監視ガイド

Version 2023.1
2024-01-02

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

監視ガイド
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

テーブル一覧

1

管理ポータルを使用した InterSystems IRIS の監視

管理ポータルのシステム・ダッシュボードでは、InterSystems IRIS® Data Platform インスタンスをさまざまな側面から監視できます。ダッシュボードを使用してパフォーマンス・インジケータを表示し、さらに選択したインジケータに関する詳細情報を参照できます。

一般的な InterSystems IRIS 監視ツールの概要については、"システム監視ツール" を参照してください。

注釈　ここで説明する**システム処理**ツールにアクセスするユーザは、**%Admin_Operate** リソースに対する特権を持つロールのメンバである必要があります。詳細は、管理ポータルの使用に関するガイドのこちらのセクションを参照してください。

1.1 システム・ダッシュボードのインジケータの監視

管理ポータルの **[システム処理]** > **[システムダッシュボード]** ページでは、重要なシステム・パフォーマンス・インジケータの状態が、以下のカテゴリにグループ化されています。各カテゴリの説明は、以下の表のいずれかにあります。

- システム・パフォーマンスのインジケータ
- ECP インジケータ
- システム時刻のインジケータ
- システム使用のインジケータ
- エラーとアラートのインジケータ
- ライセンスのインジケータ
- タスク・マネージャのインジケータ

多くの場合、いずれかのカテゴリにリストされているインジケータをクリックすると、ページの左下隅の詳細ボックスに、インジケータの説明を表示できます。

テーブル 1-1: システム・パフォーマンスのインジケータ

インジケータ	定義
グローバル/秒	最後に測定した 1 秒あたりのグローバル参照数。

インジケータ	定義
グローバル参照数	システム起動後のグローバル参照数。
グローバル更新数	システム起動後のグローバル Set と Kill 処理の数。
ルーチン参照	システム起動後に発生したルーチンのロードおよび保存の回数。
論理要求	システム起動後に発生した論理ブロック要求の数。
ディスク読込	システム起動後に発生した物理ブロック読み取り操作の回数。
ディスク書出	システム起動後に発生した物理ブロック書き込み操作の回数。
キャッシュ効率	最後に測定されたキャッシュ効率 (グローバル参照数/(物理的な読み取り数 + 書き込み数))。

下部の詳細ボックスの [... さらに詳細] リンクをクリックして、[システム処理] > [システム使用] ページを表示します。詳細は、"システム・パフォーマンスの監視" のセクションを参照してください。

テーブル 1-2: ECP インジケータ

インジケータ	定義
アプリケーション・サーバ	このシステムに接続されている ECP (エンタープライズ・キャッシュ・プロトコル) アプリケーション・サーバの状態。
アプリケーション・サーバ・トラフィック	最後に測定された ECP アプリケーション・サーバ・トラフィック (バイト数/秒)。
データ・サーバ	このシステムが接続されている ECP データ・サーバの状態。
データ・サーバ・トラフィック	最後に測定された ECP データ・サーバ・トラフィック (バイト数/秒)。

ECP インジケータの詳細は、"スケーラビリティ・ガイド" の "InterSystems 分散キャッシュによるユーザ数に応じたシステムの水平方向の拡張" の章を参照してください。

テーブル 1-3: システム時刻のインジケータ

インジケータ	定義
システム稼動時間	このシステムを起動してからの経過時間。
最終バックアップ	システムを最後にバックアップした日時。

[システム処理] > [バックアップ] ページから、バックアップを実行したり、バックアップの履歴を参照することができます。バックアップ計画の開発の詳細は、"データ整合性ガイド" の "バックアップとリストア" の章を参照してください。

テーブル 1-4: システム使用のインジケータ

インジケータ	定義
データベース容量	データベース・ファイルを格納できるだけのディスク容量があるかどうかを示します。[... さらに詳細] をクリックすると、[システム処理] > [データベース] ページが表示されます。
データベース・ジャーナル	データベース・ジャーナルの現在の状態を示します。[... さらに詳細] をクリックすると、[システム処理] > [ジャーナル] ページが表示されます。
ジャーナル空き	ジャーナル・ファイルを格納できるだけのディスク容量があるかどうかを示します。[... さらに詳細] をクリックすると、[システム処理] > [ジャーナル] ページが表示されます。

インジケータ	定義
ジャーナル・エントリ	システム・ジャーナルに書き込まれたエントリの数。[... さらに詳細] をクリックすると、[システム処理] > [ジャーナル] ページが表示されます。
ロック・テーブル	システムのロック・テーブルの現在の状態。[... さらに詳細] をクリックすると、[システム処理] > [ロック] > [ロックの管理] ページが表示されます。
ライト・デーモン	システムのライト・デーモンの現在の状態。
トランザクション	開いているローカル・トランザクションおよびリモート (ECP) トランザクションの現在の状態。開いているトランザクションがない場合、状態は [通常] になります。状態には [警告] (最も長く開いているローカル・トランザクションまたはリモート・トランザクションの時間が 10 分を超える場合) および [トラブル] (20 分より長い場合) もあります。[... さらに詳細] をクリックすると、[トランザクション] ページが表示されます。
プロセス	現在、実行されているプロセスの数。[... さらに詳細] をクリックすると、[プロセス] ページ ([システム処理] > [プロセス]) が表示されます。
Web セッション	最新の Web セッション数。[... さらに詳細] をクリックすると、[Web セッション] ページ ([システム処理] > [Web セッション]) が表示されます。
最もアクティブなプロセス	アクティビティ (実行されたコマンド数) が最も多い実行中のプロセス。[... さらに詳細] をクリックすると、[プロセス] ページ ([システム処理] > [プロセス]) が表示されます。

これらのトピックの詳細は、[... **さらに詳細**] リンクをクリックすると表示される、ポータル・ページの [**ヘルプ**] リンクをクリックしてください。

テーブル 1-5: エラーとアラートのインジケータ

インジケータ	定義
深刻な警告	深刻な警告の発生数。[... さらに詳細] をクリックすると、[メッセージログを表示] ページ ([システム処理] > [システムログ] > [メッセージログ]) が表示されます。
アプリケーション・エラー	ログに記録されたアプリケーション・エラーの数。[... さらに詳細] をクリックすると、[アプリケーションエラーログの表示] ページ ([システム処理] > [システムログ] > [アプリケーションエラーログ]) が表示されます。

詳細は、この章の "ログ・ファイルの監視" のセクションを参照してください。

テーブル 1-6: ライセンスのインジケータ

インジケータ	定義
ライセンス制限	このシステムで使用できる最大ライセンス数。
現在のライセンス使用	ライセンスの使用状況が、使用可能なライセンス数に占める割合として表示されます。
ライセンスの使用最高値	ライセンスの使用最高値が、使用可能なライセンス数に占める割合として表示されます。

下部の詳細ボックスの [... **さらに詳細**] リンクをクリックして、[**システム処理**] > [**ライセンス使用量**] ページを表示します。ライセンスの詳細は、"システム管理ガイド" の "InterSystems IRIS ライセンスの管理" の章を参照してください。

テーブル 1-7: タスク・マネージャ – タスクの実行予定タスク

インジケータ	定義
実行予定タスク	次の 5 つの実行予定タスクのリスト。
タスク	実行予定タスクの名前
時刻	タスクの実行が予定されている時間
状態	タスクの状態 (予定、完了、または実行中)。

下部の詳細ボックスの [... さらに詳細] リンクをクリックして、[システム処理] > [タスクマネージャ] > [実行予定タスク] ページを表示します。タスク・マネージャの詳細は、"システム管理ガイド" の "InterSystems IRIS の管理" の章にある "タスク・マネージャの使用" セクションを参照してください。

1.2 システムの使用状況とパフォーマンスの監視

以下のテーブルは、システム・パフォーマンスのメトリックについての説明です。

- システム使用テーブル
- 共有メモリ・ヒープ使用状況

1.2.1 システム使用テーブル

システム使用の統計情報を表示するには、[システム使用] ページ ([システム処理] > [システム使用]) に移動します。

テーブル 1-8: システム使用の統計情報

統計	定義
グローバル参照 (すべて)	グローバルへのアクセスの論理カウント (Set、Kill、$Data、$Order、$Increment、$Query、および式内のグローバル参照)。
グローバル更新参照	グローバル参照 (Set、Kill、$Increment) 処理の論理カウント。
ルーチン呼び出し	ルーチンへの呼び出し数。
ルーチン・バッファの読み取りと保存	ZLoad、ZSave、および実行中のルーチンの結果として、ルーチンの読み取りと保存が行われた合計数(適切に調整された環境の場合、ほとんどのルーチンはルーチン・キャッシュ・メモリに既に格納されており、ディスクにアクセスする必要がありません。したがって、この数値は緩やかに増加します。1 回のルーチンの読み込みまたは保存では、最大 32 KB (Unicode では 64 KB) のデータが転送されます)。
論理ブロック要求	グローバル・データベース・コードによって読み取られたデータベース・ブロックの数 (適切に調整された環境では、通常、これらの読み取りはディスクにアクセスしないで実行されます)。
ブロック読み取り	グローバル参照とルーチン参照の両方について、ディスクから読み取られた物理データベース・ブロックの数。
ブロック書き込み	グローバル参照とルーチン参照の両方について、ディスクに書き込まれた物理データベース・ブロックの数。
WIJ 書き込み	イメージ・ジャーナル・ファイルに書き込まれたブロックの数。

統計	定義
ジャーナル・エントリ	作成されたジャーナル・レコードの数。ジャーナル・レコードは、データベースの修正 (Set、Kill など)、トランザクション・イベント (TStart、TCommit)、ジャーナルに保存されたその他のイベントごとに 1 つ作成されます。
ジャーナル・ブロック書き込み数	ジャーナル・ファイルに書き込まれた 64 KB ジャーナル・ブロックの数。
ルーチン行	システム起動後に実行されたルーチン行数。
最終更新	表示された統計情報の日時とタイムスタンプ。

これらの統計情報を監視する別の方法については、“^GLOSTAT を使用したグローバル動作の統計収集” の章を参照してください。

1.2.2 共有メモリ・ヒープ使用状況

InterSystems IRIS の共有メモリ・ヒープ (gmheap) 使用状況を表示するには、[システム使用] ページ ([システムオペレーション] > [システム使用]) に移動して、[共有メモリヒープ使用状況] リンクをクリックします。

注釈　共有メモリ・ヒープ (gmheap) のサイズを変更する方法は、"構成パラメータ・ファイル・リファレンス" の “gmheap” のエントリにある "このパラメータの変更" のセクションを参照してください。

このページのテーブルにある列見出しは、以下を表します。

- **説明** – 共有メモリが割り当てられる目的。
- **割り当てられた SMH/ST** – その目的のために割り当てられた共有メモリ・ヒープ (gmheap) とストリング・テーブル・メモリの合計。
- **SMH/ST 利用可能** – その目的のために割り当てられ、現在でも使用可能な、共有メモリ・ヒープ (gmheap) およびストリング・テーブル・メモリ。
- **SMH/ST 使用中** – その目的のために割り当てられ、使用中の、共有メモリ・ヒープ (gmheap) およびストリング・テーブル・メモリ。
- **SMT 使用中** – その目的で使用されている静的メモリ・テーブルのメモリ。
- **GST 使用中** – その目的で使用されている一般ストリング・テーブル・メモリ。
- **使用中すべて** – その目的で使用されているすべてのメモリの合計。

テーブル 1-9: 共有メモリ・ヒープ使用状況

識別子	定義
その他	静的メモリ・テーブル (SMT) および一般ストリング・テーブル (GST) に割り当てられている共有メモリ
監査システム	システム監査に使用する共有メモリ
インスタンス化されたクラス	クラス・ハッシュ・テーブルおよび制御ブロックに割り当てられた/利用可能な/使用する共有メモリ
データベース暗号化キーの変更	データベース暗号化キーの変更に割り当てられた/利用可能な/使用する共有メモリ
セマフォ・オブジェクト	セマフォ・オブジェクトに割り当てられた/利用可能な/使用する共有メモリ
イベント・システム	イベント・システムに割り当てられた/利用可能な/使用する共有メモリ

識別子	定義
グローバルマッピング	グローバル・マッピングおよび添え字レベル・マッピング (SLM) に割り当てられた/利用可能な/使用する共有メモリ
ライセンスアップグレード	ライセンス・アップグレードに割り当てられた/利用可能な/使用する共有メモリ
ロック・テーブル	ロック・システムに割り当てられた/利用可能な/使用する共有メモリ
国際化言語サポート	国際化言語サポート (NLS) テーブルに割り当てられた/利用可能な/使用する共有メモリ
パフォーマンス・モニタ	パフォーマンス・モニタ (^PERFMON) に割り当てられた/利用可能な/使用する共有メモリ。
プロセステーブル	プロセス ID (PID) テーブルに割り当てられた/利用可能な/使用する共有メモリ
使用中ルーチン・バッファ・テーブル	使用中ルーチン・バッファ・テーブルに割り当てられた/利用可能な/使用する共有メモリ
セキュリティシステム	セキュリティ・システムに割り当てられた/利用可能な/使用する共有メモリ
共有ライブラリ	共有ライブラリに割り当てられた/利用可能な/使用する共有メモリ
TTYハッシュテーブル	TTY ハッシュ・テーブルに割り当てられた/利用可能な/使用する共有メモリ
DB 名 & ディレクトリ	データベース名およびディレクトリに割り当てられた/利用可能な/使用する共有メモリ
iKnow 言語モデル・データ	iKnow 言語モデルに割り当てられた/利用可能な/使用する共有メモリ
ECP とシャドウイングのインジケータ	ECP に割り当てられた/利用可能な/使用する共有メモリ
拡張デーモン	拡張デーモンに割り当てられた/利用可能な/使用する共有メモリ
合計	各列のメモリ合計 注釈　　列見出しの上にカーソルを置くと、各列の説明が表示されます。
利用可能な SMT & GST	静的メモリ・テーブル (SMT) および一般ストリング・テーブル (GST) で利用可能なメモリ。
SMT & GST 合計割り当て	静的メモリ・テーブル (SMT) および一般ストリング・テーブル (GST) の使用されているメモリと使用可能なメモリの合計。
合計 SMH ページ割り当て	直接割り当てられた共有メモリ・ヒープ (SMH) およびストリング・テーブルの割り当てられた共有メモリに、静的メモリ・テーブル (SMT) および一般ストリング・テーブル (GST) の使用されているメモリと使用可能なメモリを合わせた合計。64 KB ページの番号は括弧内に表示されます。

1.2.3 SQL アクティビティの監視

IRIS システムで現在実行されている SQL 文を検査するには、[**SQL アクティビティ**] ページ ([**システムオペレーション**] > [**SQL アクティビティ**]) に移動します。このページには、アクティブな各 SQL 文に関する以下の情報を示したテーブルが表示されます。

- それに関連付けられている [**プロセス**] ID
- それを実行している [**ユーザ**] の ID
- その文が照会する 1 つ以上のテーブルを含む [**ネームスペース**]

- その文の [**タイプ**] (例えば、ダイナミック SQL クエリの場合は DynamicQuery)
- その文が実行を開始してからの [**経過時間**]
- [**文**] 自体のテキスト抜粋

このテーブル内のいずれかの行を選択すると、さらに別の 2 つのテーブルが表示されます。それらのテーブルには、対応する SQL 文に関する以下のような追加の詳細情報が含まれています。

テーブル 1-10: 選択した文の詳細

行ラベル	値
プロセス	文に関連付けられたプロセスの ID。このフィールドは、このプロセスの [プロセス詳細] ページにリンクされています。
トランザクション	文が、SQL トランザクションの一部としてアクティブかどうか。
開始時刻	文が実行を開始した時刻。
パラメータ	該当する場合、文が対応している最初の 10 個のパラメータ。ダイナミック SQL クエリの場合、これは、クエリにパラメータとして入力されたリテラル値のリストで、出現する文字 "?" がリストの順序に置換されます。INSERT や UPDATE などのコマンドの場合、これは、レコードに対して挿入または更新されるフィールドの値のリストです。
文	文のテキスト全体。該当する場合、このフィールドには、この文の [SQL 文の詳細] ページへのリンクも含まれます。
クエリ・キャッシュ	該当する場合、文がキャッシュされるルーチンの名前。

テーブル 1-11: 実行の統計

行ラベル	値 (全体および先週)
Times executed	文が実行された回数。
Average rowcount	各実行で文が返した行数の平均。
Average runtime	この文の平均実行時間。
Standard deviation	指定の時間間隔における、文の実行時間の変化の度合い。

1.3 ロックの監視

ObjectScript のローカル変数またはグローバル変数に対して InterSystems IRIS プロセスが LOCK コマンドを発行すると、そのエンティティが他のプロセスによって既にロックされていない限り、InterSystems IRIS のロックが作成されます。ロックするエンティティは、データベースに存在していなくてもかまいません。InterSystems IRIS のロックの詳細は、"ロックと並行処理の制御" の項目を参照してください。

システム全体のロックを表示するには、[**ロック表示**] ページ ([**システム処理**] > [**ロック**] > [**ロック表示**]) に移動します。選択したシステム全体のロックを削除するには、[**ロックの管理**] ページ ([**システム処理**] > [**ロック**] > [**ロックの管理**]) に移動します。いずれの場合も、表示されるロック・テーブルでは、保持されたロックと待機しているロック要求ごとに、所有者を特定する 1 行が表示されます。1 行に、同一エンティティについて 1 人の所有者が保持している複数のロックが示される場合もあります。例えば、増分ロックを保持している場合、または共有ロックと排他ロックを両方保持している場合などです。複数のプロセスが同じエンティティのロックを保持している場合は、各所有者に専用の行が割り当てられます。

[**ロックテーブル**] には以下の列エントリがあります。

テーブル 1-12: ロックの詳細

列名	定義
所有者	ロックを保持または待機しているプロセスのプロセス ID。リモート・ロックの場合、クライアントのシステム名が含まれます。
ModeCount	ロックのモード、およびロックのインクリメント・カウント。ロック・カウントが 1 の場合、そのカウントは表示されません。ModeCount 値のリストは、"ObjectScript の使用法" の "ロック管理" の章を参照してください。
参照	ロック項目のロック参照文字列 (データベース名は含まれません)。
ディレクトリ	ロック項目が存在するデータベース。
システム	ロックが検出されたシステムの名前。ローカル・システムの場合は、この列は空白で表示されます。
ルーチン	ロックを保持または待機しているプロセスによって現在実行されているルーチン行。
削除	[ロックの管理] のみ : このロックが削除可能な場合、[プロセスが保持するすべてのロックを削除] オプション (ローカル・ロックの場合) または [リモート・クライアントからすべてのロックを削除] オプション (リモート・ロックの場合) と共にこのオプションが行に表示されます。適切なオプションをクリックして、ロックを削除します。プロセスが保持するすべてのロックを削除するか、リモート・クライアントからのすべてのロックを削除します。削除するロックが開いているトランザクションの一部の場合、削除を確認する前に警告が表示されます。

通常、アプリケーションに問題が生じた場合のみ、ロックを削除する必要があります。

LOCK コマンドとその機能の詳細は、"ObjectScript リファレンス" の "LOCK" エントリを参照してください。

システムで多数のロックを使用する場合、状況によってはロック・テーブルのサイズを大きくする必要があります。これは、管理ポータルで行うことができます。手順については、"構成パラメータ・ファイル・リファレンス" の "locksiz" のエントリを参照してください。

ロック管理の詳細および代替手法については、"ObjectScript の使用法" の "ロック管理" の章を参照してください。

1.4 InterSystems IRIS ログの監視

InterSystems IRIS では、さまざまな観点から処理を監視するために以下のログを使用できます。

- いくつかのログ・ファイルは install-dir¥mgr ディレクトリにあります。そのうちの 2 つは、管理ポータルを使用して表示できます。
- アプリケーション・エラー・ログと xDBC エラー・ログは、管理ポータルを使用して表示できます。
- InterSystems IRIS システム・エラー・ログ (syslog) の内容は、複数の方法のいずれかを使用して確認できます。

構造化ログを有効にすることもできます。構造化ログでは、messages.log と同じメッセージを機械で判読可能なファイルに書き込み、選択した監視ツールで取得できます。"構造化ログの設定" を参照してください。

1.4.1 install-dir¥mgr ディレクトリのログ・ファイル

以下のログ・ファイルは、install-dir¥mgr ディレクトリにあります。これらはプレーン・テキスト・ファイルとして保存され、任意のテキスト・エディタまたはビューワで表示できます。メッセージ・ログとシステム・モニタ・ログは、管理ポータルを使用して表示できます。

アラート・ログ

ログ・モニタはメッセージ・ログを定期的にスキャンして、構成された最小深刻度のエントリを検出し、対応する通知を生成します。この通知は、既定ではアラート・ログ (install-dir¥mgr¥alerts.log) に書き込まれます。代わりに、電子メール通知を送信するようにログ・モニタを構成することもできます。詳細は、このドキュメントの“ログ・モニタの使用”の章を参照してください。

初期化ログ

初期化ログ **iboot.log** には、InterSystems IRIS インスタンスの初期化に関する情報が含まれています。

ジャーナル履歴ログ

ジャーナル履歴ログ **journal.log** は、InterSystems IRIS インスタンスによって管理されるすべてのジャーナル・ファイルのリストを含んでおり、ジャーナル関連の機能、ユーティリティ、および API によってジャーナル・ファイルを見つけるために使用されます。ジャーナリングの詳細は、″データ整合性ガイド″ の“ジャーナリング”の章を参照してください。

メッセージ・ログ

InterSystems IRIS は、さまざまなメッセージをメッセージ・ログ・ファイル (**messages.log**) にレポートします。これには、一般的なメッセージ、起動/シャットダウン、ライセンス、およびネットワーク・エラー、特定のオペレーティング・システムのエラー、他のシステムからリモートで開始されたジョブの成功または失敗などが含まれます。システム・モニタも、メッセージ・ログへの通知の書き込みを行います。**messages.log** のディレクトリを構成できますが (″構成パラメータ・ファイル・リファレンス″ の ″コンソール″ を参照)、既定の場所は install-dir**¥mgr** です。

Windows ベースのプラットフォームでは、すべてのコンソール・メッセージがメッセージ・ログ・ファイル **messages.log** に送信されます。UNIX®/Linux プラットフォームでは、コンソール・メッセージを、メッセージ・ログ・ファイル、コンソール・ターミナル、またはこれら両方に送信するように構成できます。

messages.log ファイルのサイズは、システム・モニタによって監視されます。ファイルは構成された最大サイズに達するまで増大し、最大サイズに達すると、InterSystems IRIS はファイルを保存して新しいファイルを開始します。メッセージ・ログの最大サイズの構成の詳細は、″構成パラメータ・ファイル・リファレンス″ の ″MaxConsoleLogSize″ を参照してください。

メッセージ・ログは、管理ポータルの **[メッセージログを表示]** ページ (**[システム処理]** > **[システムログ]** > **[メッセージログ]**) から表示できます。メッセージ・ログが 1 MB よりも大きくなった場合は、最新の 1 MB の部分のみが管理ポータルに表示されます。**[ファイル全体を表示]** リンクをクリックして、ファイル全体を表示します。ファイルが非常に大きい場合、表示に時間がかかる場合があります。

注釈　InterSystems IRIS の起動で問題が発生した場合には、任意のテキスト・エディタまたはテキスト・ビューワを使用して、メッセージ・ログを参照してください。

システム・モニタ・ログ

システム・モニタの機能に関するステータス・メッセージ (このドキュメントの“システム・モニタの使用”の章を参照) は、システム・モニタ・ログ (install-dir**¥mgr¥SystemMonitor.log**) に書き込まれます。

SystemMonitor.log ファイルのサイズは、システム・モニタによって監視されます。このファイルは最大サイズの 5 MB に達するまで増大すると、**SystemMonitor.log.old** という名前に変更され、既存の **SystemMonitor.log.old** ファイルが上書きされて、新しい **SystemMonitor.log** が作成されます。したがって、システム・モニタ・ログで使用される最大 MB 数は 10 MB となります。

メッセージ・ログは、管理ポータルの **[システム・モニタ・ログ]** ページ (**[システム処理]** > **[システムログ]** > **[システム・モニタ・ログ]**) から表示できます。システム・モニタ・ログが 1 MB よりも大きくなった場合は、最新の 1 MB の部分のみが管理ポータルに表示されます。**[ファイル全体を表示]** リンクをクリックして、ファイル全体を表示します。ファイルが非常に大きい場合、表示に時間がかかる場合があります。

1.4.2 アプリケーション・エラー・ログと xDBC エラー・ログ

[**アプリケーションエラーログの表示**] ページ ([**システム処理**] > [**システムログ**] > [**アプリケーションエラーログ**]) では、アプリケーション・エラーを表示できます。

同様に、[**xDBC エラー・ログ**] ページ ([**システム処理**] > [**システムログ**] > [**xDBC エラー・ログ**]) では、ODBC エラーと JDBC エラーを表示できます。

1.4.3 InterSystems IRIS システム・エラー・ログ

InterSystems IRIS は、有意義な項目を記録するために、共有メモリの一部分を確保します。このテーブルは、さまざまな名前 (InterSystems IRIS システム・エラー・ログ、errlog、SYSLOG、syslog テーブルなど) で表されますが、重要な診断情報が含まれていることがあります。

既定では、システム・エラー・ログには、最新 500 件のログ項目が収められています。システム・エラー・ログの項目数の構成に関する詳細は、"構成パラメータ・ファイル・リファレンス" の "errlog" を参照してください。

システム・エラー・ログは、以下のいずれかの方法で表示します。

- ターミナルを開き、set $namespace = "%SYS" と入力して **%SYS** ネームスペースに切り替え、do ^SYSLOG と入力します。また、do FILTER^SYSLOG と入力してもかまいません。このコマンドには、特定のエラー・コードまたはプロセス ID に基づいて出力を制限するオプションがあります。
- 診断レポートを実行します。詳細は、このドキュメントの “診断レポートの使用法” の章を参照してください。
- irisstat コマンドに -e1 オプションを指定して実行します。このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” にある "オプションを使用した irisstat の実行" を参照してください。
- IRISHung スクリプトを実行します。付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” にある "IRISHung スクリプト" を参照してください。

ShutDownLogErrors 設定を使用して、シャットダウン時にシステム・エラー・ログをメッセージ・ログに書き込むように InterSystems IRIS を構成できます ("構成パラメータ・ファイル・リファレンス" の "ShutDownLogErrors" を参照)。

2

InterSystems 診断レポートの使用法

インターシステムズは、InterSystems IRIS® データ・プラットフォーム・インスタンスに対して診断レポートを実行するメカニズムを提供しています。診断レポートは、実行されているインスタンスに関する情報のスナップショットで、インターシステムズのサポート窓口 (WRC) に送られ、システムの問題を診断する際の助けとなります。収集される情報の種類の詳細は、"診断レポートの内容" のセクションを参照してください。

このトピックでは、管理ポータルからのタスクとして、診断レポートを構成し、実行する方法について説明します。このタスクに関する詳細は、"インターシステムズ・クラス・リファレンス" の "%SYS.Task.DiagnosticReport" エントリを参照してください。

2.1 診断レポートのタスクの実行

レポート生成の最も直接的な方法は、管理ポータルの **[診断レポート]** ページ (**[システム処理]** > **[診断レポート]**) に移動して、診断レポート・タスクのための適切な情報を入力することです。このページに戻って、いつでもこの情報を編集できます。フィールドを編集しない場合は、**[実行]** をクリックして、現在の設定を使用してレポートを生成します。

情報を何も入力せずに **[実行]** をクリックすると、タスクによって詳細レポートが生成され、InterSystems IRIS インスタンスの管理者のディレクトリ (install-dir**¥mgr**) に HTML ファイルとして配置されます。ファイル名は、顧客名 YYYYMMDDHHMM.**html** の形式です。

例えば、2019 年 9 月 24 日 8:46 p.m. に、**C:¥MyInstallDir** にインストールされたインスタンスで MyCompany に発行されたライセンス・キーを使用して診断レポート・タスクを実行すると、以下のレポート・ファイルが生成されます。

C:¥MyInstallDir¥mgr¥MyCompany201909242046.html

いつタスクを実行するか、どこにファイルを保存するか、およびファイルを WRC に送信するかどうかに影響するいくつかのフィールドをページ上で設定できます。"診断レポートの設定の構成" のセクションでこれらの設定について説明します。**[閉じる]** をクリックすると、変更が破棄され、レポートのタスクは実行されません。

診断レポート・タスクの履歴の表示

[診断レポート] ページの最上部にある **[タスク履歴]** をクリックすると、診断レポート・タスクの履歴が表示されます (タスクとタスク履歴の詳細は、"システム管理ガイド" の “InterSystems IRIS の管理” の章にある "タスク・マネージャの使用" を参照してください)。

2.2 診断レポートの設定の構成

InterSystems IRIS インストールには、事前に定義されたオンデマンド診断レポート・タスクが含まれています。初めて **[診断レポート]** ページを開いたときに、適切な情報を入力して、このタスクの設定を更新します。入力するフィールドによっては、診断レポートを処理するための以下の選択肢が提示されます。

1. マネージャのディレクトリ以外の特定のアーカイブ・ディレクトリにレポートを保存するには、ディレクトリ名を入力します。
2. レポートを WRC に送信するには、送信メールのフィールドに情報を入力します。
3. レポートを保存して、送信するには、前の 2 つのオプションに情報を入力します。
4. 定期スケジュールで自動的にレポートを実行するには、WRC HealthCheck を有効にします。

以下のリストは、診断レポートの設定および各設定の説明を示しています。

- **アーカイブ・レポート用のディレクトリ** – レポートを保存するための場所です。ページに情報を何も入力しないと、マネージャのディレクトリである install-dir**¥mgr** が既定ディレクトリとなります。この設定を空白のままにして、送信メール設定を入力すると、レポートは、マネージャのディレクトリには保存されません。**[参照]** をクリックして、既存のディレクトリを選択します。

WRC にレポートを直接送信するために必要な情報 送信メール設定を入力すると、レポートは、WRCHealthCheck@InterSystems.com に送信されます。

- **既存の WRC 問題の番号** – 診断レポートのこの実行に関連する WRC 問題の番号 (6 桁)。新しい問題を追加するには、インターシステムズのサポート窓口までお問い合わせください、または、問題を WRC Direct で入力してください。

  この WRC 問題の番号でタスクが一度だけ実行され、その後は、この設定がクリアされます。
- **送信メールのサーバの IP アドレスの名前** – 送信 SMTP (Simple Mail Transfer Protocol) メール・サーバのアドレス。
- **認証 SMTP のユーザ名**および**パスワード** – SMTP サーバの SMTP 認証でのみ必要です。詳細は ″RFC 2554″ を参照してください。
- **送信メールの [送信元:] フィールドのアドレス** – 送信者フィールドに表示する電子メール・アドレス。SMTP サーバ情報を入力する場合は必須です。既定では、DefaultDiagnosticReport@InterSystems.com です。
- **送信メールの [返信先:] フィールドのアドレス** – インターシステムズからの自動構成メッセージを受信できる企業内の有効な電子メール・アドレス。
- **送信メールの [CC:] フィールドのアドレス** – レポートを受信するその他の電子メール・アドレス。
- **WRC HealthCheck の自動更新を有効にする** – 定期レポートを WRC に送信するには、このチェック・ボックスにチェックを付けます。インターシステムズでは、WRC HealthCheck 機能を有効にすることを強くお勧めします。このオプションを選択すると、診断レポート・タスクが定期的に実行され、レポートが WRC に送信されます。これらの定期レポートによって、WRC はより良いサポートを提供できるようになります。この機能を選択すると、SMTP サーバ情報の入力が必要になります。

  重要　　レポートのタスクによって個人のアプリケーション情報が送信されることはありません。インターシステムズでは、すべての構成データの機密性が厳密に保持されます。
- この日数ごとにこの時間に WRC HealthCheck の自動更新を実行する – WRC HealthCheck を有効にすると、タスク・マネージャは、診断レポートを実行するときの間隔 (既定は 7 日間) および時間 (既定は InterSystems IRIS のインストール時間) の情報を保存します。

WRC に送信するその他の情報

- **このインスタンスの主な目的** ― InterSystems IRIS のこのインスタンスを開発、テスト、品質保証、運用のいずれを目的として使用するのかを選択します。
- **$ZV に表示されない適用済みの任意のコンテンツ** ― $ZVersion 特殊変数に表示されない適用済みの任意のコンテンツを入力します。
- **現在の CPU のタイプと数**
- **物理メモリ総量** ― マシンの物理メモリ量を入力します。
- **このシステムで使用するハードウェアのその他の詳細**
- **このシステム (InterSystems、OS、外部、その他) をバックアップする方法** ― システムをバックアップするために使用する方法を入力します。
- **このインスタンスについてのその他の関連情報** ― レポートに追加する特別な注記事項があれば入力します。

診断レポートのタスクでは、1 つを除くすべての設定で入力した情報が保持されます。タスクは、WRC 問題の番号で一度だけ実行された後、WRC 問題の番号はクリアされます。レポート実行中は、タスクの設定を編集できません。

2.3 診断レポートの内容

診断レポートのタスクを実行すると、基本情報と詳細情報の両方を記録した HTML 形式のログ・ファイルが作成されます。このファイルは、問題解決のために WRC が使用できます。以下では、このレポートの各セクションについて説明します。

- 基本情報
- 詳細情報

注釈　Microsoft Windows 32 ビット・システムの場合、レポートでは、SysInternals Software が開発した以下のサードパーティ・ユーティリティが使用されます。

- PsInfo.Exe ― 拡張システム情報を表示します。
- PsList.Exe ― オペレーティング・システム・レベルのプロセス情報を表示します。

2.3.1 基本情報

基本情報には、以下のカテゴリがあります。

一般

以下の情報が表示されます。

- 完全なホスト名 (ドメインを含む)
- IP アドレス
- ユーザ名
- レポート作成日時
- InterSystems IRIS のバージョンの文字列 ($ZVersion)
- InterSystems IRIS オブジェクトのバージョンの文字列

- InterSystems ODBC/JDBC サーバのバージョン情報
- 各国言語サポート (NLS) 情報
- フリー・ブロック・カウント情報
- オペレーティング・システムのバージョン (UNIX® システムの uname -a)
- 拡張システム情報 (**PsInfo.Exe** ユーティリティが、InterSystems IRIS **Bin** ディレクトリにある場合、Windows システムでのみ)

キー・ファイル

ライセンス・キー・ファイルの場所、ライセンス・キーの内容、ライセンスの可用性 ($System.License.CKEY() の出力) などのアクティブなライセンス情報を表示します。

ライセンス・カウント

ライセンス使用情報 ($System.License.ShowCounts() の出力) を表示します。

%SS

システムの状態の情報 (^%SS の出力 – 30 秒の間隔をおいて取られた 2 つのスナップショット) を表示します。

オペレーティング・システム・プロセス・リスト

オペレーティング・システム・プロセス情報を表示します (**PsList.Exe** ユーティリティが InterSystems IRIS **Bin** ディレクトリにある場合、Windows システムでのみ)。

スピン・カウント

スピン・カウント情報を表示します。

CPF ファイル

アクティブな InterSystems IRIS 構成ファイル (**iris.cpf**) の内容を表示します。

SysLog

InterSystems IRIS システム・エラー・ログの内容を表示します。詳細は、“管理ポータルを使用した InterSystems IRIS の監視” の章にある ″InterSystems IRIS システム・エラー・ログ″ を参照してください。

セキュリティ

以下のセキュリティ情報のリストを表示します。

- セキュリティ・パラメータ
- サービス
- リソース
- ロール
- アプリケーション
- システム・ユーザ
- 現在のログインの失敗
- ドメイン
- SSL 構成

監査

イベントのリストや監査ログ・データベースの内容などの監査情報を表示します。

メッセージ

messages.log の内容を表示します (サイズが 5 MB 以下の場合)。

注釈　基本情報のみを記録したレポートを生成する手順は次のとおりです。

1. **[タスクスケジュール表示]** ページ (**[システム処理]** > **[タスクマネージャ]** > **[タスクスケジュール表示]**) に移動します。
2. 診断レポートの行で、**[詳細]** をクリックします。
3. **[タスク詳細]** ページ (**[システム処理]** > **[タスクマネージャ]** > **[タスクスケジュール表示]** > **[タスク詳細]**) で、**[編集]** をクリックします。
4. **[タスクスケジューラウィザード]** ページで、**[拡張レポート]** チェック・ボックスのチェックを外し、**[完了]** をクリックします。
5. **[タスク詳細]** ページ (**[システム処理]** > **[タスクマネージャ]** > **[タスクスケジュール表示]** > **[タスク詳細]**) の診断レポートの行で **[実行]** をクリックします。
6. **[タスク実行]** ページで、**[すぐに実行する]** をクリックします。
7. **[閉じる]** をクリックします。

2.3.2 詳細情報

詳細情報には、以下のカテゴリがあります。

irisstat Snapshot #1

以下のオプションを使用して実行された InterSystems 統計ユーティリティ (irisstat) の出力を表示します。

```
irisstat -e2 -m-1 -n3 -j5 -g1 -m3 -L1 -u-1 -v1 -p-1 -c-1 -q1 -w2 -S-1 -E-1 -N65535 -s<mgr_dir>
```

irisstat ユーティリティの詳細は、このドキュメントの付録 "irisstat ユーティリティを使用した InterSystems IRIS の監視" を参照してください。

irisstat Snapshot #2

1 分後に最初のスナップショットと同じオプションを使用して実行された irisstat ユーティリティの出力を表示します。

irisstat 出力ファイルが大きすぎる場合は、個別のファイルに保存され、レポートと共に送信されることはありません。個別のファイルが作成されると、以下のようなメッセージが診断レポートの irisstat セクションに通知されます。

```
File /iris/iristestsys/mgr/irisstat201103151102.html is too big to be appended to
the Log File. A copy has been left in the Directory.
```

それらのファイルは html 拡張子を持ちますが、プレーン・テキストなのでブラウザではなくテキスト・エディタで表示する必要があります。

ネットワークのステータス

ネットワーク情報を表示します。以下のユーティリティの出力です。

- ipconfig /all (Windows システムのみ)

- netstat -an
- netstat -s

ダンプ・ライセンス

ローカル・ライセンス・テーブルのエントリおよびキー情報を表示します (`$System.License.DumpLocalInUse()` および `$System.License.DumpKeys()` の出力)。

マネージャのディレクトリのダンプ・ファイル

コアまたは *.dmp ファイル (存在する場合) のリストを表示します。

GloStat

グローバル統計情報 (`GLOSTAT の出力 ー 10 秒ごとに取られる 10 枚のスナップショット) を表示します。

3
ログ・モニタの使用

ログ・モニタは、InterSystems IRIS® Data Platform インスタンスのメッセージ・ログを監視して、InterSystems IRIS デーモンおよびユーザ・プロセスによってレポートされたエラーおよびトラップを検出し、対応する通知を生成します。また、電子メールが構成されている場合は、電子メールを生成します。ログ・モニタは、^MONMGR ユーティリティを使用して管理できます。

3.1 システム監視ツール

InterSystems IRIS には、以下のような InterSystems IRIS インスタンスの一般的監視用の 3 つのツール・セットが用意されています。

- 管理ポータルには、さまざまなシステム・インジケータ、システム・パフォーマンス、InterSystems IRIS ロック、エラー、およびトラップを監視できる、複数のページとログ・ファイルがあります。詳細は、このドキュメントの“管理ポータルを使用した InterSystems IRIS の監視”の章を参照してください。これらの中で、メッセージ・ログは最も包括的なもので、一般的なメッセージ、起動/シャットダウン、ライセンス、ネットワーク・エラー、特定のオペレーティング・システム・エラー、他のシステムからリモートで起動されたジョブの成功または失敗のインジケータに加えて、システム・モニタからのアラート、警告、およびメッセージが含まれます。
- この章で説明するログ・モニタは、構成された最小深刻度のメッセージ・ログ・エントリの通知を生成し、その通知をアラート・ログに書き込むか、指定された受信者に電子メールで送信します。これにより、すべてのタイプのメッセージ・ログのアラートを抽出してシステム・オペレータに知らせることができます。^MONMGR ユーティリティを使用してログ・モニタを構成できます。
- システム・モニタは、重要なシステム・ステータス・インジケータおよびリソース使用量インジケータに関連するアラートおよび警告を生成します。また、システム定義メトリックおよびユーザ定義メトリックを監視し、異常値を検出した場合にアラートおよび警告を生成するアプリケーション・モニタとヘルス・モニタが組み込まれています。システム・モニタおよびヘルス・モニタのアラートと警告はメッセージ・ログに書き込まれます。アプリケーション・モニタのアラートは、電子メールで送信したり、指定した通知方法で受け渡しできます。システム・モニタ (アプリケーション・モニタおよびヘルス・モニタを含む) は、^%SYSMONMGR ユーティリティを使用して管理できます。システム・モニタ、アプリケーション・モニタ、およびヘルス・モニタの使用に関する詳細は、このドキュメントの“システム・モニタ”の章を参照してください。

3.2 ログ・モニタの概要

ログ・モニタは、メッセージ・ログを定期的にスキャンして、構成された深刻度レベルのエントリを検出し、対応する通知を生成します。これらの通知は、アラート・ログに書き込まれるか、指定された受信者に電子メールで送信されます。

メッセージ・ログには、一般的なメッセージからエラーやトラップ、システム・モニタのアラートや警告まで、InterSystems IRIS インスタンスに関する有用な情報が含まれます。メッセージ・ログの内容に基づいて通知を生成することによって、ログ・モニタはシステム・オペレータのアラートの可視性を高めます。

注釈　ログ・モニタは、構成された深刻度のすべてメッセージ・ログ・エントリに対して通知を生成するわけではありません。約 1 時間以内に指定されたプロセスから連続したエントリがある場合、通知は最初のエントリに対してのみ生成されます。このため、ログ・モニタから 1 通の通知を受け取ったら、すぐにメッセージ・ログを確認（および該当する場合はシステム・モニタのアラートを表示）する必要があります。ただし、"ログ・モニタのエラーおよびトラップ" にリストされたメッセージ・ログ・エントリの場合、常に通知が生成されます。

既定では、ログ・モニタは以下の設定で動作します。

- ログ・モニタは、インスタンスの稼働中は常に稼働しています。
- メッセージ・ログは 10 秒ごとにスキャンされます。
- 通知は、深刻度 2（重大）および深刻度 3（致命的）のメッセージ・ログ・エントリに対して生成されます。
- 通知はアラート・ログに書き込まれます。

  注釈　ログ・モニタは、初めて通知を生成するときにアラート・ログを作成します。アラート・ログ（alerts.log）は <install-dir>/mgr ディレクトリにあります。

以下のセクションで説明するように、インタラクティブな ^MONMGR ユーティリティを使用して、ログ・モニタを構成できます。

3.3 ログ・モニタの構成

ログ・モニタ・マネージャ・ユーティリティ ^MONMGR を使用すると、ログ・マネージャを構成および管理できます。ログ・モニタを停止および起動し、デフォルト設定を変更し、電子メール通知を構成できます。

ログ・モニタ・マネージャを起動するには、以下の手順に従います。

1. ターミナルで、以下のコマンドを入力します。^MONMGR は、%SYS ネームスペースで実行する必要があります。

```
%SYS>do ^MONMGR
```

2. メイン・メニューが表示されます。実行する操作の番号を入力します。ログ・モニタ・マネージャを終了するには Enter キーを押します。

```
1) Start/Stop/Update MONITOR
2) Manage MONITOR Options
3) Exit

Option?
```

メイン・メニューのオプションでは、以下のテーブルで説明するようにログ・モニタを管理できます。

オプション	説明
1) Start / Stop / Update Monitor	ログ・モニタおよびアラート・ログを管理するための [Start/Stop/Update Monitor] サブメニューが表示されます。
2) Manage MONITOR Options	ログ・モニタの通知オプション（サンプリング間隔、深刻度レベル、電子メール）を管理するための [Manage Monitor Options] サブメニューが表示されます。
3) Exit	ログ・モニタ・マネージャを終了します。

3.3.1 Start/Stop/Update Monitor

このサブメニューでは、ログ・モニタ・マネージャの動作を管理できます。実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 1

1) Update MONITOR
2) Halt MONITOR
3) Start MONITOR
4) Reset Alerts
5) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルで説明するようにログ・モニタの動作を管理できます。

オプション	説明
1) Update MONITOR	[Manage Monitor Options] の現在の設定（間隔、深刻度レベル、電子メール）に基づいて、ログ・モニタを動的に再起動します。
2) Halt MONITOR	ログ・モニタを停止します。ログ・モニタが稼働していないと、メッセージ・ログはスキャンされません。
3) Start MONITOR	ログ・モニタを起動します。[Manage Monitor Options] の現在の設定（間隔、深刻度レベル、電子メール）に基づいて、メッセージ・ログが監視されます。
4) Reset ALERTS	アラート・ログを削除します（アラート・ログが存在する場合）。
5) Exit	メイン・メニューに戻ります。

3.3.2 Manage Monitor Options

このサブメニューでは、ログ・モニタのスキャンおよび通知のオプションを管理できます。実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 2

1) Set Monitor Interval
2) Set Alert Level
3) Manage Email Options
4) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルで説明するようにログ・モニタの動作を管理できます。

オプション	説明
1) Set Monitor Interval	メッセージ・ログをスキャンする間隔を変更できます。既定値の 10 秒を超えない間隔にすることをお勧めします。

オプション	説明
2) Set Alert Level	以下のように、通知を生成するメッセージ・ログ・エントリの深刻度レベルを設定できます。 · 1 – 警告、重大、および致命的 · 2 – 重大および致命的 · 3 – 致命的のみ
3) Manage Email Options	[Manage Email Options] サブメニューを使用して、ログ・モニタの電子メール通知を構成できます。
4) Exit	メイン・メニューに戻ります。

注釈　ログ・モニタは、約 1 時間以内に指定プロセスから連続したメッセージ・ログ・エントリがあった場合、最初のエントリに対してのみ通知を生成するため、アラート・レベルを 1 に設定した場合、警告によってアラート・ログ・エントリまたは電子メール・メッセージが生成されると、同じプロセスからの深刻度 2 の後続アラートでは通知は生成されません。

3.3.3 Manage Email Options

このサブメニューのオプションでは、電子メールの構成および有効化と無効化が可能です。電子メールが有効になっている場合、ログ・モニタは通知を電子メールで送信します。電子メールが無効になっている場合は、通知はアラート・ログに書き込まれます。実行する操作の番号を入力します。[Manage Monitor Options] サブメニューに戻るには Enter キーを押します。

```
Option? 3

1) Enable/Disable Email
2) Set Sender
3) Set Server
4) Manage Recipients
5) Set Authentication
6) Test Email
7) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルで説明するようにログ・モニタの電子メール通知を管理できます。

オプション	説明
1) Enable / Disable Email	電子メールを有効にすると、ログ・モニタは以下の動作を行います。 · 現在アラート・ログに項目がある場合、各項目の電子メール通知を送信します · alerts.log ファイルがある場合、これを削除します · その時点から、構成された深刻度のメッセージ・ログ・エントリに対する電子メール通知を送信します 電子メールを無効にすると、ログ・モニタはエントリをアラート・ログに書き込みます。 注釈　電子メールを有効にしても無効にしても、他の電子メール設定には影響しません。つまり、電子メール・オプションを再構成する必要はありません。

オプション	説明
2) Set Sender	このオプションを選択して、電子メールの送信者を示すテキストを入力します (Log Monitor など)。入力するテキストが有効な電子メール・アカウントを表している必要はありません。- (ダッシュ) を入力すると、このフィールドを NULL に設定できます。
3) Set Server	このメニュー項目を選択して、サイトの電子メールを処理する電子メール・サーバの名前およびポート番号 (既定は 25) を入力します。IT 担当者に問い合わせて、この情報を入手してください。- (ダッシュ) を入力すると、このフィールドを NULL に設定できます。
4) Manage Recipients	このオプションでは、各通知の送信先の電子メール・アドレスの表示、追加、または削除が可能な以下のサブメニューが表示されます。 注釈　有効な各電子メール・アドレスを個別に追加する必要があります。[2) Add Recipient] を選択した場合は、[Email Address?] プロンプトに応答するときに複数のアドレスを入力しないでください。
5) Set Authentication	電子メール・サーバが認証を必要とする場合、認証のユーザ名とパスワードを指定できます。IT 担当者に問い合わせて、この情報を入手してください。エントリを指定しないと、認証のユーザ名とパスワードは NULL に設定されます。- (ダッシュ) を入力すると、[ユーザ] フィールドを NULL に設定できます。
6) Test Email	指定された電子メール・サーバを使用して、指定された受信者にテスト・メッセージを送信します。
7) Exit	[Manage Monitor Options] サブメニューに戻ります。

3.4 ログ・モニタのエラーおよびトラップ

以下のメッセージ・ログ・エラーが発生した場合は、常にログ・モニタ通知が生成されます。

- セグメント違反 (アクセス違反) が発生したため、処理を停止しました。
- データベース % が <FILEFULL> です。
- 監査:エラー:監査データベースを '% に変更できませんでした。'% を監査しています。
- 監査:エラー:監査データベースを '% に設定できませんでした。
- sfn # の展開中、同期に失敗しました。新しいマップは追加されません。
- sfn # の展開中、同期に失敗しました。一部のブロックが追加されません。
- WRTDMN による wdqlist の割り当てに失敗しました。システムをフリーズします。
- WRTDMN:CP は既に終了しています。システムをフリーズします。
- ライト・デーモンに深刻なエラーが発生したため、システムをフリーズしました。
- グローバル・バッファが不足しています。WRTDMN がパニック・モードになっています。
- WRTDMN パニック:SFN x ブロック y がデータベースに直接書き込まれました。
- 予期しない書き込みエラー:dkvolblk が %d を返しました (% のブロック #%d)。
- 予期しない書き込みエラー:dkswrite が %d を返しました (% のブロック #%d)。
- 予期しない書き込みエラー:%d (% のブロック #%d)。

- クラスタ・クラッシュ – すべてのキャッシュ・システムが停止しました。
- トランザクションが開いているか、ECP がその状態を保存できないため、システムを正常にシャットダウンできません。
- 重大なジャーナル・エラー：JRNSTOP が % を開くことができません。＊ジャーナル処理を停止しますが、一部のジャーナル・データが失われている可能性があります。
- ジャーナル変換テーブルにメモリを割り当てることができません。
- ジャーナル・ファイルが最大サイズ (%u バイト) に達し、自動ロールオーバーが失敗しました。
- ジャーナル・ファイルへの書き込みに失敗しました。
- 最新のジャーナル・ファイルを開けません。
- ジャーナル・ファイルの同期に失敗しました。
- %d 秒経過するか、またはジャーナル・バッファがいっぱいになった時点で、ジャーナル処理が無効になります。ジャーナル・データの潜在的な損失を防止するには、(“管理ポータルを使用した InterSystems IRIS の監視” の章にある ″InterSystems IRIS システム・エラー・ログ″ の説明に従って InterSystems IRIS システム・エラー・ログを調べて) エラーの原因を解決するか、新しいデバイスに切り替えてジャーナリングを実行します。
- ジャーナルのログにエラーが発生しました。
- 展開後、属性の読み取りでジャーナル・エラーが発生しました。
- ECP クライアント・デーモン/接続が中断されました。
- クラスタ・フェールソフトに失敗しました。エラーが発生したシステムの locksysid を特定できません。すべてのクラスタ・システムが停止されます。
- enqpijstop に失敗しました。クラスタがクラッシュします。
- enqpijchange に失敗しました。クラスタがクラッシュします。
- WIJ 処理中にエラーが発生しました。システムがクラッシュします。
- PIJ 処理中にエラーが発生しました。システムがクラッシュします。
- ブロック読み取りエラー – 読み取りエラーをリカバリします。
- ブロック書き込みエラー – 書き込みエラーをリカバリします。
- WIJ 展開エラー：システムのフリーズ－WIJ 展開エラーが長時間におよんだため、システムがフリーズしました。WIJ 用の領域が作成された場合は、システムが再開されます。それ以外の場合は、irisforce を使用してシステムをシャットダウンする必要があります。
- CP：デーモン終了のモニタを作成できません。
- CP：WRTDMN による %d の受け渡しが %d 秒におよんでいるため、システムをフリーズしました。WRTDMN が受け渡しを完了した時点でシステムを再開します。
- WRTDMN：CP のハンドルを開く前に CP でエラーが発生したため、システムがフリーズしています。
- WRTDMN：CP モニタのハンドルを取得する際にエラー・コード %d が発生しました。CP はモニタされません。
- WRTDMN：プロセス制御でエラーが発生 (終了コード %d) したため、システムがフリーズしています。
- CP：デーモンでエラーが発生 (終了コード %d) したため、システムがフリーズしています。
- オペレーティング・システムがシャットダウンされたため、Cache の緊急シャットダウンを実行しています。
- CP：すべてのプロセスでエラーが発生しました。システムをフリーズします。
- irisforce がすべてのプロセスを終了できませんでした。
- 予備ライト・デーモンを開始できませんでした。

- 理由 # により ENQDMN を終了します。
- プライマリ・ミラー・サーバになります。

4

システム・モニタの使用

システム・モニタは、柔軟でユーザ拡張可能なユーティリティです。その用途としては、InterSystems IRIS® Data Platform インスタンスを監視して、さまざまなメトリックの 1 つまたは複数の値が潜在的な問題を示した場合に通知を生成することです。システム・モニタには、以下の InterSystems IRIS インスタンス監視ツールが付属しています。

- システム・モニタは、システムの状態およびリソースを監視し、固定パラメータに基づいて通知 (アラートおよび警告) を生成し、システム・ヘルス全体を追跡します。
- ヘルス・モニタは、主要なシステム・メトリックおよびユーザ定義メトリックをサンプリングし、それらをユーザ構成可能パラメータおよび規定の通常値と比較して、サンプルが該当しきい値を超えた場合に通知を生成します。
- アプリケーション・モニタは、重要なシステム・メトリックをサンプリングし、その値をローカル・ネームスペースに格納して、ユーザが作成したアラート定義を使用して評価します。アラートがトリガされると、電子メール通知を生成するか、指定されたクラス・メソッドを呼び出します。

既定では、これら 3 つのツールはすべて %SYS ネームスペースで動作します。システム・モニタおよびアプリケーション・モニタは、必要に応じて、他のネームスペースでネームスペース固有の構成および設定で実行することができます。要件に合わせて独自のコンポーネントを定義および構成して、各ネームスペースのシステム・モニタの機能を拡張することができます。

一般的な InterSystems IRIS インスタンス監視ツールの概要については、このドキュメントの “ログ・モニタの使用” の章の ″システム監視ツール″ を参照してください。また、メッセージ・ログ内の通知 (システム・モニタによって生成されるものを含む) から電子メール・メッセージを生成するようにログ・モニタを構成する方法については、同じ章の ″Manage Email Options″ を参照してください。この章で取り上げるログ・ファイルの詳細は、″ログ・ファイルの監視″ を参照してください。

4.1 システム・モニタ

システム・モニタは、重要なシステム・ステータス・インジケータおよびリソース使用量インジケータ (ECP 接続のステータスや使用中のロック・テーブルの割合など) をサンプリングして、固定されたステータスおよびしきい値に基づいて、通知 (アラート、警告、および “ステータス OK” メッセージ) を生成します。これらの通知は、メッセージ・ログに書き込まれます。これにより、ログ・モニタはこれらから電子メール・メッセージを生成できます (そのように構成されている場合)。また、システム・モニタは、単一のシステム・ヘルス状態全体も保持します。

システム・モニタは、^%SYSMONMGR を使用して管理されます。

このセクションの残りの部分では、以下の項目について説明します。

- システム・モニタ・プロセス
- システム・モニタ通知の追跡

- システム・モニタのステータスおよびリソース・メトリック
- システム・モニタのヘルス状態
- システム・モニタの既定値
- ^%SYSMONMGR ユーティリティの使用
- システム・モニタのコンポーネントの定義

4.1.1 システム・モニタ・プロセス

システム・モニタは、実行場所として構成されている各ネームスペースで、3 つのタイプのクラス (つまり、システム・モニタのコンポーネント) を使用して、以下のように 3 段階でシステム・メトリック情報の収集および配信を実行します。センサ・クラスは情報を収集し、サブスクライバ・クラスは情報を評価して通知を生成し、通知クラスは通知を適切なアラート・システムに送信します。以下に、その順序について詳しく説明します。

1. メトリック情報の取得

   センサ・クラスには、システム・メトリックまたはアプリケーション・メトリックの値を取得するためのメソッドが組み込まれています。例えば、システム・センサ・クラス SYS.Monitor.SystemSensors には、GetProcessCount() メソッド (InterSystems IRIS インスタンスのアクティブなプロセスの数を返す) と GetLockTable() メソッド (インスタンスのロック・テーブルのうち使用中の割合を返す) が含まれています。

   システム・モニタは、構成されている各センサ・クラスの GetSensors() メソッドを定期的に呼び出します。センサ・クラスは、以下のいずれかを実行します。

   - システム・モニタによってサブスクライバ・クラスに渡されるセンサの名前と値の組み合わせの配列を返します (第 2 段階を参照)。
   - 取得したセンサの値を評価して、システム・モニタによって通知クラスに送信される通知を返します (第 3 段階を参照)。

   システム・モニタに付属しているセンサ・クラスの 1 つである SYS.Monitor.SystemSensors が名前と値の配列を返します。もう 1 つのセンサ・クラス %SYS.Monitor.AppMonSensor がそれ自体の評価を実行して、それ自体の通知を生成します。

2. メトリック情報の評価

   サブスクライバ・クラスには、センサの値を評価し、通知を生成するためのメソッドが組み込まれています。システム・モニタは、名前と値の配列を返す各センサ・クラスを呼び出した後に、各サブスクライバ・クラスの Receive() メソッドを呼び出して、SensorReading プロパティにその配列を入力します。サブスクライバ・クラスは、Receive() メソッドに指定されたセンサの名前と値の組み合わせごとに値を評価し、必要に応じてテキストおよび深刻度コードが含まれた通知を返します。

   例えば、システム・モニタが、SYS.Monitor.SystemSensors.GetSensors() から返された値と名前の配列をサブスクライバ・クラスに渡した場合、以下のようになります。

   - システム・サブスクライバ SYS.Monitor.SystemSubscriber は、LockTablePercentFull 値がそのセンサに対する警告しきい値である 85 を超えていることを検出して、深刻度コード 1 と適切なテキストを含む通知を返します。
   - ヘルス・モニタ・スクライバ SYS.Monitor.Health.Control は、ProcessCount 値がそのセンサの構成済みパラメータおよび既定の通常値と比較して大きすぎると判定し、深刻度コード 2 と適切なテキストを含む通知を返します。

3. 通知の生成

   通知クラスには、1 つまたは複数のアラート・システムに通知を渡すためのメソッドが組み込まれています。システム・モニタは、各センサ・クラスおよびサブスクライバ・クラスを呼び出した後に、各通知クラスの Post() メソッドを呼び出して、Notifications プロパティにセンサ・クラスまたはサブスクライバ・クラスから返された通知を入力します。通知ク

ラスは、各通知を目的のアラート・メソッドに渡します。例えば、システム通知は、システム・サブスクライバから返された LockTablePercentFull に関する通知とヘルス・モニタ・サブスクライバから返された ProcessCount に関する通知を受け取ると、深刻度コードとテキストをメッセージ・ログに書き込みます。この手法では、ユーザ定義のアラート・システムのほかに、相互運用プロダクション・アラート・プロセッサや TrakCare 内のアラート・システムなどの独立したアラート・システムにも通知を渡すことができます。

システム・モニタは、インスタンスの起動時に、構成されている各実行開始ネームスペースで自動的に起動し、構成されているセンサ・クラスの呼び出しを開始して、構成されているサブスクライバ・クラスにセンサの値を渡し、次に、構成されている通知クラスに通知を渡します。ネームスペースごとにユーザ独自のシステム・モニタのセンサ・クラス、サブスクライバ・クラス、および通知クラスを定義して構成することができます。この章の "システム・モニタの既定のコンポーネント" のセクションの既定クラスを参照してください。

注釈　緊急時には、システム・モニタのシャットダウンが必要になる場合があります。クラスメソッド %SYS.Monitor.Enabled([flag]) はシステム・モニタの状態の設定、クリア、およびレポートします。flag が 0 の場合、システム・モニタは起動しません。

4.1.2 システム・モニタ通知の追跡

通常、システム・モニタのアラート (深刻度 2 の通知) またはシステム・モニタの警告 (深刻度 1) のシーケンスを調査する必要があります。ヘルス・モニタは、システム・モニタのアラートや警告も生成できます。

システム・モニタのアラート、警告、およびステータス・メッセージ (深刻度 0) はメッセージ・ログ (install-dir¥mgr¥messages.log) に書き込まれます (すべてのシステム・モニタのステータス・メッセージおよびヘルス・モニタのステータス・メッセージは、システム・モニタ・ログ (install-dir¥mgr¥SystemMonitor.log) に書き込まれます。アプリケーション・モニタのアラートはログに書き込まれませんが、電子メールで送信したり、指定した通知方法で受け渡しできます)。

システム・モニタのアラートおよび警告を追跡するために、以下のことを実行できます。

- ^%SYSMONMGR ユーティリティを使用して、システム・モニタのアラートを表示します。このオプションを使用すると、すべてのセンサまたは特定のセンサのアラートの表示および記録されているすべてのアラートまたは指定の期間内に発生したアラートのみの表示を行えます。警告は表示しません。
- メッセージ・ログを監視します ("管理ポータルを使用した InterSystems IRIS の監視" の章の "ログ・ファイルの監視" を参照してください)。システム・モニタのアラートのシーケンスが短期間内に指定されたセンサに対して生成される場合、最初のもののみがメッセージ・ログに書き込まれることに注意してください。

  注釈　メッセージ・ログでは、システム・モニタ状態通知の最初の文字が大文字で示されます (例 : `[System Monitor] started in %SYS`)。警告、アラート、OK メッセージは大文字で示されます (例 : `[SYSTEM MONITOR] CPUusage Warning: CPUusage = 90 ( Warnvalue is 85)`)。

- メッセージ・ログに表示されるアラート (およびオプションで警告) の電子メール通知を送信するようにログ・モニタを構成します (既定では、アラート・ログに書き込まれます)。この方法を使用する場合、ログ・モニタは、構成された深刻度のすべてのメッセージ・ログ・エントリに対して通知を生成するわけではないことに注意してください。約 1 時間以内に指定プロセス (システム・モニタなど) で連続エントリがある場合、通知は最初のエントリに対してのみ生成されます。例えば、ネットワークの問題によって ECP 接続および開いているトランザクションに関する複数のシステム・モニタのアラートが 15 分の間に生成された場合、ログ・モニタは、(最初に生成されたアラートに対して) 通知を 1 つのみ生成します。このため、ログ・モニタからシステム・モニタの通知を 1 通受け取ったら、すぐにシステム・モニタのアラートを表示し、メッセージ・ログを確認する必要があります。

4.1.3 システム・モニタのステータスおよびリソース・メトリック

以下のテーブルに、システム・モニタによってサンプリングされるシステム・ステータスおよびリソース使用量のメトリックと、警告 (深刻度 1)、アラート (深刻度 2)、および “ステータス OK” (深刻度 0) の通知の生成につながるそれぞれの通知しきい値および通知ルールを示します。

テーブル 4-1: システム・モニタのステータスおよびリソース通知

メトリック	説明	通知ルール

メトリック	説明	通知ルール
ディスク容量	データベース・ディレクトリの空き領域	・ < 250MB — 警告 ・ < 50MB — アラート ・ > 250MB (警告/アラート後) — OK
ジャーナル空き	ジャーナル・ディレクトリの空き領域	・ < 250MB — 警告 ・ < 50MB — アラート ・ > 250MB (警告/アラート後) — OK
ページング	使用中の物理メモリおよびページング領域の割合	・ ページング領域 > 30% — 警告 ・ (物理メモリ > 96%) + (ページング領域 > 50%) — アラート
ロック・テーブル	使用中のロック・テーブルの割合	・ > 85% — 警告 ・ > 95% — アラート ・ < 85% (警告/アラート後) — OK
ライト・デーモン	ライト・デーモンの状態	・ ライト・デーモンはアクティブ状態で、自身の (空でない) キューを処理中であるが、1 サイクルを完了するのに、構成されているライト・デーモン・サイクル時間 (既定値は 80 秒) よりも 10 秒以上長い時間がかかっている — アラート ・ アラート後にライト・デーモンがパスを完了した — OK
ECP 接続	ECP アプリケーション・サーバまたは ECP データ・サーバへの接続の状態	・ 5 秒以上障害状態になっている — アラート
共有メモリ・ヒープ (一般メモリ・ヒープ)	共有メモリ・ヒープ (SMH、別名は一般メモリ・ヒープ (gmheap)) のステータス	・ SMH (gmheap) ステータス 1 — 警告 ・ SMH (gmheap) ステータス 2 — アラート
オープン・トランザクション	最も長く開いているローカル・トランザクションおよびリモート (ECP) トランザクションの継続時間	・ > 10 分 — 警告 ・ > 20 分 — アラート
ライセンス期限切れ	ライセンスが期限切れになるまでの日数	・ 7 日 — 警告 ・ 5 日以下 — アラート (毎日)
SSL/TLS 証明書期限切れ	証明書が期限切れになるまでの日数	・ 個別の証明書が 30 日以内に期限切れ — 警告 (毎日繰り返し) ・ 1 つまたは複数の期限切れ証明書の警告 (毎日) — アラート (警告の集計、1 日 1 回)

メトリック	説明	通知ルール
ISCAgent (ミラー・メンバのみ)	ISCAgent のステータス	· 1 分未満の応答不能 – 警告 · 1 分を超える応答不能 – アラート

4.1.4 システム・モニタのヘルス状態

InterSystems IRIS インスタンスによって直接生成された両方のシステム・アラート、システム・モニタおよびそのヘルス・モニタ・コンポーネントによって生成されたアラートおよび警告など、メッセージ・ログに送信された通知に基づいて (このドキュメントの “管理ポータルを使用した InterSystems IRIS の監視” の章の ″ログ・ファイルの監視″ を参照)、システム・ヘルス全体を表す単一値を共有メモリのレジスタにシステム・モニタは保持します。

起動時、システム・ヘルスの状態は、起動プロセス中にメッセージ・ログに送信されたシステム (システム・モニタではない) のアラート数に基づいて設定されます。システム・モニタが実行されると、システムのアラートまたはシステム・モニタのアラートあるいは警告によってヘルス状態を引き上げることができます。最後のシステムのアラートまたはシステム・モニタのアラートあるいは警告が送信されてから 30 分が経過すると、ステータスは次に低いレベルに引き下げられます。以下の表に、システム・ヘルス状態がどのように決まるかを示します。

テーブル 4-2: システム・モニタのヘルス状態

状態	起動時の動作	起動後の動作	設定される状態
GREEN (0)	起動時に送信されたシステムのアラートなし	最後のシステムのアラートまたはシステム・モニタのアラートまたは警告が送信されてから 30 分 (状態が YELLOW だった場合) または 60 分 (状態が RED だった場合) 経過	N/A
YELLOW (1)	起動時に送信されたシステムのアラートが 4 個以下	状態が GREEN かつ ・ 1 つのシステムのアラートが送信された または ・ 1 つまたは複数のシステム・モニタのアラートまたは警告 (あるいは両方) が送信されたが、以下のように RED に設定するのに十分なアラートではない	最後のシステムのアラートまたはシステム・モニタのアラートあるいは警告が送信されてから 30 分経過した場合 GREEN
RED (2)	起動時に 5 つ以上のシステムのアラートが送信された	・ 状態が YELLOW で 1 つのシステムのアラートが送信された または ・ 状態が GREEN または YELLOW で 30 分以内に 5 つ以上の異なるセンサからシステム・モニタのアラートが送信されたか、または 1 つのセンサから 3 つのシステム・モニタのアラートが送信された	最後のシステムのアラートまたはシステム・モニタのアラートあるいは警告が送信されてから 30 分経過した場合 YELLOW

注釈　4 つ目の状態 HUNG は、グローバル更新がブロックされた場合に発生する可能性があります。特に、以下のイベントによって状態が HUNG に変更されます。

- ジャーナル・デーモンの一時停止が 5 秒を超えたか、またはフリーズした (″データ整合性ガイド″ の “ジャーナリング” の章の ″ジャーナル入出力エラー″ を参照してください)。
- スイッチ 10、11、13、14 のいずれかがセットされた (″専用のシステム/ツールおよびユーティリティ″ の “InterSystems IRIS のリモート管理” の章の ″スイッチの使用法″ を参照してください)。
- ライト・デーモンが何らかの理由で停止したか、**更新ロック**・フラグのセットが 3 秒を超えた。
- (データベース・キャッシュ内の) 使用可能なグローバル・バッファ数が危険な領域まで低下し、なおかつその状態が 5 秒を超えた。

ヘルス状態が HUNG に変更されると、その理由がメッセージ・ログに書き込まれます。

システム・モニタのヘルス状態は、以下を使用して表示できます。

- ^%SYSMONMGR の [システム・データを表示] メニューにある [**システム・ヘルスの表示**] オプション (HUNG はレポートしません)。
- **$SYSTEM.Monitor** API。システム・ステータスに直接アクセスできます。$SYSTEM.Monitor.State() を使用して、システム・ステータスを返します。SetState メソッド、Clear メソッド、Alert メソッド、GetAlerts メソッド、および ClearAlerts メソッドも参照してください。
- iris list コマンドおよび iris qlist コマンド (Windows のヘルス状態は含まれません)。

注釈　　システム・モニタが動作していない場合、システム・モニタのヘルス状態は常に GREEN となります。

4.1.5 システム・モニタの既定値

システム・モニタは、拡張可能な一連の付属のクラスを呼び出し、**%SYS** ネームスペースで稼働し、変更可能な 3 つの既定の設定で動作します。

4.1.5.1 システム・モニタの既定のコンポーネント

InterSystems IRIS には 5 つのクラスが付属しています。これらのクラスは、既定では **%SYS** ネームスペースのシステム・モニタで構成されます。

センサ・クラス：

- **SYS.Monitor.SystemSensors**

  構成されているサブスクライバ・クラス (システム・モニタのサブスクライバ (**SYS.Monitor.SystemSubscriber**) やヘルス・モニタのサブスクライバ (**SYS.Monitor.Health.Control**) など) に渡されるセンサ値を取得するシステム・センサ・クラス。

- %SYS.Monitor.AppMonSensor

  アプリケーション・モニタにセンサ・サービス、サブスクライバ・サービス、および通知サービスを提供するクラス。センサの値を取得し、その値をローカル・ネームスペースに保存して、ユーザ定義のアラートに基づいて値を評価します。アラート定義に基づいてアラートがトリガされたときには、電子メール・メッセージを生成するか、ユーザが指定したメソッドを呼び出します。

サブスクライバ・クラス：

- **SYS.Monitor.Health.Control**

  ヘルス・モニタのサブスクライバ・クラス。**SYS.Monitor.SystemSensors** からセンサ統計値を受け取って評価し、システム通知に通知を送信します。

- **SYS.Monitor.SystemSubscriber**

  すべてのセンサ・クラスで使用できるシステム・モニタ・サブスクライバ。**SYS.Monitor.SystemSensors** のセンサの監視および分析に必要なすべてのコードが含まれています。一部のセンサのシステム・モニタ通知およびヘルス・モニタ通知を生成します。

通知クラス：

- **SYS.Monitor.SystemNotify**

  すべてのサブスクライバ・クラスで使用できるシステム通知。システム・サブスクライバ (**SYS.Monitor.SystemSubscriber**) またはヘルス・モニタ・サブスクライバ (**SYS.Monitor.Health.Control**) から通知を受け取ると、その通知をシステム・モニタ・ログに書き込みます。通知が深刻度 2 (アラート) の場合はそれをメッセージ・ログに書き込みます (これらのログ・ファイルの詳細は、このドキュメントの “管理ポータルを使用した InterSystems IRIS の監視” の章を参照してください)。

また、システム通知は、SYS.Monitor.State() メソッドを使用して取得できるシステム・ステータスの単一の全体的評価も生成します。このメソッドは、0 (GREEN)、1 (YELLOW)、または 2 (RED) を返します。

^%SYSMONMGR を使用して、ユーザ定義のクラスを構成することができます。

4.1.5.2 システム・モニタの既定のネームスペース

すべてのシステム・モニタおよびアプリケーション・モニタの構成と設定は、ネームスペースに固有です。既定では、システム・モニタは %SYS ネームスペースでのみ起動および稼働します。^%SYSMONMGR を使用して、システム・モニタおよびアプリケーション・モニタの追加の実行開始ネームスペースを構成できます。あるネームスペースのシステム・モニタ構成またはアプリケーション・モニタ構成に変更を加えた後は、変更を有効にするためにそのネームスペースでシステム・モニタを再起動する必要があります。

ヘルス・モニタは、%SYS ネームスペースでのみ稼働します。

4.1.5.3 システム・モニタの既定の設定

既定では、システム・モニタはインスタンスの実行中は常に稼働しています。システム・モニタは、^%SYSMONMGR を使用して停止することはできますが、インスタンスが次回起動したときに自動的には再起動されません。

システム・モニタの既定の動作は以下のとおりです。

- 構成されている各センサ・クラスの GetSensors() メソッドを 30 秒ごとに呼び出します。
- システム・モニタ・ログにアラート、警告、およびメッセージのみを書き込みます。センサの読み取り値は書き込みません。
- センサの読み取り値は保存しません。

これらの設定は、^%SYSMONMGR を使用して変更できます。

4.1.6 ^%SYSMONMGR ユーティリティの使用

^%SYSMONMGR ユーティリティでは、システム・モニタを管理および構成できます。このユーティリティは任意のネームスペースで実行できます。このユーティリティを使用して行った変更は、このユーティリティが起動されたネームスペースにのみ影響します。ネームスペースで ^%SYSMONMGR を実行して構成した実行開始ネームスペースごとに個別のシステム・モニタ構成を保持する必要があります。あるネームスペースのシステム・モニタ構成に変更を加えた後は、変更を有効にするためにそのネームスペースでシステム・モニタを再起動する必要があります。

重要　ここで説明している ^%SYSMONMGR ユーティリティを使用した手動操作はすべて、%Monitor.Manager API のメソッドを使用してプログラムで実行できます。

システム・モニタを管理するには、ターミナルで以下のコマンドを入力します。

```
%SYS>do ^%SYSMONMGR
```

メイン・メニューが表示されます。

```
1) Start/Stop System Monitor
2) Set System Monitor Options
3) Configure System Monitor Classes
4) View System Monitor State
5) Manage Application Monitor
6) Manage Health Monitor
7) View System Data
8) Exit

Option?
```

実行する操作の番号を入力します。ユーティリティを終了するには Enter キーを押します。

メイン・メニューのオプションでは、以下のテーブルに示すシステム・モニタのタスクを実行できます。

オプション	説明
1) Start/Stop System Monitor	· システム・モニタの起動 · システム・モニタの停止
2) Set System Monitor Options	· 構成されているセンサ・クラスのサンプリング間隔の設定 · システム・モニタ・ログに書き込まれる情報のデバッグ・レベルの設定 · センサの読み取り値保存の有効化および保存日数の設定 · サンプリング間隔、デバッグ・レベル、およびセンサの読み取り値の保存を既定に戻す
3) Configure System Monitor Components	· ユーザ定義のセンサ・クラス、サブスクライバ・クラス、および通知クラスの構成または削除 · 実行開始ネームスペースの構成
4) View System Monitor State	· システム・モニタおよびシステム・モニタに構成されているコンポーネントの動作状態の表示
5) Manage Application Monitor	· アプリケーション・モニタのサブメニューの表示
6) Manage Health Monitor	· ヘルス・モニタのサブメニューの表示 (^%SYSMONMGR が %SYS ネームスペースで実行されている場合にのみ使用可)
7) View System Data	· 保存したセンサの読み取り値の表示 · システム・モニタのヘルス状態の表示 · 過去または現在のシステム・モニタのアラートの表示

4.1.6.1 Start/Stop System Monitor

システム・モニタは、InterSystems IRIS インスタンスの起動時に、構成されている各実行開始ネームスペースで自動的に起動し、構成されているクラスの呼び出しを開始します。これは、変更できません。ただし、インスタンスの実行中に、システム・モニタを停止することはできます。ヘルス・モニタの構成を変更するには、システム・モニタを停止する必要があります。また、あるネームスペースのシステム・モニタ構成に変更を加えた後は、変更を有効にするためにそのネームスペースでシステム・モニタを再起動する必要があります。

メイン・メニューで 1 を入力すると、以下のメニューが表示されます。

```
1) Start System Monitor
2) Stop System Monitor
3) Exit
```

実行中のシステム・モニタを停止するには 2 を入力し、停止中のシステム・モニタを起動するには 1 を入力します。

注釈　システム・モニタは、メッセージ・ログのサイズを監視し、必要に応じてロール・オーバーします。システム・モニタが停止すると、メッセージ・ログは、インスタンスが再起動されるか、PurgeErrorsAndLogs タスクが実行されるまで、MaxConsoleLogSize 構成設定で設定された制限を超える場合があります。メッセージ・ログについては、“管理ポータルを使用した InterSystems IRIS の監視” の章の ″ログ・ファイルの監視″ を参照してください。

4.1.6.2 Set System Monitor Options

システム・モニタのグローバル設定を変更するか、それらの設定を既定値に戻すには、システム・モニタが実行中の場合はシステム・モニタを停止し、メイン・メニューで 2 を入力します。

```
1) Set Sample Interval
2) Set Debugging Level
3) Reset Defaults
4) Manage Debug Data
5) Exit
```

1 を入力して、システム・モニタが構成されている各センサ・クラスを呼び出す間隔を設定します。既定値は 30 秒です。

2 を入力して、デバッグ・レベルを設定します。既定値は `0 (base)` で、システム・モニタおよびヘルス・モニタのステータスとエラー・メッセージがシステム・モニタ・ログに書き込まれます。センサの読み取り値は保存されません。デバッグ・レベル `1 (log all sensors)` では、センサ読み取り値がメッセージとともにシステム・モニタ・ログに書き込まれ、センサ読み取り値が保存されます。これは、[システム・データを表示] メニューの [センサ・データの表示] オプションを使用して表示することができます。

3 を入力して、サンプル間隔、デバッグ・レベル、およびセンサ読み取り値の保存を既定値にリセットします。

4 を入力して、センサ読み取り値を保存する日数を設定します (既定は 5)。

変更は、次回システム・モニタを起動または再起動したときに有効になります。

4.1.6.3 Configure System Monitor Components

″システム・モニタ″ で説明したように、%SYS.Monitor.AbstractSensor、%SYS.Monitor.AbstractSubscriber、および %SYS.Monitor.AbstractNotification をそれぞれ拡張して、独自のセンサ・クラス、サブスクライバ・クラス、および通知クラスを作成し、それらをシステム・モニタで構成して、″システム・モニタの既定のコンポーネント″ で説明した付属のクラスの機能を拡張することができます。システム・モニタを起動および実行する場所として %SYS 以外のネームスペースを追加することもできます。

Configure System Monitor Classes

メイン・メニューで 3 を入力すると、以下のメニューが表示されます。

```
1) Configure Components
2) Configure Startup Namespaces
3) Exit
```

1 を入力すると、クラスを構成するための以下のオプションが表示されます。

```
1) List Classes
2) Add Class
3) Delete Class
4) Exit
```

1 を入力すると、^%SYSMONMGR を起動したネームスペースで現在構成されているクラス (付属のシステム・クラスおよびユーザが構成したクラスを含む) がリストされます。

2 を入力すると、^%SYSMONMGR を起動したネームスペースのユーザ定義のクラスを構成できます。指定するクラスは、そのネームスペースに存在し、システム・モニタによって有効なセンサ・クラス、サブスクライバ・クラス、または通知クラスとして認識される必要があります。また、クラスの説明を入力することもできます。

3 を入力すると、構成したユーザ定義クラスを削除できます。

注釈　　クラスの構成または削除は、^%SYSMONMGR を起動したネームスペースにのみ影響します。

Configure System Monitor Namespaces

システム・モニタは、インスタンスの起動時に、構成されている各実行開始ネームスペース (既定では %SYS のみ) で個別のプロセスとして自動的に起動します。すべてのシステム・モニタの構成と設定は、ネームスペースに固有です。^%SYSMONMGR を使用して変更を加えた場合、その変更はこのユーティリティを起動したネームスペースにのみ影響します。

注釈　　^%SYSMONMGR のすべてのインスタンスは、同一のシステム・モニタ・ログにメッセージを書き込みます。実行開始ネームスペースは、任意のネームスペースから構成できます。

メイン・メニューで 3 を入力すると、以下のメニューが表示されます。

```
1) Configure Components
2) Configure Startup Namespaces
3) Exit
```

2 を入力すると、ネームスペースを構成するための以下のオプションが表示されます。

```
1) List Startup Namespaces
2) Add Namespace
3) Delete Namespace
4) Exit
```

1 を入力すると、現在構成されている実行開始ネームスペースがリストされます。

2 を入力すると、実行開始ネームスペースを追加できます。

3 を入力すると、実行開始ネームスペースを削除できます (%SYS は削除できません)。

4.1.6.4 View System Monitor State

メイン・メニューで 4 を入力すると、^%SYSMONMGR を起動したネームスペースのシステム・モニタとそのコンポーネントのステータスが表示されます。以下に例を示します。

```
      Component                   State
System Monitor                    OK
%SYS.Monitor.AppMonSensor         None
SYS.Monitor.SystemSensors         OK
SYS.Monitor.Health.Control        Running: Period is Thursday 09:00 - 11:30
SYS.Monitor.SystemSubscriber      OK
SYS.Monitor.SystemNotifier        OK
```

この例では、システム・モニタとそのシステム・センサ・クラス、サブスクライバ・クラス、および通知クラスが正常に動作しています。ヘルス・モニタのサブスクライバ・クラスも同様です。ただし、アプリケーション・モニタのクラスは有効化されていないため ("Manage Monitor Classes" を参照)、センサのサンプルの評価やアラートの生成は実行しません。

4.1.6.5 Manage Application Monitor

"^%SYSMONMGR を使用したアプリケーション・モニタの管理" を参照してください。

4.1.6.6 ヘルス・モニタの管理

"^%SYSMONMGR を使用したヘルス・モニタの管理" を参照してください。

4.1.6.7 システム・データの表示

メイン・メニューで 7 (または、%SYS 以外のネームスペースでは 6) を入力して、システムに関するシステム・モニタ情報を表示するオプションを表示します。

```
1) View Sensor Data
2) View System Health
3) View Alerts
4) Exit
```

[Set System Monitor Options] メニューの [Manage Debug Data] オプションを使用して、センサ・データの読み取り値の保存を有効化した場合、1 を入力して、保存されたセンサの読み取り値を表示します。すべてのセンサまたは特定のセンサの保存された読み取り値を表示できます。また、保存されているセンサの読み取り値すべてを表示するか、または指定した期間の読み取り値のみを表示できます。

2 を入力して、システム・モニタのヘルス状態を表示します。これには、前の GREEN 状態と現在の状態 (GREEN でない場合) の間のすべてのアラートが含まれます。

3 を入力してシステム・モニタのアラートを表示します。すべてのセンサまたは特定のセンサのアラートを表示できます。また、[Set System Monitor Options] メニューの [Manage Debug Data] オプションを使用して指定した期間内のすべてのアラートを表示できます。あるいは、指定した期間内のアラートのみ表示できます。

4.1.7 システム・モニタのコンポーネントの定義

SYS.Monitor API で、ユーザ独自のセンサ・クラス、サブスクライバ・クラス、および通知クラスを定義できます。

4.1.7.1 センサ・クラス

センサ・クラスは、%SYS.Monitor.AbstractSensor を拡張します。システム・モニタ・コントローラは、最初に各センサ・クラスの Start() メソッドを呼び出します。その後、各サイクルで GetSensors() メソッドを呼び出します。SetSensor() メソッドは、センサ・クラス内で使用され、SensorReading プロパティにセンサの名前と値の組み合わせを設定します。このプロパティは、GetSensors() によって返され、すべてのサブスクライバ・クラスに渡されます。

また、センサ・クラスはセンサの読み取り値を評価し、評価の結果として、通知またはユーザ定義のアラート・メソッドから電子メール・メッセージを生成するための %SYS.Monitor.Email クラスを呼び出します。

4.1.7.2 サブスクライバ・クラス

サブスクライバ・クラスは、%SYS.Monitor.AbstractSubscriber を拡張します。システム・モニタ・コントローラは、最初に各サブスクライバ・クラスの Start() メソッドを呼び出します。その後、各サイクルで、そのサイクルで呼び出されるセンサ・クラスごとに 1 回 Receive() メソッドを呼び出し、そのセンサ・クラスから受け取ったセンサの名前と値の組み合わせを設定した SensorReading プロパティを渡します。サブスクライバ・クラスは、1 つまたは複数の名前と値の組み合わせを評価し、Notify() メソッドを使用して通知を設定します。このメソッドは、Notifications プロパティに値を入力します。

また、サブスクライバ・クラスは、センサ評価の結果として、通知またはユーザ定義のアラート・メソッドから電子メール・メッセージを生成するための %SYS.Monitor.Email クラスも呼び出します。

サンプル・サブスクライバ・クラスとして、%SYS.Monitor.SampleSubscriber が用意されています。

4.1.7.3 通知クラス

通知クラスは、%SYS.Monitor.AbstractNotification を拡張します。システム・モニタ・コントローラは、最初に各通知クラスの Start() メソッドを呼び出します。その後、各サイクルで、そのサイクルで呼び出されるサブスクライバ・クラスごとに 1 回 Post() メソッドを呼び出し、そのサブスクライバから受け取った通知を設定した Notifications プロパティを渡します。次に、通知クラスは、自身のアラート・メソッド (複数可) に通知を渡します。このメソッドには、通知またはユーザ定義のアラート・メソッドから電子メール・メッセージを生成するための %SYS.Monitor.Email クラスが含まれていることがあります。

4.2 ヘルス・モニタ

ヘルス・モニタは、特定の期間にさまざまな重要なメトリックの値をサンプリングし、それらをそのメトリックの構成済みパラメータおよびその期間の規定の通常値と比較して、実行中の InterSystems IRIS インスタンスを監視します。サンプル値が大きすぎると、ヘルス・モニタはアラート (深刻度 2 の通知) または警告 (深刻度 1) を生成します。例えば、ヘルス・モニタによって月曜日の午前 10:15 にサンプリングされた CPU 使用率の値が、構成されている CPU 使用率の最大値または月曜日の午前 9:00 から午前 11:30 の期間に得られた通常の CPU 使用率サンプルと比較して大きすぎた場合、ヘルス・モニタは通知を生成します。

このセクションでは、以下のトピックについて説明します。

- ヘルス・モニタの概要
- ^%SYSMONMGR を使用したヘルス・モニタの管理

4.2.1 ヘルス・モニタの概要

ヘルス・モニタは、一定の規則を使用してサンプル値を評価し、異常に大きい値を特定します。この設計は、"NIST/SEMATECH e-Handbook of Statistical Methods" の "Process or Product Monitoring and Control" のセクションで説明されている、製造プロセスを監視するための手法に基づいています。通常値からの偏差は、WECO 統計的確率規則 (Western Electric Rules) に基づく規則を使用して判定されます。どちらも InterSystems IRIS 監視の目的に合うように特別に調整されています。

ヘルス・モニタのアラート (深刻度 2) および警告 (深刻度 1) はメッセージ・ログ (install-dir¥mgr¥messages.log) に書き込まれます。これらの通知を理解していることを確認する方法は、"システム・モニタ通知の追跡" を参照してください。

ヘルス・モニタのステータス・メッセージ (深刻度 0) は、システム・モニタ・ログ (install-dir¥mgr¥SystemMonitor.log) に書き込まれます。

注釈　システム・モニタおよびアプリケーション・モニタとは異なり、ヘルス・モニタは %SYS ネームスペースでのみ動作します。

以下のサブセクションでは、ヘルス・モニタがどのように動作するかについて説明し、ヘルス・モニタを構成および拡張するさまざまな方法に関する情報を提供します。

- ヘルス・モニタ・プロセスの説明
- センサとセンサ・オブジェクト
- 期間
- グラフ
- 通知ルール

4.2.1.1 ヘルス・モニタ・プロセスの説明

既定では、ヘルス・モニタは、インスタンスの起動時に自動的に起動しません。自動的に起動させるには、^%SYSMONMGR ユーティリティを使用してシステム・モニタ内でヘルス・モニタを有効にする必要があります (ヘルス・モニタを有効にした場合、InterSystems IRIS が起動してからヘルス・モニタが起動するまでの間隔を指定できます。これにより、インスタンスは、サンプリングの開始前に通常の動作状態に到達できます)。このユーティリティは、ヘルス・モニタの現在の状態を確認するために、常時使用可能です。詳細は、この章で後述する "^%SYSMONMGR を使用したヘルス・モニタの管理" を参照してください。

ヘルス・モニタ・プロセスの基本要素について以下に説明します。

- ヘルス・モニタは、センサ・オブジェクトとして表される多数のシステム・センサを監視します。各センサ・オブジェクトには、センサ・サンプルのベース (最小) 値があります。また、オプションで 2 つの通知しきい値 (アラート用に 1 つと、警告用に 1 つ) が含まれ、絶対値または乗数として設定できます。これらの値によって、ヘルス・モニタが通知を送信するタイミングが決定されます。

  以下の "センサとセンサ・オブジェクト" のセクションに、ヘルス・モニタが使用するすべてのセンサ・オブジェクトのリスト、およびその簡単な説明と既定値を示します。

- 事前定義された期間の間、各センサは 30 秒ごとにサンプリングされ、ベース値を下回るサンプルは破棄されます。既定では 63 の週次の期間がありますが (1 日 9 回)、独自の週、月、四半期、または年ごとの期間を構成できます。以下の "期間" のセクションに、既定の期間をリストします。

- 指定されたセンサでは、通知しきい値が絶対値として設定されている場合を除き、ヘルス・モニタはグラフに基づいてセンサの読み取り値を評価します。現行期間に必要なグラフがない場合、ヘルス・モニタはセンサを分析モードにして、グラフを生成します。

  グラフを編集または作成して、ヘルス・モニタがセンサの読み取り値を評価する方法を調整できます。詳細は、"グラフ" のセクションを参照してください。

- 分析モードでない場合、センサは監視モードになっています。監視モードでは、センサの読み取り値は、該当するサブスクライバ・クラスによって評価されます。一時的なサンプルの異常値によって通知がトリガされないように、3 分ごとに各 6 つのサンプル値が平均化されて 1 つの読み取り値が生成され、それらが評価対象の読み取り値となります。

- 連続する読み取り値が、通知の条件 (以下の "通知ルール" のセクションで説明) を満たす場合、サブスクライバ・クラスは、テキストと深刻度コードを含む通知をシステム通知 (SYS.Monitor.SystemNotify) に渡してアラートまたは警告を生成します。

  注釈　センサ・オブジェクトに最大値と警告値が指定されているセンサからの読み取り値を評価するのにグラフは必要ないため、それらのセンサの読み取り値の評価とその結果として生成される通知の送信は、SYS.Monitor.Health.Control サブスクライバ・クラスではなく SYS.Monitor.SystemSubscriber サブスクライバ・クラスによって処理されます ("システム・モニタの既定のコンポーネント" を参照してください)。結果的に、ヘルス・モニタが有効になっていない場合でも、システム・モニタが稼働している限り、これらのセンサの通知は生成されます。

  一部のセンサについては絶対値を使用して通知を生成し、その他のセンサについては乗数を使用して通知を生成する (例えば、一部のデータベースでは DBLatency センサの絶対値を使用し、その他のデータベースでは乗数を使用する) 場合は、センサ・オブジェクトで乗数を設定し、絶対値を使用するセンサのグラフを手動で作成します。詳細は、"グラフの編集" を参照してください。

4.2.1.2 センサとセンサ・オブジェクト

ヘルス・モニタのセンサ・オブジェクトは、SYS.Monitor.SystemSensors 内のいずれかのセンサを表します。各センサ・オブジェクトは、ベース値を持つ必要があります。必要に応じて、最大 (アラート) しきい値および警告しきい値 (絶対値または乗数) を持つこともできます。センサの読み取り値の評価でこれらの値がどのように使用されるかについては、"通知ルール" を参照してください。以下のテーブルに、ヘルス・モニタのセンサ・オブジェクトとその既定のパラメータを示します。

一部のセンサは、InterSystems IRIS インスタンスの全体的メトリックを示します。全体的メトリックは、以下のテーブルの**センサ・アイテム**列に値のないメトリックです。例えば、LicensePercentUsed センサは、現在使用中のインスタンスの許可済みライセンス・ユニットの割合をサンプリングします。また、JournalGrowthRate センサは、インスタンスのジャーナル・ファイルに書き込まれたデータの量 (1 分あたりの KB) をサンプリングします。

他のセンサは、特定のセンサ・アイテム (CSP サーバ、データベース、またはミラー) に関する情報を収集します。例えば、DBReads センサは、マウントされている各データベースからの 1 分あたりの読み取り数をサンプリングします。これらのセンサは、<sensor_object> <sensor_item> と指定されています。例えば、DBLatency install-dir¥IRIS¥mgr¥user センサは、USER データベースでランダム読み取りが完了するまでに要する時間を (ミリ秒単位で) サンプリングします。

センサ・オブジェクトは、以下の "ヘルス・モニタ・クラスの構成" のセクションの説明に従って ^%SYSMONMGR ユーティリティを使用して表示および編集 (ただし削除は不可) できます。センサ・オブジェクトの編集では、1 つまたはすべての値を変更できます。入力できるのは、ベース値のみ、ベース値、最大 (アラート) 値、および警告値、またはベース値、最大 (アラート) 乗数、および警告乗数です。

テーブル 4-3: ヘルス・モニタのセンサ・オブジェクト

センサ・オブジェクト	センサ・アイテム	説明	ベース値	最大値	最大乗数	警告値	警告乗数
CPUUsage		システムの CPU 使用率 (%)。	50	85	—	75	—
CSPSessions	IP_address:port	リストされている Web ゲートウェイ・サーバ上のアクティブな Web セッションの数。	100	—	2	—	1.6
CSPActivity	IP_address:port	リストされている Web ゲートウェイ・サーバへの 1 分あたりの要求数。	100	—	2	—	1.6
CSPActualConnections	IP_address:port	リストされている Web ゲートウェイ・サーバ上で作成された接続の数。	100	—	2	—	1.6
CSPInUseConnections	IP_address:port	リストされている Web ゲートウェイ・サーバへの現在アクティブな接続の数。	100	—	2	—	1.6
CSPPrivateConnections	IP_address:port	リストされている Web ゲートウェイ・サーバへのプライベート接続の数。	100	—	2	—	1.6
CSPUrlLatency	IP_address:port	IP_address:port/csp/sys/UtilHome.csp からの応答の取得に要した時間 (ミリ秒)。	1000	5000	—	3000	—
CSPGatewayLatency	IP_address:port	CSP センサ・オブジェクトによって表されるメトリックのフェッチ時に、リストされている Web ゲートウェイ・サーバからの応答の取得に要した時間 (ミリ秒)。	1000	2000	—	1000	—
DBLatency	database_directory	リストにあるマウントされたデータベースからのランダム読み取りが完了するまでに要した時間 (ミリ秒)。	1000	3000	—	1000	—
DBReads	database_directory	リストにあるマウントされたデータベースからの 1 分あたりの読み取り数。	1024	—	2	—	1.6
DBWrites	database_directory	リストにあるマウントされたデータベースへの 1 分あたりの書き込み数。	1024	—	2	—	1.6
ECPAppServerKBPerMinute		ECP データ・サーバへの 1 分あたりの送信データ量 (KB)。	1024	—	2	—	1.6

センサ・オブジェクト	センサ・アイテム	説明	ベース値	最大値	最大乗数	警告値	警告乗数
ECPConnections		アクティブな ECP 接続の数。	100	—	2	—	1.6
ECPDataServerKBPerMinute		ECP データ・サーバでの 1 分あたりの受信データ量 (KB)。	1024	—	2	—	1.6
ECPLatency		ECP データ・サーバとこの ECP アプリケーション・サーバ間のネットワーク遅延 (ミリ秒)。	1000	3000	—	3000	—
ECPTransOpenCount		開いている ECP トランザクションの数。	100	—	2	—	1.6
ECPTransOpenSecsMax		現在最も長く開いている ECP トランザクションの継続時間 (秒)。	60	—	2	—	1.6
GlobalRefsPerMin		1 分あたりのグローバル参照数。	1024	—	2	—	1.6
GlobalSetKillPerMin		1 分あたりのグローバル set 数とグローバル kill 数。	1024	—	2	—	1.6
JournalEntriesPerMin		1 分あたりのジャーナル・エントリ書き込み数。	1024	—	2	—	1.6
JournalGrowthRate		ジャーナル・ファイルへの 1 分あたり書き込みデータ量 (KB)。	1024	—	2	—	1.6
LicensePercentUsed		現在使用中の許可済みライセンス・ユニットの割合。	50	—	1.5	—	—
LicenseUsedRate		1 分あたりのライセンス取得数。	20	—	1.5	—	—
LockTablePercentFull		使用中のロック・テーブルの割合。	50	99	—	85	—
LogicalBlockRequestsPerMin		1 分あたりの論理ブロック要求数。	1024	—	2	—	1.6
MirrorDatabaseLatencyBytes	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリからは受け取ったものの、バックアップのミラーリング対象データベースには適用されていないジャーナル・データのバイト数 (バックアップ・データベースの遅滞尺度)。	$2*10^7$	—	2	—	1.6

センサ・オブジェクト	センサ・アイテム	説明	ベース値	最大値	最大乗数	警告値	警告乗数
MirrorDatabaseLatencyFiles	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリからは受け取ったものの、バックアップのミラーリング対象データベースにはまだ完全に適用されていないジャーナル・ファイルの数（バックアップ・データベースの遅滞尺度）。	3	—	2	—	1.6
MirrorDatabaseLatencyTime	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリから最終ジャーナル・ファイルを受け取った時間と、バックアップのミラーリング対象データベースに最終ジャーナル・ファイルが完全に適用された時間の差（ミリ秒）（バックアップ・データベースの遅滞尺度）。	1000	4000	—	3000	—
MirrorJournalLatencyBytes	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリからは受け取ったものの、バックアップのジャーナル・ディレクトリには書き込まれていないジャーナル・データのバイト数（バックアップの遅滞尺度）。	$2*10^7$	—	2	—	1.6
MirrorJournalLatencyFiles	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリからは受け取ったものの、バックアップのジャーナル・ディレクトリには書き込まれていないジャーナル・ファイルの数（バックアップの遅滞尺度）。	3	—	2	—	1.6
MirrorJournalLatencyTime	mirror_name	ミラーのバックアップ・フェイルオーバー・メンバにおける、プライマリから最終ジャーナル・ファイルを受け取った時間と、最終ジャーナル・ファイルがバックアップのジャーナル・ディレクトリに完全に書き込まれた時間の差（ミリ秒）（バックアップの遅滞尺度）。	1000	4000	—	3000	—
PhysicalBlockReadsPerMin		1 分あたりの物理ブロック読み取り数。	1024	—	2	—	1.6
PhysicalBlockWritesPerMin		1 分あたりの物理ブロック書き込み数。	1024	—	2	—	1.6

センサ・オブジェクト	センサ・アイテム	説明	ベース値	最大値	最大乗数	警告値	警告乗数
ProcessCount		InterSystems IRIS インスタンスのアクティブなプロセスの数。	100	—	2	—	1.6
RoutineCommandsPerMin		1 分あたりのルーチン・コマンドの数。	1024	—	2	—	1.6
RoutineLoadsPerMin		1 分あたりのルーチンのロード数。	1024	—	2	—	1.6
RoutineRefsPerMin		1 分あたりのルーチン参照数。	1024	—	2	—	1.6
SMHPercentFull		使用中の共有メモリ・ヒープ（一般メモリ・ヒープ）の割合。	50	98	—	85	—
TransOpenCount		開いているローカル・トランザクション（ローカルおよびリモート）の数。	100	—	2	—	1.6
TransOpenSecondsMax		現在最も長く開いているローカル・トランザクションの継続時間（秒）。	60	—	2	—	1.6
WDBuffers		1 ライト・デーモン・サイクルの間に更新されるデータベース・バッファの平均数。	1024	—	2	—	1.6
WDCycleTime		1 ライト・デーモン・サイクルを完了するために必要な平均の秒数。	60	—	2	—	1.6
WDWIJTime		ライト・イメージ・ジャーナル（WIJ）の更新にかかる 1 サイクルあたりの平均時間（秒）。	60	—	2	—	1.6
WDWriteSize		1 ライト・デーモン・サイクルあたりの平均書き込みバイト数。	1024	—	2	—	1.6

注釈　いくつかのセンサは、全部の InterSystems IRIS インスタンスに対してサンプリングされません。例えば、ECP . . . センサは、ECP データとアプリケーション・サーバのみについてサンプリングされます。

ミラー・メンバを監視しているときに（"高可用性ガイド" の "ミラーリング" の章を参照してください）、ヘルス・モニタには以下の特別な条件が適用されます。

- ミラーの再起動中（例えば、バックアップ・フェイルオーバー・メンバがプライマリを引き継いだ直後）や、ミラーでのメンバのステータスが不確定である場合は、センサはサンプリングされません。
- センサが一定期間分析モードになっていて、ミラーでのメンバのステータスがその期間中に変化した場合、グラフは作成されず、センサは分析モードのままになります。
- バックアップ・フェイルオーバー・ミラー・メンバでは MirrorDatabaseLatency* センサと MirrorJournalLatency* センサのみがサンプリングされます。
- プライマリ・フェイルオーバー・ミラー・メンバでは、MirrorDatabaseLatency* センサと MirrorJournalLatency* センサを除くすべてのセンサがサンプリングされます。

4.2.1.3 期間

既定では 63 の週次の繰り返し期間があり、センサはそれらの期間にサンプリングされます。これらの期間はそれぞれ、特定の曜日の以下の指定済みの間隔のいずれかを表します。

テーブル 4-4: 既定のヘルス・モニタの期間

午前 0:15 ～ 午前 2:45	午前 3:00 ～ 午前 6:00	午前 6:15 ～ 午前 8:45
午前 9:00 ～ 午前 11:30	午前 11:45 ～ 午後 1:15	午後 1:30 ～ 午後 4:00
午後 4:15 ～ 午後 6:00	午後 6:15 ～ 午後 8:45	午後 9:00 ～ 午後 11:59

^%SYSMONMGR ユーティリティの [期間の構成] オプションを使用して、期間をリスト、追加、および削除できます (以下の "ヘルス・モニタ・クラスの構成" のセクションを参照)。週次の期間のほかに、月、四半期、および年ごとの期間を追加できます。

注釈　四半期ごとの期間は、開始月として指定された月から 3 か月ごとにリスト表示されます。例えば、開始月に 5 (5 月) を指定した場合、四半期のサイクルは、8 月(8)、11 月 (11)、および 2 月 (2) に繰り返されます。

ユーザ定義期間では、説明はオプションです。

4.2.1.4 グラフ

センサ・オブジェクトの通知しきい値が乗数として与えられていない場合 (または指定されていない場合)、ヘルス・モニタでこれらのセンサ読み取り値を評価するには、グラフが必要です。ヘルス・モニタは、サンプルのセンサ読み取り値から平均値、標準偏差、および最大値を計算して、必要なグラフを生成します。このセクションでは、ヘルス・モニタによって分析モードでグラフを生成する方法、およびグラフの編集またはカスタム・グラフの作成の方法について説明します。

分析モード

センサのサンプルを評価する前に、ヘルス・モニタは、そのセンサにグラフが必要かどうかを確認します。グラフが必要であるがまだ存在しない場合、ヘルス・モニタはそのセンサを自動的に分析モードにします。

分析モードでは、ヘルス・モニタは単純に収集したサンプル値を記録し、期間の終了時にそのセンサに必要なグラフを生成します。グラフの信頼性を確保するために、分析モードでは最低 13 サンプルを取得しておく必要があります。1 回の期間で有効な 13 サンプルが取得されるまで、センサは分析モードにとどまり、その期間に対するグラフは生成されません。

注釈　グラフは、InterSystems IRIS インスタンスが通常の安定した動作をしている間に取得されたサンプルから生成される必要があります。例えば、月曜日 09:00 a.m. ～ 11:30 a.m. のグラフが存在しないとき、月曜日が休日の場合や、技術的な問題が InterSystems IRIS インスタンスの動作に影響している場合には、グラフを生成しないでください。

あるセンサまたはセンサ/アイテムの特定の期間に対してグラフが生成されてから、5 回その期間が経過すると (アラートが生成された期間を除く)、これら 5 回の通常期間における読み取り値が評価され、センサの平均値の上昇や変化が検出されます。平均値が 95% の確信度で上昇しているか、変化している場合は、グラフが再校正されます。センサのその期間の既存のグラフは、その期間の最後の発生時に取得されたサンプルから生成されたグラフと置き換えられます。例えば、あるデータベースへのユーザのアクセス数が緩やかでも着実に増加している場合、そのデータベースの DBReads 値の平均も緩やかかつ着実に増加する傾向にあり、これにより 5 期間ごとにグラフの定期的な再校正が発生し、不当なアラートが回避されます。

センサ・オブジェクトの絶対値と乗数値を同じように自動的に再校正することはできません。グラフの自動再校正はこのようなセンサには適用されないため、手動で調整する必要があります。例えば、データベースのユーザのアクセス数が増加している場合、DBLatency センサ・オブジェクトのベース値、最大 (アラート) 値、および警告値は手動で調整する必要があります。

グラフの編集

^%SYSMONMGR ユーティリティでは、現在のすべてのグラフのリストと、それぞれの平均値と標準偏差を表示できます。特定のグラフの個々の読み取り値や最大読み取り値などの詳細を表示することもできます。ユーティリティからこれらのオプションにアクセスするには、[Configure Health Monitor Classes] サブメニューから `[Configure Charts]` を選択します。

`[Configure Charts]` オプションは、グラフをカスタマイズすることでアラートをカスタマイズする 2 つの方法を提供します。

- 既存のグラフを編集して、平均値や標準偏差を任意の値に変更できます。標準の通知ルールは適用されますが、入力した値が使用されます。
- グラフを作成して、アラート値と警告値を指定できます。グラフを作成することは、通知しきい値に絶対値を設定することと似ています。アラートと警告は、グラフに指定した値のみに基づいて生成されます。

注釈　グラフをリスト、検証、編集、または作成するときに、**アイテム**の見出しまたはプロンプトは、データベース (ディレクトリ・パスで指定)、Web ゲートウェイ・サーバ (IP アドレスで指定)、またはミラー (ミラー名で指定) を参照します。詳細は、"センサとセンサ・オブジェクト" を参照してください。

また、以下の SYS.Monitor.Health.Chart クラス・メソッドにより、リストの値に基づいた統計グラフをプログラムで作成することができます。

- CreateChart() ー 特定のサンプリング期間やセンサのグラフの作成、リストの値の評価、および結果として得られる平均値と標準偏差 (σ) 値の設定を行います。
- SetChartStats() ーリストの値の評価、および特定のサンプリング期間やセンサについて、結果として得られる平均値と標準偏差 (σ) 値の設定を行います。

詳細は、SYS.Monitor.Health.Chart クラス・ドキュメントを参照してください。

注釈　上述の "分析モード" のセクションで説明されているように、ヘルス・モニタによって生成されたグラフ (ユーザが編集したグラフも含む) は、自動的に再校正できます。また、InterSystems IRIS インスタンスをアップグレードすると、編集済みのものも含め、ヘルス・モニタによって生成されたすべてのグラフは削除されます。

[Configure Charts] サブメニューまたは CreateChart() クラス・メソッドを使用して作成されたグラフは、自動的に再校正されることもアップグレード時に削除されることもありません。したがって、ユーザが作成したグラフは、[Configure Health Monitor Classes] サブメニューの [Reset Defaults] オプション内の [Reset Charts] オプションを選択するか、[Configure Charts] オプション内の [Recalibrate Charts] を選択しない限り、選択したセンサと期間の組み合わせに永続的に関連付けられます。

4.2.1.5 通知ルール

ヘルス・モニタは、ある期間のセンサの読み取り値が 3 回連続してセンサの最大しきい値を上回った場合にアラート (深刻度 2 の通知) を生成し、ある期間のセンサの読み取り値が 5 回連続してセンサの警告しきい値を上回った場合に警告 (深刻度 1 の通知) を生成します。最大しきい値および警告しきい値は、センサ・オブジェクトの設定と、適用可能なグラフがヘルス・モニタによって生成されたのかユーザによって作成されたのかによって異なります。以下のテーブルを参照してください。

以下についても注意してください。

- センサ・オブジェクトに最大値と警告値が設定されている場合は、グラフは必要ないため生成されません。通知はヘルス・モニタが無効になっている場合でも生成されます。
- センサ・オブジェクトに最大乗数と警告乗数が設定されている場合やベース値のみが設定されている場合は、グラフが必要です。分析モードで十分なサンプルが収集されてグラフが生成されるまで、通知は生成されません。
- ユーザが作成したグラフがある場合は、センサ・オブジェクトの設定がどのようになっていてもかまいません。

センサ・オブジェクトの設定	グラフ・タイプ	センサの最大値	センサの警告値	アクティブになる条件
ベース値、最大値、警告値	なし	センサ・オブジェクトの最大値	センサ・オブジェクトの警告値	システム・モニタが稼働中
ベース値、最大乗数、警告乗数	生成済み	以下のうちの大きい方の値にセンサ・オブジェクトの最大乗数を掛けた値 ・ グラフの平均値に標準偏差を 3 回加えた値 ・ グラフの最大値に標準偏差を加えた値	以下のうちの大きい方の値にセンサ・オブジェクトの警告乗数を掛けた値 ・ ベース値 ・ グラフの平均値に標準偏差を 2 回加えた値 ・ グラフの最大値	システム・モニタが稼働中、ヘルス・モニタが有効
ベース値のみ	生成済み	以下のうちの大きい方の値 ・ グラフの平均値に標準偏差を 3 回加えた値 ・ グラフの最大値	以下のうちの大きい方の値 ・ グラフの平均値に標準偏差を 2 回加えた値 ・ グラフの最大値	システム・モニタが稼働中、ヘルス・モニタが有効
(ユーザが作成したグラフが存在する場合は、該当なし)	ユーザ作成済み	グラフのアラート値	グラフの警告値	システム・モニタが稼働中、ヘルス・モニタが有効

4.2.1.6 例

この例では、月曜日の 9:00 a.m. 〜 11:30 a.m. の期間における DBReads install-dir¥IRIS¥mgr¥user センサのグラフに、USER データベースからの 1 分あたりの平均読み取り数が 2145、標準偏差が 141、および最大値が 2327 であることが示されています。DBReads の既定の通知しきい値乗数は、2 です。読み取り値が 3 回連続して次の 2 つの値のうちの大きい方を上回ると、このセンサにアラートが生成されます。

・ 最大乗数 * (グラフの平均値 + (3 * グラフの標準偏差))

  2 * (2145 + (3 * 141)) = 5136

・ 最大乗数 * (グラフの最大値 + グラフの標準偏差)

  2 * (2327 + 141) = 4936

このため、この期間にこのセンサで読み取り値が 3 回連続して 5136 を上回ると、アラートが生成されます。

乗数または最大値がないセンサは、1 の乗数で評価されます。例えば、DBReads センサ・オブジェクトが編集され、乗数が削除されて、ベース値のみが残った場合、読み取り値が 3 回連続して (以下のうち大きい方として計算された) 2568 を上回ると、DBReads install-dir¥IRIS¥mgr¥user アラートが生成されます。

・ 最大乗数 * (グラフの平均値 + 標準偏差に 3 を掛けた値)

  1 * (2145 + (3 * 141)) = 2568

・ 最大乗数 * (グラフの最大値 + 標準偏差)

  1 * (2327 + 141) = 2468

4.2.2 ^%SYSMONMGR を使用したヘルス・モニタの管理

"^%SYSMONMGR ユーティリティの使用" で説明されているように、^%SYSMONMGR ユーティリティでは、ヘルス・モニタを含め、システム・モニタを管理および構成できます。ヘルス・モニタを管理するには、ターミナルで %SYS ネームスペースに変更してから、以下のコマンドを入力します。

```
%SYS>do ^%SYSMONMGR

1) Start/Stop System Monitor
2) Set System Monitor Options
3) Configure System Monitor Classes
4) View System Monitor State
5) Manage Application Monitor
6) Manage Health Monitor
7) View System Data
8) Exit

Option?
```

注釈　ヘルス・モニタは、%SYS ネームスペースでのみ稼働します。別のネームスペースで ^%SYSMONMGR を起動すると、オプション 6 (Manage Health Monitor) は表示されません。

6 (Manage Health Monitor) を入力します。以下のメニューが表示されます。

```
1) Enable/Disable Health Monitor
2) View Alerts Records
3) Configure Health Monitor Classes
4) Set Health Monitor Options
5) Exit

Option?
```

実行する操作の番号を入力します。ヘルス・モニタ・ユーティリティを終了するには Enter キーを押します。

メイン・メニューのオプションでは、以下のテーブルに示すヘルス・モニタのタスクを実行できます。

オプション	説明
1) Enable/Disable Health Monitor	· ヘルス・モニタの有効化 (ヘルス・モニタが既定で無効になっている場合)。これにより、システム・モニタの起動時にヘルス・モニタが起動します。ヘルス・モニタは、構成されている起動待機時間が完了するまで、センサの読み取り値の収集を開始しません。 · ヘルス・モニタの無効化 (ヘルス・モニタが有効になっている場合)。これにより、システム・モニタの起動時にヘルス・モニタは起動しません。
2) View Alert Records	· 指定された日付範囲での 1 つまたはすべてのセンサ・オブジェクトのアラート・レコードの表示。
3)	· 通知ルールの表示。 · 既存の期間のリスト表示と削除、および新しい期間の追加。 · グラフのリスト表示、検証、編集、および作成。 · センサ・オブジェクトのリスト表示とセンサ・オブジェクトの設定の編集。 · ヘルス・モニタの要素を既定値にリセット。
4)	· 起動待機時間の設定。 · アラート・レコードを削除する時期の指定。

注釈　ユーティリティから、センサ、ルール、期間、グラフなど、1 つの要素を指定するように求められた場合には、番号付きリストのプロンプトで「**?** 」(疑問符) を入力して、その後で必要な要素の番号を入力できます。

このユーティリティからのすべての出力は、ターミナルで表示するか、指定のデバイスに送信できます。

4.2.2.1 アラート・レコードの表示

特定のセンサまたはすべてのセンサについて、最近生成されたアラートを表示するには、このオプションを選択します。グラフの平均値および標準偏差や通知をトリガする読み取り値など、個別のアラートおよび警告の詳細を検証することができます。(アラート・レコードは、構成された日数後に削除されます。詳細は "ヘルスモニタオプションの設定" を参照してください。)

4.2.2.2 ヘルスモニタクラスの構成

このサブメニューの各オプションでは、以下のテーブルのようにヘルス・モニタをカスタマイズできます。

注釈　システム・モニタの稼働中にこれらのオプションを使用してヘルス・モニタをカスタマイズすることはできません。まず、システム・モニタを停止し、変更を行った後にシステム・モニタを再起動する必要があります。

オプション	説明
1) /	(このリリースでは使用しません)
2)	現在構成されている期間のリストを表示し、期間を追加および削除します。
3)	以下のことができます。 ・ すべての既存のグラフの平均値と標準偏差のリストを期間ごとにまとめて表示します。 ・ 平均値や標準偏差の基になる読み取り値や、最大読み取り値など、個々のグラフの詳細情報を確認します。 ・ [Edit Charts] オプションを使用して、既存のグラフの平均値と標準偏差を変更します。 ・ グラフを作成して、アラートしきい値と警告しきい値を指定します。 ・ 最新のデータを使用して、すべてのグラフ (ユーザが作成したグラフも含む) または個々のグラフを手動で再校正します。
4) Edit Sensor Objects	SYS.Monitor.SystemSensors クラス内のセンサを表すセンサ・オブジェクトを表示し、センサ・オブジェクトのベース値、最大値、警告値、最大乗数、および警告乗数を変更します。

オプション	説明
5)	以下のことができます。 ・ 既定の期間構成にリセットし、既存のグラフをすべて削除し、すべての期間を分析モードに戻します（"ヘルス・モニタ・プロセスの説明" を参照してください）。 ・ ユーザ定義の期間構成は削除せずに、既存のグラフ（ユーザが作成したグラフも含む）をすべて削除し、すべての期間を分析モードに戻します。 ・ すべてのセンサ・オブジェクトを既定値にリセットします。 ・ ヘルス・モニタのオプション（起動待機時間およびアラートの削除時間）を既定値にリセットします。

4.2.2.3 ヘルスモニタオプションの設定

このサブメニューでは、以下のテーブルに示すヘルス・モニタのいくつかのオプションを設定できます。

オプション	説明
1)	起動後、センサの読み取り値をヘルス・モニタ・サブスクライバ SYS.Health.Monitor.Control に渡すまでのシステム・モニタの待機時間（分）を構成します（ヘルス・モニタが有効になっている場合）。これにより、InterSystems IRIS は、ヘルス・モニタがグラフの作成または読み取り値の評価を開始する前に、通常の動作状態に到達できます。
2)	アラート・レコードをいつ削除するかを指定します。既定では、アラートの生成から 5 日後です。

4.3 アプリケーション・モニタ

アプリケーション・モニタは、ユーザ拡張可能なメトリック・セットを監視し、自身が収集したデータの永続リポジトリを維持し、ユーザが構成したアラートをトリガします。

このセクションでは、以下のトピックについて説明します。

- アプリケーション・モニタの概要
- ^%SYSMONMGR を使用したアプリケーション・モニタの管理
- アプリケーション・モニタのメトリック
- ユーザ定義アプリケーション・モニタ・クラスの記述

4.3.1 アプリケーション・モニタの概要

アプリケーション・モニタは、システム・モニタに構成されている各実行開始ネームスペースで、ユーザが選択したシステム定義メトリックとユーザ定義メトリックのセットを監視する、拡張可能なユーティリティです。"システム・モニタの既定のコンポーネント" で説明されているように、%SYS.Monitor.AppMonSensor（アプリケーション・モニタ・センサ・クラス）は、システム・モニタによって呼び出されると、メトリックをサンプリングし、サンプルを評価し、それ自体の通知を生成します。（システム・モニタの通知やヘルス・モニタの通知と異なり、これらはメッセージ・ログに書き込まれません。）具体的に言うと、アプリケーション・モニタはシステム・モニタの各実行開始ネームスペースで以下を実行します。

1. システム・モニタの起動時に、起動します。

2. 付属のシステム・モニタ・クラスを登録できます (これらは既定で %SYS に登録されます)。
3. 監視対象のシステム定義クラスおよびユーザ定義クラスを有効化できます。有効化できるのは、登録されているシステム・クラスと、ローカル・ネームスペースにあるユーザ定義クラスです。例えば、ユーザ定義クラスを USER ネームスペースでのみ作成すると、そのクラスは USER ネームスペースでのみ有効化できます。
4. 各アクティブ・クラスによって指定されたメトリックをサンプリングして、それらのクラスを監視します。これらのメトリックは、モニタ・クラスの GetSample() メソッドによって呼び出されたサンプル・クラスから返されるプロパティを表します。例えば、%Monitor.System.LockTable クラスは、%Monitor.System.Sample.LockTable クラスを呼び出します。このクラスは、TotalSpace プロパティ (ロック・テーブルの合計サイズを格納) や、UsedSpace プロパティ (ロック・テーブルの使用中の領域のサイズを格納) などを返します。サンプリングされたデータとモニタ・メタデータおよびクラス・メタデータは、ローカル・ネームスペースに保存され、すべてのオブジェクトおよび SQL からアクセスできるようになります。
5. あるクラスのアラートが構成されており、そのクラスがアラートに構成されている評価式を満たすプロパティ値を返した場合、電子メール・メッセージを生成するかクラス・メソッドを呼び出します (これらのアクションのいずれかがアラートに構成されている場合)。例えば、最初に受信者への電子メール通知を構成し、次に %Monitor.System.LockTable クラス用のアラートを構成し、%Monitor.System.Sample.LockTable の UsedSpace プロパティの TotalSpace プロパティに対する比率が .9 (使用率 90%) より大きい場合に電子メールが送信されるように指定します。

注釈　アプリケーション・モニタで提供されている %Monitor.System.HistorySys クラスと %Monitor.System.HistoryPerf クラスは、有効化されると、システムの使用状況とパフォーマンスのメトリックの履歴データベースを作成および保持します。これは、システムの使用状況とパフォーマンスの問題を経時的に分析するのに役立ちます。これらのクラスと %Monitor.System.HistoryUser は、%SYS でのみ動作し、他のネームスペースに登録することはできません。これらのクラスと履歴データベースの詳細は、このドキュメントの“履歴モニタ”の章を参照してください。

4.3.2 ^%SYSMONMGR を使用したアプリケーション・モニタの管理

"^%SYSMONMGR ユーティリティの使用" で説明されているように、^%SYSMONMGR ユーティリティでは、アプリケーション・モニタを含め、システム・モニタを管理および構成できます。このユーティリティは任意のネームスペースで実行できます。このユーティリティを使用して行った変更は、このユーティリティが起動されたネームスペースにのみ影響します。ネームスペースで ^%SYSMONMGR を起動して構成した実行開始ネームスペースごとに、個別のアプリケーション・モニタ構成を保持する必要があります。

注釈　アプリケーション・モニタ構成にクラスの有効化などの変更を加えた後は、変更を有効にするために、その変更を行ったネームスペースでシステム・モニタを再起動する必要があります。

アプリケーション・モニタを管理するには、ターミナルで以下のコマンドを入力します。

```
%SYS>do ^%SYSMONMGR
```

次に、**5** (Manage Application Monitor) を入力します。以下のメニューが表示されます。

```
1) Set Sample Interval
2) Manage Monitor Classes
3) Change Default Notification Method
4) Manage Email Options
5) Manage Alerts
6) Debug Monitor Classes
7) Exit

Option?
```

実行する操作の番号を入力します。アプリケーション・モニタ・ユーティリティを終了するには Enter キーを押します。

4.3.2.1 Manage Application Monitor

メイン・メニューのオプションでは、以下のテーブルで説明するようにアプリケーション・モニタを管理できます。

オプション	説明
1) Set Sample Interval	メトリックがサンプリングされる間隔を設定します。既定値は 30 秒です。この設定は、([Manage Monitor Classes] サブメニューの [Set Class Sample Interval] オプションを使用して) クラス固有の間隔を設定することでクラスごとにオーバーライドできます。 注意：システム・モニタのサンプリング間隔 ([Set System Monitor Options] サブメニューの [Set Sample Interval] を参照) がアプリケーション・モニタ・クラスのサンプリング間隔よりも長い場合、2 つの間隔のうち長い方が使用されます。例えば、システム・モニタの間隔が 30 でアプリケーション・モニタの間隔が 120 である場合、アクティブなアプリケーション・モニタ・クラスはすべて 120 秒ごとにサンプリングされます。システム・モニタの間隔が 60 で %Monitor.System.LockTable クラスの間隔が 20 の場合、クラスは 60 秒ごとにサンプリングされます。
2) Manage Monitor Classes	アプリケーション・モニタ・マネージャを実行しているネームスペースでシステム定義モニタ・クラスとユーザ定義モニタ・クラスを管理するための [Manage Monitor Classes] サブメニューが表示されます。
3) Change Default Notification Method	アラートがトリガされたときに実行されるアラートの既定のアクションを指定できます。別途指定しない限り、作成したすべてのアラートでこのアクションが使用されます。
4) Manage Email Options	[Monitor Email Options] サブメニューが表示されます。このサブメニューでは、アラートでこのアクションを指定できるように、電子メール通知を有効化して構成することができます。
5) Manage Alerts	[Manage Alerts] サブメニューが表示されます。このサブメニューでは、システム定義およびユーザ定義のモニタ・クラスのアラートを作成できます。

4.3.2.2 Manage Monitor Classes

このサブメニューでは、システム定義モニタ・クラスおよびユーザ定義モニタ・クラスを管理できます。実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 2

1) Activate/Deactivate Monitor Class
2) List Monitor Classes
3) Register Monitor System Classes
4) Remove/Purge Monitor Class
5) Set Class Sample Interval
6) Exit

Option?
```

このサブメニューでは、以下のテーブルのように、システム定義クラスおよびユーザ定義クラスを管理するメニュー項目が表示されます。

オプション	説明
1) Activate / Deactivate Monitor Class	アプリケーション・モニタは、アクティブなクラスのみをサンプリングします。このオプションでは、非アクティブなクラスを有効化するか、アクティブなクラスを無効化できます。ローカル・ネームスペースに登録されているシステム定義クラスとユーザ定義クラスの番号付きリストとそれぞれの有効化状況を表示するには、[Class?] プロンプトで「?」と入力し、番号またはクラス名を入力します。
2) List Monitor Classes	ローカル・ネームスペースに登録されているシステム定義クラスとユーザ定義クラスのリストと、それぞれの有効化状況を表示します。
3) Register Monitor System Classes	すべてのシステム・モニタ・クラス (%Monitor.System.HistorySys クラス、%Monitor.System.HistoryPerf クラス、および %Monitor.System.HistoryUser クラスは除く) を登録して、それらのクラスをローカル・ネームスペースに保存します。サンプリングを開始するために、このメニューの [1) Activate/Deactivate Monitor Class] オプションを使用して、システム・クラスを引き続き有効にしておく必要があります。
4) Remove/Purge Class	ローカル・ネームスペースにあるクラスのリストからモニタ・クラスを削除します。ローカル・ネームスペースに登録されているシステム定義クラスとユーザ定義クラスの番号付きリストとそれぞれの有効化状況を表示するには、[Class?] プロンプトで「?」と入力し、番号またはクラス名を入力します。 注釈　このオプションでは、クラスそのものが削除されるわけではなく、有効化できる登録クラスのリストからそのクラスの名前が削除されるだけです。リストをリセットするには、このメニューのオプション [3) Register Monitor System Classes] を選択します。
5) Set Class Sample Interval	アプリケーション・モニタの既定のサンプリング間隔をクラスごとにオーバーライドできます。この間隔は [Manage Application Monitor] メニューの [1) Set Sample Interval] オプションで指定されています。既定値は 0 で、そのクラスにはクラス固有のサンプリング間隔が設定されていないことを意味します。 この設定の間隔、[Set Sample Interval] 設定の間隔、および "Set System Monitor Options" で説明しているシステム・モニタのサンプリング間隔の優先順位については、[Set Sample Interval] オプションの説明を参照してください。
6) Debug Monitor Classes	[Debug Monitor Classes] メニューが表示されます (エラーのリストも表示されます)。このメニューでは、デバッグ機能を有効化または無効化できます。

Debug Monitor Classes

このサブメニューでは、システムのデバッグ機能を管理できます。

[Debugging monitor classes] により、サンプル値の収集中に、ユーザ定義のアプリケーション・モニタ・クラスによって生成されたエラーを取得する機能が追加されます。

実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 6

1) Enable Debug
2) Disable Debug
3) List Errors
4) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルに示すように、アプリケーション・モニタのデバッグ機能を管理できます。

入力フィールド	説明
1) Enable Debug	デバッグ機能を有効にします。クラスがサンプル値を生成していない場合は、エラーが原因でサンプル値の保存が妨げられているかどうかを確認してください。
2) Disable Debug	デバッグ機能を無効にします。
3) List Errors	ローカル・ネームスペース内のエラーの定義がすべて表示されます。例えば、以下のようなものがあります。 %Save()、%New()、Initialize()、GetSample()。 ^%SYSMONMGR を使用してクラスのデバッグ機能を有効にすると、システム・モニタはクラス内の特定のメソッドで捕捉した最後のエラーを保存します。

4.3.2.3 Change Default Notification Method

アラートの作成時に、そのアラートがトリガされたときに実行されるアクションを指定します。このアクションの既定の選択肢が、このオプションを使用して設定する既定の通知方法です。実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 3
Notify Action (0=none,1=email,2=method)? 0 =>
```

このオプションでの選択は、以下のテーブルのように、既定の通知メソッドを使用するアラートの構成で使用されます。

入力フィールド	説明
0	アラートがトリガされてもアクションを実行しません。
1	アラートがトリガされると、構成されている受信者に電子メール・メッセージを送信します。電子メールの構成に関する詳細は、"Manage Email Options" を参照してください。
2	アラートがトリガされると、通知メソッドを呼び出します。このアクションを選択すると、アラートで指定されているアプリケーション名と、サンプル・クラスによってモニタ・クラスに返されるプロパティを含む %List オブジェクトを引数とする通知メソッドが呼び出されます ("アプリケーション・モニタの概要" を参照してください)。プロンプトが表示されたら、完全なクラス名とメソッド (packagename.classname.method) を入力します。このメソッドは、ローカル・ネームスペースに存在する必要があります。

4.3.2.4 Manage Email Options

このサブメニューのオプションでは、電子メールの構成および有効化が可能です。電子メールが有効になっている場合、電子メール通知を送信するよう構成されているアラートがトリガされると、アプリケーション・モニタから電子メール通知が送信されます。実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 4

1) Enable/Disable Email
2) Set Sender
3) Set Server
4) Manage Recipients
5) Set Authorization
6) Test Email
7) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルのようにアプリケーション・モニタの電子メール通知を管理できます。

オプション	説明
1) Enable / Disable Email	電子メールを有効にすると、アラートがトリガされたときに、アプリケーション・モニタが電子メール通知を送信できます（構成されている場合）。電子メールを無効にすると、アラートがトリガされても電子メール通知は送信されません。 注釈　電子メールを無効にしてから、再び有効にした場合は、電子メール・オプションを再構成する必要はありません。
2) Set Sender	このオプションは必須です。電子メールの送信者を識別するテキストを入力します。指定されている送信メール・サーバに応じて、このテキストが有効な電子メール・アカウントである必要がある場合とない場合があります。
3) Set Server	このオプションは必須です。サイトの送信電子メールを処理するサーバの名前を入力します。サーバの名前が不明の場合、IT 担当者に問い合わせてください。
4) Manage Recipients	このオプションでは、受信者の有効な電子メール・アドレスのリスト、追加、または削除が可能な次のサブメニューが表示されます。 1) List Recipients 2) Add Recipient 3) Remove Recipient 4) Exit 受信者を追加または削除する場合は、電子メール・アドレスを個別に（1 行に 1 つずつ）入力する必要があります。無効な形式のアドレスは拒否されます。
5) Set Authorization	電子メール・サーバが承認を必要とする場合、承認のユーザ名とパスワードを指定できます。IT 担当者に問い合わせて、この情報を入手してください。エントリを入力しないと、承認のユーザ名とパスワードは NULL に設定されます。
6) Test Email	指定された電子メール・サーバを使用して、指定された受信者にテスト・メッセージを送信します。送信に失敗した場合は、結果として生成されたエラー・メッセージが問題の解決に役立つことがあります。

4.3.2.5 Manage Alerts

アラートでは以下を指定します。

- 対象のネームスペース内の条件。モニタ・クラスによってサンプリングされたプロパティの値によって定義されます。
- その条件が発生した場合にそれをユーザに通知するために実行されるアクション。

前述の例に戻るには、アラートを作成して、以下を指定します。

- 条件 : ロック・テーブルの使用率が 90% を超えている。これは、%Monitor.System.LockTable クラスが %Monitor.System.Sample.LockTable を呼び出したときに返される、UsedSpace プロパティの値が TotalSpace プロパティの 90% を超える場合として定義されます。
- アクション : 電子メール通知の送信。

プロパティに基づいた条件の定義は、評価式と呼ばれます。使用するサンプル・クラスのプロパティを指定した後に、評価式を指定します。評価式では、プロパティは、ユーザがプロパティを指定した順番と対応するプレースホルダで示されます。例えば、ロック・テーブルのアラートを作成するときに、最初に UsedSpace プロパティを指定し、次に TotalSpace プロパティを指定した場合、評価式を「**%1 / %2 > .9**」と入力します。このとき、プロパティを逆の順番で入力すると、式は「**%2 / %1 > .9**」となります。

アラート・メニューが表示されたら、実行する操作の番号を入力します。メイン・メニューに戻るには Enter キーを押します。

```
Option? 2

1) Create Alert
2) Edit Alert
3) List Alerts
4) Delete Alert
5) Enable/Disable Alert
6) Clear NotifyOnce Alert
7) Exit

Option?
```

このサブメニューの各オプションでは、以下のテーブルのようにアプリケーション・モニタのアラートを管理できます。

入力フィールド	説明
1) Create Alert	新しいアラートを定義できます。プロンプトおよびそれに対する入力の説明は、"アラート・プロンプトへの入力" を参照してください。新しく作成したアラートは既定で有効になります。
2) Edit Alert	既存のアラートを変更できます。編集するアラートの名前を入力するか、「?」と入力して既存のアラートのリストを表示して、番号または名前を入力します。 注釈　内容を変更しないプロンプトも含め、すべてのプロンプトで入力する必要があります。つまり、変更するフィールドの改訂情報だけでなく、変更しないフィールドの情報も再入力する必要があります。プロンプトおよびそれに対する入力の説明は、"アラート・プロンプトへの入力" を参照してください。

入力フィールド	説明
3) List Alerts	ローカル・ネームスペースにあるすべてのアラートの定義を表示します。例えば、以下のようになります。 Alert: LockTable90 USER Action: email Class: %Monitor.System.LockTable Property: UsedSpace,TotalSpace Expression: %1/%2>.9 Notify Once: True Enabled: Yes
4) Delete Alert	既存のアラートを削除できます。編集するアラートの名前を入力するか、「?」と入力して既存のアラートのリストを表示して、番号または名前を入力します。 注釈　各アラートを個別に入力する必要があります。つまり、削除するアラートを列や範囲で指定することはできません。
5) Enable / Disable Alert	アラートを有効にすると、そのアラートはアクティブになります。アラートを無効にすると、そのアラートは非アクティブになります。 注釈　アラートを無効にしてから、再び有効にした場合は、アラート・オプションを再構成する必要はありません。
6) Clear NotifyOnce Alert	アラートがトリガされたときに、指定のアラート名に内部フラグ Notified を設定できます。これが設定されると、別のアラートが送信されることはありません。

以下のテーブルでは、アラート・プロンプトでの有効な入力について説明します。

テーブル 4-5: アラート・プロンプトへの入力

入力フィールド	説明
Alert Name?	英数字の名前を入力します。ローカル・ネームスペースで既に定義されているアラート名の番号付きリストを表示するには、[Alert Name?] プロンプトで「?」と入力します。
Application?	電子メール・メッセージまたは通知メソッドに渡す説明テキストを入力します。このテキストには、この後の手順の [Property?] プロンプトで指定するプロパティへの参照を %N の形式で含めることができます。%1 は最初のプロパティを表し、%2 は 2番目のプロパティを表します（以下同様）。

入力フィールド	説明
`Action (0=default, 1=email, 2=method)?`	アラートがトリガされたときに実行されるアクションを指定します。以下のオプションのいずれかを入力します。 ・ **0** – 既定で指定されている通知方法 (なし、電子メール、またはクラス・メソッド) を使用します。"Change Default Notification Method" を参照してください。 ・ **1** – アラートがトリガされると、説明テキストとサンプル・クラスによってモニタ・クラスに返されたプロパティの名前と値が含まれた電子メール・メッセージ ("アプリケーション・モニタの概要" を参照) を構成された電子メール受信者に送信します。電子メールの構成に関する詳細は、"Manage Email Options" を参照してください。 ・ **2** – 説明テキストと、サンプル・クラスによってモニタ・クラスに返されるプロパティを含む %List オブジェクトを引数とする、指定の通知メソッドを呼び出します ("アプリケーション・モニタの概要" を参照してください)。 プロンプトが表示されたら、完全なクラス名とメソッド (packagename.classname.method) を入力します。このメソッドは、ローカル・ネームスペースに存在する必要があります。
`Raise this alert during sampling?` または `Define a trigger for this alert?`	アラートの作成では、1 番目のプロンプトが表示されます。作成時に 1 番目のプロンプトで「No」を入力したアラートの編集では、2 番目のプロンプトが表示されます。以下のいずれかを入力します。 ・ **Yes** – 必要な情報のプロンプト表示を続行します。 ・ **No** – スキップして終了します。`[Class?]`、`[Property?]`、および `[Evaluation expression?]` の各プロンプトは省略されます。
`Class?`	ローカル・ネームスペースに登録されているシステム定義モニタ・クラスまたはユーザ定義モニタ・クラスの名前を入力します。ローカル・ネームスペースにある登録クラスの番号付きリストとそれぞれの有効化状況を表示するには、`[Class?]` プロンプトで「**?**」と入力します。 注釈　非アクティブなクラスのアラートを作成できます。アラートが構成されているクラスが削除されても、アラート自体は削除されません。
`Property?`	前のプロンプトで指定したクラスで定義されているプロパティの名前 (式の評価や説明テキストで使用しているもの) を入力します。指定のクラスで定義されているプロパティの番号付きリストを表示するには、`[Property?]` プロンプトで「**?**」と入力して、番号または名前を入力します。各プロパティは個別に入力する必要があります。入力の完了後、空白のプロンプトで Enter キーを押すと、プロパティのリストが指定した順序で表示されます。
`Evaluation expression?`	`[Property?]` プロンプトで指定したプロパティの評価に使用する式。例えば、**`(%1 = "User") && (%2 < 100)`** では、%1 はプロパティのリストの最初のプロパティを表し、%2 は 2 番目のプロパティを表します。
`Notify once only?`	以下のいずれかを入力します。 ・ **Yes** – アラートの最初のトリガでのみユーザに通知します。 ・ **No** – アラートがトリガされるたびにユーザに通知します。

4.3.3 アプリケーション・モニタのメトリック

アプリケーション・モニタに付属のシステム・モニタ・クラスは、次の表に示すように、さまざまなサンプル・クラスを呼び出します。

サンプル・クラス	アプリケーション・モニタ・システム・クラス
監査メトリック	%Monitor.System.Sample.AuditCount および %Monitor.System.Sample.AuditEvents
クライアント・メトリック	%Monitor.System.Sample.Clients
Web ゲートウェイ・メトリック	%Monitor.System.Sample.CSPGateway
ディスク容量メトリック	%Monitor.System.Sample.Diskspace
空き容量メトリック	%Monitor.System.Sample.Freespace
グローバル・メトリック	%Monitor.System.Sample.Globals
履歴データベース・メトリック（このドキュメントの “履歴モニタ” の章を参照）	%MonitorSystemSampleHistoryPerf, %MonitorSystemSampleHistorySys, %MonitorSystemSampleHistoryUser
ジャーナル・メトリック	%Monitor.System.Sample.Journals
ライセンス・メトリック	%Monitor.System.Sample.License
ロック・テーブル・メトリック	%Monitor.System.Sample.LockTable
プロセス・メトリック	%Monitor.System.Sample.Processes
ルーチン・メトリック	%Monitor.System.Sample.Routines
サーバ・メトリック	%Monitor.System.Sample.Servers
システム活動メトリック	%Monitor.System.Sample.SystemMetrics

各カテゴリのサンプル・メトリックに対応するプロパティのリストは、″インターシステムズ・クラス・リファレンス″ を参照してください。

MONITOR 機能を制御する類似の関数群が、%Monitor.System パッケージのクラスに用意されています。このパッケージを使用すると、収集したデータに名前を付け、永続オブジェクト形式として保存できます。詳細は、″インターシステムズ・クラス・リファレンス″ で、%Monitor.System.Sample パッケージ・クラスのドキュメント、および %Monitor.System.SystemMetrics クラスのドキュメントを参照してください。

4.3.3.1 メトリックの生成

%Monitor.SampleAgent クラスは実際のサンプリングを行い、メトリック・クラスの Initialize() メソッドと GetSample() メソッドを呼び出します。

%Monitor.SampleAgent.%New() コンストラクタは、実行するメトリック・クラスの名前を引数として取ります。役割は、そのクラスのインスタンスを生成し、そのクラスの Startup() メソッドを呼び出すことです。その後、%Monitor.SampleAgent.Collect() メソッドが呼び出されるたびに、Sample Agent はそのクラスの Initialize() メソッドを呼び出し、そのクラスの GetSample() メソッドを繰り返し呼び出します。さらに、GetSample() が呼び出されるたびに、%Monitor.SampleAgent は、メトリック・クラスのサンプル・クラスを作成します。これらの処理の擬似コードは以下のとおりです。

```
set sampler = ##class(%Monitor.SampleAgent).%New("MyMetrics.Freespace")
/* at this point, the sampler has created an instance of MyMetrics.Freespace,
and invoked its Startup method */
for I=1:1:10 { do sampler.Collect() hang 10 }
/* at each iteration, sampler calls MyMetrics.Freespace.Initialize(), then loops
on GetSample().  Whenever GetSample() returns $$$OK, sampler creates a new
MyMetrics.Sample.Freespace instance, with the sample data. When GetSample()
returns an error value, no sample is created, and sampler.Collect() returns. */
```

4.3.3.2 メトリック・データの表示

すべてのメトリック・クラスは、CSP 対応です。サンプル・クラスの生成時に、CSP コードが自動的に生成されます。したがって、メトリックを表示する最も簡単な方法は、Web ブラウザを使用することです。例えば、"メトリックの生成" の例に基づくと、また、スーパーサーバ・ポート番号を 52773 と仮定すると、CSP の URL は http://<instance-host>:52773/csp/user/MyMetrics.Sample.Freespace.cls です。これにより、以下のような出力が表示されます。

```
Monitor - Freespace c:\InterSystems\IRIS51\
            Name of dataset:  c:\InterSystems\IRIS51\
Current amount of Freespace:  8.2MB

Monitor - Freespace c:\InterSystems\IRIS51\mgr\
            Name of dataset:  c:\InterSystems\IRIS51\mgr\
Current amount of Freespace:  6.4MB
```

%Monitor.View クラスの Display(metric_class) メソッドを使用することもできます。例えば、以下のようになります。

```
%SYS>set mclass="Monitor.Test.Freespace"

%SYS>set col=##class(%Monitor.SampleAgent).%New(mclass)

%SYS>write col.Collect()
1
%SYS>write ##class(%Monitor.View).Display(mclass)

Monitor - Freespace    c:\InterSystems\IRIS51\
                Name of dataset:  c:\InterSystems\IRIS51\
    Current amount of Freespace:  8.2MB

Monitor - Freespace    c:\InterSystems\IRIS51\mgr\
                Name of dataset:  c:\InterSystems\IRIS51\mgr\
    Current amount of Freespace:  6.4MB
```

注釈　名前に % (パーセント記号) が含まれているクラスの URL は、その場所に %25 を使用する必要があります。例えば、%Monitor.System.Freespace クラスの場合、以下のようになります。

http://localhost:52773/csp/sys/%25Monitor.System.Freespace.cls

4.3.4 ユーザ定義アプリケーション・モニタ・クラスの記述

付属のシステム・クラスのほか、ユーザ・アプリケーションのデータとカウンタを監視する独自のモニタ・クラスおよびサンプル・クラスを記述することができます。

モニタ・クラスは、抽象モニタ・クラス %Monitor.Adaptor から継承する任意のクラスです。%Monitor.System クラスはそのようなクラスの例です。独自のユーザ定義モニタ・クラスを作成する手順は次のとおりです。

1. データを監視するネームスペースで ^%MONAPPMGR を実行します。オプション 2 を使用してモニタ・クラスをリストし、そのメニュー内のオプション 3 を使用して、モニタ・システム・クラスを登録します。

```
SAMPLES>d ^%MONAPPMGR


1) Set Sample Interval
2) Manage Monitor Classes
3) Change Default Notification Method
4) Manage Email Options
5) Manage Alerts
6) Exit

Option? 2

1) Activate/Deactivate Monitor Class
2) List Monitor Classes
3) Register Monitor System Classes
4) Remove/Purge Monitor Class
5) Set Class Sample Interval
6) Exit
```

```
Option? 3
Exporting to XML started on 06/21/2022 12:52:36
Exporting class: Monitor.Sample
Export finished successfully.

Load started on 06/21/2022 12:52:36
Loading file C:\InterSystems\SRCCTRL\mgr\Temp\t0jFhPqLkZoYAA.stream as xml
Imported class: Monitor.Sample
Compiling class Monitor.Sample
Compiling table Monitor.Sample
Compiling routine Monitor.Sample.1
Load finished successfully.


1) Activate/Deactivate Monitor Class
2) List Monitor Classes
3) Register Monitor System Classes
4) Remove/Purge Monitor Class
5) Set Class Sample Interval
6) Exit

Option?
```

2. %Monitor.Adaptor を継承するクラスを記述します。この継承は、永続性、パラメータ、プロパティ、コード生成、およびユーザのクラス定義からモニタ・メタデータを生成するプロジェクションを提供します。このクラスおよびユーザが記述する必要があるコードの詳細は、%Monitor.Adaptor クラスの説明を参照してください。
3. クラスをコンパイルします。%Monitor.Adaptor から継承したクラスをコンパイルすると、ユーザ・クラスのサブパッケージである Sample に新しいサンプル・クラスが生成されます。例えば、A.B.MyMetric をコンパイルすると、新しいクラスが A.B.Sample.MyMetric に生成されます。生成されたクラスで何らかの操作を実行する必要はありません。

   重要　アプリケーション・モニタ・クラスを削除するには、モニタ・クラスのみを削除します。つまり、生成されたサンプル・クラスは削除しないようにします。管理ポータルを使用して、サンプル・クラス（例えば、A.B.Sample.MyMetric）の生成元であるモニタ・クラス（例えば、A.B.MyMetric）のみを削除します。これにより、モニタ・クラスと生成されたサンプル・クラスの両方が自動的に削除されます。

サンプル・クラスはすべて自動的に CSP 対応になるので、A.B.Sample.MyMetric.cls を指定すれば、ユーザ・メトリックのサンプル・データを参照できます。アプリケーション・モニタはこのクラスを自動的に呼び出し、このクラスが有効な場合はデータとアラートを生成します。モニタ・クラスの有効化に関する情報は、"Manage Monitor Classes" を参照してください。

重要　**SECURITYRESOURCE** パラメータは、明示的に変更していない場合、%Monitor.Adaptor 内（したがって、%Monitor.Adaptor から継承するユーザ・クラス内）では空です。コード生成では、**SECURITYRESOURCE** 値がユーザ定義のクラスから生成されたサンプル・クラスにコピーされます。

以下の簡単な例では、InterSystems IRIS インスタンスの各データセットに空き領域を確保します。

それぞれのサンプリングでは、n インスタンスのサンプル・データ・オブジェクトを要求します。各インスタンスはデータセットに対応しています。また、この例では、各インスタンスのプロパティは 1 つだけ（サンプル収集時におけるそのデータセットの空き容量）です。

1. %Monitor.Adaptor を継承するクラスを作成します。

   Class Definition

```
Class MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
}
```

2. サンプル・データの一部となるプロパティを追加します。それらの型は %Monitor パッケージ内のクラスである必要があります。
   - Gauge
   - Integer

- Numeric
- String

例えば以下のようになります。

Class Definition

```
Class MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
/// Name of dataset
Property DBName As %Monitor.String(CAPTION = "Database Name");

/// Current amount of Freespace
Property FreeSpace As %Monitor.String;
}
```

3. サンプルのインスタンスの中で、どのフィールドが一意キーになるかを指定する INDEX パラメータを追加します。

Class Member

```
Parameter INDEX = "DBName";
```

4. 必要に応じてコントロール・プロパティを追加して、そのプロパティが生成されたクラスのストレージ定義の一部にならないように `[Internal]` のマークを付けます。

Class Member

```
/// Result Set for use by the class
Property Rspec As %SQL.StatementResult [Internal];
```

5. Initialize() メソッドをオーバーライドします。Initialize() は、各メトリックの収集を開始する時点で呼び出されます。

Class Member

```
/// Initialize the list of datasets and freespace.
Method Initialize() As %Status
{
    set stmt=##class(%SQL.Statement).%New()
    set status= stmt.%PrepareClassQuery("SYS.Database","FreeSpace")
    set ..Rspec = stmt.%Execute()
    return $$$OK
}
```

6. GetSample() メソッドをオーバーライドします。GetSample() は、状態値 `0` が返されるまで繰り返し呼び出されます。各サンプル・インスタンスにメトリック・データを取り込むためのコードを記述します。

Class Member

```
/// Get dataset metric sample.
/// A return code of $$$OK indicates there is a new sample instance.
/// A return code of 0 indicates there is no sample instance.
Method GetSample() As %Status
{
    // Get freespace data
    set stat = ..Rspec.%Next(.sc)

    // Quit if we have done all the datasets
    if 'stat {
        Quit 0
    }

    // populate this instance
    set ..DBName = ..Rspec.%Get("Directory")
    set ..FreeSpace = ..Rspec.%Get("Available")

    // quit with return value indicating the sample data is ready
    return $$$OK
}
```

7. クラスをコンパイルします。クラスは以下のようになります。

Class Definition

```
Class MyMetric.Freespace Extends %Monitor.Adaptor
{
Parameter INDEX = "DBName";

/// Name of dataset
Property DBName As %Monitor.String;

/// Current amount of Freespace
Property FreeSpace As %Monitor.String;

/// Result Set
Property Rspec As %SQL.StatementResult [Internal];

/// Initialize the list of datasets and freespace.
Method Initialize() As %Status
{
    set stmt=##class(%SQL.Statement).%New()
    set status= stmt.%PrepareClassQuery("SYS.Database","FreeSpace")
    set ..Rspec = stmt.%Execute()
    return $$$OK
}

/// Get routine metric sample.
/// A return code of $$$OK indicates there is a new sample instance.
/// Any other return code indicates there is no sample instance.
Method GetSample() As %Status
{
    // Get freespace data
    set stat = ..Rspec.%Next(.sc)

    // Quit if we have done all the datasets
    if 'stat {
        Quit 0
    }

    // populate this instance
    set ..DBName = ..Rspec.%Get("Directory")
    set ..FreeSpace = ..Rspec.%Get("Available")

    // quit with return value indicating the sample data is ready
    return $$$OK
}

}
```

8. さらに、必要であれば、Startup() メソッドと Shutdown() メソッドをオーバーライドします。これらのメソッドは、サンプリングの開始時に一度だけ呼び出されます。したがって、チャネルを開く操作など、1 回限りの初期化処理を実行できます。

```
/// Open a tcp/ip device to send warnings
Property io As %Status;

Method Startup() As %Status
{
    set ..io="|TCP|2"
    set host="127.0.0.1"
    open ..io:(host:^serverport:"M"):200
}

Method Shutdown() As %Status
{
    close ..io
}
```

9. このクラスをコンパイルすると、パッケージ MyMetric.Sample に新しいクラス MyMetric.Sample.Freespace が作成されます。

Class Definition

```
/// Persistent sample class for MyMetric.Freespace
Class MyMetric.Sample.Freespace Extends Monitor.Sample
{
Parameter INDEX = "DBName";

Property Application As %String [ InitialExpression = "MyMetric" ];

/// Name of dataset
Property DBName As %Monitor.String(CAPTION = "");

/// Current amount of Freespace
Property FreeSpace As %Monitor.String(CAPTION = "");

Property GroupName As %String [ InitialExpression = "Freespace" ];

Property MetricsClass As %String [ InitialExpression = "MyMetric.Freespace" ];

}
```

注釈　このクラスは変更しないでください。ただし、このクラスから継承して、サンプル・データに対するカスタム・クエリを記述することは可能です。

重要　アクティブなユーザ定義のアプリケーション・モニタ・クラスを変更してリコンパイルすると、そのクラスは非アクティブ化され、クラス固有のサンプリング間隔のオーバーライドは削除されます（設定されている場合）。このクラスをリストアするには、アクティブ化し、必要に応じてサンプリング間隔を再設定して、システム・モニタを再起動します。

5

^GLOSTAT を使用したグローバル動作の統計収集

InterSystems IRIS® Data Platform には、グローバル活動の統計情報を収集し、ディスク入出力操作に関するさまざまな情報を表示する ^GLOSTAT ユーティリティが備わっています。

管理ポータルから、^GLOSTAT によって報告される統計情報を表示することもできます。監視しているシステムのポータルにログインして、**[システム使用]** ページ (**[システム処理]** > **[システム使用]**) に移動します。

5.1 ^GLOSTAT の実行

^GLOSTAT を実行するには、**%SYS** ネームスペースにいる必要があります。ルーチン名は大文字と小文字が区別されます。以下のコマンドを入力して、Enter キーを押します。

ObjectScript

```
do ^GLOSTAT
```

^GLOSTAT ルーチンによって統計情報が表示されます (例 A を参照)。InterSystems IRIS を起動するたびに、^GLOSTAT の統計情報カウンタが初期化されます。このため、^GLOSTAT の初期出力には、InterSystems IRIS を起動した後の操作が反映されます。

レポートの下に、以下のプロンプトが表示されます。

```
Continue (c), Timed Stats (# sec > 0), Quit (q)?
```

以下のいずれかを入力してください。

応答	動作
c	レポートが再び表示され、前回の初期化以降の状況を反映した最新の累積統計データが示されます。
q	^GLOSTAT ルーチンを終了します。
# (秒数を示す正の整数)	統計データが初期化され、指定した秒数に相当する統計がカウントされます。さらに、1 秒あたりの平均値として統計データが表示されます (例 B)。

5.2 ^GLOSTAT 統計の概要

^GLOSTAT 統計情報は、InterSystems IRIS の起動後に発生したイベントの数をタイプごとに示します。また、指定した時間間隔について、1 秒あたりのイベント発生回数を表示することもできます。システム管理者のネームスペースから、いつでも ^GLOSTAT を実行できます。多くの場合、停止中のシステムではなく、アクティブなシステムでこのユーティリティを実行する必要があります。

InterSystems IRIS インスタンスがスタンドアロン構成または ECP データサーバである場合、レポートには“Total”列のみが表示されます。ECP アプリケーション・サーバの場合 (つまり、リモート・データベースに接続している場合) は、“Local”、“Remote”、“Total”という 3 つの列が表示されます (例 C)。

以下のテーブルは、^GLOSTAT 統計を定義します。

テーブル 5-1: ^GLOSTAT が生成する統計

統計	定義
グローバル参照 (すべて)	グローバルへのアクセスの論理カウント (式内のグローバル参照、Set、Kill、$Data、$Order、$Increment、$Query)。
グローバル更新参照	グローバル参照 (Set、Kill、$Increment) の論理カウント。
プライベート・グローバル参照	すべてのプロセス・プライベート・グローバル・アクセスのカウント。
プライベート更新参照	プロセス・プライベート・グローバル参照 (SET または KILL など) のカウント。
ルーチン呼び出し	ルーチンへの呼び出し数。
ルーチン・バッファの読み込みと保存	ZLoad、ZSave、および実行中のルーチンの結果として、ルーチンの読み込みと保存が行われた合計数 (適切に調整された環境の場合、ほとんどのルーチンはルーチン・キャッシュ・メモリに既に格納されており、ディスクにアクセスする必要がありません。したがって、この数値は緩やかに増加します。1 回のルーチンの読み込みまたは保存では、最大 32 KB (Unicode では 64 KB) のデータが転送されます)。
ルーチン・コマンド	システム起動後に実行されたルーチン・コマンドの数。
キャッシュされないルーチン	メモリにキャッシュされないルーチンの数。この情報は、ルーチン・バッファ・キャッシュが適切なサイズかどうかを判断する際に役立ちます。
論理ブロック要求	グローバル・データベース・コードによって読み取られたデータベース・ブロックの数 (適切に調整された環境では、通常、これらの読み取りはディスクにアクセスしないで実行されます)。
ブロック読み取り	グローバル参照とルーチン参照の両方について、ディスクから読み取られた物理データベース・ブロックの数。
ブロック書き込み	グローバル参照とルーチン参照の両方について、ディスクに書き込まれた物理データベース・ブロックの数。
WIJ 書き込み	ライト・イメージ・ジャーナル・ファイルへの書き込みの数。
キャッシュ効率	全グローバル参照数を、物理ブロックの読み取り数と書き込み数で割った値。百分率 (%) ではありません。
ジャーナル・エントリ	作成されたジャーナル・レコードの数。ジャーナル・レコードは、データベースの修正 (Set、Kill など)、トランザクション・イベント (TStart、TCommit)、ジャーナルに保存されたその他のイベントごとに 1 つ作成されます。

統計	定義
ジャーナル・ブロック書込	ジャーナル・ファイルに書き込まれた 64 KB ジャーナル・ブロックの数。

5.3 ^GLOSTAT の出力例

以下の出力例は、^GLOSTAT ユーティリティ・ルーチンを実行する際のさまざまなオプションを示しています。

- 例 A ー スタンドアロンまたはサーバ構成で最初に実行した場合の出力例です。
- 例 B ー 時間間隔を指定して実行した場合の出力例です。
- 例 C ー クライアント構成で最初に実行した場合の出力例です。

5.3.1 例 A

以下は、^GLOSTAT ルーチンを最初に実行した場合の出力例です。InterSystems IRIS インスタンスは、スタンドアロン構成またはサーバのいずれかです。

```
%SYS>do ^GLOSTAT

Statistics                                                      Total
--------------                                                ---------
Global references (all):                                        530,801
Global update references:                                       175,073
Private global references:                                      160,267
Private update references:                                       76,739
Routine calls:                                                  650,085
Routine buffer loads & saves:                                       570
Routine commands:                                            17,747,411
Routine not cached:                                                 710
Logical block requests:                                         289,166
Block reads:                                                      2,179
Block writes:                                                       680
WIJ writes:                                                         903
Cache Efficiency:                                                   186
Journal Entries:                                                  1,356
Journal Block Writes:                                                 6

Continue (c), Timed Stats (# sec > 0), Quit (q)?
```

5.3.2 例 B

以下の出力例は、指定した時間間隔（30 秒）における 1 秒あたりの ^GLOSTAT 統計データを示しています。InterSystems IRIS インスタンスは、スタンドアロン構成またはサーバのいずれかです。

```
Continue (c), Timed Stats (# sec > 0), Quit (q)?  30

Counts per Second for 30 Seconds...

Statistics (per second)                                         Total
--------------                                                ---------
Global references (all):                                          4.0
Global update references:                                         2.0
Private global references:                                        2.0
Private update references:                                        0.9
Routine calls:                                                    8.8
Routine buffer loads & saves:                                       0
Routine commands:                                                 222.2
Routine not cached:                                                 0
Logical block requests:                                           2.3
Block reads:                                                        0
Block writes:                                                       0
```

```
WIJ writes:                                                                  0
Cache Efficiency:                                                       no i/o
Journal Entries:                                                             0
Journal Block Writes:                                                        0

Continue (c), Timed Stats (# sec > 0), Quit (q)?
```

5.3.3 例 C

以下は、^GLOSTAT ルーチンを最初に実行した場合の出力例です。InterSystems IRIS インスタンスはクライアントです。

```
%SYS>do ^GLOSTAT

 Statistics                            Local          Remote           Total
-------------                      ---------      ---------      ---------
Global references (all):             123,783              3        123,786
Global update references:              6,628              0          6,628
Private global references:             3,558            n/a          3,558
Private update references:             1,644            n/a          1,644
Routine calls:                        55,275              0         55,275
Routine buffer loads & saves:            759              0            759
Routine commands:                                                 1,304,213
Routine not cached:                                                     167
Logical block requests:               83,959            n/a         83,959
Block reads:                           2,125              0          2,125
Block writes:                            217            n/a            217
WIJ writes:                              126            n/a            126
Cache Efficiency:                         53         no gets
Journal Entries:                         511            n/a            511
Journal Block Writes:                      3            n/a              3

Continue (c), Timed Stats (# sec > 0), Quit (q)?
```

6

^PERFMON を使用したシステム・パフォーマンスの監視

^PERFMON は MONITOR 機能を制御する ObjectScript ユーティリティです。

MONITOR 機能は、イベントの発生回数をシステム・レベルで集計し、プロセス、ルーチン、グローバル、およびネットワーク・ノード別にメトリックを並べ替えることによって、InterSystems IRIS® Data Platform システムのパフォーマンス・データを提供します。これらのデータの収集にはオーバーヘッドが伴うため、カウンタの収集を明確に有効化し、データ収集の対象とするプロセス、グローバル、ルーチン、およびネットワーク・ノードの数を指定する必要があります。MONITOR の起動時に、プロセス、ルーチン、グローバル、およびノードの数に必要なスロットを作成するためのメモリが割り当てられます。イベント・カウンタをトリガする最初のプロセスが最初のスロットを割り当て、そのカウンタ・セットに追加します。利用できるすべてのスロットがプロセスに割り当て済みになると、以降に発生するプロセスのカウントは他のスロットに記録されます。グローバル、ルーチン、およびノードについても同様の手順が実行されます。

収集の実行中でも、データのレポート内容を確認できます。収集を停止すると、メモリの割り当てが解除され、カウンタ・スロットが削除されます。したがって、データを保持するには、レポートをファイル (またはグローバル) に書き込む必要があります。既定では、1 秒あたりの比率としてデータが提示されますが、未加工の合計データを収集するためのオプションもあります。さらに、収集を一時停止/再開したり、カウンタをゼロにするための関数も用意されています。

^PERFMON の実行時に表示されるメニュー項目は、^PERFMON ルーチンで使用できる関数に直接対応しています。また、収集されたデータは、これらの関数のパラメータとしてそのまま使用されます。

同じ MONITOR 機能を制御する類似の関数群が、%Monitor.System パッケージのクラスに用意されています。詳細は、このドキュメントの “システム・モニタの使用” の章の ″アプリケーション・モニタ″ およびこのドキュメントの “^%SYS.MONLBL を使用したルーチン・パフォーマンスの検証” の章を参照してください。

6.1 ^PERFMON の使用

^PERFMON ルーチンは、InterSystems ターミナルでインタラクティブに実行するか、または関数の個別の呼び出しによって実行するかの 2 つの方法で実行できます。どちらかの方法を使用する場合も、^PERFMON のすべてのオプションを使用できます。

^PERFMON には、以下の関数があります。

- Start
- Stop
- Pause

- Resume
- Sample Counters
- Clear
- Report
- Collect

各関数は、成功 (1) または失敗 (負の数値の後にコンマと短いメッセージ) のステータスを返します。

^PERFMON と、行単位の監視ルーチンである ^%SYS.MONLBL は、同一のメモリ割り当てを共有しているため、1 つの InterSystems IRIS インスタンスで一度に実行できるのはいずれか一方のみです。^%SYS.MONLBL で監視を実行しているときに ^PERFMON を実行しようとすると、以下のメッセージが表示されます。

```
The Line-by-line Monitor is already enabled.
This must be stopped before ^PERFMON can be used.
```

6.1.1 ^PERFMON のインタラクティブな実行

以下は、ターミナルで ^PERFMON ルーチンをインタラクティブに実行する例です。

1. %SYS ネームスペースで、以下のコマンドを入力します。

   ```
   do ^PERFMON
   ```

2. 以下のメニューが表示されます。実行する操作の番号を入力します。ルーチンを終了するには Enter キーを押します。

   ```
   1. Start Monitor
   2. Stop Monitor
   3. Pause Monitor
   4. Resume Monitor
   5. Sample Counters
   6. Clear Counters
   7. Report Statistics
   8. Timed Collect and Report

   Monitor is Stopped

   Enter the number of your choice:
   ```

3. 各オプションは ^PERFMON の機能に直接対応しており、必要なパラメータがある場合は入力を求められます。例えば、1 を入力することは Start 機能を使用することに相当します。

   ```
   1. Start Monitor
   2. Stop Monitor
   3. Pause Monitor
   4. Resume Monitor
   5. Sample Counters
   6. Clear Counters
   7. Report Statistics
   8. Timed Collect & Report

   Monitor is Stopped

   Enter the number of your choice: 1


   Processes <24>:
   Routine <200>:
   Globals <100>:
   Databases <10>:
   Network nodes <5>:
   ```

6.1.2 Start

統計情報の収集を有効にします。

形式：

```
status = $$Start^PERFMON(process,routine,global,database,network)
```

パラメータ：

- process ー プロセス向けに確保するスロット数 (既定値は $$pcount (プロセス・テーブル内のプロセス数))
- routine ー ルーチン向けに確保するスロット数 (既定値は 200)
- global ー グローバル向けに確保するスロット数 (既定値は 100)
- database ー データベース向けに確保するスロット数 (既定値は 10)
- network ー ネットワーク・ノード向けに確保するスロット数 (既定値は 5)

^PERFMON をインタラクティブに実行すると、これらパラメータの値を指定するように要求されます。

状態コード：

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-2	モニタが既に実行されています
-3	メモリの割り当てに失敗しました
-4	統計情報の収集を有効にできません

6.1.3 Stop

統計の収集を中止ます。

形式：

```
status = $$Stop^PERFMON()
```

状態コード：

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-2	モニタが実行されていません

6.1.4 Pause

統計の収集を一時的に停止し、データを確認するために一貫性のある状態とします。

形式：

```
status = $$Pause^PERFMON()
```

状態コード：

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-2	モニタが実行されていません
-3	モニタが既に一時停止しています

6.1.5 Resume

一時停止している統計収集を再開します。

形式：

```
status = $$Resume^PERFMON()
```

状態コード：

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-2	モニタが実行されていません
-3	モニタが既に実行されています

6.1.6 Sample Counters

収集の一時停止と再開を連続的に行うジョブを開始して、メトリックを定期的にサンプリングします。wait_time が 0 の場合、バックグラウンド・ジョブは停止し、収集は一時停止します。

形式：

```
status = $$Sample^PERFMON(wait_time,sample_time)
```

パラメータ：

- wait_time ー 収集間隔の秒数 (既定値は 10)
- sample_time ー 収集を継続する秒数 (既定値は 1)

状態コード：

状態コード	説明
1	成功
-2	モニタが実行されていません
-8	サンプル・ジョブは既に実行中です

6.1.7 Clear

すべてのメトリック・カウンタをクリアします。

形式：

```
status = $$Clear^PERFMON()
```

状態コード：

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-2	モニタが実行されていません

6.1.8 Report

レポート機能でメトリックをまとめ、レポートとして出力します。

形式：

```
status = $$Report^PERFMON(report,sort,format,output,[list],[data])
```

パラメータ：

- report ー 出力するレポートの種類。以下の値を指定できます。
  - G – グローバル動作のレポート
  - R – ルーチン動作のレポート
  - N – ネットワーク動作のレポート
  - C – レポートの対象とするメトリックを選択して設定したカスタム・レポート
- sort ー レポートのグループ分けと並べ替えの順序。以下の値を指定できます。
  - P – Process (プロセス) で並べ替え
  - R – Routine (ルーチン) で並べ替え
  - G – Global (グローバル) で並べ替え
  - D – Database (データベース) で並べ替え
  - I – Incoming node (着信ノード) で並べ替え
  - O – Outgoing node (発信ノード) で並べ替え
- format ー 出力形式。以下の値を指定できます。
  - P – 印刷可能または表示可能なレポート (.txt ファイル、ページ付けなし)
  - D – スプレッドシートで読み取り可能なコンマ区切りデータ (.csv ファイル)
  - X – Excel へのインポートに適した Microsoft Excel XML マークアップ (.xml ファイル)
  - H – HTML ページ (.html ファイル)

- output – ファイル名を入力するか、表示された既定のファイル名を受け入れる場合は Return キーを押すか、画面に出力する場合は 0 (ゼロ) を指定します。
- list – (カスタム・レポートの場合のみ) レポートに記録する列を指定するメトリック番号のコンマ区切りリスト。使用可能なすべてのメトリックとその番号をリストする場合は、カスタム・レポートを指定した後に ? と入力します。

  グローバル、ルーチン、またはネットワークの動作を示すレポート (これは report パラメータで指定します) には、このリストの事前定義サブセットが表示されます。
- data – レポートするデータの種類。以下の値を指定できます。
  - 1 – 1 秒あたりの標準比率
  - 2 – 未加工の合計データ

状態コード :

状態コード	説明
1	成功
-1	モニタが実行されていません
-2	入力パラメータがありません
-3	レポートのカテゴリが正しくありません
-4	レポートの編成が正しくありません
-5	レポートの形式が正しくありません
-6	カスタム・レポートのリストが正しくありません
-7	データの形式が正しくありません

"レポートの例" のセクションでは、入力パラメータとしてさまざまな値を入力する例を紹介しています。

6.1.9 Collect

この時間制限付き収集およびレポート関数では、システム・パフォーマンスの自動高速スナップショットを実現します。指定された期間 (既定では 30 秒) メトリックを収集し、5 つの基本レポートと 1 つのプロセス・カウントを作成し、それらをまとめて Excel スプレッドシートまたは HTML ページとしてフォーマット設定します。

形式は以下のようになります。

```
status = $$Collect^PERFMON(time,format,output)
```

パラメータ :

- time – データ収集時間 (秒) (既定値は 30)
- format – 出力形式。以下の値を指定できます。
  - X – Excel へのインポートに適した Microsoft Excel XML マークアップ (.xml ファイル)
  - H – HTML ページ (.html ファイル)
- output – ファイル名を入力するか、表示された既定のファイル名を受け入れる場合は Return キーを押すか、画面に出力する場合は 0 (ゼロ) を指定します。

状態コード :

状態コード	説明
1	成功
-1	他のユーザがモニタを使用しています
-3	モニタが既に実行されています

6.2 レポートの例

以下の例では、グローバルの統計を収集してグローバル名で並べ替え、マネージャのディレクトリにある perfmon.txt というファイルに出力するレポートを実行します。

```
%SYS>Do ^PERFMON


1. Start Monitor
2. Stop Monitor
3. Pause Monitor
4. Resume Monitor
5. Sample Counters
6. Clear Counters
7. Report Statistics
8. Timed Collect & Report

Enter the number of your choice: 7


Category may be: G=Global, R=Routine, N=Network or C=Custom
Category ('G', 'R', 'N' or 'C'): g
Sort may be: P=Process, R=Routine, G=Global, D=Database, I=Incoming or O=Outgoing node
Sort ('P', 'R', 'G', 'D', 'I' or 'O'): g
Format may be: P=Print, D=Delimited data, X=Excel XML, H=HTML
Format ('P', 'D', 'X', 'H'): p
File name: perfmon.txt



Press RETURN to continue ...
```

以下の例では、メトリック番号が 5 10 15 20 25 30 35 40 45 50 に該当する統計を扱うカスタム・レポートを実行します。収集されたカウントはプロセス ID で並べ替えられ、管理者のディレクトリにある perfmonC.txt というファイルに出力されます。

```
1. Start Monitor
2. Stop Monitor
3. Pause Monitor
4. Resume Monitor
5. Sample Counters
6. Clear Counters
7. Report Statistics
8. Timed Collect & Report

Enter the number of your choice: 7


Category may be: G=Global, R=Routine, N=Network or C=Custom
Category ('G', 'R', 'N' or 'C'): c
List of field numbers: 5,10,15,20,25,30,35,40,45,50
Sort may be: P=Process, R=Routine, G=Global, D=Database, I=Incoming or O=Outgoing node
Sort ('P', 'R', 'G', 'D', 'I' or 'O'): p
Format may be: P=Print, D=Delimited data, X=Excel XML, H=HTML
Format ('P', 'D', 'X', 'H'): p
File name: perfmonC.txt
```

7

^PROFILE を使用したルーチン・パフォーマンスの監視

^PROFILE ユーティリティは、プログラマがアプリケーションのルーチンおよびクラスのパフォーマンスを分析する際に利用できます。このタスクは、以下のように 2 フェーズ構成で実行されます。

1. データを収集し、ルーチン・レベルで並べ替え、最も “機能” しているルーチンを特定できます。
2. 行レベルの詳細でデータ (サブルーチン、プロシージャ、および個別の行) の収集および表示が必要なルーチンを選択して、パフォーマンス上の問題が疑われる個別のルーチンへの “ドリル・ダウン” を可能にします。

既定では、^PROFILE は、最大で 5000 のルーチンのメトリックを取得します。最大数のルーチンに使用できる十分な共有メモリがない場合、このユーティリティは、このコレクションを監視するのに必要なメモリのページ数と使用可能なページ数に関するメッセージを表示します。その後、可能な限り多くのルーチンのメトリックを取得します。

7.1 ^PROFILE の使用法

%SYS ネームスペースから (^PROFILE) ユーティリティを起動します。

```
%SYS>do ^PROFILE
```

データ収集の開始を求められたら、Enter を押します。

注釈　　応答 (Yes および No 以外) を求められた場合は、「?」を入力してヘルプを表示できます。

既定では、プロファイルによってルーチンおよび以下のメトリックの番号付きリストが表示されます。初期設定の場合、このリストは RtnLine メトリックを基準にして並べ替えられます。

列タイトル (メトリック)	説明
RtnLine	実行されたコードのルーチン行数。既定では、実行された全行数に対するパーセント値で表示されます。
Time	ルーチンの実行に要した時間。既定では、すべてのルーチンの実行の合計所要時間に対するパーセント値で表示されます。
CPU	ルーチンの実行に要した CPU 時間。既定では、すべてのルーチンの実行の合計 CPU 時間に対するパーセント値でエントリが表示されます。

列タイトル (メトリック)	説明
RtnLoad	ルーチンがロードされた回数。既定では、すべてのルーチンのロードに対するパーセント値でエントリが表示されます。
GloRef	ルーチンのグローバル参照数。既定では、すべてのルーチンのグローバル参照数に対するパーセント値でエントリが表示されます。
GloSet	ルーチンのグローバル SET 数。既定では、すべてのルーチンのグローバル SET 数に対するパーセント値でエントリが表示されます。

ルーチン名 (INT または MVI ファイル)、および実行されるネームスペースは、エントリの 2 行目に表示されます。

ターミナルに表示された指示に従います。

- プロファイル・レベルでルーチンのリストが表示される場合は、以下のいずれかを指定できます。

オプション	説明
#	詳細なプロファイル・レベルのデータ収集が指定された行 (複数可) を示すフラグを指定します。 注釈　表示されるページのそれぞれに、個別の行数 (#)、コンマで区切られたリスト (#,#,#)、範囲 (#-#)、または組み合わせ (#-#,#,#-#,#) を入力できます。 いずれかのページでルーチンを選択した後は、前後のページに移動して別のルーチンを選択できます。分析対象にするすべてのルーチンを選択したら、Q を入力して、詳細レベルのプロファイル収集を開始します。
B	リストの前のページを表示します。
E	表示されたメトリックのコレクションをエクスポートします。
N	リストの次のページを表示します。
O	別のメトリックを基準にしてページを並べ替えます (選択したメトリックが最初の列に表示されます)。
Q	^PROFILE ユーティリティを終了します。 注釈　分析対象のルーチンを示すフラグを指定した場合は、このオプションを指定することでサブルーチン・レベルおよび行レベルのメトリックを収集するか、終了するかを選択できます。
R	最新のメトリックでリストを更新します。
X	選択したルーチンのすべて (別のページで選択したルーチンも含む) のフラグをクリアし、メトリックの収集を更新します。

- 詳細プロファイリング・レベルでルーチンのリストが表示される場合は、以下のいずれかを指定できます。

オプション	説明
#	詳細に分析する必要があるルーチンの行番号。Enter を押すと、選択したルーチンのサブルーチン・ラベルが表示されます。
B	リストの前のページを表示します。
N	リストの次のページを表示します。
O	別のメトリックを基準にしてページを並べ替えます (選択したメトリックが最初の列に表示されます)。
Q	^PROFILE ユーティリティを終了します。
R	最新のメトリックでリストを更新します。

- サブルーチン・ラベル (および各ラベルのメトリック) のリストが表示される場合は、以下のいずれかを指定できます。

オプション	説明
#	詳細に分析する必要があるサブルーチン・ラベルの行番号 (コード内)。Enter を押すと、指定したラベルのコードが表示されます。
B	リストの前のページを表示します。
L	サブルーチンの行レベル表示に切り替えます。
N	リストの次のページを表示します。
Q	リストを終了し、前のレベルに戻ります。
R	最新のメトリックでリストを更新します。 注釈　　リストに *Unknown* が表示された場合は、R を入力します。

- コードの行が表示されると、次の操作を指定するように求められます。使用できるオプションは以下のとおりです。

オプション	説明
#	詳細に分析する必要があるコード内の行番号。Enter を押すと、指定したラベルのコードが表示されます。
B	リストの前のページを表示します。
C	コードの表示をソース・コードと中間 (INT/MVI) コード間で切り替えます。
M	ページのマージンまたは長さを変更します。
N	リストの次のページを表示します。
O	別のメトリックを基準にしてページを並べ替えます。
Q	リストを終了し、前のレベルに戻ります。

オプション	説明
R	最新のメトリックでリストを更新します。
S	ルーチンのサブルーチン・レベルの表示に切り替えます。

7.2 ^PROFILE の例

以下は、ターミナルで ^PROFILE ユーティリティを (%SYS ネームスペースから) インタラクティブに実行する例です。

1. 以下のコマンドを入力します。

```
do ^PROFILE
```

2. 以下のメッセージが表示されます。

```
WARNING: This routine will start a system-wide collection of
data on routine activity and then display the results. There
may be some overhead associated with the initial collection,
and it could significantly affect a busy system.

The second phase of collecting line level detail activity
has high overhead and should ONLY BE RUN ON A TEST SYSTEM!

Are you ready to start the collection?  Yes =>
```

3. メトリックの収集を開始するには Enter を押します。以下のようなメトリックが表示されます。

```
 Waiting for initial data collection ...

      RtnLine     Time        CPU         RtnLoad     GloRef      GloSet
1.    41.48%      12.19%      0.00%       28.97%      10.65%      0.00%
      %Library.ResultSet.1.INT (IRISLIB)
2.    35.09%      56.16%      65.22%      9.35%       36.77%      42.55%
      SYS.Database.1.INT (IRISSYS)
3.    10.75%      6.62%       0.00%       43.30%      22.68%      46.81%
      Config.Databases.1.INT (IRISSYS)
4.    7.13%       3.22%       0.00%       6.23%       0.00%       0.00%
      %Library.Persistent.1.INT (IRISLIB)
5.    1.26%       0.71%       0.00%       4.36%       4.12%       4.26%
      PROFILE.INT (IRISSYS)
6.    1.20%       0.00%       0.00%       0.00%       5.15%       6.38%
      %SYS.WorkQueueMgr.INT (IRISSYS)
7.    0.76%       15.08%      34.78%      0.00%       0.00%       0.00%
      %SYS.API.INT (IRISSYS)
8.    0.64%       1.05%       0.00%       0.00%       17.18%      0.00%
      %Library.JournalState.1.INT (IRISLIB)
9.    0.61%       0.31%       0.00%       3.74%       0.00%       0.00%
      %Library.IResultSet.1.INT (IRISLIB)
10.   0.28%       0.93%       0.00%       0.00%       1.72%       0.00%
      %Library.Device.1.INT (IRISLIB)
11.   0.24%       0.71%       0.00%       0.62%       0.00%       0.00%
      Config.CPF.1.INT (IRISSYS)
Select routine(s) or '?' for more options  N =>
```

4. 詳細に分析するルーチンに関連付けられた番号を入力します。例えば、2-3,5,7,10 と入力してから N または B を入力すると、別のページを表示して、さらにルーチンの選択を追加できます。

5. 分析対象にするすべてのルーチンを選択して、Q を入力すると、以下のようなメッセージが表示されます。

```
There are 2 routines selected for detailed profiling. You may now
end the routine level collection and start a detailed profiler collection.

WARNING !!

This will have each process on the system gather subroutine level and line
level activity on these routines. Note that this part of the collection may
have a significant effect on performance and should only be run in a test
or development instance.

Are you ready to start the detailed collection?  Yes =>
```

6. Enter を押すと、以下のようなページが表示されます。

```
Stopping the routine level Profile collection ...

Loading ^%Library.Persistent.1 in ^^c:\intersystems\iris\mgr\irislib\

Detail level Profile collection started.




    RtnLine     Routine Name  (Database)
1.  96.72%      %Library.Persistent.1.INT (IRISLIB)
2.  3.28%       Config.CPF.1.INT (IRISSYS)
Select routine to see details or '?' for more options  R =>
```

7. コードを分析するルーチンを選択すると、コードに関する情報を含むページが表示されます。

8

^%SYS.MONLBL を使用したルーチン・パフォーマンスの検証

^%SYS.MONLBL ルーチンは、InterSystems IRIS® Data Platform の MONITOR 機能を実行するためのユーザ・インタフェースです。このユーティリティを使用すれば、ルーチンで選択したコードの実行時間を調べ、リソースを特に消費するコード行を特定することができます。これは、^PERFMON および %Monitor.System パッケージ・クラスを使用してアクセスできる既存の MONITOR ユーティリティを拡張したものです。これらのユーティリティは同じメモリ割り当てを共有するので、InterSystems IRIS インスタンスでは一度に 1 つのユーティリティしか実行できません。

8.1 行単位の監視ルーチンの呼び出し

モニタを実行していない状態で ^%SYS.MONLBL を呼び出すと、警告メッセージが表示され、モニタを起動するか、メモリ要件を確認するためのオプションが提示されます。以下に例を示します。

```
%SYS>Do ^%SYS.MONLBL

WARNING ! Starting the line-by-line monitor will enable the
  collection of statistics for *every* line of code executed by
  the selected routines and processes. This can have a significant
  impact on the performance of a system, and it is recommended
  that you do this on a 'test' system.

  The line-by-line monitor also allocates shared memory to track
  these statistics for each line of each routine selected. This is
  taken from the general shared memory already allocated and
  should be considered if you are using '*' wildcards and trying to
  analyze a large number of routines. If the monitor fails to start due
  to a problem with memory allocation, you may need to increase the
  Generic Memory Heap (gmheap) parameter in the system configuration. You may use
  the 'Memory Requirements' option to see how much memory a collection
  would need (without starting the collection).

1.) Start Monitor
2.) Memory Requirements

Enter the number of your choice:
```

- 「1」を入力すると、監視の開始に必要な情報を指定するためのダイアログが表示されます。
- 「2」を入力すると、実際に監視を開始する前に、収集に必要なメモリ量が予測計算されます。詳細は、"メモリ要件の予測計算" を参照してください。
- ルーチンを終了するには Enter キーを押します。

8.1.1 監視の開始

監視するルーチンとプロセス、および収集するメトリックを選択できます。収集のこれらの特性は、監視を停止するまで継続します。以下の順序で、必要な監視収集情報をルーチンに指定してください。

1. `Routine Names` – 監視するルーチン名のリストを入力します。ここで選択できるのは、現在のネームスペースからアクセスできるルーチンだけです。ルーチン名を入力するとき、その先頭には ^ を記述しないでください。名前では、大文字小文字が区別されます。必要であれば、アスタリスク (*) ワイルドカードを使用して、複数のルーチンを選択できます。最後のルーチン名を入力したら、Enter を 2 回押してリストを終了します。
2. `Select Metrics to monitor` – 選択したメトリックのタイプの番号を入力します。規定値は `1` の最小メトリックです。

```
Select Metrics to monitor
  1) Monitor Minimal Metrics
  2) Monitor Lines (Coverage)
  3) Monitor Global Metrics
  4) Monitor All Metrics
  5) Customize Monitor Metrics

Enter the number of your choice: <1>
```

各オプションに含まれるメトリックの説明は、以下のとおりです。

- 最小メトリック – 以下のテーブルに示したメトリックを監視します。

メトリック	説明
`Metric#: 34 - RtnLine`	ルーチン行が実行された回数
`Metric#: 51 - Time`	その行の実行に要した時間
`Metric#: 52 - TotalTime`	その行によって呼び出されたサブルーチンの実行時間を含めた、その行の合計所要時間

時間メトリックは所要時間であり、秒単位で測定されます。

注釈　再帰コードの合計時間
ルーチンに再帰コードが含まれている場合、同じサブルーチンを繰り返し呼び出す行の `TotalTime` カウンタでは、最も外側の呼び出しの時間のみが記録されますが、それは、ほとんどの場合、再帰ループを実行する実際の時間になります。以前の InterSystems IRIS リリースでは、同じコード報告時間の複数の繰り返しについて時間が累積されており、大きすぎると思われる場合がありました。

- 行 – ルーチン行が実行された回数を監視します (`Metric#: 34 - RtnLine`)。
- グローバル・メトリック – 複数のグローバル・メトリックを監視します (`Metric# 1-26 34-36 51 52`)。
- すべてのメトリック – すべての利用可能なメトリックを監視します。

- メトリックのカスタマイズ – 監視するメトリックのカスタマイズしたリストを作成できます。%Monitor.System パッケージ・クラスがサポートしている標準パフォーマンス・メトリックを選択できます。メトリック項目番号を尋ねるプロンプトで疑問符 (?) を入力すると、使用可能なすべてのメトリックが表示されます。以下に例を示します。

```
Enter the number of your choice: <1> 5

Enter metrics item number (press 'Enter' to terminate, ? for list)

Metric#: ?
1.) GloRef: global refs
2.) GloSet: global sets
.
.
.
34.) RtnLine: lines of ObjectScript
.
.
.
51.) Time: elapsed time on wall clock
52.) TotalTime: total time used (including sub-routines)
Metric#:
```

この例ではすべてのメトリックを示しません。ルーチンを実際に実行するときに、最新のメトリック・リストを取得してください。メトリック・リストを取得する方法は、"行単位の監視プログラミング・インタフェース" のセクションを参照してください。

注釈　すべての収集について、ルーチン行数および時間 (最小メトリック) が、常に収集されます。

3. `Select Processes to monitor` – メニューに表示されるとおりに、選択した番号を入力します。既定値は、1 のすべてのプロセスについてです。

```
Select Processes to monitor
  1.) Monitor All Processes
  2.) Monitor Current Process Only
  3.) Enter list of PIDs

Enter the number of your choice: <1>
```

現時点では、^%SYS.MONLBL には PID を選択するためのリストも手段もありません。ただし、特定のプロセス ID 番号を見つけるために、^%SS ユーティリティまたは管理ポータルの [**プロセス**] ページ ([**システム処理**] > [**プロセス**]) を使用できます。

```
Enter the number of your choice: <1> 3

Enter PID (press 'Enter' to terminate)

PID: 1640
PID: 2452
PID:
```

最後のプロセス ID を入力したら、Enter を 2 回押してリストを終了します。

^%SYS.MONLBL に必要な情報を指定すると、ルーチンの各行のカウンタに共有メモリの特殊部分が割り当てられます。さらに、選択したプロセスに対して、監視が有効になったことが通知されます。

注釈　共有カウンタは複数のプロセスで同時に更新され、また実行中プロセスは正確に同時にカウントを開始できない場合があるので、カウンタの精度が若干失われ、カウントが予想を下回ることがあります。

```
Monitor started.

Press RETURN to continue ...
```

行単位の監視を開始すると、より広範なメニューが表示されます。"行単位の監視オプション" のセクションで、この拡張メニューの各オプションについて説明します。

8.1.2 メモリ要件の予測計算

監視プロセスを開始する前に、このユーティリティを使用して、収集に必要なメモリ量を予測計算できます。通常、いくつかのルーチンを監視するために利用できる共有メモリは十分あります。ただし、数百以上のルーチンを監視する場合は、このオプションがメモリ要件の判断に役立ちます。

ルーチンおよびメトリックのプロンプトは、[Start Monitor] オプションのプロンプトと同じです。監視するルーチンと収集するメトリックを選択すると、ユーティリティによって、この収集の監視に必要なメモリのページ数と使用可能なページ数が表示されます。また、必要に応じて、generic memory heap パラメータのサイズを増加するよう指示されます。

[gmheap] (一般メモリ・ヒープ) の設定は、管理ポータルの [**メモリ詳細設定**] ページ ([**システム管理**]→[**構成**]→[**追加設定**]→[**詳細メモリ**]) で管理できます。

以下に、JRN で始まるすべてのルーチンについて 8 つの選択されたメトリックを監視するためのメモリ要件を予測計算する例を示します。

```
Enter the number of your choice: 2

Enter routine names to be monitored on a line by line basis.
Patterns using '*' are allowed.
Enter '?L' to see a list of routines already selected.
Press 'Enter' to terminate input.

Routine Name: JRN*                        (22 routines added to selection.)
Routine Name:

Select Metrics to monitor
  1) Monitor Minimal Metrics
  2) Monitor Lines (Coverage)
  3) Monitor Global Metrics
  4) Monitor All Metrics
  5) Customize Monitor Metrics

Enter the number of your choice: <1> 5

Enter metrics item number (press 'Enter' to terminate, ? for list)

Metric#: 1 - GloRef
Metric#: 2 - GloSet
Metric#: 3 - GloKill
Metric#: 25 - JrnEntry
Metric#: 34 - RtnLine
Metric#: 35 - RtnLoad
Metric#: 51 - Time
Metric#: 52 - TotalTime
Metric#:

9 page(s) of memory required.
82 page(s) of memory available.

The GenericHeapSize parameter can be increased if more memory is needed.
Pages are each 64kb of memory.

Press RETURN to continue ...
```

選択した収集に必要な場合はメモリを調節した後に、元のメニューから監視を開始するよう選択できます。

8.2 行単位の監視オプション

監視実行中に ^%SYS.MONLBL を呼び出すと、以下のメニュー・オプションが表示されます。

```
Line-by-Line Monitor

1.) Stop Monitor
2.) Pause Monitor / Resume Monitor
3.) Clear Counters
4.) Report - Detail
5.) Report - Summary
6.) Report - Delimited (CSV) Output
7.) Report - Procedure Level

Enter the number of your choice:
```

最初の 3 つのオプションは、一目瞭然です。

- `Stop Monitor` ー すべての ^%SYS.MONLBL 監視を停止します。カウンタ・メモリの割り当てを解除し、収集されたデータを削除します。
- `Pause Monitor` ー 収集を一時停止します。収集したデータはすべて保存されます。保存されている収集データを表示すれば、レポートの生成時にカウントが変更されていないことを確認することができます。このオプションは、監視を実行している場合のみ表示されます。

  `Resume Monitor` ー 一時停止していた収集を再開します。このオプションは、監視を一時停止した場合のみ表示されます。
- `Clear Counters` ー 収集したすべてのデータをクリアします。ただし、監視および新しいデータの収集は続行されます。

"行単位の統計レポート" のセクションで、4 つのレポート・オプションを詳細に説明します。

8.2.1 行単位の統計レポート

収集したメトリックの統計をレポートとして得る場合 (オプション 4 ～ 7) は、ルーチンから統計をレポートする方法に関する情報を指定します。

4 つのタイプのレポートから選択できます。

- `Detail` ー 選択したルーチンの行ごとに選択したメトリックのレポートを生成します。レポートには、パフォーマンス欄ごとの合計が累積されて表示されます。
- `Summary` ー 対象範囲および時間など、選択したルーチンごとの要約情報のレポートを生成します。レポートには、対象範囲の割合の順序でルーチンが表示されます。
- `Delimited (CSV) Output` ー 詳細レポートと同じレポート情報を生成しますが、コンマで区切って出力して、スプレッドシートへのインポートを容易にします。
- `Procedure Level` ー 選択したルーチン内のサブルーチン・レベルで選択したメトリックのレポートを生成します。InterSystems IRIS では、個々のサブルーチン、プロシージャ、および関数のレベルで使用状況のプロファイルリングを行えます。実行に最も時間を要しているサブルーチンを迅速に確認し、パフォーマンスを分析して向上することができます。

選択するレポートのタイプによって、情報の表示方法を選択します。

1. 詳細または要約レポートを選択した場合は、選択した各ルーチンで実行される行の対象範囲分析を含めるかどうかを選択できます。次に例を示します。

```
Enter the number of your choice: 4
Include Coverage Analysis summary (Y/N)? y
```

2. 次に、要約レポートを除くすべてについて、統計を利用できる監視対象のルーチンのリストから 1 つまたは複数のルーチンを選択します。利用できるすべてのルーチンを選択する場合はアスタリスク(*)を入力します。次に例を示します。

```
The following routines have been executed during the run,
and have detail statistics available for them.
1) JRNDUMP
2) JRNOPTS
3) JRNSTART
4) JRNSWTCH
5) JRNUTIL
6) JRNUTIL2

Enter list of routines, or * for all
Routine number (*=All)? * - All
```

3. ルーチン名を入力している場合は、最後のルーチンを入力した後、Enter をもう一度押してリストを終了します。次に例を示します。

```
Enter list of routines, or * for all
Routine number (*=All)? 1 - JRNDUMP
Routine number (*=All)? 2 - JRNOPTS
Routine number (*=All)? 5 - JRNUTIL
Routine number (*=All)?
FileName:
```

4. 出力するファイルの名前またはディレクトリの完全パスを入力できます。何も入力せずに Enter を押すと、ターミナル上にレポートが表示されます。

   パスではなくファイル名を入力すると、%SYS.MONLBL によって現在のネームスペースの既定のグローバル・データベースのディレクトリにファイルが作成されます。例えば、USER ネームスペースで %SYS.MONLBL を実行する場合、次のようになります。

```
FileName: monlbl_JRN_dtl.txt
```

   install-dir¥mgr¥user に monlbl_JRN_dtl.txt という名前で、レポートのファイルが作成されます。

5. Enter を押して、選択した形式で、収集しているメトリックのレポート作成を開始します。

"行単位の監視レポートのサンプル" のセクションで、各レポート・オプションの例を示します。

8.3 行単位の監視レポートのサンプル

このセクションでは、^%SYS.MONLBL ルーチンが生成する各種レポートのサンプルを示しています。

8.3.1 行単位の詳細レポート

以下に、対象範囲分析など、選択したジャーナル・ユーティリティの最小メトリックの詳細のレポートの例を示します。レポートは、monlbl_JRN_dtl.txt ファイルに送信され、その一部が表示されます。

```
Line-by-Line Monitor

1.) Stop Monitor
2.) Pause Monitor
3.) Clear Counters
4.) Report - Detail
5.) Report - Summary
6.) Report - Delimited (CSV) Output
7.) Report - Procedure Level

Enter the number of your choice: 4
Include Coverage Analysis summary (Y/N)? y

The following routines have been executed during the run,
and have detail statistics available for them.
1) JRNDUMP
2) JRNOPTS
3) JRNSTART
4) JRNSWTCH
5) JRNUTIL
6) JRNUTIL2

Enter list of routines, or * for all
Routine number (*=All)? 1 - JRNDUMP
Routine number (*=All)? 2 - JRNOPTS
Routine number (*=All)? 5 - JRNUTIL
Routine number (*=All)?
FileName: monlbl_JRN_dtl.txt


Press RETURN to continue ...
```

選択したルーチンの各行について、行番号、各メトリックのカウント、そのコード行のテキスト(ソース・コードを使用できる場合)がレポートとして表示されます。対象範囲分析を要求した場合、それは選択した各ルーチンの後に表示されます。

```
Routine ^JRNDUMP ...

Line    RtnLine       Time  TotalTime
1             0   0          0        JRNDUMP ;dump the contents...
2             0   0          0         /*
.
.
.
85            0   0          0         n (l,usecluster)
86            3   0.000016   0.000016  i +$g(usecluster) d showlistclu(.l) q
87            3   0.000008   0.000008  s diroff=((3+12+1)+10+1)
88            3   0.000072   0.000072  s i="" f  s i=$o(l(i)) q:i=""  d
89           11   0.001542   0.001542  . w /cup(i+3,1),?3,$S($F(l(i),";")...
90           11   0.028125   0.028220  . w ?(3+12+1),l(i,"info"),?diroff...
91           11   0.000378   0.000895  . w $$GJrnPrefix(l(i))
92            3   0.000027   0.000027  q
93            0   0          0        listjrn(f,list,n) ;list at most...
.
.
.
Total       582  17.258963

Total Lines = 579
Total Lines Hit = 100
Coverage Percentage = 17.27%
```

これは、1 つの選択したルーチンの表示の一部です。

8.3.2 行単位の要約レポート

以下に、対象範囲分析など、選択したジャーナル・ユーティリティの最小メトリックの要約のレポートの例を示します。レポートは、monlbl_JRN_summ.txt ファイルに送信され、その一部が表示されます。

```
Line-by-Line Monitor

1.) Stop Monitor
2.) Pause Monitor
3.) Clear Counters
4.) Report - Detail
5.) Report - Summary
6.) Report - Delimited (CSV) Output
7.) Report - Procedure Level

Enter the number of your choice: 5
Include Coverage Analysis summary (Y/N)? Y
FileName: monlbl_JRN_summ.txt


Press RETURN to continue ...
```

レポートには、選択した各ルーチンが、行、対象範囲、および時間の要約と共に表示されます。対象範囲の割合が最も高いルーチンが、リストの最初に表示されます。

```
Routine              Lines   LinesHit    Percent    RtnLine       Time
JRNOPTS                109         60     55.05%        155  14.172230
JRNSWTCH               249         58     23.29%         69   0.926131
JRNDUMP                579        100     17.27%        582  17.265002
JRNSTART               393         23      5.85%         23   0.005541
JRNUTIL                872         39      4.47%         39   0.116995
JRNUTIL2               276          8      2.90%         56   0.006056
JRNCHECK                18          0      0.00%
JRNCLFOR               416          0      0.00%
JRNCLUREST             193          0      0.00%
JRNCLUREST2            229          0      0.00%
JRNINFO                263          0      0.00%
JRNMARK                195          0      0.00%
JRNRESTB              1315          0      0.00%
JRNRESTC              1245          0      0.00%
JRNRESTC2              540          0      0.00%
JRNRESTCHELP           122          0      0.00%
JRNRESTD               445          0      0.00%
JRNRESTO               859          0      0.00%
JRNROLL                827          0      0.00%
JRNSTAT                 62          0      0.00%
JRNSTOP                119          0      0.00%
JRNWUTL                235          0      0.00%

TOTAL 22 rtns         9561        288      3.01%        924  31.591955
```

これは、完全なサンプル・レポートです。

8.3.3 行単位の区切り出力レポート

この例は、選択したジャーナル・ユーティリティの最小メトリックの区切られた詳細のレポートです。レポートは、monlbl_JRN_csv.txt ファイルに送信され、その一部が表示されます。

```
Line-by-Line Monitor

1.) Stop Monitor
2.) Pause Monitor
3.) Clear Counters
4.) Report - Detail
5.) Report - Summary
6.) Report - Delimited (CSV) Output
7.) Report - Procedure Level

Enter the number of your choice: 6
```

```
The following routines have been executed during the run,
and have detail statistics available for them.
1) JRNDUMP
2) JRNOPTS
3) JRNSTART
4) JRNSWTCH
5) JRNUTIL
6) JRNUTIL2

Enter list of routines, or * for all
Routine number (*=All)? * - All
FileName: monlbl_JRN_csv.txt


Press RETURN to continue ...
```

選択したルーチンの各行について、行、ルーチン名、行番号、各メトリックのカウント、そのコード行のテキスト (ソース・コードを使用できる場合) がレポートとして表示され、すべてコンマで区切られています。ソース・コード行は、引用符で囲まれます。

```
Row,Routine,Line,RtnLine,Time,TotalTime,Code
1,JRNDUMP,1,0,0,0,"JRNDUMP ;dump the contents of a journal file ;
,2,0,0,0," /*"
.
.
.
85,JRNDUMP,85,0,0,0," n (l,usecluster)"
86,JRNDUMP,86,3,0.000016,0.000016," i +$g(usecluster) d showlistclu(.l) q"
87,JRNDUMP,87,3,0.000008,0.000008," s diroff=((3+12+1)+10+1)"
88,JRNDUMP,88,3,0.000072,0.000072," s i="""" f  s i=$o(l(i)) q:i=""""  d"
89,JRNDUMP,89,11,0.001542,0.001542," . w /cup(i+3,1),?3,$S($F(l(i),"";""):$E(l(i),...
90,JRNDUMP,90,11,0.028125,0.028220," . w ?(3+12+1),l(i,""info""),?diroff...
91,JRNDUMP,91,11,0.000378,0.000895," . w $$GJrnPrefix(l(i))"
92,JRNDUMP,92,3,0.000027,0.000027," q"
93,JRNDUMP,93,0,0,0,"listjrn(f,list,n) ;list at most n journal files...
.
```

これは、1 つの選択したルーチンの表示の一部です。

8.3.4 行単位のプロシージャ・レベル・レポート

以下は、サブルーチン関数ごとの選択したジャーナル・ユーティリティの最小メトリックの詳細のレポートの例です。レポートは、monlbl_JRN_proc.txt ファイルに送信され、その一部が表示されます。

```
Line-by-Line Monitor

1.) Stop Monitor
2.) Pause Monitor
3.) Clear Counters
4.) Report - Detail
5.) Report - Summary
6.) Report - Delimited (CSV) Output
7.) Report - Procedure Level

Enter the number of your choice: 7

The following routines have been executed during the run,
and have detail statistics available for them.
1) JRNDUMP
2) JRNOPTS
3) JRNSTART
4) JRNSWTCH
5) JRNUTIL
6) JRNUTIL2

Enter list of routines, or * for all
Routine number (*=All)? * - All
FileName: monlbl_JRN_proc.txt


Press RETURN to continue ...
```

選択したルーチンの各サブルーチンについて、タグ番号、各メトリックのカウント、およびサブルーチン・ラベル (ソース・コードを使用できる場合) がレポートとして表示されます。

```
Routine ^JRNDUMP ...

Tag     RtnLine       Time  TotalTime
1             6   0.000154   0.000154            JRNDUMP
2             0   0          0                   INT
3             0   0          0                   getkey1
4             0   0          0                   progress
5             6   0.000050   0.000050            listhdr
6            21   0.000240   0.000322            showlist
7            20   0.136909   0.330301            listjrn
8             7   0.188435   0.188435            getjrninfo
9             0   0          0                   guijrn
.
.
.
```

これは、1 つの選択したルーチンのレポートの一部です。

8.4 行単位の監視プログラミング・インタフェース

プログラマは %Monitor.System.LineByLine クラスを使用して、InterSystems IRIS MONITOR 機能にアクセスすることもできます。^%SYS.MONLBL の各メニュー・オプションに対応するメソッドが用意されています。例えば、監視を開始する場合は以下のメソッドを呼び出します。

```
Set status=##class(%Monitor.System.LineByLine).Start(Routine,Metric,Process)
```

監視するルーチンとプロセスを選択できます。また、%Monitor.System クラスがサポートしているその他の標準パフォーマンス・メトリックを選択することもできます。メトリック名のリストを取得するには、Monitor.System.LineByLine.GetMetrics() メソッドを使用します。

```
Set metrics=##class(%Monitor.System.LineByLine).GetMetrics(3)
```

パラメータとして 3 を選択すると、使用可能なすべてのメトリック、および各メトリックの簡単な説明が現在のデバイスに出力されます。

監視を停止するには、以下のメソッドを呼び出します。

```
Do ##class(%Monitor.System.LineByLine).Stop()
```

収集したカウントを取得するには、%Monitor.System.LineByLine:Result クエリを使用します。この場合、各行のカウンタは $LIST 形式で返されます。

詳細は、オンラインの "インターシステムズ・クラス・リファレンス" の "%Monitor.System.LineByLine" クラス・エントリを参照してください。

9

^TRACE を使用したプロセス・パフォーマンスのトレース

^TRACE ユーティリティは InterSystems IRIS プロセスの実行をトレースする機能を提供します。トレースされたプロセスは、ルーチン行、イベントの発生場所、および (該当する場合) グローバル参照に関する情報と共に、イベントをトレース・ファイルに書き込みます。

9.1 ^TRACE の使用法

注釈　トレース・ファイルには、グローバル参照やサブルーチンに渡されるパラメータなど、機密情報が含まれる場合があります。これらにはグローバルの値は含まれません。

トレースが可能なイベントは、パフォーマンス監視ツール (^PERFMON、%SYS.MONLBL など) で報告されるメトリックに対応します。未加工のデータは、指定されたディレクトリのトレース・ファイル iristrace_pid.txt に書き込まれます。

注釈　トレース・ディレクトリは、トレースされるプロセスによって書き込み可能である必要があります。

さまざまなトレース・イベントのセットを選択し、さまざまな目的のトレースを生成できます。非常に詳細なアプリケーション実行トレースを実現できます。これには、すべてのグローバル参照 (GloRef)、すべてのアプリケーションのサブルーチン呼び出し (RtnLoad)、実行されたアプリケーション・コードのすべての行 (RtnLines) が含まれます。あるいは、トレースを、物理ブロック読み取り (DataBlkRd、UpntBlkRd など)、ネットワーク・キャッシュ・ミス (NCacheMiss)、ブロック衝突 (BlkWait) などのあまり一般的でないイベントに限定し、アプリケーション内でパフォーマンスに影響を及ぼす可能性のあるイベントの発生場所をすべて見つけることもできます。

注釈　トレースの構成、プロセスのトレースの開始、^TRACE ユーティリティの使用には、**%Admin_Manage:USE** が必要です。

10

^SystemPerformance を使用したパフォーマンスの監視

この章では、^SystemPerformance ユーティリティについて説明します。このユーティリティは、InterSystems IRIS® Data Platform インスタンスおよびインスタンスが稼働するプラットフォームに関する詳細なパフォーマンス・データを収集するためのツールです。このユーティリティで生成されたレポートをインターシステムズのサポート窓口に送信して、システムの問題の診断に役立てることができます。^SystemPerformance は診断レポート (このドキュメントの “診断レポートの使用法” を参照してください) に似ていますが、パフォーマンス・データに重点を置いています。

注釈　このユーティリティは、リリース間で更新されている可能性があります。最新バージョンは、WRC の配布サイトの [ツール] で入手できます。

プロファイルをターミナルで実行することも (この章の ″^SystemPerformance ユーティリティの実行″ を参照)、管理ポータルのタスク・マネージャを使用して実行をスケジュールすることもできます (この章の ″タスク・マネージャを使用した ^SystemPerformance ユーティリティのスケジュール″ を参照)。また、このユーティリティに用意されている API を使用して、プロファイルの追加、変更、および削除ができます。

10.1 ^SystemPerformance ユーティリティの実行

^SystemPerformance ユーティリティでは、実行する 1 つ以上のプロファイルを選択できます。(使用できるプロファイルは、ご使用の製品バージョンおよび実行したカスタマイズによって異なります。)このユーティリティは、選択されたプロファイルに基づいて一連のログ・ファイルを生成し、出力ディレクトリに格納します。既定の出力ディレクトリは、InterSystems IRIS インスタンスの install-dir¥mgr ディレクトリです。この章の ″出力ディレクトリの変更″ のセクションの説明に従って出力ディレクトリを指定することもできます。

既定では、^SystemPerformance には以下のプロファイルが用意されています。

- 12hours — 10 秒おきにサンプリングを行う 12 時間の実行
- 24hours — 10 秒おきにサンプリングを行う 24 時間の実行
- 30mins — 1 秒おきにサンプリングを行う 30 分間の実行
- 4hours — 5 秒おきにサンプリングを行う 4 時間の実行
- 8hours — 10 秒おきにサンプリングを行う 8 時間の実行
- test — 30 秒おきにサンプリングを行う 5 分間のテスト実行

^SystemPerformance ユーティリティを実行するには、以下の手順に従います。

1. ターミナルで、以下のコマンドを入力します (大文字と小文字が区別されます)。このコマンドは、%SYS ネームスペースで実行する必要があります。

```
%SYS>do ^SystemPerformance
```

2. 表示されたメイン・メニューで、実行するプロファイルの番号を入力するか、Enter キーを押して、ユーティリティを終了します。

```
Current log directory: c:\intersystems\iris\mgr\
Windows Perfmon data will be left in raw format.
Available profiles:
     1 12hours - 12-hour run sampling every 10 seconds
     2 24hours - 24-hour run sampling every 10 seconds
     3 30mins  - 30-minute run sampling every 1 second
     4 4hours  - 4-hour run sampling every 5 seconds
     5 8hours  - 8-hour run sampling every 10 seconds
     6 test    - 5-minute TEST run sampling every 30 seconds

select profile number to run:
```

3. 実行するプロファイルを入力すると、収集中のデータに関する情報が表示されます。

```
select profile number to run: 1
Collection of this sample data will be available in 1920 seconds.
The runid for this data is 20111007_1041_30mins.
```

生成されたログ・ファイルは出力ディレクトリに格納されます。これらのファイルは runid によって一意の名前が付けられて識別されます。名前の形式は YYYYMMDD_HHMM_profile_name.log です。YYYYMMDD_HHMM はユーティリティがデータの収集を開始した年月日および時刻、profile_name は選択したプロファイルの名前です。

ユーティリティによるデータ収集の完了後 (つまり、プロファイルで指定された期間の最後に)、読みやすいパフォーマンス・レポートを生成できます。詳細は、この章の "^SystemPerformance パフォーマンス・レポートの生成" を参照してください。

10.1.1 ^SystemPerformance の中止

実行中のプロファイルを停止する場合には、$$Stop^SystemPerformance(runid) コマンドによってデータの収集を中止し、必要に応じてそのプロファイルのすべての .log ファイルを削除できます。例えば、runid `20111220_1327_12hours` で識別されるレポートのデータ収集を中止して、これまでに書き込まれたすべての .log ファイルを削除するには、%SYS ネームスペースのターミナルで以下のコマンドを入力します。

```
do Stop^SystemPerformance("20111220_1327_12hours")
```

ログ・ファイルを削除せずにジョブを停止し、それらのログ・ファイルから HTML パフォーマンス・レポートを生成するには、以下を入力します。

```
do Stop^SystemPerformance("20111220_1327_12hours",0)
```

このコマンドの詳細は、"プログラムによる ^SystemPerformance の実行" のサブセクションにある "$$Stop^SystemPerformance("*runid*")" を参照してください。

注釈　　ジョブを停止し、ファイルを削除する権限が必要です。

10.1.2 プログラムによる ^SystemPerformance の実行

以下のテーブルで説明するように、start 関数、collect 関数、preview 関数、および stop 関数のエントリ・ポイントを使用して、プログラムによって ^SystemPerformance ユーティリティを実行することができます。

注釈　　同時に複数のプロファイルを実行することができます。

コマンド	説明
`$$run^SystemPerformance("profile")`	指定した profile を起動します。成功した場合は runid を、失敗した場合は `0` を返します。
`$$literun^SystemPerformance("profile")`	オペレーティング・システム・データを含まないことを除いて、`$$run^SystemPerformance("profile")` と同じです。 注釈　オペレーティング・システム・データが重複している複数の InterSystems IRIS インスタンスを実行するサーバをこのコマンドは想定しています。
`$$Collect^SystemPerformance("runid")`	指定した runid について、読みやすい HTML パフォーマンス・レポート・ファイルを生成します。成功した場合は `1` とレポートのファイル名を返し、失敗した場合は `0` に続けてキャレット (^) と失敗の理由を返します。
`$$Preview^SystemPerformance("runid")`	指定した runid について、読みやすい HTML 中間 (未完了) パフォーマンス・レポート・ファイルを生成します。成功した場合は、`1` に続けてキャレット (^) とファイルの場所を返します。失敗した場合は、`0` に続けてキャレット (^) と失敗の理由を返します。
`$$Stop^SystemPerformance("runid",[0])`	^SystemPerformance による指定した runid のデータ収集を停止 (中止) し、既定ではこのユーティリティによって生成された関連する .log ファイルを削除します。.log ファイルを削除せずにデータ収集を停止して、それらのログ・ファイルから HTML パフォーマンス・レポートを生成するには、runid の後に `0` のパラメータを含めます。 失敗した場合、この関数は `0` に続けてキャレット (^) と失敗の理由を返します。成功した場合は、1:2:3:4_1:2:3:4 を返します。"成功" ステータスは、アンダースコアで区切られた 2 つの部分 (OS 専用と InterSystems IRIS 専用の部分) で構成されています。それぞれの部分で、コロンで区切られた値によって以下が指定されます。 1. 正常に停止したジョブの数 2. 停止に失敗したジョブの数 3. 正常に削除されたファイルの数 4. 削除されていないファイルの数
`$$waittime^SystemPerformance("runid")`	指定した runid について、最終 HTML ファイルが完成するまでの時間を報告します。runid が完了している場合は `ready now` を返し、完了していない場合は *XX* `hours` *YY* `minutes` *ZZ* `seconds` の形式の文字列を返します。

次の例では、^SystemPerformance ユーティリティによって作成される runid がプログラムによって取得され、完全なレポートまたは中間レポートが生成されたかどうかがテストされ、判定されます。プロファイルが完了していないため完全なレポートが作成されておらず (“0^not ready” が返される)、中間レポートが作成されました (“1” が返される)。この情報を基に、HTML ファイルが生成されていることがわかります。

```
%SYS>set runid=$$run^SystemPerformance("30mins")

%SYS>set status=$$Collect^SystemPerformance(runid)
SystemPerformance run 20181004_123815_30mins is not yet ready for collection.

%SYS>write status
0^not ready

%SYS>set status=$$Preview^SystemPerformance(runid)

%SYS>write status
1^c:\intersystems\iris\mgr\USER_IRIS_20181004_123815_30mins_P1.html
%SYS>
```

10.2 ^SystemPerformance パフォーマンス・レポートの生成

^SystemPerformance ユーティリティは、^SystemPerformance ユーティリティにより生成されるログ・ファイルから、完全で読みやすい HTML パフォーマンス・レポートを自動的に生成します。また、Preview^SystemPerformance エントリ・ポイントを使用して、^SystemPerformance ユーティリティの実行時に選択したプロファイルによって収集されているデータを使用して、中間 (不完全な) レポートを生成することもできます。

生成されたレポート・ファイルは出力ディレクトリに格納されます。このディレクトリは、既定では、InterSystems IRIS インスタンスの install-dir**¥mgr** ディレクトリです。ファイルは名前で一意に識別されます。名前の形式は hostname_instance_runid.**html** です。hostname は InterSystems IRIS のインスタンスが実行されているシステムのホスト名、instance はパフォーマンス・データ収集の対象となったインスタンスの名前、runid は ^SystemPerformance ユーティリティの実行時に生成された一意の識別子です。レポートが中間レポートの場合、ファイル名に _Pn が付加されます。P は暫定レポートであることを示し、n は暫定レポートの番号を示します。

10.3 タスク・マネージャを使用した ^SystemPerformance ユーティリティのスケジュール

このセクションでは、管理ポータルのタスク・マネージャ (**[システム処理]→[タスクマネージャ]**) を使用して、^SystemPerformance の実行をスケジュールする例を示します。タスクをスケジュールする一般的な手順は、″システム管理ガイド″ の “InterSystems IRIS の管理” の章にある ″タスク・マネージャのスケジュール″ で説明されている手順を参照してください。

注釈　この例では、必要なフィールドのみを説明しています。その他のフィールドは必要に応じて編集してください。

例 1：週に一度、24 時間実行

この例では、24 時間、パフォーマンス・データを収集する **24hours** というプロファイルを毎週木曜日午前 9 時に実行するように ^SystemPerformance ユーティリティをスケジュールするタスクを作成します。

1. 管理ポータルの **[タスクマネージャ]** ページ (**[システム処理]→[タスクマネージャ]**) で、**[新規タスク]** オプションを選択して **[タスクスケジューラウィザード]** を起動します。次に、指定されたフィールドに以下の情報を入力します。
   - **[タスク名]** – `24HourRun` と入力します。
   - **[説明]** – `Start 24-hour ^SystemPerformance Run` と入力します。

- **[このタスクを実行するネームスペース]** – ドロップダウン・リストから **[%SYS]** を選択します。
- **[タスクタイプ]** – ドロップダウン・リストから **[レガシータスク実行]** を選択します。

  [実行コード] テキスト・ボックスに以下のコードを入力します。

  ```
  do run^SystemPerformance("24hours")
  ```

- **[出力ファイル]** – 空白のままにします。このタスクには出力がありません (^SystemPerformance 出力ディレクトリのカスタマイズについての詳細は、"出力ディレクトリの変更" を参照)。

2. **[次へ]** をクリックします。次に、指定されたフィールドに以下の情報を入力します。
   - **[頻度]**– ドロップダウン・リストから **[週次]** を選択します。

     [木曜日] チェック・ボックスにチェックを付けます。
   - **[開始日]** – テキスト・ボックスに開始日を入力します。

     [この時刻に一度実行] をクリックし、テキスト・ボックスに「09:00:00」と入力します。
3. **[完了]** をクリックします。

例 2：1 日に一度、2 分間実行

この例では、2 分間、パフォーマンス・データを収集する **2mins** というプロファイルを毎日午後 12 時に実行するように ^SystemPerformance ユーティリティをスケジュールするタスクを作成します。

1. 管理ポータルの **[タスクマネージャ]** ページ (**[システム処理]**→**[タスクマネージャ]**) で、**[新規タスク]** オプションを選択して **[タスクスケジューラウィザード]** を起動します。次に、指定されたフィールドに以下の情報を入力します。
   - **[タスク名]** – `2MinRun` と入力します。
   - **[説明]** – `Start 2-minute ^SystemPerformance Run` と入力します。
   - **[このタスクを実行するネームスペース]** – ドロップダウン・リストから **[%SYS]** を選択します。
   - **[タスクタイプ]** – ドロップダウン・リストから **[レガシータスク実行]** を選択します。

     [実行コード] テキスト・ボックスに以下のコードを入力します。

     ```
     do run^SystemPerformance("2mins")
     ```

   - **[出力ファイル]** – 空白のままにします。このタスクには出力がありません (^SystemPerformance 出力ディレクトリのカスタマイズについての詳細は、"出力ディレクトリの変更" を参照)。
2. **[次へ]** をクリックします。次に、指定されたフィールドに以下の情報を入力します。
   - **[頻度]**– ドロップダウン・リストから **[日次]** を選択します。
   - **[開始日]** – テキスト・ボックスに開始日を入力します。

     [この時刻に一度実行] をクリックし、テキスト・ボックスに「12:00:00」と入力します。
3. **[完了]** をクリックします。

10.4 ^SystemPerformance ユーティリティのカスタマイズ

このセクションでは、API で実行できるタスクについて説明します。

- 出力ディレクトリの変更
- バージョン情報の取得
- プロファイルの操作

10.4.1 出力ディレクトリの変更

ログ・ファイルおよび結果として得られる HTML レポート・ファイルの既定の出力ディレクトリは、^SystemPerformance ユーティリティの実行対象である InterSystems IRIS インスタンスの install-dir¥mgr です。以下のテーブルで説明するコマンドを使用して、既定のディレクトリを変更することができます。

注釈　これらのコマンドは、HTML レポート・ファイルが生成済みであるかどうかにかかわらず、現在実行中のプロファイルには影響を与えません。つまり、現在実行中のプロファイルに関連付けられているファイルは、新しい出力ディレクトリに移動されません。

コマンド	説明
`do setlogdir^SystemPerformance("directory")`	出力ディレクトリのパス名を directory に設定します。directory が存在しない場合は作成されます。 注釈　絶対パス名（例：`C:\Reports`）を指定しなかった場合、そのディレクトリは、install-dir¥mgr の相対パスであると見なされます。
`set x = $$getlogdir^SystemPerformance()`	出力ディレクトリのパス名に等しい変数 x を設定します。
`do clrlogdir^SystemPerformance()`	出力ディレクトリのパス名を既定のディレクトリ (install-dir¥mgr) にリセットします。

10.4.2 バージョン情報の取得

^SystemPerformance ユーティリティの現在のバージョンを調べるには、以下のコマンドを使用します。

- `write $$version^SystemPerformance()`
- `set ver=$$version^SystemPerformance()`

10.4.3 プロファイルの操作

プロファイル定義を操作するには、以下のセクションで説明する API を使用します。

- 新規プロファイルの作成
- プロファイルの編集
- プロファイルのコピー
- プロファイルの削除

10.4.3.1 新規プロファイルの作成

以下の API コマンドを使用して、新しいプロファイルを作成できます。

```
set rc=$$addprofile^SystemPerformance("profilename","description",interval,count)
```

以下を指定する必要があります。

変数	説明
profilename	プロファイルの名前。一意の値でなければなりません。また、スペースや空白文字は使用できません。
description	^SystemPerformance メニューに表示されるプロファイルの説明。
interval	各サンプルを実行する頻度 (秒単位)。1 ～ 300 秒の範囲で指定します。1 秒の間隔は、プロファイルの継続時間が 1 時間以下の場合にのみ選択可能です。
count	プロファイルを実行する回数。

この関数は、成功すると 1、失敗すると 0 を返します。エラーがある場合は、キャレット (^) とエラーの理由が続きます。

例えば、サンプリングが 12 回行われるまで、10 秒ごとに (合計 120 秒、または 2 分間) サンプリングを実行するプロファイル 2minrun を作成するには、次のように入力します。

```
set rc=$$addprofile^SystemPerformance("2minrun","A 2-minute run sampling every 10 seconds",10,12)
```

次回、^SystemPerformance ユーティリティを実行したときには、プロファイルのリストに、以下のプロファイル名と説明が表示されます。

```
2minrun     A 2-minute run sampling every 10 seconds
```

プロファイルの生成

また、以下の API コマンドを使用して、わかりやすい名前と説明が指定された新規プロファイルをすばやく生成することができます。

```
set rc=$$genprofile^SystemPerformance("duration"[,interval])
```

以下は値の説明です。

変数	説明
duration	プロファイルを実行する期間。有効な形式は、"hh:mm"、"hh:"、または mm です。
interval (オプション)	各サンプルを実行する頻度 (秒単位)。1 ～ 300 秒の範囲で指定します。1 秒の間隔は、プロファイルの継続時間が 1 時間以下の場合にのみ選択可能です。

この関数は、成功すると 1 を返します。失敗した場合は、0 に続けてキャレット (^) とエラーの理由を返します。

注釈　duration の最大値は 24 時間 (86,400 秒) です。これよりも長い継続時間を指定すると、^SystemPerformance により 24 時間に引き下げられます。duration の値は、時間を表すコロン (:) を含む場合に限り、二重引用符で囲む必要があります。

interval (指定されている場合) の最小値は 2 秒ですが、継続時間 (つまり、interval * count) が 1 時間よりも短い場合、interval の最小値は 1 秒になります。interval に無効な値を指定すると、^SystemPerformance により最低限必要な値まで引き上げられます。interval が指定されていない場合の既定値は 10 秒です。

例えば、5 分おきに 12 時間、サンプリングを行う 12hours というプロファイル (生成されたプロファイル名と説明を持つもの) を生成するには、以下のように入力します。

```
set rc=$$genprofile^SystemPerformance("12:",300)
```

また、10 分おきに 90 分間サンプリングを実行する 90mins というプロファイルを生成するには、以下のように入力します。

```
set rc=$$genprofile^SystemPerformance(90)
```

次回、^SystemPerformance ユーティリティを実行したときには、プロファイルのリストに、以下のプロファイル名と説明が表示されます。

```
12hours     A 12 hour run sampling every 300 seconds
90mins      A 90 minute run sampling every 10 seconds
```

10.4.3.2 プロファイルの編集

次の API コマンドを使用して、既存のプロファイルを編集できます (ただし、事前定義された “test” プロファイルは例外です)。

```
set rc=$$editprofile^SystemPerformance("profilename","description",[interval],[count])
```

以下は値の説明です。

変数	説明
profilename	編集する既存のプロファイルの名前。
description	^SystemPerformance メニューに表示されるプロファイルの説明。
interval (オプション)	各サンプルを実行する頻度 (秒単位)。1 ～ 300 秒の範囲で指定します。1 秒の間隔は、プロファイルの継続時間が 1 時間以下の場合にのみ選択可能です。
count (オプション)	プロファイルを実行する回数。

この関数は、成功すると 1、失敗すると 0 を返します。エラーがある場合は、キャレット (^) とエラーの理由が続きます。

注釈　引数は位置によって決められます。例えば、count 引数を編集し、interval 引数で指定された値はそのままにしておくには、`set rc=$$editprofile^SystemPerformance("2minrun","A 5-minute run sampling every 30 seconds",,50)` のようにコンマ区切り文字を使用する必要があります。

継続時間が 24 時間 (86,400 秒) を超える場合は、自動的に 24 時間に引き下げられます。

例えば、サンプリングが 10 回行われるまで、30 秒ごとに (合計 300 秒、または 5 分間) サンプリングを実行するように、プロファイル 2minrun を変更するには、次のように入力します。

```
set rc=$$editprofile^SystemPerformance("2minrun","A 5-minute run sampling every 30 seconds",30,10)
```

次回、^SystemPerformance ユーティリティを実行したときには、プロファイルのリストに、以下のプロファイル名と説明が表示されます。

```
2minrun     A 5-minute run sampling every 30 seconds
```

10.4.3.3 プロファイルのコピー

既存のプロファイルを、異なる名前のファイルにコピーするには、以下の API コマンドを使用します。

```
set rc=$$copyprofile^SystemPerformance("sourceprofilename","targetprofilename")
```

以下を指定する必要があります。

変数	説明
sourceprofilename	既存のプロファイルの名前。
targetprofilename	作成するプロファイルの名前。二重引用符で囲む必要があります。

この関数は、成功すると 1 を返します。失敗した場合は、0 に続けてキャレット (^) とエラーの理由を返します。

sourceprofilename は既存のプロファイル名です。targetprofilename は一意の値でなければなりません。また、スペースや空白文字は使用できません。

例えば、2minrun プロファイルのコピーを作成するには、以下のように入力します。

```
set rc=$$copyprofile^SystemPerformance("2minrun","5minrun")
```

次回、^SystemPerformance ユーティリティを実行したときには、プロファイルのリストに、以下のプロファイル名と説明が表示されます。

```
2minrun     A 2-minute run sampling every 30 seconds
5minrun     A 2-minute run sampling every 30 seconds
```

これで、ガイドのこのセクションにある "プロファイルの編集" で説明されているとおり、新規プロファイルを編集できます。

10.4.3.4 プロファイルの削除

次の API コマンドを使用して、既存のプロファイルを削除できます (ただし、事前定義された "test" プロファイルは例外です)。

```
set rc=$$delprofile^SystemPerformance("profilename")
```

以下を指定する必要があります。

変数	説明
profilename	削除するプロファイルの名前。二重引用符で囲む必要があります。

この関数は、成功すると 1 を返します。失敗した場合は、0 に続けてキャレット (^) とエラーの理由を返します。

例えば、2minrun プロファイルを削除するには、以下のように入力します。

```
set rc=$$delprofile^SystemPerformance("2minrun")
```

次回、^SystemPerformance ユーティリティを実行したときには、プロファイルのリストに 2minrun プロファイルは表示されません。

10.5 ^SystemPerformance ユーティリティで作成されるパフォーマンス・レポート

^SystemPerformance ユーティリティは、この章の説明のとおり、プラットフォーム固有のレポートを生成します。このレポートは、以下のリストに示すとおり、複数のセクションに分かれています。

```
Configuration

IRISTEST3 on machine testsystem

Customer: InterSystems Development
License : 123456

InterSystems IRIS Version String: InterSystems IRIS for Windows (x86-32) 2018.1 (Build 508) Fri Jan 26
 2018 17:51:22 EDT
------------------------------------------------------------
Profile

Profile run "test" started at 10:07 on Jun 01 2016.
Run over 10 intervals of 30 seconds.
------------------------------------------------------------
license

Product=Enterprise
License Type=Concurrent User
Server=Multi
Platform=Heterogeneous
Licensed Users=1000
Licensed CPUs=16
.
.
.
------------------------------------------------------------
End of InterSystems IRIS Performance Data Report
```

このセクションのテーブルで、プラットフォーム固有の各レポートのセクションについて説明します。セクションは、具体的なセクションを簡単に検索できるように、各テーブルでアルファベット順にリストされています。1 回だけ収集されるデータには、アスタリスク (*) のフラグが立てられています。残りのデータは、プロファイルの実行中に収集されます。

プラットフォーム固有のデータの詳細は、以下のテーブルを参照してください。

- InterSystems IRIS パフォーマンス・データ・レポート (Microsoft Windows プラットフォーム)
- InterSystems IRIS パフォーマンス・データ・レポート (Apple macOS プラットフォーム)
- InterSystems IRIS パフォーマンス・データ・レポート (IBM AIX® プラットフォーム)
- InterSystems IRIS パフォーマンス・データ・レポート (Linux プラットフォーム)

注釈　以下の表すべてにおいて、* の付いたデータは、実行ごとに 1 回収集されます。

テーブル 10-1: InterSystems IRIS パフォーマンス・データ・レポート (Microsoft Windows プラットフォーム)

セクション	説明
%SS	ALL^%SS コマンドを使用して実行中に取得される 4 つのサンプル。
Configuration *	InterSystems IRIS インスタンス名およびサーバのホスト名、InterSystems IRIS バージョンの完全な文字列、ライセンスが供与された顧客名、およびライセンス注文番号。
cpf file *	現在アクティブな構成ファイルのコピー。

セクション	説明
irisstat -c	コマンド .¥bin¥irisstat -s -p-1 -c-1 -e1 -m8 -n2 -N127 を使用して、実行中に等間隔で取得される 4 つのサンプル。次に各引数について簡単に説明します。 ・ -p-1 : プロセスおよびグローバル状態の情報を含めるためにプロセス・テーブルをサンプリングします。 ・ -c-1 : ジャーナル、ロック、ディスク、およびリソース使用量の統計を表示するために、共有メモリのカウンタ・セクションをサンプリングします。 ・ -e1 : SYSLOG エラー・テーブル。 ・ -m8 : すべての IRIS.DAT ファイルおよびそれらの属性を含むファイル・テーブル。 ・ -n2 : ローカルからリモートへのデータベース・マッピングを含むネットワーク構造テーブル。 ・ -N127 : クライアントとサーバ両方の接続の ECP 統計。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。
irisstat -D	コマンド irisstat cache --f1 -D10,100 を使用して、実行中に等間隔で取得される 8 つのサンプル。次に各引数について簡単に説明します。 ・ -fl : 基本的なフラグ。 ・ -D10,100 : 総サンプリング期間 10 秒の間、100 ミリ秒ごとに発生するブロック衝突のサンプリング。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。^BLKCOL ユーティリティを使用したブロック衝突の監視の詳細は、このドキュメントの “^BLKCOL を使用したブロック衝突の監視” の章を参照してください。
license *	Decode^%LICENSE および counts^%LICENSE を使用した InterSystems IRIS ライセンス使用情報。
mgstat	^mgstat ユーティリティを使用して実行中に取得される InterSystems IRIS 固有のデータ。″監視ガイド″ の ″^mgstat を使用したパフォーマンスの監視″ のセクションを参照してください。

セクション	説明

perfmon	Microsoft Windows perfmon ユーティリティからの出力。
	
	Microsoft Windows perfmon データは、既定では未処理の形式で表示されます。表示形式は切り替えて、繰り返し表示されるサーバ名を削除したり、日時の列を複数の列に分離するなど、見やすくなるように処理を加えることができます。
	
	以下の関数では、perfmon データを操作するかどうか決定するフラグのクエリと更新が可能です。
	
	`set rc=$$setperfmonpostproc^SystemPerformance(<onoroff>)`
	
	onoroff には、1 (オン) または 0 (オフ)、あるいは "on" または "off" (大文字小文字は区別されない) の単語を指定できます。
	
	返りコードの 1 は、フラグの正常な更新を示し、0 は更新の失敗、-1 は非 Windows プラットフォームを示します。
	
	現在の形式 (未処理または処理済み) を判断するには、以下のようにします。
	
	`set status=$$getperfmonpostproc^SystemPerformance()`
	
	返りコードの 1 は、処理済みの形式を示し、0 は未処理の形式を示します。
	
	さらに、フラグの現在の状態は、^SystemPerformance のインタラクティブ実行で、プロファイル・メニューを表示する前に表示されます。
	
	既定で、perfmon は、既定の pbctrs.txt ファイルで指定されたカウンタの定義を監視します。前に定義された perfmon カウンタを監視するには、以下を使用して定義を ^SystemPerformance にインポートします。
	
	`write $$importctrs^SystemPerformance(`*`WindowsCtrName`*`[,`*`SystemPerformanceCtrName`*` [,`*`SystemPerformanceFileName`*`]])`
	
	返りコードの 0 は成功を示し、負の数字とそれに続く理由の文字列は失敗を示します。SystemPerformance カウンタ名の重複は許可されていません。必要に応じて、^SystemPerformance は内部カウンタ名とファイル名の両方を生成します。
	
	既定の SystemPerformance カウンタの定義を既存の定義に変更するには、以下を使用します。
	
	`write $$setctrdefault(`*`SystemPerformanceCtrName`*`)`
	
	返りコードの 1 は成功を示し、0 とそれに続く理由の文字列は失敗を示します。無効なカウンタが指定された場合、組み込みの既定が設定されます。
	
	既定の SystemPerformance カウンタ定義をリセットするには、以下を使用します。
	
	`do clrctrdefault^SystemPerformance()`
	
	特定の SystemPerformance カウンタ定義を既存のプロファイルに関連付けるには、以下を使用します。
	
	`write $$addctrtoprofile(`*`ProfileName`*`,`*`SystemPerformanceCtrName`*`)`
	
	返りコードの 1 は成功を示し、0 とそれに続く理由の文字列は失敗を示します。プロファイルかカウンタ定義のいずれかが存在しない場合、コマンドは実行されません。

セクション	説明
Profile *	このログを作成した ^SystemPerformance プロファイルに関する情報。
tasklist	実行中に等間隔で取得される tasklist -V コマンドの 4 つの出力。tasklist -V コマンドは、システムで実行中のすべてのプロセスのリストを提供します。
Windows info *	Windows バージョン (ホットフィックス情報は除く) やハードウェアの情報を含む、systeminfo コマンドから出力される情報で、プロセッサの個数、インストールされているメモリ、使用されているメモリなどがあります。

テーブル 10-2: InterSystems IRIS パフォーマンス・データ・レポート (Apple macOS プラットフォーム)

セクション	説明
%SS	ALL^%SS コマンドを使用して実行中に取得される 4 つのサンプル。
Configuration *	InterSystems IRIS インスタンス名およびサーバのホスト名、InterSystems IRIS バージョンの完全な文字列、ライセンスが供与された顧客名、およびライセンス注文番号。
cpf file *	現在アクティブな構成ファイルのコピー。
irisstat -c	コマンド irisstat cache -p-1 -c-1 -e1 -m8 -n2 -N127 を使用して、実行中に等間隔で取得される 4 つのサンプル。次に各引数について簡単に説明します。 ・ -p-1 : プロセスおよびグローバル状態の情報を含めるためにプロセス・テーブルをサンプリングします。 ・ -c-1 : ジャーナル、ロック、ディスク、およびリソース使用量の統計を表示するために、共有メモリのカウンタ・セクションをサンプリングします。 ・ -e1 : SYSLOG エラー・テーブル。 ・ -m8 : すべての IRIS.DAT ファイルおよびそれらの属性を含むファイル・テーブル。 ・ -n2 : ローカルからリモートへのデータベース・マッピングを含むネットワーク構造テーブル。 ・ -N127 : クライアントとサーバ両方の接続の ECP 統計。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。
irisstat -D	コマンド irisstat cache --f1 -D10,100 を使用して、実行中に等間隔で取得される 8 つのサンプル。次に各引数について簡単に説明します。 ・ -fl : 基本的なフラグ。 ・ -D10,100 : 総サンプリング期間 10 秒の間、100 ミリ秒ごとに発生するブロック衝突のサンプリング。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。^BLKCOL ユーティリティを使用したブロック衝突の監視の詳細は、このドキュメントの “^BLKCOL を使用したブロック衝突の監視” の章を参照してください。

セクション	説明
ipcs *	共有メモリ、セマフォ、およびメッセージ・キューを含むプロセス間の通信構成情報。ipcs -a コマンドから出力されます。
license *	Decode^%LICENSE および counts^%LICENSE を使用した InterSystems IRIS ライセンス使用情報。
macOS Info *	OS バージョンおよびハードウェアに関する情報。sw_vers、uname -a、mount、netstat の各コマンドから出力されます。
mgstat	^mgstat ユーティリティを使用して実行中に取得される InterSystems IRIS 固有のデータ。"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションを参照してください。
Profile *	このログを作成した ^SystemPerformance プロファイルに関する情報。
ps :	コマンド ps -eflv を使用して、実行中に等間隔で取得される 4 つのサンプル。
sar -d	ディスク (ブロック) デバイスのスループットおよび遅延の統計。
sar -g	ページ・アウト率。
sar -n DEV	ネットワーク・デバイスのスループット。
sar -n EDEV	ネットワーク・デバイスのエラー率。
sar -p	ページ・イン率およびページ・フォルト率。
sar -u	CPU 使用量の統計。
sysctl -a *	カーネルおよびシステム・パラメータの設定。
vm_stat *	メモリ・ページ情報。

テーブル 10-3: InterSystems IRIS パフォーマンス・データ・レポート (IBM AIX® プラットフォーム)

セクション	説明
%SS	ALL^%SS コマンドを使用して実行中に取得される 4 つのサンプル。
AIX info *	oslevel、uname -a、prtconf、lspv の各コマンドから出力されます。
Configuration *	InterSystems IRIS インスタンス名およびサーバのホスト名、InterSystems IRIS バージョンの完全な文字列、ライセンスが供与された顧客名、およびライセンス注文番号。
cpf file *	現在アクティブな構成ファイルのコピー。
cpu type *	取り付けられているプロセッサの情報および SMT が有効かどうか。lsattr -El proc0 から出力されます。

セクション	説明
irisstat -c	コマンド irisstat cache -p-1 -c-1 -e1 -m8 -n2 -N127 を使用して、実行中に等間隔で取得される 4 つのサンプル。次に各引数について簡単に説明します。 ・ -p-1 : プロセスおよびグローバル状態の情報を含めるためにプロセス・テーブルをサンプリングします。 ・ -c-1 : ジャーナル、ロック、ディスク、およびリソース使用量の統計を表示するために、共有メモリのカウンタ・セクションをサンプリングします。 ・ -e1 : SYSLOG エラー・テーブル。 ・ -m8 : すべての IRIS.DAT ファイルおよびそれらの属性を含むファイル・テーブル。 ・ -n2 : ローカルからリモートへのデータベース・マッピングを含むネットワーク構造テーブル。 ・ -N127 : クライアントとサーバ両方の接続の ECP 統計。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。
irisstat -D	コマンド irisstat cache --f1 -D10,100 を使用して、実行中に等間隔で取得される 8 つのサンプル。次に各引数について簡単に説明します。 ・ -fl : 基本的なフラグ。 ・ -D10,100 : 総サンプリング期間 10 秒の間、100 ミリ秒ごとに発生するブロック衝突のサンプリング。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。^BLKCOL ユーティリティを使用したブロック衝突の監視の詳細は、このドキュメントの “^BLKCOL を使用したブロック衝突の監視” の章を参照してください。
df -k *	マウント・ポイント、論理ボリューム、および空き領域を含む、マウントされたファイル・システムに関する情報。df -k コマンドから出力されます。
filesystems *	現在の /etc/filesystems ファイル。
ioo -a *	I/O 調整可能パラメータの現在値。ioo -a コマンドから出力されます。^SystemPerformance プロファイルを起動するユーザに root アクセス権がある場合にのみ含まれます。
iostat -DIT	IBM AIX® 5.3 以降向けの、サンプル時間を含めた拡張ディスク/デバイスの統計の長いリスト。iostat -DIT コマンドから出力されます。 IBM AIX® 5.3 より前のリリースでは情報が異なります。
ipcs *	共有メモリ、セマフォ、およびメッセージ・キューを含むプロセス間の通信構成情報。ipcs -a コマンドから出力されます。
license *	Decode^%LICENSE および counts^%LICENSE を使用した InterSystems IRIS ライセンス使用情報。

セクション	説明
mount *	すべてのファイル・システムおよびそのマウント・オプションに関する情報。
mgstat	^mgstat ユーティリティを使用して実行中に取得される InterSystems IRIS 固有のデータ。"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションを参照してください。
Profile *	このログを作成した ^SystemPerformance プロファイルに関する情報。
ps :	コマンド ps aux を使用して、実行中に等間隔で取得される 4 つのサンプル。
sar -d	^SystemPerformance プロファイルを起動するユーザに root アクセス権があり、/usr/sbin/sar が存在する場合にのみ含まれます。
sar -r	^SystemPerformance プロファイルを起動するユーザに root アクセス権があり、/usr/sbin/sar が存在する場合にのみ含まれます。
sar -u	マイクロパーティショニング情報 (使用している場合) を含む CPU の統計。 ^SystemPerformance プロファイルを起動するユーザに root アクセス権があり、/usr/sbin/sar が存在する場合にのみ含まれます。
vmo -a	仮想メモリ調整可能パラメータの現在値。vmo -a コマンドから出力されます。 ^SystemPerformance プロファイルを起動するユーザに root アクセス権がある場合にのみ含まれます。
vmstat -s *	ページ・インとページ・アウトの合計を含む仮想メモリ統計の絶対数。
vmstat -t	タイムスタンプを含む、仮想メモリおよび CPU (ページング、キューイングおよび CPU) の統計。
vmstat -v *	空きページ、pbuf 使用量、および fsbuf 使用量を含む仮想メモリ統計をサンプリングします。

テーブル 10-4: InterSystems IRIS パフォーマンス・データ・レポート (Linux プラットフォーム)

セクション	説明
%SS	ALL^%SS コマンドを使用して実行中に取得される 4 つのサンプル。
Configuration *	InterSystems IRIS インスタンス名およびサーバのホスト名、InterSystems IRIS バージョンの完全な文字列、ライセンスが供与された顧客名、およびライセンス注文番号。
cpf file *	現在アクティブな構成ファイルのコピー。

セクション	説明
irisstat -c	コマンド irisstat cache -p-1 -c-1 -e1 -m8 -n2 -N127 を使用して、実行中に等間隔で取得される 4 つのサンプル。次に各引数について簡単に説明します。 ・ -p-1 : プロセスおよびグローバル状態の情報を含めるためにプロセス・テーブルをサンプリングします。 ・ -c-1 : ジャーナル、ロック、ディスク、およびリソース使用量の統計を表示するために、共有メモリのカウンタ・セクションをサンプリングします。 ・ -e1 : SYSLOG エラー・テーブル。 ・ -m8 : すべての IRIS.DAT ファイルおよびそれらの属性を含むファイル・テーブル。 ・ -n2 : ローカルからリモートへのデータベース・マッピングを含むネットワーク構造テーブル。 ・ -N127 : クライアントとサーバ両方の接続の ECP 統計。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。
irisstat -D	コマンド irisstat cache --f1 -D10,100 を使用して、実行中に等間隔で取得される 8 つのサンプル。次に各引数について簡単に説明します。 ・ -fl : 基本的なフラグ。 ・ -D10,100 : 総サンプリング期間 10 秒の間、100 ミリ秒ごとに発生するブロック衝突のサンプリング。 irisstat ユーティリティの詳細は、このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” を参照してください。^BLKCOL ユーティリティを使用したブロック衝突の監視の詳細は、このドキュメントの “^BLKCOL を使用したブロック衝突の監視” の章を参照してください。
df -m *	マウント・ポイント、論理ボリューム、および空き領域を含む、マウントされたファイル・システムに関する情報。df -m コマンドから出力されます。
free -m	MB 単位でのメモリ使用量の統計 (-m)。
iostat	CPU およびディスクのスループット。
license *	Decode^%LICENSE および counts^%LICENSE を使用した InterSystems IRIS ライセンス使用情報。
mgstat	^mgstat ユーティリティを使用して実行中に取得される InterSystems IRIS 固有のデータ。"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションを参照してください。
Profile *	このログを作成した ^SystemPerformance プロファイルに関する情報。
ps :	コマンド ps -efly を使用して、実行中に等間隔で取得される 4 つのサンプル。
sar -d	ディスク (ブロック) デバイスのスループットおよび遅延の統計。

セクション	説明
sar -u	CPU 使用量の統計には iowait パーセンテージが含まれます。
vmstat -n	CPU、キューイング、ページングの統計。1 つのヘッダのみが印刷されます (-n)。
CPU *	lscpu と /proc/cpuinfo から収集した情報。
Linux info *	OS およびハードウェアに関する一般情報。uname -a、lsb_release -a、id、および ulimit -a コマンドからの出力、および /etc/issue.net、/proc/partitions、および /dev/mapper から収集された情報が含まれます。
ipcs *	共有メモリ、セマフォ、およびメッセージ・キューを含むプロセス間の通信構成情報。ipcs -a コマンドから出力されます。
mount *	すべてのファイル・システムおよびそのマウント・オプションに関する情報。
fdisk -l *	/proc/partitions に記述されているすべてのデバイスのパーティション・テーブル。^SystemPerformance プロファイルの実行を開始するユーザに root アクセス権がある場合にのみ含まれます。
ifconfig *	現在アクティブなネットワーク・インタフェースのステータス情報。
sysctl -a *	カーネルおよびシステム・パラメータの設定。

11

^mgstat を使用したパフォーマンスの監視

この章では、基本的なパフォーマンス・データを収集するためのツール、^mgstat ユーティリティについて説明します。

注釈　このユーティリティは、リリース間で更新されている可能性があります。ftp://ftp.intersys.com/pub/performance/ からの newmgstat.xml のダウンロードについては、インターシステムズのサポート窓口までお問い合わせください。

^mgstat は %SYS ネームスペースから呼び出す必要があり、以下の位置を示す引数を使用できます。

引数	説明
sample time	この引数は、カウンタのサンプリング周期 (秒単位) を指定します。指定されていない場合、既定は 2 秒です。 注釈　sample time に 10 秒を超える値が指定されている場合、^mgstat により 10 秒に引き下げられます。この表の number of samples 引数も参照してください。
number of samples	この引数は、取得するサンプルの最大数を指定します。指定されていない場合、既定は 10 回の反復です。 注釈　^mgstat により sample time が引き下げられた場合は、いずれの引数も変更されていないときと実行時間 (sample time * number of samples) が事実上同じになるように、指定されている number of samples が引き上げられます。
filename	この引数は、^mgstat で生成される .mgst ファイルの install-dir¥mgr を基準としたファイル名を指定します。指定されていない場合、既定のファイル名は ServerName_InstanceName_Date_Time.mgst です。
page length	^mgstat を対話的に実行する場合、この引数にはヘッダ行が繰り返されるまでに表示する行数を指定します。既定値は 0 で、ヘッダはページの最初に一度だけ表示されます。5 行未満の値 (0 以外) を指定した場合、^mgstat は値を 5 に増やします。 注釈　^mgstat をバックグラウンド・ジョブとして実行する場合、この引数は無視されます。

例えば、^mgstat をバックグラウンド・ジョブとして実行する場合、17,280 個のファイル・サンプルが収集されるまで、5 秒おきにサンプルを取得するように指定するには、(ターミナルを使用して、%SYS ネームスペースから) 以下のように入力します。

```
%SYS>JOB ^mgstat(5,17280)
```

また、^mgstat を対話的に実行して同様のサンプリングを指定し、データが 10 行出力されるたびにヘッダが表示されるようにするには、以下のように入力します。

```
%SYS>DO ^mgstat(5,17280,,10)
```

既定では、^mgstat は server name、configuration name、および date and time に基づいてファイル名を生成します。拡張子は“**mgst**”です。この拡張子は、Microsoft Excel で記述された、データのグラフ化を支援するアナライザ・ツールにより認識されます。既定では、このファイルは InterSystems IRIS® Data Platform インスタンスの install-dir**¥mgr** ディレクトリに格納されます。ただし、^SystemPerformance ユーティリティを使用して出力ディレクトリが変更されている場合 (このドキュメントの“^SystemPerformance を使用したパフォーマンスの監視”の章にある″出力ディレクトリの変更″を参照)、^mgstat はその出力ディレクトリを使用します。

注釈　**mgst** ファイルは、^SystemPerformance ユーティリティを実行した場合にも生成され、HTML パフォーマンス・レポートに含められます (このドキュメントの“^SystemPerformance を使用したパフォーマンスの監視”の章を参照)。

システムのパフォーマンスに対する影響を最小限に抑えるために、^mgstat ユーティリティは共有メモリからさまざまなカウンタ情報を抽出します。このユーティリティを実行中にパフォーマンスの問題の発生が明らかになった場合には、データを使用して問題の調査を実施できます。分析を行う場合は、インターシステムズのサポート窓口にご相談ください。^mgstat の実行とファイルの削除の両方を自動化するタスクが用意されています。

報告されたデータの大半は、後述のテーブルに注記されているものを除き、1 秒単位の値に平均化されます。生成された出力ファイルは、Microsoft Excel などのスプレッドシート・ツールでより簡単に解釈することが可能で読みやすいコンマ区切り値 (CSV) 形式をとります。ファイルの最初の行はヘッダ行で、ファイル名とユーティリティのバージョンが記載されています。また、バッファの割り当てや監視対象の製品のバージョンに関する情報もあります。データの列数は、製品のバージョンによって異なりますが、先頭の 2 列は日付と時刻で、残りの列は以下のとおりです。

列名	説明	留意事項
Glorefs	グローバル参照 (データベース・アクセス)。 現在のワークロードに対して発生する処理の量を示します。グローバル参照は CPU 時間を消費しますが、バッファ・プールがあるので、必ずしも物理的な読み取りを要するわけではありません。	
RemGrefs *	リモート・グローバル参照 (データベース・アクセス)。 分散キャッシュ・クラスタ・アプリケーション・サーバの代わりに生成されるグローバル参照の数を示します。	
GRratio	リモート・グローバル参照に対するグローバル参照の割合。	
PhyRds	ディスクからの物理的な読み取り。 物理的な読み取りの回数が多い場合、パフォーマンスの問題が発生することがあります。その場合は、データベース (グローバル) のバッファの数を増やすことでパフォーマンスを向上させることができます。	

列名	説明	留意事項
Rdratio	論理ブロック読み取りと物理ブロック読み取りの比率。ただし、物理ブロック読み取りがゼロの場合は、ゼロになります。	
Gloupds	グローバル更新 (set または kill)。	
RemGupds *	リモート・グローバル更新。	
Rourefs	ルーチン参照 (tag^routine など)。	
RemRrefs *	リモート・ルーチン参照。	
RouLaS	ルーチンのロード、および保存 (ディスクからのフェッチまたはディスクへの保存)。 ルーチンのロードや保存の回数が多い場合、パフォーマンスの問題が発生することがあります。その場合はルーチン・バッファの数を増やすことでパフォーマンスを向上できます。	
RemRLaS *	リモートでのルーチンのロードと保存。	
PhyWrs	ディスクへの物理書き込み。	
WDQsz	ライト・デーモン・キューのサイズ (ブロック単位)。	単位は秒ではありません。
WDtmpq	IRISTEMP 内の更新されたブロック。	単位は秒ではありません。
WDphase	ライト・デーモンのフェーズ。 最も一般的なフェーズは以下のとおりです。 ・ 0: アイドル状態です (WD は実行されていません)。 ・ 5: WD はライト・イメージ・ジャーナル (WIJ) ファイルを更新しています。 ・ 7: WD は WIJ とジャーナルをコミットしています。 ・ 8: データベースを更新しています。	単位は秒ではありません。
Wijwri	WIJ に書き込まれた 256KB ブロックの数。 WD が WIJ にデータを書き込み中のときは、この値は 0 以外になります。	
RouCMs	ルーチン・キャッシュ・ミスの回数。	
Jrnwrts	ジャーナルに書き込まれたブロックの数。	
GblSz	グローバル・リソース上の Seize の数。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
pGblNsz	グローバル・リソース上の NSeize の割合。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	

列名	説明	留意事項
pGblAsz	グローバル・リソース上の ASeizes の割合。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
RouSz	ルーチン・リソース上の Seize の数。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
pRouAsz	ルーチン・リソース上の ASeizes の割合。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
ObjSz	オブジェクト・リソース上の Seize の数。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
pObjAsz	オブジェクト・リソース上の ASeizes の割合。この章の “Seize、ASeize、および NSeize に関する注意事項” を参照してください。	
ActECP *	アクティブな ECP 接続の数。	単位は秒ではありません。
Addblk *	ECP クライアントのキャッシュに追加されたブロックの数。	
PrgBufL *	ECP クライアントにおけるグローバル・バッファの不足のために ECP クライアントのキャッシュから削除されたブロックの数。 この数が多い場合、ECP クライアントでパフォーマンスの問題が発生することがあります。その場合は、ECP クライアントのグローバル・バッファの数を増やすことでパフォーマンスを向上できます。	
PrgSrvR *	ECP サーバにより ECP クライアントのキャッシュから削除されたブロックの数。	
BytSnt *	ECP クライアントとして送信したバイト数	
BytRcd *	ECP クライアントとして受信したバイト数。	
WDPass	起動以降の WD サイクル。	単位は秒ではありません。
IJUCnt	このサイクルを継続するために WD が待機しているジョブの数。	単位は秒ではありません。
IJULock	IJULock フラグが設定されているかどうかを示します。 IJULock が設定されている場合、WD の書き込みサイクルが終了するまですべての更新がロックされます。	単位は秒ではありません。
PPGrefs	すべてのプロセス・プライベート・グローバル・アクセスのカウント。	
PPGupds	すべてのプロセス・プライベート・グローバル更新のカウント。	

* ECP 構成を使用していない場合は、0 が表示されます。

11.1 Seize、ASeize、および NSeize に関する注意事項

他のプロセスの干渉を受けずに更新が実行されるようにするため、指定されたリソースに対してジョブが排他的にアクセスする必要がある場合には Seize が発生します。所定の Seize が即座に満たされない場合は、満たされるまで更新が延期されます。

シングル CPU システムの場合、このプロセスは直ちに休止状態になります(リソースを保持しているプロセスがその更新を完了してリソースを解放するまでは何もできないためです)。

マルチ CPU システムの場合、プロセスは妥当な時間内にリソースを取得することを“見込んで”保留ループに入ることで、休止状態による負荷の発生を回避します。保留ループ中にプロセスがリソースにアクセス可能になった場合は、直ちにループが終了し、プロセスの更新が続行されます。更新が終了すると、プロセスはそのリソースを待機している可能性のある他のプロセスにリソースを解放します。これが Aseize です。保留ループの最後でリソースが別のプロセスによって保持されている場合、プロセスは実行されず、リソースが解放されるまで引き続き休止状態のまま待機します。これが Nseize です。

シングル CPU システムで複数プロセスが実行されるときには Nseize のような状況が当然生じ、マルチ CPU システムで複数プロセスが実行されるときには Aseize のような状況が当然生じます。この 2 つの違いは、Nseize では、実行中のプロセスのコンテキストをオペレーティング・システムが変更する必要があることから、CPU のシステム時間または特権時間が発生するのに対し、Aseize では、リソースが解放され確保されるまで、あるいは中断して休止状態になるまで継続して CPU が実行されるため、CPU のユーザ時間が発生する点です。一般的に、マルチ CPU システムでは、コンテキスト・スイッチを実行するほうが、そうした操作を回避するために数回ループするよりもオペレーティング・システムの負荷が高くなります。これは、マルチ CPU システム上でのコンテキスト・スイッチに付随して CPU のオーバーヘッドやメモリの遅延が発生することに起因します。

12

履歴モニタ

履歴モニタは、パフォーマンスおよびシステム使用のメトリックの履歴データベースを保持します。その主な目的は以下のとおりです。

- パフォーマンスの基準値を提供し、パフォーマンスの問題の分析を支援します。
- 処理能力を計画するために、長期にわたるシステムの使用状況の分析を支援します。

このデータベースは、SYS.History クラス・パッケージで定義され、%SYS ネームスペースに保持されます。データベース構造の全詳細がそこにパブリッシュされ、データには SQL または通常の永続オブジェクト・アクセスを通してアクセスできます。SYS.History のクラス・ドキュメントにも、利用可能なすべての個々のプロパティ、メソッド、およびクエリの説明が含まれています。

データは通常、パフォーマンス（"SYS.History.Performance" を参照）データとシステム使用（"SYS.History.SystemUsage" を参照）データにまとめられます。パフォーマンス・メトリックは短い間隔（既定では 30 秒間）でサンプルが抽出され、システム使用データはこれより長い間隔（既定では 5 分間）でサンプルが抽出されるように意図されています。毎朝、個々の間隔のサンプルが、1 時間ごとの毎時テーブルと 1 日ごとの日次テーブルに集計され、平均値、最大値、最小値、標準偏差、中央値、および合計が示されます。それぞれのメトリック・クラスについて、集計関数がある場合はどれを保持するかを選択できます。間隔ごとのデータと時間ごとのデータは、定義された日数（既定では、それぞれ 7 日と 60 日）が過ぎたら、自動的に削除されます。毎日の集計データは、長期的な分析での使用を意図されており、手動で削除できます。

12.1 基本メトリック

収集されるメトリックはすべて、SYS.History の 4 つの %SerialObject クラスで定義されます。これらと同じクラスが、間隔ごと、毎時、および日次のデータベースの基礎として使用されるため、すべてのプロパティは集計で 10 進数値として表示できるように %Numeric 型として定義されます。

パフォーマンス関連のメトリックは、以下で定義されます。

- SYS.History.Performance — このクラスのプロパティは、グローバル参照やルーチン呼び出しのような一般的なパフォーマンス・メトリックです。

  注釈　これらのプロパティはすべて “カウンタ” 型であり、間隔データは、最終間隔におけるカウンタの変化を示すデルタとして収集されます。このデータは、毎時および日次の値に集計される際、1 秒あたりの比率に正規化されます。

- SYS.History.WriteDaemon ― このクラスのプロパティは、ライト・デーモン・サイクルのパフォーマンスを示します。システムは最後の 20 のライト・デーモン・サイクルを自動的に追跡し、履歴モニタはそれぞれの間隔で発生したサイクルのデータを格納します。通常、それぞれの間隔に複数のサイクルがあります。

システム使用のメトリックは、以下で定義されます。

- SYS.History.SystemUsage ― このクラスのプロパティは、システムの使用率を追跡しますが、一般的にパフォーマンス・データ (InterSystems IRIS® Data Platform 内のプロセス数やライセンス情報など) ほど迅速または劇的には変化しません。
- SYS.History.Database ― このクラスは、それぞれのローカル・データベースについて、データベースの増大、ファイル・サイズ、および空き容量を追跡します。

12.2 データの収集

データの収集を開始するには、以下の操作を実行する必要があります。

- %SYS ネームスペースでシステム・モニタの ^%SYSMONMGR ユーティリティを使用して、アプリケーション・モニタ (システム・モニタの一部) で目的のモニタ・クラス (%Monitor.System.HistorySys または %Monitor.System.HistoryPerf) を有効化します。これらのクラスは、既定で %SYS ネームスペースに登録されます。
- %SYS ネームスペースでシステム・モニタを再起動します。

これらの手順については、このドキュメントの “システム・モニタの使用” の章の ″^%SYSMONMGR を使用したアプリケーション・モニタの管理″ と ″Start/Stop System Monitor″ を参照してください。

一定間隔でのデータ収集の詳細は、次の 2 つの永続クラスで定義します。

- SYS.History.PerfData ― 埋め込みオブジェクトとしてパフォーマンスおよびライト・デーモン・クラスを含みます。
- SYS.History.SysData ― システム使用およびデータベース・クラスを含みます。

データを収集して履歴データを構築するために、アプリケーション・モニタで対応する %Monitor クラスを有効化する必要があります。

- %Monitor.System.HistoryPerf ― SYS.History.PerfData のサンプルのインスタンスを収集します。
- %Monitor.System.HistorySys ― SYS.History.SysData のサンプルを収集します。

既定では、システム・モニタ (アプリケーション・モニタを含む) は、InterSystems IRIS インスタンスの起動時に %SYS ネームスペースで起動します。ただし、他の実行開始ネームスペースを構成することができます。%Monitor クラスは、既定では %SYS でのみ提供されますが、^%SYSMONMGR を使用して他の実行開始ネームスペースに追加することができます。

12.3 集計

アプリケーション・モニタが実行する %Monitor.System.HistoryPerf クラスと %Monitor.System.HistorySys クラスは、毎日の終わりに毎時および日次集計も作成します。集計は、永続クラス SYS.History.Hourly および SYS.History.Daily として定義されます。これらは、4 つのベース・クラスをすべて埋め込みオブジェクトとして含みます。

それぞれのメトリック・プロパティについて、システムは、毎時間および丸 1 日の平均値、最大値 (高水位)、標準偏差、最小値、中央値、または合計を計算できます。集計関数は、各ベース・クラス (SYS.History.Performance、SYS.History.WriteDaemon、SYS.History.SystemUsage、または SYS.History.Database) および各集計期間クラスにつ

いて、それぞれのベース・クラスの SetSummary() メソッドを使用して選択できます (または無効にできます)。既定では、履歴モニタは、毎時と日次の両方の集計用に、各クラスの平均値、最大値、および標準偏差を計算します。

注釈　SYS.History.Performance クラスのカウンタ・プロパティは、これらの計算時 (合計を除く) に 1 秒あたりの比率に正規化されます。

データの削除

集計の作成後、アプリケーション・モニタは間隔データベースと毎時データベースを自動的に削除します。既定の保持期間は、間隔データは 7 日間、毎時データは 60 日間ですが、これらは SYS.History.PerfData クラスと SYS.History.Hourly クラスの SetPurge() メソッドを使用して変更できます。SYS.History.Daily データは自動的に削除されませんが、SYS.History.Daily:Purge() メソッドを使用して手動で削除できます。

12.4 データへのアクセス

データベースは永続クラスとして定義されるため、データは標準 SQL または永続オブジェクト・アクセスを使用して利用できます。管理ポータルで SQL ブラウザを使用すると、個々のプロパティ値を含め、作成されたさまざまな SQL スキーマやテーブルをすばやく簡単に表示できます。

SYS.History のそれぞれの永続クラス (SYS.History.PerfData、SYS.History.SysData、SYS.History.Hourly、および SYS.History.Daily) には、いくつかの基本クエリが実装されており、これを使用して、特定の日付範囲の個々のテーブルにアクセスできます。クエリの詳細については、クラスリファレンスのドキュメントを参照してください。

個々のテーブルを CSV 形式のファイルにエクスポートできるように、それぞれの永続クラスにはいくつかの Export() メソッドも提供されており、Microsoft Excel といったスプレッドシートでの使用に適しています。特に、SYS.History.PerfData:Export() メソッドは、^mgstat ユーティリティが作成するものに非常によく似た形式のファイルを作成します (詳細は、このガイドの “^mgstat を使用したパフォーマンスの監視” の章を参照してください)。

12.5 ユーザ定義メトリックの追加

履歴モニタにはユーザ定義メトリックを追加できます (SYS.History パッケージ)。

1. SYS.History.Adaptor から継承される 1 つまたは複数のクラスを作成し、%Numeric プロパティを追加してメトリックを定義します。

   注釈　ユーザ記述のクラスは %SYS ネームスペースに置く必要があります。また、システム・クラスとの名前の競合やアップグレード時の問題を防ぐために、“Z” または “z” で開始する必要があります。

2. Sample() メソッドをコーディングしてクラスをインスタンス化し、それぞれのプロパティに定期的な値を指定します。このメソッドは、間隔データが収集されたときに呼び出されます。

3. クラスをコンパイルすると、埋め込みオブジェクトとして SYS.History の間隔永続クラスに追加されます。SYS.History.Adaptor クラスで提供される INTERVAL パラメータを使用して、いつどこで収集するかを選択できます。これにより、以下のテーブルに示されているように、追加先の間隔クラスと、収集を行う %Monitor クラスが選択されます。

選択される INTERVAL	使用される間隔クラス	使用される %Monitor クラス
“User”（既定）	SYS.History.User	%Monitor.System.HistoryUser
“UserPerf”	SYS.History.UserPerf	%Monitor.System.HistoryPerf
“UserSys”	SYS.History.UserSys	%Monitor.System.HistorySys

“UserPerf”または“UserSys”を選択すると、SYS.History.PerfData または SYS.History.SysData と同じ間隔および同じタイムスタンプでデータを収集できるため、収集されたデータをシステム・データと簡単に関連付けることができます。“User”を使用すると、3 番目の（無関係の）時間間隔を選択できます。

注釈　SYS.History.Adaptor クラスには、プロパティの収集および集計方法のオプションを提供するパラメータがいくつかあります。詳細は、SYS.History.Adaptor クラスリファレンスのドキュメントを参照してください。

4. ユーザ定義クラスも埋め込みオブジェクトとして SYS.History.UserHourly および SYS.History.UserDaily 集計クラスに追加されます。ユーザ定義メトリックは集計されて、システム・メトリックと同じように自動的に削除されます。

重要　ユーザ定義メトリック・クラスは、永続データの埋め込みオブジェクトになります。データ収集が開始した後は、定義を変更しないでください。オブジェクトを削除すると、データが孤立する可能性があります。既存のクラスやプロパティを再定義すると、既に保存されたデータが誤って解釈される可能性があります。

ただし、スキーマ進化機能により、新しいオブジェクトおよびプロパティを安全に追加することができます。″クラスの定義と使用″の“永続クラスの定義”の章にある“スキーマ進化”を参照してください。

13

^BLKCOL を使用したブロック衝突の監視

ブロック衝突は、プロセスがブロックへのアクセスを待機させられる際に発生します。過度のブロック衝突は、アプリケーションのパフォーマンスを低下させます。

13.1 ^BLKCOL の使用法

InterSystems IRIS® Data Platform において、^BLKCOL ユーティリティは、指定された期間 (既定では 10 秒間) ブロック衝突をサンプル化し、この間指定された間隔 (既定では 10 ミリ秒) 内で最新のブロック衝突を記録します。記録された各衝突について、^BLKCOL はブロックだけでなく、ブロックにアクセスしようとするプロセスを作成したルーチンと行、ならびにブロック内の関連するグローバルおよびその最初と最後のリファレンスを識別します。

注釈　irisstat -D オプション (このドキュメントの付録 “irisstat ユーティリティを使用した InterSystems IRIS の監視” の ″オプションを使用した irisstat の実行″ を参照) もブロック衝突をサンプル化しますが、関連するブロックのみを識別します。

irisstat -D の出力は、^SystemPerformance ユーティリティで生成されたレポートに含まれます (このドキュメントの ″^SystemPerformance を使用したパフォーマンスの監視″ の章を参照)。

^BLKCOL を実行する際、以下を指定できます。

- サンプリング期間の長さ (秒)
- サンプル間の間隔 (ミリ秒)
- ルーチンの詳細を収集するかどうか (既定は [はい] です)
- 出力を以下としてフォーマットするかどうか
  - 最大衝突数のブロックのリスト (既定)
  - 衝突に関連するすべてのブロックのリスト
  - ブロック番号およびルーチンによって検出、ソート、およびカウントされているすべてのブロック衝突のコンマで区切られた値
  - 検出され、ソートされていない (未処理の) すべてのブロック衝突のコンマで区切られた値
  - ルーチンの衝突のホット・スポットのリスト
- 表示するブロックの数 (該当する場合)
- 出力をファイルに送信するかどうか

13.2 ^BLKCOL の出力

^BLKCOL ユーティリティの使用については、以下のサンプル・ターミナル・セッションで示されています。

```
%SYS>do ^BLKCOL

Block Collision Analysis

How many seconds should we sample: <10>
How long to wait (ms) between each sample: <10>
Collect routine details? <Y>
Format for 'T'op counts, 'D'isplay all, 'S'orted CSV, 'H'ot spot, or 'R'aw CSV: <T>
Number of blocks to display: <10>
Output to file: <0>

Sampling ... (any key to interrupt)


625 block collisions in 735 samples.

Block # (count) - Global refs (first - last in block) - Routine refs (SFN)

767       (395) in c:\InterSystems\iris\mgr\user\
  ^acctest - ^acctest(10220," 167") (T/BPtr)
    325 at ^AccessTest+156(4)
    25 at ^AccessTest+121(4)
    24 at ^AccessTest+92(4)
    8 at ^AccessTest+109(4)
    8 at ^AccessTest+127(4)
    4 at ^AccessTest+170(4)
    1 at ^AccessTest+163(4)

3890      (11)  in c:\InterSystems\iris\mgr\user\
  ^acctest(2552,"  371") - ^acctest(2552,"  38") (Data)
    6 at ^AccessTest+164(4)
    3 at ^AccessTest+163(4)
    1 at ^AccessTest+134(4)
    1 at ^AccessTest+156(4)

15572     (9)   in c:\InterSystems\iris\mgr\user\
  ^acctest(6980," 4795") - ^acctest(6988," 3259") (Data)
    7 at ^AccessTest+134(4)
    1 at ^AccessTest+164(4)
    1 at ^AccessTest+170(4)

15818     (8)   in c:\InterSystems\iris\mgr\user\
  ^acctest(9124," 173") - ^acctest(9124," 1743") (Data)
    5 at ^AccessTest+164(4)
    3 at ^AccessTest+170(4)

971       (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(484," 3927") - ^acctest(484," 3938") (Data)
    5 at ^AccessTest+170(4)
    2 at ^AccessTest+164(4)

1137      (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(756," 4063") - ^acctest(756," 4073") (Data)
    3 at ^AccessTest+109(4)
    2 at ^AccessTest+134(4)
    1 at ^AccessTest+156(4)
    1 at ^AccessTest+163(4)

2999      (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(2092,"  666") - ^acctest(2092,"  674") (Data)
    3 at ^AccessTest+170(4)
    1 at ^AccessTest+109(4)
    1 at ^AccessTest+121(4)
    1 at ^AccessTest+134(4)
    1 at ^AccessTest+164(4)

6173      (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(3684,"  528") - ^acctest(3684,"  536") (Data)
    3 at ^AccessTest+163(4)
    1 at ^AccessTest+109(4)
    1 at ^AccessTest+156(4)
    1 at ^AccessTest+164(4)
    1 at ^AccessTest+170(4)

14617     (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(9688,"   18") - ^acctest(9688,"   26") (Data)
```

```
    4 at ^AccessTest+170(4)
    2 at ^AccessTest+164(4)
    1 at ^AccessTest+134(4)

15282     (7)   in c:\InterSystems\iris\mgr\user\
  ^acctest(8700," 4889") - ^acctest(8760," 1402") (Data)
    4 at ^AccessTest+170(4)
    3 at ^AccessTest+164(4)
%SYS>d ^BLKCOL

Block Collision Analysis

How many seconds should we sample: <10>
How long to wait (ms) between each sample: <10>
Collect routine details? <Y>
Format for 'T'op counts, 'D'isplay all, 'S'orted CSV, 'H'ot spot, or 'R'aw CSV: <T> H
Number of blocks to display: <10>
Output to file: <0>

Sampling ... (any key to interrupt)


571 block collisions in 768 samples.


Sorted by routine/line that waits for block ownership
_____________________________________________________

(571) AccessTest
    (324) +156^AccessTest :  s @G@($J,node)=$$getdata($E(Str,1,$r(1000)))  ;SMLXXX+, AFH
    (54) +164^AccessTest :  k @G@($J,node)
    (43) +134^AccessTest :  .  k @G@($J,node)
    (31) +92^AccessTest :  .  .  k @G@($j)
    (28) +109^AccessTest :  .  s x=$O(@G@($J,x))

Sorted by routine that owns the block
_____________________________________

(472) AccessTest
    (472) +AccessTest
```

14

^PERFSAMPLE を使用したプロセスの監視

ここでは、InterSystems IRIS® データ・プラットフォームのプロセスを分析するためのツールである ^PERFSAMPLE ユーティリティについて説明します。このユーティリティはライブ・システム上のアクティビティを処理し、サンプリングされたアクティビティについて容易にナビゲートできる分析結果を示します。これにより、システムに対する洞察を提供できます。例えば、ECP 要求を確認することでアプリケーションのボトルネックを発見したり、待機イベントのタイプを確認することでシステム全体のボトルネックを特定することができます。

開始するには、目的の InterSystems IRIS インスタンスの %SYS ネームスペースから ^PERFSAMPLE を実行します。

Terminal

```
USER>set $namespace = "%SYS"

%SYS>do ^PERFSAMPLE
```

InterSystems プロセスのより一般的な情報は、"InterSystems IRIS プロセスの管理" を参照してください。

14.1 サンプルの収集

^PERFSAMPLE を実行するとすぐに、次のメッセージが表示されます。

Terminal

```
This utility performs high frequency sampling of processes on the system,
analyzing and counting data points in different ways to understand where
processes are spending most of their time. On ECP Data Servers, this also
offers sampling of the current request being processed and the states of
the ECPSvrW daemons doing the processing.

1) Sample Local Process Activity
2) Sample ECP Server Requests

Option?
```

インスタンスに ECP クライアント (アプリケーション・サーバ) からの受信 ECP 接続がない場合、上記の `option 1` が自動的に選択されます。

その後、次の情報を入力するよう求められます。

1. どのプロセスまたは ECP 接続をサンプリングするか。
2. プロセスが以下のいずれかの状態のサンプルを無視するかどうか: READ、READW、EVTW、HANG、SLCT、SLCTW、および RUNW。ECP 接続のサンプリングでは、ECPSvrW プロセスがアイドルでないイベントのみが記録されます。

YES（既定値）を選択すると、^PERFSAMPLE で記録されるイベント数が削減されます。多くのプロセスを監視する場合、これによって分析が迅速化され、使用するメモリが少なくなります。

3. 1 秒あたりの収集サンプル数。
4. サンプルを収集する合計秒数。

ターミナルに表示されるプロンプトは次のようになります。

Terminal

```
Enter a list of PIDs, * for all, or ? for ^%SS display: *
Ignore samples where the process appears idle (READ, HANG, etc)?
  Yes =>
Sample rate per second: 1000 =>
Number of seconds to sample: 30 =>
```

14.2 サンプルの検査と分析

サンプルを収集したら、分析を表示できます。分析は、サンプリングされたプロセスの 1 つ以上のディメンジョン、またはコンポーネントの要約です。つまり、分析は選択したディメンジョンに従ってサンプリングした情報をソートします。

このセクションでは、以下の項目について説明します。

- 事前定義分析の使用例
- カスタム分析の作成に関する情報
- 利用可能な分析のディメンジョンの説明

アナライザ内の移動には、次のキーを使用します。

キー入力	移動アクション
上矢印または U	セレクタを上に移動します。
下矢印または D	セレクタを下に移動します。
右矢印または Enter	現在の項目を選択します。
左矢印または Backspace	前のレベルに戻ります。
C	次のカウント表示を繰り返します。 ・ 合計のパーセンテージ ・ 未加工数 ・ 現在のサブセットのパーセンテージ ・ （複数のジョブがサンプリングされる場合）この状態で同時に見つかったジョブ数の平均
N または CTRL-D	次のページへ（複数のページがある場合）
P または CTRL-U	前のページへ（複数のページがある場合）
Q	^PERFSAMPLE を終了します。

メインのランディング・ページは次のようになります。

Terminal

```
– PERFSAMPLE for Local Process Activity. 1.710949s at 11/17/2020 15:58:31
28479 samples | CPULoad* 0.91
Multiple jobs included: 1290 samples per job
----------------------------'?' for help----------------------------
Select an analysis to view:
New Analysis (press '+' any time)
Using CPU? -> PID -> Process State
Using CPU? -> Routine -> Namespace -> Process State
Process State -> Routine -> PID
Kernel Wait State -> Routine -> PID
```

14.2.1 事前定義分析の例

以下に、Process State ディメンジョンで始まる分析例を示します。

この例では、^PERFSAMPLE により、合計で 319994 のサンプル数の中から、サンプリング可能な状態（アイドルを無視するオプションが選択されていた場合）のプロセスのサンプルが 76755 件見つかりました。

Terminal

```
– PERFSAMPLE for Local Process Activity. 3.89s at 11/17/2020 16:59:59
76755 events in 319994 samples [24.0 %-total] | CPULoad* 8.22
Multiple jobs included: 2191 samples per job
----------------------------'?' for help----------------------------
Process State [24.0 %-total]
GGET [8.46 %-total]
RUN [5.88 %-total]
GDEF [3.16 %-total]
GSETW [1.63 %-total]
BSETW [1.21 %-total]
GDEFW [1.18 %-total]
GGETW [0.931 %-total]
SEMW [0.685 %-total]
GSET [0.311 %-total]
LOCKW [0.144 %-total]
LOCK [0.0644 %-total]
INCRW [0.0641 %-total]
BSET [0.0513 %-total]
```

最初に、サンプルの合計数のパーセンテージとして値が表示されます。このケースでサンプリングされた最も一般的な Process State 値は GGET でした。これは、全部で 319994 件のサンプルの 8.46% を占めています。

c を押すと、このカウントの表示方法が切り替えられます。例えば、上記の情報を未加工のサンプル数として表示できます。

Terminal

```
Process State [76755]
> GGET [27083]
RUN [18823]
GDEF [10121]
```

この情報を適格なサンプルのパーセンテージ（この場合はアイドルではない Process State を持つサンプル）として表示することもできます。

Terminal

```
Process State [24.0 %-total]
> GGET [35.3 %-subset]
RUN [24.5 %-subset]
GDEF [13.2 %-subset]
```

最後に、各状態で同時に見つかったジョブ数の平均を表示できます。

Terminal

```
Process State [24.0 %-total]
GGET [12.4 jobs]
RUN [8.59 jobs]
GDEF [4.62 jobs]
```

右矢印キーを使用して GGET を選択すると、次のディメンジョンに移動し、サンプルの最初のディメンジョンの値が GGET となるように、そのディメンジョンの値が並べられます。ディメンジョン間は、矢印キーを使用して自由に移動できます。

14.2.2 カスタム分析の作成

メインのランディング・ページの New Analysis オプションを選択し、カスタム分析を作成します。次のショートカットのいずれかを使用してカスタム分析を作成することもできます。

キー入力	ショートカット
+ キー	(分析内で) 現在の分析にディメンジョンを追加します。
* キー	最初のディメンジョンとして現在の項目を使用して、新しい分析を開始します。

新しい分析を追加すると、次の画面が表示されます。

Terminal

```
New Analysis:

Specify a comma-delimited list of dimensions upon which to analyze samples.
For example, "state,ns,rou" means first count each unique state the sampled
processes were in; then for each state, count the namespace from the samples
in that state; and finally for each state->namespace pair, count each unique
routine name. In other words, report on routines by namespace by state.

The following dimensions are available:
cpu - Using CPU? (process state indicates expected CPU use)
ns - Namespace (current namespace)
pid - PID (process ID)
rou - Routine (name of current routine)
state - Process State (process state string, e.g. GSETW)
trace - Kernel Trace (alternative to 'state' w/ kernel-level detail)
waits - Kernel Wait State (kernel-level condition that delayed the process)
wtrace - Reverse Kernel Trace (revese kernel trace, stop at any wait state)

Enter dimension list:
```

ここから、プロンプトの説明に従って、分析するディメンジョンのリストを表示します。Enter を押すと、上記のとおり分析に移動できます。

14.2.3 分析のディメンジョン

分析のディメンジョンは、^PERFSAMPLE ツール内に示されます。このセクションでは追加情報を示します。

- cpu – CPU を使用するかどうか(プロセス状態により、CPU の使用量が予測される)

  注釈　cpu に対する yes または no の値は、CPU の使用時には真の基準とはなりませんが、見積もりには使用できます。^PERFSAMPLE はプロセス状態から CPU の使用量を推測し、InterSystems IRIS の状態トラッキングは CPU の使用量には直接関係しません。

  CPU の (瞬間的または永続的な) 過剰使用により、OS スケジューラが CPU を使用可能にするのをプロセスが待機している場合にも、cpu に誤差が生じる場合があります。

- ns – ネームスペース (現在のネームスペース)

- pid – PID (プロセス ID)
- rou – ルーチン (現在のルーチンの名前)
- state – プロセス状態 (プロセス状態を表す文字列。GSETW など)
- waits – カーネル待機状態 (プロセスを遅延させたカーネルレベルの状態)。詳細は、以下のセクションを参照してください。

一般に、インターシステムズのサポート窓口でのトラブルシューティングの際に有用なのは、次のディメンジョンのみです。

- trace – Kernel Trace (カーネルレベルの詳細を含む 'state' の代替)
- wtrace – Reverse Kernel Trace (逆カーネル・トレース、任意の待機状態で停止)

trace および wtrace ディメンジョンには階層構成があります。省略記号 (...) で表される祖先を選択すると、階層レベルを下ることができます。祖先以外の項目を選択すると、分析の次のディメンジョンに進みます。h キーを使用すると、この階層ビューと平坦化ビューを切り替えることができます。祖先上で a キーを押すと、そのすべての子孫のディメンジョンが集約されます。

14.2.3.1 waits ディメンジョン

InterSystems IRIS のカーネル内部の何かをプロセスが待機していることが認められなかった場合、waits ディメンジョンは NULL です。NULL 以外の値は、プロセスが待機する (内部的にブロックする) 必要がある状態を示します。

これらは、アプリケーションの直接の制御外でプロセスが待機状態になる内部状態であることに注意することが重要です。したがって、競合する LOCK コマンド、$SYSTEM.Event などによる待機は、ここでは対象外です。

それでも、多くの値 (特に頻度の高い値) が、間接的にアプリケーションの影響を受ける可能性があります。例えば、主なアプリケーション・プロセスがよく diskio を待機していることがサンプルで示された場合、プロセスはディスクからのデータベース・ブロックの読み取りを待機しており、これは、並列処理、事前フェッチ、またはデータベース・キャッシュの増加が効果的である可能性があることを示しています。同様に、inusebufwt を待機していることが多いことがサンプルによって示されているプロセスは、データベース・ブロックの衝突が発生しており、アプリケーション・レベルでの調査 (^BLKCOL utilityを使用して) が必要な可能性があります。このディメンジョン内の値は、次のニーモニック値を持ちません。これは将来的には変更される可能性があります。

- diskio : データベース物理ブロック読み取りを待機中
- inusebufwt : ブロック衝突による待機中 (アプリケーションの原因の特定には ^BLKCOL utility が役立つ場合があります)
- expand : データベースの拡張を待機中
- ecpwait : ECP サーバからの応答を待機中
- jrniowait : ジャーナル・バッファに空き領域がないため、ジャーナル入出力を待機中
- jrnsyncblk : ジャーナル・データがコミットされるのを待機中
- jrnlckwait : ジャーナル・バッファへのアクセスを待機中
- mirrorwait : アクティブなバックアップ・ミラー・メンバを待機中
- mirrortrouble : ミラーの障害状態によりブロックされている
- globwait : グローバル更新が内部の状態によりブロックされているため待機中
- aiowait : 非同期ディスク I/O が完了するのを待機中
- wdqwait : 書き込みサイクルが完了するのを待機中

- `freebuf`：グローバル・バッファがすべて使用されており、データベースの書き込みを待機中
- `gfownwait`：データベースへのアクセスがブロックされている
- `resenqXYZ`：内部リソース XYZ で待機中

注釈　これらの多くは、`W` の文字フラグを含むキャノニック形式のプロセス状態（`GSETW`、`GORDW` など）に対応していますが、すべてがそうであるとは限らず（`diskio` がよくある例です）、`W` 状態フラグのすべてのケースでここに反映される内部的な理由があるわけではありません（上記の `LOCKW` など）。

14.3 分析の保存

サンプルを表示したら、今後の分析のためにこれらを保存できます。保存するには、分析のランディング・ページで**左矢印**を押します。これにより、最初のサンプルの収集ページに戻りますが、`Save Samples to File` というオプションが追加されています。このオプションを選択して、`perfsample001.txt` など、目的のファイル名を入力します。^PERFSAMPLE により、ファイルが install-dir**¥mgr** ディレクトリに保存されます。

保存されている分析を開くには、LOAD タブを使用して ^PERFSAMPLE を起動し、開くファイルを指定します。次に、例を示します。

Terminal

```
USER>set $namespace = "%SYS"
%SYS>do LOAD^PERFSAMPLE
File: C:\MyIRIS\mgr\perfsample001.txt
```

^PERFSAMPLE によりファイルがロードされ、保存されているサンプルを分析および検証できます。

A

SNMP を使用した InterSystems IRIS の監視

この付録では、InterSystems IRIS® Data Platform と SNMP (Simple Network Management Protocol) 間のインタフェースについて説明します。SNMP は、ネットワーク・デバイスやコンピュータ・デバイスなど、TCP/IP ネットワーク全体を管理する手段として広く使用されている通信プロトコルです。その普及率の高さから、現在では、重要な基盤構造およびプロトコルとして多くのエンタープライズ管理ツールに取り入れられています。これは、InterSystems IRIS にとって非常に重要で、さまざまな種類の管理ツールに管理情報と監視情報を提供する標準的な方法です。

SNMP は、標準のメッセージ形式であると同時に、管理対象オブジェクトの標準定義セットでもあります。また、カスタム管理対象オブジェクトを追加するための標準構造でもあります。InterSystems IRIS では、この機能を使用して、他のアプリケーションで使用する管理情報を定義します。

A.1 InterSystems IRIS での SNMP の使用法

SNMP は、クライアント (ネットワーク管理アプリケーション) がサーバ・プログラム (SNMP エージェント) に接続するクライアント・サーバ関係を定義します。このサーバ・プログラムは、リモート・ネットワーク・デバイスまたはコンピュータ・システム上で実行されます。クライアントは、エージェントに対して情報を要求し、エージェントから情報を受け取ります。SNMP メッセージには次の 4 つの基本タイプがあります。

- GET – 特定の管理対象オブジェクトのデータを取得します。
- GETNEXT – 階層ツリーで、次に位置している管理対象オブジェクトのデータを取得します。これによって、システム管理者は、デバイスのすべてのデータを参照できます。
- SET – 特定の管理対象オブジェクトの値を設定します。
- TRAP – 管理対象のデバイスまたはシステムが送信した非同期アラート。

SNMP MIB (Management Information Base) には、管理対象オブジェクトの定義が格納されます。各デバイスは、標準 MIB のどの部分をサポートしているかを定義するファイル (**MIB**)、および管理対象オブジェクトのカスタム定義を発行します。InterSystems IRIS では **ISC-IRIS.mib** ファイルがこれに相当します。このファイルは、install-dir**¥SNMP** ディレクトリに配置されています。

A.2 サブエージェントとしての InterSystems IRIS

SNMP クライアントは、既知のアドレス (ポート **161**) で待ち受け状態にある SNMP エージェントに接続します。クライアントは常にこのポートを介して接続するので、コンピュータ・システム上では 1 つの SNMP エージェントしか実行できませ

ん。システム上の複数のアプリケーションにアクセスする必要がある場合は、マスタ・エージェントを実装します。これによって、複数のサブエージェントに接続できるようになります。インターシステムズでは、SNMP マスタ・エージェントを介して通信するサブエージェントとして InterSystems IRIS SNMP インタフェースを実装しています。

InterSystems IRIS がサポートしているほとんどのオペレーティング・システムには、複数のサブエージェントをサポートするように拡張できる SNMP マスタ・エージェントが備わっています。ただし、多くの場合、これらのエージェントの拡張性は互換性のない独自の方法で実装されています。InterSystems IRIS は、RFC 2741 で定義されている IETF 推奨の標準プロトコル AgentX (Agent Extensibility) を使用してサブエージェントを実装しています。

一部の標準 SNMP マスタ・エージェントは AgentX をサポートしています。オペレーティング・システムで提供されている SNMP マスタ・エージェントが AgentX 互換でない場合は、代わりに、パブリック・ドメイン Net-SNMP エージェントを使用できます。

注釈　ただし、Windows 標準エージェントは例外で、AgentX をサポートしていません。また、Net-SNMP バージョンも使用できません。この例外に対処するため、InterSystems には、Windows 拡張エージェント DLL **iscsnmp.dll** が用意されています。この DLL は、標準 Windows SNMP サービス拡張 API と InterSystems IRIS AgentX サーバ間の接続を制御します。

A.3 InterSystems IRIS での SNMP の管理

SNMP は標準プロトコルなので、InterSystems IRIS サブエージェントの管理は最小限で済みます。最も重要な作業は、システム上の SNMP マスタ・エージェントが AgentX (Agent Extensibility) プロトコル ("サブエージェントとしての InterSystems IRIS" を参照してください) と互換性があることを確認し、標準の AgentX TCP ポート **705** で接続要求を待ち受けるように設定することです。Windows システムの場合、標準 Windows SNMP サービスに接続するための DLL がシステムによって自動的にインストールされます。Windows SNMP サービスがインストールされており、自動または手動で開始されていることを確認してください。

重要　特に Linux での Net-SNMP など、SNMP マスタ・エージェントの中には、既定で AgentX が有効にならず、有効になっても既定で TCP ポート **705** を使用しないものがあります。Net-SNMP の場合は、InterSystems IRIS サブエージェントとの通信が有効になるように **snmpd.conf** ファイルを変更する必要があります。最新バージョンの Net-SNMP も VACM (ビューベースのアクセス制御モデル) のセキュリティを実装しており、既定で、**mib-2.system** サブツリーへのアクセスのみが許可されます。InterSystems IRIS サブエージェントは、エラーなく起動および実行されますが、SNMP 要求は、InterSystems IRIS に転送されません。**snmpd.conf** で定義されている“ビュー”を拡張して、InterSystems IRIS MIB サブツリーを追加する必要があります。

次に、以下の手順で、監視サービスを有効にします。

1. 管理ポータルの **[サービス]** ページ (**[システム管理]** > **[セキュリティ]** > **[サービス]**) に移動します。
2. **%Service_Monitor** サービスをクリックします。
3. **[サービス有効]** チェック・ボックスにチェックを付けて、**[保存]** をクリックします。
4. サービス・リストのページに戻り、**%Service_Monitor** サービスが有効になっていることを確認してください。

最後に、InterSystems IRIS を起動したとき、InterSystems IRIS SNMP サブエージェントが自動的に開始されるように構成するには、以下の手順に従います。

1. 管理ポータルの **[モニタ設定]** ページ (**[システム管理]** > **[構成]** > **[追加設定]** > **[モニタ]**) に移動します。
2. **[システム開始時にSNMPエージェントを開始]** の設定で、**[　]** を選択して、**[保存]** をクリックします。
3. この設定を編集した場合、InterSystems IRIS 側の SNMP インタフェースは即座に停止した後に開始します。

^SNMP ルーチンを使用して、InterSystems IRIS SNMP サブエージェントを手動で、またはプログラム的に開始および停止することもできます。

```
Do start^SNMP(<port>,<timeout>)
Do stop^SNMP
```

ここで、<port> は、接続に使用する TCP ポート (既定は 705)、<timeout> は TCP ポートの読み取りタイムアウト値 (既定は 20 秒) です。InterSystems IRIS は、<timeout> 値に達するまで、接続の確立時や要求への応答時に発生した問題を install-dir¥mgr ディレクトリにある SNMP.LOG ファイルに記録します。

注釈　SNMP マスタ・エージェントを再起動する場合、^SNMP ルーチンを使用して InterSystems IRIS SNMP サブエージェントを手動で再起動する必要がある場合があります (前述の説明を参照してください)。

A.4 SNMP のトラブルシューティング

InterSystems IRIS サブエージェント (^SNMP ルーチンを実行) は、オペレーティング・システムが提供する SNMP マスタ・エージェントが適切にインストールおよび構成されていることを前提としています。"サブエージェントとしての InterSystems IRIS" の説明のとおり、^SNMP ルーチンはこのマスタ・エージェントと主に 2 つの方法で通信します。

- 基本的に、^SNMP は、TCP ポート 705 で AgentX プロトコルを使用します。
- Windows では ^SNMP は、iscsnmp.dll としてインストールされた Windows 拡張エージェント DLL を使用します。

SNMP エージェントの構成手順の詳細は、オペレーティング・システムで提供されており、システム管理者はこの構成方法を理解するために時間を取る必要があります。以下に、InterSystems IRIS と SNMP エージェントの通信を行う際に問題が発生した場合のトラブルシューティングの基本的なガイドラインとヒントを示します。

A.4.1 すべてのシステム

- SNMP エージェントが InterSystems IRIS から独立して動作し、少なくとも一般的なシステム情報について mib-2.system ツリーへクエリを実行できることを確認してください。これに失敗したときには、Windows の場合は Windows SNMP サービスを確認してください。Unix®/Linux の場合は SNMP デーモン (snmpd) が実行されているかどうかを確認してください。
- SNMP システム情報を正常にクエリできるが、InterSystems IRIS MIB をクエリできない場合は、InterSystems IRIS で ^SNMP ルーチンを実行するバックグラウンド・プロセスを確認します。$$start^SNMP() 関数を使用してこのルーチンの起動を試行します。このルーチンが起動はしても、実行を継続しない場合は、InterSystems IRIS の install-dir/mgr ディレクトリにある、messages.log および SNMP.log ログ・ファイルでエラーを確認してください。Windows では、発生したすべてのエラーが iscsnmp.dll によって Windows¥System32¥snmpdbg.log に記録されます (64 ビット Windows システムでは、このファイルは SysWOW64 サブディレクトリにあります)。
- InterSystems IRIS の %Service_Monitor サービスが有効化されていることを確認します。
- %SYS ネームスペースで、^SYS("MONITOR","SNMP","DEBUG")=1 を設定し、^SNMP InterSystems IRIS サブエージェント・プロセスを再起動すれば、より多くの情報を SNMP.log ファイルに記録させることができます。このログでは、送受信されるメッセージごとに詳細情報が記録されます。

A.4.2 Windows システム

- Windows のすべてのバージョンが、既定で Windows SNMP サービスをインストールするわけではありません。これには、追加の手順が必要になる場合があります。このサービスの [プロパティ] ダイアログの [セキュリティ] タブに、少なくとも 1 つ読み取り権限のあるパブリック・コミュニティが存在することを確認してください。SNMP トラップを送信

するには、プロパティ・ダイアログの [**トラップ**] タブで [**コミュニティ名**] と [**トラップ送信先**] を定義する必要があります。

- InterSystems IRIS では、InterSystems IRIS をインストールする前に SNMP サービスがインストールされていることを想定しているため、**iscsnmp.dll** を適切なレジストリ・キーに追加できます。InterSystems IRIS をインストールしたら、SNMP サービスを再起動して、これが **iscsnmp.dll** を適切にロードし、新しい InterSystems IRIS インスタンスを検出して通信できるようにする必要があります。

  注釈　InterSystems IRIS が SNMP サービスよりも先にインストールされていると、**iscsnmp.dll** が適切に登録されません。Windows SNMP サービスのインストール後に、set myStatus=$$Register^SNMP() 関数を使用して登録する必要があります。これを行ったら、SNMP サービスを再起動する必要があります。

- Windows では、$$start^SNMP() 関数は SNMP サービスにシグナルを送信するだけで、InterSystems IRIS の ^SNMP プロセスは、実際には SNMP サービスから InterSystems IRIS へのコールバックによって起動されます。プロセスが開始されるまでには数秒かかり、クエリに応答できるまでにはさらに数秒かかる場合があります。

A.4.3 UNIX® システム

現時点において、多くの UNIX オペレーティング・システム (IBM AIX®) では、AgentX プロトコルがサポートされていません。システムで AgentX がサポートされていない場合は、AgentX をサポートする Net-SNMP などの独立した SNMP エージェントをインストールする必要があります。

A.4.4 Linux および macOS と Net-SNMP

- 既定では AgentX サポートは有効化されず、既定のポートは 705 ではありません。**snmpd.conf** ファイルを修正して、master agentx および agentXSocket TCP:localhost:705 を追加するか、コマンド行で snmpd -x TCP:localhost:705 を使用する必要があります。
- `syslocation`、`syscontact`、`sysservices` などの基本的なシステム情報を **snmpd.conf** で定義して、snmpd デーモンが正常に起動できるようにする必要があります。
- 最新バージョンの Net-SNMP も VACM (ビューベースのアクセス制御モデル) のセキュリティを実装しており、既定で、`mib-2.system` サブツリーへのアクセスのみが許可されます。InterSystems IRIS サブエージェントは、エラーなく起動および実行されますが、SNMP 要求は、InterSystems IRIS に転送されません。**snmpd.conf** で定義されている “ビュー” を拡張して、InterSystems IRIS MIB サブツリーを追加する必要があります。
- SNMP トラップを送信するには、**snmpd.conf** ファイルで trapsink パラメータを使用して送信先を定義する必要があります。例えば、trapsink 192.16.61.36 public のように指定します。

A.5 InterSystems IRIS MIB 構造

InterSystems IRIS SNMP インタフェースを介して取得できるすべての管理対象オブジェクト・データは、InterSystems IRIS MIB ファイル **ISC-IRIS.mib** で定義されています。このファイルは、install-dir**¥SNMP** ディレクトリにあります。通常、SNMP 管理アプリケーションが情報を理解して、適切に表示するためには、管理対象アプリケーションの MIB ファイルをロードする必要があります。この手順はアプリケーションによって異なるため、InterSystems IRIS MIB のロード方法については、使用する管理アプリケーションのドキュメントを参照してください。

InterSystems IRIS MIB で定義されているデータはこのファイル自体に詳しく記載されているので、ここでは説明しません。InterSystems IRIS MIB ツリーの全体的な構造を理解しておくと、同一システム上で複数のインスタンスを実行する場合などに非常に役立ちます。

注釈　MIB ツリーを表示する最適な方法は、MIB を管理アプリケーションまたは MIB ブラウザにロードすることです。これらのツールでは、オブジェクト ID (OID)、オブジェクトの内容に一致するテキストの表示、およびオブジェクトの説明を伴ったツリーとして MIB が表示されます。

SNMP では、すべての管理対象オブジェクトを網羅する特別な階層ツリー構造である SMI (Structure of Management Information) が定義されています。詳細は RFC 1155 で規定されています。それぞれの管理対象オブジェクトには、一連の整数をピリオドで区切って表される一意なオブジェクト識別子 (OID) が割り当てられます (例:`1.3.6.1.2.1.1.1`)。MIB は、このドット区切り整数識別子をテキスト名に変換します。

標準の SNMP MIB では、多数の標準管理対象オブジェクトが定義されています。標準 MIB のアプリケーション特有拡張を定義する場合、InterSystems IRIS と同様、アプリケーションでは以下のように enterprise 分岐を使用します。

```
iso.org.dod.internet.private.enterprises (1.3.6.1.4.1)
```

IANA (Internet Assigned Numbers Authority) では、階層の次のレベルに該当するプライベートなエンタープライズ番号を各組織に割り当てています。InterSystems IRIS の場合、この番号は intersystems を表す `16563` です。

この下に、以下のように InterSystems IRIS のエンタープライズ・プライベート・サブツリーが実装されています。

- intersystems の下位レベルは、“製品” ID またはアプリケーション ID のレベルです。InterSystems IRIS の場合、これは `.4` (`iscIris`) です。この番号は、MIB モジュール ID として使用できます。
- 次のレベルは “オブジェクト” のレベルで、ここでは通知からデータ・オブジェクトが抽出されます。InterSystems IRIS の場合、これらは `.1` (`irisObjects`) と `.2` (`irisTraps`) です。慣例により、intersystems ツリーでは、すべてのデータ・オブジェクトと通知名に小文字の短い接頭語が追加されます。InterSystems IRIS の場合、これは iris です。
- 次のレベルは “テーブル” (グループ・レベル) です。すべてのデータ・オブジェクトはテーブルとして管理されます。テーブルに含まれるインスタンス (“行”) が 1 つしかない場合も同様です。これによって、管理データ・オブジェクトをグループ化することができます。また、1 台のマシン上で複数の InterSystems IRIS インスタンスを実行する場合も、テーブルが必要になります。テーブルの最初のインデックスでは、必ず InterSystems IRIS のインスタンス名が使用されます。テーブルに複数のインデックスが含まれる場合もあります。
- 次のレベルは、(SNMP SMI で義務付けられている) テーブルの “概念行” で、常に `.1` です。
- 最後に、インデックスとして指定されているデータを含め、テーブル内の個々のデータ・オブジェクトが配置されます。
- 通知 (トラップ) は、“テーブル” と同じ階層レベルで、個々のエントリとして定義されます。詳細は、この付録の ″InterSystems IRIS SNMP トラップ″ を参照してください。
- 通知 (トラップ) を介して送信される InterSystems IRIS 固有の予備オブジェクトは、“テーブル” と同じ階層レベルで、個々のエントリとして定義されます。詳細は、この付録の ″InterSystems IRIS SNMP トラップ″ を参照してください。

例えば、データベースのサイズが、

```
1.3.6.1.4.1.16563.4.1.3.1.6.4.84.69.83.84.1
```

としてコード化されている場合、このコードは以下の内容を表しています。

```
iso.org.dod.internet.private.enterprises.intersystems.isciris.irisObjects
irisDBTab.irisDBRow.irisDBSize.TEST(instname).1(DBindex)
```

A.5.1 InterSystems IRIS MIB の拡張

アプリケーション・プログラマは、管理対象オブジェクトの定義を追加し、InterSystems IRIS サブエージェントがデータを提供する MIB を拡張できます。この拡張は完全な MIB エディタや SNMP ツールキットとすることを意図したものではなく、SNMP を介した参照やクエリの対象とすることができる簡潔なアプリケーション・メトリックを追加する 1 つの手段です。

注釈　オブジェクトは基本的な InterSystems IRIS SNMP 構造に従う必要があります。SNMP テーブル構造のサポートには制限があり (インデックスは整数値のみがサポートされています)、SNMP トラップは作成されません (%Monitor.Alert クラスの説明を参照してください)。管理情報の SNMP 構造の基本を理解していると役に立ちます。

これらのオブジェクトを作成する手順は以下のとおりです。

1. %Monitor.Adaptor クラスを継承するクラスに、InterSystems IRIS オブジェクトの定義を作成します。%Monitor パッケージに対する管理対象オブジェクトの追加の詳細は、"インターシステムズ・クラス・リファレンス" を参照してください。
2. SNMP クラス・メソッドを実行して、それらの管理対象オブジェクトを SNMP で有効にし、管理アプリケーションに必要な MIB 定義ファイルを作成します。それを実行するためのメソッドは MonitorTools.SNMP.CreateMIB() です。

CreateMIB() メソッドのパラメータについては、"インターシステムズ・クラス・リファレンス" の MonitorTools.SNMP クラスに関するドキュメントを参照してください。

このメソッドは、%Monitor データベースで定義されているアプリケーションについて、プライベート・エンタープライズ MIB ツリーの分岐を作成します。アプリケーションの実際の MIB ファイルに加え、このメソッドは、MIB ツリーの大まかな内部構造も作成します。InterSystems IRIS サブエージェントはこれを使用して、MIB サブツリーを登録し、GETNEXT 要求に応じてツリーを参照し、GET 要求のインスタンス・データを収集するためのオブジェクト・メソッドを参照します。

すべての管理対象オブジェクト定義は、InterSystems IRIS エンタープライズ MIB ツリーと同じ一般構成 (`application.objects.table.row.item.indices`) を使用します。どのテーブルでも、最初のインデックスは InterSystems IRIS アプリケーション ID です。すべてのアプリケーションは IANA に登録して、そのアプリケーションのプライベート・エンタープライズ番号を取得する必要があります。この番号は、CreateMIB() メソッドに指定するパラメータの 1 つです。

SNMP でアプリケーションを無効にするには、MonitorTools.SNMP.DeleteMIB() メソッドを使用します。このメソッドを実行すると、アプリケーション MIB の内部構成が削除されます。したがって、InterSystems IRIS サブエージェントは、プライベート・エンタープライズ MIB サブツリーに登録したり、プライベート・エンタープライズ MIB サブツリーに関する要求に応じられなくなります。

ユーザ・モニタ・クラスの定義の例については、この付録の "サンプル・ユーザ定義 SNMP モニタ・クラス" を参照してください。

A.5.2 InterSystems IRIS SNMP トラップ

SNMP クエリで使用可能なオブジェクト・データおよびメトリックのほか、InterSystems IRIS では、非同期アラートである SNMP トラップを送信できます。以下のテーブルでは、InterSystems IRIS 固有の SNMP トラップについて説明します。

テーブル I-1: InterSystems IRIS SNMP 通知オブジェクト (トラップ)

トラップ名 (番号)	説明
irisStart (1)	InterSystems IRIS インスタンスを起動しました。
irisStop (2)	InterSystems IRIS インスタンスはシャットダウン処理中です。
irisDBExpand (3)	InterSystems IRIS データベースが正常に拡張されました。
irisDBOutOfSpace (4)	その後の InterSystems IRIS データベースの拡張が制限される可能性があります。ファイル・システムに追加の 10 個の拡張に対応する空き容量が存在しないか、または空き容量が 50 MB 未満です。
irisDBStatusChange (5)	InterSystems IRIS データベースの読み取り/書き込み状況が変更されました。
irisDBWriteFail (6)	InterSystems IRIS データベースへの書き込みに失敗しました。失敗した書き込みの InterSystems IRIS エラー・コードが記録されています。

トラップ名 (番号)	説明
irisWDStop (7)	InterSystems IRIS インスタンスのライト・デーモンが停止しました。
irisWDPanic (8)	InterSystems IRIS インスタンスのライト・デーモンが“パニック”モードになりました。これは、ライト・デーモンがバッファの全容量を使い果たしたので、データベース・ブロックをライト・イメージ・ジャーナル (WIJ) ファイルにコミットせずに、直接ディスクに書き込むことが必要になっている状況です。
irisLockTableFull (9)	InterSystems IRIS インスタンスのロック・テーブルに空きがないので、以降のロックは失敗します。
irisProcessFail (10)	アクセス違反が発生したので、プロセスにより InterSystems IRIS が異常終了しました。詳細は、messages.log ファイルを参照してください。
irisECPTroubleDSrv (11)	この ECP データ・サーバと InterSystems IRIS データベースの接続に深刻な通信障害が発生しました。詳細は、messages.log ファイルを参照してください。
irisECPTroubleASrv (12)	この ECP アプリケーション・サーバとリモート InterSystems IRIS データベースの接続に深刻な通信障害が発生しました。詳細は、messages.log ファイルを参照してください。
irisAuditLost (13)	InterSystems IRIS は、監査イベントの記録に失敗しました。考えられる原因として、監査データベース用の領域の問題があります。オペレータにご相談ください。
irisLoggedError (14)	“重大な”エラーが messages.log ファイルに記録されました。このトラップには、irisSysErrorMsg で定義されたエラー・メッセージがあります。
irisLicenseExceed (15)	ライセンスの要求が、現在使用可能なライセンス数または現在許可されているライセンス数を超えました。
irisEventLogPost (16)	エントリが Interoperability イベント・ログに送信されました。
irisAppAlert (100)	これは、SNMP を介してアラートを生成するために InterSystems IRIS アプリケーションで使用可能な一般トラップです。このトラップの使用方法の詳細は、%Monitor.Alert.External クラス・メソッドを参照してください。

以下のテーブルでは、上記のテーブルで説明したトラップで送信可能な InterSystems IRIS 固有の予備オブジェクトについて説明します。

テーブル I-2: トラップで送信される InterSystems IRIS 固有の予備オブジェクト

予備オブジェクト名 (番号)	説明
irisDBWriteError (1)	失敗したデータベース書き込みの InterSystems IRIS 固有エラー・コード。可能な値は、<DATABASE>、<DISKHARD>、<BLOCKNUMBER>、<FILEFULL>、または <DATABASE MAP LABEL> です。
irisApp (2)	irisAppAlert トラップを生成したアプリケーション (このトラップの発生元) を識別する短いテキスト文字列 (最長 20 文字)。
irisAppSeverity (3)	irisAppAlert トラップの問題の深刻度を表すコード。このコードは、0 (情報)、1 (警告)、2 (重大)、または 3 (致命的) のいずれかです。
irisApptext (4)	irisAppAlert トラップの原因となった問題、エラー、またはイベントの、テキスト文字列による説明 (最長 1024 文字)。

A.6 サンプル・ユーザ定義 SNMP モニタ・クラス

このセクションでは、SNMP を介したクエリの対象とすることができるユーザ・アプリケーション・モニタ・クラス (このドキュメントの “システム・モニタの使用” の章の ″アプリケーション・モニタ″ を参照) の定義方法の例を示します。アプリケーション・モニタは、SNMP データの %Monitor データ型のプロパティのみを含んでいます。

サンプル・クラスの例

この例のサンプル・クラスは以下のとおりです。

Class Definition

```
Class SNMP.Example Extends %Monitor.Adaptor
{

  /// Give the application a name. This allows you to group different
  /// classes together under the same application level in the SNMP MIB.
  /// The default is the same as the Package name.
  Parameter APPLICATION = "MyApp";

  /// This groups a set of properties together at the "table" level of the
  /// SNMP MIB hierarchy. The default is the Class name.
  Parameter GROUPNAME = "MyTable";

  /// An integer metric counter
  Property Counter1 As %Monitor.Integer;

  /// Another integer metric counter
  Property Counter2 As %Monitor.Integer;

  /// A status indicator as a string data type
  Property Status As %Monitor.String;

  /// The method is REQUIRED. It is where the Application Monitor
  /// calls to collect data samples, which then get picked up by the
  /// ^SNMP server process when requested.
  Method GetSample() As %Status
  {
    set ..Counter1=$r(200)
    set ..Counter2=200+$r(100)
    set n=$r(4)
    set ..Status=$s(n=1:"Crashed",n=2:"Warning",n=3:"Error",1:"Normal")
    Quit $$$OK
  }

}
```

このクラスをユーザのネームスペースでコンパイルする前に、InterSystems IRIS はサポートするクラスをそのユーザのネームスペースにロードする必要があります。これらのクラスは、SNMP のデータ・サンプルを保存するために必要です。

クラスをロードするには、このドキュメントの “システム・モニタの使用” の章の ″^%SYSMONMGR を使用したアプリケーション・モニタの管理″ の説明に従って ^%SYSMONMGR を実行し、以下の操作を行います。

- オプション 2 の `Manage Monitor Classes` (モニタ・クラスの管理) を選択します。
- オプション 3 の `Register Monitor System Classes` (モニタ・システム・クラスの登録) を選択します。

サンプル・クラスをコンパイルするときに、サンプル・データを格納するための SNMP.Sample.Example クラスが作成されます。

重要　生成されたサンプル・クラスを明示的に削除しないでください。アプリケーション・モニタおよび生成されたサンプル・クラスの両方を削除すると、モニタ・クラス・ルーチンが削除されているにもかかわらず、サンプル・クラス・ルーチンが残っているので、エラーが発生します。すべてのサンプル・クラス・ルーチンを確実に正しく削除するには、サンプル・クラスを生成したアプリケーション・モニタ・クラスのみを削除します。このモニタ・クラスを削除すると、このモニタ・クラスおよび生成されたサンプル・クラスが、両方のクラスに関連するルーチンと共に削除されます。例えば、サンプル・クラス (SNMP.Sample.Example など) を削除するには、管理ポータルを使用して、そのサンプル・クラスの生成元のモニタ・クラス (つまり、SNMP.Example) を削除します。

^%SYSMONMGR を実行して、サンプル・クラスを有効化し、アプリケーション・モニタを起動してサンプルを収集します。

1. オプション 2 の `Manage Monitor Classes` (モニタ・クラスの管理) を選択します。
2. オプション 1 の `Activate/Deactivate a Monitor Class` (モニタ・クラスの有効化/無効化) を選択します。
3. 登録済みのモニタ・クラスの番号付きリストを表示するには「?」を入力します。
4. 有効化するモニタ・クラスの番号を入力します。例えば、SNMP.Example というユーザ定義クラスを有効にするには、クラス名の横にある番号を入力します。
5. オプション 6 の `Exit` (終了) を選択します (アプリケーション・モニタのメイン・メニューに戻ります)。
6. オプション 1 の `Manage Application Monitor` (アプリケーション・モニタの管理) を選択します。
7. オプション 1 の `Start Application Monitor` (アプリケーション・モニタの開始) を選択します。
8. オプション 5 の `Exit` (終了) を選択します (アプリケーション・モニタのメイン・メニューに戻ります)。
9. オプション 6 の `Exit` (終了) を選択します (アプリケーション・モニタのメイン・メニューを終了します)。

注釈　アプリケーション・モニタの構成と使用については、このドキュメントの “システム・モニタの使用” の章の ″アプリケーション・モニタ″ を参照してください。

ユーザ MIB の作成例

SNMP MIB を作成するには、%SYS ネームスペースで MonitorTools.SNMP:CreateMIB メソッドを実行します。詳細は、MonitorTools.SNMP クラス・ドキュメントを参照してください。

このメソッドに対する入力パラメータは、以下のようになります。

```
CreateMIB("MyApp","USER",99990,1,"mycorp","myapp","mc","MC-MYAPP","Unknown",1)
```

重要　実稼動では Enterprise ID に `99990` を使用しないでください。組織ごとに専用の ID を IANA に登録する必要があります。

```
USER>set $namespace = "%SYS"

%SYS>Do ##class(MonitorTools.SNMP).CreateMIB("MyApp","USER",99990,1,"mycorp",
"myapp","mc","MC-MYAPP","Unknown",1)


Create SNMP structure for Application - MyApp

     Group - MyTable
          Counter1 = Integer
          Counter2 = Integer
          Status = String


Create MIB file for MyApp

     Generate table MyTable
          Add object Counter1
          Add object Counter2
          Add object Status
```

```
%SYS>
```

これにより、既定のディレクトリ (install-dir¥mgr¥User) に MC-MYAPP.MIB ファイルが作成されます。このファイルを、使用する SNMP 管理アプリケーションにロードできます。

注釈　SNMP マスタ・エージェントおよび InterSystems IRIS ^SNMP サービスでこの MIB が認識されるためには、これらのエージェントとサービスを再起動する必要があります。

```
--
-- MIB file generated for mcMyApp product.
--
-- Sep 16, 2008
--

MC-MYAPP DEFINITIONS ::= BEGIN

IMPORTS

  MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
  Counter32, Gauge32, Integer32
  FROM SNMPv2-SMI
  DisplayString
  FROM SNMPv2-TC
  enterprises
  FROM RFC1155-SMI
  cacheSysIndex
  FROM ISC-IRIS;


mcMyApp MODULE-IDENTITY
    LAST-UPDATED "200809161700Z"
    ORGANIZATION "mycorp"
    CONTACT-INFO "
                  Unknown"
    DESCRIPTION ""
    ::= { mycorp 1 }

mycorp OBJECT IDENTIFIER ::= { enterprises 16563 }

myappObjects OBJECT IDENTIFIER ::= { mcMyApp 1 }

--
-- Begin tables
--

-- Table myappMyTable

myappMyTable         OBJECT-TYPE
       SYNTAX        SEQUENCE OF myappMyTableR
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION
       ""
   ::= { myappObjects 1 }

myappMyTableR        OBJECT-TYPE
       SYNTAX        myappMyTableR
       MAX-ACCESS    not-accessible
       STATUS        current
       DESCRIPTION
       "Conceptual row for MyTable table."
       INDEX  { cacheSysIndex }
   ::= { myappMyTable 1 }

myappMyTableR ::=
       SEQUENCE {
        myappCounter1   Integer32
        myappCounter2   Integer32
        myappStatus   DisplayString
       }

myappCounter1        OBJECT-TYPE
       SYNTAX        Integer32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION
       ""
       ::= { myappMyTableR 1 }
```

```
myappCounter2        OBJECT-TYPE
       SYNTAX        Integer32
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION
       ""
       ::= { myappMyTableR 2 }

myappStatus          OBJECT-TYPE
       SYNTAX        DisplayString
       MAX-ACCESS    read-only
       STATUS        current
       DESCRIPTION
       "Status"
       ::= { myappMyTableR 3 }

--  End of MyTable table

myappTraps  OBJECT IDENTIFIER ::= { mcMyApp 2 }

-------------------------------------------------------------
 END
```

B

Web サービスを使用した InterSystems IRIS の監視

この付録では、InterSystems IRIS® Data Platform のサポートにより WS-Management 仕様準拠機能を使用する方法について簡単に説明します。それらの機能を使用すると、SOAP を介してリモートで InterSystems IRIS インスタンスを監視できます。

B.1 InterSystems IRIS での WS-Monitoring のサポートの概要

SYS.WSMon パッケージは、WS-Management 仕様に従って、InterSystems IRIS インスタンスのリモート監視に使用できる Web サービスを提供します。このパッケージは、機能面で SNMP インタフェース (このドキュメントの付録 “SNMP を使用した InterSystems IRIS の監視” を参照) と類似していますが、組み込みの InterSystems IRIS Web サービス・サポートを使用しています。

WS-Management のサポートには、以下が含まれます。

- ログ・モニタリング Web サービス (SYS.WSMon.Service) は、InterSystems IRIS インスタンスに関する情報を返すメソッドを提供します。
- InterSystems IRIS Web サービス・クライアント (SYS.WSMon.Client) は、そのモニタリング Web サービスまたは別の InterSystems IRIS インスタンスのモニタリング Web サービスのメソッドを呼び出すことができます。

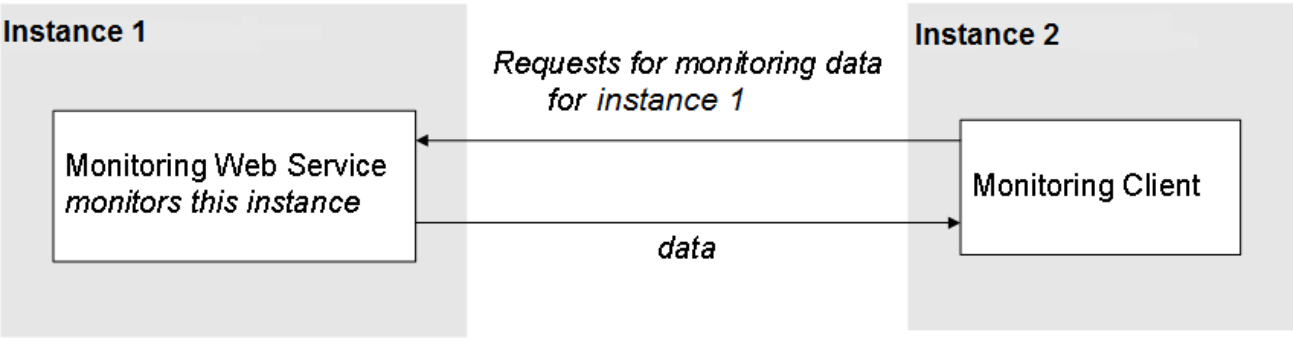
  この Web クライアントを使用する代わりに、独自の Web クライアントを作成することができます (場合によってはサードパーティ・テクノロジの使用も可能)。
- 各種 XML 対応クラスは、Web サービスとクライアントが監視情報を表示するために使用されます。

  これらのクラスには SYS.WSMon.wsEvent が含まれ、それによってイベントの表示が可能になります。
- サンプル・イベント・シンク Web サービス (SYS.WSMon.EventSink) は、イベントを受信して処理します。SOAP 呼び出しを使用し、このサンプル・イベント・シンク・サービスをサブスクライブすることにより、すべてのモニタリング Web サービスのイベントを受信できます。

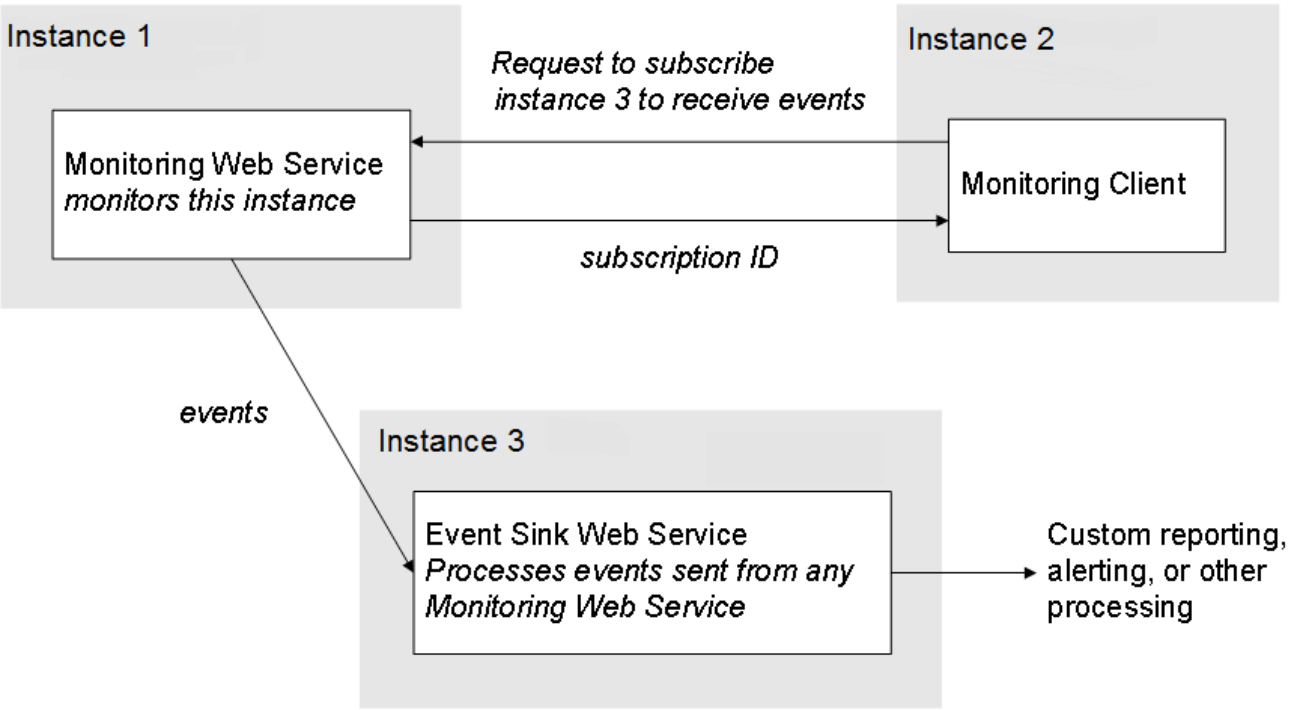
  このサンプルを使用する代わりに、独自のサンプルを作成することができます (場合によってはサードパーティ・テクノロジの使用も可能)。

これらのクラスは、%SYS ネームスペースでのみ使用できます。

基本的な監視として、ログ・モニタリング Web サービスと別インスタンスの Web クライアントを使用できます。以下の図は、監視クライアントがターゲット・インスタンスに監視データを要求する様子を示しています。ターゲット・インスタンス上のログ監視 Web サービスが応答し、クライアント監視データを送信しています。



より高度な例としては、別の InterSystems IRIS インスタンス上で実行されている可能性もある イベント・シンク・サービスを Web クライアントがサブスクライブすることがあります。例えば、以下の図では、監視クライアントはプライマリ・インスタンスに 3 番目のインスタンスをサブスクライブするよう要求を送信しています。プライマリ・インスタンスの監視 Web サービスは、クライアントにサブスクリプション ID を送信して応答し、その後、3 番目のインスタンスにイベントを送信します。3 番目のインスタンスはカスタム・レポートやアラートを作成するなど、さまざまな方法でこれらのイベントを処理できます。



イベント・シンク Web サービスでは、業務に必要なあらゆる処理を実行できます。

スタジオに用意されている SOAP ウィザードを使用して、Web サービスおよび Web クライアントを WSDL から生成できます。このウィザードの使用法の詳細は、"Web サービスおよび Web クライアントの作成" を参照してください。(同様のウィザードは、さまざまなサードパーティ・テクノロジによっても提供されています。)

B.2 サポートの詳細

インターシステムズでは、WS-Management の仕様の以下の部分をサポートします。

- wxf:Get
- wsen:Enumerate
- wsen:Pull
- wsen:Release
- wse:Subscribe
- wse:Renew
- wse:Unsubscribe

詳細は、WS-Management の仕様 (https://www.dmtf.org/standards/published_documents/DSP0226_1.1.pdf) を参照してください。

B.3 モニタリング Web サービスの URL

任意の InterSystems IRIS インスタンスに対して、ログ・モニタリング Web サービスを次の URL で使用できます。

```
http://server:port/csp/sys/SYS.WSMon.Service.cls
```

server は、InterSystems IRIS が実行されているサーバです。また、port は、InterSystems IRIS Web サービス・サーバが使用するポートです。以下はその例です。

```
http://localhost:52773/csp/sys/SYS.WSMon.Service.cls
```

同様に、この Web サービスの WSDL を次の URL で使用できます。

```
http://server:port/csp/sys/SYS.WSMon.Service.cls?WSDL=1
```

B.4 モニタリング Web サービスの Web メソッド

SYS.WSMon.Service クラスには、以下の Web メソッドが用意されています。

EnumBuffer()

```
method EnumBuffer() as %XML.DataSet
```

すべてのバッファ・サイズの統計を列挙する %XML.DataSet のインスタンスを返します。このインスタンスのデータセットには SYS.Stats.Buffer クラスの Sample() クラス・クエリが使用されます。

%XML.DataSet を使用した作業の詳細は、"Web サービスおよび Web クライアントの作成" の "SOAP メッセージでのデータセットの使用" の章、または %XML.DataSet のクラス・リファレンスを参照してください。

SYS.Stats.Buffer のクラスリファレンスも参照してください。

EnumDatabase()

```
method EnumDatabase() as %XML.DataSet
```

そのインスタンスのすべてのデータベースを列挙する %XML.DataSet のインスタンスを返します。このインスタンスのデータセットには SYS.WSMon.wsDatabase クラスの List() クラス・クエリが使用されます。

EnumBuffer() のコメントおよびクラスリファレンスの SYS.WSMon.wsDatabase を参照してください。

EnumResource()

```
method EnumResource() as %XML.DataSet
```

あらゆるシステム・リソースの Seize の統計を列挙する %XML.DataSet のインスタンスを返します。このインスタンスのデータセットには SYS.Stats.Resource クラスの Sample() クラス・クエリが使用されます。

EnumBuffer() のコメントおよびクラスリファレンスの SYS.Stats.Resource を参照してください。

EnumWriteDaemon()

```
method EnumWriteDaemon() as %XML.DataSet
```

すべてのライト・デーモンの統計を列挙する %XML.DataSet のインスタンスを返します。このインスタンスのデータセットには SYS.Stats.WriteDaemon クラスの Sample() クラス・クエリが使用されます。

EnumBuffer() のコメントおよびクラスリファレンスの SYS.Stats.WriteDaemon を参照してください。

EventCancel()

```
method EventCancel(id) as %Integer
```

指定された Web サービスのサブスクリプションをキャンセルします。EventSubscribe() を参照してください。

EventSubscribe()

```
method EventSubscribe(location) as %String
```

指定された Web サービスをサブスクライブして、その InterSystems IRIS インスタンスのイベントに関する情報を受信します。この Web サービスには、例で示されているように、独自の Web サービスまたは SYS.WSMon.EventSink Web サービスを指定することができます。独自の Web サービスを作成する場合、SYS.WSMon.EventSink Web サービスの WSDL に従う必要があります。

location には、その Web サービスの EventSink() メソッドを呼び出すのに必要な URL を指定します。SYS.WSMon.EventSink では、次のように location を指定できます。

```
http://server:port/csp/sys/SYS.WSMon.EventSink.cls
```

server は、InterSystems IRIS が実行されているサーバです。また、port は、InterSystems IRIS が使用するポートです。

それぞれのイベントに対して、InterSystems IRIS は、指定された Web サービスの EventSink() メソッドの呼び出しを試行して、SYS.WSMon.wsEvent のインスタンスを送信します。

このメソッドが返す ID は、サブスクリプションのキャンセルに使用できます。EventCancel() を参照してください。

GetDisk()

```
method GetDisk() as SYS.Stats.Disk
```

そのインスタンスのグローバルに対するディスク使用量のメトリックを含む SYS.Stats.Disk のインスタンスを返します。

クラスリファレンスの SYS.Stats.Disk を参照してください。

GetECPAppSvr()

```
method GetECPAppSvr() as SYS.Stats.ECPAppSvr
```

そのインスタンスの ECP アプリケーション・サーバのメトリックを含む SYS.Stats.ECPAppSvr のインスタンスを返します。

クラスリファレンスの SYS.Stats.ECPAppSvr を参照してください。

GetECPDataSvr()

```
method GetECPDataSvr() as SYS.Stats.ECPDataSvr
```

そのインスタンスの ECP データベース・サーバのメトリックを含む SYS.Stats.ECPDataSvr のインスタンスを返します。

クラスリファレンスの SYS.Stats.ECPDataSvr を参照してください。

GetGlobal()

```
method GetGlobal() as SYS.Stats.Global
```

そのインスタンスのグローバル・メトリックを含む SYS.Stats.Global のインスタンスを返します。

クラスリファレンスの SYS.Stats.Global を参照してください。

GetRoutine()

```
method GetRoutine() as SYS.Stats.Routine
```

そのインスタンスのルーチン・メトリックを含む SYS.Stats.Routine のインスタンスを返します。

クラスリファレンスの SYS.Stats.Routine を参照してください。

GetSystem()

```
method GetSystem() as SYS.WSMon.wsSystem
```

InterSystems IRIS インスタンスのシステム情報を含む SYS.WSMon.wsSystem のインスタンスを返します。

クラスリファレンスの SYS.WSMon.wsSystem を参照してください。

B.5 Web クライアントの監視

SYS.WSMon.Client クラスとその関連クラスは、同一または別の InterSystems IRIS インスタンスにある SYS.WSMon.Server Web サービスのメソッドを呼び出すことのできる、InterSystems IRIS Web サービス・クライアントです。

この Web クライアント・クラスでは、次の LOCATION パラメータを使用します。

```
Parameter LOCATION = "http://server:port/csp/sys/SYS.WSMon.Service.cls"
```

server は、InterSystems IRIS が実行されているサーバです。また、port は、InterSystems IRIS Web サービス・サーバが使用するポートです。

以下に示すように、その他の InterSystems IRIS Web サービス・クライアントを使用するときと同様にこの Web クライアントを使用します。

1. Web クライアント・クラスのインスタンスを作成します。
2. 必要に応じて、Location プロパティを設定します。

   この設定は、使用する SYS.WSMon.Server Web サービスがクライアントとは異なるマシンにある場合、または 52773 以外のポートを使用する場合に必要です。
3. 必要に応じて、その他のプロパティを設定します。

   "Web サービスおよび Web クライアントの作成" を参照してください。
4. Web メソッドを呼び出します。
5. Web メソッドによって返される値を確認します。

   詳細は、呼び出す Web メソッドによって異なります。“モニタリング Web サービスの Web メソッド”、およびクラス・リファレンスの返りタイプの説明を参照してください。

ターミナル・セッションの例を以下に示します。

```
USER>set $namespace = "%SYS"

%SYS>set client=##class(SYS.WSMon.Client).%New()

%SYS>set client.Location="http://localhost:57799/csp/sys/SYS.WSMon.Service.cls"

%SYS>set myroutinestats=client.GetRoutine()

%SYS>write myroutinestats.RtnCallsLocal
19411581
%SYS>write myroutinestats.RtnCallsRemote
0
%SYS>write myroutinestats.RtnCommands
432764817
%SYS>
```

データを取得してユーザ・インタフェースに表示するときには、プログラムによりクライアントを作成して使用するほうがより一般的です。

注釈　SYS.WSMon パッケージを使用できるのは、%SYS ネームスペースに限られます。つまり、ここに記載されている手順を実行するには、%SYS ネームスペースを使用する必要があります。

B.6 イベントの処理

InterSystems IRIS が提供するサンプル Web サービス (SYS.WSMon.EventSink) は、ログ・モニタリング Web サービスによって送信されるイベントを受信して処理できます。この Web サービスを使用することも、独自の Web サービスを作成することもできます。

B.6.1 サンプル・イベント・シンク Web サービスの使用

SYS.WSMon.EventSink は、イベントを受信して処理することのできる、InterSystems IRIS のサンプル Web サービスです。

任意の InterSystems IRIS インスタンスに対して、ログ・モニタリング Web サービスを次の URL で使用できます。

```
http://server:port/csp/sys/SYS.WSMon.EventSink.cls
```

server は、InterSystems IRIS が実行されているサーバです。また、port は、InterSystems IRIS Web サービス・サーバが使用するポートです。

この Web サービスにはメソッドが 1 つあります。

CacheEventSink()

```
Method CacheEventSink(event As SYS.WSMon.wsEvent) As %Integer
```

Windows プラットフォームでは、イベントが発生すると、このサンプル・メソッドによってポップアップ・ウィンドウが表示されます。その他のプラットフォームでは、`^SYS("MONITOR","WSMON","EVENT_RECEIVED",$h)` にエントリが追加されます。

このメソッドは常に 1 を返します。

このサンプル・サービスをサブスクライブして、モニタリング Web サービスからイベントを受信するには、ターミナルで以下のように実行します。

```
USER>set $namespace = "%sys"

%SYS>set client=##class(SYS.WSMon.Client).%New()

%SYS>set eventsinklocation="http://localhost:52773/csp/sys/SYS.WSMon.EventSink.cls"

%SYS>set subscriptionid=client.EventSubscribe(eventsinklocation)

%SYS>write subscriptionid
CacheEventSubscription_2
```

eventsinklocation は、イベントを処理するイベント・シンク Web サービスのURL です。

B.6.2 独自のイベント・シンク・Web サービスの作成

独自のイベント・シンク Web サービスを作成するには、スタジオで SOAP ウィザードを使用して、以下の WSDL から Web サービスを生成します。

```
http://server:port/csp/sys/SYS.WSMon.EventSink.cls?WSDL=1
```

server は、InterSystems IRIS が実行されているサーバです。また、port は、InterSystems IRIS Web サービス・サーバが使用するポートです。

このウィザードの使用法の詳細は、"Web サービスおよび Web クライアントの作成" を参照してください。

続いて、生成された Web サービスの CacheEventSink() メソッドを変更して、カスタム・ロジックを組み込みます。

C

REST API を使用した InterSystems IRIS の監視

すべての InterSystems IRIS® Data Platform インスタンスには、そのインスタンスの統計情報を提供する REST インタフェースが含まれています。この REST API を使用することで、InterSystems IRIS が実行されている複数のマシンから情報を収集し、アプリケーションを構成するすべてのインスタンスを詳細に監視できます。

この付録では、/api/monitor サービスが提供するメトリックについて説明します。これらのメトリックは、オープンソースの監視およびアラート・ツールである Prometheus と互換性があります。接続された複数の InterSystems IRIS インスタンスをスクレイプするように Prometheus を構成することで、システム全体をまとめたビューを提供し、システムが正しく効率的に動作しているかどうかを容易に評価できます。

注釈　REST インタフェースの作成および使用の概要は、"REST サービスの作成の概要" を参照してください。ご自分ですぐに作成してみる場合は、"Developing Rest Interfaces" を参照してください。

C.1 /api/monitor サービス

/api/monitor サービスは、このサービスが実行されている InterSystems IRIS インスタンスに関する情報を提供します。既定では、/api/monitor Web アプリケーションは “認証なし” アクセスで使用できます。このサービスの認証の設定については、"REST サービスの作成" の "REST サービスの保護" の章を参照してください。

この API には、以下の 2 つのエンドポイントがあります。

- /metrics エンドポイントは、すべてのインスタンスのメトリックを返します。また、特定のアプリケーションのメトリックを返すように構成できます。
- /alerts エンドポイントは、エンドポイントが最後にスクレイプされた後に送信されたすべてのシステム・アラートを返します。

注釈　InterSystems IRIS は、install-dir/mgr ディレクトリにある SystemMonitor.log ファイルにエラーを記録します。

C.1.1 /metrics エンドポイント

/metrics エンドポイントは、"メトリックの説明" で説明されているメトリックのリストを返します。"相互運用メトリック" で説明されているように、アクティブな相互運用プロダクションに関するその他のメトリックのコレクションを有効にすることもできます。"アプリケーションのメトリックの作成" には、カスタム・メトリックを定義するための手順が記載されています。

InterSystems IRIS のインスタンスをスクレイプするように Prometheus を構成するには、"First Steps With Prometheus" (https://prometheus.io/docs/introduction/first_steps/) の手順に従います。

C.1.1.1 メトリックの説明

メトリックはテキストベース形式で返されます。これについては、Prometheus のドキュメントの "Exposition Formats" ページ (https://prometheus.io/docs/instrumenting/exposition_formats/) で説明されています。各メトリックは 1 行に記述し、名前と値は 1 つのスペースで区切ります。

以下のテーブルに、InterSystems IRIS メトリックを表示します。ここでは、読みやすくするために、ラベルがあるメトリック名には改行を入れています。

注釈　このテーブルには、ここで説明している InterSystems IRIS のバージョンのメトリックが含まれます。新しいバージョンではメトリックが追加される可能性があるため、このドキュメントがお使いの InterSystems IRIS のバージョンと一致することを確認してください。

メトリック名	説明
`iris_cpu_pct {id="ProcessType"}`	InterSystems IRIS プロセス・タイプ別の CPU 使用率。ProcessType は、以下のいずれかにできます。`ECPWorker ECPCliR ECPCliW ECPSrvR ECPSrvW LICENSESRV WDAUX WRTDMN JRNDMN GARCOL CSPDMN CSPSRV ODBCSRC MirrorMaster MirrorPri MirrorBack MirrorPre MirrorSvrR MirrorJrnR MirrorSK MirrorComm` (InterSystems IRIS プロセスの詳細は、"インターシステムズのプロセスおよびオペレーティング・システム・リソースの保護" を参照してください。)
`iris_cpu_usage`	オペレーティング・システム上のすべてのプログラムの CPU 使用率
`iris_csp_activity {id="IPaddress:port"}`	Web ゲートウェイ・サーバが起動してからそのサーバによって処理された Web 要求の数
`iris_csp_actual_connections {id="IPAddress:port"}`	Web ゲートウェイ・サーバからこのサーバへの現在の接続の数
`iris_csp_gateway_latency {id="IPaddress:port"}`	`iris_csp_` メトリックのフェッチ時に、Web ゲートウェイ・サーバから応答を取得するのに要する時間 (ミリ秒単位)
`iris_csp_in_use_connections {id="IPaddress:port"}`	Web 要求を処理している Web ゲートウェイ・サーバからこのサーバへの現在の接続の数
`iris_csp_private_connections {id="IPaddress:port"}`	ステート認識アプリケーションのために予約されている (保持モード 1) Web ゲートウェイ・サーバからこのサーバへの現在の接続の数
`iris_csp_sessions`	このサーバで現在アクティブな Web セッション ID の数
`iris_cache_efficiency`	物理読み込みと物理書き込みに対するグローバル参照の割合 (パーセント)

メトリック名	説明
iris_db_expansion_size_mb {id="database"}	database を拡張する量 (MB単位)
iris_db_free_space {id="database"}	database で使用可能な空き容量 (MB単位) (このメトリックは 1 日 1 回しか更新されないため、最新の変更が反映されていないことがあります。)
iris_db_latency {id="database"}	database からのランダム読み取りが完了するまでに要した時間 (ミリ秒単位)
iris_db_max_size_mb {id="database"}	database が拡大可能な最大サイズ (MB単位)
iris_db_size_mb {id="database",dir="path"}	database のサイズ (MB単位)
iris_directory_space {id="database",dir="path"}	database ディレクトリのストレージ・ボリュームで利用可能な空き領域 (MB 単位)
iris_disk_percent_full {id="database",dir="path"}	database ディレクトリのストレージ・ボリュームで埋まっている領域の割合
iris_ecp_conn	この ECP アプリケーション・サーバでアクティブなクライアント接続の合計数
iris_ecp_conn_max	この ECP アプリケーション・サーバからのアクティブなクライアント接続の最大数
iris_ecp_connections	この ECP アプリケーション・サーバと構成されている ECP データ・サーバを同期する際に同期されるサーバの数
iris_ecp_latency	ECP アプリケーション・サーバと ECP データ・サーバとの間の遅延 (ミリ秒)
iris_ecps_conn	この ECP データ・サーバへの 1 秒あたりのアクティブなクライアント接続の合計数
iris_ecps_conn_max	この ECP データ・サーバへのアクティブなクライアント接続の最大数
iris_glo_a_seize_per_sec	グローバル・リソース上の 1 秒あたりの Aseize の数 (詳細は、"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションの "Seize、ASeize、および NSeize に関する注意事項" を参照してください。)
iris_glo_n_seize_per_sec	グローバル・リソース上の 1 秒あたりの Nseize の数 (詳細は、"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションの "Seize、ASeize、および NSeize に関する注意事項" を参照してください。)
iris_glo_ref_per_sec	ローカル・データベースにあるグローバルへの 1 秒あたりの参照の数

メトリック名	説明
iris_glo_ref_rem_per_sec	リモート・データベースにあるグローバルへの 1 秒あたりの参照の数
iris_glo_seize_per_sec	グローバル・リソース上の 1 秒あたりの Seize の数 (詳細は、"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションの "Seize、ASeize、および NSeize に関する注意事項" を参照してください。)
iris_glo_update_per_sec	ローカル・データベースにあるグローバルへの 1 秒あたりの更新 (SET および KILL コマンド) の数
iris_glo_update_rem_per_sec	リモート・データベースにあるグローバルへの 1 秒あたりの更新 (SET および KILL コマンド) の数
iris_jrn_block_per_sec	ディスクに書き込まれた 1 秒あたりのジャーナル・ブロック
iris_jrn_free_space {id="JournalType",dir="path"}	各ジャーナル・ディレクトリのストレージ・ボリュームで利用可能な空き領域 (MB 単位)。JournalType は WIJ、primary、または secondary です。
iris_jrn_size {id="JournalType"}	各ジャーナル・ファイルの現在のサイズ (MB 単位)。JournalType は WIJ、primary、または secondary です。
iris_license_available	現在未使用のライセンスの数
iris_license_consumed	現在使用中のライセンスの数
iris_license_percent_used	現在使用中のライセンスの割合
iris_log_reads_per_sec	1 秒あたりの論理読み取り数
iris_obj_a_seize_per_sec	オブジェクト・リソース上の 1 秒あたりの Aseize の数 (詳細は、"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションの "Seize、ASeize、および NSeize に関する注意事項" を参照してください。)
iris_obj_del_per_sec	1 秒あたりに削除されたオブジェクトの数
iris_obj_hit_per_sec	プロセス・メモリ内での 1 秒あたりのオブジェクト参照の数
iris_obj_load_per_sec	共有メモリ内にない、1 秒あたりのディスクからロードされたオブジェクトの数
iris_obj_miss_per_sec	メモリ内で見つからなかった 1 秒あたりのオブジェクト参照の数
iris_obj_new_per_sec	1 秒あたりに初期化されたオブジェクトの数
iris_obj_seize_per_sec	オブジェクト・リソース上の 1 秒あたりの Seize の数 (詳細は、"監視ガイド" の "^mgstat を使用したパフォーマンスの監視" のセクションの "Seize、ASeize、および NSeize に関する注意事項" を参照してください。)
iris_page_space_percent_used	使用済みの最大割り当てページ・ファイル領域の割合
iris_phys_mem_percent_used	現在使用中の物理メモリ (RAM) の割合

メトリック名	説明
iris_phys_reads_per_sec	1 秒あたりのディスクからのデータベース物理ブロック読み取り
iris_phys_writes_per_sec	1 秒あたりのディスクへのデータベース物理ブロック書き込み
iris_process_count	アクティブな InterSystems IRIS プロセスの合計数
iris_rtn_a_seize_per_sec	ルーチン・リソース上の 1 秒あたりの Aseize の数（詳細は、″監視ガイド″ の “^mgstat を使用したパフォーマンスの監視” のセクションの ″Seize、ASeize、および NSeize に関する注意事項″ を参照してください。）
iris_rtn_call_local_per_sec	リモート・データベースにあるグローバルへの 1 秒あたりのローカル・ルーチン呼び出しの数
iris_rtn_call_miss_per_sec	メモリ内で見つからなかった 1 秒あたりのルーチン呼び出しの数
iris_rtn_call_remote_per_sec	1 秒あたりのリモート・ルーチン呼び出しの数
iris_rtn_load_per_sec	ローカルでディスクからロードされた、またはディスクに保存された 1 秒あたりのルーチンの数
iris_rtn_load_rem_per_sec	リモートでディスクからロードされた、またはディスクに保存された 1 秒あたりのルーチンの数
iris_rtn_seize_per_sec	ルーチン・リソース上の 1 秒あたりの Seize の数（詳細は、″監視ガイド″ の “^mgstat を使用したパフォーマンスの監視” のセクションの ″Seize、ASeize、および NSeize に関する注意事項″ を参照してください。）
iris_sam_get_db_sensors_seconds	iris_db* センサの収集に要した時間（秒単位）
iris_sam_get_jrn_sensors_seconds	iris_jrn* センサの収集に要した時間（秒単位）
iris_sam_get_sql_sensors_seconds	iris_sql* センサの収集に要した時間（秒単位）
iris_sam_get_wqm_sensors_seconds	iris_wqm* センサの収集に要した時間（秒単位）
iris_smh_available {id="purpose"}	purpose 別の利用可能共有メモリ（KB 単位）（purpose, の ID のリストを含め、詳細は、″監視ガイド″ の “管理ポータルを使用した InterSystems IRIS の監視” のセクションの ″一般（共有）メモリ・ヒープ使用状況″ を参照してください。）
iris_smh_percent_full {id="purpose"}	purpose 別の使用中の割り当て済み共有メモリの割合（purpose, の ID のリストを含め、詳細は、″監視ガイド″ の “管理ポータルを使用した InterSystems IRIS の監視” のセクションの ″一般（共有）メモリ・ヒープ使用状況″ を参照してください。）
iris_smh_total	現在のインスタンスに割り当てられた共有メモリ（KB 単位）
iris_smh_total_percent_full	現在のインスタンスで使用中の割り当て済み共有メモリの割合

メトリック名	説明
iris_smh_used {id="purpose"}	purpose 別の使用中の共有メモリ (KB 単位) (purpose, の ID のリストを含め、詳細は、"監視ガイド" の "管理ポータルを使用した InterSystems IRIS の監視" のセクションの "一般 (共有) メモリ・ヒープ使用状況" を参照してください。)
iris_sql_active_queries {id="namespace"}	現在実行中の SQL 文の数
iris_sql_active_queries_95_percentile {id="namespace"}	アクティブな SQL 文の現在のセットについて、文の実行開始から 95 パーセンタイルの経過時間
iris_sql_active_queries_99_percentile {id="namespace"}	アクティブな SQL 文の現在のセットについて、文の実行開始から 99 パーセンタイルの経過時間
iris_sql_commands_per_second {id="namespace"}	SQL クエリを実行するために実行された ObjectScript コマンドの秒あたりの平均数
iris_sql_queries_avg_runtime {id="namespace"}	SQL 文の平均実行時間 (秒単位)
iris_sql_queries_avg_runtime_std_dev {id="namespace"}	SQL 文の平均実行時間の標準偏差
iris_sql_queries_per_second {id="namespace"}	1 秒あたりの SQL 文の平均数
iris_system_alerts	システム起動後にメッセージ・ログに送信されたアラートの数
iris_system_alerts_log	現在アラート・ログにあるアラートの数
iris_system_alerts_new	/api/monitor/alerts エンドポイントで新規アラートを利用できるかどうかを示すブーリアン値
iris_system_state	システム・モニタのヘルス状態を示す数値 (詳細は、"監視ガイド" の "システム・モニタの使用" のセクションの "システム・モニタのヘルス状態" を参照してください。)
iris_trans_open_count	現在のインスタンス上の開いているトランザクションの数
iris_trans_open_secs	現在のインスタンス上の開いているトランザクションの平均継続時間 (秒単位)
iris_trans_open_secs_max	現在のインスタンス上の開いているトランザクションの最長継続時間 (秒単位)
iris_wd_buffer_redirty	最近のサイクル中にライト・デーモンが書き込み、前のサイクルでも書き込まれたデータベース・バッファの数
iris_wd_buffer_write	最近のサイクル中にライト・デーモンが書き込んだデータベース・バッファの数

メトリック名	説明
iris_wd_cycle_time	最近のライト・デーモン・サイクルが完了するのに要した時間 (ミリ秒単位)
iris_wd_proc_in_global	最近のライト・デーモン・サイクルの開始時に、グローバル・バッファをアクティブに保持しているプロセスの数
iris_wd_size_write	最近のサイクル中にライト・デーモンが書き込んだデータベース・バッファのサイズ (KB 単位)
iris_wd_sleep	最近のサイクルが始まるまで、ライト・デーモンが非アクティブであった時間 (ミリ秒単位)
iris_wd_temp_queue	最近のサイクルの開始時点でライト・デーモンによって使用済みのメモリ内バッファの数
iris_wd_temp_write	最近のサイクル中にライト・デーモンが書き込んだメモリ内バッファの数
iris_wdwij_time	最近のサイクル中に WIJ ファイルへの書き込みにライト・デーモンが要した時間 (ミリ秒単位)
iris_wd_write_time	最近のサイクル中にデータベースへのバッファの書き込みにライト・デーモンが要した時間 (ミリ秒単位)
iris_wij_writes_per_sec	1 秒あたりの WIJ 物理ブロック書き込み数
iris_wqm_active_worker_jobs {id="category"}	ブロックされていないロジックを実行中のワーカ・ジョブの平均数
iris_wqm_commands_per_sec {id="category"}	この作業キュー管理カテゴリで 1 秒あたりに実行された平均コマンド数
iris_wqm_globals_per_sec {id="category"}	この作業キュー管理カテゴリでの 1 秒あたりの平均グローバル参照実行数
iris_wqm_max_active_worker_jobs {id="category"}	最後のログ・エントリが記録されてからの最大アクティブ・ワーカ数
iris_wqm_max_work_queue_depth {id="category"}	最後のログ以降のこの作業キュー管理カテゴリのキュー内の最大エントリ数
iris_wqm_waiting_worker_jobs {id="category"}	グループが接続および作業を待機している、アイドル状態のワーカ・ジョブの平均数

C.1.1.2 相互運用メトリック

前のセクションで説明したメトリックの他にも、InterSystems IRIS インスタンスでは、アクティブな相互運用プロダクションに関するメトリックを記録し、それを /metrics エンドポイントの出力に含めることもできます。このような相互運用メトリックの記録は、既定で無効になっています。これを有効にするには、監視する相互運用プロダクションごとに以下の手順を実行する必要があります。

1. 監視するプロダクションを実行している InterSystems IRIS インスタンスのターミナル・セッションを開きます。必要に応じて、以下のコマンドを実行して、プロダクションに関連付けられているネームスペースに切り替えます。

```
set $namespace = "[interopNS]"
```

[interopNS] はネームスペース名です。

2. ターミナルで以下のコマンドを実行して、現在のネームスペース内のアクティブなプロダクションに対してメトリックのコレクションを有効にします (SAM は、インターシステムズのモニタリング・ソリューションである System Alerting and Monitoring を指します)。

```
do ##class(Ens.Util.Statistics).EnableSAMForNamespace()
```

注釈　ネームスペースに対してメトリックの記録は有効になっているが、対応するプロダクションがアクティブでない場合、/metrics エンドポイントはいずれのメトリックも返しません。

Ens.Util.Statistics クラスには、/metrics エンドポイントの出力をカスタイマイズするためのメソッドが用意されています。例えば、DisableSAMIncludeHostLabel メソッドを呼び出すと、集約メトリックがホストごとに別々に提供されるのではなく、プロダクション全体の集約メトリックが提供されます。

この手順の完了後に使用できるメトリックについては、以下のテーブル "基本的な相互運用メトリック" を参照してください。

3. プロダクションのアクティビティ量に関する追加のメトリックを収集する場合は、ターミナルを使用して、対応するネームスペースで、クラスメソッド Ens.Util.Statistics.EnableStatsForProduction を呼び出すことでアクティビティ・モニタリングを有効にする必要があります。また、プロダクションに、Ens.Activity.Operation.Local ビジネス・オペレーションを追加することも必要です。このプロセスの詳細は、"アクティビティ量の監視" ページの "アクティビティ・モニタリングの有効化" を参照してください。

この手順の完了後に使用できる追加メトリックについては、以下のテーブル "アクティビティ量メトリック" を参照してください。

4. EnsLib.HTTP.OutboundAdapter または EnsLib.SOAP.OutboundAdapter を使用する相互運用 Web クライアントについて、追加の HTTP 転送メトリックを収集する場合は、以下の手順を実行して、対応するビジネス・オペレーションの HTTP メトリックのレポートを有効にする必要があります。

   a. 監視する Web クライアントを含む InterSystems IRIS インスタンスの管理ポータルを開きます。

   b. **[相互運用性]** を選択して、Web クライアントを含むネームスペースを選択します。

   c. **[構成]→[プロダクション]** を選択して、**[プロダクション構成]** ページを開きます。

   d. HTTP または SOAP 送信アダプタを使用するオペレーションを選択します。

   e. **[プロダクション設定]→[設定]** パネルの **[アラートの制御]** セクションで、**[SAMにメトリックを提供]** チェックボックスにチェックを付けます。

   f. **[適用]** を選択して、設定を保存します。

この手順の完了後に使用できる追加メトリックについては、以下のテーブル "HTTP メトリック" を参照してください。

注釈　現在、HTTP 転送メトリックは、(**[プロセス内]** スタイルではなく) **[キュー]** スタイルを使用してアクターを呼び出すビジネス・オペレーションについてのみ収集されます。これらの呼び出しスタイルの相違についての詳細は、"ビジネス・オペレーションの定義" の "ビジネス・オペレーション・クラスの定義" を参照してください。

以下のテーブルに、InterSystems IRIS 相互運用メトリックを示します。ここでは、読みやすくするために、ラベルがあるメトリック名には改行を入れています。

注釈　これらのテーブルには、ここで説明している InterSystems IRIS のバージョンのメトリックが含まれます。新しいバージョンではメトリックが追加される可能性があるため、このドキュメントがお使いの InterSystems IRIS のバージョンと一致することを確認してください。

テーブル III-1: 基本的な相互運用メトリック

メトリック名	説明
iris_interop_alert_delay {id="namespace",host="host",production="production"}	production および namespace 内で、キュー待機警告をトリガしたホストの数。出力が、ホスト・ラベルを含めるよう構成されている場合、キュー待機警告をトリガしたホストが個別に提供され、値は 1 になります。
iris_interop_hosts {id='namespace',status='status',host='host',production='production'}	production および namespace 内で、現在、指定された status になっているホストの数。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストのステータスが個別に提供され、値は 1 になります。status は、OK、Error（エラー）、Retry（再試行）、Starting（開始中）、Inactive（非アクティブ）、または Unconfigured（未構成）のいずれかです。
iris_interop_messages {id="namespace",host="host",production="production"}	production の開始以降に処理されたメッセージ数。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで処理されたメッセージの数が個別に提供されます。
iris_interop_messages_per_sec {id="namespace",host="host",production="production"}	直近のサンプリング時間において、production および namespace 内で 1 秒あたりに処理されたメッセージの数の平均。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで処理されたメッセージの数が個別に提供されます。
iris_interop_queued {id="namespace",host="host",production="production"}	production および namespace 内で現在キューに入っているメッセージの数。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで現在キューに入っているメッセージの数が個別に提供されます。

テーブル III-2: アクティビティ量メトリック

メトリック名	説明
iris_interop_avg_processing_time {id="namespace",hosttype="hostType",host="host",production="production",msgtype="messageType"}	production および namespace 内で、指定された MessageType のメッセージの処理にかかった平均所要時間 (秒単位)。HostType は、service (サービス)、operation (オペレーション)、または actor (アクター、つまりプロセス) のいずれかです。MessageType は、ユーザ定義です。MessageType を指定しないと、"-" が返されます。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストのメッセージ処理時間が個別に提供されます。
iris_interop_avg_queueing_time {id="namespace",hosttype="hostType",host="host",production="production",msgtype="messageType"}	production および namespace 内で、HostType のホストにより処理される際に、指定された MessageType のメッセージがキューに入っていた平均時間 (秒単位)。HostType は、service (サービス)、operation (オペレーション)、または actor (アクター、つまりプロセス) のいずれかです。MessageType は、ユーザ定義です。MessageType を指定しないと、"-" が返されます。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストのキュー時間が個別に提供されます。
iris_interop_sample_count {id="namespace",hosttype="hostType",host="host",production="production",msgtype="messageType"}	直近のサンプリング時間において、production および namespace 内で、HostType のホストによって処理された、指定された MessageType のメッセージの数。HostType は、service (サービス)、operation (オペレーション)、または actor (アクター、つまりプロセス) のいずれかです。MessageType は、ユーザ定義です。MessageType を指定しないと、"-" が返されます。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで処理されるメッセージの数が個別に提供されます。
iris_interop_sample_count_per_sec {id="namespace",hosttype="hostType",host="host",production="production",msgtype="messageType"}	直近のサンプリング時間において、production および namespace 内で、HostType のホストによって 1 秒あたりに処理された、指定された MessageType のメッセージの数の平均。HostType は、service (サービス)、operation (オペレーション)、または actor (アクター、つまりプロセス) のいずれかです。MessageType は、ユーザ定義です。MessageType を指定しないと、"-" が返されます。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで処理されるメッセージの数が個別に提供されます。

テーブル III-3: HTTP メトリック

メトリック名	説明
iris_interop_avg_http_received_chars {id="namespace",host="host",production="production"}	直近のサンプリング時間において、production および namespace 内で受信された HTTP または SOAP 応答あたりの文字数の平均。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで受信された文字数の平均が個別に提供されます。
iris_interop_avg_http_sent_chars {id="namespace",host="host",production="production"}	直近のサンプリング時間において、production および namespace 内で送信された HTTP または SOAP 要求あたりの文字数の平均。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで送信された文字数の平均が個別に提供されます。
iris_interop_avg_http_ttfc {id="namespace",host="host",production="production"}	最初の文字までの時間 (TTFC) : HTTP または SOAP 要求の開始から対応する応答の最初の文字までの平均時間 (秒単位)。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストの TTFC が個別に提供されます。
iris_interop_avg_http_ttlc {id="namespace",host="host",production="production"}	最後の文字までの時間 (TTLC) : HTTP または SOAP 要求の開始から対応する応答の最後の文字までの平均時間。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストの TTLC が個別に提供されます。
iris_interop_http_sample_count {id="namespace",host="host",production="production"}	直近のサンプリング時間において、production および namespace 内で送信された HTTP または SOAP 転送の数。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで送信された転送数が個別に提供されます。
iris_interop_http_sample_count_per_sec {id="namespace",host="host",production="production"}	直近のサンプリング間隔において、production および namespace 内で送信された HTTP または SOAP 転送の数の秒あたりの平均。出力が、ホスト・ラベルを含めるよう構成されている場合、各ホストで送信された秒あたりの転送数が個別に提供されます。

C.1.1.3 アプリケーションのメトリックの作成

/metrics エンドポイントによって返されたメトリックにカスタム・アプリケーションのメトリックを追加するには、以下の手順に従います。

1. %SYS.Monitor.SAM.Abstract を継承する新規クラスを作成します。
2. PRODUCT パラメータをアプリケーションの名前として定義します。InterSystems IRIS メトリックに予約されている iris 以外の任意の値を指定できます。
3. GetSensors() メソッドを実装して、必要なカスタム・メトリックを次のように定義します。
   - このメソッドには、SetSensor() メソッドへの 1 つ以上の呼び出しが含まれている必要があります。このメソッドは、アプリケーション・メトリックの名前と値を設定します。Prometheus および InterSystems SAM との互換性を確保するために、値は整数または浮動小数点数にする必要があります。

     オプションでメトリックのラベルを定義できますが、定義する場合は常に、その特定のメトリックのラベルを定義する必要があります。

注釈　メトリックおよびラベル名の選択のベスト・プラクティスについては、Prometheus のドキュメントの "Metric and Label Naming" を参照してください (https://prometheus.io/docs/practices/naming/)。

- 成功すると、このメソッドは $$$OK を返します。

重要　GetSensors() の実装が低速であると、システム・パフォーマンスに悪影響を及ぼす可能性があります。GetSensors() の実装が効率的であることをテストして、タイムアウトや停止が発生する可能性のある実装は避けてください。

4. クラスをコンパイルします。以下に例を示します。

Class Definition

```
/// Example of a custom class for the /metric API
Class MyMetrics.Example Extends %SYS.Monitor.SAM.Abstract
{

Parameter PRODUCT = "myapp";

/// Collect metrics from the specified sensors
Method GetSensors() As %Status
{
   do ..SetSensor("my_counter",$increment(^MyCounter),"my_label")
   do ..SetSensor("my_gauge",$random(100))
   return $$$OK
}

}
```

5. SYS.Monitor.SAM.Config クラスの AddApplicationClass() メソッドを使用して、カスタム・クラスを /metrics 構成に追加します。クラスの名前とそれが配置されているネームスペースを引数として渡します。

例えば、ターミナルで %SYS ネームスペースから以下のように入力します。

```
%SYS>set status = ##class(SYS.Monitor.SAM.Config).AddApplicationClass("MyMetrics.Example", "USER")

%SYS>w status
status=1
```

6. /api/monitor Web アプリケーションに、カスタム・メトリックにアクセスするために必要なアプリケーション・ロールがあることを確認します。アプリケーション・ロールの編集方法の詳細は、"アプリケーションの編集：[アプリケーションロール] タブ" を参照してください。

この手順では、カスタム・メトリックに必要なデータへの /api/monitor アクセスを許可します。例えば、カスタム・メトリック・クラスが USER データベース (%DB_USER リソースによって保護されている) にある場合、/api/monitor に %DB_USER ロールを付与します。

7. ブラウザで http://<instance-host>:52773/api/monitor/metrics を指して、/metrics エンドポイントの出力を確認します (52773 は、既定の WebServer ポート)。以下のように、InterSystems IRIS メトリックの後に、定義したメトリックが表示されます。

```
[...]
myapp_my_counter{id="my_label"} 1
myapp_my_gauge 92
```

/metrics エンドポイントは、定義したカスタム・メトリックを返すようになります。InterSystems IRIS メトリックには接頭語 "iris_" が含まれるのに対し、カスタム・メトリックでは PRODUCT の値が接頭語として使用されます。

C.1.2 /alerts エンドポイント

/alerts エンドポイントは、以下のように alerts.log ファイルから最新のアラートを取得して JSON 形式で返します。

```
{"time":"2019-08-15T10:36:38.313Z","severity":2,\
"message":"Failed to allocate 1150MB shared memory using large pages.  Switching to small pages."}
```

/alerts を呼び出すと、前回 /alerts が呼び出されてからこれまでに生成されたアラートを返します。
`iris_system_alerts_new` メトリックは、新規アラートが生成されているかどうかを示すブーリアンです。

アラートが生成されるタイミングと方法の詳細は、このドキュメントの "ログ・モニタの使用" の章を参照してください。

D

irisstat ユーティリティを使用した InterSystems IRIS の監視

このトピックでは、irisstat ユーティリティの使用方法の概要を説明します。これは、初めて使用するユーザは入門書として、熟練ユーザはリファレンスとしてご利用いただけます。

重要　このユーティリティを使用する際、適切な irisstat オプションを指定するためのガイダンスやユーティリティが生成するデータを解釈するためのサポートが必要な場合は、インターシステムズのサポート窓口までお問い合わせください。

irisstat は InterSystems IRIS® Data Platform と共に配布される C 実行可能ファイルで、InterSystems IRIS の停止、ネットワークの問題、およびパフォーマンスの問題を含むシステム・レベルの問題の診断ツールです。実行すると、irisstat は、起動時に InterSystems IRIS によって割り当てられた共有メモリ・セグメントにアタッチし、InterSystems IRIS インスタンスの内部構造とテーブルをユーザが読み取り可能な形式で表示します。共有メモリ・セグメントには、グローバル・バッファ、ロック・テーブル、ジャーナル・バッファ、およびすべての InterSystems IRIS プロセスがアクセスできることが必要な他のさまざまなメモリ構造が含まれます。プロセスはまた、それぞれの変数とスタック情報のために、自らのプロセス・プライベート・メモリを保持します。irisstat の基本的な表示専用オプションは高速で、InterSystems IRIS に対して非侵襲的です。

注意　より詳細な（ドキュメント化されていない）オプションは共有メモリを変更する可能性があるため、注意して使用することが必要です。こうした詳細オプションは、インターシステムズのサポート担当者の指示に従ってのみ使用してください。詳細は、インターシステムズのサポート窓口までお問い合わせください。

D.1 irisstat 実行の基礎

システムの問題が発生した場合、irisstat レポートは多くの場合、インターシステムズが問題の原因を判別するうえでの最も重要なツールになります。以下のガイドラインを使用して、irisstat レポートに必要な情報がすべて含まれるようにしてください。

- イベント発生時に irisstat を実行します。
- インターシステムズのサポート担当者が他の方法を指示した場合を除き、診断レポート・タスクまたは IRISHung スクリプトを使用します。
- irisstat レポートの内容をチェックして、それが有効であることを確認します。

irisstat は InterSystems IRIS に含まれる別個の実行可能ファイルであるため、オペレーティング・システムのプロンプトで InterSystems IRIS の外部で実行されます。したがって、実行の詳細はオペレーティング・システムによって異なります。

- Windows での irisstat の実行
- UNIX® での irisstat の実行

オプションなしで irisstat を実行するのは一般的ではありませんが、オプションなしで実行すると、以下の既定のオプションを使用して実行した場合と同じ基本的なレポートが生成されます。

- -f (グローバル・モジュール・フラグ)
- -p (PID テーブル)
- -q (セマフォ)

irisstat オプションについては、"オプションを使用した irisstat の実行" を参照してください。

D.1.1 Windows での irisstat の実行

irisstat の実行可能ファイルは、install-dir¥bin ディレクトリにあります。Windows のコマンド・プロンプトを管理者として起動すると、以下のようにコマンドを実行できます。

```
C:\>cd install-dir\bin
C:\install-dir\bin>irisstat
```

install-dir¥bin または install-dir¥mgr 以外のディレクトリから irisstat を実行する場合は、-s 引数を使用して、install-dir¥mgr ディレクトリの場所を指定する必要があります。以下に例を示します。

```
C:\Users>\install-dir\bin\irisstat -s\install-dir\mgr
```

D.1.2 UNIX® での irisstat の実行

irisstat の実行可能ファイルは、install-dir/bin ディレクトリにあります。install-dir¥bin または install-dir¥mgr 以外のディレクトリから irisstat を実行する場合は、-s 引数を使用して、install-dir¥mgr ディレクトリの場所を指定する必要があります。

UNIX® のコマンド・プロンプトを root として起動し、install-dir/bin ディレクトリまたは install-dir/mgr ディレクトリに移動し、irisstat コマンドを実行します。

```
bash-3.00$ ./irisstat
```

InterSystems IRIS のインストール・ディレクトリからのコマンドは以下のようになります。

```
bash-3.00$ ./bin/irisstat -smgr
```

また、以下の例で示されているように、irisstat は、任意のディレクトリから実行できる iris コマンドを介しても起動できます。

```
bash-3.00$ iris stat iris_instance_name
```

iris_instance_name は、irisstat を実行する InterSystems IRIS インスタンスの名前です。

D.2 オプションを使用した irisstat の実行

オプションを指定しないで irisstat を実行すると、基本的なレポートが生成されます。通常は、irisstat は特定の情報を取得するために実行します。ターゲット情報を指定するために、以下のようにオプションを含めたり除外したりできます。

- オプションを含める (オンにする) には、フラグを指定し、その後に 1 (または他のレベル) を付けます。
- オプションを除外する (オフにする) には、フラグを指定し、その後に 0 を付けます。

例えば、irisstat レポートにグローバル・ファイル・テーブル (GFILETAB) セクションを含めるには、-m1 オプションを使用します。

```
C:\iris-install-dir\Bin\irisstat -m1
```

既定の基本オプションをオフにするには、-a0 オプションを使用します。

```
C:\iris-install-dir\bin\irisstat -a0
```

多くのオプションでは、0 と 1 より細かいレベルを指定できます。こうした追加のレベルは、「2 の累乗として 10 進数で表示されてオプションに関する特殊な情報を制御する“ビット”を持っている」と説明できます。例えば、PID テーブルを表示する基本的な -p オプションは 1 でオンにできますが、2 を使用すると swcheck 列が追加され、4 を使用すると pstate 列が追加され、以下同様になります。これらのビットは結合できます。例えば、2 ビットと 4 ビットの両方に表示される情報が必要な場合、-p6 と指定します。すべてのビットをオンにするには、次のように -1 を使用します。

```
bash-3.00$ ./irisstat -p-1
```

さらに、複数のフラグを 1 つの irisstat コマンドに結合できます。例えば、以下のコマンドは、基本オプションをオフにしてから、グローバル・モジュール・フラグと PID テーブルの全ビット、および GFILETAB の詳細レベルをオンにします。

```
bash-3.00$ ./irisstat -a0 -f-1 -p-1 -m3
```

複雑な問題の診断を開始する際は irisstat コマンドで多数のフラグを使用するのが一般的ですが、変更を加えるためのオプションは通常は単独で使用します。例えば、-d オプションはプロセス・ダンプを要求します。このオプションを使用する前に、ダンプするプロセスを特定するために、複数のオプションを使用して irisstat を実行することもあります。しかし、-d を使用するときは、通常は他のオプションは選択しません。

″irisstat オプション″ のテーブルでは、irisstat コマンドで使用できるオプションについて説明しています。

注釈　このテーブルに示されている irisstat オプションによって生成されるデータの解釈についてサポートが必要な場合は、インターシステムズのサポート窓口までお問い合わせください。

テーブル IV-1: irisstat オプション

オプション	説明
-a[0/1]	このテーブルに説明されているすべての情報を表示します。

オプション	説明
-b[bits]	グローバル・バッファ記述子ブロック (BDB) に関する情報を表示します。以下のビットの組み合わせを指定できます。 ・ 1 (すべて) ・ 2 (クラスタ) ・ 4 (ECP サーバ) ・ 8 (ECP クライアント) ・ 16 (ブロックのコンテンツ) ・ 64 (ブロックの整合性のチェック) ・ 128 (ブロックおよび LRU 概要) 注釈 irisstat -b64 を実行すると、余計に時間がかかる場合があります。

オプション	説明
-c[bits]	システム・パフォーマンスの統計であるカウンタを表示します。以下のビットの組み合わせを指定できます。 ・ 1 (グローバル) ・ 2 (ネットワーク) ・ 4 (ロック) ・ 8 (最適) ・ 16 (ターミナル) ・ 32 (symtab) ・ 64 (ジャーナル) ・ 128 (ディスク入出力) ・ 256 (クラスタ) ・ 262144 (bshash) ・ 2097152 (ジョブ・コマンド) ・ 4194304 (sem) ・ 8388608 (非同期ディスク入出力) ・ 16777216 (fsync) ・ 33554432 (オブジェクト・クラス) ・ 67108864 (WD) ・ 134217728 (ビッグ・ストリング) ・ 268435456 (SWD) ・ 536870912 (並べ替え) ・ 1073741824 (symsave) ・ 2147483648 (freeblkpool)
-d[pid,opt]	InterSystems IRIS プロセスのダンプを作成します。以下のオプションを指定できます。 ・ 0 (フル)、既定 ・ 1 (一部)
-e[0/1/2]	InterSystems IRIS システム・エラー・ログを表示します (“管理ポータルを使用した InterSystems IRIS の監視” の章にある ″InterSystems IRIS システム・エラー・ログ″ を参照してください)。-e2 を指定すると、追加のプロセス情報が表示されます (16 進数)。

オプション	説明
-f[bits]	グローバル・モジュール・フラグを表示します。以下のビットの組み合わせを指定できます。 ・ 1 (基本) ・ 64 (リソース) ・ 128 (詳細) ・ 256 (アカウント詳細) ・ 512 (incstrtab) ・ 1024 (監査)
-g[0/1]	^GLOSTAT 情報を表示します。詳細は、このドキュメントの “^GLOSTAT を使用したグローバル動作の統計収集” の章を参照してください。
-h	irisstat の使用量情報を表示します。
-j[0/1/2/3/4/5/6]	ジャーナリング状態に関する情報をリストするジャーナル・システムのマスタ構造を表示します。-j32 はミラー・サーバ情報を表示します。
-k	$PREFETCHON 関数によって使用される事前フェッチ・デーモンに関する情報を表示します。″ObjectScript リファレンス″ の ″$PREFETCHON″ を参照してください。
-l[bits]	最近最も使用されていない (LRU) グローバル・バッファ記述子ブロック (BDB) キューに関する情報を表示しますが、BDB のコンテンツは表示しません。以下のビットの組み合わせを指定できます。 ・ 1 (すべて) ・ 2 (クラスタ) ・ 4 (ECP サーバ) ・ 8 (ECP クライアント) ・ 16 (ブロックのコンテンツ) ・ 32、1 (最近使用された (MRU) 順) 以外 注釈 ″-b″ も参照してください。
-m[0/1/3/4/8/16]	InterSystems IRIS のインスタンスの起動以来マウントされ SFN によってリストされる全データベースの情報を含むグローバル・ファイル・テーブル (GFILETAB) を表示します。以下のビットの組み合わせを指定できます。 ・ 3 (詳細) ・ 4 (ボリューム・キュー) ・ 8 (ディスク・デバイス ID テーブル) ・ 16 (このデータベースをリモートからマウントするシステム)

オプション	説明
-n[0/1]	ネットワーク構造およびローカル/リモート SFN 変換に関する情報を表示します。irisstat -n-1 と指定すると、ネームスペース構造も表示します。
-o1	irisstat -c によって表示されたリソースの統計をクリアし、InterSystems IRIS を再起動せずに基本状況を再度確立します。出力は生成されません。
-p[bits]	InterSystems IRIS で実行されるプロセスに関する情報を表示します。この情報はプロセス ID テーブル (PIDTAB) から取得されます。以下のフラグの組み合わせを指定できます。 ・ 2 (swcheck) ・ 4 (pstate と %SS) ・ 5 (NT メールボックスのロック)、Windows のみ ・ 8 (js 合計) ・ 16 (js リスト) ・ 32 (grefcnt 情報) ・ 64 (gstatebits) ・ 128 (gstate 概要) ・ 256 (jrnhib) ・ 512 (トランザクション概要) ・ 1024 (pidflags) ・ 2048 (pgbdbsav)、pgshared テーブルを追加ダンプ ・ 4096 (freeblk テーブル)
-q[0/1]	ハイバネーション・セマフォに関する情報を表示します。
-s[dir]	mgr ディレクトリまたは bin ディレクトリ以外の場所からコマンドを実行したときに irisstat 実行可能ファイルを含むディレクトリを指定します。
-t[seconds]	停止されるまで、seconds 秒ごとにループで irisstat を繰り返し実行します。-f1 を指定したときのように、グローバル・モジュール・フラグ・セクションのみが表示されます。

オプション	説明
-u[bits]	ロック・テーブルに格納された InterSystems IRIS ロックに関する情報を表示します (このドキュメントの “管理ポータルを使用した InterSystems IRIS の監視” の章の ″ロックの監視″ を参照してください)。以下のビットの組み合わせを指定できます。 · 1 (概要) · 2 (待機) · 4 (中間) · 8 (詳細) · 16 (透かし) · 32 (バディ・メモリ) · 64 (リソース情報)
-v1	共有メモリ・セグメント irisstat に関連付けられた InterSystems IRIS 実行可能ファイルが実行中で、irisstat 実行可能ファイルが同じバージョンであることを確認します。このようになっていない場合、irisstat は実行されません。
-w[bits]	ライト・デーモン・キューの BDB に関する情報を表示します。
-B[0/1]	GBFSPECQ に保持されているブロックのコンテンツを 16 進数で表示します。
-C[0/1]	ジョブ間コミュニケーション (IJC)・デバイスの構成情報を表示します。
-D[secs],[msecs][,0]	'secs' 秒間隔のリソース統計情報を表示します。'msec' ミリ秒ごとのブロック衝突をサンプリングします。 注釈 -c と同じリソース情報。 このドキュメントの “^BLKCOL を使用したブロック衝突の監視” の章で説明されている ^BLKCOL ユーティリティは、ブロック衝突に関する詳細情報を提供します。
-E[bits]	クラスタリングをサポートするプラットフォーム上のクラスタの状態を表示します。以下のビットの組み合わせを指定できます。 · 1 (vars) · 2 (ライト・デーモンのロック) · 4 (enqinuse) · 8/16 (allenq)
-G[bdb]	特定のバッファ記述子ブロック (BDB) のグローバル・バッファ記述子とグローバル・バッファのコンテンツを 16 進数で表示します。 注釈 情報が BDB 別に表示されるという点を除き、-H と同じです。

オプション	説明
-H[sfn],[blk]	特定のシステム・ファイル番号 (sfn) とブロック番号 (blk) のペアのグローバル・バッファ記述子とグローバル・バッファのコンテンツを 16 進数で表示します。 注釈　情報がシステム・ファイル番号とブロック番号のペア別に表示されるという点を除き、-G と同じです。 ブロックはバッファ・プールにあることが必要です。
-I[0/1]	インクリメンタル・バックアップ・データ構造を表示します。
-L[0/1]	ライセンスを表示します。 注釈　^CKEY および %SYSTEM.License.CKEY メソッドと同じです。
-M[0/1]	メールボックスのログを表示します。 注釈　既定では無効になっています。メールボックスのメッセージを取得して記録するには、特殊なビルドが必要で、追加のログが必要とされる場合があります。
-N[value]	ECP ネットワーク情報を表示します。以下の値の組み合わせを指定できます。 ・ 1 (クライアント) ・ 2 (サーバ) ・ 4 (クライアント・バッファ) ・ 8 (サーバ・バッファ) ・ 16 (クライアント・バッファの詳細) ・ 32 (応答待ちのユーザ・ジョブ) ・ 64 (サーバ応答バッファ詳細) ・ 128 (グローバル要求) ・ 256 (サーバ送信応答バッファ詳細、-1 以外) ・ 1024 (サーバが受信した要求のバッファをダンプ) ・ 2048 (クライアント trans ビットマップ) ・ 4096 (クライアント GLO Q) ・ 8192 (16 進数のグローバル参照ダンプを要求) ・ 65536 (クライアントにダウンロードされた ECP ブロック) ・ 131072 (クライアントがリリースした要求バッファ詳細、-1 以外)

オプション	説明
-R[value]	使用中 (または変更中) のルーチン・バッファ、クラス制御ブロック (CCB)、および最近最も使用されていない (LRU) キューに関する情報を表示します。以下の値の組み合わせを指定できます。 ・ 1 (使用中のルーチン・バッファ) ・ 4 (RCT – 変更されたルーチン・テーブル) ・ 8 (RCT 詳細) ・ 16 (0x10 = すべてのルーチン・バッファ) ・ 32 (0x20 = LRU キュー) ・ 64 (0x40 = すべての CCB) ・ 128 (0x80 = 無効化された CCB) ・ 0x100 (無効化されたサブクラス) ・ 0x200 (バッファ・アドレス) ・ 0x400 (バッファ記述子) ・ 0x800 (プロシージャ・テーブルとキャッシュされたルーチンのバッファ番号) ・ 0x1000 (プロセスでキャッシュされたルーチン名) ・ 0x2040 (CCB と CCB 詳細) ・ 0x4000 (cls NS キャッシュ) ・ 0x6000 (cls NS キャッシュ詳細) ・ 0x8000 (shm cls キャッシュの検証) ・ 0x10000 (全クラス階層のダンプ) ・ 0x20000 (全クラス階層詳細のダンプ) ・ 0x40000 (プロセス・クラスとルーチン統計のダンプ) ・ 0x80000 (プロセスでキャッシュされたクラス名)
-S[bits]	システムが停止しているかどうかの自己診断に基づいて、停止の原因に関する情報を表示します。以下のビットの組み合わせを指定できます。 ・ 1 (診断を表示) ・ 2 (疑わしいジョブの部分プロセス・ダンプ) ・ 4 (最初の疑わしいジョブの完全プロセス・ダンプと疑わしい他のジョブの部分ダンプ) 注釈 クラスタでは、すべてのクラスタ・メンバについてこのオプションを実行する必要があります。
-T[0/1]	各国言語の設定 (NLS) テーブルを含む多数のメモリ内テーブルの 16 進数値を表示します。

オプション	説明
-V[pid]	プロセス・メモリ構造の一部である変数を表示します。ソース・コードにアクセスできる場合を除き、制限された値になります。 注釈　Windows でのみ有効です。pid.dmp ファイルを含むディレクトリから実行します。
-W	Backup.General.ExternalThaw() クラス・メソッドと同じ機能を実行します。また、新しい InterSystems IRIS セッションを起動できない場合に、Backup.General.ExternalFreeze() が呼び出された後にライト・デーモンを再開するために使用できます (これらのメソッドの使用の詳細は、"データ整合性ガイド" の "バックアップとリストア" の章の "外部バックアップ" を参照してください)。このオプションは、バックアップ以外が原因で発生したハングまたは一時停止からライト・デーモンのフリーズを解除しません。このオプションの使用は、メッセージ・ログに記録されます。
-X[0/1]	デバイス変換テーブルのコンテンツを表示します。これはデバイス番号別にまとめられ、数値と平文の両方でクラス ID を表示します。

D.3 irisstat 出力の表示

irisstat データは、(ターミナルを通して) 即座に表示できます。または、出力ファイルにリダイレクトして、後で分析できます。データを表示するための最も一般的な方法は、以下のとおりです。

- irisstat テキスト・ファイル
- 診断レポート・タスク
- IRISHung スクリプト
- ^SystemPerformance ユーティリティ

注釈　InterSystems IRIS が強制的にシャットダウンされると、システムの現在の状態を取得するために irisstat が実行されます。出力は、緊急シャットダウン・プロシージャの一部としてメッセージ・ログに追加されます。

D.3.1 irisstat テキスト・ファイル

irisstat レポートは、ターミナルの代わりにファイルにリダイレクトできます。これは、InterSystems IRIS ツール (診断レポート・タスク、IRISHung スクリプト、^SystemPerformance ユーティリティ) のいずれかで提供されない一連の irisstat オプションを収集したい場合、またはこれらのツールをうまく実行できない場合に役立つ可能性があります。

D.3.2 診断レポート・タスク

診断レポート・タスクを実行すると、基本情報と詳細情報の両方を記録した HTML 形式のログ・ファイルが作成されます。このファイルは、システムの問題を解決するためにインターシステムズのサポート窓口が使用できます。診断レポート・タスクの詳細 (使用される irisstat オプションを含む) は、このドキュメントの "InterSystems 診断レポートの使用法" の章を参照してください。

注釈　診断レポート・タスクは、停止したシステムでは実行できません。システムが停止した場合は、この付録の "IRISHung スクリプト" を参照してください。

D.3.3 IRISHung スクリプト

IRISHung スクリプトは、InterSystems IRIS インスタンスが停止した場合にシステムのデータを収集するために使用される OS ツールです。このスクリプトは install-dir¥bin ディレクトリにあり、以下のテーブルに示されているように、その名前はプラットフォーム固有になります。

プラットフォーム	スクリプト名
Microsoft Windows	IRISHung.cmd
UNIX®/Linux	IRISHung.sh

IRISHung スクリプトは管理者特権で実行する必要があります。診断レポート・タスクと同じように、IRISHung スクリプトは irisstat を 2 回実行します。状態が変化している場合に備えて、これは 30 秒の間隔を空けて行われ、レポートは収集された他のデータと共に html ファイルにまとめられます。 IRISHung から取得された irisstat レポートは以下のオプションを使用します。

```
irisstat -e2 -f-1 -m-1 -n3 -j5 -g1 -L1 -u-1 -v1 -p-1 -c-1 -q1 -w2 -E-1 -N65535
```

IRISHung はまた、-S2 オプションだけを使用して 3 回目の irisstat を実行し、これは“Self-Diagnosis”という別個の出力セクションに書き込まれます。-S2 オプションを使用すると、疑わしいプロセスがミニダンプを出力します。このため、IRISHung を実行することで、停止の原因となった特定プロセスの情報を収集できる可能性がありますが、単にインスタンスを強制終了しても、この情報は収集できません。

さらに、IRISHung は irisstat 出力ファイルを生成します。この出力ファイルはしばしば非常に大きくなり、その場合は別個の .txt ファイルに保存されます。出力を収集する際は、こうした別個のファイルをチェックすることを忘れないでください。

D.3.4 ^SystemPerformance ユーティリティ

^SystemPerformance ユーティリティは、InterSystems IRIS インスタンスおよびそのインスタンスが稼働するプラットフォームに関する詳細なパフォーマンス・データを収集します。これは設定可能な期間だけ InterSystems IRIS 内部で実行され、その期間のサンプルを収集し、終了するとレポートを生成します。^SystemPerformance ユーティリティの詳細 (使用される irisstat オプションを含む) は、このドキュメントの“^SystemPerformance を使用したパフォーマンスの監視”の章を参照してください。