



Caché 監視ガイド

Version 5.1

2006-03-14

Caché 監視ガイド

Caché Version 5.1 2006-03-14

Copyright © 2006 InterSystems Corporation.

All rights reserved.

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



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



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



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

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

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

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

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

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

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

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

InterSystems ワールドワイド カスタマサポート

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

目次

はじめに	1
1 システム・ダッシュボードを使用した Caché の監視	3
1.1 システム・ダッシュボードのインジケータの監視	3
1.1.1 システム・パフォーマンス	4
1.1.2 ECP とシャドウイング	5
1.1.3 システム時刻	5
1.1.4 システム使用	6
1.1.5 エラーとアラート	6
1.1.6 ライセンス	7
1.1.7 タスク・マネージャ	7
1.2 ロックの監視	7
1.2.1 ロック・モード	8
1.3 ログ・ファイルの監視	10
1.4 システム・パフォーマンスの監視	10
1.5 CSP セッションの表示	11
1.6 バックグラウンド・タスクの表示	12
2 Caché モニタ	13
2.1 モニタ・アーキテクチャ	13
2.2 システム・モニタ	14
2.2.1 システム・モニタ・マネージャ	14
2.3 アプリケーション・モニタ	16
2.4 Caché メトリック・クラス	16
2.5 メトリック	17
2.5.1 システム活動カウンタ	17
2.5.2 グローバル・メトリック	18
2.5.3 サーバ (%Monitor.System.Servers)	19
2.5.4 クライアント (%Monitor.System.Clients)	19
2.5.5 空き容量 (%Monitor.System.Freespace)	20
2.5.6 監査ログ (%Monitor.System.Audit)	20
2.5.7 コンソール・ログ (%Monitor.System.Console)	20
2.6 アラート	20
2.7 アプリケーション・モニタの管理	21
2.7.1 Manage Application Monitor (アプリケーション・モニタの管理)	21
2.7.2 Manage Monitor Classes (モニタ・クラスの管理)	22
2.7.3 Manage Email Options (電子メール・オプションの管理)	22
2.7.4 Manage Alerts (アラートの管理)	23

2.8 ユーザ定義モニタ・クラスの記述	23
2.8.1 ユーザ定義モニタ・クラスの例	24
2.9 メトリックの生成	27
2.10 メトリック・データの表示	28
2.11 Caché モニタのエラー	28
3 ^GLOSTAT を使用したグローバル動作の統計収集	31
3.1 ^GLOSTAT の実行	31
3.2 ^GLOSTAT 統計の概要	33
3.3 ^GLOSTAT の出力例	35
3.3.1 例 A	35
3.3.2 例 B	36
3.3.3 例 C	36
3.3.4 例 D	37
3.3.5 例 E	39
3.3.6 例 F	39
3.3.7 例 G	40
4 ^PERFMON を使用したシステム・パフォーマンスの監視	43
4.1 PERFMON の実行	45
4.1.1 PERFMON レポート	46
5 %SYS.MONLBL を使用したルーチン・パフォーマンスの検証	51
5.1 監視の開始	51
5.2 監視オプション	54
5.3 サンプル出力	54
5.4 プログラミング・インタフェース	56
A: BMC PATROL を使用した Caché の監視	57
A.1 Caché での PATROL の実行	57
A.1.1 Caché PATROL ルーチン	58
A.1.2 PATROL の自動起動	59
A.2 Caché PATROL ナレッジ・モジュール	59
A.2.1 Caché モジュールを PATROL に追加	60
A.3 BMC PATROL で使用される Caché メトリック	61
B: SNMP を使用した Caché の監視	65
B.1 Caché での SNMP の実行	65
B.2 Caché での SNMP の管理	66
B.3 サブエージェントとしての Caché	67
B.4 Caché MIB 構造	67
B.4.1 Caché MIB の拡張	68

テーブル一覧

システム・パフォーマンスのインジケータ	4
ECP とシャドウイングのインジケータ	5
システム時刻のインジケータ	5
システム使用のインジケータ	6
エラーとアラートのインジケータ	6
ライセンスのインジケータ	7
タスク・マネージャ - タスクの実行予定タスク	7
ロック・テーブルの情報	8
ロック・モード	9
表示される統計	11
アラート入力フィールド	23
^GLOSTAT が生成する統計	34
カスタム・レポートのフィールド	48
Caché PATROL メトリック	61

はじめに

このガイドでは、Cache を監視する際に使用するさまざまなツールを紹介します。システム管理ポータルを監視するための操作、および使用可能なユーティリティとインタフェースについて説明します。このガイドは、以下の各章と付録で構成されています。

- ・ システム・ダッシュボードを使用した Cache の監視
- ・ Cache モニタ
- ・ ^GLOSTAT を使用したグローバル動作の統計収集
- ・ ^PERFMON を使用したシステム・パフォーマンスの監視
- ・ ^%SYS.MONLBL を使用したルーチン・パフォーマンスの検証

以下の付録では、各種サードパーティ製品を使用して Cache を監視する方法を説明します。

- ・ BMC PATROL を使用した Cache の監視
- ・ SNMP を使用した Cache の監視

1

システム・ダッシュボードを使用した Caché の監視

システム管理ポータルシステム・ダッシュボードでは、Caché インスタンスをさまざまな側面から監視できます。このページを使用してパフォーマンス・インジケータを表示し、さらに他の操作ページへ移動して、特定のトピックに関する詳細情報を参照できます。この章では、以下の監視タスクについて説明します。

- ・ [システム・ダッシュボードのインジケータの監視](#)
- ・ [ロックの監視](#)
- ・ [ログ・ファイルの監視](#)
- ・ [システム・パフォーマンスの監視](#)
- ・ [CSP セッションの表示](#)
- ・ [バックグラウンド・タスクの表示](#)

1.1 システム・ダッシュボードのインジケータの監視

システム管理ポータルの [[ホーム]→[システム・ダッシュボード]] ページには、重要なシステム・パフォーマンス・インジケータの状態がリアルタイムで表示されます。対象となるカテゴリは以下の通りです。

- ・ [システム・パフォーマンス](#)
- ・ [ECP とシャドウイング](#)

- ・ システム時刻
- ・ システム使用
- ・ エラーとアラート
- ・ ライセンス
- ・ タスク・マネージャ

1.1.1 システム・パフォーマンス

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。
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 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))。

上記いずれかのシステム・パフォーマンス・インジケータをクリックすると、下部の詳細ボックスに、そのメトリックの説明および [\[\[ホーム\]→\[システム使用\]\]](#) ページへのリンクが表示されます。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。

1.1.2 ECP とシャドウイング

ECP とシャドウイングのインジケータ

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 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)。
シャドウ・クライアント	このシステムに接続されているシャドウ・クライアントの状態。
シャドウ・サーバ	このシステムにデータ・ソースとして接続されているシャドウ・サーバの状態。

1.1.3 システム時刻

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。

1.1.4 システム使用

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。[[ホーム]→[データベース]→[空き容量] ページで詳細を確認できます。
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。
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。
ロック・テーブル	システム・ロック・テーブルの現在の状態を表します。
システム・プロセス	システムのさまざまなバックグラウンド・プロセスの状態。
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を表します。
プロセス	現在、実行されているプロセスの数。
CSP セッション	最新の CSP セッション数。
最もアクティブなプロセス	動作 (実行されたコードの行数) が最も多い実行プロセス。

1.1.5 エラーとアラート

エラーとアラートのインジケータ

インジケータ	詳細ページへのリンク	定義
深刻な警告	[[ホーム]→[システム・ログ]→[コンソール・ログの表示]]	深刻な警告の発生数。
アプリケーション・エラー	[[ホーム]→[システム・ログ]→[アプリケーション・エラー・ログの表示]]	ログに記録されたアプリケーション・エラーの数。

エラーとアラートのインジケータをクリックすると、そのメトリックの説明と、該当する [[ホーム]→[システム・ログ]] ページへのリンクが下部の詳細ボックスに表示されます。詳細は、“[ログ・ファイルの監視](#)” のセクションを参照してください。

1.1.6 ライセンス

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	使用可能なライセンス数に占める最大使用数の割合が表示されません。

上記いずれかのライセンス・インジケータをクリックすると、下部の詳細ボックスに、そのメトリックの説明と[[ホーム]→[ライセンスの使用]→[ライセンスのアクティビティの要約]] ページへのリンクが表示されます。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。

1.1.7 タスク・マネージャ

タスク・マネージャ - タスクの実行予定タスク

列名	定義
タスク	これらのタスクは次の 1 時間以内に実行されます。最大 5 つのタスクが表示されます。
時刻	
状態	

1.2 ロックの監視

ローカル変数やグローバル変数など、Cache のエンティティに対して Cache プロセスが Lock コマンドを発行した場合、そのエンティティが他のプロセスによってすでにロックされていない限り、Cache のロックが作成されます。ロックする項目は、データベースに存在していなくてもかまいません。

ロックを表示または削除するには、システム管理ポータルホーム・ページにある **[運用]** セクションから、[[ホーム]→[ロック]] ページを表示します。

[ロック・テーブル] には以下の列があります。

ロック・テーブルの情報

列名	定義
所有者	ロックを保持しているプロセスのプロセス ID。
モード/カウント	ロック項目のロック・モードとカウント。
参照	ロックされている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	
システム	
Exclusive	排他ロック・モードが適用されています。カウントの最後に “D” が追加されている場合、処理中のトランザクションでロックされていないことを示しています。
Shared	共有ロック・モードのロック・カウント。
待機数	このロックを待機しているジョブの総数。
フラグ	ロックの属性。

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、この列に NULL 文字列が表示されます。削除可能: このロック (行) が削除可能かどうかを表すフラグ。値 1 は削除可能、値 0 は削除不可能、値 2 は削除可能 (ただし、“DelockPending” 状態のため注意が必要) です。

1.2.1 ロック・モード

モード/カウント: ロック項目のロック・モードとカウント。

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。それ以外の場合は “/count” が追加されます。ロックが “Deferred Delock” (トランザクション内でロック解除) 状態の場合、“->Delock” が追加されます。

以下の列があります。

ロック・モード

Lock	排他ロック・モード。
Share	共有ロック・モード。
LockZA	ZALLOCATE ロック・モード。
WaitLock	排他ロック・モードの待機。
WaitShare	共有ロック・モードの待機。
WaitLockZA	ZALLOCATE ロック・モードの待機。
LockPending	排他ロック・ペンディング。サーバから排他ロックを許可されるのを待機している状態。
SharePending	共有ロック・ペンディング。サーバから共有ロックを許可されるのを待機している状態。
DelockPending	ロック解除ペンディング。サーバによるロック解除を待機している状態。
Lost	ネットワークがリセットされたため、ロックが失われました。

詳細は、SYS.Lock クラス情報を参照してください。

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、“Caché ObjectScript リファレンス” ガイドの “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。ロック・テーブルのサイズを変更するには、システム管理ポータルで以下のように操作します。

1. [[ホーム]→[構成]→[詳細設定]] ページへ移動し、[フィルタ] ボックスに「Lock」と入力します。ロック関連の項目のみがリストに表示されます。
2. [LockTableSize] 行で [編集] をクリックし、[構成設定] ページを表示します。
3. [値] ボックスで、そのシステムでロックに割り当てるメモリ容量 (バイト単位) を変更し、[OK] をクリックします。

最小値は 65536 です。最大値は [一般メモリ・ヒープ] の値によって異なります。ロック・テーブルにそれ以上の領域が必要な場合は、ヒープ・サイズを増やしてください。Caché では、値が 64 KB の倍数に丸められます。既定の範囲は 65536 1769472 です。

4. [変更を適用] をクリックします。Caché を再起動すると、新しい値が有効になります。

1.3 ログ・ファイルの監視

Caché では、オペレータ・コンソールを通じて、一般的なメッセージ、システム・エラー、オペレーティング・システム・エラー、およびネットワーク・エラーが報告されます。Windows ベースの Caché システムには、このようなオペレータ・コンソールがありません。すべてのコンソール・メッセージはログ・ファイル `cconsole.log` に送信されます。このログ・ファイルはシステム管理者のディレクトリ (通常は `CacheSysMgr`) にあります。UNIX ベースまたは OpenVMS プラットフォーム上の Caché システムでは、オペレータ・コンソール・メッセージをコンソール・ログ・ファイルまたはコンソール・ターミナルに送信できます。

コンソール・ログ・ファイルは単純なテキスト・ファイルなので、一般的なエディタやテキスト・ビューアで参照できます。また、次のように、システム管理ポータルを使用して表示することもできます。

1. ホーム・ページの [運用] 列で [システム・ログ] をクリックして、[[ホーム]→[システム・ログ]] ページに移動します。
2. [システム・ログ] 列で [コンソール・ログの表示] をクリックして、コンソール・ログ・ファイルのテキスト出力を表示します。

Caché を起動できない場合は、コンソール・ログ・ファイルを表示することを推奨します。`cconsole.log` ファイルが最大サイズに達するまで増加すると、`cconsole.old` という名前に変更されます。最大ファイル・サイズは、システム管理ポータルの [[ホーム]→[構成]→[詳細設定]] ページで設定できます (既定値は 5 MB)。ConsoleLogSize 設定の MB 数を変更してください。

さらに、システム管理ポータルの [[ホーム]→[システム・ログ]→[アプリケーション・エラー・ログの表示]] ページおよび [[ホーム]→[システム・ログ]→[ODBC エラー・ログの表示]] ページでは、それぞれアプリケーション・エラーと ODBC エラーに関するログ情報を参照できます。

1.4 システム・パフォーマンスの監視

ここでは、システム管理ポータルの [システム使用] ページの出力について説明します。システム・パフォーマンスの監視に役立つメトリックを表示するには、[[ホーム]→[システム使用]] ページに移動します。これらの統計情報を取得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](#)” を参照してください。

表示される統計

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。
ルーチン呼び出し	
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・メモリにすでに格納されており、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、ディスクから読み取られた物理データベース・ブロック (2 KB または 8 KB) 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 (2 KB または 8 KB) の数。
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	
最終更新	

1.5 CSP セッションの表示

現在有効になっているすべての CSP セッションを表示するには、[[ホーム]→[CSP セッション]] ページへ移動します。

1.6 バックグラウンド・タスクの表示

完了したタスクまたは実行中のバックグラウンド・タスクのログを表示するには、[[バックグラウンド・タスク]] ページへ移動します。

2

Caché モニタ

Caché システム・モニタは、拡張可能なシステム・モニタ・ユーティリティです。パフォーマンス・データとシステム・データを収集し、監視するためのサービス、アラート生成サービス、通知処理サービス、これらデータの永続リポジトリを生成および管理するサービスを提供します。

2.1 モニタ・アーキテクチャ

Caché モニタは、複数のモニタ・インスタンスを複数のネームスペースで実行できるモジュール・システムです。

それぞれが有効化されているネームスペースでは、どのモニタ・インスタンスも同じように動作します。

- ・ システムを起動すると、Caché モニタはまず、どのモニタ・メトリックが有効化されているかを調べます。有効な各メトリックについて、呼び出すべきメトリック・クラスが指定されているメタデータと、それらの各クラスに対して定義されているアラートを読み込みます。
- ・ 次に、有効な各メトリック・クラスの標準メソッドを呼び出します。各メトリック・クラスから返されたメトリック・データを検証し、その結果に応じて以下のように処理します。
 - メトリック・データが永続としてマークされている場合は、永続サンプル・クラスを保存します。
 - そのクラスにアラートが定義されている場合は、サンプル・データをアラート・ハンドラに渡します。アラート・ハンドラはしきい値に達しているかどうかを調べ、達している場合は、ユーザの指示に従って通知を実行します。

2.2 システム・モニタ

システム・モニタは、システムの特種な利用状況に合わせて、標準の Caché モニタを少し変更したものです。

Caché を起動すると、システム・モニタが %SYS ネームスペースで起動されます。システム・モニタは、このモニタで動作するように設計された内部 Caché システム・トラップを無条件に監視します。永続データは保持されません。

アラートは、標準モニタの場合と同じように処理されます。システム・トラップが発生するとアラートが生成され、ユーザが定義した指示に従って電子メールで通知されます。電子メール・オプションの設定については、この後の説明を参照してください。

システム・モニタによって処理されるシステム・トラップについては、“Caché モニタのエラー”を参照してください。

2.2.1 システム・モニタ・マネージャ

システム・モニタ・マネージャ (^MONMGR) は、システム・モニタ管理機能を実行するための文字ベースのアプリケーションです。このアプリケーションは必ず %SYS ネームスペースで実行され、システム・モニタのパラメータのみを設定します。

^MONMGR のオプションは以下の通りです。

```
%SYS>Do ^MONMGR
```

- 1) Manage MONITOR
- 2) Set Email options
- 3) List Email options
- 4) Test Email options
- 5) Set Sample Interval
- 6) Exit

Option?

システム・モニタの起動、停止、更新。モニタの更新を選択すると、システムを・モニタを再起動しなくても、新しい電子メール・オプションとサンプル間隔を確認できます。

Option? 1

- 1) Refresh MONITOR
- 2) Halt MONITOR
- 3) Start MONITOR
- 4) Exit

Option?

電子メールで通知する際の電子メール・オプション。ここでは以下の項目を指定します。

1. Mail server (メール・サーバ) – その環境で、電子メールを処理する電子メール・サーバの名前を指定します。この情報は、各環境の IT 担当者に問い合わせてください。
2. Recipients (受信者) – 電子メールの受信者リストです。有効な電子メール・アドレスを指定する必要があります。
3. Sender (送信者) – 電子メールの送信者を表すテキストです。返信可能な電子メール・アカウントでなくてもかまいません。

Option? 2

```
Mail server? doc.server
Recipients? userdoc@company.com
Sender? userdoc@company.com
```

- 1) Manage MONITOR
- 2) Set Email options
- 3) List Email options
- 4) Test Email options
- 5) Set Sample Interval
- 6) Exit

Option? 3

```
Mail server: doc.server

Recipients: userdoc@company.com

Sender: userdoc@company.com
```

- 1) Manage MONITOR
- 2) Set Email options
- 3) List Email options
- 4) Test Email options
- 5) Set Sample Interval
- 6) Exit

Option? 4

```
Sending email on Mail Server doc.server
From: userdoc@company.com
To: userdoc@company.com
```

- 1) Manage MONITOR
- 2) Set Email options
- 3) List Email options
- 4) Test Email options
- 5) Set Sample Interval
- 6) Exit

Option?

システム・モニタを一度起動すると、それ以降は繰り返し実行されます。サンプル収集とアラート処理の既定の時間間隔は 10 秒です。この時間を変更するには、該当するオプションを設定します。

2.3 アプリケーション・モニタ

アプリケーション・モニタでは、Caché システムで定義されているメトリック、およびユーザが定義したメトリックを監視できます。ユーザは最初に、システム・クラスをローカル・ネームスペースに登録し、独自に定義したメトリック・クラスを作成します。次に、監視対象となるクラスを有効にして、アプリケーション・モニタ (%MONAPP) のインスタンスをローカル・ネームスペースで起動します。

アプリケーション・モニタをローカル・ネームスペースで起動すると、有効化されているすべてのクラスが検索されます。さらに、有効化されている各クラスについて、ローカル・ネームスペース・グローバルにサンプル・データが作成されます。モニタ・メタデータは、グローバル ^ISCMonitor(…) に保持されます。サンプル・データは、サンプル・クラスの永続データとして保持されます。

このデータを表示するには、クラス・メソッドを呼び出すか、またはブラウザを使用します。すべてのサンプル・クラスは自動的に CSP 対応になります。

この後の、アプリケーション・モニタの管理、およびユーザ定義モニタ・クラスの記述に関するセクションを参照してください。

```
^ISCMonitor("Monitor", "AppInterval")=300
^ISCMonitor("Monitor", "Interval")=10
^ISCMonitor("Monitor", "Metric", "%Monitor.System.Dashboard", "Active")=1
^ISCMonitor("Monitor", "Recipients")=""
^ISCMonitor("Monitor", "SmtServer")=""
```

2.4 Caché メトリック・クラス

Caché システムで事前に定義されているメトリック・クラスは以下の通りです。

- ・ %Monitor.System.Audit
- ・ %Monitor.System.Clients
- ・ %Monitor.System.Console
- ・ %Monitor.System.Freespace
- ・ %Monitor.System.Globals
- ・ %Monitor.System.Processes
- ・ %Monitor.System.Routines
- ・ %Monitor.System.Servers
- ・ %Monitor.System.SystemMetrics

これらのクラスは、システム活動のカウンタを監視します。現在のネームスペース (`%Monitor.System.Globals`) のすべてのグローバルについて、グローバル・メトリックを取得することもできます。システム・メトリックは、全体的なシステム活動 (`%Monitor.System.SystemMetrics`) について取得できます。さらに、以下を対象にしたシステム・メトリックとグローバル・メトリックを取得することができます。

- ・ ルーチン - 現在実行されている各ルーチン (`%Monitor.System.Routines`)
- ・ プロセス - 各キャッシュ・プロセス (`%Monitor.System.Processes`)

2.5 メトリック

以下の種類のメトリックがあります。

- ・ システム活動カウンタ
- ・ グローバル・メトリック
- ・ サーバ・メトリック
- ・ クライアント・メトリック
- ・ 空き容量メトリック
- ・ 監査メトリック
- ・ コンソール・メトリック

2.5.1 システム活動カウンタ

- ・ 送信されたネットワーク・バッファ
- ・ 読み取られたローカル IJC メッセージ
- ・ 書き込まれたローカル IJC メッセージ
- ・ 書き込まれたネットワーク IJC メッセージ
- ・ ロック・コマンド
- ・ 失敗したロック・コマンド
- ・ 成功したロック・コマンド
- ・ ネットワーク・ロック
- ・ ネットワーク再送
- ・ ルーチンの取得

- ・ 実行されたルーチン行
- ・ ルーチン・ロード
- ・ シーケンシャル・リード
- ・ シーケンシャル・ライト
- ・ ターミナル読み取り文字数
- ・ ターミナル書き込み文字数
- ・ ターミナル読み取り
- ・ ターミナル書き込み

2.5.2 グローバル・メトリック

各グローバルについて、以下のメトリックが監視されます。

- ・ 割り当てブロック
- ・ バッファによって満たされた下部ポインタ・ブロック要求
- ・ 下部ポインタ・ブロックの読み取り数
- ・ 下部ポインタ・ブロックの書き込み数
- ・ バッファによって満たされたデータ・ブロック要求
- ・ データ・ブロックの読み取り数
- ・ データ・ブロックの書き込み数
- ・ グローバル・バッファによって満たされたディレクトリ・ブロック要求
- ・ ディレクトリ・ブロックの読み取り数
- ・ ディレクトリ・ブロックの書き込み数
- ・ ジャーナル・エントリ
- ・ グローバル Kill
- ・ グローバル・バッファによって満たされたマップ・ブロック要求
- ・ マップ・ブロックの読み取り数
- ・ マップ・ブロックの書き込み数
- ・ ネットワーク・キャッシュ・ヒット
- ・ ネットワーク・キャッシュ・ミス
- ・ ネットワーク Kill

- ・ ネットワーク・グローバル参照
- ・ ネットワーク Set
- ・ 送信されたネットワーク要求
- ・ グローバル・バッファによって満たされたその他のブロック要求
- ・ その他のブロックの読み取り数
- ・ その他のブロックの書き込み数
- ・ グローバル参照
- ・ グローバル・バッファによって満たされたルーチン・ブロック要求
- ・ ルーチン・ブロックの読み取り数
- ・ ルーチン・ブロックの書き込み数
- ・ グローバル Set
- ・ 物理ブロックの合計読み取り数 (合計 * BlkRd カウンタ)
- ・ グローバル・バッファによって満たされた上部ポインタ・ブロック要求
- ・ 上部ポインタ・ブロックの読み取り数
- ・ 上部ポインタ・ブロックの書き込み数

2.5.3 サーバ (%Monitor.System.Servers)

各システム・サーバについて、以下のメトリックが監視されます。

- ・ バッファの割り当て
- ・ 受信したグローバル Kill
- ・ 受信したグローバル参照
- ・ 受信したグローバル Set
- ・ IJC デバイスが受信し、そのデバイスに書き込まれたネット IJC メッセージ
- ・ IJC デバイスが受信したが、書き込まれなかったネット IJC メッセージ
- ・ 受信したロック・コマンド
- ・ 受信した要求

2.5.4 クライアント (%Monitor.System.Clients)

各システム・クライアントについて、以下のメトリックが監視されます。

- ・ クライアントのバッファ
- ・ クライアントに送信されたグローバル Kill
- ・ クライアントに送信されたグローバル参照
- ・ クライアントに送信されたグローバル Set
- ・ ネットワーク・キャッシュ・ヒット
- ・ クライアントの発信ネットワーク IJC 書き込み
- ・ クライアントに送信されたロック・コマンド
- ・ ネットワーク・キャッシュ・ミス
- ・ クライアントから送信された要求
- ・ クライアントの再送

2.5.5 空き容量 (%Monitor.System.Freespace)

システムで定義されている各ネームスペースについて、現在の空き容量 (単位は MB) が監視されます。

2.5.6 監査ログ (%Monitor.System.Audit)

監査レポートの最新のレコードが一覧表示されます。

2.5.7 コンソール・ログ (%Monitor.System.Console)

cconsole.log ファイルを監視します。深刻度レベルごとにアラートを定義できます。システム・モニタは常に cconsole.log ファイルを監視し、指定した深刻度レベルまたはそれより高いレベルの問題が生じた場合にアラートを発します。

2.6 アラート

アラートは、クラス内で、監視対象となるプロパティのリストを指定します。例えば、“%1 < 100 && %1 > 20” のような式を指定します。モニタ・サンプリングの最中、クラスのプロパティ値が評価されます。式の評価が真の場合、アラートがトリガされます。アラートがトリガされると、通知アクションが必ず実行されます。

式では、パラメータが実際の値に置き換えられます。%1 は、監視する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。

アラートを定義するには、モニタ・マネージャのアラート管理機能を使用します。

2.7 アプリケーション・モニタの管理

モニタ・マネージャ(%MONAPPMGR)は、ユーザが管理機能を実行するための文字ベースのユーティリティです。このユーティリティはユーザのネームスペースで実行され、そのネームスペースのパラメータのみを設定します。したがって、有効化されているクラスと通知オプションが異なる複数のマネージャを同時に実行することができます。

```
%SYS>Do ^%MONAPPMGR
```

- 1) Manage Application Monitor
- 2) Manage Monitor Classes
- 3) Manage Email options
- 4) Manage Alerts
- 5) Exit

Option?

2.7.1 Manage Application Monitor (アプリケーション・モニタの管理)

Option? 1

- 1) Start Application Monitor
- 2) Halt Application Monitor
- 3) Refresh Application Monitor
- 4) Activate/Deactivate a Monitor Class
- 5) Set Sample Interval
- 6) Exit

Option?

モニタの起動、停止、更新 – モニタを更新すると、有効なクラスの最新リストが表示されます。

Activate/Deactivate a Monitor Class (モニタ・クラスの有効化/無効化) – ネームスペースで、“Monitor-enabled”として定義されているすべてのクラスのうち、有効化されているクラスのみがアプリケーション・モニタによってサンプリングされます。

間隔 – システム・モニタを一度起動すると、それ以降はモニタが繰り返し実行されます。サンプル収集とアラート処理の既定の時間間隔は 20 秒です。この時間を変更するには、該当するオプションを設定します。

2.7.2 Manage Monitor Classes (モニタ・クラスの管理)

Option? 2

- 1) Activate/Deactivate a Monitor Class
- 2) List Monitor Classes
- 3) Register Monitor System Classes
- 4) Exit

Option?

1. Activate/Deactivate a Monitor Class (モニタ・クラスの有効化/無効化) – ネームスペースで、“Monitor-enabled”として定義されているすべてのクラスのうち、有効化されているクラスのみがアプリケーション・モニタによってサンプリングされます。
2. List Monitor Classes (モニタ・クラスのリスト表示) – 有効かどうかにかかわらず、ネームスペースで定義されているすべてのモニタ・クラスが表示されます。
3. Register Monitor System Classes (モニタ・システム・クラスの登録) – システム・クラスは、それを監視するローカル・ネームスペースに登録する必要があります。システム・クラスを登録することによって、モニタ・グローバルをローカル・ネームスペースに設定し、定義されているクラスの有効化状態をモニタに知らせます。

Option? 2

Class	Active
%Monitor.System.AuditCount	N
%Monitor.System.Clients	N
%Monitor.System.Dashboard	Y
%Monitor.System.Database	N
%Monitor.System.Freespace	N
%Monitor.System.Globals	N
%Monitor.System.Processes	N
%Monitor.System.Routines	N
%Monitor.System.Servers	N
%Monitor.System.SystemMetrics	N

- 1) Activate/Deactivate a Monitor Class
- 2) List Monitor Classes
- 3) Register Monitor System Classes
- 4) Exit

Option?

2.7.3 Manage Email Options (電子メール・オプションの管理)

Option? 3

- 1) Set Email options
- 2) List Email options
- 3) Test Email options
- 4) Exit

Option?

電子メール通知は、システム・モニタと同じ方法で構成できます。電子メールの定義については、システム・モニタに関するセクションを参照してください。

定義したアラートは、通知の手段として電子メールを使用するように構成できます。アラートに関する以下のセクションを参照してください。

2.7.4 Manage Alerts (アラートの管理)

Option? 4

- 1) Create Alert
- 2) Delete Alert
- 3) List Alerts
- 4) Exit

Option?

- 1) Create Alert (アラートの作成) – アラートを作成するには、以下を指定します。

アラート入力フィールド

Alert Name	アラートの名前。
Application Name	アラートの修飾子。ユーザ定義の“アプリケーション”ごとにアラートをグループ化します。
Action	なし、電子メールで通知、またはユーザ・クラスメソッドで通知。
Notify Method	完全なクラス名とメソッド (A.B.MyNotify など)。動作が“クラスメソッドを通じて通知”の場合、アラートがトリガされた時点で、値リストに基づいてそのメソッドが呼び出されます。
Class	アラートを監視するメトリック・クラスのクラス名。
Properties	評価されるモニタ・クラスのプロパティ。
Expression	プロパティを評価するための式。例えば、“%1 = “User” && %2 < 100”などの式を指定します。%1 は、プロパティ・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。

2.8 ユーザ定義モニタ・クラスの記述

Cache モニタには、システム・メトリックを監視するために事前定義されたシステム・クラスが用意されています。さらに、必要であれば、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・クラスを記述することもできます。

モニタ・クラスは、抽象モニタ・クラス %Monitor.Abstract を継承するクラスです。例えば、%Monitor.System クラスもモニタ・クラスの 1 つです。独自のモニタ・クラスを記述するには、以下の手順に従います。

1) %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、およびユーザー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しなければならないコードの詳細は、%Monitor.Adaptor クラスに関するドキュメントを参照してください。

2) 記述した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 Sample のサブパッケージに新しい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、A.B.Sample.MyMetric に新しいクラスが生成されます。ユーザは、このクラスについて何もする必要がありません。

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。

モニタはこのクラスを自動的に呼び出し、このクラスが有効化されている場合はデータとアラートを生成します。クラスの有効化については、モニタ・マネージャを参照してください。

2.8.1 ユーザ定義モニタ・クラスの例

ここでは、キャッシュ構成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・オブジェクトを使用します。各インスタンスはデータセットに対応しています。また、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.Abstract を継承するクラスを作成します。

```
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。これらのプロパティは %Monitor タイプであることが条件となります。

```
Class MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
  /// Name of dataset
  Property DBName As %Monitor.String;

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

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 MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
Parameter INDEX = "DBName";
}
```

4. 必要に応じて、コントロールのプロパティを追加します。

```
Class MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
/// Result Set
Property Rspec As %Library.ResultSet;
}
```

5. Initialize という名前の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 MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
/// Initialize the list of datasets and freespace.
Method Initialize() As %Status
{
s ..Rspec = ##class(%Library.ResultSet).%New("%SYSTEM.Database:FreeSpace")
d ..Rspec.Execute("*",0)
Quit $$$OK
}
}
```

6. GetSample という名前の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 MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
/// 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 Q 0

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

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

7. 7. クラスをコンパイルします。完成したクラスは以下のようになります。

```

Class MyMetric.Freespace Extends %Monitor.Adaptor [ ProcedureBlock ]
{
Parameter INDEX = "DBName";

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

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

/// Result Set
Property Rspec As %Library.ResultSet;

/// Initialize routine metrics.
Method Initialize() As %Status
{
s ..Rspec = ##class(%Library.ResultSet).%New("%SYSTEM.Database:FreeSpace")
d ..Rspec.Execute("*",0)
Quit $$$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.
{
// Get freespace data
Set stat = ..Rspec.Next(.sc)

// Quit if we have done all the datasets
If 'stat Q $$$Error($$$GeneralError,"End of Sample")

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

// quit with return value indicating the sample data is ready
Q $$$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、チャンネルをオープンするなど、1 回限りの初期化処理を実行することができます。

```

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

/// Open a tcp/ip device to send warnings
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 Freespace が作成されます。

```

/// Persistent sample class for MyMetric.Freespace
Class MyMetric.Sample.Freespace Extends %Monitor.Sample [ClassType=persistent]
{

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.Integer(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、このクラスを継承して、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。

2.9 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うクラスです。Metrics Class の Initialize() メソッドと GetSample() メソッドは、このクラスによって呼び出されます。

%Monitor.SampleAgent.%New(n) コンストラクタは、1 つの引数 (実行する Metrics Class の名前) を取ります。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() の呼び出し時には、Metrics Class のサンプル・クラスを生成します。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 d sampler.Collect() h 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. */

```

2.10 メトリック・データの表示

メトリックを表示するための最も簡単な方法は、Web ブラウザを使用することです。前述したように、Metrics Class はすべて CSP 対応です。したがって、どのサンプルも、ブラウザを使用して表示することができます。この例では、CSP URL は以下のようになります。

```
http://localhost:1972/csp/user/MyMetrics.Sample.Freespace.cls
```

The output looks like:

```
Monitor - Freespace c:\cache51\  
Name of dataset: c:\cache51\  
Current amount of Freespace: 8  
Monitor - Freespace c:\cache51\mgr\  
Name of dataset: c:\cache51\mgr\  
Current amount of Freespace: 10  
Monitor - Freespace c:\cache51\mgr\cacheaudit\  
Name of dataset: c:\cache51\mgr\user\  
Current amount of Freespace: 5
```

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。

または、%Monitor.Sample の Display(<metrics class>) クラスメソッドを使用して、汎用 CHUI ビューアを呼び出すこともできます。

```
%SYS>s mclass="Monitor.Test.Freespace"  
  
%SYS>s col=##class(%Monitor.SampleAgent).%New(mclass)  
  
%SYS>w col.Collect()  
1  
%SYS>w ##class(%Monitor.Sample).Display(mclass)  
Monitor - Freespace c:\cache51\  
Name of dataset: c:\cache51\  
Current amount of Freespace: 8.2  
Monitor - Freespace c:\cache51\mgr\  
Name of dataset:c:\cache51\mgr\  
Current amount of Freespace: 6.4
```

2.11 Caché モニタのエラー

以下のシステム・エラーが発生した場合は、システム・モニタによって無条件に通知されます。

1. 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。
2. データベース % が <FILEFULL> です。
3. 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。
4. 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。

5. sfm # 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。
6. sfm # 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。
7. WD 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。
8. WD:CP はすでに終了しています。システムをフリーズします。
9. 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。
10. グローバル・バッファが不足しています。WD がパニック・モードになっています。
11. WD 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。
12. 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)。
13. 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)。
14. 予期しない書き込みエラー:%d (% のブロック #%d)。
15. 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。
16. トランザクションがオープンしているか、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。
17. 重大なジャーナル・エラー:JRNSTOP が % をオープンできません。*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。
18. 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。
19. ジャーナル・ファイルが最大サイズ(%u バイト)に達したため、自動的にロールオーバーされました。
20. ジャーナル・ファイルへの書き込みに失敗しました。
21. 最新のジャーナル・ファイルを開けません。
22. ジャーナル・ファイルの同期に失敗しました。
23. %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・データが失われるのを防ぐため、エラーの原因を解消するか(SYSLOG を参照)、ジャーナル処理を別のデバイスに切り替えてください。
24. ジャーナルのログにエラーが発生しました。
25. 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。
26. ECP クライアント・デーモン/接続が中断されました。
27. 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。
28. enqpijstop に失敗しました。クラスタがクラッシュします。
29. enqpijchange に失敗しました。クラスタがクラッシュします。

30. 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。
31. 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。
32. 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。
33. 書き込みブロックエラー - 書き込みエラーをリカバリします。
34. 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、cforce 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。
35. CP:デーモン終了のモニタを作成できません。
36. CP:WD による %d の受け渡しに %d 秒におよんでいるため、システムをフリーズしました。WD が受け渡しを完了した時点でシステムを再開します。
37. WD:CP のハンドルをオープンする前に 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。
38. WD: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。
39. WD: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。
40. 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。
41. 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。
42. 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。
43. cforce によってすべてのプロセスを終了できませんでした。
44. mlock プロセスの開始に失敗しました (OpenVMS)。
45. スレーブ書き込みデーモンを開始できませんでした。
46. 理由 # により ENQDMN を終了します。
47. デーモン %c が、すでにほかのプロセス (OpenVMS) が所有している有効なロックを検出しました。
48. デーモン %c が有効なロックを取得できませんでした (OpenVMS)。

3

^GLOSTAT を使用したグローバル動作の統計収集

Caché 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。この章では、このルーチンの使用方法を説明します。内容は以下の通りです。

- ・ [^GLOSTAT の実行](#)
- ・ [^GLOSTAT 統計の概要](#)
- ・ [^GLOSTAT の出力例](#)

インターシステムズのシステム管理ポータルを使用して、^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。監視するシステムのポータル・アプリケーションにログオンし、[[ホーム]→[システム使用]] ページに移動します。

3.1 ^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。

1. 以下のコマンドを入力します。

```
Do ^GLOSTAT
```

2. 以下のプロンプトが表示されます。

```
Should detailed statistics be displayed for each block type? No =>
```

上記で示されているように、既定は “No” です。ブロック全体の要約を表示するには Enter キーを押し (例 A)、ブロック・タイプによる詳細な統計を表示するには “Yes” とタイプして Enter キーを押します (例 B)。

3. ^GLOSTAT ルーチンは、ユーザの要求に従って統計を表示します。Cache 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。したがって、最初の実行時の出力には、Cache を起動した後の操作が反映されます。

^GLOSTAT の統計レポートを最初に表示した後、または 2 回目以降に表示した後、以下のプロンプトが表示されます。

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

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。
z	^GLOSTAT 統計カウンタをゼロに初期化します。
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 C と 例 G)。

4. 特定の時間間隔についての統計データを表示するには (例 D)、以下の手順に従います。
 - a. “z” を入力して、統計データをゼロに初期化します。
 - b. “q” を入力して、このユーティリティを終了します。または、プロンプトに何も入力せず、そのままにしておきます。
 - c. 希望する時間だけ待機します。
 - d. ^GLOSTAT を終了した場合は、もう一度実行します。それ以外の場合は、“c” と入力して続行します。
 - e. ^GLOSTAT のレポートに、カウンタを初期化した後の操作を反映した統計データが表示されます。

注釈: ^GLOSTAT ユーティリティでは、通常、32 ビットのカウンタを使用して統計データが収集されます。時間間隔が極端に長いと、一部のカウンタでオーバーフローが発生する可能性があります。

3.2 ^GLOSTAT 統計の概要

^GLOSTAT 統計情報は、Cache を起動した後、またはカウンタを初期化した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ことが重要です。

Cache インスタンスがスタンドアロン構成または ECP 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 クライアントの場合（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 E）。

以下のテーブルは、^GLOSTAT 統計を定義します。

^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。
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・メモリにすでに格納されており、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)。
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。
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。
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)。
物理ブロック読取	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 (2 KB または 8 KB) の数。
物理ブロック書込	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 (2 KB または 8 KB) の数。
書込待機ブロック	ディスクへの書き込みを待機しているデータベース・ブロック (2-KB または 8-KB) の数。

オプションが選択されている場合、^GLOSTAT は、“論理ブロック要求”、“物理ブロック読込”、“書込待機ブロック” について、次のような詳細なブロック統計もレポートします。各カテゴリ内のすべてのブロック・タイプの総数は、右端の列に表示されます (例 F)。

ブロック・タイプは、以下の通りです。

ラベル	説明
Data	データ・ブロック
Dir	ディレクトリ・ブロック
Bdata	大きいデータ・ブロック (大きな文字列を含むブロック)
Map	マップ・ブロック
Upper ptr	上部ポインタ・ブロック
Bottom ptr	下部ポインタ・ブロック
Other	上記以外のブロック (増分バックアップやストレージ割り当て情報など)

3.3 ^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** – スタンドアロンまたはサーバ構成の Caché インスタンスで、^GLOSTAT ルーチンを最初に行った場合の出力例です。詳細な統計データは要求していません。
- ・ **例 B** – スタンドアロンまたはサーバ構成の Caché インスタンスで、詳細なブロック統計データを要求して、^GLOSTAT ルーチンを最初に行った場合の出力例です。
- ・ **例 C** – 対象となる時間間隔を指定して実行した場合の出力例。詳細な統計データは要求していません。
- ・ **例 D** – ^GLOSTAT ルーチンを複数回実行し、プロンプトに対して “c”、“z”、“q” で応答した場合の出力例を示します。詳細な統計データは要求していません。
- ・ **例 E** – クライアント構成の Caché インスタンスで、詳細な統計データを要求せずに、^GLOSTAT ルーチンを最初に行った場合の出力例です。
- ・ **例 F** – クライアント構成の Caché インスタンスで、詳細なブロック統計データを要求して、^GLOSTAT ルーチンを最初に行った場合の出力例です。
- ・ **例 G** – 詳細なブロック統計データを要求し、^GLOSTAT ルーチンを 2 回続けて実行します。時間間隔も指定しています。

3.3.1 例 A

以下は、^GLOSTAT ルーチンを最初に行った場合の出力例です。詳細な統計データは要求していません。Caché インスタンスはスタンドアロンまたはサーバ構成です。

^GLOSTAT を使用したグローバル動作の統計収集

```
%SYS>Do ^GLOSTAT
```

```
Should detailed statistics be displayed for each block type? No =>n
```

Statistics	Total
Global references (all):	6,445,330
Global update references:	1,322,207
Routine calls:	896,625
Routine buffer loads and saves:	9,008
Logical block requests:	2,997,805
Block reads:	34,674
Block writes:	1,918
Cache Efficiency:	176
Journal Entries:	211,164
Journal Block Writes:	363

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.2 例 B

以下は、^GLOSTAT ルーチンを最初に実行した場合の出力例です。詳細なブロック統計データを要求しています。Cache インスタンスはスタンドアロンまたはサーバ構成です。

```
%SYS>do ^GLOSTAT
```

```
Should detailed statistics be displayed for each block type? No =>y
```

Statistics	Total
Global references (all):	6,444,341
Global update references:	1,322,017
Routine calls:	896,130
Routine buffer loads and saves:	9,008
Cache Efficiency:	176
Journal Entries:	211,164
Journal Block Writes:	363
Logical Block Requests	
Dir: 13,147	Data: 1,643,739
BData: 164,102	Upper ptr: 138,837
Map: 3,043	Bottom ptr: 1,034,472
	Other: 1
Physical Block Reads	2,997,341
Dir: 42	Data: 33,846
BData: 452	Upper ptr: 31
Map: 12	Bottom ptr: 290
	Other: 1
Physical Block Writes	34,674
Blocks Queued to be Written	1,918
Dir: 11	Data: 1,952
BData: 317	Upper ptr: 0
Map: 22	Bottom ptr: 65
	Other: 6

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.3 例 C

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。各ブロック・タイプの詳細な統計データは要求していません。Cache インスタンスは、スタンドアロン構成またはサーバ構成のいずれかです。

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit? 30
```

```
Counts per Second for 30 Seconds...
```

Statistics (per second)	Total
Global references (all):	1,369.8
Global update references:	299.6
Routine calls:	1,057.6
Routine buffer loads and saves:	12.0
Logical block requests:	696.9
Block reads:	0.7
Block writes:	3.2
Cache Efficiency:	354.3
Journal Entries:	277.5
Journal Block Writes:	0.5

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.4 例 D

以下の例は、^GLOSTAT ルーチンを複数回実行した場合の出力結果です。

最初の出力の後、“z”を入力してカウンタを初期化し、“q”を入力してルーチンを終了します。一定の時間待機した後、再度ルーチンを実行し、“z”入力後の処理が反映された統計データを表示します。

再び“z”を入力して、統計データを再度初期化し、その後すぐに“c”を入力して、初期化された統計データを表示します。“c”をもう1度入力すると、累積統計データが再表示されます。

^GLOSTAT を使用したグローバル動作の統計収集

%SYS>Do ^GLOSTAT

Should detailed statistics be displayed for each block type? No =>

Statistics	Total
Global references (all):	85,414
Global update references:	23,073
Routine calls:	67,092
Routine buffer loads and saves:	477
Logical block requests:	64,587
Block reads:	140
Block writes:	49
Cache Efficiency:	452
Journal Entries:	25,341
Journal Block Writes:	44

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? z

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? q

%SYS>Do ^GLOSTAT

Should detailed statistics be displayed for each block type? No =>

Statistics	Total
Global references (all):	4,332
Global update references:	568
Routine calls:	3,487
Routine buffer loads and saves:	95
Logical block requests:	1,423
Block reads:	0
Block writes:	130
Cache Efficiency:	33
Journal Entries:	218
Journal Block Writes:	1

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? z

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? c

Statistics	Total
Global references (all):	0
Global update references:	0
Routine calls:	0
Routine buffer loads and saves:	0
Logical block requests:	0
Block reads:	0
Block writes:	0
Cache Efficiency:	no i/o
Journal Entries:	0
Journal Block Writes:	0

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? c

Statistics	Total
Global references (all):	9,362
Global update references:	1,742
Routine calls:	7,382
Routine buffer loads and saves:	96
Logical block requests:	4,404
Block reads:	4
Block writes:	0
Cache Efficiency:	2,341
Journal Entries:	1,548

```
Journal Block Writes: 3
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.5 例 E

以下は、^GLOSTAT ルーチンを最初に実行した場合の出力例です。詳細な統計データは要求していません。Cache インスタンスはクライアントです。

```
%SYS>DO ^GLOSTAT

Should detailed statistics be displayed for each block type? No =>n

Statistics                Local          Remote          Total
-----
Global references (all):  1,558,696      0                1,558,696
Global update references:  531,676        0                531,676
Routine calls:           95,987         0                95,987
Routine buffer loads and saves:  747           0                747
Logical block requests:  1,121,187      n/a              1,121,187
Block reads:             3,155          0                3,155
Block writes:            1,450          n/a              1,450
Cache Efficiency:        338            no gets
Journal Entries:         525,177        n/a              525,177
Journal Block Writes:    12,546         n/a              12,546

Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.6 例 F

以下は、^GLOSTAT ルーチンを最初に実行した場合の出力例です。詳細なブロック統計データを要求しています。Cache インスタンスはクライアントです。

^GLOSTAT を使用したグローバル動作の統計収集

```
%SYS>DO ^GLOSTAT
```

```
Should detailed statistics be displayed for each block type? No =>y
```

Statistics	Local	Remote	Total
Global references (all):	1,558,611	0	1,558,611
Global update references:	531,676	0	531,676
Routine calls:	95,979	0	95,979
Routine buffer loads and saves:	747	0	747
Cache Efficiency:	338	no gets	
Journal Entries:	525,177	n/a	525,177
Journal Block Writes:	12,546	n/a	12,546
Logical Block Requests	Data:	608,909	
Dir:	792	Upper ptr:	2,475
BData:	81,750	Bottom ptr:	426,330
Map:	879	Other:	1
Physical Block Reads	Data:	2,823	1,121,136
Dir:	11	Upper ptr:	9
BData:	240	Bottom ptr:	66
Map:	5	Other:	1
Physical Block Writes			3,155
Blocks Queued to be Written	Data:	959	1,450
Dir:	10	Upper ptr:	0
BData:	430	Bottom ptr:	38
Map:	17	Other:	7

```
Continue (c), zero statistics (z), timed stats (# sec > 0), quit?
```

3.3.7 例 G

以下の例は、各ブロック・タイプの詳細な統計データを要求して、^GLOSTAT ルーチンを最初に実行した後、指定した時間間隔 (30 秒) における 1 秒あたりの統計データを示しています。Cache インスタンスは、スタンドアロン構成またはサーバ構成のいずれかです。

%SYS>Do ^GLOSTAT

Should detailed statistics be displayed for each block type? No =>y

Statistics				Total
-----				-----
Global references (all):				50,818
Global update references:				8,920
Routine calls:				25,279
Routine buffer loads and saves:				1,027
Cache Efficiency:				100
Journal Entries:				236
Journal Block Writes:				4
Logical Block Requests		Data:	16,375	
Dir:	1,760	Upper ptr:	263	
BData:	165	Bottom ptr:	2,538	
Map:	83	Other:	1	21,185
Physical Block Reads		Data:	153	
Dir:	6	Upper ptr:	5	
BData:	165	Bottom ptr:	38	
Map:	4	Other:	1	372
Physical Block Writes				134
Blocks Queued to be Written		Data:	104	
Dir:	6	Upper ptr:	0	
BData:	0	Bottom ptr:	29	
Map:	9	Other:	5	153

Continue (c), zero statistics (z), timed stats (# sec > 0), quit? 30

Counts per Second for 30 Seconds...

Statistics (per second)				Total
-----				-----
Global references (all):				4,232.7
Global update references:				1,453.4
Routine calls:				446.8
Routine buffer loads and saves:				14.1
Cache Efficiency:				507.9
Journal Entries:				33.3
Journal Block Writes:				0.1
Logical Block Requests		Data:	909.8	
Dir:	15.6	Upper ptr:	21.3	
BData:	0.2	Bottom ptr:	589.3	
Map:	1.3	Other:	0	1,537.5
Physical Block Reads		Data:	6.8	
Dir:	0	Upper ptr:	0	
BData:	0.2	Bottom ptr:	0.1	
Map:	0	Other:	0	7.1
Physical Block Writes				1.2
Blocks Queued to be Written		Data:	2.6	
Dir:	0.0	Upper ptr:	0	
BData:	0	Bottom ptr:	0.2	
Map:	0.1	Other:	0	2.9

Continue (c), zero statistics (z), timed stats (# sec > 0), quit?

4

^PERFMON を使用したシステム・パフォーマンスの監視

Caché には ^PERFMON というユーティリティが用意されています。このユーティリティは、システム MONITOR の制御に必要な機能呼び出すための基本メニューと付帯機能を提供します。

注釈: MONITOR は、以前のバージョンの Caché で、^MONITOR ルーチンを使用してアクセスしていたシステム内部機能です。^PERFMON の機能は、以前のバージョンでは ^VPMON ルーチンを使用して実行していました。

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、さらにプロセス、グローバル、ルーチン、およびネットワーク・ノード別に並べ替えることによって、Caché 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ので、カウンタの収集を明確に有効化し、データを収集するプロセス、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、指定した数のプロセス、ルーチン、グローバル、ノード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。使用可能なすべてのスロットがプロセスに割り当てられると、それ以降のプロセスのカウンタは“その他”スロットに追加されます。グローバル、ルーチン、ノードについても同様です。

データの収集中でも、それらデータのレポートを表示できます。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 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。

MONITOR ユーティリティを制御するための関数は、%SYSTEM.Monitor パッケージのクラスにも用意されています。この場合、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することができます。詳細は、“Cache クラス・リファレンス”の、%SYSTEM.Monitor.Collection、%SYSTEM.Monitor.Sample、および %SYSTEM.Monitor.Counters クラスのドキュメントを参照してください。

以下の関数を使用できます。

Start

統計情報の収集を有効にします。

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

引数は以下の通りです。

- ・ process = 確保するプロセス・スロットの数
- ・ global = 確保するグローバル・スロットの数
- ・ routine = 確保するルーチン・スロットの数
- ・ network = 確保するネットワーク・ノード・スロットの数

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

```
process = the number of process slots to reserve  
global = the number of global slots to reserve  
routine = the number of routine slots to reserve  
network = the number of network node slots to reserve
```

```
status = 1 if successful or,  
"-1,Somebody else is using Monitor."  
"-2,Monitor is already running."  
"-3,Memory allocation failed"  
"-4,Couldn't enable stats collection"
```

```
status = $$Stop^PERFMON() stops collection  
status = 1 if successful or,  
"-1,Somebody else is using Monitor."  
"-2,Monitor is not running."
```

```
status = $$Pause^PERFMON() momentarily pauses collection  
          (allows a consistent state for viewing data)  
status = 1 if successful or,  
"-1,Somebody else is using Monitor."  
"-2,Monitor is not running."
```

```
status = $$Resume^PERFMON() resumes collection  
status = 1 if successful or,  
"-1,Somebody else is using Monitor."  
"-2,Monitor is not running."
```

```
status = $$Clear^PERFMON() clear all counters  
status = 1 if successful or,  
"-1,Somebody else is using Monitor."  
"-2,Monitor is not running."
```

4.1 PERFMON の実行

以下の手順に従って PERFMON を実行します。

1. DO ^PERFMON
2. 以下のメニューが表示されます。該当するオプションの番号を入力してください。ルーチンを終了するには **Enter** を押します。

```

1. Start Monitor
2. Stop Monitor
3. Pause Monitor
4. Resume Monitor
5. Clear Counters
6. Report Statistics

```

Enter the number of your choice:

最初の 4 つは、それぞれモニタを起動、終了、一時停止、再開するためのオプションです。

```

status = $$Start^PERFMON(process, global, routine, network)
          turns on collection of the statistics.

process = the number of process slots to reserve
global  = the number of global slots to reserve
routine = the number of routine slots to reserve
network = the number of network node slots to reserve

```

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	モニタがすで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

4.1.1 PERFMON レポート

```

;;The menu items available from ^PERFMON correspond directly to callable
;;extrinsic functions in ^PERFMON, and the input collected is used to
;;directly supply the parameters of these functions. Each function returns
;;a success/failure status, where success is "1" and failure is indicated by
;;a string with a negative number followed by a comma and a brief message.
;:*PAGE*
;;The following functions are available:
;;
;:;$Start^PERFMON(process, global, routine, network) turns on collection
;; of the statistics.
;;
;; - process = the number of process slots to reserve
;; - global = the number of global slots to reserve
;; - routine = the number of routine slots to reserve
;; - network = the number of network node slots to reserve
;;
;:;$Stop^PERFMON() stops collection
;;
;:;$Pause^PERFMON() pauses collection
;; (allows a consistent state for viewing data)
;;
;:;$Resume^PERFMON() resumes collection (after viewing data)
;;
;:;$Clear^PERFMON() clear all counters
;:*PAGE*
.;;
;:;$Report^PERFMON(report, sort, format, output, [list], [data])
;; gathers and outputs a report of the counters
;;
;; - report = type of report to output.
;; - "G" = Global Activity
;; - "R" = Routine Activity
;; - "N" = Network Activity
;; - "C" = Custom report (user selects columns)
;;
;; - sort = sort order
;; - "P" = organize report by Process
;; - "R" = organize report by Routine
;; - "G" = organize report by Global
;; - "I" = organize report by Incoming Node
;; - "O" = organize report by Outgoing Node
;;
;; - format = output format
;; - "P" = printable/viewable report
;; - "D" = comma delimited data

format = output format
"P" = printable/viewable report (no pagination)
"D" = comma delimited data (can be read into spreadsheet)
"A" = return data in local array
;;
;; - output = file name, or 0 for output to screen
;; - list = comma separated list of numbers to specify cols in Custom report
;:*END*

data = 1 for standard rates/second, or 2 for raw totals

status = 1 if successful or,
"-1, Monitor is not running"
"-2, Missing input parameter."
"-3, Invalid report category."
"-4, Invalid report organization."
"-5, Invalid report format."
"-6, Invalid list for Custom report."
"-7, Invalid data format (1=rates or 2=raw only)."
```

以下は、カスタム・レポート(タイプ “C”) で使用できるメトリックの一覧です。グローバル (“G”)、ルーチン (“R”)、およびネットワーク (“N”) の活動レポートには、この一覧内の決められたメトリックが表示されます。Report 関数の list パラメータには、この一覧の左側列に示されている番号をコンマで区切って指定する必要があります。

カスタム・レポートのフィールド

番号	列タイトル	説明
1	GloRef	グローバル参照
2	GloSet	グローバル Set
3	GloKill	グローバル Kill
4	TotBlkRd	物理ブロックの合計読み取り数 (次の 7 つのカウンタ)
5	DirBlkRd	ディレクトリ・ブロックの読み取り数
6	UpntBlkRd	上部ポインタ・ブロックの読み取り数
7	BpntBlkRd	下部ポインタ・ブロックの読み取り数
8	DataBlkRd	データ・ブロックの読み取り数
9	RouBlkRd	ルーチン・ブロックの読み取り数
10	MapBlkRd	マップ・ブロックの読み取り数
11	OthBlkRd	その他のブロックの読み取り数
12	DirBlkWt	ディレクトリ・ブロックの書き込み数
13	UpntBlkWt	上部ポインタ・ブロックの書き込み数
14	BpntBlkWt	下部ポインタ・ブロックの書き込み数
15	DataBlkWt	データ・ブロックへの書き込み
16	RouBlkWt	ルーチン・ブロックへの書き込み
17	MapBlkWt	マップ・ブロックへの書き込み
18	OthBlkWt	その他のブロックへの書き込み
19	DirBlkBuf	グローバル・バッファによって満たされたディレクトリ・ブロック要求
20	UpntBlkBuf	グローバル・バッファによって満たされた上部ポインタ・ブロック要求
21	BpntBlkBuf	グローバル・バッファによって満たされた下部ポインタ・ブロック要求

番号	列タイトル	説明
22	DataBlkBuf	グローバル・バッファによって満たされたデータ・ブロック要求
23	RouBlkBuf	グローバル・バッファによって満たされたルーチン・ブロック要求
24	MapBlkBuf	グローバル・バッファによって満たされたマップ・ブロック要求
25	OthBlkBuf	グローバル・バッファによって満たされたその他のブロック要求
26	JrnEntry	ジャーナル・エントリ
27	BlkAlloc	割り当てられたブロック
28	NetGloRef	ネットワーク・グローバル参照
29	NetGloSet	ネットワーク Set
30	NetGloKill	ネットワーク Kill
31	NetReqSent	送信されたネットワーク要求
32	NCacheHit	ネットワーク・キャッシュ・ヒット
33	NCacheMiss	ネットワーク・キャッシュ・ミス
34	NetLock	ネットワーク・ロック
35	RtnLine	M コマンド
36	RtnLoad	ルーチンのロード
37	RtnFetch	ルーチンの取得
38	LockCom	ロック・コマンド
39	LockSucc	成功したロック・コマンド
40	LockFail	失敗したロック・コマンド
41	TermRead	ターミナルの読み取り
42	TermWrite	ターミナルの書き込み
43	TermChRd	ターミナル読み取り文字数
44	TermChWrt	ターミナル書き込み文字数
45	SeqRead	シーケンシャル・リード
46	SeqWrt	シーケンシャル・ライト

番号	列タイトル	説明
47	IJCMsgRd	読み込まれたローカル IJC メッセージ
48	IJCMsgWt	書き込まれたローカル IJC メッセージ
49	IJCNetMsg	書き込まれたネットワーク IJC メッセージ
50	Retransmit	ネットワーク再送
51	BuffSent	送信されたネットワーク・バッファ

5

^%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、Caché MONITOR ユーティリティを実行するためのユーザ・インタフェースです。この機能を使用すれば、Caché ObjectScript ルーチンおよび Caché Basic 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、Caché 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。

このドキュメントには、以下のセクションが含まれます。

- ・ [監視の開始](#)
- ・ [監視オプション](#)
- ・ [サンプル出力](#)
- ・ [プログラミング・インタフェース](#)

5.1 監視の開始

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
```

1.) Start Monitor

```
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 major
impact on the performance of a system, and it is recommended
that you do this on a 'test' system.
```

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

「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。

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。以下の順序で、必要な監視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。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。

```
Enter routine names to monitor on a line by line basis. Patterns are allowed.
Enter blank line to terminate input
Routine Name:
```

2. Select Metrics to monitor - 該当する番号を入力します。既定のメトリックを監視する場合は 1 を選択します (既定値)。使用可能なすべてのメトリックを監視する場合は 2 を選択します。独自のメトリック・リストを使用する場合は 3 を選択します。

```
Select Metrics to monitor
1.) Monitor Default Metrics
2.) Monitor All Metrics
3.) Customize Monitor Metrics
```

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

収集される既定のメトリックは以下の通りです。

- ・ RtnLine - 対象行が実行される回数
- ・ Time - その行の実行に要したクロック時間
- ・ 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

メトリック・リストをカスタマイズする場合は、**%Monitor.System** クラスがサポートしている標準のパフォーマンス・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> 3

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

Metric#: ?
1.) GloRef: global refs
2.) GloSet: global sets
.
.
.
34.) RtnLine: lines of Cache Object Script
.
.
.
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、“プログラミング・インタフェース”を参照してください。

3. **Select Processes to monitor** - 該当する番号を入力します。すべてのプロセスを監視する場合は 1 を選択します。現在のプロセスを監視する場合は 2 を選択します。プロセス ID 番号 (PID) でプロセス・リストを指定する場合は 3 を選択します。

```

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>

```

この時点では、`^%SYS.MONLBL` によってプロセスのリストも、PID を選択する方法も示されません。ただし、`^%SS` ユーティリティを使用するか、システム管理ポータルの [[ホーム]→[プロセス]] ページに移動すれば、プロセス ID 番号を確認することができます。

```

Enter the number of your choice: <1> 3

Enter PID (blank to terminate)

PID: 1640
PID: 2452
PID:

```

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。

`^%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。

```

Monitor started.

Press RETURN to continue ...

```

5.2 監視オプション

以下のメニュー・オプションが表示されます。

- ・ 2.) Stop Monitor – 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。
- ・ 3.) Pause Monitor – 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、^%SYS.MONLBL を実行している場合のみ表示されます。
- ・ 4.) Resume Monitor – 一時停止していた収集を再開します。このオプションは、^%SYS.MONLBL を一時停止している場合のみ表示されます。
- ・ 5.) Clear Counters – 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。
- ・ 6.) Report Statistics – 収集したデータのレポートが表示されます。監視対象のすべてのルーチンが提示され、いずれかのルーチンを選択するよう求められます。**Enter** を押すと、監視したすべてのルーチンが表示されます。最後のルーチンを入力したら、**Enter** をもう一度押してリストを終了します。出力するファイルの名前を入力するか、何も入力せずに **Enter** を押すと、端末上にレポートが表示されます。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。

5.3 サンプル出力

以下は、^%SYS.MONLBL ルーチン自体を監視した場合の出力例です。

```
Routine ^%SYS.MONLBL ...
```

```

Line   RtnLine      Time  TotalTime
.
.
79      0      0      0      read !,"FileName: ", filename s:...
80      0      0      0      open filename:"NW":1 if '$t set ...
81      0      0      0      use filename
82      1      0.000010  0.000010  ; Write Rtn Data
83      1      0.000005  0.000005  if (rtnnum > 0) {
84      0      0      0      do writertndata(rtnnum)
85      0      0      0      write !!
86      0      0      0      }
87      0      0      0      else {
88      1      0.000011  0.000011  for rtnnum=1:1:($zu(84,16)) {
89      163    0.001803  0.852597  do writertndata(rtnnum)
90      162    0.004931  0.004931  write !!
91      162    0.000428  0.000428  }
92      0      0      0      }
93      0      0      0      goto writertndataend
94      163    0.000337  0.000337  writertndata(rtnnum)
95      163    0.000333  0.000333  ; Write Data For a Single Rtn
96      163    0.000512  0.000512  set rtnname = $zu(84,16,2,rtnnum)
97      163    0.000334  0.000334  s sp=11
98      163    0.030024  0.030024  w !!,"Routine ^",rtnname," ..."
99      163    0.000601  0.000601  set lns = $zu(84, 16, 1, rtnnum)
100     163    0.003862  0.003862  if lns=0 w " no data yet." quit
101      7      0.000015  0.000015  ; Write Column Headers
102      7      0.000309  0.000309  write !!,"Line " s col=5
103      7      0.000039  0.000039  for metric=0:1:($zu(84,13)-1) {
104      21    0.000059  0.000059  s column = $zu(84,13,11,metric) + 1
105      21    0.000146  0.000146  s out=$piece($text(@("Flist+"_column)...
106      21    0.000571  0.000571  w ?col,$j(out,sp-1)
107      21    0.000060  0.000060  s col=col+sp
108      21    0.000049  0.000049  }
109      7      0.000019  0.000019  for line=0:1:(lns-1) {
110     2958   0.103596  0.103596  write !,(line+1)
111     2959   0.007729  0.007729  s col=5
112     2960   0.009972  0.009972  for metric=0:1:($zu(84,13)-1) {
113     8880   0.029456  0.029460  s out=$zu(84,16,3,line,metric)
114     8883   0.019621  0.019623  ; Convert clock/CPU time to seconds
115     8886   0.021176  0.021178  s n=$zu(84,13,11,metric)
116     8889   0.027933  0.027938  i (n=50)!(n=51) s out=$select(out=0...
117     8892   0.306696  0.306726  write ?col,$j(out,sp-1)
118     8895   0.024750  0.024753  s col=col+sp
119     8898   0.020944  0.020947  }
120     2967   0.169251  0.169251  write ?col,$TEXT(@("+" _ (line+1) _ ...
121     2968   0.015208  0.015208  }
122      6      0.000034  0.000034  quit

```

反復コードの合計時間

ルーチンに反復コードが含まれている場合、ルーチン全体の合計時間と比べて、同じサブルーチンを繰り返し呼び出す行の TotalTime カウンタが極端に大きく見える場合があります。これは、同じコードが繰り返し実行され、その累積が特定行の TotalTime になるためです。つまり、TotalTime には、サブルーチンを n 回実行するための合計所要時間、n-1 回実行するための合計所要時間、n-2 回実行するための合計所要時間 (以下同様) が含まれます。技術的には間違いはありませんが、混乱を招く可能性があります。このルーチンを修正し、一番外側のループが TotalTime に反映されるようにすれば、より実用的な数値が表示されるようになります。

5.4 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、Cache 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、各行のカウントは \$LIST 形式で返されます。

`%Monitor.System.LineByLine` の詳細は、“Cache オンライン・クラス・リファレンス”を参照してください。

A

BMC PATROL を使用した Caché の監視

この付録では、Caché と BMC PATROL 間のインタフェースについて説明します。

BMC PATROL は、さまざまなソフトウェア・システムを監視および管理するためのツールです。Caché には、Caché を監視し、その情報を収集する PATROL 拡張機能が備わっています。

このインタフェースを使用すれば、PATROL コンソールから、1 つまたは複数の Caché システムのメトリックを監視できます。そのためには、Caché システム上で PATROL デーモンを実行し、メトリック値を収集および出力する必要があります。さらに、Caché ナレッジ・モジュール・ファイル (*.km) を PATROL コンソールにロードし、それらの値を読み込んで、表示する必要があります。

この付録で説明する内容は以下の通りです。

- ・ [Caché での PATROL の実行](#)
- ・ [Caché PATROL ナレッジ・モジュール](#)
- ・ [BMC PATROL で使用される Caché メトリック](#)

A.1 Caché での PATROL の実行

監視する各 Caché インスタンス上で、`^PATROL Caché ObjectScript` ルーチンを実行します。その場合、Caché に付属のシステムクラス・メソッドを使用する方法と、システムを起動した時点で自動的に実行されるように設定する方法があります。

このルーチンを実行すると、Caché 管理者のディレクトリ (既定では `c:¥CacheSys¥Mgr`) にある `patrol.dat` ファイルにメトリックを出力するバックグラウンド・プロセスが開始されます。このファイルは

収集期間ごとに書き換えられるので、ファイル・サイズは変化しません。また、**patrol.dat** ファイルには識別ヘッダとタイム・スタンプも記録されるので、そのファイルが現在アクティブかどうか、最新の情報が収集されているかどうかを PATROL コンソールで確認できます。

Caché で PATROL を実行する方法は 2 つあります。

- ・ [Caché PATROL ルーチン](#)
- ・ [PATROL の自動起動](#)

A.1.1 Caché PATROL ルーチン

Caché には、BMC PATROL を起動および停止するための `PATROL` ルーチンに対するエントリー・ポイントが備わっています。

PATROL を起動する方法は、以下の通りです。

```
Do start^PATROL(display,process,timer)
```

このルーチンの引数は下表の通りです。

引数	説明	既定値	システム管理ポータルの詳細設定
display	表示モード。出力する数値の種類 (total、delta、または rate) を指定します。	total	[Patrol 表示モード]
process	%SS 統計データを渡すプロセスの数。	20	[表示する Patrol プロセスの数]
timer	収集期間 (単位は秒)。	30	[Patrol の収集期間]

以下に例を示します。

```
Do start^PATROL("total",20,30)
```

上記の例では、収集開始後、上位 20 のプロセスの合計カウントを表示するよう PATROL コンソールを設定しています。Caché は 30 秒ごとに情報を送信します。収集期間の引数も PATROL コンソールに渡されるので、収集と表示の更新を同期することができます。

PATROL を停止する方法は、以下の通りです。

```
Do stop^PATROL
```

A.1.2 PATROL の自動起動

システム管理ポータルでは、Caché の起動時に、PATROL デーモンを自動的に起動するためのオプションを設定できます。

1. システム管理ポータル・アプリケーションにログインします。
2. [[ホーム]→[構成]→[詳細設定]] ページへ移動し、[カテゴリ] ボックスで [Monitoring] をクリックして、リストに表示される項目を制限します。
3. [PatrolStart] (システム起動時に Patrol を起動) 行で [編集] をクリックし、[構成設定] ページを表示します。
4. [値] リスト・ボックスで [はい] をクリックします。Caché の起動時に Patrol も起動します。

前のセクションで説明した PATROL ルーチンの引数を設定することもできます。各設定の横にある [編集] をクリックし、必要に応じて以下の値を変更します。

- ・ [Patrol の収集期間]
- ・ [Patrol 表示モード]
- ・ [表示する Patrol プロセスの数]

A.2 Caché PATROL ナレッジ・モジュール

PATROL のアーキテクチャは、ナレッジ・モジュールの概念を基にしています。ナレッジ・モジュールには、一連のコマンド、監視するパラメータ、および PATROL で使用するアクションが含まれています。PATROL 用の Caché プラグ・インは複数のナレッジ・モジュールで構成されています。ユーザは、これらのモジュールを PATROL コンソールにロードします。

- ・ ISC_CACHE.km
- ・ ISC_CACHE_CONFIG.km
- ・ ISC_CACHE_DISK.km
- ・ ISC_CACHE_GLOBAL.km
- ・ ISC_CACHE_NETWORK.km
- ・ ISC_CACHE_OTHER.km
- ・ ISC_CACHE_OVERVIEW.km
- ・ ISC_CACHE_ROUTINE.km

これらの KM を PATROL コンソールにロードすると、接続されているすべてのシステムで Caché インスタンスが自動的に検出されます。その際、NT システムの場合はレジストリが検索され、UNIX または OpenVMS システムの場合は `ccontrol list` コマンドの出力が解析されます。各 Caché インスタンスについて、`patrol.dat` ファイルが Caché 管理者のディレクトリに存在するか、またそのファイルのタイムスタンプが現在のものであるかが確認されます。現在、Caché メトリックを PATROL に報告している Caché インスタンスが PATROL コンソールに表示されます。

A.2.1 Caché モジュールを PATROL に追加

Caché ナレッジ・モジュールをコンソールに追加して、有効にするには、以下の手順に従います。

1. [PATROL Console] の [File] メニューで、[Load KM] をクリックします。
2. Caché Patrol ディレクトリ (既定では `c:\CacheSys\Patrol`) にある、すべての `*.km` ファイルを選択します。
3. `ISC_CACHE` モジュールが、[Console] の [Desktop] タブに表示されます。
4. [ISC_CACHE] を右クリックし、[KM Commands] メニューから [Add Configuration] を選択します。
5. [Add Configuration] ダイアログ・ボックスで構成名を入力し、インストール・ディレクトリとして Caché ディレクトリの `C:/CacheSys` を指定します。
6. PATROL が Caché 統計情報を認識するには、30 秒 (PATROL の既定の同期期間) 程度の時間がかかります。

詳細は、“BMC の Web サイト” を参照してください。

システム上で Caché インストールが検出された場合、Caché のメイン・エントリ (Caché クラス) がシステム・エントリの下に表示されます。Caché クラスの下には、各 Caché インスタンス (そのシステムにインストールされている各 Caché 構成) が表示されます。さらに、各 Caché インスタンスの下には、標準メトリック・カテゴリ (Overview、Global、Routines、Disk Activity、Network、Other) が配置されます。

以下に例を示します。

```
- PATROLMainMap
- TEST1
  - ISC_CACHE
    - ISC_Config_CACHE
      + ISC_DiskActivity
      + ISC_Global
      + ISC_Network
      + ISC_Other
      + ISC_Overview
      + ISC_Routine
```

これらのカテゴリを展開すると、そのカテゴリに属する個々のメトリックが表示されます。Overview の下に表示されるメトリックは、現在のレベルを示す測定値です。その他は、値の時間的推移を示すグラフです。

[Cache configuration] を右クリックすると Caché 専用のコマンドが表示されます。[Remove Configuration] を選択して構成を削除するか、[Process Status] を選択してそのためのウインドウを表示することができます。

Caché のインストールはすべて自動的に検出されるので、通常は、構成を手動で追加する必要はありません。ただし、特定のインストールで不明点や問題が生じた場合は、手動で行うことも可能です。

Caché KM からのエラー・メッセージは、[System Output] ウィンドウに出力されます。Caché インスタンスが自動的に検出されない場合は、これらのメッセージを参照して、問題がないかどうかを確認してください。

A.3 BMC PATROL で使用される Caché メトリック

以下は、Caché のメトリックの一覧です。

Caché PATROL メトリック

カテゴリ	メトリック
Overview	Global Refs (gauge)
	Global Sets, Reads, Kills (graph)
	Net Global Refs (gauge)
	Net Global Sets, Reads, Kills (graph)
	Routine Lines (gauge)
	Routine Loads (gauge)
	Locks (gauge)
	Process Count (graph)
	Cache Efficiency (graph) (= $100 * (\text{LogicalReads} / (\text{LogicalReads} + \text{Physical Reads}))$)
	Licenses Used (gauge)

カテゴリ	メトリック
Global	Global Refs
	Global Sets
	Global Kills
	Global Reads
	Blocks Allocated
	Locks
	Successful Locks
	Failed Locks
	Job InGlobal
	WD QueSize
	Global AvailBufs
	Que Gaccess
	Que GaccUpd
	Que GBFAny
	Que GBFSpec
	Journal Entries
	Jrn FileSize
	Jrn EndOffset
	Tot Global Bufs
	GThrottle Cur
	GThrottle Max
GThrottle Cnt	
Routine	Routine Lines
	Routine Loads
	Routine Fetches

カテゴリ	メトリック
Disk Activity	Physical Directory Reads
	Physical U-Ptr Reads
	Physical B-Ptr Reads
	Physical Data Reads
	Physical Routine Reads
	Physical Map Reads
	Physical Other Reads
	Physical Directory Writes
	Physical U-Ptr Writes
	Physical B-Ptr Writes
	Physical Data Writes
	Physical Routine Writes
	Physical Map Writes
	Physical Other Writes
	Logical Directory Reads
	Logical U-Ptr Reads
	Logical B-Ptr Reads
	Logical Data Reads
	Logical Routine Reads
	Logical Map Reads
	Logical Other Reads

カテゴリ	メトリック
Network	Net Global Refs
	Net Global Sets
	Net Global Kills
	Net Global Reads
	Net Requests Sent
	Net Cache Hits
	Net Cache Misses
	Net Locks
	Net Retransmits
	Net Buffer
	Net GblJobs
Other	Terminal Reads
	Terminal Writes
	Terminal Read Char
	Terminal Write Char
	Sequential Read
	Sequential Write

B

SNMP を使用した Caché の監視

このドキュメントでは、Caché 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開発され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、Caché にとって非常に重要な利点です。つまり、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として、SNMPを利用することができます。

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。Caché 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。

ここで説明する内容は以下の通りです。

- ・ [Caché での SNMP の実行](#)
- ・ [Caché での SNMP の管理](#)
- ・ [サブエージェントとしての Caché](#)
- ・ [Caché MIB 構造](#)

B.1 Caché での SNMP の実行

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。Caché では **ISC-CACHE.mib** ファイルがこれに相当します。このファイルは、Caché インストール・ディレクトリの **SNMP** サブディレクトリに配置されています。

B.2 Caché での SNMP の管理

SNMP は標準プロトコルなので、Caché サブエージェントの管理は最小限ですみます。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 プロトコル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 Caché 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。

Caché を起動したとき、Caché 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. システム管理ポータル の [[ホーム]→[セキュリティ管理]→[サービス]] ページに移動します。
2. **%System_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、**%System_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。

SNMP ルーチンを使用して、Caché SNMP サブエージェントを開始および停止することもできます。

```
Do start^SNMP(<port>,<timeout>)
Do stop^SNMP
w $$start^SNMP(port,timeout)
w !,"          port = TCP port for connection (default is 705)"
w !,"          timeout = TCP port read timeout (default is "_20_" seconds)"
w !!,"      w $$stop^SNMP()
```

接続時または要求の応答時に問題が発生した場合は、Caché 管理者のディレクトリにある **SNMP.LOG** ファイルに記録されます。

B.3 サブエージェントとしての Caché

SNMP クライアントは、既知のアドレス (UDP ポート 161) で受信待機している 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 Caché SNMP インタフェースを実装しています。

Caché 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。Caché 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。OpenVMS と Tru64 UNIX など、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・デーモン NET-SNMP エージェントを使用できます。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。これに対処するため、Caché 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 Caché AgentX サーバ間の接続を制御します。

B.4 Caché MIB 構造

Caché 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、Caché MIB ファイル `ISC-CACHE.mib` で定義されています。既定では、このファイルは、Caché 管理者ディレクトリの `SNMP` サブ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。そのための手順はアプリケーションによって異なります。Caché 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。

Caché 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、ここではあらためて説明しません。Caché 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。

SNMP では、すべての管理対象オブジェクトを網羅する階層ツリー構造が定義されています。これを SMI (Structure of Management Information) といい、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拡張を定義する場合は、Caché と同様、以下のように “enterprise” ブランチを使用します。

```
iso.org.dod.internet.private.enterprises (1.3.6.1.4.1)
```

IANA (Internet Assigned Numbers Authority) 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ます。Caché の場合、この番号は intersystems を表す 16563 です。

この下に、Caché 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 の下のレベルは製品 ID です。Caché の場合、この番号は .1 (iscCache) になります。この番号は MIB モジュール ID として利用され、アプリケーション ID としても知られています。
- ・ その次のレベルでは、データ・オブジェクトと通知が別々に処理されます。これらの番号は、.1 (cacheObjects) と .2 (cacheTraps) 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。Caché の場合、この接頭語は cache 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 Caché 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 Caché 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す。これは常に .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、上記の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。

例えば、データベースのサイズが 1.3.6.1.4.1.16563.1.1.3.1.6.4.84.69.83.84.1 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.iscCache.cacheObjects
.cacheDBTab.cacheDBRow.cacheDBSize.TEST(instname).1(DBindex)
```

B.4.1 Caché 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、Caché サブエージェントが提供するデータの MIB を拡張することができます。

1. **%Monitor.Adaptor** クラスを継承するクラスに、Caché オブジェクトの定義を作成します。管理対象オブジェクトを %Monitor パッケージに追加する方法については、“Caché クラス・リファレンス”の %Monitor に関するドキュメント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。そのためのメソッドは以下の通りです。

```
$SYSTEM.MonitorTools.SNMP.CreateMIB()
```

CreateMIB() メソッドのパラメータについては、“Caché 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 ツリーのブランチを作成します。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。Caché 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。

すべての管理オブジェクト定義は、Caché エンタープライズ MIB ツリーと同じ一般構成 (*application.objects.table.row.item.indexes*) を使用します。どのテーブルでも、最初のインデックスは Caché 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、Caché 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。

