device{:{(parameters)}{:{timeout}{:"mnespace"}}}
```

引数	説明
device	デバイス名、ID 番号、またはニーモニックです。device の最大長は 256 文字です。
parameters	オプション — デバイスに必要な追加情報を指定する 1 つまたは複数のパラメータです。このパラメータのリストは括弧で囲まれ、リスト内のパラメータはコロンで区切られます。個別のパラメータは、"プロセス間通信"、"シーケンシャル・ファイルの入出力"、"ターミナル入出力" の章のテーブルにリストされています。
timeout	オプション — 要求の成功を待機する時間を指定します。先頭のコロンは必須です。timeout は、必ず整数値または式で指定します。timeout がゼロ (0) の場合、OPEN はファイルを 1 回だけ開こうとします。実行が失敗した場合、OPEN も即座に失敗します。試行に成功した場合は、ファイルが無事に開きます。timeout を設定していない場合、InterSystems IRIS は OPEN が成功するまで、またはプロセスが手動で終了されるまで、デバイスを開く操作を繰り返します。
mnespace	オプション — 引用符付き文字列として指定された、このデバイスで使用する制御ニーモニックを含むニーモニック・スペース名です。このデバイスへ入出力を指示するときに、これらの制御ニーモニックを WRITE /mnemonic コマンドで使用できます。

詳細は、"ObjectScript リファレンス" の "OPEN" コマンドを参照してください。

2.1.2.2 例

以下の例は、さまざまなプラットフォームで OPEN コマンドを使用する方法を示しています。このコマンドは、コマンド行での入力やルーチンでの使用が可能です。ルーチンで使用する場合、プラットフォーム固有のアイテムを変数で置き換える必要があることもあります。

Windows システムでの OPEN の使用例

以下のコマンドは、Windows システムからターミナル・サーバに、外部の Telnet を接続します。

ObjectScript

```
OPEN "|TNT|node:port"
```

node はノード名、port はサーバ上の IP ポートです。

以下のコマンドは、既存の Windows ファイルとの間に入出力チャネルを開きます。

ObjectScript

```
OPEN "c:\abc\test.out":"WS"
```

UNIX® システムでの OPEN の使用例

以下のコマンドは、UNIX® ターミナル・デバイス /dev/tty06 への入出力チャネルを開きます。

ObjectScript

```
OPEN "/dev/tty06/"
```

2.1.3 USE コマンド

このコマンドは、特定のデバイスを現在のデバイスにし、そのデバイスに特殊変数 $IO を設定します。主デバイス以外のデバイスに USE を使用するには、最初に OPEN コマンドを発行する必要があります。このコマンドを発行しない場合、<NOTOPEN> エラーが返されます。引数は、OPEN コマンドと同じ意味を持ちます。

2.1.3.1 構文

```
USE device:(args):"mnespace"
```

引数	説明
device	デバイス名、ID 番号、またはエイリアスです。device の最大長は 256 文字です。
args	オプション – 一部のデバイスに必要な追加情報です。これは、"プロセス間通信"、"シーケンシャル・ファイルの入出力"、"ターミナル入出力" の章のコマンド・キーワードのテーブルにリストされています。
mnespace	オプション – このデバイスへ入出力を指示するときに、WRITE /mnemonic コマンドで使用できる制御ニーモニックの定義を含む InterSystems IRIS ルーチン名です。

詳細は、"ObjectScript リファレンス" の "USE" コマンドを参照してください。

2.1.3.2 例

以下の例は、さまざまなプラットフォームで USE コマンドを使用する方法を示しています。このコマンドは、コマンド行での入力やルーチンでの使用が可能です。ルーチンで使用する場合、プラットフォーム固有のアイテムを変数と置換することもできます。

Windows システムでの USE の使用例

次の Windows の例では、TCP 経由でリモート・ホスト “larry” 上の時刻サーバに接続するために使用するコマンドを示しています。このコマンドではサービス名 `daytime` を使用しますが、これはローカル・システムでポート番号に解決されます。USE コマンドは、OPEN C モードを PSTE モードに置き換え、すべてのユーザ・ターミネータをオフにします。

ObjectScript

```
OPEN "|TCP|4":("larry":"daytime":"C")
USE "|TCP|4":(::"PSTE")
```

UNIX® システムでの USE の使用例

以下の UNIX® コマンドの例では、“/dev/tty06” デバイスとの間に入出力チャネルを開き、X364 ターミナル・ニーモニックで WRITE/mnemonic を使用するオプションを指定して、このデバイスを現在のデバイスに設定します。

ObjectScript

```
OPEN "/dev/tty06"
USE "/dev/tty06"::"^%x364"
```

2.1.4 READ コマンド

このコマンドは、現在のデバイスからデータを読み取ります。デバイスによっては、アスタリスクで始まる引数に ASCII の数値情報を返すものがあります。その他のデバイスでは、この引数は制御関数を示します。

2.1.4.1 構文

```
READ variable:timeout
```

詳細は、″ObjectScript リファレンス″ の ″READ″ コマンドを参照してください。

2.1.5 WRITE コマンド

このコマンドは、現在のデバイスにデータを書き込みます。デバイスによっては、最初にアスタリスクが付いた引数を指定すると、ASCII の数値を使用して ASCII 文字を書き込むことができるものがあります。その他のデバイスには、この引数は制御関数を示します。デバイスによっては、先頭に # 文字が付いた引数で、指定された文字を書き込む回数を示すものもあります。

WRITE /mnemonic 構文により、ニーモニック空間の InterSystems IRIS コードに定義されたニーモニック・デバイスを制御できます。ニーモニック空間は、InterSystems IRIS ルーチンであり、OPEN コマンドまたは USE コマンドでアクティブにするか、管理ポータルを使用してデバイスの既定として構成する必要があります。ニーモニック空間を定義して、アクティブにする方法については、″既定のニーモニック空間の定義″ を参照してください。

2.1.5.1 構文

```
WRITE variable
```

詳細は、″ObjectScript リファレンス″ の ″WRITE″ コマンドを参照してください。

2.1.5.2 例

以下のコマンドを発行し、事前定義された ^%X364 ニーモニック空間を使用して、ターミナル画面の 2 行目の 1 列目にカーソルを移動します。

ObjectScript

```
WRITE /CUP(1,2)
```

2.1.6 CLOSE コマンド

CLOSE コマンドは、指定されたデバイスの所有権を解放します。CLOSE は、OPEN コマンドと逆の動作を実行します。

2.1.6.1 構文

```
CLOSE device[:params]
```

引数	説明
device	デバイス名、ID 番号、またはニーモニックです。
params	パラメータ “K” は、オペレーティング・システム・レベルではデバイスを閉じず、InterSystems IRIS レベルで閉じます。 K パラメータは、Windows システムでは何も動作しません。ファイルは、オペレーティング・システム・レベルで閉じます。

CLOSE コマンドを主デバイスに発行すると、主デバイスは、ユーザがログオフするまでプロセスに割り当てられたままになります。

他のいくつかの条件は、CLOSE の振る舞いに影響します。

- 何らかの原因によりデバイスへの出力が停止した場合、InterSystems IRIS は、そのデバイスへの出力を完了できないことがあります。この場合、デバイスを閉じたり、停止したりできないことがあります。例えば、ターミナルがオペレーティング・システムに Ctrl-S を送信し、ターミナルへの出力を停止するよう命令した場合、Ctrl-Q を押して、ターミナルへの出力を再開する必要があります。
- 現在のデバイスを閉じると、CLOSE は、システム変数 $IO の値を主デバイスの値に変更します。CLOSE コマンドは、デバイスへのすべての出力が完了した後にのみ、現在のデバイスの所有権を解放します。
- プロセスが停止すると、InterSystems IRIS で作業中にそのプロセスが開いたすべてのデバイスが自動的に閉じます。

  何らかの原因によりデバイスへの出力が停止した場合、InterSystems IRIS がそのデバイスへの出力を完了できないことがあります。この場合、デバイスを閉じたり、停止したりできないことがあります。

詳細は、″ObjectScript リファレンス″ の ″CLOSE″ コマンドを参照してください。

2.2 入出力デバイスの指定

InterSystems IRIS アプリケーションを開発したり、InterSystems IRIS プログラマ・プロンプトで入出力デバイスを動かすときに、入出力デバイスを指定するには以下の 2 つの方法があります。

- %IS ユーティリティを呼び出すと、%IS グローバルで定義されたニーモニックを使用して、デバイスを指定できます。
- InterSystems IRIS デバイス番号、あるいはデバイスのオペレーティング・システムのファイル指定を使用して、入出力コマンドの OPEN、USE、CLOSE を発行します。

2.3 ユーザによるデバイスの指定

%IS は、文字ベース・アプリケーションの汎用的なデバイス選択ユーティリティです。組み込みの %IS ユーティリティを使用すると、入出力処理の対象となるデバイスをユーザが選択できます。デバイスを選択すると常に、アプリケーション・プログラムは %IS ユーティリティを呼び出す必要があります。このユーティリティにより、ユーザは使用するデバイスと、適切

な OPEN コマンド・パラメータを指定できます。また、選択済みデバイスを開いて、呼び出しプログラムにデバイス固有の情報を返します。ユーザは、^%IS グローバルで定義されたニーモニックを入力します。%IS は、管理ポータルで設定された IO 構成の既定値に依存します。

この章では、以下の項目について説明します。

- %IS の動作
- %IS ニーモニック
- ^%IS グローバルの構造

2.3.1 %IS の動作

2.3.1.1 デバイス・プロンプト

%IS ユーティリティを呼び出すと、InterSystems IRIS はデバイス名を要求します。以下のいずれかの方法で入力します。

- デバイス名あるいは ID 番号を入力します。
- デバイスのニーモニックを入力します。
- 現在のデバイスを選択するには、Enter キーを押します。

%IS は以下のように応答します。

- デバイスのニーモニックを入力すると、%IS は、^%IS グローバル内で対応するデバイスを見つけて開きます。
- デバイス名を入力した場合、%IS は、そのデバイスに OPEN コマンドを発行します。
- デバイスが InterSystems IRIS デバイス ID の場合、%IS は、番号が実際のデバイス番号に再度マップされたかどうか、デバイス・テーブルをチェックします。その後、デバイスに OPEN を発行します。

代替デバイスの使用についての情報は、"%IS ニーモニック" のセクションの “代替デバイス” を参照してください。

2.3.1.2 その他の質問

ターミナルをデバイスとして指定すると、ユーティリティは、既定の right margin を表示します。Enter キーを押してそのマージンを選択するか、異なる値を入力します。指定された right margin を超えてプログラムを記述しようとすると、オペレーティング・システムは、マージンに到達したときに “CR LF” (キャリッジ・リターンと改行) を挿入します。ターミナル以外のデバイスを選択すると、ユーティリティは、別の種類の質問を表示します。

2.3.1.3 例

以下の例では、Enter キーを押して、ターミナルを指定します。ユーティリティは、right margin というプロンプトに、既定値の 80 を表示しますが、ユーザは => プロンプトに新規マージン設定として 132 と入力します。

```
%SYS>DO ^%IS
Device: <RETURN>
Right margin: 80 => 132
%SYS>
```

2.3.1.4 %IS は IO 変数を設定し、他の変数値を返す

デバイスを選択すると、%IS は、OPEN コマンドで使用するデバイス名あるいは番号を IO 変数に設定します。%IS は、以下のテーブルの変数値も返します。

テーブル 2-1: %IS デバイス変数値

変数	例	説明
%ANS	あり	一般的なダイアログの応答。
IO	64	選択したデバイスのデバイス番号またはデバイス・ニーモニック。
IOF	#	改ページ。WRITE # は、改ページ発行し、$Y を変更します。改ページには WRITE @IOF を使用する必要があります。
IOBS	*8	バックスペース。WRITE $CHAR(8) は、バックスペースを発行し、$X を変更します。WRITE *8 は、バックスペースを発行しますが、$X を変更しません。バックスペースには、WRITE @IOBS を使用する必要があります。
IOM	80	右マージン。
IOSL	66	画面/ページの長さ。
IOT	TRM	デバイス・タイプ。
IOST	C-VT220	デバイス・サブタイプ (この例では VT220)。
IOPAR	(auv:0:2048)	他の OPEN パラメータ。
MSYS	M/WNT	システム・タイプ (UNIX®、Windows NT など)。
POP	0	0 以外の場合、デバイスが未選択であることを示します。つまり、ユーザが Device: プロンプトで「STOP」と入力します。
RMSDF	RW	Read/Write 権限。

2.3.1.5 OPEN パラメータ

OPEN コマンドは、既定として ^%IS グローバルに定義されたデバイス指定を使用します。%IS を使用するとき、他の設定を指定して、現在の設定をオーバーライドできます。

2.3.1.6 USE コマンドの発行

%IS の実行後、アプリケーションは、%IS で開いたデバイスに USE コマンドを発行する必要があります。%IS を呼び出すたびに値が変更されますが、IO 変数を使用できます。その後、READ や WRITE など、後続の InterSystems IRIS 入出力コマンドは、そのデバイスを参照します。

2.3.1.7 CLOSE コマンドの発行

ユーザまたはアプリケーション開発者は、%IS ユーティリティで開いたデバイスを閉じる必要があります。

2.3.2 %IS ニーモニック

%IS には、簡単に使用するための機能がいくつかあります。例えば、入出力を自身のターミナルに送信する場合は、“Device” プロンプトで Enter キーを押すだけで済みます。また、既定で組み込まれているニーモニックや、独自に定義した新しいニーモニックも使用できます。

2.3.2.1 デバイス・ニーモニック

さまざまなデバイスに、ニーモニックを設定すると大変便利です。1 つのデバイスに複数のニーモニックを設定することもあります。複数のニーモニックにより、デバイスごとに異なるデバイスの特性を指定し、使用方法に応じて特性を変更できます。例えば、通常データ入力に使用するターミナルには、ターミナルとしての特性がありますが、補助プリンタ機能が

備わっている場合があります。同じデバイスを異なった特性で開くさまざまなニーモニックを割り当てることにより、ハード・コピーが必要なときに、ターミナル/プリンタの組み合わせをプリンタとして使用できます。

デバイスのニーモニックと特性は管理ポータルを使用して構成できます。ニーモニック空間を定義して、アクティブにする方法については、"既定のニーモニック空間の定義" を参照してください。

2.3.2.2 既定ニーモニック

^%IS グローバルは、インストール時にいくつかの既定ニーモニックで初期化されています。例えば、InterSystems IRIS スプーラには、SPOOL と 2 の既定ニーモニックがあります。“2” または “SPOOL” を入力すると、InterSystems IRIS スプーラへの出力を送信します。

RT:、LT:、または VT: のいずれかのタイプのデバイスにログインしていて、ユーザのターミナルが現在のデバイスである場合、%IS は “Device” プロンプトへの応答として、0、“ ”、または IO の値を受け入れます。ターミナル・タイプに適切なテンプレート (RT0:、LT0:、または VT0:) を使用し、ターミナル情報を生成します。

2.3.2.3 代替デバイス

Device プロンプトに “A” を入力すると、現在のデバイスで定義された代替デバイスに出力されます。代替デバイスは、通常プリンタです。システムで、各デバイスごとに代替デバイスを定義する代わりに、ニーモニック “A” を使用して、プリンタを示すデバイスを生成することができます。その後、ユーザが %IS の “ Device” プロンプトに「“A”」を入力すると、そのデバイスに出力されます。

2.3.2.4 CURRENT^%IS エントリ・ポイント

CURRENT は、%IS ユーティリティの内部的なエントリ・ポイントで、現在のデバイスのデバイス・パラメータの取得に使用できます。この %IS への呼び出しは、さまざまな変数値を返すため、主デバイス用に 1 セットのパラメータ設定と、異なる特性を持つデバイス用に別のパラメータ設定を保持できます。通常、ログイン時にこの内部エントリ・ポイントへの呼び出しを実行します。これによりアプリケーションは、主デバイスのデバイス特性にアクセスできます。CURRENT^%IS は、以下の一覧表示された変数値を返します。

テーブル 2-2: CURRENT の返り値

変数	例	説明
FF	3	WRITE @FF は、このデバイスで改ページに使用
BS	*8	WRITE @BS は、バックスペースに使用
RM	80	右マージン
SL	24	画面/ページの長さ
SUB	C-VT100	デバイス・サブタイプ
XY	(以下の例を参照)	カーソル位置を変更するために、$X に DX、$Y に DY を設定

2.3.2.5 例

CURRENT^%IS を呼び出した後、$X および $Y を DX および DY に設定し、カーソル位置を決定します。

ObjectScript

```
DO CURRENT^%IS
WRITE *27,*61,*DY+32,*DX+32
SET $X=DX,$Y=DY
```

2.3.2.6 IN^%IS エントリ・ポイント

IN は、%IS の内部エントリ・ポイントです。これは、デバイスから入力のみを行うルーチンで呼び出されます。プリンタなどの出力専用のデバイスを選択できないようにするために、このエントリ・ポイントを使用します。

```
%SYS> Do IN^%IS

Device: 3
Right margin: 132= <RETURN>
[you can't read from this device]
Device: <RETURN>
Right margin: 80= <RETURN>
%SYS>
```

2.3.2.7 OUT^%IS エントリ・ポイント

OUT は、%IS の内部エントリ・ポイントです。これは、デバイスから出力のみを行うルーチンで呼び出されます。

2.3.2.8 スプーリング

InterSystems IRIS スプーリングは、ユーザのオペレーティング・システムで実行されるスプーリングから独立しています。InterSystems IRIS のスプーリングは、プログラムの出力を直ちに印刷するのではなく、グローバルに自動的に保存する技術です。プリンタにグローバルのコンテンツを送信することで、後で出力を印刷できます。

SPOOL は既定のニーモニックです。スプーリングを指定するには、Device プロンプトで「“SPOOL”」と入力します。その後システムは、スプール・ファイル名とその詳細の入力を要求します。^SPOOL グローバルで使用する名前を指定します (オペレーティング・システム・レベルで設定された個別のファイル名ではありません)。

指定した名前で始まるファイル名が複数存在する場合や同じ名前が存在する場合は、それらがプロンプトに表示され、いずれかを選択するように要求されます。既存ファイルからファイルを選択しない場合、システムでは、指定された名前と詳細を使用して新規ファイルを生成できます。以下はその例です。

```
Device: SPOOL
Name:TEST
1. 1 TEST 02 Nov 1999 10:17 am First test
2. 2 TEST 02 Nov 1999 10:18 am Second Test
Select one: <Return> not found
Create new document 'TEST'? Yes => yes
Description: Third Test
```

既存のファイルに続けて追加するために既存のドキュメントを再度選択すると、システムは以下のオプションを提供します。

1. ファイルの最後に追加します。
2. 最終ページの最初で再開します。この場合、削除される行が画面に表示されます。
3. 1 ページ目 (最初) で再開します。

スプーリングに %IS を呼び出すと、以下のテーブルの変数を %IS に渡すことができます。

テーブル 2-3: %IS に渡すスプール変数

変数	機能
IODOC	ドキュメント名 (この変数が存在し、すべての質問を禁止する NULL 文字列以外の場合、この名前の新規ドキュメントが自動的に生成されます)。
IODES	フリー・テキストの記述
IOPGM	適切な書式設定でプリンタを調整できるよう、印刷時に呼び出されるルーチン名

2.3.2.9 その他の %IS の特徴

%IS を使用して、以下のタスクも実行できます。

- Right margin 禁止 ー デバイスを選択しても、Right margin を要求しないようにターミナル行を設定できます。既定値は、自動的に設定されます。
- デバイスの自動選択 ー %IS ユーティリティが呼び出されたときに IOP 変数が存在する場合、ユーティリティはデバイスについて問い合わせず、自動的にデバイスを開きます。%IS が正常に動作しない場合、POP 変数に 1 を設定します。
- 事前設定ターミナル ー 管理ポータルを使用すると、ユーザにデバイス情報を要求しないデバイスを構成できます。

2.3.3 ^%IS グローバルの構造

%IS グローバルは %SYS ネームスペースに保存されます。これには、2 つのサブスクリプトが含まれています。最初のサブスクリプトは、管理ポータルでデバイスに対して設定されたニーモニック名です。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[IO設定]** の順に選択すると、さまざまなデバイス・タイプの既定のニーモニックを表示できます。2 番目のサブスクリプトは 0 または 1 です。

2.3.3.1 ノード 0 のコンテンツ

ノード 0 は、デバイス・パネルの位置情報を含みます。

```
^%IS(mnemonic,0) = Location
```

2.3.3.2 ノード 1 のコンテンツ

ノード 1 は、キャレット (^) で区切られた、デバイス・パネルの他のフィールド値を含みます。

```
^%IS(mnemonic,1) = Device #^Type^Subtype^Prompt code^not used
^Other Open parameters^Alternate device
```

以下の例では、ニーモニック名 2 のデバイス (InterSystems IRIS スプーラの既定名) は、デバイス番号が 2、デバイス・タイプが SPL (スプール)、デバイス・サブタイプが PK-DEC です。他の値は、スプール・タイプのデバイスには定義されません。

```
^%IS(2,1) = 2^SPL^PK-DEC^^^^^
```

2.4 入出力コマンドでのデバイスの指定

入出力コマンド OPEN、USE、CLOSE を使用して、作業中以外のデバイスで入出力操作を実行する場合、入出力デバイスを指定する必要があります。デバイス・タイプにより、以下の 3 つの方法でデバイスを指定できます。

テーブル 2-4: 入出力コマンドでのデバイスの指定

指定する種類	デバイスの用途
InterSystems IRIS デバイス名	ターミナルとプリンタ
InterSystems IRIS デバイス ID またはデバイス・エイリアス	シーケンシャル・ファイル以外のすべてのデバイス
ファイル名	シーケンシャル・ファイル

Windows および UNIX® では、プリンタ入出力の扱いがそれぞれ異なっています。詳細は、このドキュメントの "プリンタ" の章を参照してください。

2.4.1 デバイス名によるターミナルとプリンタの指定

ターミナル (あるいは、いくつかのプラットフォームではプリンタ) に対して入出力操作を行う場合、オペレーティング・システム (UNIX® または Windows) が提供するデバイス名を使用してデバイスを指定できます。形式は以下のとおりです。

```
OPEN "device"  USE "device"  CLOSE "device"
```

パラメータ	説明
device	引用符で囲んだ、オペレーティング・システムでのデバイス名device の最大長は 256 文字です。

2.4.1.1 Windows システムでのターミナルの指定

シリアル通信ポートに接続された入出力デバイスを開くには、以下の構文で OPEN コマンドを指定します。

```
OPEN "comn:"
```

n は、デバイスに接続されたポート番号です。

パラメータ	説明
n	デバイスが接続されたポート番号

ObjectScript

```
OPEN "com1:"
```

2.4.1.2 UNIX® でのターミナルとプリンタの指定

UNIX® デバイス名 /dev/tty06 を持つターミナルで入出力デバイスを開くには、以下のコマンドを入力します。

ObjectScript

```
OPEN "/dev/tty06"
```

UNIX® システムでは、プリンタは、OPEN コマンドで指定した名前で識別され、tty デバイスの “character special” ファイルとして処理されます。したがって、サポートされる OPEN と USE のコマンド引数は、ターミナル入出力のコマンド引数と同じであり、シーケンシャル・ファイル入出力のコマンド引数とは異なります。Windows システムでは、プリンタ入出力は、シーケンシャル・ファイル入出力と同じように処理されます。

2.4.2 InterSystems IRIS ID によるデバイスの指定

インターシステムズ社の他の製品との互換性と利便性のため、デバイス番号を使用してデバイスを参照できます (デバイス・テーブルに格納されています)。システム管理者は、管理ポータルを使用して、これらの番号をデバイスにリンクできます。[**システム管理**]、[**構成**]、[**デバイス設定**]、[**デバイス**] を選択して、新規デバイスの作成や既存デバイスの編集を行います。

システム管理者は、特定のデバイス番号を別の番号へ変換できます。例えば、ユーザが発行した `OPEN 47` を、InterSystems IRIS は `OPEN 49` に変換できます。

以下のテーブルはデバイス番号を示します。

テーブル 2-5: InterSystems IRIS デバイス番号とデバイス

デバイス番号	デバイス
0	主デバイス (ログインしたデバイス)
2	InterSystems IRIS スプーラ。UNIX® : このデバイスにはニーモニック SPOOL が適用されます。
3	無効なデバイス番号。無効な番号のデバイスをオープンしようとすると、timeout の期限切れを待たずに <NOTOPEN> エラーが返されます。
63	表示バッファ
20–46, 200–223	ルーチン・インタロック・デバイス
224–255	インタジョブ・コミュニケーション・デバイス

2.4.2.1 例

スプーラを開くには、以下のコマンドを発行します。

ObjectScript

```
OPEN 2
```

2.4.3 ディスクでファイルの指定

二重引用符で囲まれた、オペレーティング・システムのファイル指定を使用して、ディスク・ファイルを開くことができます。

Windows のファイル指定は以下の形式です。

```
device:\directory\file.type
```

UNIX® のファイル指定は以下の形式です。

```
/directory/name
```

詳細は、このドキュメントの “シーケンシャル・ファイルの入出力” の章にある ″ファイルの指定″ を参照してください。

2.4.3.1 UNIX の例

UNIX® システムまたは Windows システムの現在の既定ディレクトリが /usr/user の場合、そのディレクトリに格納されている pat_rec.dat というファイルを次のように開くことができます。

ObjectScript

```
OPEN "pat_rec.dat"
```

システムは、自動的にこのファイルを開きます。新規ファイルの場合は、パラメータ文字列 “WN” を追加して、システムが停止しないようにします。

別のディレクトリに格納された、pat_rec.dat と同じ名前を持つファイルを開くには、以下のようにディレクトリを指定する必要があります。

ObjectScript

```
OPEN "/usr/elsewhere/pat_rec.dat"
```

2.5 プロセスとデバイス

2.5.1 主デバイスと現在のデバイス

2.5.1.1 主デバイスを持つ各プロセス

各 InterSystems IRIS プロセスは、主入力デバイスと主出力デバイスを 1 つずつ持ちます。既定では、これらは同じデバイスです。ターミナルからログインし InterSystems IRIS を起動すると、そのターミナルが主デバイスになります。InterSystems IRIS は、暗黙の OPEN コマンドと USE コマンドをターミナルに発行するので、そのターミナルに直ちに READ コマンドと WRITE コマンドを発行できます。InterSystems IRIS の主デバイスとは、オペレーティング・システムによって主入力デバイスとして割り当てられたデバイスです。$PRINCIPAL 特殊変数は、主デバイスのデバイス ID を含みます。

2.5.1.2 InterSystems IRIS による現在のデバイスへの入出力コマンド

InterSystems IRIS は、READ、WRITE、PRINT、および ZLOAD コマンドなどの入出力処理を現在のデバイスに命令します。プロセスの $IO 特殊変数は、現在のデバイスのデバイス ID を含みます。ターミナルで InterSystems IRIS にログインする際、$IO は、ターミナルのデバイス名を最初に含みます。つまり、主デバイスと現在のデバイスは、ログイン直後は同じになります。USE コマンドの実行後、現在のデバイス（$IO に保持されています）は通常、最後に実行した USE コマンドで指定したデバイスになっています。

プログラマ・モードで、主デバイス以外のデバイスに対して OPEN と USE を発行することはできますが、InterSystems IRIS は“>”プロンプトに戻るたびに、暗黙に USE 0 を発行します。0 以外のデバイスの使用を続けるためには、“>”プロンプトで入力する行ごとに USE コマンドを発行する必要があります。

2.5.1.3 主デバイスを現在のデバイスに設定

以下のいずれかの場合に、自動的に主デバイスが現在のデバイスに設定されます。

- 初めてサインオンしたとき。
- USE 0 コマンドを発行したとき。
- %Library.Device クラスの ChangePrincipal() メソッドの呼び出しを発行したとき。
- エラー・トラップが設定されていない状態でエラーが発生したとき。
- 現在のデバイスを閉じたとき。
- プログラマ・モードに戻ったとき。
- HALT コマンドを発行して InterSystems IRIS を終了したとき。

2.5.1.4 USE 0 による主デバイスのオープン

USE 0 は、主デバイスに対する OPEN コマンドを意味します。他のプロセスがそのデバイスを所有している場合、このプロセスは OPEN コマンドが発生した場合と同じように、暗黙の OPEN で停止します。

（以前の OPEN コマンドによって）プロセスが所有しない他のデバイスに USE コマンドを発行すると、<NOTOPEN> エラーが生成されます。

timeout を設定していない OPEN コマンドは、プロセスがデバイスを取得した場合にのみ、プロセスに制御を返します。キーボードから Ctrl-C などの割り込みコマンドを入力することで、OPEN コマンドの実行に割り込むことができます。保護問題や無効なデバイス名が原因で OPEN コマンドが失敗すると、停止した状態が続きます。OPEN コマンドで timeout を指定すると、OPEN は、timeout の期限が切れたときにプロセスに制御を戻します。

2.5.2 NULL デバイス

2.5.2.1 入出力を転送する NULL デバイスの使用

画面に表示しない無関係の出力がアプリケーションで生成された場合、NULL デバイスにその出力を転送できます。適切な引数を持つ InterSystems IRIS OPEN コマンドを発行して、NULL デバイスを指定します (以下のテーブルを参照してください)。InterSystems IRIS は、そのデバイスをダミー・デバイスとして扱います。

テーブル 2-6: NULL デバイス引数

プラットフォーム	NULL デバイス引数
UNIX®	/dev/null/
Windows	//./nul

この後の READ コマンドは、即座に空の文字列を返します。また、WRITE コマンドの場合は、直ちに成功を返します。いずれも、実データは、読み取りまたは書き込みされていません。NULL デバイスは、UNIX® のシステム呼び出し open、write、read を無視します。

注釈　InterSystems IRIS の外部から NULL デバイスを開く場合、(例えば InterSystems IRIS の出力を UNIX® シェルから /dev/null に転送する場合)、他のデバイスに行われるのと同じように、UNIX® システム・コールが発生します。

2.5.2.2 NULL デバイスを使用するジョブ起動プロセス

あるプロセスが JOB コマンドで別のプロセスを開始すると、ジョブ起動プロセスの既定の主入出力デバイスは、NULL デバイスになります。

2.5.3 プロセスによるデバイスの所有

シーケンシャル・ファイルを除き、1 つのプロセスが一度に所有できるデバイスは 1 つだけです。

つまり、プロセスがデバイスに OPEN コマンドを正常に発行した後、他のプロセスは、最初のプロセスが解放されるまで、そのデバイスを開くことはできません。以下の方法で、プロセスはデバイスを解放します。

- CLOSE コマンドを明示的に発行
- プロセスの停止

2.6 アプリケーション開発に関する入出力コマンド

InterSystems IRIS ルーチンをロード、編集、印刷、保存するには、入出力コマンドの特別なセットがあります。これらのコマンドは、ルーチンを現在のデバイスからロードし、保存します。以下はその概要です。

テーブル 2-7: アプリケーション開発に関する入出力コマンド

コマンド	説明
ZLOAD [routine]	引数なしの ZLOAD コマンドは、現在のデバイスから InterSystems IRIS ルーチンをロードします。OPEN と USE コマンドと併せて ZLOAD を使用して、異なるデバイスからルーチンを入力あるいは出力できます。ZLOAD は、ターミナル入力から NULL 行を受け取った場合、あるいはファイルの最後に達した場合に終了します。
PRINT [args] または ZPRINT [args]	現在のデバイスに、メモリのルーチンを出力します。ルーチンの最後の行の後に、空の行を書き込みます。オプションの引数により、出力する行数を制御できます。
ZSAVE [routine]	ZSAVE は、メモリ内のルーチンにユーザが指定した名前を付けて、ディスクに書き込みます。名前を指定しない場合、ZLOAD でロードされたルーチン名を使用します。

2.7 デバイス特殊変数

入出力コマンドには、特定のシステム変数値に影響を与えるものもあります。このセクションでは、これらの変数を定義し、その使用目的について説明します。特殊変数は、入出力コマンドが現在のデバイスに発行されたときにのみ、変更されます。以下のテーブルは、デバイス特殊変数の概要です。

テーブル 2-8: デバイス特殊変数

変数	目的
$IO	すべての出力処理が対象となる現在のデバイスのデバイス ID を示します。InterSystems IRIS は、ログイン時に $IO の値を主出力デバイスに設定します。この値は、USE コマンドと CLOSE コマンド、BREAK コマンドによって、またはプログラマ・モードに制御が戻ったときにのみ変更できます。
$X	現在のデバイスで最後のキャリッジ・リターン以降に記述された、実行中の出力可能文字の総数です。この範囲は、0 からデバイスの幅までです。
$Y	現在のデバイスで最後の改ページ以降に記述された、実行中の改行の総数です。この範囲は、0 からデバイスの長さまでです。
$ZA	ターミナル・デバイスへ READ コマンドを発行した後の READ の状態情報です。
$ZB	現在のデバイスで最後の READ 操作を終了させた文字シーケンスあるいはイベントです。
$ZMODE	現在のデバイスに対し、OPEN コマンド、あるいは USE コマンドと併用したパラメータを示します。

$X と $Y は、出力表示をフォーマットするのに役立ちます。詳細は、“ターミナル入出力” の章を参照してください。$ZA および $ZB に関するデバイス固有情報については、このドキュメントのそれぞれの章を参照してください。

2.8 ニーモニック空間によるデバイス制御

ニーモニック空間は、カーソルの動作やデバイスの属性など、デバイスの制御を実行する InterSystems IRIS ルーチンです。各動作はラベルに対応します。これらのラベルは、WRITE /mnemonic コマンドで使用するニーモニックです。

WRITE /mnemonic 構文についての詳細は、このドキュメントの他の章にある、各デバイス・タイプでの WRITE コマンドの説明を参照してください。

2.8.1 事前定義のニーモニック空間

InterSystems IRIS には、以下のテーブルで示されているような事前定義のニーモニック空間があります。

テーブル 2-9: 事前定義のニーモニック空間

ルーチン名	既定のデバイス・タイプ	説明
^%X364	ターミナル、シーケンシャル・ファイル、他のデバイス	X3.64 (ANSI) ターミナル用ニーモニック空間です。詳細は、“X3.64 用ニーモニック空間”を参照してください。
^%XDTM	DTM PC コンソール	DTM PC コンソール用ニーモニック空間です。詳細は、“DTM PC コンソール用ニーモニック空間”を参照してください。

2.8.1.1 既定のニーモニック空間の設定

管理ポータルで、以下のデバイス・タイプの既定のニーモニック空間を変更できます。[**システム管理**]、[**構成**]、[**デバイス設定**]、[**IO設定**] の順に選択します。次のニーモニックが表示されます。

- ターミナル
- シーケンシャル・ファイル
- その他

既定のニーモニック空間を定義した後は、現在のデバイスについて OPEN コマンドまたは USE コマンドでニーモニック引数を指定して既定のニーモニック空間をオーバーライドしない限り、`WRITE /mnemonic` コマンドが発行された場合に、現在のデバイスでは既定のニーモニック空間にある制御ニーモニックが使用されます。

2.8.2 ニーモニック空間の生成

独自のニーモニック空間ルーチンを生成できます。例えば、ターミナル入出力用に独自のニーモニック空間を生成できます。

1. 必要な制御ニーモニックを含む InterSystems IRIS ルーチンを生成します。このルーチンでは、以下の点に注意してください。
   - このルーチンのエントリ・ポイントは必ず大文字です。これらのエントリ・ポイントは、WRITE /mnemonic コマンドで参照するニーモニックです。
   - エントリ・ポイントには、引数が必要なものもあります。ニーモニック空間のコードは、エントリ・ポイントで現在のデバイスに対して実行します。
   - カーソル移動ルーチンは、画面を超えてカーソルを動かしたり、カーソルを折り返すことはできません。
2. このニーモニック空間をすべてのユーザから使用できるようにするには、“%”で始まる InterSystems IRIS ルーチン名を付け、システム・マネージャのネームスペース (%SYS) に置きます。

2.8.3 ニーモニック空間の選択

デバイスに WRITE /mnemonic コマンドを発行する前に、そのデバイス・タイプに対して管理ポータルの構成設定で指定されている既定のニーモニック空間を使用するかどうかを決定します。

- 既定のニーモニック空間を使用する場合、デバイスに対して OPEN や USE コマンドを発行するときに、mnespace パラメータを組み込まないようにする必要があります。
- 別のニーモニック空間を使用するには、デバイスに発行する OPEN や USE コマンドの mnespace パラメータにその名前を指定します。

ObjectScript

```
USE "device"::"^%X364"
```

mnespace パラメータの使用方法についての詳細は、OPEN コマンドと USE コマンドと、個々のデバイス・タイプに関する章を参照してください。

3
ターミナル入出力

この章では、InterSystems IRIS® Data Platform のターミナル入出力について説明します。

3.1 ターミナル入出力機能の概要

ObjectScript は、直列の非同期 ASCII ターミナルをサポートするコマンドを提供します。また、これらのコマンドをコンソール入出力で使用できます。

ターミナル入出力を使用すると、ルーチンで以下を実行できます。

- 入力文字のエコーを有効、あるいは無効にできます。
- ANSI 標準のエスケープ・シーケンスを送受信できます。
- キーボード割り込みを制御し、特有のユーザ対話用プログラムを生成できます。これには、フォーマット画面、反転映像、フィールドをスキップする特殊キーなどがあります。
- Ctrl-C 割り込みを有効、あるいは無効にできます。
- XON (Ctrl-Q) と XOFF (Ctrl-S) により、入出力データのフローを制御できます。
- COM ポートの状態パラメータおよびモデムのボー・レートを指定します。
- 独自の終端文字を指定したとき、外部プロトコルに適合させることができます。
- 自動装置のような非ターミナル・デバイスと通信できます。

プリンタは、ほとんどのプラットフォームでターミナル入出力デバイスとして処理されます。UNIX® システムでは、常にプリンタがターミナル入出力デバイスとして処理されます。Windows では、シリアル通信ポートを通じて接続されているプリンタは、ターミナル入出力デバイスとして処理されます。それ例外のプリンタは、Windows システムではシーケンシャル・ファイル入出力デバイスとして処理されます。詳細は、このドキュメントの "プリンタ" の章を参照してください。

3.1.1 主デバイスとなるログイン・ターミナルとコンソール

InterSystems IRIS へのログインに使用したターミナルやコンソールは、主デバイスです。ユーザ側で主デバイスを開く必要はありません。OPEN と USE が発行されていない場合、プロセスは最初にREAD または WRITE を発行します。`OPEN 0 USE 0` が明示的に発行されたかのように、システムは自動的に主デバイスを開き、現在のデバイスとして確立します。

注釈　この章でいうターミナルとは、ターミナルとコンソールの両方を指します。

3.2 入出力状態を表示する特殊変数

入出力コマンドは、特殊変数値に影響します。これらの変数をテストすることで、入出力条件を判断できます。

- $IO は、現在のデバイス名を含みます。
- $TEST は、直前に実行された処理が成功したかどうかを示すブーリアン値を含みます。
- $X と $Y は、カーソル位置を示します。
- $ZA、$ZB、$KEY は READ 処理の情報を示します。$ZB と $KEY は類似していますが、同じものではありません。

デバイスに依存しない $IO 特殊変数の詳細は、“入出力デバイスとコマンド”の章を参照してください。次のセクションでは、その他の特殊変数に関しターミナル固有の情報を説明しています。

3.2.1 $X と $Y とカーソル位置

カーソルや印字ヘッドについて、$X は水平方向の位置情報、$Y は垂直方法の位置情報を持ちます。`$X=0,$Y=0` は、CRT 画面あるいは印刷ページの上部左端を意味します。$X と $Y の両方について、InterSystems IRIS ではそのモジュロ 256 が計算されます。つまり、範囲は 0 から 255 で、その後は再度 0 から始まります。

以下のテーブルは、入力した文字とその入力に対する画面上の応答 (エコー) を示しています。

テーブル 3-1: 文字エコーの効果

文字	ASCII コード	$X に対する効果	$Y に対する効果
Form Feed	12	$X=0	$Y=0
Return	13	$X=0	$Y=$Y
Line Feed	10	$X=$X	$Y=$Y+1
Backspace	8	$X=$X-1	$Y=$Y
Tab	9	$X=$X+1	$Y=$Y
出力できる ASCII 文字	32 ～ 126	$X=$X+1	$Y=$Y

OPEN と USE の S プロトコルはエコーを無効にします。また、入力中の $X と $Y の変更も無効にします。したがって、これらの変数はカーソルの真の位置を示します。

3.2.1.1 WRITE * および $X と $Y

WRITE * は、$X と $Y を変更しません。したがって、ターミナルに制御シーケンスを送信しても、$X と $Y は真のカーソル位置を示します。制御シーケンスの中には、カーソルを動かすものもあるため、必要なときに、$X あるいは $Y を直接設定できます。

3.2.1.2 $X と $Y の例

以下の例では、制御シーケンスは、VT100 ターミナルのカーソルを 10 行、20 列目に移動し、それに応じて $X と $Y を設定します。

ObjectScript

```
 ; set DY and DX to desired
 ; values for $Y and $X
 SET DY=10
 SET DX=20
 ; ...
 ; escape sequence moves
 ; cursor to desired position
 WRITE *27, *91, DY+1, *59, DX+1, *72
 ; ...
 ; updates $X and $Y
 SET $Y=DY
 SET $X=DX
```

3.2.1.3 $X と $Y に対するエスケープ・シーケンスのさまざまな影響

エスケープ・シーケンスは、$X と $Y の値に対する効果を変更できます。この効果を制御する要因には、以下の 3 つがあります。

- 既定の動作を設定するオペレーティング・システム
- /NOXY ($X および $Y の処理を無効化) が、OPEN コマンドまたは USE コマンドに指定されたかどうか
- %SYSTEM.Process クラスの DX() メソッドを使用すると、$X による現在のプロセスのエスケープ・シーケンスの処理方法を設定できます。システム全体の既定の動作は、Config.Miscellaneous クラスの DX プロパティで設定できます。

$X と $Y に対するエスケープ・シーケンスの影響 (Windows システムおよび UNIX® システム)

既定の UNIX® と Windows では、ASCII のエスケープ文字 (10 進数値 27) を含む文字列を入力したときやそれに対するエコーが表示されるとき、他の文字シーケンスと同様に、InterSystems IRIS によって $X と $Y が更新されます。したがって、ターミナルで動作し、画面には表示されない ANSI 標準の制御シーケンスでは、$X と $Y の値とカーソル位置との関係に矛盾が生じます。

この問題を防ぐ一番簡単な方法は、その動作を変更する DX() メソッドを使用することです (次セクション参照)。あるいは `Write *` 文の文字列で、各文字の ASCII 値を使用できます。

制御シーケンスの例

以下のコードを考えます。

```
%SYS>WRITE $CHAR(27)_"[lm"
```

このコードを使用する代わりに、$X と $Y を更新しない以下の文を使用できます。

```
%SYS>WRITE *27,*91,*49,*109
```

エスケープ・シーケンスに対する $X の更新制御の変更

実際のプロセスで、既定ではない方法で $X を更新するには、%SYSTEM.Process クラスの DX(n) メソッドを発行します。

システム管理者は、Config.Miscellaneous クラスの DX プロパティを設定することによりシステム全体の既定の動作を変更できます。

いずれの場合も、n には以下のような 0 から 4 までの値を指定します。

値	$X を更新するための既定の動作
0	InterSystems IRIS の既定値
1	DSM による動作
2	DTM/MSM による動作

詳細は、"ObjectScript ランゲージ・リファレンス" の "$X" を参照してください。

3.2.2 タイムアウト付き処理結果を示す $TEST

$TEST 特殊変数は、タイムアウト値を持つコマンドで設定されます。これらのコマンドには OPEN およびREAD があります。$TEST の値は、1 あるいは 0 に設定されます。

- タイムアウトになる前に時間制限コマンドが成功した場合、$TEST には 1 が設定されます。
- 時間制限コマンドがタイムアウトになった場合、$TEST には 0 が設定されます。

注釈　タイムアウトなしの OPEN および READ コマンドは、$TEST に影響を与えません。

詳細は、"ObjectScript ランゲージ・リファレンス" の "$TEST" を参照してください。

3.2.3 READ のステータスを示す $ZA

$ZA 特殊変数にはビット・フラグがあり、現在のデバイスで最後に実行された READ のステータスを示します。ユーザは $ZA を設定できず、InterSystems IRIS がその値を制御します。$ZA の値は、次の READ が実行されるまで有効な状態で保持されます。以下のテーブルのように、$ZA 値には、この変数をプログラムがどのようにテストするのかを示す合計値が含まれます。($ZA 値には、モデム接続ステータス用のビット・フラグが含まれますが、ここに記載されていません。$ZA ビット・フラグの値の全リストについては、"ObjectScript ランゲージ・リファレンス" の "$ZA" を参照してください。)

テーブル 3-2: $ZA の Read ステータス値

値	テスト	意味
1	$ZA#2	ブレークが有効かどうかに関係なく、Ctrl-C を受信しました。
2	$ZA¥2#2	READ がタイムアウトになりました。
256	$ZA¥256#2	InterSystems IRIS が無効なエスケープ・シーケンスを検出しました。
512	$ZA¥512#2	ハードウェアがパリティ・エラーもしくはフレーミング・エラーを検出しました。

$ZA がエラーを示す状況は数多くありますが、$ZTRAP 特殊変数にトラップすることによるプログラム・フローへの割り込みは発生しません。エラーに関連するプログラムは、READ を実行するたびに $ZA を検証する必要があります。当然、ブレークを有効にする Ctrl-C は、$ZTRAP にトラップします。エラー・トラップと $ZTRAP の詳細は、"ObjectScript の使用法" の "エラー処理" の章と "ObjectScript ランゲージ・リファレンス" の "$ZTRAP" を参照してください。

3.2.4 READ の終了原因を示す $ZB

$ZB は、現在のデバイスで最後の READ 処理を終了させた文字シーケンスあるいはイベントを示します。ユーザ側で $ZB を設定することはできません。READ を実行するたびに、InterSystems IRIS によって $ZB の値が設定されます。この値を使用して、上向き矢印やファンクション・キーなど、出力されない文字を操作できます。

$ZB には、以下のいずれかが含まれます。

- キャリッジ・リターンなどの終端文字
- エスケープ・シーケンス
- 固定長 READ x#y の y 番目の文字
- READ *x の 1 文字
- 時間制限つきの READ がタイムアウトになったときの空文字列

$ZB に含まれる文字は、64 文字以下です。それを超える長さのエスケープ・シーケンスは無効です。

3.2.4.1 $ZB の例

以下の例は、ユーザ定義の入力文字を READ コマンド変数 x に割り当て、入力ターミネータ (通常は Enter) を $ZB 特殊変数に割り当てます。ターミナル・プロンプトからこのコマンドを発行する場合、READ コマンドと同じコマンド行で、$ZB 値をトラップする変数を設定する必要があります。コマンド行の作成に使用する改行は、終端文字として $ZB に記述されるからです。この例では、ZZDUMP を使用して $ZB でトラップされる文字の値を表示します。

```
USER>READ x SET y=$ZB
USER>ZZDUMP y

0000: 0D
USER>
```

3.3 OPEN コマンドと USE コマンド

3.3.1 OPEN コマンド

ターミナルの所有権を構築します。オプションのパラメータ・リストでは、右マージンの設定、デバイス・プロトコルの指定、および 1 つ以上の終端文字の指定が可能です。以下のパラメータ・リストに従って、timeout 引数または mnespace 引数 (またはその両方) を必要に応じて指定できます。mnespace 引数は、`WRITE /mnemonic` で使用する制御ニーモニックを定義している InterSystems IRIS ルーチンを指定します。

OPEN は、デバイスが完全に開くまで、プロセスを休止します。Ctrl-C を押して OPEN コマンドに割り込むと、<NOTOPEN> エラーを生じます。

OPEN は、タイムアウトを指定しない限り、デバイスが開くまで制御を保持します。タイムアウトを設定している場合、指定した時間内にデバイスが開かないと $TEST が 0 に設定され、制御はプロセスに戻ります。オペレーティング・システム・レベルで無効なデバイスであっても、OPEN は、そのデバイスが正常に開くまで、またはタイムアウトになるまでデバイスの取得を続けます。

3.3.1.1 OPEN 構文

OPEN コマンドは、以下の引数をとります。

```
OPEN terminal:(margin:protocols:terminator:portstate:baud):timeout:"mnespace"
```

terminal 引数のみが必須です。terminal 引数には、値がターミナル・デバイス名となる式を指定することもできます。ゼロ (0) はプロセスの主デバイスを示します。$IO は現在のデバイスです。terminal の最大長は 256 文字です。

複数の引数はコロン (:) で区切ります。リスト内の引数を省略するには、プレースホルダとしてコロンを指定する必要があります。ただし、末尾にはコロンを使用できません。コマンドまたはパラメータ・リストの末尾にコロンを記述しないでください。

オプションのパラメータ・リストは括弧で囲み、次のオプション・パラメータを記述できます。

- margin は、右マージンを指定することにより行ごとの文字数を指定する整数です。
- protocols は、ターミナルのオプションを指定する 1 文字以上のコードです。
- terminator は、READ 処理を終了する 1 文字以上の文字列です。これらの文字は、特定の protocols に対して定義する終端文字を補足します。
- portstateは、COM ポート状態を指定する文字列です。
- baud は、COM ポートのボー・レートを指定する整数です。

これらのオプションのパラメータは、記述した順序による位置パラメータまたは /KEYWORD=value 構文によるキーワード・パラメータとして指定できます。キーワード・パラメータは、任意の順序で指定できます。InterSystems IRIS では、パラメータが左から右の順に実行されるので、パラメータ間の相互関係により優先の順序が決まる場合もあります。位置パラメータとキーワード・パラメータを同じパラメータ・リスト内で混在して使用することもできます。複数のパラメータを含む場合、それらを囲む括弧は必須です。

以下のパラメータ・リストは同じものです。

ObjectScript

```
OPEN $IO:(80:"BFU":$CHAR(13))
 ; all positional
OPEN $IO:(80::$CHAR(13):/PARAMS="BFU")
 ; mixed positional and keyword, using the /PARAMS keyword
 ; to specify a protocol letter code string.
OPEN $IO:(/MARGIN=80:/TERMINATOR=$CHAR(13):/BREAK:/FLUSH:/UPCASE)
 ; all keyword, using separate keywords
 ; for each protocol letter code.
```

パラメータ・リスト (パラメータ・リストを指定していない場合は代替のコロン) の後に、timeout の秒数をオプションで指定できます。また、このデバイスの制御ニーモニックを保持するルーチンを指定する mnespace 引数を指定できます。

詳細は、"ObjectScript ランゲージ・リファレンス" の "OPEN" を参照してください。

3.3.2 USE コマンド

指定されたターミナルを現在のデバイスにします。プログラマ・モードでは、同じコード行にある連続したすべての入出力コマンドが、そのデバイスを対象とします。アプリケーション・モードでは、USE コマンドで指定したデバイスが、次の USE コマンドまで現在のデバイスとなります。

3.3.2.1 USE 構文

USE コマンドは、以下の引数をとります。

```
USE terminal:(margin:protocols:terminator):"mnespace"
```

terminal 引数には、値がターミナル・デバイス名となる式を指定することもできます。ゼロ (0) はプロセスの主デバイスを示します。$IO は現在のデバイスです。terminal の最大長は 256 文字です。

複数の引数はコロン (:) で区切ります。引数を省略するにはコロンを指定する必要があります。コマンドまたはパラメータ・リストの末尾にはコロンを記述しないでください。

オプションのパラメータ・リストは括弧で囲み、margin パラメータ、protocols パラメータ、および terminator パラメータを記述できます。オプションの margin パラメータ、protocols パラメータ、および terminator パラメータは、記述した順序による位置パラメータまたは /KEYWORD=value 構文によるキーワード・パラメータとして指定できます。キーワード・パラメータは、任意の順序で指定できます。InterSystems IRIS では、パラメータが左から右の順に実行されるので、パラメータ間の相互関係により優先の順序が決まる場合もあります。位置パラメータとキーワード・パラメータを同じパラメータ・リスト内で混在して使用することもできます。複数のパラメータを含む場合、それらを囲む括弧は必須です。

USE コマンドで COM ポート状態とボー・レートを指定するには、適切なキーワード・パラメータを使用します。

パラメータ・リスト (パラメータ・リストを指定していない場合は代替のコロン) の後に、mnespace 引数をオプションで指定できます。これは、`WRITE /mnemonic` で使用する制御ニーモニックを定義している ObjectScript ルーチンを特定します。

詳細は、"ObjectScript ランゲージ・リファレンス" の "USE" を参照してください。

3.3.3 OPEN コマンドと USE コマンドの位置パラメータ

以下の位置パラメータは、OPEN コマンドと USE コマンドで使用できます。デバイスに対するこれらのパラメータは OPEN コマンドと USE コマンドのいずれでも設定できます。あるいは、管理ポータルで構成した既定の設定を使用することもできます。これらのパラメータは位置を示します。パラメータを省略すると、その代わりとして、前にコロンを置く必要があります。

3.3.3.1 margin

第 1 位置パラメータ：右マージンを指定する整数値で、これによって 1 行の文字数が決まります。1 から 255 までの値が出力用の右マージンとして設定でき、他の値は無効となります。空文字列を指定すると、マージンの設定は変更されません。Windows プラットフォームでは、“:n” を使用して出力マージンを制御することはできません。そのため、InterSystems IRIS ではこのような記述は無視されます。例えば、“|PRN| :121” のようなコードは無視されます。出力幅を制御するには、プリンタに適切な制御文字を送信します。これは、他のプラットフォームにも当てはまります。

さまざまなターミナル・タイプの既定マージンは、管理ポータルで定義します。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[デバイスサブタイプ]** の順に選択します。リストされている各デバイス・サブタイプの “[編集]” をクリックすると、**[右マージン :]** の既定のオプションが表示されます。

3.3.3.2 protocols

第 2 位置パラメータ：引用符で囲まれた文字コードの文字列です (例えば ″BNFU″)。それぞれの文字により、ターミナルの通信規則の 1 つが有効になります。文字コードは、大文字と小文字を区別しません。文字コードは、任意の順序で指定できます。InterSystems IRIS では、左から右の順に実行されるので、文字コード間の相互関係により優先の順序が決まる場合もあります。文字コードのテーブルは ″文字コード・プロトコル″ を参照してください。

文字列の前にプラス記号やマイナス記号を記述すると、プロトコルは以下のような影響を受けます。

- 前にプラスやマイナスがない場合 ： 新規の文字列で既存のプロトコル文字列が置換されます。
- 文字コード文字列の前にプラス (+) を記述した場合：既存のプロトコル文字列に新規の文字列のプロトコルが追加されます。
- 文字コード文字列の前にマイナス (-) を記述した場合 ： 新規の文字列のプロトコルは無効になりますが、他のプロトコルは引き続き有効です。

プロトコルを有効にする + オプションと無効にする - オプションは、DSM-11 互換モードでは使用できません。

3.3.3.3 terminator

第 3 位置パラメータ：READ を終了する文字を含む、最大 8 文字の文字列です。プロトコル文字列に使用しているターミネータに、これらのターミネータが付加されます。″入出力操作を終了するためのターミネータ″ を参照してください。

3.3.3.4 portstate

第 4 位置パラメータ：COM ポート状態を制御する最大 8 バイトの文字列で、それぞれのバイト位置に制御内容が割り当てられています。portstate バイトは、以下のとおりです (バイトは左から右の順に 1 から番号が付けられます)。

バイト	意味	値
1	切断	D = ポートを切断 (停止) する。空白 = ポートを切断しない。
2	モデム制御	1 = モデム制御を使用する。0 = モデム制御を使用しない。空白 = モデム制御を変更しない。
3	データ・ビット	5 = 5 データ・ビット。6 = 6 データ・ビット。7 = 7 データ・ビット。8 = 8 データ・ビット。空白 = データ・ビット設定を変更しない。

バイト	意味	値
4	パリティ	0 = パリティなし。0 = 奇数パリティ。2 = 偶数パリティ。3 = マーク・パリティ。4 = スペース・パリティ。空白 = パリティ設定を変更しない。
5	ストップ・ビット	1 = 1 ストップ・ビット。5 = 1.5 ストップ・ビット。2 = 2 ストップ・ビット。空白 = ストップ・ビット設定を変更しない。
6	フロー制御	X = Xon/Xoff フロー制御を使用する。C = CTS/RTS フロー制御を使用する。D = DSR/DTR フロー制御を使用する。N = フロー制御を無効にする。空白 = フロー制御を変更しない。
7	DTR 設定	0 = DTR を無効にする (オフに設定、オフを保持)。1 = DTR を有効にする (オンに設定、オンを保持)。空白 = DTR 状態を変更しない。
8	$ZA エラー報告	0 = $ZA エラー報告を無効にする (既定)。1 = $ZA エラー報告を有効にする。空白 = $ZA エラー報告を変更しない。

以下の例は、COM ポート状態の文字列です。

ObjectScript

```
OPEN "COM2":(:::" 0801x0")
```

この文字列の値は、次のとおりです。空白 (ポートを切断しない)、0 (モデム制御を使用しない)、8 (8 データ・ビット)、0 (パリティなし)、1 (1 ストップ・ビット)、X (Xon/Xoff フロー制御)、0 (DTR を無効にする)、既定 ($ZA エラー報告を無効にする)。

Disconnect パラメータは、DTR シグナルのレベルを 2 秒間下げた後、元に戻すことで、モデム制御ポートを停止します。この切断によってポートは閉じません。切断後に、COM デバイスを開き直さなくても、再度ダイヤルできます。

Modem Control パラメータは、InterSystems IRIS が RLSD (Received Line Signal Detector) ピンの状態にどのように応答するかを決定します。このピンは、DCD (Data Carrier Detect) ピンとも呼ばれます。回線がモデム制御されている場合 (modem control=1)、InterSystems IRIS は RLSD の状態を監視しています。キャリアが存在しない状態で READ コマンドを発行すると、<ENDOFFILE> エラーが生成されます。キャリアが存在しない状態で WRITE コマンドを発行した場合は、エラーは生成されません。これは、接続が確立する前にモデムにダイヤル・コマンドを送信できる必要があるからです。InterSystems IRIS モデム制御は、いつでも有効 (1) あるいは無効 (0) にできます。モデムにコマンドを送信する際にはモデム制御を無効にし、キャリアが検出されて接続が確立された後はモデム制御を有効にすることをお勧めします。

DTR Setting パラメータは、接続されたモデムからのログインを制御するために使用されます。DTR 設定が 0 (ゼロ) の場合、DTR 制御シグナルはオフになり、モデムはコンピュータと通信できません。この場合は、ダイヤルイン接続できません。DTR 設定が 1 の場合、DTR 制御シグナルはオンになり、モデムはコンピュータと通信できます。これでダイヤルイン接続が可能になります。DTR をオフ (0) に設定している場合、接続されたモデムを使用してダイヤルアウトするためには、OPEN コマンドまたは USE コマンドで DTR をオン (1) に設定する必要があります。多くの場合、NULL モデム・ケーブルを使用して直接ターミナル・デバイスまたはシリアル・プリンタに接続すれば、DTR 設定は重要ではなくなります。これは、NULL モデム・ケーブルによって DTR 制御ピンが強制的にオンになるためです。

$ZA Error Reporting パラメータは、$ZA 特殊変数へのモデム制御ピンの状態の報告を有効にします。このチェックは、COM ポートのモデム制御バイト設定に関係なく実行できます。$ZA エラー報告が有効になっている場合、COM ポート・エラーは、Windows ClearCommError() 関数の呼び出しでクリアされます。ポート・エラー状態は、$ZA ビット 16 から 22 で報告されます。$ZA ビット値のテーブルについては、"ObjectScript リファレンス" の "$ZA" を参照してください。

3.3.3.5 baud

第 5 位置パラメータ : 目的の COM ポートのボー・レートを指定する整数値です。サポートされているボー・レートは、110、300、600、1200、4800、9600、14400、19200、38400、56000、57600、115200、128000、256000 です。

3.3.4 OPEN コマンドと USE コマンドのキーワード・パラメータ

以下のテーブルは、OPEN コマンドと USE コマンドの両方を使用して、ターミナル・デバイスを制御するキーワード・パラメータの説明です。それぞれのキーワードについて、OPEN および USE の対応する文字コード・プロトコルがテーブルに一覧表示されています。これらのプロトコルの使用に関する詳細は、文字コード・プロトコルのテーブルにあります。

テーブル 3-3: ターミナル・デバイスの OPEN と USE のキーワード・パラメータ

キーワード	既定値	文字コード・プロトコル	説明
/BAUD=n			baud 位置パラメータに相当します。/BAUD=n はポートのモデムのボー・レートを設定します。サポートされる値は、オペレーティング・システムのサポートによって異なります。/SPEED は、/BAUD のエイリアスです。
/BREAK[=n] または /BRE[=n]	0	B	/BREAK を指定した場合、または /BREAK=n の n が 0 以外の場合、このプロトコルが有効になります。/BREAK=n の n が 0 の場合、このプロトコルが無効になります。
/COMPARAMS=str			portstate 位置パラメータに相当します (このキーワードは、位置に依存しない方法で COM ポート状態バイト・コード文字列を指定する方法を提供します)。str を指定できる portstate バイト・コードは、この章で前述しているテーブルに一覧表示されています。
/COMPRESS=str	""		ストリーム・データの圧縮タイプを指定します。圧縮タイプ ZLIB または ZSTD を有効にすることができます。/COMPRESS="" を指定すると、圧縮を無効にすることができます。/COMPRESS="zlib" は /GZIP=1 と同等です。文字列を圧縮するには、%SYSTEM.Util.Compress() を使用します。
/CRT[=n]	オペレーティング・システムのターミナル設定に依存します。	C と P	C プロトコルと P プロトコルに関連します。/CRT を指定した場合、または /CRT=n の n が 0 以外の場合、C プロトコルが有効になり、P プロトコルが無効になります。/CRT=n の n が 0 の場合、C プロトコルが無効になり、P プロトコルが有効になります。
/DATABITS=n			シリアル・ポートのデータ・ビット数を設定します。有効な値は、5、6、7、または 8 です。
/DISCONNECT			portstate 位置パラメータの 1 番目のバイトに相当します。/DISCONNECT は COM ポートを切断 (停止) します。これによってポートは閉じません。COM デバイスを開き直さずに、再度ダイヤルアウトできます。

キーワード	既定値	文字コード・プロトコル	説明
/ECHO[=n]	1	S	/ECHO を指定した場合、または /ECHO=n の n が 0 以外の場合、このプロトコルが無効になります。/ECHO=n の n が 0 の場合、このプロトコルが有効になります。
/EDIT[=n]	0	R と N	/EDIT を指定した場合、または /EDIT=n の n が 0 以外の場合、R プロトコルが有効になり、N プロトコルが無効になります。/EDIT=n の n が 0 の場合、R プロトコルが無効になり、N プロトコルが有効になります。
/FLOW=str			シリアル・ポートに使用するフロー制御タイプを指定します。有効な値は NONE と XON です。一部のオペレーティング・システムでは、RTSCTS もサポートされます。
/FLUSH[=n] または /FLU[=n]	0	F	/FLUSH を指定した場合、または /FLUSH=n の n が 0 以外の場合、このプロトコルが有効になります。/FLUSH=n の n が 0 の場合、このプロトコルが無効になります。
/GZIP[=n]	1		GZIP と互換性のあるストリーム・データ圧縮を指定します。/GZIP を指定した場合、または /GZIP=n (n は 0 以外) を指定した場合、WRITE の発行時に圧縮、READ の発行時に解凍が有効になります。/GZIP=0 を指定した場合は、圧縮と解凍が無効になります。/GZIP=0 を発行して、圧縮/解凍を無効にする前に、$ZEOS 特殊変数をチェックして、ストリーム・データの読み込みが実行中でないことを確認してください。/GZIP 圧縮は、/IOTABLE を使用して構築した変換などの入出力変換には影響しません。これは、圧縮がその他すべての変換 (暗号化を除く) の後に適用され、解凍がその他すべての変換 (暗号化を除く) の前に適用されるためです。
/IMAGE[=n] あるいは /IMA[=n]	0	I	/IMAGE を指定した場合、または /IMAGE=n の n が 0 以外の場合、このプロトコルが有効になります。/IMAGE=n の n が 0 の場合、このプロトコルが無効になります。
/IOTABLE[=name] または /IOT[=name]	name が指定されない場合、デバイスの既定の入出力変換テーブルを使用します。		K¥name¥ プロトコルに対応します。これは、デバイスの入出力変換テーブルを確立します。

キーワード	既定値	文字コード・プロトコル	説明
/MARGIN=n または /MAR=n	0 (マージンなし)		ターミナル・デバイスの右マージンを設定する margin 位置パラメータに対応します。
/MODE=n	既定なし		n の値に従って、プロトコルをリセットしてターミナル・モードを設定します。 n=0 は、LF と ESC を既定のターミネータに設定します。 n=1 は、モード 0 と同様、S プロトコルを有効にします。 n=2 は、モード 0 と同様、T プロトコルを有効にします。
/NOXY [=n]	0		$X および $Y の処理なし : /NOXY を指定した場合、または /NOXY=n (n は 0 以外の値) を指定した場合、$X および $Y の処理が無効になります。これにより READ 操作および WRITE 操作のパフォーマンスを大幅に向上させることができます。$X および $Y の変数値が不確定であるため、マージン処理 ($X に依存) は無効になります。/NOXY=0 の場合は、$X および $Y の処理が有効になります。これが既定です。
/OBUFSIZE=nnn	256		ターミナル出力バッファのサイズをバイト単位で指定します。出力バッファのサイズを大きくすると、待ち時間の長いワイド・エリア・ネットワークを通じた Telnet の画面描画パフォーマンスが向上することがあります。/OBUFSIZE の有効値は 256 から 65536 です。既定値は、256 です。
/PARAMS=str または /PAR=str	既定なし		protocols 位置パラメータに相当します (このキーワードは、位置に依存しない方法でプロトコル文字コード文字列を指定する方法を提供します)。str に含めることができる文字コードのテーブルは "文字コード・プロトコル" を参照してください。
/PARITY=str			シリアル・ポートのパリティ・チェック・タイプを指定します。有効な値は NONE、EVEN、ODD です。一部のオペレーティング・システムでは、MARK と SPACE もサポートされます。
/SPEED=n			/SPEED は、/BAUD のエイリアスです。
/STOPBITS=n			シリアル・ポートのストップ・ビット数を設定します。有効な値は 1 または 2 です。

キーワード	既定値	文字コード・プロトコル	説明
/TERMINATOR=str または /TER=str	既定なし		ユーザ定義のターミネータを構築する terminator 位置パラメータに相当します。str を構成するには、"入出力操作を終了するためのターミネータ" を参照してください。
/TPROTOCOL[=n] または /TPR[=n]	0	T	/TPROTOCOL を指定した場合、または /TPROTOCOL=n の n が 0 以外の場合、このプロトコルが有効になります。/TPROTOCOL=n の n が 0 の場合、このプロトコルが無効になります。
/TRANSLATE[=n] または /TRA[=n]	1	K	/TRANSLATE を指定した場合、または /TRANSLATE=n の n が 0 以外の場合はデバイスの入出力変換が有効になります。/TRANSLATE=n の n が 0 の場合は、デバイスの入出力変換が無効になります。
/UPCASE[=n] または /UPC[=n]	0	U	/UPCASE を指定した場合、または /UPCASE=n の n が 0 以外の場合、このプロトコルが有効になります。/UPCASE=n の n が 0 の場合、このプロトコルが無効になります。
/XYTABLE[=name] または /XYT[=name]	name が指定されていない場合、デバイスの既定の $X/$Y アクション・テーブルを使用します。	Y¥name¥	Y¥name¥ プロトコルに対応します。これは、デバイスの $X/$Y アクション・テーブルを確立します。

3.3.5 OPEN コマンド実行の検証

OPEN コマンドが成功したかどうかを判断するために、コードでは $TEST または $ZE (またはその両方) の使用をお勧めします。$TEST は、OPEN コマンドが timeout 引数で指定された場合にのみ設定されます。Ctrl-C が OPEN コマンドに割り込んだ場合にのみ、<NOTOPEN> エラーが発生します。したがって、コードは <NOTOPEN> エラーに依存していないことが必要です。

3.3.6 OPEN と USE の文字コード・プロトコル

特別な状況やターミナルでは、異なるプロトコルが必要になる場合があります。protocols 文字コード・パラメータ (あるいは対応するキーワード・パラメータ) を使用し、InterSystems IRIS がターミナルでやり取りするための規則を変更できます。プロトコルは、通常の読み取りや単一文字の読み取りに影響します。

すべての特殊プロトコルが無効になっている通常モードで、大半のターミナル入出力には十分です。通常モードでは、InterSystems IRIS は、受信する各 ASCII 文字をエコーし、ターミナルで表示するよう返送します。Return キーあるいは有効なエスケープ・シーケンスは、READ コマンドを終了します。

ターミナルに OPEN を発行すると、+ オプションまたは - オプションで指定されたプロトコル以外の既存のプロトコルはすべて無効になります。

以下のテーブルは、有効な protocols 文字とその効果の説明です。

テーブル 3-4: OPEN と USE の文字コード・プロトコル

プロトコル文字	名前	定義
B	BREAK	ブレークが有効の場合 (+B)、Ctrl-C は、実行中のルーチンに <INTERRUPT> エラーを生成して割り込みます。ブレークが無効の場合 (-B)、Ctrl-C キーを押しても割り込みが発生せず、“^C” は表示されません。このプロトコルの使用は、以下のように、ログイン・モードで確立される BREAK コマンドの既定に依存します。 プログラマ・モードでログインすると、割り込みが必ず有効になります (BREAK 1)。OPEN コマンドまたは USE コマンドで指定した B (または /BREAK) プロトコルは何の影響も与えません。 アプリケーション・モードでログインすると BREAK 0 が既定になり、OPEN コマンドまたは USE コマンドで指定した B (または /BREAK) プロトコルで割り込みを有効または無効にできます。
C	CRT ターミナル	C モードでは、ASCII 表記で 3、8、10、13、21、および 127 の 6 つを除くすべての 8 ビット文字をデータとして使用できます。Delete 文字である 127 (ASCII 表記) は BackSpace キーと同じ動作を実現します。つまり、画面上で現在の位置の前にある 1 文字を消去します。ASCII 表記で 21 (Ctrl-U) は、BackSpace キーと同じ機能を提供して文字を消去しながら READ の開始位置までカーソルを移動します。右マージンの設定またはターミナルの種類に従って強制的に新しい行が始まっている場合、Ctrl-U は、物理的に最後の行にある文字のみを消去できます。いずれの場合も、Ctrl-U は、READ 以降のすべての入力をキャンセルします。C は、P プロトコルと相互に排他的です。
F	フラッシュ	READ 以前の入力バッファをフラッシュ (空に) します。入力バッファをフラッシュし、ユーザが READ 処理の前にターミナルに入力しないようにできます。InterSystems IRIS は、READ 処理間の入力を破棄するからです。F プロトコルに関係なく、`WRITE *-1` コマンドは、いつでも入力バッファをフラッシュします。
I	イメージ・モード	I モードでは、terminator パラメータで明示的に指定したターミネータを除き、256 個すべての 8 ビット文字をデータとして使用できます。READ を終了する機能を実現する文字はありません。ターミネータを明示的に指定しない場合、単一文字および固定長の READ 処理のみを使用する必要があります。ターミネータを定義せずに通常どおりに READ を実行すると、<TERMINATOR> エラーが発生します。 イメージ・モード (I) の protocol は、別の protocol 文字と結合できます。イメージ・モードでは、プロトコル文字 P、C、および B は無視されますが、F、N、R、S、および T は機能します。イメージ・モードでない場合、デバイスは N (通常) モードです。
K¥name¥ または Knum	入出力変換モード	K プロトコルをデバイスで使用する際に、システム全体で変換が使用できる場合、入出力変換がデバイスで発生します。テーブル名を指定することで、変換の基本となっている既存の定義済みテーブルを特定します (テーブルをロードするスロット数の “num” を示す従来の Knum は廃止されましたが、現在もサポートされています)。
N	通常モード	N モードでは、ASCII 表記で 3、8、10、13、21、および 127 の 6 つを除くすべての 8 ビット文字をデータとして使用できます。これらの暗黙のターミネータおよび制御文字を編集するコマンド行については、この章の後半で説明します。R (行呼び出し) プロトコルが有効の場合、N モードを指定すると R プロトコルが無効になります。protocols を指定していない場合は、このモードが既定になります。

プロトコル文字	名前	定義
P	出力デバイス	ASCII の Delete 文字は、バックスラッシュ (¥) としてエコーし、Ctrl-U は、“^U” に続くキャリッジ・リターンと改行をエコーします。ターミナルに OPEN コマンドを発行すると、InterSystems IRIS は自動的にプロトコル C あるいは P を実行します。これは、オペレーティング・システムのターミナル設定に依存します。これらのプロトコルは、プロトコルを明示的なデバイスに変更するまで有効です。C および P 以外のプロトコル文字列は、このプロトコルをキャンセルしません。
R	行呼び出しモードを有効にする	R プロトコルは、デバイスの行呼び出しモードを有効にします。現在のプロセスの行呼び出しをアクティブにするには、%SYSTEM.Process クラスの LineRecall() メソッドを使用します。システム全体の行呼び出しを既定に設定するには、Config.Miscellaneous クラスの LineRecall プロパティを使用します。指定デバイスに対して、R プロトコルがこれら既定の設定よりも優先して適用されます。既に開いているデバイスの行呼び出しモードを変更するには、そのデバイスに対し、もう一度 OPEN コマンドで明示的に指定する必要があります。行呼び出しモードは、N プロトコルを指定することで無効になります。
S	隠し入力	READ に何もエコーしません。READ コマンドは、$X と $Y を変更しません。行呼び出しモードは無効です。
T	ターミネータ	T モードでは、どの制御文字もデータとして扱われません。制御文字は、10 進数値の 0 から 31 および 127 から 159 の ASCII 文字です。これらの制御文字のほとんどが READの ターミネータ文字として扱われます。例外は、ターミネータ以外の制御操作を実行する ASCII 3 (Ctrl-C)、ASCII 8 (バックスペース)、ASCII 17 (Ctrl-Q)、ASCII 19 (Ctrl-S)、ASCII 21 (Ctrl-U)、ASCII 24 (Ctrl-X)、ASCII 27 (ESC)、および ASCII 127 (DEL) の各制御文字です。 T モードと I モードを組み合わせると (IT プロトコル)、すべての制御文字 (ASCII 0 から 31 および 127 から 159) は、出力制御文字である Ctrl-Q (XOFF) と Ctrl-S (XON) および入力制御文字である Ctrl-H と Ctrl-Y を除き、READ のターミネータ文字として扱われます。出力制御文字である Ctrl-Q と Ctrl-S は、大半のターミナルで遮断されるため、IT モードの場合も READ を終了しません。
U	大文字モード	U モードでは、すべての入力文字が大文字に変換されます。
Y¥name¥ または Ynum	$X/$Y アクション・モード	デバイスに Y プロトコルを使用する場合、システムは、$X/$Y というアクション・テーブルを使用します。テーブル名を指定することで、変換に使用された事前定義の $X/$Y アクション・テーブルを識別します。名前がわからない場合、システム管理者に問い合わせてください。$X/$Y アクションは常に使用できます。Y が指定されず、システム既定の $X/$Y が定義されていない場合、組み込みの $X/$Y アクション・テーブルを使用します。+ オプションは、Y プロトコルを有効にし、- は無効にします。$X/$Y の関連を無効にするには、コマンド USE 0:(:"Y0") を発行します (テーブルをロードするスロット数であるnum を使用する従来の Ynum については記述されていませんが、現在もサポートされています)。

3.3.6.1 プロトコル文字列の例

以下の一連の例は、プロトコル文字列の使用方法を示しています。以下の各 USE コマンドは、その前に実行した USE コマンドで構築したプロトコル上で実行します。

ObjectScript

```
USE 0:(80:"BP" )
```

文字コード BP は、B と P プロトコルを有効にします。この例はブレーク (B) を有効にし、ターミナルを出力デバイス (P) として処理するよう InterSystems IRIS に命令します。

ObjectScript

```
USE 0:(80:"P")
```

上述の例にある USE コマンドに続いてこのコマンドを実行すると、P プロトコルは有効のままで、B プロトコルが無効になります。

ObjectScript

```
USE 0:(80:"+R" )
```

+R は、他のプロトコル設定に影響せずに、行呼び出しモードを有効にします。

ObjectScript

```
USE 0:(80:"")
```

空の文字列は、すべてのプロトコルを無効にします。しかし、P あるいは C は有効のままです。

ObjectScript

```
USE 0:(80)
```

プロトコル・パラメータを省略すると、プロトコルと明示的なターミネータは、変更されない状態のままとなります。

3.3.7 プロトコル・ターミネータ文字

OPEN プロトコルと USE プロトコル は、READ 入力文字、制御シーケンス、およびキーストロークのうち、暗黙のターミネータ文字として処理される文字を定義します。これには、I (イメージ・モード)、N (通常モード (既定))、R (行呼び出しモード)、および T (ターミネータ・モード) の 4 種類のプロトコルがあります。

- I (イメージ・モード) では、256 個すべての 8 ビット文字をデータとして使用できます。READ 入力ターミネータやコマンド行編集文字として扱われる文字はありません。このため、イメージ・モードでは、単一の文字または固定長の READ 処理のみを使用する必要があります。ターミネータを定義せずに通常どおりに READ を実行すると、<TERMINATOR> エラーが発生します。
- N (通常モード) および C (CRT モード) では、ASCII の 3、8、10、13、21、および 127 の 6 つを除くすべての文字をデータとして使用できます。ASCII の 10 (改行) と 13 (キャリッジ・リターン) の2 つは、READ を終了し、入力を送信します。ASCII 3 (Ctrl-C) は、入力を破棄し、BREAK が有効になっている場合は <INTERRUPT> エラーを発行します。ASCII の 8 (バックスペース) および 127 (削除) は、画面上で後退して 1 文字を消去してから READ を続行します。ASCII の 21 は、後退して前にあるすべての文字を消去してから READ を続行します。
- R (行呼び出しモード) では、ASCII の 1 から 8、10 から 14、16、18、21、23、24、27 および 127 の 20 個を除くすべての文字をデータとして使用できます。ASCII の 10 (改行) と 13 (キャリッジ・リターン) は、READ を終了し、入力を送信します。ASCII 3 (Ctrl-C) は、入力を破棄し、BREAK が有効になっている場合は <INTERRUPT> を発行します。他の文字は、以下のコマンド行編集機能を実行します。

```
1  ^A = beginning of line
2  ^B = back word
3  ^C = interrupt
4  ^D = delete current character
5  ^E = end of line
6  ^F = forward word
7  ^G = delete to beginning of word ("wipe word backward")
8  ^H = BS = destructive backspace
9  ^I = HT = horizontal tab (echoed as a SPACE)
10 ^J = LF = end of input
11 ^K = VT = forward character
12 ^L = FF = erase to end of line
```

```
13  ^M = CR = end of input (same as LF)
14  ^N = recall next input line
16  ^P = recall previous input line
18  ^R = back char (reverse)
21  ^U = erase to start of line
23  ^W = delete to end of word "gobble word forward")
24  ^X = erase entire line
27  ESC lead character for arrow and function keys
127 DEL = destructive backspace (same as BS)
```

- T (ターミネータ・モード) では、ASCII の 0 から 31 および 127 から 159 の 65 個の制御文字を除くすべての文字をデータとして使用できます。これらの文字のほとんどは、READ 終端文字として扱われます。これには、他のすべてのプロトコルではデータとして扱われるタブ文字 (ASCII 9) も含まれます。コマンド行制御文字として扱われる文字もいくつかあります。ASCII の 3 (Ctrl-C) は、入力を破棄し、BREAK が有効になっている場合は <INTERRUPT> を発行します。ASCII の 8 (バックスペース) および 127 (削除) は、画面上で後退して 1 文字を消去してから READ を続行します。ASCII 21 (Ctrl-U) および ASCII 24 (Ctrl-X) は、後退して前にあるすべての文字を消去してから READ を続行します。ASCII 27 は、エスケープ文字です。
- IT (イメージ・モード + ターミネータ・モード) では、ASCII の 0 から 31 および 127 から 159 の 65 個の制御文字を除くすべての文字をデータとして使用できます。このモードでは、すべての制御文字が READ ターミネータ文字として扱われます。

これらのモードのいずれでも、terminator パラメータを使用して、追加のターミネータ文字を明示的に指定できます。イメージ・モードは、ビット・ストリーム・データに使用することが多いので、通常は、任意の文字をターミネータとして指定することは避けます。

3.3.8 明示的ターミネータ文字

OPEN コマンドまたは USE コマンドの terminator パラメータを使用して、特定の文字を READ コマンドまたは WRITE コマンドのターミネータとして定義できます。これらの明示的ターミネータは、指定した protocol で決まるターミネータ文字を補足するために使用できます。この terminator パラメータは、protocol による文字の指定を無視し、指定した文字を代わりのターミネータ文字として指定するためにも使用できます。文字をターミネータとして再定義する機能に対する例外は、ASCII 0 (NULL)、ASCII 3 (Ctrl-C)、および 2 つの出力制御文字 Ctrl-Q (XON) および Ctrl-S (XOFF) です。これらの文字は、元の機能が保持され、ターミネータ文字として再定義できません。

3.3.8.1 例

以下の例は、Z、Backspace、Tab を主デバイスのターミネータとして定義します。アンダースコアは、連結演算子です。

ObjectScript

```
USE 0:("":"":"Z"_$CHAR(8,9))
```

所有していないターミナルに対して OPEN コマンドを発行すると、明示的なターミネータの InterSystems IRIS 内部リストが暗黙的にクリアされます。プロトコル文字列を検出すると、InterSystems IRIS は以下を実行します。

1. 明示的なターミネータ・リストをクリアします。
2. プロトコル文字列に従い、プロトコルを設定します。
3. ターミネータ文字列が存在する場合、それを内部的な明示的ターミネータ・リストにコピーします。

   以下のテーブルは、明示的なターミネータ文字列の例です。

テーブル 3-5: ターミネータ文字列 : 例

ターミネータ文字列	定義
USE 0:(80:"C":$CHAR(27))	このエスケープ文字は、エスケープ・シーケンスを開始せず、READ を終了します。
USE 0:(80:"C":"")	空の文字列は、すべてのターミネータをクリアします。
USE 0:(80:"C")	protocol の指定時に terminator パラメータを省略すると、すべてのターミネータをクリアします。
USE 0:(80) または U 0:80	protocol と terminator の両方を省略すると、ターミネータは変更されません。

3.3.9 Read 操作でのプロトコルとターミネータの概要

以下の文字は、通常モード (非イメージ) の READ を終了します。

- Enter
- ASCII NUL 以外のターミネータ文字列の文字と Ctrl-C、Ctrl-O、Ctrl-Q、Ctrl-S
- T プロトコルが有効な場合の、出力制御文字以外の制御文字。制御文字は、10 進数コード 0 から 31、および 127 から 159 の出力されない文字です。
- あらゆるエスケープ・シーケンス
- 固定長 READ x#y の y 番目の文字。

以下の文字は、イメージ・モードの READ を終了します。

- ASCII NUL 以外のターミネータ文字列で指定された文字
- T プロトコルが有効な場合の制御文字
- 固定長 READ x#y の y 番目の文字。

3.4 READ コマンド

キーボードから入力した 0 から 32KB までを、指定した変数に読み取ります。timeout はオプションです。コマンドは、シャープ記号 # やコロン : で終了できません。

3.4.1 構文

```
READ variable:timeout                                          ; Variable-length
 read
READ variable#length:timeout  ; Fixed-length read
READ *variable:timeout                                        ; Single-character
 read
```

詳細は、"ObjectScript ランゲージ・リファレンス" の "READ" を参照してください。

3.4.2 例

以下のテーブルは、Read コマンド引数の使用方法です。

テーブル 3-6: READ コマンド引数 : 例

例	結果
READ ^GLO	ターミネータを検出するまで現在のデバイスから文字列を読み取り、結果の文字列をグローバル ^GLO に置きます。
READ X:60	ターミネータを検出するまで現在の文字列から読み取り、読み取った文字列を変数 X に置きます。入力が終了するまで最大 60 秒間待機します。キーを押しても、タイムアウト値はリセットされません。
READ *X	現在のデバイスから 1 つの文字を読み取り、10 進値をローカル変数 X に置きます。
READ X#1	現在のデバイスから 1 つの文字を読み取り、その文字列値をローカル変数 X に置きます。
READ X#45:60	現在のデバイスから最大 45 文字までを読み取り、その文字列値をローカル変数 X に置きます。入力が終了するまで最大 60 秒間待機します。

3.4.3 行呼び出し

行呼び出しモードでは、ターミナルからの READ 操作に対する入力として、編集可能な行の行呼出しが提供されます。これらの呼び出し可能な行には、以前の READ 入力行と以前のコマンド行の両方が含まれます。行呼出しの前提条件として、入力行のエコーが必要です。

InterSystems IRIS は、可変長ターミナル読み取り (`READ variable`) と固定長ターミナル読み取り (`READ variable#length`) の両方で、行呼び出しをサポートしています。InterSystems IRIS は、単一文字ターミナル読み取り (`READ *varaiable`) では行呼び出しをサポートしていません。行呼び出しでは、オプションの timeout 引数をサポートしています。

固定長ターミナル読み取りの場合、呼び出された行は、READ で指定した文字数よりも 1 文字少ない文字数まで切り捨てられます。READ で指定した最終文字の位置は、行終了文字の入力、編集文字の指定、またはデータ文字 1 文字の追加のために予約されています。

行呼び出しがアクティブである場合、**上矢印**キーと**下矢印**キーを使用して以前のターミナル入力行を呼び出すことにより、READ に入力を提供できます。その後、**左矢印**キー、**右矢印**キー、Home キーおよび End キーを使用して、呼び出された行を編集するためにカーソルの位置を変更できます。文字を削除するには Backspace キー、行全体を削除するには Ctrl-X キー、行の中でカーソルの左側にある部分をすべて削除するには Ctrl-U キーをそれぞれ使用します。

行呼び出しがアクティブでない場合、4 つの **矢印** キー、Home キー、および End キーはすべて、行終了文字を発行します。1 つの入力文字を削除するには Backspace キー、入力行全体を削除するには Ctrl-X キー (または Ctrl-U キー) を使用します。

OPEN コマンドまたは USE コマンドを使用すると、R プロトコルを指定することで行呼び出しをアクティブにでき、N、I、S、または T のいずれかのプロトコルを指定することで行呼び出しを非アクティブにできます。

3.4.4 ターミナル入出力に影響する特殊なプロトコル文字

各オペレーティング・システムは、特定のプロトコル文字 (UNIX® の場合) やキーの組み合わせ (例えば、Windows プラットフォームでの CTR-ALT-DEL) を無効にし、これらの文字が InterSystems IRIS に影響を与えないようにします。Windows のコンソールは、これらのオペレーティング・システム機能を阻害しません。

他の特殊文字は、ルーチンの実行方法を変更できますが、READ コマンドの変数には使用されません。イメージ・モードでターミナルを操作すると、これらの機能は無効となり、InterSystems IRIS はこれらの文字を他の文字と同様に扱います。

READ は、入出力制御文字に影響されます。READ は、終端文字以外の制御文字すべてを読み取りますが、エコーしません。

出力制御文字は、ルーチン・フローとルーチン出力の両方に影響します。以下はその説明です。

テーブル 3-7: 出力制御文字

出力制御文字	10 進数値	定義
Ctrl-C	3	ブレークが有効の場合、Ctrl-C はルーチンの実行に割り込みます。ルーチンは、<INTERRUPT> エラーが発生したかのように動作します。ブレークが無効の場合、Ctrl-C により InterSystems IRIS は現在の READ までの入力をすべて破棄します。Ctrl-C を使用して、ネットワーク処理を必要とするグローバル・モジュール要求に割り込むことができます。Ctrl-C をトラップするには、特殊変数 $ZTRAP を設定します。詳細は、ブレークの有効化についてのセクションを参照してください。
Ctrl-S	19	Ctrl-S は、ターミナルへの出力を中断します。InterSystems IRIS で Ctrl-Q が検出されると、ターミナルへの出力が再開します。
Ctrl-Q	17	Ctrl-Q は、Ctrl-S で一時停止している出力を再開します。

入力制御文字は、入力に影響を与えます。これらの文字は、イメージ・モード (I プロトコル) ではデータとして扱われますが、通常モードでは現在の READ コマンドへの入力に影響します。以下は、これらの文字についての説明です。

テーブル 3-8: 入力制御文字

入力制御文字	10 進数値	定義
Delete	127	Delete 文字は、最後に入力された文字を削除します。Delete キーを繰り返し押すと、右から左へ文字が削除されます。しかし、現在の READ コマンドの開始位置より前の文字は削除されません。Delete は、バックスペースを使用して、CRT 画面の最後の文字を消去します。Delete は、(テレタイプなどの) 印刷ターミナルではバックスラッシュ ("¥") としてエコーします。
Ctrl-U	21	最後のキャリッジ・リターンまで、現在の READ を開始後に入力されたすべての文字、あるいは UNIX® で先行入力のバッファ・コンテンツを削除します。Ctrl-U は、CRT では削除された文字を消去します。プリンタ上では ^U をエコーし、Enter と LineFeed を送信します。先行入力されたバッファを完全にフラッシュするには、Ctrl-X を使用します。
Ctrl-H	8	システムによっては、Delete と同じ機能を実行します。
Return	13	キャリッジ・リターンは、"I" (イメージ・モード) を除くすべてのプロトコルで READ を終了します。
Escape	27	エスケープ・シーケンスを開始します。このシーケンスは READ を終了し、$ZB は、先行の Escape を含むすべてのシーケンスを取得します。InterSystems IRIS にシーケンス文字がエコー表示されません。しかし、WRITE * コマンドでエスケープ・シーケンスを指定していない場合、$X と $Y は変更されます。この章で前述している "$X と $Y とカーソル位置" を参照してください。無効なエスケープ・シーケンスは、$ZA のビット 8 を設定します。READ X の例を考えてみます。文字 "AB"、Escape、および "E" を入力すると、X には "AB" の 2 文字が含まれます。一方、$ZB には、Escape E の 2 文字が含まれます。$X は、AB で 2 つ増加しますが、E では増加しません。

入力制御文字	10 進数値	定義
LineFeed	10	InterSystems IRIS は、すべてのターミナル入出力に対し、LineFeed をターミネータと解釈します。
Tab	9	Tab は、スペースとしてエコーするデータ値で、$X を 1 増加し、READ が返した文字列に Tab 文字として格納されます。これは、“T”（ターミネータ）を除くすべてのプロトコルに当てはまります。“T” プロトコルでは、タブはターミネータ制御文字です。

3.4.4.1 UNIX® ジョブ制御の無効化

UNIX® ジョブ制御文字 Ctrl-Z を InterSystems IRIS 内で使用すると、深刻な問題が生じる場合があります。このため、ジョブ制御をサポートする UNIX® シェルを備えたプラットフォームで InterSystems IRIS を起動すると、自動的に Ctrl-Z が無効になります。InterSystems IRIS を終了し、UNIX® シェル・コマンドを実行するために $ZF(-1) を呼び出すと、Ctrl-Z が再び有効になります。

3.4.5 READ コマンドによる入力の処理

READ コマンドは、入力バッファから文字を受け取るたびにそれを処理します。以下のテーブルは、このプロセスが通常モードでどのように発生するかを示しています。以下の図は、READ コマンドがイメージ・モード・データを処理する方法を示しています。

図 3-1: 通常（非イメージ）モードでの READ コマンド処理

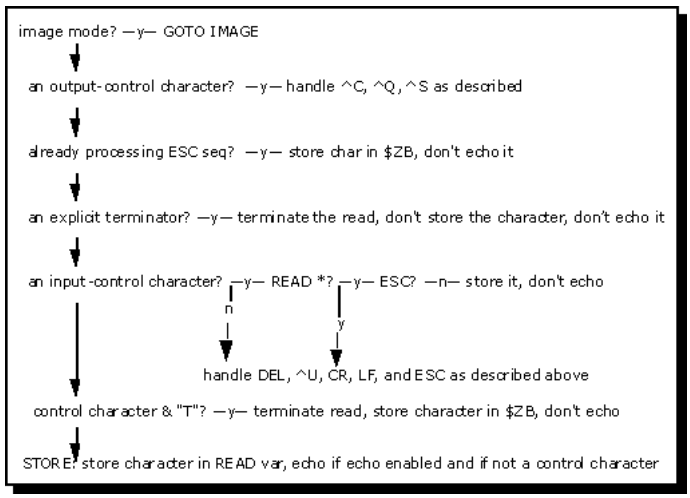
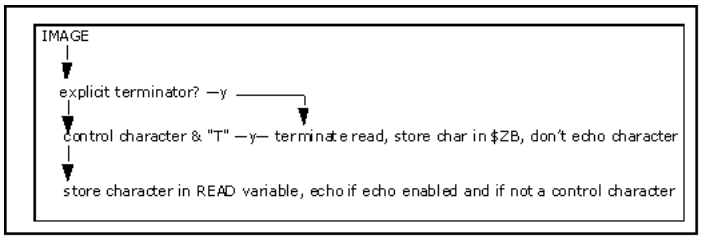


図 3-2: イメージ・モードでの READ コマンド処理



3.5 WRITE コマンド

ゼロ個以上の文字をターミナルに書き込みます。

3.5.1 構文

```
WRITE *variable
WRITE *-n
WRITE #
WRITE /mnemonic
```

各項目の内容は次のとおりです。

引数	定義
(なし)	引数なしの WRITE は、すべてのローカル変数を現在のデバイスに書き込みます。
*variable	WRITE *variable は、10 進数値が x と等しい ASCII 文字を書き込みます。変数の値は、ASCII 文字の 0 から 255 までの整数になります。Unicode 文字の場合は 254 以下です。規定上は、0 から 127 までの値は 7 ビット ASCII 文字を表します。128 から 255 までの値は拡張 ASCII 文字セットを表しますが、アプリケーション自体に関連する文字です。ハードウェアとソフトウェアが正常に設定されている場合、InterSystems IRIS は 8 ビットのデータを処理できます。例 : 8 番目のビットを使用して、国際的な文字セットを表すことができます。InterSystems IRIS ルーチンは WRITE * を使用して、デバイス依存機能に制御文字を送信します。例 : WRITE *27,*91,*50,*74 は、ターミナル画面をクリアします。WRITE * は、$X または $Y を変更しません。WRITE * の出力は、出力デバイスに特有であると想定しているためです。
*-1	WRITE *-1 は、次の READ が発行されると、入力バッファをクリアします。次の READ コマンドのために保留になっているすべての文字がクリアされます。したがって、先行入力文字がクリアされます。 入力バッファは、キーボードから入力された文字を保持します。また、ルーチンが READ コマンドを実行する前にユーザが入力した文字も保持します。これにより、画面に入力が表示される前に質問に応答できます。READ コマンドがバッファから文字を取得するとき、InterSystems IRIS は、質問と応答を一緒に表示するようにターミナルにエコー・バックします。ルーチンがエラーを検出すると、WRITE *-1 を発行し、これらの応答をキャンセルする場合があります。
*-10	WRITE *-10 は、直ちに入力バッファをクリアします。この処理は、次の READ コマンドを待ちません。したがって、WRITE *-10 の前に発行されたすべての先行入力文字をクリアします。WRITE *-10 の後に発行された先行入力文字は、次の READ コマンドが使用するために入力バッファに残ります。
#	WRITE # を CRT ターミナルに発行すると、画面をクリアし、ホーム位置 (0,0) にカーソルを移動します。ハードコピー・ターミナルには、キャリッジ・リターンと書式送りが書き込まれます。WRITE # は、$Y を 0 に設定します。

引数	定義
/mnemonic	WRITE /mnemonic を発行すると、InterSystems IRIS は、ニーモニックをアクティブなニーモニック空間で定義されたものとして解釈します。アクティブなニーモニック空間がない場合は、エラーが返されます。2 つの方法で、アクティブなニーモニック空間を指定できます。その 1 つは、ネームスペースとネットワーク構成エディタを使用して、デバイス・タイプごとに既定のニーモニック空間を指定する方法です。もう 1 つは、デバイスに対する OPEN コマンドまたは USE コマンドでニーモニック空間を指定する方法です。詳細は、"ニーモニック空間によるデバイス制御" を参照してください。

詳細は、"ObjectScript ランゲージ・リファレンス" の "WRITE" を参照してください。

3.5.2 例

以下の例にある WRITE * では、ユーザ・ターミナルで警告音を発してプロンプトを表示し、受信済みで使用されていない文字を入力バッファからクリアします。

ObjectScript

```
SET eek="No. I can't accept that reply"
WRITE *7,eek,*-10
```

以下の 2 つの例は、WRITE *-1 と WRITE *-10 との違いを示します。どちらの場合も、ユーザは、最初の READ に応答して ENTER キーを押し、その後、HANG コマンドに起因する 2 つの一時停止の間に先に入力します。

ObjectScript

```
READ "type:",x HANG 4 WRITE *-1 HANG 4 READ "type:",y
```

上記の例で、InterSystems IRIS は、2 番目の READ が発行されると、入力バッファをクリアします。したがって、いずれかの HANG 中に入力したテキストはすべてバッファからクリアされます。

ObjectScript

```
READ "type:",x HANG 4 WRITE *-10 HANG 4 READ "type:",y
```

上記の例で、InterSystems IRIS は、WRITE *-10 が発行されると直ちに入力バッファをクリアします。したがって、最初の HANG の間に入力したテキストはすべてクリアされますが、2 番目の HANG 中に入力したテキストはすべて 2 番目の READ コマンドに提供されます。

以下の例は、WRITE /mnemonic で制御ニーモニック CUP (CUrsor Position) を使用し、ターミナルの 4 行 3 列目にカーソルを移動します。この例では、事前定義のニーモニック空間 ^%X364 は USE コマンドで指定し、開いているターミナル・デバイスの名前は terminal 変数を使用して指定します。^%X364 の詳細は、"ターミナルの事前定義ニーモニック空間" を参照してください。

ObjectScript

```
USE terminal:(80:"BP"):"%X364"
SET %1=3,%2=4
WRITE /CUP(%1,%2)
```

3.6 CLOSE コマンド

OPEN コマンドで取得したデバイスの所有権を解放します。

3.6.1 構文

```
CLOSE device
```

詳細は、"ObjectScript ランゲージ・リファレンス" の "CLOSE" を参照してください。

3.7 ターミナルの事前定義ニーモニック空間

InterSystems IRIS では、ターミナルでの使用のために、以下の 2 つの事前定義ニーモニック空間を提供します。

- ANSI X3.64 ターミナル用 ^%X364
- DTM PC コンソール用 ^%XDTM

これらのニーモニック空間の 1 つをアクティブにすると、関連する制御ニーモニックも WRITE /mnemonic コマンドで使用できます。また、独自のニーモニック空間も生成できます。ニーモニック空間の詳細は、"ターミナル入出力" の章の "ニーモニック空間によるデバイス制御" を参照してください。

以下のセクションは、ニーモニック空間用制御ニーモニックを説明しています。

3.7.1 X3.64 のニーモニック空間

InterSystems IRIS は、ANSI X3.64 を定義するために、組み込みのニーモニック空間を提供します。このニーモニック空間は、管理者のネームスペース内にある InterSystems IRIS ルーチン %X364 です。以下のいずれかの方法で、ルーチン %X364 を使用します。

- InterSystems IRIS システム管理者が [IO 設定] 構成設定で、既定のニーモニック空間として %X364 を入力します。管理ポータルから、[システム管理]、[構成]、[デバイス設定]、[IO設定] の順に選択します。
- このニーモニック空間を指定する OPEN コマンドを発行します。

ObjectScript

```
OPEN "terminal"::"^%X364"
```

以下のテーブルは、ニーモニックのリストです。

テーブル 3-9: %X364 ニーモニック空間の制御ニーモニック

呼び出しシーケンス	名前	影響を受けるシステム変数
APC	アプリケーション・プログラム・コマンド	
BEL	警戒音を鳴らす	
CBT(%1)	カーソル後退タブ	$X
CCH	取り消し文字	
CHA(%1)	カーソル水平絶対位置	$X
CHT(%1)	カーソル水平タブ	$X
CNL(%1)	カーソル行送り	$X,$Y

呼び出しシーケンス	名前	影響を受けるシステム変数
CPL(%1)	カーソル前行移動	$X,$Y
CPR	カーソル位置の報告	
CTC(%1,%2,%3,%4, %5,%6,%7,%8,%9)	カーソル・タブ制御	
CUB(%1)	カーソル後退	$X
CUD(%1)	カーソル下方移動	$Y
CUF(%1)	カーソル前方移動	$X
CUP(%1,%2)	カーソル位置	$X, $Y
CUU(%1)	カーソル上方移動	$Y
CVT(%1)	カーソル垂直タブ	$Y
DA	デバイス属性	
DAQ(%1,%2,%3,%4, %5,%6,%7,%8,%9)	適用範囲の定義	
DCH(%1)	文字削除	
DCS	デバイス制御文字列	
DL(%1)	行削除	
DMI	手動入力の無効	
DSR(%1)	デバイス・ステータスの報告	
EA(%1)	範囲内消去	
ECH(%1)	文字消去	
ED(%1)	表示内消去	
EF(%1)	フィールド内消去	
EL(%1)	行内消去	
EMI	手動入力の有効	
EPA	保護範囲の終了	
ESA	選択範囲の終了	
FNT	フォント選択	
GSM	図形サイズの変更	
GSS	図形サイズの選択	
HPA(%1)	水平位置属性	$X
HPR(%1)	相対水平位置	$X
HTJ	調整付き水平タブ	$X
HTS	水平タブ設定	$X
HVP(%1,%2)	水平垂直位置	$X, $Y
ICH(%1)	文字挿入	

呼び出しシーケンス	名前	影響を受けるシステム変数
IL(%1)	行挿入	
IND	インデックス	$Y
INT	割り込み	
JFY	位置調整	
MC	媒体のコピー	
MW	メッセージ待機	
NEL	次行	$X, $Y
NP(%1)	次ページ	
OSC	オペレーティング・システム・コマンド	
PLD	一部行下方送り	$Y
PLU	一部行上方送り	$Y
PM	秘匿メッセージ	
PP(%1)	先行ページ	
PU1	非公開使用 1	
PU2	非公開使用 2	
QUAD	QUAD	
REP(%1)	REPEAT	$X, $Y
RI	リバース索引	$Y
RIS	初期状態へのリセット	$X=0 $Y=0
RM(%1,%2,%3,%4,% 5,%6,%7,%8,%9)	リセット・モード	
SEM	範囲編集モードの選択	
SGR(%1,%2,%3,%4, %5,%6,%7,%8,%9)	図形表示の選択	
SL	左スクロール	
SM(%1,%2,%3,%4,% 5,%6,%7,%8,%9)	設定モード	
SPA	保護範囲の開始	
SPI	スペースの追加	
SR	右スクロール	
SS2	単一シフト 2	
SS3	単一シフト 3	
SSA	選択範囲の開始	
ST	ターミネータ文字列	
STS	転送状態の設定	
SU	上方スクロール	

呼び出しシーケンス	名前	影響を受けるシステム変数
TBC	タブ・クリア	
TSS	半角スペース指定	
VPA(%1)	垂直位置属性	$Y
VPR(%1)	相対垂直位置	$Y
VTS	垂直タブ設定	

3.7.2 DTM PC コンソール用ニーモニック空間

InterSystems IRIS は、DTM アプリケーション開発用に使用するニーモニックに適合した InterSystems IRIS ルーチン %XDTM を提供します。このニーモニック空間は有効ですが、ターミナルの既定のニーモニック空間としては設定されません。DTM 用に生成したアプリケーションを InterSystems IRIS に移植する場合、以下のいずれかを実行できます。

- 管理ポータルで、^%XDTM をターミナルの既定のニーモニック空間 ([MnemonicTerminal]) として構成します。
- OPEN コマンドまたは USE コマンドを使用して、^%XDTM ニーモニック空間を参照します。

3.7.3 DTM の例

3.7.3.1 UNIX®

ObjectScript

```
OPEN "/dev/tty04/"::"^%XDTM"
```

3.7.3.2 Windows

ObjectScript

```
OPEN "c:\sys\user"::"^%XDTM"
```

その後、InterSystems IRIS は、以下のテーブルのように、DTM 制御ニーモニックを `WRITE /mnemonic` コマンドに適切に変換できます。

テーブル 3-10: DTM PC コンソール用制御ニーモニック

ニーモニック	説明
AA	通常モード
AB	太字モード
AC	下線モード
AD	太字、下線モード
AE	反転映像
AF	反転映像/太字モード
AG	反転映像/下線モード
AH	反転映像/太字、下線モード

ニーモニック	説明
AI	点滅モード
AJ	太字、点滅モード
AK	下線、点滅モード
AL	太字、下線、点滅モード
AM	反転映像/太字、点滅モード
AN	反転映像/太字、点滅モード
AO	反転映像/下線、点滅モード
AP	反転映像/太字、下線、点滅モード
AZ	モード Z
B(%1,%2)	映像属性の設定:%1 は文字属性、%2 はフレーム・クリア属性
BOX	ウィンドウ相対ユーティリティ・ボックスの描画
C(%1,%2)	カーソルを列 %1、行 %2 に配置
CLR	現在のフレームをクリア
COLOR(%1,%2)	IBM PC 標準色に設定 : 前景 %1、背景 %2
DC(%1)	%1 文字の削除
EC(%1)	%1 文字の消去
EF	フレームの最後まで消去
EL	行の最後まで消去
F(%1,%2,%3, %4,%5)	左上隅の行幅 %4、行高 %5 の四角い領域を $CHAR(%1) で埋める
GETCUR	ターミナル・カーソル位置に戻る
HF	画面の半輝度化の解除
HIDECURSOR	マウス・カーソルを隠す
HN	画面を半輝度化
IC(%1)	%1 文字の挿入
LF	リテラル・モードを無効
LN	リテラル・モードを有効にし、PC の画面に制御文字を画像で表示
MARK(%1)	画面に印付け
NORM	標準表示属性を有効
PAD(%1)	パディング用として %1 NULLS を書き込み
PF	一時停止の解除
PN	一時停止
RF	反転映像画面オフ
RN	反転映像画面
SD(%1,%2,%3)	現在のフレームを %3 行ずつスクロール・ダウン

ニーモニック	説明
SHOWCURSOR	マウス・カーソルの表示
SU(%1,%2,%3)	現在のフレームを %1 から %2 に %3 行ずつスクロール・アップ (ただし、行 %2 は含まない)
VF	カーソル表示オフ
VN	カーソル表示オン
WBOX	画像相対ユーティリティ・ボックスの描画
WCLOSE	ユーティリティ・ウィンドウを閉じる
WINDOW	スクロール・ウィンドウの設定
WOPEN	ユーティリティ・ウィンドウを開く
Y(%1)	バイナリ・フレーム属性の設定

3.8 PRINT コマンドと ZPRINT コマンド

現在ロードしている InterSystems IRIS ルーチンの 1 行以上を、現在のデバイスに書き込みます。

ZPRINT は、PRINT と同じ機能と引数を持ちます。

3.8.1 構文

```
PRINT
ZPRINT
PRINT x
ZPRINT x
PRINT x:y
ZPRINT x:y
```

各項目の内容は次のとおりです。

引数	定義
(なし)	引数なしの PRINT あるいは ZPRINT コマンドは、すべてのルーチンを出力します。
x,y	変数 x と y は、出力する行範囲を示します。TAG+OFFSET 形式の行参照、あるいは +7 の形式の行数のいずれかになります。ルーチン内以外の行を参照すると、ルーチンの最終行に続く空の行があるものと見なします。x は、出力する最初の行あるいは唯一の行です。y は、出力する最後の行です。

詳細は、"ObjectScript ランゲージ・リファレンス" の "PRINT" を参照してください。

3.8.1.1 例

以下の例では、現在のルーチンの最初の行、INIT で開始する 4 行目、FINI から最後までの行すべてを出力します。

ObjectScript

```
INIT
  SET a=1
  SET b=2
  SET c=3
  SET d=4
FINI
  SET x=24
  SET y=25
  SET z=26
  PRINT +1,INIT:INIT+3,FINI:+9999
```

3.9 ターミナルのプログラミング

3.9.1 InterSystems IRIS を使用したフォーマット済み CRT 画面のプログラム

ターミナル入出力の機能の中には、次のようにフォーマット済み画面のプログラムに役立つものもあります。

- `WRITE *` を使用して、制御文字を簡単に送信する
- READ を使用して、エスケープ・シーケンスの応答を受信する
- `SET $X = expression` と `SET $Y = expression` を使用して、現在のカーソル位置を更新する

固定長の READ とプログラム固有の終端文字により、個々のフィールドの読み取りが簡単になります。隠しプロトコルを使用して、パスワードを非表示にできます。

`WRITE *` は、$X も $Y も変更しません。変更するには、`WRITE $C(X)` を使用するか、それらを明示的に設定します。

3.9.1.1 例

この例は、VT100 カーソルを行 10 列 20 に設定します。

```
%SYS>SET DY=10,DX=20
%SYS>WRITE *27,*91,DY+1,*59,DX+1,*72 SET $Y=DY,$X=DX
```

3.9.1.2 CURRENT^%IS を使用した変数設定

ユーティリティ・ルーチン CURRENT^%IS は、現在のデバイスで作業するのに役立つローカル変数を設定します。このルーチンを呼び出すには、以下を入力します。

```
%SYS>DO CURRENT^%IS
```

このコマンドは、以下の変数を設定します。

テーブル 3-11: CURRENT^%IS で使用できる機能

コード	定義
W @FF	画面をクリアし、左上隅 (列 0、行 0) にカーソルを移動して $X=0、$Y=0 とします。
S DX=42,DY=10 X XY	$X=42、$Y=10 をそのままにした状態で、カーソルを直接列 42、行 10 に移動します。

3.9.2 エスケープ・シーケンスのプログラミング

エスケープ・シーケンスの ANSI 規格により、スマート・ターミナル・プログラムを実用できます。**エスケープ**文字とそれに続くすべての文字は画面に表示されませんが、$X と $Y が更新されます。WRITE * 文を使用してターミナルにエスケープ・シーケンスを送信し、$X と $Y を直接設定して最新の状態にします。

ANSI 規格は、エスケープ・シーケンスの標準構文を規定しています。特定のエスケープ・シーケンスの機能は、使用するターミナルの種類によって異なります。

各 READ の後に続く $ZB に、入力されたエスケープ・シーケンスが記述されます。InterSystems IRIS は、ANSI 規格のエスケープ・シーケンスと、ANSI 形式を使用する他のエスケープ・シーケンスを $ZB に置きます。InterSystems IRIS は、エスケープ・シーケンスの以下の 2 つの形式を認識します。

3.9.2.1 通常形式

- ESC
- オプションとして、文字“O”、10 進数値 79
- 10 進数値 32 ～ 47 のゼロ個以上の文字
- 10 進数値 48 ～ 126 の 1 文字

3.9.2.2 制御形式

- ESC 文字、10 進数値 27
- “[” 文字、10 進数値 91
- 10 進数値 48 ～ 63 のゼロ個以上の文字
- 10 進数値 32 ～ 47 のゼロ個以上の文字
- 10 進数値 64 ～ 126 の 1 文字

さらに、シーケンスは 16 文字以下とする必要があります。この形式や規則に従わないエスケープ・シーケンスを使用すると $ZA のビット 8 が設定され、その値は 256 になります。

3.9.3 例

ここでは、Help キーを押すと、2 文字のシーケンス Escape-? が送信されるターミナルをプログラムしているとします(? は、10 進数値 63 です)。

```
%SYS>SET HELP=$C(27,63)
ASK READ !,"Enter ID: ",X I $ZB=HELP Do GIVEHELP GoTo ASK
```

このルーチンは、以下のように非標準のエスケープ・シーケンスを検出できます。

1. ESC をターミネータにします。
2. ESC が $ZB に表示された場合、以下を実行します。
   a. 隠しプロトコルのエコーを無効にし、$X/$Y が変更されないようにします。
   b. READ * で残りのシーケンスを読み取ります。
   c. 隠しプロトコルを無効にし、再度エコーを使用できるようにします。

以下の図では、ユーザに ID を入力するよう要求しています。ユーザが、Esc-? を押すと、ヘルプ画面が表示されます。サブルーチン ESCSEQ は、非標準エスケープ・シーケンスの最後にアスタリスク“*”が使用されていることを前提としています。

ObjectScript

```
DEMOS
  SET HELP=$C(27,63) ;Get Help with <ESC>?
    SET ESC=$C(27) USE 0:(""":""":ESC) ; Make <ESC> a READ terminator
                                       ; character
ASK READ !,"Enter ID: ",X I $ZB=ESC Do ESCSEQ G:SEQ=HELP ASK
    . ;Input ID. Handle Help request.
    .
    Quit
HELPSCR  ;Process Help request
    .
    Quit
ESCSEQ  USE 0:(""":"S") SET SEQ=ESC ;Set terminal to no echo,init SEQ
    FOR I=1:1 {
      READ *Y
      SET SEQ=SEQ_$C(Y)
      QUIT:Y=42 }
    ; Read in Escape sequence,
    ; end at "*"
    USE 0:(""":""":ESC) Quit ;Redefine terminator
```

3.9.4 全二重および半二重ターミナルとエコーのサポート

InterSystems IRIS は、全二重ターミナルの使用を想定しています。つまり、キーボードの動作は、プリンタや画面に依存していない必要があります。

全二重とは、双方向で同時に独立して送信が行われるということです。半二重とは、一度に片方向のみの送信を意味します。二重ターミナルは、エコーと関係ありません。しかし、リモート・エコーを全二重、ローカル・エコーを半二重と表示しているターミナルもあります。これは、入力した文字がターミナル自身で表示され、InterSystems IRIS はその文字をエコーしないことを意味します。

ターミナルのローカル・エコーあるいは全二重を無効に設定し、InterSystems IRIS がエコーできるようにします。エコーが発生するのは、コンピュータが文字を受け取ったときではなく、READ コマンドが入力バッファから文字を取得したときです。したがって、プロンプトとダイアログの応答は、ユーザが事前に入力したかどうかに関係なく、画面の決められた位置に表示されます。

公開ネットワークの中には、ターミナルに独自のエコーを供給するものもあります。

Windows システムでは、コンソールはローカル・エコーの設定変更を許可していません。ターミナル・エミュレータ（例えば VT220）経由のターミナルでは、ターミナル・エミュレータ・ドキュメントを参照して、ローカル・エコーを無効にしてください。

UNIX® では、stty コマンドを使用して二重エコーを防ぎ、$X と $Y がカーソル位置と一致しているようにします。

3.9.5 InterSystems IRIS がサポートするコンピュータ間リンクと特殊デバイス

InterSystems IRIS は、柔軟なプロトコルと大容量バッファを提供しており、ルーチンが特殊なデバイスとそのプロトコルを処理できるようにしています。例えば、InterSystems IRIS は、ターミナル・リンクで接続された 2 台のコンピュータ間の全二重通信を簡単にサポートしています。2 つの InterSystems IRIS システムで必要な操作は、物理的な接続、適切なプロトコルの指定、および速度、パリティ、文字列長の同一設定のみです。外部コンバータを使用すると、InterSystems IRIS は、同期 EBCDIC ターミナルのように IBM ポートと通信します。

コンピュータ間のリンクを設定する場合、以下の点に注意してください。

- OPEN コマンドまたは USE コマンドで S プロトコルを指定するか、オペレーティング・システムのターミナル・パラメータを使用して、両方のエコーを無効にします。

- 通信プロトコルが XON/XOFF フロー制御 (Ctrl-Q と Ctrl-S) をサポートしていない場合、未応答の送信はオペレーティング・システムの入力バッファ範囲内に制限してください。
- イメージ・モードの場合、InterSystems IRIS は XON/XOFF をサポートしません。非イメージ (通常) モードでは、オペレーティング・システムの入力バッファが一杯の場合、コンピュータが XOFF を発行するかどうかは、オペレーティング・システムのターミナル・パラメータで決まります。XOFF と XON がサポートされていない場合、その必要がないようバッファに余裕を持たせてください。
- 読み取り操作の後で $ZA をテストして、パリティやデータのオーバーラン状態などの送信エラーを検出します。

4

ローカル・プロセス間通信

この章では、InterSystems IRIS® Data Platform のローカル・プロセス間の通信および InterSystems IRIS 外部の他のプロセスとの通信を設定する方法について説明します。

TCP/IP によるリモート・クライアント/サーバ通信の詳細は、このドキュメントの "TCP クライアント/サーバ通信" の章を参照してください。

4.1 パイプを使用したプロセス通信

InterSystems IRIS プロセスと外部の UNIX® プロセスや Windows プロセスとの間で、パイプを通じてそのオペレーティング・システム・レベルでの通信と同様に通信できます。パイプへの出力の送信と、パイプからの入力の受信ができます。パイプは単方向なので、同じプログラムで読み書きを同時に実行することはできません。

別のプログラムに出力用のパイプを開くと、シーケンシャル・ファイルのように、そのプログラムに書き込むことができます。その後プログラムは、書き込んだ内容を入力ストリームとして使用します。この機能は、InterSystems IRIS プロセスと外部プロセスでリソースを共有する場合に、特に役に立ちます。

例えば、InterSystems IRIS とワード・プロセッシング・プログラムを同時に実行していると、この 2 つのアプリケーション間で適切にプリンタを共有することが必要になる場合があります。InterSystems IRIS では、プロセスで情報をデバイスに送信できるフルアクセス権限を持っていることを前提としています。しかし、大半の UNIX® アプリケーションは、UNIX® の標準ユーティリティ lpsched を使用して、プリンタやスプーリング・ファイルへのアクセスを管理します。

このような UNIX® アプリケーションで出力すると、プリンタ・ポートには直接書き込まれず、lp または lpr というユーティリティが呼び出されます。lp (あるいは lpr) ユーティリティは lpsched を呼び出し、lpsched は、lp (あるいは lpr) を呼び出したジョブのためのプリンタへのアクセスをスケジュールします。lp を使用する場合、印刷開始を待機する必要はありません。lp への印刷ジョブの書き込みが終了したら、すぐにファイルを閉じてもかまいません。印刷の順番の待機中、lp がディスクへのジョブのスプーリングを処理します。

InterSystems IRIS は、OPEN コマンドの拡張機能により、この協調的な環境を実現できます。このコマンドは直接発行できるほか、このコマンドを使用する ObjectScript ユーティリティから発行することもできます。

4.1.1 InterSystems IRIS ユーティリティへのパイプの使用

UNIX® プロセスや Windows プロセスへのパイプと同様に、InterSystems IRIS ユーティリティへのパイプを開くことができます。入出力ユーティリティでパイプを使用する前に、システム管理者は、InterSystems IRIS システムでパイプ・デバイスを定義しておく必要があります。

システム管理者がパイプ・デバイスを定義した後、(%RD のような) ユーティリティの起動時、"Device:" プロンプトに管理者が定義したニーモニックを入力します。出力は、自動的にデバイスに送信されます。

4.1.2 パイプとコマンド・パイプ

InterSystems IRIS は、標準パイプとコマンド・パイプ (CPIPE) の両方をサポートしています。標準パイプは、比較的短いコマンド文字列に使用します。この場合のコマンド文字列は、コマンド名およびその引数が 256 文字未満であることが必要です。コマンド・パイプは、コマンド文字列の長さが 256 文字以上の場合に使用します。どちらの場合も、パイプは UNIX® システムと Windows システムでのみ使用できます。

4.1.2.1 標準パイプの OPEN

標準パイプの OPEN コマンド構文は以下のとおりです。

```
OPEN program:(parameters):timeout
```

program は最初の引数 (デバイス引数) なので、256 文字という OPEN コマンド・デバイス名の制限に従う必要があります。

既に開いている標準パイプに対して OPEN コマンドが発行された場合、2 つ目の OPEN は無視されます。エラーは発行されません。

4.1.2.2 コマンド・パイプの OPEN

コマンド・パイプの OPEN コマンド構文は以下のとおりです。

```
OPEN cpipename:program:timeout OPEN
cpipename:(program:parameters:::closetimeout):timeout
```

同時に開くコマンド・パイプが 1 つのみである場合、cpipename 引数の値は "|CPIPE|" とすることができます。同時に複数のパイプを開くには、"|CPIPE|xxxxxx" を指定します。xxxxxx は、ユーザ指定の一意な識別子です。この cpipename 引数は、このコマンド以降に実行する USE コマンドおよび CLOSE コマンドで使用するために指定する引数です。

program は 2 番目の引数なので、256 文字に制限されません。program の最大文字数は、プラットフォームに依存します。

既に開いているコマンド・パイプに対して OPEN コマンドが発行された場合、2 つ目の OPEN は無視されます。エラーは発行されません。

4.1.3 プロセス間通信の OPEN コマンド

OPEN コマンドを使用すると、プログラムで InterSystems IRIS 外部のプロセスと通信できます。

4.1.3.1 OPEN の引数

引数	説明
cpipename	コマンド・パイプのみで使用し、"|CPIPE|" または "|CPIPE|xxxxxx" を指定します。xxxxxx は、ユーザ指定の一意な識別子です。

引数	説明

program	コマンド・パイプは、コマンド・シェルを使用して、またはコマンド・シェルを使用せずに（直接）program を実行できます。ほとんどの場合、コマンド・シェルなしで実行することをお勧めします。標準パイプは、コマンド・シェルを使用して program を実行します。
	
	コマンド・パイプのみ – コマンド・シェルなしで実行するには、/COMMAND=program を指定します。program に引数がある場合は、/ARGS キーワードを使用して引数を指定する必要があります。/COMMAND または /ARGS キーワードを指定した場合、program はコマンド・シェルなしで実行されます：(/COMMAND=program)、(/COMMAND=program:/ARGS=arg1)、および (program:/ARGS=arg1) は、すべて有効な構文です。/ARGS は、1 つの引数、引数のコンマ区切りリスト、または配列を取ります。例えば、(/COMMAND=program:/ARGS=arg1,arg2) のようになります。配列を使用して、可変個数の引数を指定できます。
	
	ObjectScript
	
	`SET array(1)=arg1, array(2)=arg2, array=2`
	`OPEN device:(/COMMAND=cmd:/ARGS=array...)`
	
	コマンド・シェルを使用して実行するには、program を指定し、/COMMAND と /ARGS キーワードは両方とも省略します。
	
	program 文字列には、システムにインストールされているプログラムの完全パス名が含まれます。ホスト・システムで実行するコマンドの名前と引数（ある場合）で構成します。標準パイプでは、256 文字未満に制限されます。コマンド・パイプでは、最大文字数はプラットフォームに依存しますが、ほとんどは 256 文字以上が可能です。

引数	説明
parameters	Read。標準パイプでは、“Q”または“QR”を指定して、他のプロセスからの入力を受け取るキューまたはパイプを開きます。コマンド・パイプの場合、コマンド・パイプは間違いなくパイプであるため、“Q”文字コードは必要ありません。“R”を指定します。 Write。標準パイプでは、“QW”を指定して、他のプロセスに入力を送信するキューを開きます。コマンド・パイプの場合、コマンド・パイプは間違いなくパイプであるため、“Q”文字コードは必要ありません。“W”を指定します。 Read および Write。Read パイプまたは Write パイプのどちらにもなれる標準パイプの場合は、“QRW”を指定して、別のプロセスから入力を受け入れるキューまたはパイプ、あるいは別のプロセスに入力を送信するキューまたはパイプを開きます。コマンド・パイプの場合、パイプであることは間違いないので“Q”文字コードは必要ありません。“RW”を指定します。 /キーワード・パラメータを使用し、これらのパラメータおよび他のパラメータをコロンで区切って指定できます。例えば、OPEN "|CPIPE|":(cmd:/READ:/IOTABLE="UTF8") のようになります。以下のオプションのキーワード・パラメータは、パイプで一般的に使用します。 “K/name/”（あるいは“Knum”）：変換がシステム全体で有効になっている場合、入出力変換を有効にします。テーブル名を指定することで、変換の基本となっている既存の定義済みテーブルを特定します。プロトコルのオンとオフを切り替える“+”オプションと“-”オプションは、“K”プロトコルでは使用できません。 “Y/name/”（あるいは“Ynum”）：$X/$Y というアクション・テーブルを使用するようシステムに指示します。テーブル名を指定することで、変換の基本となっている既存の定義済み $X/$Y アクション・テーブルを特定します。$X/$Y アクションは常に使用できます。Y が指定されず、システム既定の $X/$Y が定義されていない場合、組み込みの $X/$Y アクション・テーブルを使用します。プロトコルのオンとオフを切り替える“+”オプションと“-”オプションは、Y プロトコルでは使用できません。 上記のパラメータでは、“S”（ストリーム）、“F”（固定長）、または“U”（未定義長）の各モード・パラメータを指定できます。“V”（可変長）モード・パラメータを指定することはできません。 文字コードとキーワード・パラメータの一覧は、このドキュメントの“シーケンシャル・ファイルの入出力”の章の″OPEN モード・パラメータ″を参照してください。
closetimeout	オプション – UNIX® のみ：パイプしたコマンドのデバイスを閉じるときに、CLOSE コマンドがコマンド処理の終了を待機する秒数を指定できます。既定値は 30 秒です。この closetimeout は、プロセス間通信の CLOSE コマンドの引数に “I”（即時）指定することで無効にできます。
timeout	オプション：InterSystems IRIS が OPEN の正常終了を待機する最大秒数で、正の整数です。InterSystems IRIS がタイムアウトになる前にプロセス間通信を開くことができた場合、$TEST を 1 に設定します。開くことができない場合、$TEST を 0 に設定します。タイムアウトを省略するか 0 を指定すると、OPEN は、直ちに制御をプロセスに戻します。

4.1.3.2 OPEN コマンド・パイプの例

有効なコマンド・パイプ OPEN 文は以下のとおりです。それぞれの例では、10 秒のタイムアウトを指定しています。

ObjectScript

```
OPEN "|CPIPE|1":"/nethome/myprog":10                                    // using shell, no args
OPEN "|CPIPE|1":("/nethome/myprog":/WRITE):10                          // using shell, no args, WRITE

OPEN "|CPIPE|2":/COMMAND="/nethome/myprog":10                           // no shell, no args
OPEN "|CPIPE|3":("":/COMMAND="/nethome/myprog"):10                      // no shell, no args
OPEN "|CPIPE|4":(/COMMAND="/nethome/myprog":/ARGS=arg1):10              // no shell, 1 arg
OPEN "|CPIPE|5":("/nethome/myprog":/ARGS=arg1):10                       // no shell, 1 arg
OPEN "|CPIPE|6":("/nethome/myprog":/ARGS=arg1:/WRITE):10                // no shell, 1 arg, WRITE
OPEN "|CPIPE|7":(/COMMAND="/nethome/myprog":/ARGS=arg1,arg2):10         // no shell, 2 args
OPEN "|CPIPE|8":(/COMMAND="/nethome/myprog":/ARGS=args...:/WRITE):10  // no shell, args array, WRITE
```

Windows システムでは、引数に空白または二重引用符 (") 文字を含めることができます。この場合、次のように、引数を引用符で囲むことができ、リテラルの二重引用符文字は、二重にすることによってエスケープできます。

ObjectScript

```
OPEN "|CPIPE|9":("/nethome/myprog":/ARGS="string with blanks"):10
OPEN "|CPIPE|10":("/nethome/myprog":/ARGS="string with literal "" character"):10
```

4.1.3.3 OPEN のエラー

IPC ではないデバイスに“QW”パラメータを指定して OPEN コマンドを発行すると、このデバイスに書き込もうとしたときに <WRITE> エラーが発生します。

以下の UNIX® の例では、lp プログラムへの出力パイプを開きます。このプログラムのパス名は /usr/bin/lp です。その後、パイプを通じて、グローバル ^TEXT からプリンタに出力を送信します。

ObjectScript

```
print ; Send the first layer of global ^TEXT to the printer.
      SET IO="/usr/bin/lp"
      OPEN IO:"QW"   ; Open the pipe to lp
      USE IO WRITE "The first layer of ^TEXT",!   ; Print the title
      ;  . . .
      ;  Print each line, using $ORDER on the global ^TEXT
      USE IO WRITE !,"The End.",#
      CLOSE IO    ; close the pipe, spooling the file to lpsched
      QUIT
```

この例を、OPEN コマンドが lp プログラムに引数を渡すように変更できます。例えば、lp から laserjet というプリンタ・デバイスに出力を送信するように指定する場合、SET コマンドを以下のように記述します。

ObjectScript

```
   SET IO="/usr/bin/lp -dlaserjet"
```

以下の例は、外部プログラムからの読み取り方法を示しています。このプロセスは UNIX® プログラム who への入力パイプを開くので、who では現在 UNIX® にログインしているすべてのユーザの ID を読み取ることができます。

ObjectScript

```
getids ; read the login IDs of everybody currently on
       SET IO="/usr/bin/who"
       SET $ZTRAP="EOT"
       KILL LOGINS
       OPEN IO:"Q"
       ; note that "R" (the default) is understood
       SET users=0
       FOR I=0:0 {
          USE IO
          READ USER
          SET users=users+1
          SET LOGINS(USER)=""
          }
       QUIT
EOT    SET $ZTRAP=""
       USE 0
       WRITE !,USERS," is/are currently logged on.",!
       CLOSE IO
       QUIT
```

Windows システムでは、CPIPE OPEN program 引数が /COMMAND または /ARGS を指定する場合、コマンドの実行に CreateProcess() が使用されます。CreateProcess() が失敗すると、OPEN は <NOTOPEN> エラーで失敗します。GetLastError() の値は、$SYSTEM.Process.OSError() を使用して取得できます。

UNIX® システムでは、CPIPE OPEN program 引数が /COMMAND または /ARGS を指定する場合、コマンドを実行するための exec() を発行する新しいプロセスが作成されます。exec() が失敗すると、OPEN は <NOTOPEN> エラーで失敗します。exec() の errno は、$SYSTEM.Process.OSError() を使用して取得できます。

4.1.3.4 OPEN コマンド・キーワードと USE コマンド・キーワード

以下のテーブルは、OPEN コマンドと USE コマンドを使用して、プロセス間通信パイプを制御するキーワードの説明です。

テーブル 4-1: プロセス間通信パイプの OPEN コマンド・キーワードと USE コマンド・キーワード

キーワード	既定値	説明
/IOTABLE[=name] または /IOT[=name]	name が指定されない場合、デバイスの既定の入出力変換テーブルを使用します。	K¥name¥ パラメータ・コードに相当します。デバイスの入出力変換テーブルを構築します。
/TRANSLATE[=n] または /TRA[=n]	1	K パラメータ・コードに関連します。/TRANSLATE を指定した場合、または /TRANSLATE=n の n が 0 以外の場合、デバイスの入出力変換が有効になります。/TRANSLATE=n の n が 0 の場合はデバイスの入出力変換が無効になります。
/XYTABLE[=name] または /XYT[=name]	name が指定されていない場合、デバイスの既定の $X/$Y アクション・テーブルを使用します。	Y¥name¥ パラメータ・コードに相当します。デバイスの $X/$Y アクション・テーブルを構築します。

4.1.3.5 OPEN のみのキーワード

以下のテーブルは、OPENコマンド・キーワードのみを使用して、プロセス間通信パイプを制御するキーワードの説明です。

テーブル 4-2: プロセス間通信パイプの OPEN のみのコマンド・キーワード

キーワード	既定値	説明
/IGNOREEOF[=n] または /IGN[=n]	0	I パラメータ・コードに相当します。READ 処理を、無制限に、または指定したタイムアウトを超えるまで (EOF 条件は無視して) 再試行するように指定します。/IGNOREEOF を指定した場合、または /IGNOREEOF=n の n が 0 以外の場合、パラメータ・コードが有効になり、/IGNOREEOF=n の n が 0 の場合は無効になります。
/PARAMS=str または /PAR=str	既定なし	パラメータ・コードの位置パラメータに相当します (位置に依存しない方法でパラメータ・コード文字列を指定する方法を提供します)。
/QUEUE または /QUE	デバイスはプロセス間通信パイプとして認識されません。	“Q” パラメータ・コードに相当します。プロセス間通信パイプを開くように指定します。このコマンドを使用するには、%System_Callout リソースに対する Use 許可が必要になる点に注意してください。
/Read	/READ も /WRITEも指定されない場合、既定は Read です。	“R” パラメータ・コードに相当します。キューまたはパイプを開き、他のプロセスからデータを受け入れるように指定します。
/Write または /WRI	/READ も /WRITEも指定されない場合、既定は Read です。	“W” パラメータ・コードに相当します。キューまたはパイプを開き、他のプロセスにデータを送信するように指定します。

4.1.4 プロセス間通信の READ コマンド

4.1.4.1 構文

```
READ:pc readargument,...
```

READ コマンドは、パイプからデータを読み取ります。

readargument には、以下を指定できます。

```
formatting-mode string variable:timeout *variable:timeout variable#n:timeout
```

パイプで I formatting-mode パラメータを使用します。I パラメータは、<ENDOFFILE> エラーの後ろに続く特定のレコードで発生するデータを損失せずに、名前付きパイプに時間制限の付いた READ を発行できます。このパラメータを READ で使用する場合、READ は <ENDOFFILE> メッセージを無視します。

I formatting-mode の既定値は “off” です。このパラメータを、タイムアウトなしに READ コマンドに組み込むと、プロセスにデータが指定されるまで、プロセスを停止します。

4.1.5 CPIPE の終了コード

コマンド・パイプ (|CPIPE|) プロセスの終了コードを取得できます。この終了コードは、|CPIPE| デバイスが閉じる前に取得する必要があります。これを取得するには、%SYSTEM.Process クラスの PipeExitCode メソッドを使用します。終了コー

ドは常に整数値です。終了コードが使用可能でない場合、以下の例のように、このメソッドは NULL 文字列を返し、説明と共にステータス引数を設定します。

ObjectScript

```
SET exitcode=$SYSTEM.Process.PipeExitCode(device, .status)
IF exitcode="" {DO $SYSTEM.OBJ.DisplayError(status)}
ELSE {WRITE "CPIPE exit code is ",exitcode }
```

UNIX® システムでは、非シェル・コマンド、つまり /COMMAND または /ARGS で開いた CPIPE デバイスについてのみ終了コードが使用可能です。

4.1.6 プロセス間通信の CLOSE コマンド

“Q”(/QUEUE) パラメータ・コードと共に OPEN を使用して子プロセスを作成すると、その子プロセスが CLOSE 操作の後でデバイスに残ることがあります。キュー待機プロセス間通信パイプの残存性はプラットフォームによって異なります。UNIX® システムでは、子プロセスは常に CLOSE の後まで残ります。Windows システムでは、プロセスの残存性はプロセスの古さに依存します。起動したばかりの子プロセスは CLOSE 操作の後まで残りませんが、完全に確立された子プロセスは CLOSE の後まで残ります。

UNIX® システムの場合は、パイプしたコマンドのデバイスを閉じるときに、CLOSE コマンドが待機する時間を指定できます。このタイムアウトの既定値は 30 秒です。この既定値は、OPEN コマンドの closetimeout 位置指定引数を指定することで変更できます。CLOSE コマンドのデフォルトのタイムアウト値または指定したタイムアウト値は、オプションの “I” 位置指定引数を指定することで無効にできます。“I” 引数は、即時に閉じる (1 秒後に閉じる) ことを指定します。CLOSE の構文は、次のとおりです。

```
CLOSE cpipename:"I"
```

4.1.7 名前付きパイプを使用した Visual Basic との通信

Windows では、InterSystems IRIS の名前付きパイプを TCP デバイスと同様に使用しますが、“|TCP|nnn” ではなく “|NPIPE|nnn” というデバイス名を使用します。OPEN 引数は以下のとおりです。

```
OPEN "|NPIPE|3":(server:pipename)
```

server は Windows NT マシン名、pipename は接続されているパイプ名です。Windows 95/98 マシンは、名前付きのパイプ・サーバにはなれませんが、サーバへの接続は可能です。

ローカル・パイプ名に接続するには、サーバとして **"."** (引用符で囲まれたピリオド) を使用します。(サーバとして) パイプを生成するには、サーバ名として **""** (内容なしの引用符) を使用します。以下は、すべて有効な server 名です。

ObjectScript

```
OPEN "|NPIPE|3":(".":"localpipe")
OPEN "|NPIPE|3":("mother":"test")
OPEN "|NPIPE|3":("":"info")
```

サーバは、名前付きパイプを開き、クライアント側が同じ名前付きパイプを開く前に、すぐに書き込みを発行できます。クライアント側が名前付きパイプを開くまで書き込み操作はハングします。ユーザは、Control-C を発行してこのハングを中断できます。

パイプを開くと、それは通常のデバイスと同様に動作します。サーバ側では、以下のコードを使用して、TCP で実行されるようにクライアントから切断します。

ObjectScript

```
USE "|NPIPE|3":"DISCONNECT"
```

または、以下を実行します。

ObjectScript

```
USE "|NPIPE|3" WRITE *-2
```

4.1.7.1 OPEN コマンド・キーワード

以下のテーブルは、OPEN コマンド・キーワードのみを使用して、名前付きパイプを制御するキーワードの説明です。

テーブル 4-3: 名前付きパイプの OPEN コマンド・キーワード

キーワード	既定値	説明
/HOSTNAME=str または /HOS=str	既定は "" (内容なしの引用符) です。サーバとしてパイプを開きます。	Windows NT ワークステーション/サーバ名を指定するサーバ位置パラメータに相当します。サーバとしてパイプを開いている場合、このキーワードを指定する必要はありません。ローカル・パイプ名に接続するには、"." (引用符で囲まれたピリオド) を使用します。
/IBUFSIZE=n または /IBU=n	2048	パイプからデータを受け取り、アプリケーションに送信するまで保持する名前付きパイプの入力バッファ・サイズを指定します。
/INSTANCES=n または /INS=n	1	名前付きパイプに可能な最大インスタンス数を指定します。1 以上の値を指定すると、複数のサーバが名前付きパイプのインスタンスを開くことができます。したがって、一度に複数のクライアントに対処できます。
/OBUFSIZE=n または /OBU=n	2048	オペレーティング・システムが使用する出力バッファのサイズを指定します。オペレーティング・システムは、システムによる制約に従ってバッファを分割するので、このバッファ・サイズを推奨します。
/PIPENAME=str または /PIP=str	既定なし	パイプ名を指定する pipename 位置パラメータに相当します。

4.2 InterSystems IRIS プロセス間通信

インタジョブ・コミュニケーション (IJC) デバイスは、複数の InterSystems IRIS プロセス間で情報を送受信できる一連の特別なデバイス番号です。プロセスは、ジョブ起動型プロセスあるいは対話型プロセスのいずれかです。

IJC デバイスはペアで機能します。IJC デバイスは 256 ペアまで使用可能です。データの読み取りには、レシーバと呼ばれる偶数番号のデバイスを使用します。データの書き込みには、トランスミッタと呼ばれる奇数番号のデバイスを使用します。トランスミッタからの読み取り、またはレシーバへの書き込みを実行しようとすると、<NODEV> エラーを生じます。

他のデバイスと同様に、入出力コマンドを IJC デバイスに発行します。OPEN コマンドと USE コマンドをデバイスへ発行後、プロセスは以下のコマンドを発行できます。

- レシーバ・デバイスへの READ コマンド

- トランスミッタ・デバイスへの WRITE コマンド

デバイスを開くことができるのは、一度に 1 つのプロセスのみです。

デバイスのペアは、InterSystems IRIS デバイス・テーブルでマップされた相対順序に基づいています。この順序は、管理ポータルの構成オプションを使用して表示および編集できます。

デバイスの各ペアは、IJC メモリ・バッファに対応します。プロセスが、WRITE コマンドを奇数番号の IJC デバイスに発行すると、InterSystems IRIS は、デバイスとペアになっているバッファにデータを書き込みます。別のプロセスが、ペアになっている偶数番号のデバイスに READ コマンドを発行した場合、InterSystems IRIS は同じバッファからデータを読み取ります。

書き込まれたデータは、先入れ先出し法でメモリにバッファされます。デバイスが空の場合に READ を実行すると、他のプロセスが対応する WRITE を発行するまでプロセスは中断します。バッファが一杯の場合に WRITE を実行すると、他のプロセスがバッファを読み取るまで中断します。

バッファにメッセージを書き込んだ後、トランスミッタを終了した場合も、メッセージが読み取られるまではバッファに残されます。複数のユーザが、OPEN、USE、WRITE、および CLOSE の各コマンドを、一度に 1 つずつトランスミッタに発行できます。その後の READ コマンドにより、記述された順番ですべてのメッセージを取得します。

4.2.1 インタジョブ・コミュニケーション・デバイスへのメモリ・バッファの指定

システム管理者は、管理ポータルを使用して IJC バッファを構成できます。**[システム管理]**、**[構成]**、**[追加設定]**、**[メモリ詳細]** の順に選択します。以下の 2 つのパラメータを設定できます。

- [ijcnum] : IJC デバイスの最大数。範囲は 0 ～ 256 です。既定値は 16 です。この設定を編集した場合、変更内容を適用するには、InterSystems IRIS を再起動する必要があります。
- [ijcbuff] : 各 IJC バッファの最大サイズ (バイト単位)。範囲は 512 ～ 8192 です。既定のサイズは 512 バイトです。この設定を編集した場合、変更内容を適用するには、InterSystems IRIS を再起動する必要があります。

各 IJC デバイスは、[ijcbuff] で指定されたサイズを持つ 1 つの IJC バッファに対応しています。メッセージは、[ijcbuff] マイナス 1 の長さで書き込むことができます。

4.2.1.1 IJC バッファを無効にする

IJC デバイスを使用しない場合は、IJC デバイスの最大サイズ ([ijcnum]) を 0 に設定して、メモリの占有を避けることができます。

4.2.2 IJC デバイス番号

IJC デバイスは、InterSystems IRIS で自動的に番号付けされます。実際の識別番号は、システムで構成される IJC バッファの最大数に依存します。

以下のテーブルは、割り当てる IJC バッファ数に応じてシステムで設定可能な IJC デバイス番号の範囲を示しています。

例えば、8 つの IJC バッファを割り当てる場合、224 から 239 までのデバイス番号がシステムに定義されます (READ デバイスに偶数、WRITE デバイスには奇数が割り当てられます)。

また、94 個の IJC バッファを割り当てる場合、224 から 255、64 から 199、4 から 19、2048 から 2051 のデバイス番号の範囲が定義されます。任意の奇数/偶数番号のペアを、OPEN、USE、READ、WRITE、および CLOSE の各コマンドで使用できます。

テーブル 4-4: IJC デバイス番号

バッファの割り当て数	READ デバイス番号	WRITE デバイス番号
1	224	225
2	226	227
3	228	229
...	...	
15	252	253
16	254	255
17	64	65
18	66	67
...	...	...
83	196	197
84	198	199
85	4	5
86	6	7
87	8	9
88	10	11
89	12	13
90	14	15
91	16	17
92	18	19
93	2048	2049
94	2050	2051
95	2052	2053
...	...	...
254	2370	2371
255	2372	2373
256	2374	2375

4.2.3 IJC デバイスへの入出力コマンド

すべての標準入出力コマンド、OPEN、USE、READ、WRITE、および CLOSE を IJC デバイスで使用できます。

4.2.3.1 OPEN コマンド

OPEN コマンドは、IJC デバイスの使用を確保します。

構文

```
OPEN device::timeout
```

各項目の内容は次のとおりです。

引数	説明
device	上記のテーブルのデバイス番号を記述します。偶数番号のデバイスを OPEN で開き、READ コマンドを発行します。奇数番号のデバイスを OPEN で開き、WRITE コマンドを発行します。2 つのプロセス間で通信するには、デバイスをペアで開く必要があります。
timeout	オプション：InterSystems IRIS が OPEN の正常終了を待機する最大秒数で、正の整数です。0 を指定すると、OPEN は即座に制御をプロセスに戻します。

この例は、読み書き用のそれぞれのデバイスを開き、2 つのプロセス間で通信する方法を示しています。

```
Process A                              Process B
OPEN 227 USE 227 WRITE "MSG_1"
WRITE "MSG_2"                           OPEN 226 USE 226 READ X
CLOSE 227                               CLOSE 226
OPEN 224 USE 224 READ X                 WRITE X
CLOSE 224                               MSG_1
WRITE X                                 .
MSG_3                                   .
                                        .
                                        OPEN 225 USE 225 WRITE "MSG_3"
                                        CLOSE 225
```

Process A は、デバイス 227 を開き、そこに MSG_1 を書き込んで開始します。InterSystems IRIS は、デバイス 226 と 227 で共有しているバッファにこのメッセージを書き込みます。その後、Process A は 2 番目のメッセージを同じバッファに書き込みます。新規の Process B が、対のデバイス 226 を開き、バッファから最初のメッセージ (MSG_1) を読み取ります。

現在の Process A は、メッセージを読み取るために、別のデバイス 224 を開く必要があります。このデバイスと対になる 225 のバッファは現在空なので、Process B がデバイス 225 を開き、MSG_3 を書き込むまで、Process A は待機します。InterSystems IRIS が、デバイス 224 と 225 で共有しているバッファにこのメッセージを配置した後、デバイス 224 への READ コマンドが成功します。

5

TCP クライアント/サーバ通信

この章では、TCP/IP を使用した InterSystems IRIS® Data Platform プロセス間のリモート通信の設定方法について説明します。パイプ、またはインタジョブ・コミュニケーション (IJC) デバイスを使用したプロセス間のローカル通信については、このドキュメントの ″ローカル・プロセス間通信″ の章を参照してください。

InterSystems IRIS は、TCP および UDP の 2 つのインターネット・プロトコル (IP) をサポートします。これらのインターネット・プロトコルにより、InterSystems IRIS では、プロセスが InterSystems IRIS を実行中かどうかにかかわらず、ローカルあるいはリモート・システムでプロセスと通信できます。

- TCP：InterSystems IRIS は、転送制御プロトコル (TCP) バインディングをサポートします。サーバと 1 つのクライアント間の双方向通信を確立します。エラーのチェックと修正を伴った信頼性のあるデータのバイト・ストリーム送信、およびメッセージ応答を提供します。
- UDP：InterSystems IRIS は、ユーザ・データグラム・プロトコル (UDP) バインディングをサポートします。サーバと多数のクライアント間の双方向メッセージ転送を提供します。UDP は、接続ベースではありません。各データ・パケットの転送は独立したイベントになります。ローカル・パケット・ブロードキャストおよびリモート・マルチキャストにおいて高速で軽量なデータ転送を提供します。通常、TCP よりも信頼性が低くなっています。メッセージ応答を提供しません。詳細は、このドキュメントの ″UDP クライアント/サーバ通信″ の章を参照してください。

TCP バインディングは、基盤となるネットワーク・プロトコルの基本的な機能を InterSystems IRIS ユーザが入出力コマンドから使用できるように、InterSystems IRIS を広範囲の標準ネットワークに接続します。

TCP/IP プロトコルにより、システムは、異なるタイプのネットワーク・ハードウェアを使用している場合でも通信できます。例えば、インターネット経由の TCP では、Ethernet を使用するシステムと Token Ring を使用するシステム間でメッセージを送受信できます。TCP は、データ送信の精度を管理します。IP、つまりインターネット・プロトコルは、ネットワークあるいはインターネット上の異なるシステム間の実データ転送を実行します。

TCP バインディングを使用すると、クライアント・サーバ・システムのクライアント部とサーバ部の両方を生成できます。分散データベース・システムのクライアント・サーバでは、1 つあるいは複数のクライアント・システムのユーザは、別のシステム、つまりサーバ上のデータベースに格納された情報を処理できます。

5.1 TCP 接続の概要

システム間のクライアント・サーバ・リレーションシップを生成するには、以下の規則に従う必要があります。

- システムは、TCP/IPプロトコル・ソフトウェアを含め、適切なネットワーク・ハードウェアとソフトウェアに接続されている必要があります。
- システムは、TCP ポートを経由して相互通信します。接続の両端にあるプロセスは、同じポート番号を使用する必要があります。

- InterSystems IRIS の OPEN、USE、および CLOSE の各コマンドでは、TCP ポート番号あるいは TCP ポート番号を示す devicename のいずれかをデバイスとして指定します。

これらの規則を使用して、TCP バインディング接続を構築します。以下は、一般的な手順です。

1. サーバ・プロセスは、TCP デバイスに OPEN コマンドを発行します。
2. サーバ・プロセスは、USE コマンドを発行し、その後に READ コマンドを発行し、クライアント・プロセスからの入力を待機します。サーバは、クライアントが接続を構築する前に、リッスン状態になる必要があります。クライアントが接続を開いてデータを送信すると、最初の READ コマンドは完了します。OPEN コマンドに“A”モード・パラメータを指定すると、サーバに接続したとき、すぐに最初の READ が完了します。
3. クライアント・プロセスは、接続中の TCP デバイスを指定する OPEN コマンドを発行します。
4. クライアント・プロセスは、USE コマンド、その後 WRITE コマンドを発行して、接続を完了します。InterSystems IRIS は、WRITE コマンドのすべての文字をバッファにコピーします。`WRITE !` あるいは `WRITE #` コマンドを発行してバッファをフラッシュした後、ネットワークに書き込みます。
5. 最初の WRITE コマンドでクライアントが送信した文字をサーバが読み取った後は、サーバとクライアントの両方で継続して READ コマンドと WRITE コマンドを発行できます。同じポートでは、これらのコマンドの順番に制約はありません。
6. どちら側からでも、CLOSE コマンドまたは HALT コマンドで接続を切断できます。最初にクライアント側を閉じることをお勧めします。別のクライアント・プロセスからの接続を受け入れることができるよう、サーバとの接続を切断する必要がある場合は、代わりに `WRITE *-2` コマンドを発行します。

注釈　この手順では、クライアントとサーバの両方が InterSystems IRIS プロセスであると想定しています（実際の環境では、いずれかのプロセスが InterSystems IRIS プロセスではないことがあります）。

以下のセクションは、InterSystems IRIS 入出力コマンドを使用して、クライアントとサーバ・プロセスの間に TCP バインディングを生成する方法を説明しています。

5.2 TCP デバイスの OPEN コマンド

サーバ・プロセスとクライアント・プロセスの両方が、ObjectScript の OPEN コマンドを使用して接続を開始します。サーバは、READ コマンドを発行して接続を完了し、クライアントの OPEN コマンドと最初のデータ転送を受け取ります。

注釈　既に開いている TCP デバイスで、OPEN コマンドを発行すると、この 2 番目の OPEN コマンドは USE コマンドとして扱われます。つまり、hostname および port パラメータは無視され（最初の OPEN コマンドの値は保持）、mode および terminators パラメータが更新されます。

5.2.1 OPEN コマンドの使用法

OPEN コマンドにより、使用する TCP バインディング・デバイスを確保します。構文は、以下のとおりです。

```
OPEN devicename:parameters:timeout:mnespace
```

各項目の内容は次のとおりです。

引数	説明
devicename	フォーム |TCP| の後ろにいくつかの数字を持つ文字列です。デバイス名の数字部分は、デバイス識別子といいます。ポート番号が OPEN の parameters で指定されていない場合は、このデバイス識別子は一意の 5 桁の TCP ポート番号である必要があります。OPEN のパラメータでポート番号を指定している場合は (指定することをお勧めします)、1 つのジョブで使用するすべての TCP デバイス名が識別できる限り、このデバイス識別子には任意の一意の番号 (最大 2147483647) を指定できます。
parameters	オプション — 括弧で囲み、コロン (:) で区切ったデバイス・パラメータを 1 つ以上列記したものです。パラメータを省略する場合は、そのパラメータに該当する位置にコロン区切り文字を記述します (サーバ側で実行する OPEN コマンドでは、最初のパラメータは必ず省略することになります)。具体的なパラメータについては、以下で説明します。 最初のパラメータ (hostname) のみ指定する場合は、括弧を省略できます例えば、クライアント側で開く操作は `OPEN "|TCP|7000":"127.0.0.1":10` となります。パラメータをまったく指定しない場合は括弧を省略できますが、区切り文字のコロンは記述する必要があります。例えば、サーバ側で開く操作は `OPEN "|TCP|7000"::10` となります。
timeout	オプション — InterSystems IRIS が TCP デバイスを開こうとする最大秒数です。この間にファイルを開くことができない場合、$TEST を 0 に設定し、制御をプロセスに戻します。成功した場合、$TEST を 1 に設定します。クライアントからの Open コマンドにタイムアウトを組み込むと、サーバが別のクライアントと通信中に接続を試行した場合に、クライアント・システムが停止することを防ぎます。サーバで一度に開くことができる接続は 1 つのみです。
mnespace	オプション — ObjectScript のすべての OPEN コマンドで、そのままサポートされます。TCP バインディング向けに、事前に定義されたニーモニック空間はありません。

OPEN 引数を省略する場合は、コロン区切り文字を指定することで、引数が省略されていることを表現できます。

timeout 引数は、オプションですが、強くお勧めします。これは、OPEN の成功または失敗が、$TEST 特殊変数の値によって示され、$TEST は、timeout を指定した場合のみ設定されるためです。$TEST は、時間内にオープンが成功すると 1 に設定されます。時間が切れると、$TEST は 0 に設定されます。

Windows システムで TCP 接続の試行が失敗すると、TCP 接続エラーが InterSystems IRIS システムのエラー・ログに書き込まれます (″監視ガイド″ の “管理ポータルを使用した InterSystems IRIS の監視” の章の ″InterSystems IRIS システム・エラー・ログ″ を参照)。例えば、エラー・コード 10061 = WSAECONNREFUSED などが記録されます。

クライアント側の OPEN の例を以下に示します。この例の、7000 はポート番号です。また、″127.0.0.1″ は parameters 引数 (IPv4 アドレス形式で指定した hostname) です。

ObjectScript

```
SET dev="|TCP|7000"
OPEN dev:("127.0.0.1":7000)
```

5.2.1.1 hostname パラメータ

hostname パラメータは、クライアント側の OPEN には必須になります。クライアント側の parameters 引数は、単独の hostname になるか、hostname と、それに続くコンマ区切りのパラメータになります。hostname パラメータを単独で指定する場合は、parameters の括弧を省略できます。

サーバ側の parameters 引数では、hostname を省略します。

hostname は、IP ホストの名前 (リモート・ホストのローカル・システム・データベースから取得) になるか、IPv4 または IPv6 プロトコル形式の IP アドレスになります。これらのプロトコルは互換性がないため、サーバとクライアントの両方ともが同じインターネット・プロトコルを使用しない場合には転送が失敗します。

IPv4 アドレスは以下の形式です。n は、0 から 255 の範囲の 10 進数の整数です。

```
n.n.n.n
```

IPv6 アドレスは以下の完全な形式です。h は、4 桁の16 進数の数値です。

```
h:h:h:h:h:h:h:h
```

通常、IPv6 アドレスは先頭ゼロを削除してゼロの連続したセクションを二重コロン (::) に置き換えることによって省略されます。IPv6 アドレスで使用される二重コロンは 1 つのみです。IPv4 の省略ルールを使用すると、IPv6 ループバック・アドレスを `"::1"` として指定できます (これは、最初から 7 番目までの連続する h のセクションの値が 0000 であり、8 番目のセクションの先行するゼロが削除されていることを表しています)。

OPEN キーワード /USEIPV6 を使用して、使用するプロトコルを指定できます。IPv4 および IPv6 の形式に関する詳細は、"サーバ側プログラミングの入門ガイド" の "サーバ構成オプション" の章にある "IPv6 アドレスの使用" のセクションを参照してください。

5.2.1.2 サポートされるパラメータ

parameters 引数の書式は、以下のいずれかになります。

```
hostname
```

```
(hostname{:port{:mode{:terminators{:ibufsiz{:obufsiz{:queuesize{:keepalivetime}}}}}}})
```

parameters 引数に含まれるパラメータは、以下のとおりです。

パラメータ	意味
hostname	オプション – IP ホストの名前、IPv4 プロトコル形式の IP アドレス、または IPv6 プロトコル形式の IP アドレスのいずれかです。引用符で囲んだ文字列で指定します。hostname は、クライアント側の OPEN には必須です。サーバ側の OPEN では省略します (プレースホルダのコロンで省略を表します)。
port	オプション – 存在する場合、接続に使用する TCP ポート番号です。ポート番号が NULL もしくは省略されている場合、ポート番号は devicename の数値部から取得します。このパラメータは、10 進数のポート番号、またはサービス名のいずれかです。これは、ローカル・システムの TCP サービス名リゾルバに送信されます。

パラメータ	意味
mode	オプション ― 引用符で囲んだ文字コードの文字列です。文字コードは、任意の順序で指定できます。InterSystems IRIS では、左から右の順に実行されるので、文字コード間の相互関係により優先の順序が決まる場合もあります。既定は、パケット・モードです。mode 文字列は、以下の文字 1 つ以上で構成します。 ・ A ― 受け入れモード。A がオンの場合、サーバの最初の読み取りは、クライアント・ジョブからの接続を受け入れるとすぐに、長さゼロの文字列を読み取って終了します。A がオフの場合、タイムアウトになるかデータが入手可能になるかのいずれかになるまで読み取りがブロックされます。 ・ C ― 以下の "キャリッジ・リターン・モード" を参照してください。 ・ D ― 以下の "切断の監視モード" を参照してください。 ・ E ― 以下の "エスケープ・シーケンス処理モード" を参照してください。 ・ G ― port パラメータは、既に開いているデータ・ソケットのソケット記述子として解釈されます。 ・ M ― ストリーム・モードの標準 InterSystems IRIS デバイス。このモードは、"PSTE" のオプション一式を呼び出す省略表現です。デバイスが双方向に任意のデータ行を渡すために使用可能な標準 InterSystems IRIS デバイスのように動作します。バッファの制限を超えずに、あらゆる文字列のシーケンスを送受信できるよう、ストリーム・モードにします。出力には改行が追加され、入力からは改行が削除されます。READ コマンドは、ターミネータ文字が検出されるまで、タイムアウトになるまで、または指定された読み取り長さが読み取られるまでブロックされます。 ・ P ― レコード・ターミネータ文字で出力を埋め込みます。このモードを設定すると、WRITE ! では LF (改行)、WRITE # では FF (改ページ) が送信されます。さらに、書き込みバッファがフラッシュされます。WRITE *-3 コマンドを使用すると、データ・ストリームに文字を挿入することなく、バッファされたデータの送信を開始できます。WRITE *-3 は、ターミネータ文字を送信せずに書き込みバッファをフラッシュするため、データが完了したことを受信プログラムに通知しません。WRITE *-3 は、ターミネータを必要としない待機 (W) モードでより一般的に使用されます。 ・ Q ― 以下の "即時送信モード" を参照してください。 ・ S ― 以下の "ストリーム・モード" を参照してください。 ・ T ― 入力用標準ターミネータ。これを設定すると、CR、LF、および FF の各制御文字が読み取りターミネータとして機能します。 ・ W ― 待機モードです。このモードでは、WRITE ! と WRITE # コマンドによって、TCP デバイスでネットワーク出力バッファがフラッシュされることはありません。待機モードでは、次の WRITE *-3 コマンドがバッファをフラッシュし、データを送信するまで TCP デバイスは待機します。
terminators	オプション ― TCP バインディング・デバイスで読み取りを終了するユーザ・ターミネータ文字を 8 つまでリストできます。T モードと terminators の両方を同時に指定すると、T モードが無視されます。
ibufsiz	オプション ― 入力バッファ・サイズです。内部的に、ネットワークから読み取られる文字は、InterSystems IRIS プログラムへ送信されたものではない場合も、ibufsiz バイトを保持可能なデータ部にバッファされます。

パラメータ	意味
obufsiz	オプション－出力バッファ・サイズです。連続する“SEND”処理の間に TCP デバイスでバッファできる最大データ量です。SEND 処理は、バッファされたデータをネットワーク外に送信することを意味します。WRITE !、WRITE #、および WRITE *-3 の各コマンドは、SEND 処理を生成できます。 S モードを指定した場合、バッファが一杯になると SEND 処理が自動的に生成され、出力バッファの内容が送信されます。それでも、メッセージの生成を完了した後、プログラマは SEND 処理の 1 つを使用し、メッセージが送信されたことを確認する必要があります。 S モードを指定していない場合は、出力バッファ・サイズを超えるデータを WRITE 処理でバッファに置くと、<WRITE> エラーが発生します。出力バッファ・サイズよりも長い文字列を書き込もうとすると、必ず失敗します。
queuesize	オプション－サーバ接続を待機する状態に置けるクライアント・ジョブの最大数を指定する整数です。サーバ側の OPEN のみで使用できます。既定値は 5 です。最大値は TCP の実装で異なりますが、どのような場合も 1,000 を超えることはできません。
keepalivetime	オプション－(Windows、AIX、Linux のみ) このデバイスに、システムの既定とは異なるキープアライブ・タイマを設定することができます。TCP 接続を保持する秒単位の整数を指定します。有効値は、30 ～ 432000 の範囲 (432,000 秒は 5 日) です。30 未満の値は、既定で 30 となります。省略した場合または 0 に設定した場合は、システム全体の既定のキープアライブ・タイマが使用されます。詳細は、/KEEPALIVE キーワード・オプションを参照してください。 キープアライブ・タイマーは、必ずしも TCP デバイスが開かれたときに時間計測を開始するわけではありません。通常、接続が確立されたときに時間計測を開始します。すなわち、最初の接続のための読み取りが正常に完了したときに開始します。

5.2.1.3 パケット・モード

mode が指定されていない場合、既定でパケット・モードになります。ストリーム・モードが無効の場合、モードは既定でパケット・モードになります。

パケット・モードでは、返すデータが存在する場合はすぐに READ コマンドを完了します。パケット・モードにより、出力バッファに TCP セグメント全体を構築でき、WRITE *-3 あるいは WRITE ! コマンドを発行して、一度にセグメントを送信できます。

送信する文字が存在しないときに、WRITE *-1 を発行し TCP SEND 処理を起動すると、<WRITE> エラーを受け取ります。文字列が空の場合に WRITE を発行すると、<COMMAND> エラーを受け取ります。

パケット・モードで送信できる最大文字列サイズは 1,024 文字です。バッファをフラッシュせずにこの制限サイズを超えると、<WRITE> エラーを受け取ります。

TCP/IP は長さが 0 のレコードを無視するため、文字が存在しない場合に書き込みバッファをフラッシュすると、<WRITE> エラーを受け取ります。

サーバが接続要求を受け取る前に、サーバからクライアントに WRITE コマンドを要求すると、サーバ上で <WRITE> エラーを生成します。

5.2.1.4 キャリッジ・リターン・モード (C モード)

このモードは、入出力でキャリッジ・リターン処理を変更します。

出力では、WRITE ! は“CR LF”を生成し、WRITE # は“CR FF”を生成します。

入力では、T モードが有効の場合、サーバは隣接する CR と LF、あるいは隣接する CR と FF を、単一のターミネータとして $ZB に記録します。CR と LF が、それぞれ短いインターバルの間に生成されない場合、別のターミネータとして処理されます。既定のインターバルは 1 秒です。

5.2.1.5 切断の監視モード (D モード)

このモードで、非同期切断の監視をオンまたはオフにします。このモードは、“D” モード文字、/POLL キーワード・パラメータ、または /POLLDISCON キーワード・パラメータを指定することで有効になります。+D を指定すると、TCP 切断の監視がアクティブになり、-D を指定すると TCP 切断の監視が非アクティブになります。

アクティブの間は、InterSystems IRIS によって約 60 秒間隔で TCP 接続がポーリングされます。切断が検出された場合は、<DISCONNECT> エラーが発行されます。切断検出は、HANG コマンドによって中断されたジョブや READ 処理で待機中のジョブなどアイドル・ジョブでは行われません。InterSystems IRIS は、<DISCONNECT> エラーが発生しないように、ロールバック処理中は切断の監視をすべて一時停止します。ロールバックが終了すると、InterSystems IRIS は切断の監視を再開します。この一時停止は、切断の監視が有効な現在の TCP デバイスのほか、切断の監視が無効な現在のデバイスのうち、切断の監視が有効な TCP デバイスに接続しているデバイスにも適用されます。

%SYSTEM.INetInfo クラスの Connected() メソッドを使用して、TCP 切断をチェックすることもできます。

5.2.1.6 エスケープ・シーケンス処理モード (E モード)

E モードが設定された場合、入力ストリームのエスケープ・シーケンスは解析され、$ZB 特殊変数に格納されます。エスケープ・シーケンスは、15 文字あるいはそれ以下の必要があり、構文は以下のとおりです。

```
esc_seq::=type1 | type2
```

以下はその説明です。

```
type1 ::= '['['0':'?']*['':'/']*{'@':DEL}
type2 ::= [';'|'?'|'O'][':'/']*{'0':DEL}
```

ここで使用されている構文記号の意味は次のとおりです。

記号	説明
:	x:y は、ASCII シーケンスで x から y の指定範囲の文字を意味します。
|	x|y は、x または y を意味します。
[ ]	0 あるいは指定されたセットのいずれか 1 つのメンバを指定します。
[ ]*	0 あるいは指定されたセットのいずれか 1 つ以上のメンバを指定します。
{ }	指定されたセットのいずれか 1 つのメンバを指定します。

InterSystems IRIS が ESCAPE を検出すると、他のエスケープ・シーケンスが出現するかどうか 1 秒間待機します。エスケープ・シーケンスがこの構文に一致しない場合、長さが 15 文字を超過する場合、あるいは有効なエスケープ・シーケンスが 1 秒以内に到着しない場合、InterSystems IRIS は、一部のエスケープ・シーケンスを $ZB に配置し、“BADESC” ビット (256) を $ZA に設定します。

5.2.1.7 即時送信モード (Q モード)

即時送信モードでは、WRITE コマンドごとにそれぞれのパケットを出力できます。即時送信モードを使用しない場合は、ターミネータを指定するか WRITE *-3 コマンドを発行してパケットを出力する必要があります。

このモードにするには、“Q” モード文字または /SENDIMMEDIATE キーワード・パラメータ (/SEN という表記も可) を指定します。このオプションを無効にするには、次のいずれかを指定します。

ObjectScript

```
USE TCPDEVICE:(/SEN=0)
USE TCPDEVICE:(::"-Q")
```

このオプションを有効にするには、次のいずれかを指定します。

ObjectScript

```
USE TCPDEVICE:(/SEN=1)
USE TCPDEVICE:(::"+Q")
```

書き込みごとに 1 つのパケットを生成する即時送信モードは、各パケットを作成と同時に直ちに送信する /NODELAY モードと組み合わせて使用できます。両方を有効にすると、1 回のデータの転送速度がきわめて速くなります。これは、マウス移動の転送のように、適切な時点で各データ・ユニットを送信する必要がある場合に役立ちます。両方を無効にすると、複数の書き込みを収めたパケットや、複数のパケットの同時転送などが可能になります。その場合は、ネットワーク・トラフィックが減少するので全体的なパフォーマンスが向上します。既定では即時送信モードは無効になっています。/NODELAY モードは既定で有効です。

5.2.1.8 ストリーム・モード (S モード)

ストリーム・モードでは、InterSystems IRIS は データ・ストリームに TCP メッセージの範囲を保存しようとしません。送信時、データがメッセージ・バッファ・サイズに適合しない場合、InterSystems IRIS は、データをバッファに配置する前に、バッファをフラッシュします。

受信時、最大長までの文字列データを受信できます。すべての読み取りは、ターミネータが完全なタイムアウトに達するまで、あるいはバッファが一杯になるまで待機します。このモードが無効の場合 (既定)、パケット・モードになります。

TCP デバイスを継承するジョブ起動プロセスは、自動的にストリーム形式に設定されます。USE コマンドを使用して、この形式をリセットできます。

5.2.1.9 バッファ・サイズ

TCP デバイスの ibufsiz パラメータおよび obufsiz パラメータは、TCP 入出力の内部 InterSystems IRIS バッファのサイズを指定します。これにはサポート対象のすべてのプラットフォーム上で、1 KB から 1 MB の範囲内の値を指定できます。ただし、オペレーティング・システムのプラットフォームは、それ自体の入出力バッファに異なるサイズを使用する場合があります。オペレーティング・システム・プラットフォームのバッファが InterSystems IRIS バッファよりも小さい場合 (1 MB に対して 64 KB しかないなど)、パフォーマンスが影響を受ける可能性があります。例えば、WRITE 操作は InterSystems IRIS バッファ全体を送信するために OS に何度も要求を出すことが必要になる可能性があり、READ 操作は OS バッファ・サイズにより制限される複数の小さいチャンクを返す可能性があります。最適なパフォーマンスを得るために、ibufsize および obufsize にどの値を指定すると最適な結果になるかを見極めるには、現在の OS でテストすることが必要です。

5.2.2 サーバ側の OPEN コマンド

サーバ側で OPEN コマンドが処理されると、TCP ソケットが構築され、適切なポート番号に接続要求が入力されるのをソケットで待ち受けます。ポート番号は、パラメータ・リストで明示的に指定します。指定していない場合は、devicename の数値部から取得されます。ソケットが待ち受け状態になるとすぐに、OPEN コマンドが返されます。

OPEN が失敗した場合、別のプロセスがポート番号の接続要求を既に待ち受けている場合があります。

以下の例は、サーバ側の OPEN でデバイスを指定します。これにより、最大文字列サイズ以下の文字列を読み書きできます。また、最大文字列読み取りを使用し、TCP チャネルの使用を統合する処理を書き込みます。

ObjectScript

```
OPEN "|TCP|4":(:4200:"PSTE"::32767:32767)
```

この例の parameters 引数は次のようになります。これは、サーバ側の OPEN コマンドなので、最初のパラメータ (hostname) が省略されています。2 番目のパラメータには明示的にポート番号 (4200) を指定します。3 番目のパラメータは、mode コード文字です。4 番目のパラメータ (terminators) は省略されています。5 番目のパラメータは入力バッファのサイズを指定します。6 番目のパラメータは出力バッファのサイズを指定します。

以下の例では、ポート番号はパラメータとして指定されません。devicename の数値部から取得します。この例では、パラメータを指定せず、10 秒のタイムアウトでポート 4200 を開きます。

ObjectScript

```
OPEN "|TCP|4200"::10
```

サーバ側の OPEN の入力バッファのサイズ (ibufsiz) と出力バッファのサイズ (obufsiz) の既定のパラメータ値は 1,048,576 バイト (1 MB) です。

サーバ側の OPEN コマンドは、オプションの queuesize パラメータと、オプションの “G” モード・パラメータをサポートしています。これらのオプションは、クライアント側の OPEN コマンドでは利用できません。

サーバ側の OPEN は、オプションの /CLOSELISTEN キーワード・パラメータをサポートします。このオプションは、クライアント側の OPEN コマンドでは利用できません。

5.2.3 クライアント側の OPEN コマンド

クライアント側の OPEN コマンドがサーバ側の OPEN コマンドと異なる点は、接続するホストを最初のデバイス・パラメータで指定する必要があるということだけです。ホストを指定するには、クライアントでホストあるいはインターネット・アドレスとして認識できる名前を使用します。

OPEN コマンドは、接続されるとすぐに実行します。この時点で、TCP デバイスへの読み取り、あるいは書き込みが可能です。ただし、サーバ側の接続が別の InterSystems IRIS プロセスの場合、WRITE コマンドによるクライアントからサーバへのデータ送信が実行されない限り、サーバ側の接続は完了しません。したがって、READ コマンドの発行前に WRITE コマンドを発行する必要があります。

詳細は、“TCP デバイスの WRITE コマンド” のセクションを参照してください。

以下は、クライアント側の OPEN コマンドの例です。

ObjectScript

```
OPEN "|TCP|4":("hal":4200::$CHAR(3,4)):10
```

このコマンドは、ポート 4200 にホスト hal への接続を開きます。モード文字は指定されていません。2 つのターミネータ (ASCII $CHAR(3) および $CHAR(4))、既定の入力バッファ・サイズと出力バッファ・サイズを指定します。さらに、10 秒のタイムアウトを指定します。

以下のコマンドは、宛先の IP アドレスを IPv4 形式で明示的に記述している以外は、上記のコードと同じです。

ObjectScript

```
OPEN "|TCP|4":("129.200.3.4":4200::$CHAR(3,4)):10
```

OPEN キーワード /USEIPV6 を使用して、使用するプロトコルを指定できます。IPv4 および IPv6 の形式に関する詳細は、″サーバ側プログラミングの入門ガイド″ の “サーバ構成オプション” の章にある “IPv6 アドレスの使用” のセクションを参照してください。

以下のコマンドは、リモート・ホスト “larry” の時刻サーバに接続し、主要入力デバイスにリモート・ホストの日時を ASCII 形式で出力します。このコマンドではサービス名 `daytime` を使用しますが、これはローカル・システムでポート番号に解決されます。

ObjectScript

```
OPEN "|TCP|4":("larry":"daytime":"M")
USE "|TCP|4"
READ x
USE 0
WRITE x
```

以下のコマンドは、x を“hello”に設定します。

ObjectScript

```
OPEN "|TCP|4":("larry":"echo":"M")
USE "|TCP|4"
WRITE "hello",!
READ x
```

以下のコマンドは、インターネット・アドレス 128.41.0.73、ポート番号 22101 を、30 秒のタイムアウトで接続します。

ObjectScript

```
OPEN "|TCP|22101":"128.41.0.73":30
```

5.2.4 TCP デバイスの OPEN コマンド・キーワードと USE コマンド・キーワード

上記で説明した位置パラメータ、またはキーワード・パラメータを使用できます。以下のテーブルは、OPEN コマンドと USE コマンドの両方を使用して、TCP デバイスを制御するキーワードの説明です。OPEN コマンドのみで指定可能な OPEN コマンド専用のキーワード (この章で後述) もあります。キーワード・パラメータはすべてオプションです。

テーブル 5-1: TCP デバイスの OPEN コマンド・キーワードと USE コマンド・キーワード

キーワード	既定	説明
/ABSTIMEOUT[=1]	0	読み取りタイムアウトの動作を指定します。データを受信した際に TCP によってタイムアウト期間を再初期化するかどうかを決定します。/ABSTIMEOUT=0 (既定) の場合は、各時刻データの受信時にタイムアウトを元の値にリセットします。/ABSTIMEOUT もしくは /ABSTIMEOUT=1 の場合は、データを受信する間、タイムアウト期間のカウント・ダウンを続けます。
/ACCEPT[=n] または /ACC[=n]	0	“A” モード・パラメータ文字に相当します。クライアント・ジョブからの接続が受け入れられるとすぐに、サーバでの初期の読み取りが長さ 0 の文字列で終了するように指定します。/ACCEPT と /ACCEPT=n の n が 0 以外の場合、A モードが有効になります。/ACCEPT=n の n が 0 の場合、A モードが無効になります。
/CLOSEFLUSH[=n]	1	デバイスが閉じられている場合、出力バッファに残っているデータの処理を指定します。/CLOSEFLUSH と /CLOSEFLUSH=n の n が 0 以外の場合、残りのデータをフラッシュします。/CLOSEFLUSH=n の n が 0 の場合、残りのデータを破棄します。

キーワード	既定	説明
/COMPRESS=str	""	ストリーム・データの圧縮タイプを指定します。圧縮タイプ ZLIB または ZSTD を有効にすることができます。/COMPRESS="" を指定すると、圧縮を無効にすることができます。/COMPRESS="zlib" は /GZIP=1 と同等です。文字列を圧縮するには、%SYSTEM.Util.Compress() を使用します。
/CRLF[=n]	0	“C” モード・パラメータ文字に相当します。入出力を戻すキャリッジ・リターン処理を変更します。/CRLF と /CRLF=n の n が 0 以外の場合、C モードが有効になります。/CRLF=n の n が 0 の場合、C モードが無効になります。
/ESCAPE[=n] または /ESC[=n]	0	“E” モード・パラメータ文字に相当します。入力ストリームでエスケープ・シーケンスを構文解析し、$ZB に配置するように指定します。/ESCAPE と /ESCAPE=n の n が 0 以外の場合、E モードが有効になります。/ESCAPE=n の n が 0 の場合、E モードが無効になります。
/GZIP[=n]	1	GZIP と互換性のあるストリーム・データ圧縮を指定します。/GZIP を指定した場合、または /GZIP=n (n は 0 以外) を指定した場合、WRITE の発行時に圧縮、READ の発行時に解凍が有効になります。/GZIP=0 を指定した場合は、圧縮と解凍が無効になります。/GZIP=0 を発行して、圧縮/解凍を無効にする前に、$ZEOS 特殊変数をチェックして、ストリーム・データの読み込みが実行中でないことを確認してください。/GZIP 圧縮は、/IOTABLE を使用して構築した変換などの入出力変換には影響しません。これは、圧縮がその他すべての変換 (暗号化を除く) の後に適用され、解凍がその他すべての変換 (暗号化を除く) の前に適用されるためです。圧縮データで使用する WRITE に関する詳細は、この章の "WRITE コントロール・コマンド" を参照してください。
/IOTABLE[=name] または /IOT[=name]	name が指定されない場合、デバイスの既定の入出力変換テーブルを使用します。	デバイスの入出力変換テーブルを構築します。
/KEEPALIVE=n	システムの既定	(Windows、AIX、Linux のみ) このデバイスに、システムの既定とは異なるキープアライブ・タイマを設定することができます。TCP 接続を保持する秒数を指定する整数。keepalivetime 位置パラメータと同じです。有効値は、30 ～ 432000 の範囲 (432,000 秒は 5 日) です。30 未満の値は、既定で 30 となります。省略した場合または 0 に設定した場合は、システムの既定値が使用されます。/NOKEEPALIVE を使用して、この設定を無効にできます。一度無効にすると、TCP デバイスが閉じるまで再度有効にすることはできません。

キーワード	既定	説明
/NODELAY=n	1	パケットをまとめて送信するか、個々に送信するかを指定します。/NODELAY=1 (既定) に指定すると、パケット単位で直ちに転送されます。/NODELAY=0 に指定すると、最適化アルゴリズムを使用して TCP ドライバでパッケージが 1 つにまとめられます。これにより、パケット単位で見るとわずかな転送遅延が生じる場合がありますが、ネットワーク・トラフィックが減少することで、全体的なパフォーマンスの向上が望めます。/NODELAY に相当するモード・パラメータ文字はありません。/NODELAY は、/SENDIMMEDIATE との関係を考慮して使用する必要があります。
/NOKEEPALIVE		指定した場合、システム規模の TCP キープアライブ・タイマはこのデバイスに対して無効になります。InterSystems IRIS の既定では、TCP デバイスを開く際に、このタイマが有効になります。OPEN または USE で /NOKEEPALIVE オプションを使用すると、この既定がオーバーライドされます。/KEEPALIVE を使用して既定以外のキープアライブ・タイマを設定している場合は、/NOKEEPALIVE によりそのキープアライブ・タイマを無効にできます。一度キープアライブ・タイマを無効にすると、TCP デバイスを閉じるまで再度有効にすることはできません。"/KEEPALIVE" を参照してください。
/NOXY [=n]	0	$X および $Y の処理なし : /NOXY を指定した場合、または /NOXY=n (n は 0 以外の値) を指定した場合、$X および $Y の処理が無効になります。例えば、CSP などで、デバイスの $X/$Y を使用しない場合に、パフォーマンスを向上できます。これにより READ 操作および WRITE 操作のパフォーマンスを大幅に向上させることができます。このオプションは、スーパーサーバ・ワーカ・ジョブの既定の設定です。/NOXY=1 の場合、$X および $Y の変数値が不確定であるため、マージン処理 ($X に依存) は無効になります。/NOXY=0 の場合は、$X および $Y の処理が有効になります。これが既定です。/TCPNOXY は /NOXY の非推奨の同義語です。
/OBCOUNT=n	16	/ZEROCOPY で使用する出力バッファの数。既定の出力バッファ数は 16 です。出力バッファの最小数は 2 で、最大数は 128 です。n の値は 2 の累乗である必要があります。2 の累乗でない値を指定すると、2 の累乗に切り上げられます。
/PAD[=n]	0	“P” モード・パラメータ文字に相当します。これは WRITE ! (LF ターミネータ) または WRITE # (FF ターミネータ) の実行時に、出力にレコード・ターミネータ文字が埋め込まれることを指定します。/PAD と /PAD=n の n が 0 以外の場合、P モードが有効になります。/PAD=n の n が 0 の場合、P モードが無効になります。

キーワード	既定	説明
/PARAMS=str または /PAR=str	既定なし	mode 位置パラメータに相当します（位置に依存しない方法でモード文字列を指定する方法を提供します）。
/POLL[=n] または /POLLDISCON[=n]		切断の非同期監視を指定する“D”モード・パラメータ文字に相当します。/POLL または /POLL=1 は +D に相当し、/POLL=0 は -D に相当します。
/PSTE[=n]	0	“M”モード・パラメータ文字に相当します。これは、P、S、T、E モード・パラメータ文字を指定する省略表現です。/PSTE と /PSTE=n の n が 0 以外の場合、P、S、T、E モードが有効になります。/PSTE=n の n が 0 の場合、これらのモードが無効になります。
/SENDIMMEDIATE[=n] または /SEN[=n]	0	“Q”モード・パラメータ文字に相当します。これは、即時送信モードを指定します。
/SSL="cfg[|pw] [|DNShost]" or /TLS="cfg[|pw] [|DNShost]"	既定なし	クライアントが指定した構成とサーバ要件に従い、SSL/TLS で保護された接続のネゴシエーションをデバイスで試行するようにクライアントから指定します。サーバとしてソケットを保護する場合は、サーバが指定した構成とクライアント要件に従い、SSL/TLS で保護された接続をサーバで必要とすることを指定します。 cfg は、接続またはソケットの構成名を指定します。pw は、秘密鍵ファイルのオプションのパスワードを指定します。DNShost は、特定のサーバの完全修飾された DNS ホスト名を指定します。これは、サーバー ネーム インディケーション（SNI）の TLS Extension で使用します。詳細は以下を参照してください。 この構成名は、OPEN または USE コマンド後の初回の I/O 動作時にのみ使用されます。それ以降の呼び出しは無視されます。/SSL="" または /TLS="" は無視されます。詳細は、インターシステムズの "TLS ガイド" を参照してください。 重要 : SSL/TLS を使用している新規の TCP 接続を開くときまたはそのような既存の接続を保護するときにパスワードを含める機能は、リアルタイムのインタラクティブを使用する場合にのみ有効です。保護していない秘密鍵パスワードを永続的に保存することは絶対に避けてください。そのようなパスワードを保存する必要がある場合は、Security.SSLConfigs クラスの PrivateKeyPassword プロパティを使用します。

キーワード	既定	説明
/STREAM[=n] または /STR[=n]	0	“S” モード・パラメータ文字に相当します。TCP メッセージ範囲を保存しないデータを処理するストリーム・モードを指定します。/STREAM と /STREAM=n の n が 0 以外の場合、S モードが有効になります。/STREAM=n の n が 0 の場合、S モードが無効になります。
/TCPNOXY		非推奨。/NOXY の同義語です。
/TCPRCVBUF=n	既定の受信バッファ・サイズ。	受信キューのバッファ・サイズをバイト単位で設定します。TCP プロトコルでサイズの大きいウィンドウをサポートするために、バッファ・サイズを既定値から増やす場合に使用できます。ウィンドウのサイズが大きいと、長い遅延を持つリンクまたは広帯域リンクのパフォーマンスが向上します。適切な値については、使用しているオペレーティング・システムやハードウェアのドキュメントを参照してください。
/TCPSNDBUF=n	既定の送信バッファ・サイズ。	送信キューのバッファ・サイズをバイト単位で設定します。TCP プロトコルでサイズの大きいウィンドウをサポートするために、バッファ・サイズを既定値から増やす場合に使用できます。ウィンドウのサイズが大きいと、長い遅延を持つリンクまたは広帯域リンクのパフォーマンスが向上します。適切な値については、使用しているオペレーティング・システムやハードウェアのドキュメントを参照してください。
/TERMINATOR=str または /TER=str	既定なし	ユーザ定義のターミネータを構築する terminators 位置パラメータに相当します。
/TMODE[=n] または /TMO[=n]	0	“T” モード・パラメータ文字に相当します。標準読み取りターミネータで CR、LF、FF を指定します。/TMODE と /TMODE=n の n が 0 以外の場合、T モードが有効になります。/TMODE=n の n が 0 の場合、T モードが無効になります。
/TRANSLATE[=n] または /TRA[=n]	1	/TRANSLATE を指定した場合、または /TRANSLATE=n の n が 0 以外の場合、デバイスの入出力変換が有効になります。/TRANSLATE=n の n が 0 の場合はデバイスの入出力変換が無効になります。
/WAIT[=n]	0	“W” モード・パラメータ文字に相当します。出力バッファが、WRITE ! と WRITE # コマンドでフラッシュされないようにします。フラッシュは次の WRITE *-3 コマンドまで待機します。/WAIT と /WAIT=n の n が 0 以外の場合、W モードが有効になります。/WAIT=n の n が 0 の場合、W モードが無効になります。

キーワード	既定	説明
/WRITETIMEOUT[=n]	-1	TCP の書き込み操作のためのタイムアウト(秒)を設定します。書き込みが n 秒以内で完了しない場合、InterSystems IRIS は <TCPWRITE> エラーを発行します。<TCPWRITE> エラーが発行された場合、ユーザ・アプリケーションで即座に TCP デバイスを閉じて、データ損失を防止する必要があります。InterSystems IRIS では、<TCPWRITE> エラーに続けて TCP の書き込み操作を試行することはありません。n の最小値はシステムによって異なります。n がプラットフォームの最小タイムアウト値よりも小さい場合、InterSystems IRIS ではプラットフォームの最小値が使用されます。n を 2 未満とすることはできません。既定 (-1) はタイムアウトの強制がないことを示しています。
/XYTABLE[=name] または /XYT[=name]	name が指定されていない場合、デバイスの既定の $X/$Y アクション・テーブルを使用します。	デバイスの $X/$Y アクション・テーブルを構築します。/NOXY を参照してください。
/ZEROCOPY[=bool]	0	/ZEROCOPY または /ZEROCOPY=1 が指定されている場合、この TCP デバイスの ZEROCOPY 機能を有効にします。ZEROCOPY は send() 時に MSG_ZEROCOPY を利用するために、TCP デバイスの複数の出力バッファをサポートします。バッファが TCP スタックからの通知によって確認されるまで、TCP デバイスは出力バッファを再利用できません。TCP デバイスの出力バッファ数を設定するには、/OBCOUNT キーワードを使用します。/ZEROCOPY=0 が指定されている場合、この TCP デバイスの ZEROCOPY 機能を無効にします。 MSG_ZEROCOPY は、Linux 4.15 以降でサポートされます。オペレーティング・システムで MSG_ZEROCOPY がサポートされない場合、TCP デバイスの ZEROCOPY は常に無効です。

5.2.4.1 SSL/TLS コンポーネント

TCP デバイス OPEN または USE /SSL または /TLS キーワード・パラメータの値は、引用符付き文字列です。この文字列は、1 つ、2 つ、または 3 つのコンポーネントを '|' 文字で区切って持つことができます。

パラメータ	説明
cfg	この接続に使用する SSL 構成の名前。このコンポーネントは必須です。
pw	オプション – ローカルの秘密鍵ファイルのパスワード。ユーザが昇格されて実行時にパスワードを入力するとき、これはインタラクティブなアプリケーション専用です。これは、永続的に格納されるパスワードでは使用しないでください。永続ストレージには Security.SSLConfigs.PrivateKeyPassword プロパティを使用します。
DNShost	オプション – SSL クライアント専用。サーバ選択証明書 (ホスト名検証用) または特定のサーバの完全修飾 DNS ホスト名 (サーバ名表示用) のいずれかを指定します。 pw を省略した場合、プレースホルダ '|' 文字を指定する必要があります。 ホスト名検証は、クライアントがサーバから受信した証明書に、クライアントが接続しようとしたホスト名のフィールドが含まれていることを、クライアントが確認できるようにする機能です。これは、URL のサーバ名に対応する、完全修飾されたサーバ DNS ホスト名がサーバの X.509 証明書 (subjectAltName Extension または Subject CN フィールド) に含まれていることを確認するクライアント・アプリケーション (%Net.HttpRequest() など) が使用します。これにより、クライアントは、中間者攻撃で間違ったドメインに対して有効な証明書を使用するケースを検出することができます。 サーバ名表示 (SNI) は、クライアントが要求しているホスト名をサーバに送信できるようにする機能です。これにより、複数のドメインを処理するサーバが、複数の証明書のうちの 1 つを選択して返すことができます。サーバは、クライアント上のホスト名の照合に一致するものを選択できます。

以下に示すのは、有効な /TLS キーワード・パラメータの例です。

```
/TLS="Client"
/TLS="Client|password"
/TLS="Client|www.intersystems.com"
/TLS="Client|password|www.intersystems.com"
```

5.2.5 TCP デバイスの OPEN コマンドのみのキーワード

以下の表は、OPEN コマンドのみで指定可能な TCP デバイスを制御するキーワードの説明です。OPEN コマンドまたは USE コマンドで指定可能な OPEN/USE コマンドのキーワード (この章で前述) もあります。キーワード・パラメータはすべてオプションです。

テーブル 5-2: TCP デバイスの OPEN コマンドのみのキーワード

キーワード	既定	説明
/BINDTO[=address]		接続の開始時に使用される指定ローカル・アドレスにバインドします。クライアントの場合は、InterSystems IRIS からの TCP/IP 接続をオープンするときに使用されるソース・アドレスです。サーバの場合は、InterSystems IRIS プロセスが TCP/IP 接続をオープンするときに接続を受け付ける IP アドレスです。/BINDTO=address は、接続で使用されるネットワーク・インタフェースを制御するために使用します。/BINDTO を指定した場合、または /BINDTO="" を指定した場合、以前に指定したアドレスは削除されます。指定された address が存在しない場合、OPEN コマンドはタイムアウトします。

キーワード	既定	説明
/CLOSELISTEN		(サーバのみ) 待ち受けポートへの複数のリモート接続を防止します。指定すると、最初の接続が受け入れられた後、待ち受けソケットが閉じられます。接続を試みる追加クライアントは、OPEN コマンドでタイムアウトします。
/CONNECTIONS=n または /CON=n	5	queuesize 位置パラメータに相当します。サーバへの接続のために待機できるクライアント数を決定します。
/HOSTNAME=str または /HOS=str	既定なし	hostname 位置パラメータに相当します。IP ホスト名か、IPv4 または IPv6 アドレス形式の IP アドレスを指定します。/USEIPV6 キーワードを使用して、使用するプロトコルを指定できます。IPv4 および IPv6 の形式に関する詳細は、"サーバ側プログラミングの入門ガイド" の "サーバ構成オプション" の章にある "IPv6 アドレスの使用" のセクションを参照してください。
/IBUFSIZE=n または /IBU[=n]	1024	ibufsiz 位置パラメータに相当します。ネットワークからデータを読み取り、アプリケーションに送信するまで保持する TCP 入力バッファのサイズを指定します。
/OBUFSIZE=n または /OBU[=n]	1024	obufsiz 位置パラメータに相当します。連続する "SEND" 処理間でデータを保持する TCP 出力バッファのサイズを指定します。
/PORT=n	既定なし	port 位置パラメータに相当します。接続に使用する TCP ポート番号あるいはサービス名です。
/SOCKET=n または /SOC=n	既定なし	"G" モード・パラメータ文字に相当します。port 位置パラメータが、既に開いているデータ・ソケットのソケット記述子として解釈されます。このキーワードはソケット記述子の値を取得し、/PORT=n キーワードの代わりに使用されます (ソケット記述子は、InterSystems IRIS のコールインまたはコールアウト・メカニズムを使用して、C など他のプログラミング環境から ObjectScript に渡されます)。

次は、キーワード構文を使用して TCP/IP デバイスを開く例です。

ObjectScript

```
SET dev="|TCP|"_123
SET portnum=57345
OPEN dev:(/PSTE:/HOSTNAME="128.41.0.73":/PORT=portnum)
```

5.3 現在の TCP デバイス

%SYSTEM.TCPDevice クラスのメソッドを使用して、現在の TCP デバイスの IP アドレスおよびポート番号を返すことができます。以下のように Help() メソッドを使用してこれらのメソッドをリスト表示できます。

ObjectScript

```
DO $SYSTEM.TCPDevice.Help()
```

以下の例に示すように、Help() でメソッド名を指定して、特定のメソッドの情報を表示できます。

ObjectScript

```
DO $SYSTEM.TCPDevice.Help("LocalAddr")
```

5.4 TCP デバイスの USE コマンド

クライアント、サーバのいずれかから発行された USE コマンドにより、事前に開いている TCP 接続を使用して、データの送受信の準備をします。このコマンドの構文は次のとおりです。

```
USE devicename:(::mode:terminators)
```

各項目の内容は次のとおりです。

パラメータ	説明
devicename	フォーム |TCP| の後ろにいくつかの数字を持つ文字列です。デバイス名の数字部分は、デバイス識別子といいます。OPEN のパラメータでポート番号を指定していない場合、このデバイス識別子には一意な 5 桁の TCP ポート番号を指定する必要があります。OPEN のパラメータでポート番号を指定している場合は (指定することをお勧めします)、1 つのジョブで使用するすべての TCP デバイス名が識別できる限り、このデバイス識別子には任意の一意の番号を指定できます。
mode	オプション — USE コマンドは、OPEN と同じパラメータをサポートします。詳細は、“TCP デバイスの OPEN コマンド・キーワードと USE コマンド・キーワード” を参照してください。
terminators	オプション — TCP バインディング・デバイスで読み取りを終了するユーザ・ターミネータ文字を 8 つまでリストできます。T モードとユーザ・ターミネータの両方を同時に指定する必要はなく、両方を指定すると、T モードが無視されます。

USE コマンドの最も簡素な形は、次の例に示すように、モード・パラメータとターミネータ・パラメータを OPEN コマンドから受け取るものです。

ObjectScript

```
USE "|TCP|4"
```

デバイスを開いた後で、モード・パラメータとユーザ・ターミネータを置き換え、追加、または削除できます。

OPEN コマンドで指定したパラメータを置き換えるには、置き換える値を USE コマンドで指定します。以下の例にある USE コマンドは、OPEN モードを PSTE モードに置き換え、すべてのユーザ・ターミネータをオフにします。

ObjectScript

```
USE "|TCP|4":(::"PSTE")
```

OPEN で指定したモード・パラメータを追加または削除するには、“+” 記号を使用して、オンにするモード・パラメータを導き、“-” 記号を使用して、オフにするモード・パラメータを導きます。“+” も “-” も指定しない場合、既存のすべてのモード・パラメータは新規のモード・パラメータ一式に置き換えられます。次の例にある USE コマンドでは、Q モード (直ちに送信) をオンにし、W モード (待機) をオフにします。モード文字列の残り部分は、変更されないままとなります。

ObjectScript

```
USE "|TCP|4":(::"-Q+W")
```

以下の例では、USE コマンドはモード文字列を変更せずに、新規のユーザ・ターミネーター一式を指定します。

ObjectScript

```
USE "|TCP|4":(::"+":$CHAR(3,4))
```

5.5 TCP デバイスの READ コマンド

サーバ、クライアントのいずれかから発行した READ コマンドは、クライアント、サーバのいずれかで設定されたあらゆる文字を読み取ります。

構文は、以下のとおりです。

```
READ var:timeout
READ *var:timeout
READ var#length:timeout
```

timeout 引数は、オプションですが、強くお勧めします。これは、timeout が指定されている場合、READ の成功または失敗が、$TEST 特殊変数の値によって示されるためです。$TEST は、時間内に読み取りが成功すると 1 に設定されます。時間が切れると、$TEST は 0 に設定されます。

timeout 引数は、秒および 1/100 秒までの小数部をサポートします。例えば、10、10.5、.5、.05 です。

SSL 接続では、接続の確立後に他方が読み取りコマンドまたは書き込みコマンドを発行したことがない場合、最初の読み取りコマンドまたは最初の書き込みコマンドでジョブは待機できます。この状況で InterSystems IRIS は、READ コマンドの読み取りタイムアウトおよび WRITE コマンドの書き込みタイムアウト (/WRITETIMEOUT=n オプション) をサポートします。読み取りまたは書き込みのタイムアウトが指定されていない場合、ジョブは他方が読み取りコマンドまたは書き込みコマンドを発行するまで待機します。

%SYSTEM.INetInfo クラスの TCPStats() メソッドを使用すると、現在の TCP 接続で実行する読み取りの回数を指定できます。

5.5.1 READ による $ZA および $ZB の変更

アプリケーションは、$ZA と $ZB の値を検証し、接続と読み取りがどのように成功したかを認識できます。

5.5.1.1 $ZA と READ コマンド

$ZA は、接続の状態を通知します。設定が 0x1000 ビット (4096) の場合、TCP デバイスはサーバ・モードで機能しています。0x2000 ビット (8192) の場合、リモート・ホストと対話している接続状態です。

例えば、サーバ側の TCP デバイスが、新規 TCP 接続を受け取ると仮定します。InterSystems IRIS プログラムでは、初期のタイムアウト付き読み取り後の $ZA と $TEST は、以下の 3 つに区分できます。

$ZA 値	$TEST 値	意味
4096	0	接続は構築されませんでした
12288	0	接続されましたが、データは受信しませんでした
12288	1	接続され、データを受信しました

以下のテーブルは、$ZA の各ビットの意味を示しています。

$ZA の 10 進数値	$ZA の 16 進数値	意味
2	0x2	読み取りがタイムアウトになりました
4	0x4	入出力エラーです
256	0x80	不正なエスケープ・シーケンスを受信しました
4096	0x1000	サーバ・モードです
8192	0x2000	接続されました

5.5.1.2 $ZB と READ コマンド

$ZB は、読み取りを終了した文字を保持します。文字は以下のうちの 1 つです。

- キャリッジ・リターンなどの終端文字
- 固定長 READ x#y の y 番目の文字
- READ *X の単一文字
- 読み取りがタイムアウトになったときの空文字列
- エスケープ・シーケンス

文字列が CR LF で終了した場合、CR のみが $ZB に配置されます。

5.6 TCP デバイスの WRITE コマンド

OPEN と USE で接続を構築後、WRITE コマンドは、クライアントあるいはサーバから TCP デバイスにデータを送信します。

構文は、以下のとおりです。

```
WRITE x
WRITE !
WRITE #
```

5.6.1 WRITE の動作

接続の構築後、WRITE x は、クライアントあるいはサーバからバッファに x を送信します。

WRITE ! と WRITE # は、改行と改ページを示しません。代わりに、InterSystems IRIS に、バッファに残っている文字をフラッシュし、ネットワークからターゲット・システムへ送信するよう命令します。

%SYSTEM.INetInfo クラスの TCPStats() メソッドを使用すると、現在の TCP 接続で実行する書き込みの回数を指定できます。

5.6.2 WRITE による $X および $Y の変更

InterSystems IRIS は、$X 特殊変数に、バッファ内の文字数を格納します。

ASCII 文字 <return> と <line feed> はレコードの一部と見なされないため、文字数にカウントされません。WRITE ! でバッファをフラッシュすると、$X を 0 にリセットし、$Y の値を 1 増加します。WRITE # でバッファをフラッシュすると、別のレコードとして ASCII 文字 <form feed> を書き込み、$Y を 0 にリセットします。

5.6.3 WRITE コマンド・エラー

以下のような場合、<WRITE> エラーを受け取ります。

- バッファをフラッシュせず、最大文字列サイズ (1,024 文字) を超えた場合
- 文字が存在しない書き込みバッファ (TCP/IP は長さ 0 のレコードを無視します) をフラッシュした場合
- サーバがクライアントから接続要求を受け取る前に、サーバからクライアントに WRITE コマンドを送信した場合 (InterSystems IRIS は、サーバに <WRITE> エラーを生成します)

5.6.4 WRITE コントロール・コマンド

InterSystems IRIS TCP バインディング・デバイスは、WRITE *-n 構文の一連のコントロール・コマンドをサポートします。

構文	説明
WRITE *-2	現在、クライアントに接続されているサーバ・モード・セッションで、このコマンドはセッションから切断します。新規セッションを受け入れるには、デバイスで新規に READ コマンドを実行します。
WRITE *-3	TCP 接続にバッファされた出力を送信します。つまり、出力バッファのデータに TCP SEND 操作を実行します。/GZIP を指定して圧縮したストリーム・データでは、*-3 を指定すると、圧縮エンドポイントのマークなしでデータが送信されます。$X を 0 にリセットし、$Y を 1 ずつインクリメントします。バッファする出力が存在しない場合、このコマンドは何も実行しません。
WRITE *-99	/GZIP を指定して圧縮されたストリーム・データを送信します。出力バッファにあるデータに圧縮エンドポイントのマークを付けた後、その出力バッファのデータに対して TCP SEND 操作を実行することで圧縮ストリーム・データを送信します。

5.7 接続管理

サーバは、一度に 1 つの接続のみ維持できます。したがって、あるクライアントがサーバに接続しているときに 2 番目のクライアントが接続しようとすると、TCP/IP はそのクライアントをキューに格納します。2 番目のクライアントはキューの間、あたかも接続されているかのように、ポートに書き込むことができます。2 番目のクライアントが書き込むデータは、最初の接続が閉じ、2 番目のクライアントに接続するまでバッファに維持されます。

接続が構築される前に READ を発行すると、2 番目のクライアントは停止します。2 番目の接続がキューに入っている間に、3 番目のクライアントが接続を試行しても、その接続は失敗します。

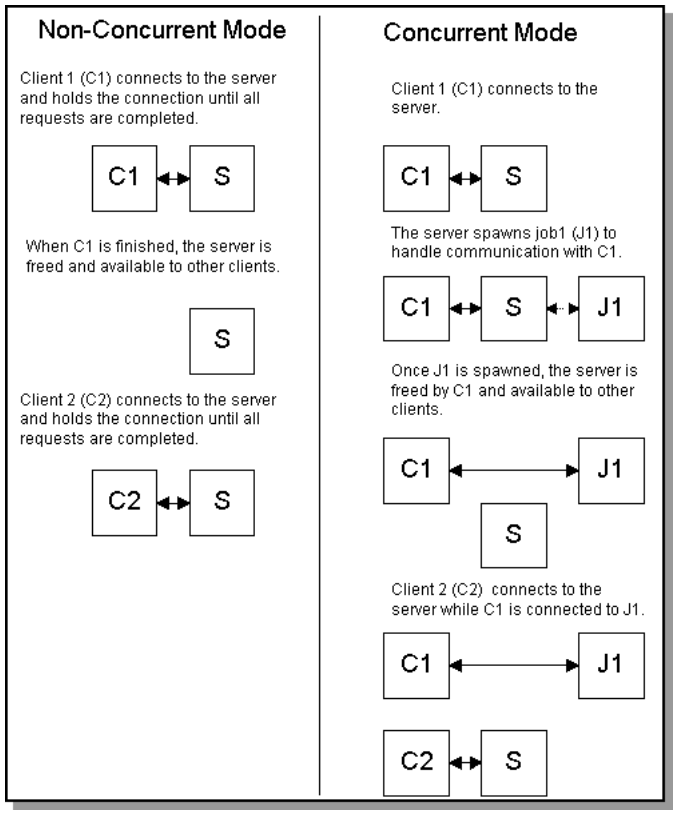
最初の接続がまだ存在しているときに、TCP デバイスを既に開いているクライアントが、2 回目の接続を試行した場合、2 回目の OPEN コマンドでは <COMMAND> エラーが発生します。この状況を USE コマンドではなくエラーとして処理することで、予想できない結果を予防します。例えば、誤ったプログラムが新規に接続を開いたと判断した場合でも、実際には異なる宛先あるいは異なるパラメータを持つ既存の接続を再利用しているたけであると、予測不可能な結果が生じる場合があります。

複数のクライアントを処理する方法は、以下を参照してください。

5.7.1 TCP デバイスを使用した JOB コマンド

JOB コマンドを使用して、TCP 並行サーバ を実装することができます。TCP 並行サーバを使用すると、複数のクライアントを同時に処理できます。このモードでは、クライアントは、サーバが他のクライアント処理を終了するまで待機する必要はありません。代わりに、クライアントがサーバに要求を送信するたびに、開いているクライアントに対して必要に応じて個別のサブジョブを生成します。このサブジョブが生成された直後に (JOB コマンドから制御が戻ります)、他のクライアントが処理を要求することがあります。サーバは、そのクライアントのサブジョブも同様に生成します。

図 5-1: 非並行モードおよび並行モードのクライアント/サーバ接続



並行サーバは、switch 並行サーバ・ビット (ビット 4 またはビット 16) を設定した JOB コマンドを使用します。お勧めの設定はビット 16 です。

- ビット 4 が設定されている場合、JOB コマンドは、生成したプロセスに、principal input および principal output のプロセス・パラメータの TCP デバイスを渡します。switch にビット 4 が含まれている場合、常に、プロセス・パラメータ principal input および principal output の両方の TCP デバイスを指定する必要があります。また、principal input および principal output の両方に同じデバイスを使用する必要があります。ビット 4 の使用はお勧めしません。詳細は、"ObjectScript リファレンス" の "JOB" コマンドを参照してください。
- ビット 16 が設定されている場合、JOB コマンドは、生成したプロセスに、TCP デバイス、principal input プロセス・パラメータ、および principal output プロセス・パラメータの 3 つの異なるデバイスを渡します。principal input プロセス・パラメータと principal output プロセス・パラメータを使用して、JOB コマンドでこれらの TCP デバイスのうち 2 つを指定します。以下の例に示すように、これらのパラメータを既定に設定することも可能です。`JOB child:(:16:input:output)` または `JOB child:(:16::))`。

詳細は、"ObjectScript リファレンス" の "JOB" コマンドを参照してください。

JOB コマンドを実行する前に、principal input および principal output に指定するデバイスは、以下の状態になる必要があります。

- 接続されている
- TCP ポートで待ち受け状態にある
- 接続を受け入れ済みである

JOB コマンドの後、生成されているプロセスのデバイスは継続して TCP ポートで待ち受け状態にありますが、アクティブな接続は既に存在していません。アプリケーションは、JOB コマンドの実行後に $ZA を確認し、TCP デバイスの状態で CONNECTED ビットがリセットされていることを確認する必要があります。

生成されたプロセスは、指定の TCP デバイスを使用して指定のエントリ・ポイントで開始します。子プロセスの TCP デバイス名は、親プロセスの名前と同じです。TCP デバイスには、付属のソケットが 1 つあります。継承された TCP デバイスは、S (ストリーム) モードです。しかし、子プロセスは、USE コマンドでモードを変更できます。サーバは、TCP デバイスを A (受信) モードで開くことをお勧めします。

生成したプロセスにある TCP デバイスは、接続状態です。これは、デバイスをクライアントから開いた後に受け取る状態と同じです。生成されたプロセスは、TCP デバイスを `USE 0` あるいは `USE $P` で使用します。TCP デバイスを暗黙的に使用することもできます (switch=4 の場合)。ただし、以下の理由により switch=16 が switch=4 よりも優先されます。

- switch=4 の場合、<READ> エラーが主デバイス上で発生すると、エラー・トラップをせず、ジョブは単に停止します。これは、switch=4 の場合、TCP デバイスが主デバイスであるためです。エラー・トラップをサポートするには、switch=16 を使用し、TCP デバイスに別のデバイスを指定します。
- switch=4 の場合、リモート TCP デバイスが接続を終了すると、エラー・トラップをせず、ジョブは単に停止します。この既定の動作をオーバーライドし、<DSCON> エラーを生成するには、**%SYSTEM.Process** クラスの DisconnectErr() メソッドを設定する必要があります。

JOB コマンドではなく **%SYSTEM.Socket** クラス・メソッドを使用して並行 TCP サーバ接続を作成できます。ただし、**%SYSTEM.Socket** メソッドはワーカ・ジョブが既に開始されていることを想定していることに注意してください。ワーカ・ジョブを開始するためにリスナ・ジョブが不要な場合、並行 TCP サーバ接続にこれらのメソッドを使用できます。リスナ・ジョブはワーカ・ジョブのプロセス ID (PID) を認識しています。

5.7.2 Job コマンドの例

以下の例は、非常に単純な並行サーバです。このサーバは、クライアントからの接続を検出するたびに子ジョブを生成します。JOB により並行サーバ・ビットの switch の値 (値 16) を指定して、シンボル・テーブル (値 1) を渡します :16+1=17。

ObjectScript

```
server
  SET io="|TCP|1"
  SET ^serverport=7001
  OPEN io:(:^serverport:"MA"):200
  IF ('$TEST) {
       WRITE !,"Cannot open server port"
       QUIT }
  ELSE { WRITE !,"Server port opened" }
loop
     USE io READ x ; Read for accept
     USE 0 WRITE !,"Accepted connection"
     JOB child:(:17:io:io) ;Concurrent server bit is on
     GOTO loop
child
  WRITE $JOB,! ; Send job id on TCP device to be read by client
  QUIT
client
  SET io="|TCP|2"
  SET host="127.0.0.1"
  OPEN io:(host:^serverport:"M"):200 ;Connect to server
   IF ('$TEST) {
       WRITE !,"cannot open connection" Quit }
   ELSE {
       WRITE !,"Client connection opened"
       USE io READ x#3:200 ;Reads from subjob
       }
   IF ('$TEST) {
```

```
      WRITE !,"No message from child"
      CLOSE io
      QUIT }
   ELSE {
      USE 0 WRITE !,"Child is on job ",x
      CLOSE io
      QUIT }
```

子ジョブは、継承した TCP 接続を使用して、ジョブ ID (この場合は 3 文字とします) をクライアントに渡します。その後、子プロセスは終了します。クライアントは、サーバとの接続を開き、その状態で子のジョブ ID を読み取ります。この例では、変数 host に IPv4 形式で指定した値“127.0.0.1”により、ローカル・ホスト・マシンへのループバック接続であることを示しています。host にサーバの IP アドレスまたは名前を設定すれば、そのサーバとは別のマシンにクライアントをセットアップすることもできます。IPv4 および IPv6 の形式に関する詳細は、″サーバ側プログラミングの入門ガイド″の“サーバ構成オプション”の章にある“IPv6 アドレスの使用”のセクションを参照してください。

原則では、子プロセスとクライアントは広範囲の通信を実行できます。複数のクライアントが同時に対応するサーバの子と通信できます。

この単純な例には、切断または失敗した読み取り処理を検出し、処理するロジックは含まれていません。

5.8 レコードの連結

特定の環境で、TCP は、別々のレコードを 1 つのレコードに連結します。クライアントあるいはサーバ・プロセスが、バッファをフラッシュするために **WRITE !** あるいは **WRITE #** コマンドで区切られた一連の WRITE コマンドを TCP ポートに発行した場合、READ コマンドが他の接続の切断を待機しているかどうかにかかわらず、レコードの連結が可能です。

最初の例は、Process B が TCP ポートに 2 つのレコードを書き込む間、READ コマンドを持つ Process A がどのように 2 つのレコードを受信するのかを説明しています。

```
Process A                              Process B
%SYS> USE "|TCP|41880" R A U 0 W A     %SYS> USE "|TCP|41880" WRITE "ONE",!,"TWO"
<RETURN>                               <RETURN>
ONE
%SYS> USE 41880 R A U 0 W A
<RETURN>
TWO
```

2 番目の例は、Process B が TCP ポートに 2 つのレコードを記述し、その後、READ コマンドを発行した Process A がどのように 1 つの連結レコードを受信するのかを説明しています。

```
Process A                              Process B
.                                      %SYS> USE "|TCP|41880" WRITE "ONE",!,"TWO"
.                                      <RETURN>
ONE
%SYS> USE "/TCP/41880" R A U 0 W A
<RETURN>
ONETWO
```

5.9 InterSystems IRIS TCP デバイスの多重化

%SYSTEM.Socket クラスは InterSystems IRIS TCP デバイスの多重化のためのメソッドを提供します。Fork() および Select() メソッドにより、新規接続の受け入れ、および接続した TCP デバイスからのデータ読み取りを両方同時に処理する単一ジョブが可能になります。待ち受け TCP デバイスが接続を受け入れた後に、Fork() を使用して、データ読み取りのための新規 TCP デバイスを作成します。元の待ち受け TCP デバイスは着信接続の受け入れを続行します。Select() メソッドを使用して、待ち受けおよび接続した TCP デバイスの両方に対して待機を行います。新規接続が着信するか、またはデータが使用可能になった場合、Select() は信号を受けたデバイス名を返します。

Select()、Publish()、Export()、および Import() メソッドを使用すれば、リスナ・ジョブの着信接続受入れとワーカ・ジョブへの接続デバイス受け渡しが可能になります。このワーカ・ジョブはリモート・クライアントとの通信が可能となっています。

詳細およびプログラム例については、"インターシステムズ・クラス・リファレンス" の %SYSTEM.Socket クラスを参照してください。

5.10 接続の切断

クライアントあるいはサーバのいずれでも、TCP バインディング接続を切断できます。接続を閉じるには、クライアントが TCP デバイスに対して CLOSE コマンドを発行することをお勧めします(または、クライアントが HALT コマンドを発行します)。サーバは、その後、そのデバイスに別の READ コマンドを発行して <READ> エラーを受け取り、さらに TCP デバイスに対して CLOSE コマンドを発行する必要があります。

この順序にするのは、TCP/IP 標準に従って、CLOSE の後 2 分間、"アクティブなクローザ" (CLOSE を最初に実行するプロセス) に対してのみ、接続リソースを維持するためです。したがって、サーバのリソースは通常クライアントのリソースよりも制約が多いため、クライアントを最初に閉じることをお勧めします。

5.10.1 CLOSE コマンドによる切断

以下の形式で、クライアントあるいはサーバから CLOSE コマンドを発行します。

```
CLOSE "|TCP|devicenum"
```

上述のように、クライアントが最初に CLOSE コマンドを発行することをお勧めします。サーバが CLOSE コマンドを発行する場合、クライアントは <WRITE> エラーを取得するので、CLOSE コマンドを発行する必要があります。

5.10.1.1 JOBSERVER リソース

管理下にないクライアントにアクセスするよう InterSystems IRIS サーバを記述している場合、サーバ・プロセスは、CLOSE を発行して TCP 接続を閉じる必要があります。CLOSE コマンドは、InterSystems IRIS に関しては接続を閉じますが、内部的に TCP/IP は、最大 2 分間サーバ上でこの接続のためのリソースを保持します。

これによって、TCP/IP ジョブへのサービスの提供のために JOBSERVER が使用される場合、予期しない結果が生じる可能性があります。JOBSERVER プロセスが停止を実行すると、プロセスは、直ちに使用可能な JOBSERVER プロセスのプールに戻りますが、そのリソースは、最大 2 分間内部的に保持されます。JOBSERVER プロセスは、最初に使用可能なものから割り当てられるので、比較的少数のクライアントからの重い負荷が JOBSERVER プロセスのリソースを使い果たす可能性があります。

この問題を回避するために、JOBSERVER の下で実行されている JOB によって開かれた TCP/IP サーバは、明示的に CLOSE を発行し、その後、最後の QUIT (または HALT) コマンドの前に短い HANG コマンドを発行する必要があります。TCP/IP の仕様に従って、JOBSERVER における各状態の間でリソースが使用中のままにならないことを保証するには HANG 120 が必要です。実際には、通常、JOBSERVER プロセス間でリソースの負荷を均等に分散するには、1 秒の HANG で十分です。

5.10.2 自動切断

以下の条件の場合、TCP バインディング接続は自動的に切断します。

- InterSystems IRIS の致命的なエラー
- クライアント・プロセスあるいはサーバ・プロセスの RESJOB
- Iris 停止

- Iris 強制

5.10.3 切断の影響

出力バッファに残っているデータへの切断の影響は、OPEN または USE 中に確立される /CLOSEFLUSH 設定によって決まります。既定では、データをフラッシュします。

片方で接続を切断し、もう一方で新規に WRITE コマンドを発行すると、最初の WRITE コマンドは成功する場合があります。続けて WRITE コマンドを発行すると、<WRITE> エラーを受け取ります。

クライアント側から、接続を切断した側へ READ コマンドを発行すると、<READ> エラーを受け取ります。サーバで通信を再構築するため、デバイスを一度閉じてから再度開く必要があります。

<READ> エラーまたは <WRITE> エラーの後で、サーバ側から発行した最初の READ コマンドは、新規接続を待機して受け入れます。

%SYSTEM.TCPDevice.GetDisconnectCode() を使用すれば、現在の TCP デバイスで <READ> または <WRITE> エラーとなった内部エラーを返すことができます。$IO は TCP デバイスとする必要があります。

6

UDP クライアント/サーバ通信

この章では、UDPを使用したプロセス間のリモート通信の設定方法について説明します。パイプ、またはインタジョブ・コミュニケーション (IJC) デバイスを使用したプロセス間のローカル通信については、このドキュメントの "ローカル・プロセス間通信" の章を参照してください。

InterSystems IRIS® Data Platform は、TCP と UDP の 2 つのインターネット・プロトコル (IP) をサポートしています。これらのインターネット・プロトコルにより、InterSystems IRIS は、プロセスが InterSystems IRIS を実行中かどうかにかかわらず、ローカルあるいはリモート・システムでプロセスと通信できます。

- TCP：InterSystems IRIS は、転送制御プロトコル (TCP) バインディングをサポートします。サーバと 1 つのクライアント間の双方向通信を確立します。エラーのチェックと修正を伴った信頼性のあるデータのバイト・ストリーム送信、およびメッセージ応答を提供します。詳細は、このドキュメントの "TCP クライアント/サーバ通信" の章を参照してください。
- UDP：InterSystems IRIS は、ユーザ・データグラム・プロトコル (UDP) バインディングをサポートします。サーバと多数のクライアント間の双方向メッセージ転送を提供します。UDP は接続ベースではありません。各データ・パケット転送は独立したイベントです。ローカル・パケット・ブロードキャストおよびリモート・マルチキャストにおいて高速で軽量なデータ転送を提供します。本質的に、TCP よりも信頼性が低くなっています。メッセージ応答を提供しません。

UDP は、**%Net.UDP** クラスを介してサポートされています。このクラスは、パケットを指定した宛先およびポートに Send() するメソッド、パケットをソケットから Recv() するメソッド、および最後に受け取ったパケットのトランスミッタへ Reply() するメソッドを提供します。

宛先は、ローカル・ホスト名あるいは IPv4 または IPv6 ホスト・アドレスとして識別されます。ポートは、指定されたポート番号または動的ポート割り当てのいずれかです。

6.1 UDP ソケットの確立

UDP を使用するには、%New() メソッドを使用して、UDP ソケット・オブジェクトを作成します。このオブジェクト・インスタンスはパケット転送の送信、受信、および応答に使用されます。

UDP ソケット・オブジェクトを作成する場合は、以下の例に示すように、ポート番号およびホスト・アドレスを指定できます。

ObjectScript

```
SET UPDOref=##class(%Net.UDP).%New(3001,"0.0.0.0")
```

ポート番号およびホスト・アドレスはどちらもオプションです。%New() メソッドは、UDP ソケット・オブジェクト・インスタンスの oref (オブジェクト参照) を返します。

UDP 転送には、以下の 2 つの側面があります。

- サーバは、要求の受信を待機し、要求された情報を提供します。したがって、転送のこの側面を、レシーバまたはプロバイダと呼ぶことができます。プロバイダが UDP オブジェクトを作成する場合には、要求を受け取るポート番号を定義する必要があります。
- クライアントは、情報の要求を送信し、応答を受け取ります。したがって、転送のこの側面を、センダまたはリクエスタと呼ぶことができます。リクエスタが UDP オブジェクトを作成すると、動的ポート番号を使用できます。既定値は 0 です。パケットを送信するときには、プロバイダのホスト名とポート番号を指定する必要があります。

6.2 ホスト・アドレス

Send() メソッドは宛先のバイナリ・アドレスを指定します。これは、ホスト・アドレスのバイナリ・バージョンです。このバイナリ・ホスト・アドレスは、以下のように GetHostAddr() メソッドを使用して作成する必要があります。

ObjectScript

```
SET client=##class(%Net.UDP).%New()
SET addrbin=##class(%Net.UDP).GetHostAddr("172.16.61.196")
WRITE client.Send("message text",addrbin,3001)
```

以下の例に示すように、GetHostAddr() にホスト名、IPv4 アドレス、または IPv6 アドレスを指定することができます。

ObjectScript

```
SET hostname="MYLAPTOP"
SET IPv4="172.16.61.196"
SET IPv6="::1"
SET flag=$SYSTEM.INetInfo.CheckAddressExist(hostname)
IF flag=1 { SET addrbin=##class(%Net.UDP).GetHostAddr(hostname)
            WRITE "host name valid",! }
   ELSE { WRITE "not a hostname: ",hostname,! }
SET flag=$SYSTEM.INetInfo.CheckAddressExist(IPv4)
IF flag=1 { SET addrbin=##class(%Net.UDP).GetHostAddr(IPv4)
            WRITE "IPv4 valid",! }
   ELSE { WRITE "not an IPv4 address: ",IPv4,! }
SET flag=$SYSTEM.INetInfo.CheckAddressExist(IPv6)
IF flag=1 { SET addrbin=##class(%Net.UDP).GetHostAddr(IPv6)
            WRITE "IPv6 valid",! }
   ELSE { WRITE "not an IPv6 address: ",IPv6,! }
```

以下の例に示すように、AddrToHostName() メソッドを使用して、ホスト名にこのバイナリ・ホスト・アドレスを拡張することができます。

ObjectScript

```
SET addrbin=##class(%Net.UDP).GetHostAddr("MYLAPTOP")
WRITE $SYSTEM.INetInfo.AddrToHostName(addrbin)
```

LocalHostName() メソッドを使用してホスト名を決定することができます。以下の例に示すように、HostNameToAddr() メソッドを使用してホスト名を IPv4 アドレスまたは IPv6 アドレスに変換することができます。

ObjectScript

```
SET localhost=$SYSTEM.INetInfo.LocalHostName()           /* get host name */
   WRITE "local host name is ",localhost,!
SET addrbin=##class(%Net.UDP).GetHostAddr(localhost)     /* compress to binary address */
   WRITE "binary form of IP address is ",addrbin,!
SET hostname=$SYSTEM.INetInfo.AddrToHostName(addrbin)    /* expand binary address to host name */
   WRITE "binary IP address expands to ",hostname,!
SET ipaddr=$SYSTEM.INetInfo.HostNameToAddr(hostname)     /* host name to IP address */
   WRITE "hostname corresponds to IP address ",ipaddr,!
```

6.2.1 IPv4 と IPv6

UDP は、IPV4 および IPV6 インターネット・プロトコルの両方をサポートします。これらのプロトコルは互換性がないため、サーバとクライアントの両方ともが同じインターネット・プロトコルを使用しない場合には転送が失敗します。

IPv4 アドレスは以下の形式です。n は、0 から 255 の範囲の 10 進数の整数です。

```
n.n.n.n
```

IPv4 プロトコルを "0.0.0.0" として指定できます。

IPv6 アドレスは以下の完全な形式です。h は、4 桁の16 進数の数値です。

```
h:h:h:h:h:h:h:h
```

通常、IPv6 アドレスは先頭ゼロを削除してゼロの連続したセクションを二重コロン (::) に置き換えることによって省略されます。IPv6 アドレスで使用される二重コロンは 1 つのみです。IPv4 省略ルールを使用することによって、IPv6 プロトコルを "::" (8 つすべての h セクションの値が 0000 であることを意味する) として指定できます。

インターネット・プロトコルを確立するには、以下の手順を実行します。

- クライアントは %New() メソッド内で IPv4 または IPv6 のいずれかを確立する必要があります。既定値は IPv4 です。
- これは、GetHostAddr() メソッドに指定されていて Send() メソッドに (バイナリ形式で) 提供されている、IPv4 または IPv6 プロトコルと一致している必要があります。

以下は IPv4 転送の例です。

ObjectScript

```
Server
 SET sobj=##class(%Net.UDP).%New(3001,"127.0.0.1")


  SET inmsg=sobj.Recv()
```

ObjectScript

```
Client
 SET cobj=##class(%Net.UDP).%New()   /* the default is IPv4 */
 SET bhost=##class(%Net.UDP).GetHostAddr("127.0.0.1")
 SET outmsg="this is the message to send"
 WRITE cobj.Send(outmsg,bhost,3001)
```

以下は IPv6 転送の例です。

ObjectScript

```
Server
 SET x=##class(%SYSTEM.INetInfo).IsIPV6Enabled()
 IF x=1 {
    SET sobj=##class(%Net.UDP).%New(3001,"::1")


  SET inmsg=sobj.Recv() }
  ELSE {WRITE "IPv6 not enabled" }
```

ObjectScript

```
Client
 SET cobj=##class(%Net.UDP).%New(0,"::")
 SET bhost=##class(%Net.UDP).GetHostAddr("::1")
 SET outmsg="this is the message to send"
 WRITE cobj.Send(outmsg,bhost,3001)
```

ホスト・アドレスを処理するためのメソッドは %SYSTEM.INetInfo クラス・ドキュメントを参照してください。IPv4 および IPv6 の形式に関する詳細は、"サーバ側プログラミングの入門ガイド" の "サーバ構成オプション" の章にある "IPv6 アドレスの使用" のセクションを参照してください。

7

シーケンシャル・ファイルの入出力

この章では、InterSystems IRIS® Data Platform でのシーケンシャル・ファイルの使用法について説明します。すべてのオペレーティング・システムで、ディスク入出力ファイルはシーケンシャル・ファイルと見なされます。Windows システムでは、(プリンタがシリアル通信ポートに接続されていない限り) プリンタをシーケンシャル・ファイル入出力デバイスとして処理します。UNIX® システムでは、プリンタはターミナル入出力デバイスと見なされます。プリンタの詳細は、このドキュメントの "プリンタ" の章を参照してください。

7.1 シーケンシャル・ファイルの使用法

このセクションでは、InterSystems IRIS によるシーケンシャル・ファイルの処理方法について説明します。具体的には、シーケンシャル・ファイル入出力の概要と、関連コマンドについて説明します。

- シーケンシャル・ファイルにアクセスするには、最初に OPEN コマンドを使用し、引数としてファイル名を指定して、ファイルを開く必要があります。また、必要に応じて OPEN の各種モード・パラメータを指定します。モード・パラメータは、OPEN で指定されたファイルが存在しない場合に、新しいファイルを作成するかどうかを指定します。同時に複数のファイルを開くことができます。
- シーケンシャル・ファイルを開いた後、USE コマンドを使用し、そのファイル名を引数として指定して、ファイルにアクセスする必要があります。USE コマンドによって、指定されたファイルが現在のデバイスになります。したがって、一度に使用できるのは 1 つのファイルのみです。USE コマンドでは、モード・パラメータも指定できます。
- その後、そのファイルに対して複数の READ コマンドまたは WRITE コマンドを発行できます。各 READ コマンドはファイルから 1 レコードを受け取り、各 WRITE コマンドはファイルに 1 レコードを引き渡します。"W" モード・パラメータを指定してファイルを開かないと、ファイルに書き込むことはできません。ファイルの範囲を超えて読み取ろうとすると、<ENDOFFILE> エラーを生じます。
- $ZSEEK 関数を使用すると、シーケンシャル・ファイルの先頭、現在位置、または末尾からの文字数オフセットによって指定されるファイル位置を設定できます。$ZPOS 特殊変数には、現在のシーケンシャル・ファイルの先頭からの現在の文字数位置が含まれます。
- ファイル入出力を完了したら、CLOSE コマンドを発行し、シーケンシャル・ファイルを閉じます。

これらの操作は、**%Library.File** クラスのメソッドを使用して実行することもできます。

%Library.File.Exists() メソッドは、指定された名前のシーケンシャル・ファイルが既に存在するかどうかを示します。

%Library.File.Size プロパティは、シーケンシャル・ファイル内の現在の文字数を返します。

%Library.File.DateModified プロパティは、ファイルが開いたときおよび閉じたとき (ファイルが変更された場合) に現在のローカル日付とローカル時刻で更新されます。

%Library.File.IsOpen プロパティは、ファイルが %Library.File.Open() メソッドによって開かれた場合にのみ 1 を返します。OPEN コマンドでは、このブーリアン・プロパティは設定されません。

7.1.1 ファイルの指定

シーケンシャル・ファイルは、キャノニック形式のパス名 (完全パス名)、またはシステムによって完全パス名に展開される相対パス名 (パス名の一部) で指定できます。パス名は、キャノニック形式にすることも (c:¥InterSystems¥IRIS¥mgr¥user¥myfiles¥testfile.txt)、現在のディレクトリを基準にすることもできます (testfile.txt)。先頭のピリオドが 1 つ (.) の場合、現在のディレクトリを示します。先頭のピリオドが 2 つ (..) の場合、現在のディレクトリの親を示します。OPEN コマンドで新しいファイルを作成する場合は、指定したディレクトリが既に存在している必要があります。

「 」など、オペレーティング・システムによって返されるファイル・アクセス・エラーは %SYSTEM.Process.OSError() メソッドによって返されます。このメソッドは、山括弧で囲まれたオペレーティング・システムのエラー番号に続けて、エラー・テキストを返します。これを以下の Windows の例に示します。

```
USER>OPEN "C:\InterSystems\IRIS\mgr\nodir\testfile.txt":("WNS"):5

USER>w $SYSTEM.Process.OSError()
<3> The system cannot find the path specified.

USER>w ##class(%File).TempFilename("txt","C:\InterSystems\IRIS\mgr\nodir\testfile",.oserrnum)

USER>w $SYSTEM.Process.OSError()
<3> The system cannot find the path specified.
```

以下の Windows の例ではすべて、現在のネームスペース (USER) ディレクトリにファイルが作成されます。

- 完全パス名 : `OPEN "C:\InterSystems\IRIS\mgr\user\testfile1.txt":("WNS"):10`
- ファイル名の展開 : `OPEN "testfile2.txt":("WNS"):10`
- 現在のディレクトリの展開 : `OPEN ".\testfile3.txt":("WNS"):10`

以下の Windows の例では、現在のネームスペース (USER) ディレクトリの既存の子ディレクトリにファイルが作成されます。

- 現在のディレクトリの子 : `OPEN "mytemp\testfile4.txt":("WNS"):10`

以下の Windows の例では、親ディレクトリ (..) 構文を使用してファイルが作成されます。

- 親ディレクトリ (`C:\InterSystems\IRIS\mgr\`) : `OPEN "..\testfile5.txt":("WNS"):10`
- 現在のディレクトリ (親ディレクトリの子) `C:\InterSystems\IRIS\mgr\user\` : `OPEN "..\user\testfile6.txt":("WNS"):10.`
- 親ディレクトリの別の子 `C:\InterSystems\IRIS\mgr\temp\` : `OPEN "..\temp\testfile7.txt":("WNS"):10.`
- 親ディレクトリの親 `C:\InterSystems\IRIS\` : `OPEN "..\..\testfile8.txt":("WNS"):10.`

Windows のパス名では、ディレクトリの区切り文字に ¥ (円記号) が使用されます。UNIX のパス名では、ディレクトリの区切り文字に / (スラッシュ) が使用されます。有効な文字は、8 ビット ASCII または ISO Latin-1 Unicode です。

Windows のファイル・パス名は以下の形式で指定します。

```
device:\directory\file.type
```

例えば、`C:\InterSystems\IRIS\mgr\user\myfiles\testfile.txt` のように指定します。type 接尾語はオプションです。

UNIX® のファイル・パス名は以下の形式で指定します。

```
../directory/name
```

ファイル・パス名は、完全に展開されたときに 256 文字を超えてはいけません。すべてのディレクトリのパス名の長さが 256 文字を超えると、<DIRECTORY> エラーが生成されます。ファイル名の長さが原因でパス名の長さが 256 文字を超えた場合は、<NAMEADD> エラーが生成されます。

UNIX® のファイル・パス名には、あらゆる種類の文字を 255 個まで使用できます。ファイル名に含まれるピリオド (“.”) とアンダースコア (“_”) はどの位置でも使用できますが、通常、その名前を意味的に区切るために使用します。例えば、ファイル・タイプとして .dat を使用して、ファイル名 pat_rec.dat を定義できます。

現在の UNIX® 既定ディレクトリのファイルにアクセスする場合、通常はその名前を指定するだけで済みます。システムは、ディレクトリに既定値を埋め込みます。

DLL 名は、完全パス名として指定することも、パス名の一部を指定することもできます。パス名の一部を指定した場合、InterSystems IRIS によって現在のディレクトリに展開されます。一般的に、DLL はバイナリ・ディレクトリ (″bin″) に格納されます。バイナリ・ディレクトリの位置を確認するには、%SYSTEM.Util クラスの BinaryDirectory() メソッドを呼び出します。

7.1.1.1 ファイル・パス名ツール

現在のデバイスがシーケンシャル・ファイルである場合、$ZIO には、そのファイルの完全パス名が含まれます。

$ZSEARCH を使用して、指定したファイルまたはディレクトリの完全なファイル指定 (パス名とファイル名) を返すことができます。ファイル名にワイルドカードを含めると、$ZSEARCH は、ワイルドカードを満たす一連の完全修飾パス名を返します。

%Library.File クラスには、ファイル・システム・サービスを提供するさまざまなメソッドが用意されています。これには、以下のものがあります。

- NormalizeDirectory()。指定されたファイルまたはディレクトリの完全パス名を返します。
- NormalizeFilenameWithSpaces()。パス名に含まれているスペースをホスト・プラットフォームに合わせて適切に処理します。パス名にスペース文字が含まれている場合、パス名の処理はプラットフォームによって異なります。Windows および UNIX® の場合はパス名にスペース文字を使用することができますが、スペースを含むパス名は、追加の二重引用符 () で全体を囲む必要があります。これは、Windows の cmd /c 文に従っています。詳細は、Windows のコマンド・プロンプトで cmd /? を指定してください。

7.1.1.2 チルダ (~) 展開

Windows のパス名では、チルダ (~) は、長い名前が 8.3 形式で圧縮されていることを示します。例えば、c:¥PROGRA~1¥ のように表示されます。圧縮されたディレクトリ名を変換するには、%Library.File クラスの NormalizeDirectory() メソッドを使用します。

UNIX® のパス名では、チルダ (~) 展開を使用して、現在のユーザのホーム・ディレクトリまたは指定したユーザのホーム・ディレクトリを示すことができます。

- ~ と ~/myfile.txt は、現在のユーザのホーム・ディレクトリに展開されます (それぞれ /Users/techwriter/ と /Users/techwriter/myfile.txt)。
- ~guest/myfile.txt は、ユーザ “guest” のホーム・ディレクトリに展開されます (/Users/guest/myfile.txt)。ただし、ユーザ “guest” が存在しない場合は、現在のユーザの完全なディレクトリ・パス名に展開され、~guest/myfile.txt がリテラルとして追加されます (/Users/techwriter/iris/mgr/user/~guest/myfile.txt)。
- ~myfile.txt と ~123.txt は、現在のユーザの完全なディレクトリ・パス名にリテラルとして追加されます (それぞれ /Users/techwriter/iris/mgr/user/~myfile.txt と /Users/techwriter/iris/mgr/user/~123.txt)。

7.1.2 OPEN コマンド

OPEN は、シーケンシャル・ファイルを開きます。OPEN コマンドを使用して、InterSystems IRIS データベース・ファイルを開くことはできません。

OPEN コマンドで開いたシーケンシャル・ファイルを、別のプロセスでも開くことができます。OPEN コマンドの “L” モード・パラメータまたは ObjectScript の LOCK コマンド、またはその両方を使用してシーケンシャル・ファイルへの同時アクセスを制御できます。ファイル・ロックのサポートは、基礎となるオペレーティング・システムのファイル・アクセス規則によって提供されます。

ObjectScript の OPEN コマンドで開くファイルとデータベース・ファイルとの間で、プロセスごとに開くことができるファイルの数を InterSystems IRIS で割り当てます。OPEN コマンドで開くファイルが多すぎて OPEN コマンドへの割り当てができない場合、<TOOMANYFILES> エラーを生じます。InterSystems IRIS ではプロセスごとに開くことができるファイルの最大数は 1,024 です。プロセスごとに開くことができるファイルの実際の最大数は、プラットフォーム固有の設定です。例えば、Windows の既定では、プロセスごとに開くことができるファイルの最大数は 998 です。詳細は、オペレーティング・システムのマニュアルを参照してください。

7.1.2.1 OPEN 構文

```
OPEN filename{{:({parameters{:reclength{:terminators}}})}{:timeout}}
```

各項目の内容は次のとおりです。

引数	説明
filename	引用符で囲まれた有効なファイル指定です。このファイル・パス名は 255 文字以内にする必要があります。有効な文字は、8-bit ASCII または ISO Latin-1 Unicode です。UNIX のパス名には、チルダ (~) 展開を使用して、現在のユーザのホーム・ディレクトリを示すことができます。例 : ~myfile や ~/myfile。
parameters	オプション – 引用符で囲まれた 1 文字コードの文字列です。実行できるファイル形式と処理タイプを定義します (スラッシュ (/) で始まるキーワードを使用して parameters を指定することもできます)。コード定義については、テーブル “OPEN モード・パラメータ” を参照してください。パラメータに R と W のいずれも指定しない場合は、R が既定値です。このシステム全体の既定のオープン・モードは、Config.Miscellaneous クラスの OpenMode プロパティを設定して構成できます。新規ファイルを開くには、N パラメータを指定する必要があります。指定しない場合、OPEN は、停止あるいはタイムアウトにより失敗します。パラメータに S、V、F、および U のいずれも指定しない場合、新しい Windows ファイルまたは UNIX® ファイルに対する既定値は S です。既存ファイルの既定値は、ファイルの作成時に指定されたモードです。A が指定されていない場合、WRITE 処理によって、以前のファイルの内容が上書きされます。パラメータは、左から右の順に適用されます。
reclen	オプション – Windows システムおよび UNIX システムでは、(S) レコードおよび (U) レコードの場合は最大長、固定長 (F) レコードの場合は絶対レコード長を指定します。可変長 (V) レコードの場合は無視されます。既定値は 32767 です。
terminators	オプション – ストリーム・モードにのみ対応するユーザ定義のレコード・ターミネータ文字列です。既定のターミネータのキャリッジ・リターン、改行、改ページをオーバーライドします。ユーザ定義のターミネータは入力にのみ適用され、ファイルへのデータの記述方法には影響を与えません (ターミネータは特殊文字としてファイルに書き込まれます)。ユーザ定義のターミネータが複数ある場合は、単一のターミネータとして使用される複数文字シーケンスではなく、ターミネータのリストとして処理されます。
timeout	オプション – InterSystems IRIS がファイルを開こうと試みる際の秒数です。この間にファイルを開くことができない場合、$TEST を 0 に設定し、制御をプロセスに戻します。成功した場合、$TEST が 1 に設定されます。

timeout 引数は、オプションですが、強くお勧めします。これは、OPEN の成功または失敗が、$TEST 特殊変数の値によって示され、$TEST は、timeout を指定した場合のみ設定されるためです。$TEST は、時間内にオープンが成功すると 1 に設定されます。時間が切れると、$TEST は 0 に設定されます。

7.1.2.2 OPEN モード・パラメータ

OPEN モード・パラメータは次の 2 つの方法で指定できます。

- “VRWN” など、引用符で囲まれた文字コード文字列。これらの文字 1 つで 1 つのパラメータを指定します。文字コードは、任意の順序で指定できます。InterSystems IRIS では、左から右の順に実行されるので、文字コード間の相互関係により優先の順序が決まる場合もあります。
- 一連の /keyword パラメータ。引用符では囲みません。パラメータどうしは、コロンで区切って記述します。キーワード・パラメータは、任意の順序で指定できます。InterSystems IRIS では、パラメータが左から右の順に実行されるので、パラメータ間の相互関係により優先の順序が決まる場合もあります。

文字コード・パラメータとキーワード・パラメータを組み合わせて指定する場合、文字コード文字列を先に指定し、続けてキーワード・パラメータをコロンで区切って指定します。以下の例では、3 つの文字コード・パラメータの後に、2 つのキーワード・パラメータと reclen 引数および timeout 引数を指定しています。

ObjectScript

```
OPEN "mytest":("WNS":/OBUFSIZE=65536:/GZIP=0:32767):10
```

テーブル 7-1: OPEN モード・パラメータ

文字コード	キーワード	説明
N	/NEW	新しいファイル。指定されたファイルが存在しない場合は、ファイルが作成されます。指定されたファイルが読み取り専用ファイルとして既に存在する場合は、古いファイルが削除され、同じ名前の新しいファイルに置き換えられます (許可されている場合)。このパラメータを使用する並行プロセスが同じファイルを上書きすることを防止するには、ファイル・ロックを使用する必要があります。 “N” モード (または “T” モード) の指定がなく、OPEN で指定されたファイルが存在しない場合、Windows および UNIX® では、既定で新しいファイルは作成されません。この動作は、%SYSTEM.Process クラスの FileMode() メソッドを使用して構成できます。システム全体の既定の動作は、Config.Miscellaneous クラスの FileMode プロパティで設定できます。
E	/CREATE または /CRE	ファイルが存在しない場合は、ファイルを作成します。“N” モードのように、既存ファイルを削除し再作成しません。既定では、新しいファイルは作成されません。この既定の動作は、%SYSTEM.Process クラスの FileMode() メソッド、または Config.Miscellaneous クラスの FileMode プロパティが有効になっている場合にオーバーライドされます。
T	/TRUNCATE	ファイルの切り捨て : 指定されたファイルが存在し、書き込み可能な場合、そのファイルは切り捨てられ、その属性は変更されません。指定されたファイルが存在しない場合、“N” モードが指定されているかのように、新しいファイルが作成されます。“WT” と “WNT” は機能的に同一です。
D	/DELETE[=n] または /DEL[=n]	ファイルの削除 : ファイルを閉じたときに、そのファイルを自動的に削除することを指定します。/DELETE を指定した場合、または /DELETE=n の n が 0 以外の場合、このパラメータ・コードが有効になります。/DELETE=n の n が 0 の場合、パラメータ・コードが無効になります。既定では、ファイルは削除されません。
R	/READ	読み取り : READ でファイルにアクセスできます。他のプロセスもこのファイルにアクセスできます (ただし、“L” パラメータを参照のこと)。“R” モードで存在しないファイルを開こうとすると、プロセスは停止します。これを防ぐには、タイムアウトを使用します。“R” は、すべてのプラットフォームの既定値です。このシステム全体の既定のオープン・モードは、Config.Miscellaneous クラスの OpenMode プロパティを設定して構成できます。

文字コード	キーワード	説明
W	/WRITE または /WRI	書き込み：WRITE でファイルにアクセスできます。Windows および UNIX® では、“W” によって、プロセスにレコードへの排他書き込みアクセスと共に、ファイルへの共有書き込みアクセスが指定されます。ファイルへの排他書き込みアクセスを指定するには “WL” を使用します。“W” モードで存在しないファイルを開こうとすると、ファイルが作成されるまで、あるいはプロセスがタイムアウト、Process Terminate、または RESJOB で解決されるまで、プロセスは停止します。“R” は、すべてのプラットフォームの既定値です。このシステム全体の既定のオープン・モードは、Config.Miscellaneous クラスの OpenMode プロパティを設定して構成できます。/OBUFSIZE と共に使用できます。
L		排他ロック：ファイルへの排他書き込みアクセスを指定するには、“L” モードを “W”（書き込み）モードと共に使用します。“WL” または “WRL” は、現在のプロセスがファイルへの排他書き込みアクセスを持つことを指定します。“RL” で開いたファイルに対しては、そのまま共有読み取りアクセスが維持されます。ファイルを同時に開く操作に対する “L” モードの効果は、Windows と UNIX® とでは異なります。詳細は、以下の “OPEN モード・ロック“ セクションを参照してください。UNIX® システムでは、1 つのプロセスでファイルへの “WL”（または “WRL”）アクセスが指定されている場合、そのファイルへの読み取りアクセスを要求する他のプロセスは、“RL” を指定する必要があります。これは、UNIX® がファイル・ロックを調整できるようにするためです。
A	/APPEND または /APP	追加：WRITE 処理によって、既存のファイルの最後にデータが追加されます。既定では、追加されるのではなく、既存のデータが上書きされます。
S	/STREAM	既定ターミネータとしてキャリッジ・リターン、改行、または改ページを使用するストリーム形式です。S、V、F、および U の各モードは、相互排他的です。ストリーム・レコード形式が既定です。
V	/VARIABLE	可変長：各 WRITE によって 1 つのレコードが作成されます。Windows および UNIX® の場合、可変長レコードの長さは任意です。reclen 引数は無視されます。 可変長シーケンシャル・ファイルの末尾以外のどの場所にもレコードを挿入しないでください。WRITE では、WRITE コマンドが実行された時点以降のファイル内のデータがすべてアクセスできなくなっています。S、V、F、および U の各モードは、相互排他的です。ストリーム・レコード（S）形式が既定です。 UTF8 変換を使用した Unicode データなど、変換テーブルを使用して記述された可変長レコードは、入力データとは異なる文字列長の格納レコードになる場合があります。このレコードを読み取る際に、InterSystems IRIS では元の入力文字列長を使用します。

文字コード	キーワード	説明

F	/FIXED
	または
	/FIX
		固定長：それぞれのレコードは、reclen 引数で指定された長さです。例えば、
		
		ObjectScript
		
		`OPEN "myfile":("RF":4)`
		`USE "myfile":0`
		`READ x:5`
		
		この例では、最初の 4 文字のレコードを変数 x に読み取ります。READ 処理でのみ使用します (WRITE 処理では使用しません)。S、V、F、および U の各モードは、相互排他的です。

U	/UNDEFINED	未定義長：ファイル・レコードの長さを未定義に指定します。したがって、READ 処理では、読み取る文字数を指定する必要があります。最大レコード長は、reclen 引数で指定されます。出力時に変換しません。既定値は、最大文字列長です。S、V、F、および U の各モードは、相互排他的です。
K¥name¥ または Knum	/TRANSLATE[=n]: /IOTABLE[=name] または /TRA[=n]: /IOT[=name]	入出力変換モード：“K” パラメータをデバイスに使用すると、システム全体で変換が有効になっていれば、そのデバイスで入出力変換が発生します。テーブル名を指定することで、変換の基本となっている既存の定義済みテーブルを特定します。キーワードを使用するときは、/TRANSLATE を指定して入出力変換を有効にし (有効にするには n=1、無効にするには n=0)、/IOTABLE=name を使用して、使用する変換テーブルを指定します。使用可能な変換テーブルのリストは、$ZCONVERT 関数のドキュメントの、“3 つのパラメータ形式：エンコード変換” を参照してください。プロトコルのオンとオフを切り替える + オプションと - オプションは、K プロトコルでは使用できません (テーブルをロードするスロット数の “num” を示す従来の Knum は廃止されましたが、現在もサポートされています。システム管理者は、テーブル・タイプごとの選択ウィンドウにある % NLS ユーティリティにスロット番号を表示できます)。このパラメータは、OPEN コマンドと USE コマンドのいずれでも使用できます。
Y¥name¥ または Ynum	/XYTABLE[=name] あるいは以下のようになります。 /XYT[=name]	$X/$Y アクション・モード：デバイスに対して “Y” パラメータを使用すると、$X/$Y というアクション・テーブルが使用されます。テーブル名を指定することで、変換の基本となっている既存の定義済み $X/$Y アクション・テーブルを識別します。$X/$Y アクションは常に使用できます。“Y” が指定されず、システムで既定の $X/$Y が定義されていない場合、組み込みの $X/$Y アクション・テーブルが使用されます。プロトコルのオンとオフを切り替える + オプションと - オプションは、Y プロトコルでは使用できません (テーブルをロードするスロット数の “num” を示す従来の Ynum は廃止されましたが、現在もサポートされています。システム管理者は、テーブル・タイプごとの選択ウィンドウにある NLS ユーティリティにスロット番号を表示できます)。このパラメータは、OPEN コマンドと USE コマンドのいずれでも使用できます。

文字コード	キーワード	説明
	/NOXY [=n]	$X および $Y の処理なし : /NOXY を指定した場合、または /NOXY=n (n は 0 以外の値) を指定した場合、$X および $Y の処理が無効になります。これにより READ 操作および WRITE 操作のパフォーマンスを大幅に向上させることができます。$X および $Y の変数値が不確定であるため、マージン処理 ($X に依存) は無効になります。/NOXY=0 の場合は、$X および $Y の処理が有効になります。これが既定です。このパラメータは、OPEN コマンドと USE コマンドのいずれでも使用できます。
	/OBUFSIZE=int	出力バッファリング : WRITE 用の出力バッファを作成します。int 変数は、バッファ・サイズをバイト単位で指定する整数です。ファイルを書き込み専用 (“W”) でオープンする場合にのみ使用できます (“R” または “RW” の場合は不可)。特に WAN 経由で少量の書き込みを頻繁に実行する場合に、パフォーマンスを大幅に向上させることができます。ただし、システム・クラッシュが発生した場合、バッファ内のデータは失われます。CLOSE、WRITE *-1、または WRITE *-3 が実行されると、バッファ内のデータがディスクにフラッシュされます。
	/GZIP[=n]	GZIP 圧縮 : GZIP と互換性のあるストリーム・データ圧縮を指定します。/GZIP を指定した場合、または /GZIP=n (n は 0 以外) を指定した場合、WRITE の発行時に圧縮、READ の発行時に解凍が有効になります。/GZIP=0 を指定した場合は、圧縮と解凍が無効になります。/GZIP=0 を発行して、圧縮/解凍を無効にする前に、$ZEOS 特殊変数をチェックして、ストリーム・データの読み込みが実行中でないことを確認してください。/GZIP 圧縮は、/IOTABLE を使用して構築した変換などの入出力変換には影響しません。これは、圧縮がその他すべての変換 (暗号化を除く) の後に適用され、解凍がその他すべての変換 (暗号化を除く) の前に適用されるためです。
	/COMPRESS=str	ストリーム・データの圧縮タイプを以下のいずれかの値で指定します。 · "zlib" — zlib 圧縮ライブラリを使用します。/COMPRESS="zlib" は /GZIP=1 と同等です。 · "zstd" — Zstandard 圧縮アルゴリズムを使用します。 · "lz4" — LZ4 圧縮アルゴリズムを使用します。 · "deflate" — DEFLATE 圧縮アルゴリズムを使用します。 圧縮を無効にするには、/COMPRESS="" を指定します。文字列を圧縮するには、%SYSTEM.Util.Compress() を使用します。

7.1.2.3 OPEN 引数キーワード

次のテーブルは、シーケンシャル・ファイルの OPEN コマンド引数キーワードを示しています。

テーブル 7-2: シーケンシャル・ファイルの OPEN キーワード引数

キーワード	既定値	説明
/PARAMS=str または /PAR=str	既定なし	parameters 位置パラメータに相当します (位置に依存しない方法でパラメータ文字コード文字列を指定する方法を提供します)。

キーワード	既定値	説明
/RECORDSIZE=int または /REC=int	既定なし	reclen 位置パラメータに相当します。固定長レコードのレコード・サイズを設定します (現在 READ 処理のみに実装されます)。
/TERMINATOR=str または /TER=str	既定なし	ユーザ定義のターミネータを構築する terminators 位置パラメータに相当します。str は、ストリーム・モードにのみ適用されるユーザ定義のレコード・ターミネータ文字列です。既定のターミネータのキャリッジ・リターン、改行、および改ページをオーバーライドします。ユーザ定義のターミネータは入力にのみ適用され、ファイルへのデータの記述方法には影響を与えません (ターミネータは特殊文字としてファイルに書き込まれます)。ユーザ定義のターミネータが複数ある場合は、単一のターミネータとして使用される複数文字シーケンスではなく、ターミネータのリストとして処理されます。

7.1.2.4 OPEN モード・ロック

2 つのプロセスが同じシーケンシャル・ファイルを開こうとした場合、2 番目の OPEN が成功するか失敗するかは、最初の OPEN が使用するモードによって決まります。以下のテーブルは、排他 (“L”) と非排他の読み取りモードと書き込みモードを使用してファイルを開こうとする 2 つのプロセス間の相互作用を示しています。これらの相互作用の解釈はプラットフォームによって異なることに注意してください。Windows オペレーティング・システムと UNIX® オペレーティング・システムについてテーブルを示します。

以下のテーブルでは、横軸が、最初の OPEN でファイルを開くモードを示し、縦軸が 2 番目の OPEN でファイルを開くモードを示しています。A 1 は 2 番目の OPEN が成功することを示し、0 は 2 番目の OPEN が失敗することを示します。

テーブル 7-3: Windows の OPEN モードでの相互作用

	W	RW	RL	WL	RWL	R
W	1	1	1	0	0	1
RW	1	1	1	0	0	1
RL	1	1	1	0	0	1
WL	0	0	0	0	0	0
RWL	0	0	0	0	0	0
R	1	1	1	0	0	1

Windows システムの場合、このテーブルの相互作用は、同じ InterSystems IRIS インスタンスからの同時オープン、2 つの異なる InterSystems IRIS インスタンスからの同時オープン、InterSystems IRIS および非 InterSystems IRIS アプリケーションによる同時オープン (非 InterSystems IRIS アプリケーションでは以下に示す制限があります) に同等に適用されます。

テーブル 7-4: UNIX® の OPEN モードでの相互作用

	W	RW	RL	WL	RWL	R
W	1	1	1	1	1	1
RW	1	1	1	1	1	1
RL	1	1	1	0	0	1
WL	1	1	0	0	0	1
RWL	1	1	0	0	0	1
R	1	1	1	1	1	1

UNIX® システムの場合、このテーブルに示す相互作用は、同じ InterSystems IRIS インスタンスから同時にファイルを開く操作にのみ該当します。2 つの異なる InterSystems IRIS インスタンスから同時に開く操作や、InterSystems IRIS および非 InterSystems IRIS アプリケーションで同時に開く操作には適用されません。

非 InterSystems IRIS ソフトウェアでの相互作用

Windows システムでは、InterSystems IRIS で“WL”書き込みアクセスしてシーケンシャル・ファイルを開くと、通常、非 InterSystems IRIS アプリケーションで書き込みアクセスしても、同じシーケンシャル・ファイルを開くことはできなくなります。同様に、シーケンシャル・ファイルを書き込みアクセスで開いている非 InterSystems IRIS アプリケーションが存在すると、通常、InterSystems IRIS プロセスからは同時“WL”書き込みアクセスはできなくなります。

ただし、メモ帳やワードパッドなど特定の非 InterSystems IRIS アプリケーションは、ファイルを開き、共有モードでファイルのコピーを作成した後、直ちにファイルを閉じます。そのため、InterSystems IRIS プロセスは、“WL”モードでファイルを開くことができます。これらの非 InterSystems IRIS アプリケーションで、コピーしたファイルを変更し、それを元のファイルに保存しようとした場合や、元のファイルを再び開こうとした場合はエラーが発生します。さらに重大な状況が発生することもあります。例えば、これらの非 InterSystems IRIS アプリケーションの 1 つでファイルを開き、InterSystems IRIS でもそのファイルを開いて変更し、閉じたとします。ここで非 InterSystems IRIS アプリケーションでそのファイルの変更を保存すると、両方のプロセスで発生した変更が保存されるので、ファイル・データの整合性が損なわれる可能性があります。

UNIX® システムでは、“WL”書き込みアクセスのために InterSystems IRIS でシーケンシャル・ファイルを開いても、InterSystems IRIS 以外のアプリケーションの動作には何も影響ありません。非 InterSystems IRIS アプリケーションからの書き込みアクセスを確実に制限するには、ロックを使用する必要があります。

7.1.2.5 例

以下の例では、現在のディレクトリのファイル“LUDWIG.B”を開きます。モード・パラメータが指定されていないので、ファイルは以下のように既定の読み取りアクセスのストリーム・モードで開きます。

ObjectScript

```
OPEN "LUDWIG.B"
```

以下の例では、現在のディレクトリにある新規ファイル“LIST.FILE”を書き込みアクセスのストリーム形式で開きます。通常、括弧で囲む引数のうち、最初の引数のみを指定する場合、括弧は必要ありません。

ObjectScript

```
OPEN "LIST.FILE":"WNS"
```

以下の例は、読み取りアクセスおよび書き込みアクセスを使用し、現在のディレクトリ内のファイル“CARDS”を 80 文字の固定レコード長で開きます。

ObjectScript

```
OPEN "CARDS":("FRW":80)
```

以下の例は、ディレクトリ c:¥usr¥dir のストリーム形式ファイル“STRNG”を、既定のターミネータは使用せずに開きます。

ObjectScript

```
OPEN "c:\usr\dir\STRNG":("S"::$CHAR(0)_$CHAR(255))
```

7.1.3 USE コマンド

USE コマンドは、開いたシーケンシャル・ファイルを現在のデバイスにします。多数のシーケンシャル・ファイルを開くことができますが、一度に使用できるシーケンシャル・ファイルは 1 つのみです。

7.1.3.1 構文

```
USE file:position
```

各項目の内容は次のとおりです。

テーブル 7-5: USE コマンド・パラメータ

引数	説明
file	引用符で囲まれた有効なファイル指定です。指定するファイルは、既に開いている必要があります。UNIX のパス名には、チルダ (~) 展開を使用して、現在のユーザのホーム・ディレクトリを示すことができます。例 : ~myfile や ~/myfile。
position	オプション – ファイル内の次の READ あるいは WRITE の位置です。position の値は数式で記述され、意味はファイルのレコード形式に依存します。固定長の場合、position はゼロを基準とした絶対レコード番号で、各レコードは事前に OPEN コマンドで指定された文字数で構成されます。ストリームあるいは可変長レコードの場合、position はゼロを基準とした絶対バイト位置です。既定では、ファイルの先頭から順番にレコードが読み取りまたは書き込まれます。 $ZSEEK 関数を使用すると、シーケンシャル・ファイルの先頭、現在位置、または末尾からの文字数オフセットによって指定されるファイル位置を設定できます。$ZPOS 特殊変数には、シーケンシャル・ファイルの先頭からの現在の文字数位置が含まれます。

7.1.3.2 USE のみのコマンド・キーワード

上述のような OPEN で共有するコマンド・キーワードの他に、USE コマンドには独自のキーワード・セットがあります。

テーブル 7-6: シーケンシャル・ファイルで使用する USE のみのコマンド・キーワード

キーワード	既定値	説明
/POSITION=n	現在のファイル位置です (ファイルを付加モードで開かない限り、ファイルを最初に開くと、ポインタはファイルの先頭にあります。付加モードでは、ポインタがファイルの末尾にあります)。	位置パラメータに相当します。ファイル内で次の READ あるいは WRITE の位置を設定します。

7.1.4 READ コマンドと WRITE コマンド

READ コマンドまたは WRITE コマンドの後には、位置パラメータを持つ次の USE コマンドが発行されるまで、READ 処理または WRITE 処理を続けることができます。

7.1.4.1 READ コマンド

READ コマンドは、現在のデバイスから一度に 1 レコード読み取ります。ファイルの範囲を超えて読み取ると、<ENDOFFILE> エラーを生じます。

構文

```
READ x#n:timeout
```

各項目の内容は次のとおりです。

引数	説明
x	ファイルから読み取ったレコードを保持する変数です。
n	オプション — 可変長読み取りの場合、読み取る文字数です。整数で指定します。固定長読み取りの場合、この引数は無視されます。
timeout	オプション — 読み取り処理の完了を待機するタイムアウトまでの秒数。整数値、または整数値に変換される変数のいずれかです。

timeout 引数は、オプションですが、強くお勧めします。これは、timeout が指定されている場合、READ の成功または失敗が、$TEST 特殊変数の値によって示されるためです。$TEST は、時間内に読み取りが成功すると 1 に設定されます。時間が切れると、$TEST は 0 に設定されます。

次の例は、Windows シーケンシャル・ファイルから固定長レコードを読み取る READ 処理を示しています。シーケンシャル・ファイルを作成し、それにデータを書き込み、ファイルを閉じます。次に、このファイルを 4 文字の固定長読み取り用に開きます (`"RF":4`)。USE 位置引数を、最初のレコード (レコード 0) に設定します。読み取り処理が実行されるたびに、この位置が進みます。FOR ループによって、4 文字のレコードごとに、添え字付き変数に読み込まれます。その後、ZWRITE コマンドが、これらの添え字付きローカル変数とそれらの値をすべて表示します。

ObjectScript

```
SET myf="C:\InterSystems\IRIS\mgr\temp\myfixedlengthfile"
OPEN myf:("NW") USE myf WRITE "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
CLOSE myf
OPEN myf:("RF":4) USE myf:0 FOR i=1:1:7 {READ x(i):5}
CLOSE myf
ZWRITE
```

7.1.4.2 WRITE コマンド

WRITE コマンドは、現在のデバイスのシーケンシャル・ファイルに、一度に 1 レコードずつ書き込みます。

構文

```
WRITE x
```

各項目の内容は次のとおりです。

引数	説明
x	変数 x のデータは、シーケンシャル・ファイルに 1 レコードとして書き込まれます。

7.1.4.3 例

以下の例は、固定長ファイルの 3、4、5 番目のレコードを読み取ります。

ObjectScript

```
SET myfile="FIXED.LEN"
OPEN myfile:("FR":100)
USE myfile:2
READ var1(3),var1(4),var1(5)
```

7.1.5 CLOSE コマンド

CLOSE コマンドは、シーケンシャル・ファイルの所有権を放棄します。

指定のファイルが開かれていないか、存在しない場合、InterSystems IRIS は、CLOSE を無視し、エラーを発行せずに返します。

7.1.5.1 構文

```
CLOSE file
CLOSE file:"D"
CLOSE file:("R":newname)
```

引数	説明
file	引用符で囲まれた有効なファイル指定です。指定するファイルは、既に開いている必要があります。UNIX のパス名には、チルダ (~) 展開を使用して、現在のユーザのホーム・ディレクトリを示すことができます。例 : ~myfile や ~/myfile。
"D"	この引数で指定された名前のファイルを閉じて削除します。
("R":newname)	この引数で指定された名前のファイルを閉じ、newname という名前を付けます。

7.1.5.2 CLOSE のみで使用するコマンド・キーワード

以下のテーブルは、CLOSE コマンドを使用してシーケンシャル・ファイルを制御する場合にのみ使用するキーワードの説明です。

テーブル 7-7: シーケンシャル・ファイルに対して CLOSE のみで使用するコマンド・キーワード

キーワード	既定値	説明
/DELETE[=n] または /DEL[=n]	ファイルを開くときに削除するようマークされていない限り、0	D パラメータ・コードに相当します。ファイルを削除するように指定します。/DELETE を指定した場合、または /DELETE=n の n が 0 以外の場合はパラメータ・コードが有効になり、/DELETE=n の n が 0 の場合は無効になります。
/RENAME=name または /REN=name	ファイル名を変更しない	R パラメータ・コードとファイル名位置パラメータに相当します。R パラメータ・コードは、ファイル名を変更し、ファイル名位置パラメータがファイルに新しい名前を付けるように指定します。

8

スプール・デバイス

InterSystems IRIS® Data Platform では、印刷出力をプリンタまたは画面に直接送信することも、後で印刷するためにスプール・グローバルに保持することもできます。InterSystems IRIS スプーリングは、ユーザのオペレーティング・システムで実行されるスプーリングから独立しています。

InterSystems IRIS スプーリングは、今すぐに印刷する代わりに、プログラムの出力を ^SPOOL 添え字付きグローバルに自動的に保存する技術です。プリンタに ^SPOOL グローバルのコンテンツを送信することで、後で出力を印刷できます。この章では、このスプーリング機能の 2 つの使用方法について説明します。その 1 つは ObjectScript コマンド (OPEN、USE、WRITE、CLOSE) を使用する方法で、もう 1 つは %IS ユーティリティと %SPOOL ユーティリティを使用する方法です。

8.1 スプール・デバイスのオープンと使用

現在のネームスペースのスプール・グローバルに出力を送信するには、そのスプーラを開き、出力デバイスとして指定します。

スプーラは InterSystems IRIS で提供される事前定義のデバイスです。スプーラには、デバイス・テーブルでデバイス番号 2 が割り当てられています。このデバイス番号を使用して、OPEN コマンド、USE コマンド、CLOSE コマンド内でスプーラ・デバイスを識別します。

スプーラ・デバイス情報には、管理ポータルからアクセスできます。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[デバイス]** の順に選択します。番号が 2 のデバイスと SPOOL というデバイスがあります。既定では、これらは同じ物理デバイス (デバイス 2) にマップされ、同じオプション値を持っています。

InterSystems IRIS スプーラを現在のデバイスとして設定する場合、InterSystems IRIS は、グローバル ^SPOOL のデバイス 2 に送信されるあらゆる出力を現在のネームスペースに格納します。各 ^SPOOL 行は、別々のグローバル・ノードに格納されます。

InterSystems IRIS スプーラを開き、現在のデバイスを設定する方法には以下の 2 種類があります。

- OPEN コマンドと USE コマンドを発行します。
- %IS ユーティリティを起動します。

8.1.1 スプーリング・デバイスへの OPEN コマンドと USE コマンド

OPEN コマンドと USE コマンドを発行して、スプーリング・デバイスを直接開くことができます。

```
OPEN 2:(doc_num:index) USE 2
```

テーブル 8-1: スプーリングの OPEN 位置パラメータ

パラメータ	定義
doc_num	開くスプール・ドキュメント (ファイル) の番号。スプール・ドキュメントは、^SPOOL グローバルに格納されています。既定は 1 です。
index	スプール・ドキュメント内の 1 以上の行番号。既定は 1 です。

これらは位置パラメータです。両方のパラメータを省略すると、パラメータ値は既定の (1:1) になります。第 1 パラメータ (doc_num) を設定し、第 2 パラメータ (index) を省略できます。この場合の既定は 1 です。第 2 パラメータを設定する場合、第 1 パラメータは必ず指定します。

InterSystems IRIS は、これらの値を使用し、出力する行を判断します。doc_num パラメータは、^SPOOL グローバルの最初の添え字として処理されます。index パラメータは、^SPOOL グローバルの 2 番目の添え字として処理されます。

8.1.1.1 USE コマンド

コマンド `OPEN 2:(doc_num:index)` の実行後、デバイス 2 に USE 2 コマンドを発行すると、InterSystems IRIS は以降の出力をすべて、`^SPOOL(doc_num:index)` のスプーラに送信します。各出力行は、^SPOOL に別々のグローバル・ノードとして格納されます。

8.1.1.2 WRITE コマンド

^SPOOL グローバルに行を書き込むには、行終了文字で終了する WRITE コマンドを発行します。以下はその例です。

ObjectScript

```
/* Writing to the ^SPOOL global */
OPEN 2
USE 2
  WRITE "First line of text",!
  WRITE "Second line of text",!
CLOSE 2

/* Displaying the ^SPOOL global */
WRITE ^SPOOL(1,1),^SPOOL(1,2)
```

各行は、行終了文字 (感嘆符) で終了し、別のグローバル・ノードに格納されます。

しかし、一行の出力を作成するために、複数の WRITE コマンドを使用する場合もあります。WRITE に行終了文字がない場合、次に続く WRITE コマンドが、同じ出力行に出力を追加します。両方とも同じグローバル・ノードに書き込みます。この行は、行終了文字が発行されるか、スプーラが閉じるまでバッファに保持され、スプール・グローバルには書き込まれません。

以下の例では、CLOSE が発行されると 1 つのグローバル・ノードが書き込まれます。

ObjectScript

```
/* Writing to the ^SPOOL global */
OPEN 2
USE 2
  WRITE "First half of line "
  WRITE "Second half of line"
CLOSE 2

/* Displaying the ^SPOOL global */
WRITE ^SPOOL(1,1)
```

行終了文字は、一般的には WRITE コマンド・コード文字の！(感嘆符) です。これは、キャリッジ・リターン (ASCII 13) と改行 (ASCII 10) の組み合わせと同等のものです。行を終了するには、これらの制御文字の両方が必要です。キャリッジ・リターン (ASCII 13) のみを発行すると、新規の行ノードを開始するのではなく、キャリッジ・リターンが行ノードに連結されることになります。ターミナルでは、このような行は、キャリッジ・リターンの後のテキストによってその前のテキストを上書きしたものとなります。

次の例では、^SPOOL ファイルの 2 つの行ノードのみが書き込まれます。

ObjectScript

```
/* Writing to the ^SPOOL global */
OPEN 2
USE 2
  WRITE "AAAAAAAAAA",$CHAR(10),$CHAR(13)
  WRITE "BBBBBBBBBB",$CHAR(13)
  WRITE "XXXX",!
CLOSE 2

/* Displaying the ^SPOOL global */
WRITE ^SPOOL(1,1),^SPOOL(1,2)
```

詳細は、"ObjectScript ランゲージ・リファレンス" の "OPEN" コマンド、"USE" コマンド、"WRITE" コマンド、および "CLOSE" コマンドを参照してください。

8.2 スプーリングと特殊変数

^SPOOL に書き込む場合、InterSystems IRIS は特殊変数である $X と $Y を継続的に更新します。$X は現在のインデックス行に書き込まれた文字数を示し、$Y は現在 OPEN されている行数を示します。$Y の値は、ノード・インデックスと同じである必要はありません。以下はその例です。

ObjectScript

```
/* Writing to the ^SPOOL global */
OPEN 2:(2:3)
USE 2
  WRITE "Hello " SET x1=$X,y1=$Y,z1=$ZA
  WRITE "world",! SET x2=$X,y2=$Y,z2=$ZA
  WRITE "Good to see you",! SET x3=$X,y3=$Y,z3=$ZA
CLOSE 2

/* Displaying the ^SPOOL global */
WRITE ^SPOOL(2,3),^SPOOL(2,4)
WRITE !,"$X=",x1," ",x2," ",x3
WRITE !,"$Y=",y1," ",y2," ",y3
WRITE !,"$ZA=",z1," ",z2," ",z3
```

例えば、最初の WRITE は $X=6 (現在の列数) を設定し、2番目、3番目のWRITE は (改行があるため) いずれも $X=0 を設定します。また、最初の WRITE は $Y=0 を設定し、2 番目は(改行があるため) $Y=0、3 番目は $Y=2 を設定します。書き込まれる行は ^SPOOL(2,3) と ^SPOOL(2,4) です。インデックス番号を決定するには $ZA を使用します。

スプール・ファイルに書き込むには、次に有効なインデックス番号で $ZA 特殊変数を設定します。したがって、index=3 に書き込み、行ターミネータを含まない場合、(次の WRITE コマンドがインデックス 3 に継続して書き込むので) $ZA=3 になります。しかし、行ターミネータを含む場合、$ZA=4 となります。

USE コマンドは、OPEN コマンドで指定されたスプール・ファイルの doc_num を含むように $ZB を設定します。

注釈　$IO 特殊変数は、スプール・ファイルへの書き込みで変更されません。通常 $IO は、現在のデバイスの ID を含むように USE コマンドでリセットされます。しかし、スプーラのように出力のみを行うデバイスを使用している場合、$IO は現在の入力デバイスの ID を継続して持ちつづけます。

詳細は、"ObjectScript ランゲージ・リファレンス" の特殊変数 "$X"、"$Y"、"$ZA"、"$ZB"、および "$IO" を参照してください。

8.3 スプール・デバイスのクローズ

デバイス 2 で CLOSE を発行すると、システムは、ノード ^SPOOL(doc_num,2147483647) を自動的に設定し、スプール・ドキュメントのクローズに関する情報と出力で使用された最大のインデックス番号に関する情報を格納します。

8.3.1 ネームスペースの変更

開いている状態のスプール・デバイスでネームスペースを変更すると、そのスプール・デバイスは自動的に閉じ、その後でネームスペースの変更が反映されます。^SPOOL グローバルを閉じると、そのレコードは正しいデータベースに書き込まれます。

8.3.2 ジョブの中止処理

スプール・デバイスを開き、現在のディレクトリをディスマウントした後、HALT コマンドまたは SYS.Process クラスの Terminate($JOB) メソッドを発行すると、InterSystems IRIS は、このスプール・デバイスに対する以降のアクセスに対して毎回必ず <PROTECT> エラーを返すようになります。この状態を避けるには、開いているすべての SPOOL デバイスが自動的に閉じるようにネームスペースを変更します。

8.4 ^SPOOL グローバルの表示

すべての添え字付きのグローバルと同様に、以下に示すように WRITE コマンドを発行することによってスプール・ファイルにある行を表示できます。

ObjectScript

```
WRITE "1st spool file node: ",^SPOOL(1,1),!
```

ただし、スプール・ファイル自体を表示および編集するには、管理ポータルに移動し、[**システムエクスプローラ**] の [**グローバル**] を選択します。現在のネームスペースを選択し、SPOOL グローバルを見つけ、[**データ**] をクリックします。これによって、次の例のようなスプール・ファイル・データが表示されます。

次のスプール・ファイルでは、(!) 終了文字によってスプール・ファイルの各ノード行が終了します。終了文字はスプール・ファイルの一部で、$CHAR(13,10)(Return と Line Feed) としてテキスト文字列に連結されます。

```
^SPOOL(1,1)=<<"First line of text"_$C(13,10)>>
^SPOOL(1,2)=<<"Second line of text"_$C(13,10)>>
^SPOOL(1,2147483647)={59605,43605{3{
```

次のスプール・ファイルには、行終了文字がありません。2 つの WRITE コマンドが 1 つのノード行を書き込みました。このノード行はスプール・ファイルを閉じることによって終了しました。

```
^SPOOL(1,1)=First half of line Second half of line
^SPOOL(1,2147483647)={59605,43725{2{
```

次のスプール・ファイルでは、キャリッジ・リターンと改行文字は WRITE コマンドに明示的にコード化されました。$CHAR(10) 改行文字は新規のノード行を開始し、$CHAR(13) キャリッジ・リターン文字はこれらのノード行に連結されます。

```
^SPOOL(1,1)=<<"AAAAAAAAAA"_$C(10)>>
^SPOOL(1,2)=<<$C(13)_"BBBBBBBBBB"_$C(13)_"XXXX"_$C(13,10)>>
^SPOOL(1,2147483647)={59605,44993{3{
```

このスプール・ファイルを閉じるとき、InterSystems IRIS がファイルの最終行を作成します。この行は、リテラル 1,2147483647、つまり、$HOROLOG 形式 (59605,44993) の日付と時刻、最終行を含むスプール・ファイルの行数から構成されています。

これらのスプール・ファイルのテキスト行は、管理ポータルの **[システムエクスプローラ]** の **[グローバル]** オプションで、その SPOOL グローバルの **[データ]** 表示を使用して編集または削除できます。

8.5 %IS ユーティリティを使用するスプーラのオープン

%IS は、ユーザがスプール・デバイスと、%SYS ネームスペースの ^%IS グローバルで定義された他のデバイスを選択できる便利なユーザ・インタフェースを提供します。%IS を使用すると、指定したスプール・ファイルを作成し、そのファイルにテキスト行を書き込めます。%SPOOL ユーティリティを使用して、このスプール・ファイルを印刷できます。

注釈　%IS ユーティリティを使用して開いたスプール・ファイルのみが、%SPOOL ユーティリティを使用して操作できます。

%IS を使用してスプール・ファイルを作成するには、以下の手順を実行します。

1. %IS ユーティリティを起動し、以下のスプーラを開きます。

   ```
   >DO ^%IS
   ```

2. “Device” プロンプトで「“2”」またはニーモニック「“SPOOL”」を入力します。
3. “Name” プロンプトで、スプール・ドキュメント (ファイル) 名を入力します (スプール・デバイスを開かない場合は、“Name” プロンプトで **Enter** キーを押します)。既存のスプール・ドキュメント名を入力すると、%IS はその名前が正確かどうか、ファイルの最後の行を表示するかどうか、また新規情報を追加する場所を選択するかどうかを尋ねます。新規の名前を入力すると、%IS は新規ドキュメントを生成するかどうかを尋ねます。新規スプール・ドキュメントを生成するには、**Enter** キーを、“Name” プロンプトを再表示するには「No」を入力します。
4. “Description” プロンプトで、説明を 1 行入力します。見やすいコードにするには、スプールされたドキュメントの詳細を別々の行に置き、1 行が長すぎる場合は、70 文字で次の行に送ります。

以下の例は、“TESTING SPOOL FUNCTIONALITY” 行を ^SPOOL グローバルに書き込みます。IO は、%IS が “Device” プロンプトで指定したデバイスに設定された変数です。

```
%SYS>DO ^%IS
Device: 2
Name: SPOOLFILE not found
Create new document 'SPOOLFILE'? Yes => <RETURN>
Description: This is my test spool file
%SYS>USE IO WRITE "TESTING SPOOLING FUNCTIONALITY",!
%SYS>CLOSE IO
```

8.6 %SPOOL を使用するスプールされたドキュメントの管理

%SPOOL ユーティリティを使用して、InterSystems IRIS スプール・デバイスにアクセスするときに作成されたスプール・ファイルを管理します。InterSystems IRIS スプーリングは、システム・スプーリングとは独立しています。

InterSystems IRIS のスプーリングは、プログラムの出力を直ちに印刷するのではなく、グローバル ^SPOOL に自動的に保存する技術です。プリンタにグローバルのコンテンツを送信することで、後で出力を印刷できます。

%SPOOL ユーティリティを使用して、現在のネームスペースの ^SPOOL グローバルで、スプール・ドキュメントの印刷、リスト、または削除を行うことができます。特定のネームスペースからスプーラにドキュメントを送信する場合、そのネームスペースから %SPOOL ユーティリティを実行して目的のドキュメントにアクセスする必要があります。

注釈　%IS ユーティリティを使用して開いたスプール・ファイルのみが、%SPOOL ユーティリティを使用して操作できます。

%SPOOL は、どのスプーリング・オプションを選択するかを尋ねます。以下を入力することで、3 つの機能のいずれかを選択できます。

- 機能のメニュー番号
- 機能名の最初の文字

また、疑問符 (?) を入力すると、これらの機能のリストを表示できます。

以下の例は、スプーリング機能の選択方法を示しています。ここでは、Print を選択しています。

```
%SYS>DO ^%SPOOL

Spool function: ?

The available spool functions are:

  1) Print
  2) List documents
  3) Delete document

Enter the number or first few characters of the name of the
spool function you want.

Spool function: 1 Print
```

以下のセクションでは、以下のタスクを実行する %SPOOL ユーティリティの使用方法を説明します。

- スプール・ドキュメントの印刷
- スプール・ドキュメントのリスト
- スプール・ドキュメントの削除

8.6.1 %SPOOL を使用した印刷

%SPOOL ユーティリティ・メニューのオプション 1 の Print を使用して、任意のデバイスでの ^SPOOL グローバルのドキュメントの印刷、中断された印刷の再開、高品質プリンタへの用紙の手送りを行うことができます。出力をスプーラに送信することで、出力デバイスがドキュメントを印刷する間は、他のプロセスで使用できるようにターミナルが開放されます。

スプール・ドキュメントが完全に作成される前または作成された後のどちらかで、印刷を開始できます。プリンタが新規の出力を再開する場合、印刷プロセスを 5 秒間停止した後、その間蓄積されたすべての出力を印刷します。この印刷プロセスは、スプール・ドキュメントが閉じた時期を認識し、そのドキュメントの処理が完了すると終了します。

%SPOOL がドキュメントを印刷するとき、印刷が完了したページの記録を保持します。また、ページ番号でドキュメントのソートを行い、選択したページの最初から印刷を行うことができるように、ページ・インデックスの作成も行います。

印刷を停止した場合 (例えば、ターミナル出力中に Ctrl-c を押して停止した場合や、プリンタが故障した場合)、一部が印刷されて中断したページの先頭から印刷できるほか、ドキュメント内の任意のページの先頭から印刷を再開することもできます。InterSystems IRIS は、ページ・カウント内のドキュメントの開始点から、フォーム・フィードをページとしてカウントしません。

%SPOOL では、デスプールという語で印刷処理を表現します。ドキュメントの印刷が完了 (デスプール) した場合のみ、Despool start-end 列と説明行に値が存在します。

8.6.1.1 印刷機能の使用法

1. “Spool function:” プロンプトで「1」を入力します。
2. “Name:” プロンプトで、ヘルプ・テキストを表示するには「?」を入力します。現在のネームスペースにある既存のスプール・ドキュメントをすべて一覧表示するには「??」を入力し、スプール・ドキュメントを印刷するにはその名前を入力します。%SPOOL は、それが適切なドキュメントであるかを確認します。
3. %SPOOL から印刷を開始するページを尋ねられた場合は、ドキュメントのページ番号を入力します。最初のページから開始するには Enter キーを押します。印刷プロセスがまだ取得していないページの先頭から印刷を開始しようとすると、WARNING: Printing hasn't reached this point というメッセージが表示されます。この警告メッセージの後、%SPOOL は、指定したページから印刷を開始するかどうかを尋ねます。「No」と入力すると、“Start at page:” プロンプトに戻ります。「Yes」と入力すると、%SPOOL はページの最初の数行を表示して、印刷するページであるかどうか確認することを求めます。
4. 印刷する部数の入力を求められます。
5. %SPOOL では、印刷するドキュメントとして、他のスプール・ドキュメント名を入力できます。Enter キーを押して “Name:” プロンプトに応答すると、出力デバイスとその右側のマージンを尋ねられます。それに対して応答し、印刷を開始します。

   %SPOOL は各ページの後にフォーム・フィードを発行して、画面に表示するか、プリンタで印刷するかを確認します。

以下の例では、(この場合は SPOOLFILE と呼ばれる) ^SPOOL グローバルでドキュメントを印刷する方法を示しています。ドキュメントは、MYPRINTER と呼ばれるデバイスで印刷されます。

```
%SYS>DO ^%SPOOL

Spool function: 1  Print
Name: ??

#  Name        Lines   Spool start      Despool start-end
1  SPOOLFILE   1      30 Aug  2:23 pm    30 Aug  2:25 pm-2:25 pm
  This is my test spool file

Name: SPOOLFILE

1  SPOOLFILE    30 Aug 2003  2:23 pm  this is my test spool file
SPOOLFILE has 1 pages.
Is this correct?  Yes=>Y
Start at page:  1=>Y
How many copies?  1=>Y

Name:RETURN
Print spooled files on
Device: MYPRINTER RETURN Parameters: "WNS"=>
Free this terminal? Yes =>Y
Starting Job in background . . . started.

Spool function:
```

8.6.2 スプールされたドキュメントのリスト

%SPOOL ユーティリティ・メニューのオプション 2 は、%SPOOL の現在の実行場所であるディレクトリについて、現在スプールされているドキュメントのリストを表示します。Despool start-end 値がない場合は、ドキュメントはまだデスプール (印刷) されていないということです。

スプールされた各ドキュメントの説明は、ドキュメントに関する残りの情報の後に、1 つまたは複数の行に表示されます。

以下の例では、オプション 2 が選択されています。その結果、スプーラに保存されている 2 つのドキュメントが表示されます。最初は 8 月 30 日の午後 2 時 23 分に保存され、同じ日の午後 2 時 25 分に印刷されました。2 番目は 3 月 4 日の午前 11 時 39 分に保存され、同じ日の午前 11 時 42 分に印刷されました。

```
Spool function: 2   List documents

# Name      Lines   Spool start     Despool start-end
1 SPOOLFILE  1     30 Aug  2:23 pm  30 Aug  2:25 pm- 2:25 pm
  This is my test spool file

3 LONGFILE   1     04 Mar 11:39 am  04 Mar 11:42 am- 11:42 am
  This is a very long description line that shows you what happens when you
have a long description. It shows you how the text wraps from line to line.
This particular description was made intentionally long, so as to wrap at least
twice.
```

8.6.3 スプールされたドキュメントの削除

%SPOOL ユーティリティ・メニューのオプション 3 の Delete Documents で、1 つ以上のスプール・ドキュメントを削除できます。%SPOOL で名前の入力を要求されたら、削除するドキュメント名を入力するか、?? を入力して現在のネームスペースにあるスプール・ドキュメントを表示します。ヘルプ・テキストを表示するには ? を入力します。

%SPOOL は、それが適切なドキュメントであるかどうか、さらにそのドキュメントを本当に削除するかどうかを尋ねます。「“Yes”」と答えると、%SPOOL によってドキュメントが削除され、次に削除するドキュメント名を入力できるようになります。

以下の例は、SPOOLFILE と呼ばれるスプールされたドキュメントを削除します。

```
Spool function: 3   Delete document
Name: ??

# Name        Lines  Spool start        Despool start-end
1 SPOOLFILE     1    30 Aug  2:23 pm  30 Aug  2:25 pm- 2:25 pm
  This is my test spool file

Name: SPOOLFILE

1 SPOOLFILE    30 Aug 2003  2:23 pm  this is my test spool file
SPOOLFILE has 1 pages.
Is this correct?  Yes=>Y
Delete SPOOLFILE? No=> Y [Deleted]

Name:
```

9
プリンタ

この章では、InterSystems IRIS® Data Platform で印刷デバイスを構成および使用する方法について説明します。プリンタは、物理的な出力限定デバイスです。プリンタは、文字プリンタ、またはファックスやプロッタなどの文字以外のデバイスです。

ほとんどの場合、出力は、直接プリンタには送信されません。多くの場合、印刷される出力は、まず、論理スプール・デバイス(^SPOOL グローバル)に送信されます。その後、^SPOOL グローバルの内容を、物理的プリンタに送信できます。スプーリングの詳細は、このドキュメントの "スプール・デバイス" の章を参照してください。

9.1 プリンタの概要

Windows および UNIX® では、プリンタ入出力の扱いがそれぞれ異なっています。

- Windows システムではプリンタをシーケンシャル入出力デバイスとして扱い、シーケンシャル・ファイル入出力と同じ構文に従います。ただし、シリアル通信ポートを通じて接続されているプリンタは、ターミナル入出力デバイスとして処理されます。
- UNIX® システムでは、常にプリンタがターミナル入出力デバイスとして処理されます。UNIX® ではプリンタを tty デバイスの "character special" ファイルとして扱い、ターミナル入出力と同じ構文に従います。

Windows システムでは、%Library.Device.InstalledPrinters() メソッドを使用して、ご使用のシステムにおける現在のプリンタ数を返すことができます。%Library.Device.GetPrinters() メソッドを使用すれば、ご使用のシステムにおける現在のプリンタのリストを返すことができます。

9.2 プリンタの指定

プリンタには、256 から 2047 までのデバイス番号を割り当てることができます。このデバイス番号の範囲は、ターミナルおよびフラット・ファイルにも使用されます。

Windows システムでは、そのデバイス番号または割り当てられたデバイス・ニーモニックを使用してプリンタを参照できます。Windows の既定のプリンタ・ニーモニックは、|PRN| です。

プリンタを指定するには、以下の 2 つの方法があります。

- %IS ユーティリティを呼び出すと、%IS グローバルで定義されたニーモニックを使用して、デバイスを指定できます。このユーティリティにより、デバイスが開かれ、そのパラメータが設定されます。

- 引用符で囲まれた文字列として指定したオペレーティング・システム・デバイス名を使用して、入出力コマンドの OPEN、USE、および CLOSE を発行します。

9.2.1 プリンタのオープン

プリンタを開く際、デバイス名を使用して、デバイスを指定できます。デバイス名は、引用符で囲む必要があります。このデバイスの名前の最大長は 256 文字です。形式は以下のとおりです。

```
OPEN "device" USE "device" CLOSE "device"
```

Windows では、プリンタをシリアル通信ポートに接続することもできます。この場合、プリンタは、以下の構文を持つターミナル入出力と同様に扱われます。

```
OPEN "comn:"
```

n は、プリンタが接続されるポート番号です。

9.2.2 Windows でのプリンタの指定

Windows で既定のプリンタを使用するには、以下を入力します。

ObjectScript

```
OPEN "|PRN|"
```

使用しているコンピュータに対して既定の Windows プリンタを設定してある場合は、これによってその既定のプリンタが使用されます (既定のプリンタの設定方法は、Windows のドキュメントを参照してください)。

既定以外のプリンタを使用するには、以下を入力してください。

ObjectScript

```
OPEN "|PRN|printer"
```

パラメータ	説明
printer	UNC (Universal Naming Convention) 名、あるいは、マシンのプリンタ・リストに表示される名前 (Windows NT/2000/XP のプリント・マネージャ)。

以下の例は、UNC 名の使用方法です。

ObjectScript

```
OPEN "|PRN|\\business\accounting"
```

以下の例は、マシンのプリンタ・リストに表示される UNC 名以外の名前の使用方法です。

ObjectScript

```
OPEN "|PRN|HP LaserJet 5P"
```

注釈　インターシステムズは、COM1、LPT1 のようなプリンタ・ポート名を使用しないことをお勧めします。このような名前を使用すると、どのプリンタであるか検索され、それが存在する場合、その名前が参照されて使用されます。しかし、この処理には時間がかかるため、Windows では適切ではありません。

Windows システムでは、プリンタは、OPEN コマンドの名前によって識別され、ターミナル入出力モジュールではなくシーケンシャル入出力モジュールによって処理されます。したがって、サポートされている OPEN および USE のコマンド引数は、シーケンシャル・ファイル入出力のコマンド引数と同じであり、ターミナル入出力のコマンド引数とは異なります。例外は、シリアル通信ポートで Windows システムに接続されているプリンタで、その場合、プリンタはターミナル入出力デバイスとして処理されます。

Windows システムでは、“:n” 位置パラメータを使用して出力マージンを制御することはできません。そのため、InterSystems IRIS ではこのような記述は無視されます。例えば、`"|PRN|":121` のようなコードは無視されます。出力幅を制御するには、プリンタに適切な制御文字を送信します。これは、他のプラットフォームにも当てはまります。

Windows では、OPEN は、ほとんどのシーケンシャル入出力キーワード・パラメータをサポートします (このドキュメントの″シーケンシャル・ファイルの入出力″ の章を参照)。以下のテーブルは、OPEN コマンドを使用する Windows システムで (シーケンシャル・ファイルとして扱われる) プリンタを制御する追加のプリンタ・キーワード・パラメータの説明です。

テーブル 9-1: Windows プリンタ用の追加 OPEN キーワード・パラメータ

キーワード	既定	説明
/DOCNAME= “name”	“IRIS”	/DOCNAME により、プリンタ・ジョブ名を再定義できます。
/OUTPUTFILE= “filename”	NULL	/OUTPUTFILE により印刷をファイルに転送します。完全修飾パス名を指定してください。
/DATATYPE= “type”	“RAW”	/DATATYPE により、プリンタ・スプール・データのデータ型を再定義できます。頻繁に使用されるデータ型の 1 つは TEXT です。

Windows システムでは、OPEN で直接プリンタに印刷する (論理スプール・デバイスを使用しない) と、プリンタがオフの状態、もしくは存在しない場合、OPEN コマンドの timeout 引数は期限切れになりません。InterSystems IRIS プロセスは、プリンタが使用可能になるまで、または Windows のコントロール パネルでドキュメントの印刷がキャンセルされるまで、中断された状態のままになります。

9.2.3 UNIX® でのプリンタの指定

UNIX® デバイス名 /dev/tty06 を持つターミナルで入出力デバイスを開くには、以下のコマンドを入力します。

ObjectScript

```
OPEN "/dev/tty06"
```

UNIX® システムでは、プリンタは、OPEN コマンドで指定した名前で識別され、tty デバイスの “character special” ファイルとして処理されます。したがって、サポートされる OPEN と USE のコマンド引数は、ターミナル入出力のコマンド引数と同じであり、シーケンシャル・ファイル入出力のコマンド引数とは異なります。

UNIX® では、OPEN は、ほとんどのターミナル入出力キーワード・パラメータをサポートします (このドキュメントの ″ターミナル入出力″ の章を参照)。

9.3 プリンタへの出力先指定

%IS ユーティリティを使用して、プリンタを出力先に指定できます。`DO ^%IS` コマンドを使用して、%IS ユーティリティを呼び出すことができます (`DO OUT^%IS` を使用して、出力限定デバイスを選択していることを指定することもできます)。いずれの場合でも、InterSystems IRIS は、`Device:` プロンプトを返します。プリンタを指定するには、既定のニーモニック ″|PRN|″ または別の構成済みプリンタのニーモニックで応答します。その後、%IS ユーティリティは、OPEN パラメータを

提示します。プリンタの場合、既定は “W” (書き込み専用) です。以下の例に示すように、Enter キーを押して、既定のパラメータを受け入れることができます。

```
%SYS>DO ^%IS
Device: |PRN|
Parameters? "W" => <RETURN>
%SYS>
```

これによって、現在のプロセスの出力デバイスとして指定されたプリンタが開かれます。

%IS ユーティリティは、各種変数を設定します。以下は、Windows システムでのプリンタの既定値です。

テーブル 9-2: %IS により設定される変数

変数	値	説明
IO	|PRN|	選択したデバイスのデバイス・ニーモニック
IOF	#	改ページ文字。WRITE # は、改ページ発行し、$Y を変更します。改ページには WRITE @IOF を使用する必要があります。
IOBS	$C(8)	バックスペース/オーバープリント文字 (ASCII 8)。WRITE $CHAR(8) は、バックスペースを発行し、$X を変更します。WRITE *8 は、バックスペースを発行しますが、$X を変更しません。バックスペースには、WRITE @IOBS を使用する必要があります。
IOM	132	右マージン。文字単位での行の長さ。
IOSL	66	文字単位でのページの長さ。
IOT	OTH	デバイス・タイプ。ここでは “その他” です。
IOST	P-DEC	デバイス・サブタイプ名。
IOPAR	("W")	OPEN パラメータ。プリンタは、書き込み専用デバイスなので、ここでは “W” です。
MSYS	M/WNT	システム・タイプ (UNIX® または Windows NT)。M/WNT は、Windows NT での InterSystems IRIS です。
POP	0	%IS が実行された (これらの変数が初期化された) ことを示します。0 の場合、デバイスが指定されました。1 の場合、デバイスは指定されていません (ユーザが Device: プロンプトで「STOP」と入力しました)。

これらの値のほとんどは、管理ポータルから表示および設定することもできます。[**システム管理**]、[**構成**]、[**デバイス設定**] の順に選択します。[**デバイス**] および [**デバイスサブタイプ**] の現在の内容を確認します。特定のプリンタの設定を表示するには、[**編集**] を選択します。

9.3.1 %IS プリンタ設定変数

スプーリングのために %IS を呼び出すと、プリンタの書式を設定するルーチンの名前を指定する IOPGM 変数をこの %IS に渡すことができます。

9.4 代替デバイスとしてのプリンタ

“A” という名前の新規デバイスを定義し、|PRN| の物理デバイスを指定することによって、すべてのプロセスについて、代替デバイスとしてプリンタを指定できます。その後、%IS を使用する時に、`Device:` プロンプトで A を指定します。

プリンタをターミナル・プロセスのための代替入出力デバイスとして設定できます。管理ポータルに移動します。**[システム管理]**、**[構成]**、**[デバイス設定]**、**[デバイス]** の順に選択します。現在のターミナル・デバイスに対して [編集] を選択し、[代替デバイス] の値を指定します。