



InterSystems IRIS Business Intelligence の実装

Version 2023.1
2024-01-02

InterSystems IRIS Business Intelligence の実装
InterSystems IRIS Data Platform Version 2023.1 2024-01-02
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble®, InterSystems HealthShare®, InterSystems IRIS®, および TrakCare は、InterSystems Corporation の登録商標です。HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, および InterSystems TotalView™ For Asset Management は、InterSystems Corporation の商標です。TrakCare は、オーストラリアおよび EU における登録商標です。

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。

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働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。

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

1 アプリケーション内への Business Intelligence の埋め込み	1
1.1 Business Intelligence の機能	1
1.2 アプリケーションに追加される Business Intelligence コンポーネント	2
1.3 推奨アーキテクチャ	2
1.4 主な実装手順	3
1.5 実装ツール	4
1.6 サンプルのアクセス方法	5
2 Business Intelligence の初期設定の実行	7
2.1 Web アプリケーションの設定	7
2.2 Business Intelligence グローバルの別のデータベースへの配置	7
2.3 グローバルの代替マッピング	8
2.4 Web セッション・タイムアウト期間の調整	10
3 設定の構成	13
3.1 Business Intelligence の設定へのアクセス	13
3.2 基本設定の指定	14
3.3 電子メールをサポートするための Business Intelligence の構成	15
3.4 ワークリストのカスタマイズ	16
3.5 フィルタの既定値として使用する実行時変数の作成	17
3.5.1 実行時変数の編集	18
3.5.2 実行時変数の削除	18
3.6 指定可能なフィルタの既定値	18
3.7 アイコンの作成	19
3.8 カスタム・カラー・パレットの作成	19
4 データ・コネクタの定義	21
4.1 データ・コネクタの概要	21
4.2 基本データ・コネクタの定義	21
4.2.1 XData ブロックでのクエリの定義	22
4.2.2 出力仕様の定義	23
4.3 クエリ結果のプレビュー	24
4.4 実行時におけるクエリの定義	24
4.4.1 更新が要求された場合のレコードの制限	25
4.4.2 リストが要求された場合のレコードの制限	25
4.4.3 その他のコールバック	27
4.5 プログラムによるデータ・コネクタの使用	27
5 パフォーマンスのヒント	29
5.1 結果キャッシュおよびキューブの更新	29
5.2 キャッシュのバケットとファクトの順序	29
5.3 非アクティブのキャッシュ・バケットの削除	30
5.4 キューブ・セルの事前計算	30
5.4.1 セル・キャッシュの定義	30
5.4.2 キューブ・セルの事前計算	31
5.5 インデックス圧縮ユーティリティの使用法	32
5.6 バックグラウンド・タスクのワーカの割り当ての制限	32
6 カスタム・アクションの定義	33
6.1 概要	33

6.1.1 コンテキスト情報	34
6.2 アクションの動作の定義	34
6.2.1 アクションの宣言	34
6.2.2 アクションの動作の定義	35
6.3 使用可能なコンテキスト情報	36
6.3.1 シナリオ:ピボット・テーブルをデータ・ソースとして使用するピボット・テーブル・ウィジェ ト	36
6.3.2 シナリオ: KPI をデータ・ソースとして使用するピボット・テーブル・ウィジェット	37
6.3.3 シナリオ:ピボット・テーブルまたは KPI をデータ・ソースとして使用するスコアカード	38
6.4 クライアント側コマンドの実行	39
6.4.1 使用可能なコマンド	40
6.4.2 applyFilter と setFilter の詳細	41
6.5 別のダッシュボードの表示	42
6.6 キューブ・コンテキストからの SQL テーブルの生成	42
7 アプリケーションからダッシュボードへのアクセス	43
7.1 ダッシュボードへのアクセス	43
7.1.1 URL のエンコード	43
7.2 使用可能な URL パラメータ	44
7.3 SETTINGS パラメータのオプション	47
7.4 アプリケーションから他の Business Intelligence ページへのアクセス	50
8 キューブの最新状態の維持	51
8.1 概要	51
8.1.1 キューブの更新および関連キューブ	52
8.1.2 キューブの更新および結果キャッシュ	52
8.2 キューブの同期が機能する方法	53
8.2.1 キューブの同期が可能なとき	53
8.2.2 キューブの同期が不可能なとき	54
8.2.3 ミラーリング環境におけるキューブの同期	54
8.2.4 キューブの同期のグローバルの構造	54
8.3 キューブ同期の有効化	56
8.4 ^OBJ.DSTIME グローバルのクリア	57
8.5 キューブ・マネージャを使用する方法	57
8.5.1 キューブ・マネージャの概要	58
8.5.2 更新計画の概要	58
8.5.3 キューブ・マネージャへのアクセス	59
8.5.4 レジストリの詳細の変更	61
8.5.5 キューブ・グループの登録	62
8.5.6 更新計画の指定	62
8.5.7 グループのマージ	63
8.5.8 登録されているすべてのキューブの構築	64
8.5.9 オンデマンド構築の実行	64
8.5.10 キューブ・グループの登録解除	65
8.5.11 キューブ・マネージャ・イベントの表示	65
8.5.12 キューブ・マネージャへのアクセスの制限	65
8.6 %SynchronizeCube() の使用	66
8.7 キューブの無効化	66
8.8 DSTIME の削除	67
8.9 キューブの手動更新	67
8.10 その他のオプション	68

8.10.1	DSTIME=MANUAL を使用する方法	68
8.10.2	ファクト・テーブルへのファクトの挿入	69
8.10.3	ディメンジョン・テーブルの事前構築	69
8.10.4	手動でのディメンジョン・テーブルの更新	70
8.11	例	71
9	Business Intelligence クエリのプログラムによる実行	73
9.1	結果セット API の使用	73
9.2	基本的な例	74
9.3	クエリの準備と実行	75
9.4	クエリ結果の出力	76
9.5	クエリ結果の検証	77
9.5.1	列数と行数の取得	77
9.5.2	指定のセルの値の取得	78
9.5.3	列または行のラベルの取得	78
9.5.4	セル・コンテンツの詳細の取得	80
9.6	DRILLTHROUGH クエリのクエリ結果の検証	82
9.7	クエリ・メタデータの検証	83
9.8	その他のメソッド	85
9.9	クエリ・ファイルの実行	85
9.9.1	クエリ・ファイルについて	86
9.9.2	クエリ・ファイルの実行	86
10	Business Intelligence のローカライズの実行	89
10.1	Business Intelligence のローカライズの概要	89
10.1.1	モデルのローカライズ	89
10.1.2	フォルダ項目のローカライズ	89
10.2	モデルのローカライズの準備	90
10.3	フォルダ項目のローカライズの準備	91
10.3.1	既定のドメイン	91
10.3.2	メッセージ・ディクショナリへの文字列の追加	91
10.3.3	ダッシュボード、ピボット・テーブル、または他のフォルダ項目でのローカライズ可能な文字列の使用	91
10.4	文字列のローカライズ	93
11	クラスへの Business Intelligence 要素のパッケージ化	95
11.1	概要	95
11.2	コンテナ・クラスへのフォルダ項目のエクスポート	96
11.3	移植のための Business Intelligence のフォルダ項目の編集	97
11.3.1	<filterState> 要素の削除	97
11.3.2	ローカル・データの削除	97
11.4	エクスポートしたコンテナ・クラスのインポート	98
11.5	フォルダマネージャの使用法	98
11.5.1	フォルダ項目の依存関係の表示	98
11.5.2	サーバへの Business Intelligence フォルダ項目のエクスポート	99
11.5.3	ブラウザへの Business Intelligence フォルダ項目のエクスポート	100
11.5.4	Business Intelligence フォルダ項目のインポート	101
12	ダッシュボードで使用するポートレットの作成	103
12.1	ポートレットの基本	103
12.2	設定の定義と使用	104
12.2.1	設定のタイプ	104
12.2.2	URL を介して渡される設定の受け取り	105

12.2.3 設定の使用	106
12.3 例	106
13 Business Intelligence のその他の開発作業	109
13.1 用紙サイズの追加	109
13.2 ユーザ・アクティビティの監査	109
13.2.1 監査コードの要件とオプション	110
13.2.2 例	111
13.3 サーバ初期化コードの定義	111
14 Business Intelligence のセキュリティの設定	113
14.1 セキュリティの概要	113
14.2 基本要件	114
14.3 Business Intelligence の一般的なタスクに関するセキュリティ要件	115
14.4 モデル要素に対するセキュリティの追加	117
14.5 ダッシュボードまたはピボット・テーブルのリソースの指定	118
14.6 フォルダのリソースの指定	118
付録A: キューブ・バージョンの使用	121
A.1 キューブ・バージョン機能の概要	121
A.1.1 キューブの最新状態の維持	122
A.1.2 クエリ中断の原因となり得るモデルの変更	122
A.2 バージョンをサポートするためのキューブの変更	123
A.2.1 キューブのバージョンとリレーションシップ	124
A.2.2 %ActivatePendingCubeVersion() の詳細	125
A.3 キューブ・バージョンの更新	125
A.4 操作対象のキューブの指定	127
A.5 追加のオプション	128
A.5.1 キューブ・バージョン機能の無効化	128
付録B: Analytics エンジンの仕組み	129
B.1 概要	129
B.1.1 ビットマップ・インデックスの使用	129
B.1.2 キャッシュ処理	129
B.1.3 バケット	131
B.2 エンジン・ステップ	132
B.3 軸のたたみ込み	133
B.4 クエリ・プラン	134
B.5 クエリ統計	134
付録C: MDX パフォーマンス・ユーティリティの使用	137
付録D: InterSystems Business Intelligence の診断	139
付録E: Business Intelligence のその他のエクスポート/インポート・オプション	141
E.1 Business Intelligence コンテナ・クラスの作成	141
E.2 フォルダ項目のエクスポートとインポート	142
E.2.1 プログラムによるフォルダ項目のエクスポート	142
E.2.2 プログラムによるフォルダ項目のインポート	143
付録F: Business Intelligence と災害復旧	145
F.1 構成	145
F.2 災害復旧	145

1

アプリケーション内への Business Intelligence の埋め込み

InterSystems IRIS® Business Intelligence は、ビジネス・インテリジェンス (BI) をアプリケーション内に埋め込むことを可能にすることで、ユーザがデータに関する高度な質問をし、回答できるようにします。ここでは、追加可能な機能、全体的なプロセス、および使用するツールの概要を示します。

この章では、InterSystems IRIS Business Intelligence の概要、実装ツール、および実装プロセスについて説明します。

Business Intelligence のシステム要件に関する情報については、このリリース向けのオンライン・ドキュメント “インターシステムズのサポート対象プラットフォーム” を参照してください。

1.1 Business Intelligence の機能

アプリケーションには、グラフィカルなウィジェットが含まれるダッシュボードを含めることができます。ウィジェットはデータを表示し、ピボット・テーブルおよび KPI (重要業績評価指標) に従って動作します。ピボット・テーブルでは、ソース値を表示するリストを表示できます。

ピボット・テーブル、KPI およびリストはクエリで、実行時に実行されます。

- ・ ピボット・テーブルは、ユーザが行うフィルタ選択など、実行時の入力に応答できます。内部的には、キューブと通信を行う MDX (多次元式) クエリが使用されます。

キューブは、ファクト・テーブルとインデックスで構成されます。ファクト・テーブルは、一連のファクト (行) で構成され、各ファクトは、ベース・レコードに対応します。例えば、ファクトを患者や診療科に対応させることができます。また、システムにより一連のレベル・テーブルも生成されます。すべてのテーブルが動的に維持されます。

構成および実装に応じて、システムはトランザクション・テーブルの変更を検出し、必要に応じてファクト・テーブルに反映させます。

ユーザがアナライザでピボット・テーブルを作成すると、システムは、MDX クエリを自動生成します。

- ・ KPI も実行時のユーザ入力に反応させることができます。内部的には、MDX クエリ (キューブの場合)、または SQL クエリ (任意のテーブルの場合) を使用します。

いずれの場合も、クエリは手動で作成することも、別の場所からコピーすることもできます。

- ・ リストには、ユーザが選択したピボット・テーブルの行に使用されたソース・レコードから選択された値が表示されます。内部的には、リストは SQL クエリです。

使用するフィールドを指定して、システムに実際のクエリを生成させることができます。また、クエリ全体を指定することもできます。

ダッシュボードには、アクションを起動するボタンやその他のコントロールを含めることができます。アクションによって、フィルタの適用または設定、ダッシュボードの更新、他のダッシュボードや URL のオープン、カスタム・コードの実行などが可能です。システムには一連の標準アクションが用意されていますが、ユーザはカスタム・アクションを定義できます。

1.2 アプリケーションに追加される Business Intelligence コンポーネント

アプリケーションに Business Intelligence を追加するには、以下のコンポーネントの一部またはすべてを追加します。

- ・ データ・コネクタ・クラス。データ・コネクタを使用すると、任意の SQL クエリをキューブまたはリストのソースとして使用できます。
- ・ キューブ定義クラス。キューブでは、Business Intelligence のピボット・テーブル内で使用される要素を定義し、対応するファクト・テーブルおよびインデックスの構造やコンテンツを制御します。

キューブ定義は、その基礎として使用するトランザクション・クラス（または、データ・コネクタ）をポイントします。

使用できるキューブの数に制限はありません。また、指定のクラスを複数のキューブの基礎として使用できます。

システムは、キューブごとにファクト・テーブル・クラスおよびその他のクラスを生成し、データを入力します。

- ・ サブジェクト領域クラス。
サブジェクト領域は、主にフィルタ処理されたキューブです（サブジェクト領域には、必要に応じて、フィルタおよびキューブ定義のさまざまな部分に対するオーバーライドが含まれます）。Business Intelligence では、キューブとサブジェクト領域を互換的に使用できます。
- ・ KPI 定義クラス。
KPI は、カスタム・クエリ（特に、ユーザ入力に基づいて実行時に決定されるクエリ）が必要な場合に定義します。
また、カスタム・アクションが必要な場合にも KPI を定義します。これは、アクションが KPI クラスに含まれるためです。
- ・ ピボット・テーブル。これらはドラッグ・アンド・ドロップ操作により作成します。システムは、基礎となる MDX クエリを生成します。
- ・ ダッシュボード。基礎となるクエリを実行して結果を表示することで、ピボット・テーブルや KPI を表示します。
- ・ ユーザ・ポータル。ピボット・テーブルやダッシュボードを表示します。

1.3 推奨アーキテクチャ

“高可用性ガイド”で述べているとおり、一般的に高可用性手法の一環としてミラーリングの使用をお勧めします。大規模なアプリケーションの場合、フェイルオーバー・ペア、非同期レポート・メンバ、および 1 つ以上の非同期災害復旧メンバを含むミラーが推奨されます。

分析アプリケーションは、実行時にインスタンスで使用可能なすべての処理能力を使用できます。レポート非同期メンバを介して分析機能を提供することで、ミラーのフェイルオーバー・メンバは高いトランザクション・ボリュームを適切に維持できます。

具体的には以下のとおりです。

- ・ コードとデータが別々のデータベースに存在するようにアプリケーションを定義します。これは必須ではありませんが、一般的なアーキテクチャです。
- ・ アプリケーション・データが非同期レポート・メンバにミラーリングされるようにミラーリングを設定します。
- ・ システムがアプリケーション・データにアクセスできるように、一部またはすべてのアプリケーション・クラスとその他のコードもレポート非同期メンバにコピーします。
一般に、アプリケーション・コードをミラーリングする必要はありません。
- ・ レポート非同期メンバに、キューブの定義と（必要に応じて）データを格納するデータベースを作成します。
必要に応じて、Business Intelligence ファクト・テーブルやその他の大容量 Business Intelligence データを格納する別のデータベースを作成します。システムで使用するグローバルについては、[以下の章](#)で説明します。
- ・ 非同期レポート・メンバに、Business Intelligence を実行するネームスペースを定義します。このネームスペースで、このサーバ上のアプリケーション・データ、アプリケーション・コード、キューブ定義、および Business Intelligence データにアクセスするためのマッピングを定義します。

小規模なアプリケーションやデモについては、すべてのコードおよびデータを同じデータベースに格納してもかまいません。

Business Intelligence の災害復旧に関する推奨事項は、“[Business Intelligence と災害復旧](#)”を参照してください。

また、Business Intelligence のシステム要件に関する情報については、“[インターシステムズのサポート対象プラットフォーム](#)”を参照してください。

1.4 主な実装手順

実装プロセスには、以下の手順が含まれます。

1. Business Intelligence を使用するネームスペースで Web アプリケーションがまだ定義されていない場合は、そのネームスペースに対して Web アプリケーションを定義します。“[初期設定の実行](#)”を参照してください。
2. 必要に応じて、パフォーマンス向上のために、他のデータベースから Business Intelligence グローバルをマップします。“[初期設定の実行](#)”を参照してください。
3. キューブおよびオプションのサブジェクト領域を作成します。このプロセスには以下の手順が含まれます。これらの手順は必要に応じて繰り返してください。
 - a. 1 つ以上のキューブを定義します。この手順では、アーキテクト、スタジオ、またはその両方を使用します。
 - b. キューブを構築します。ここでは、アーキテクトまたはターミナルを使用します。
 - c. アナライザを使用してキューブを表示し、検証します。

キューブを定義したら、それらのキューブに基づいて任意のサブジェクト領域を定義します。

キューブとサブジェクト領域の作成の詳細は、“[InterSystems Business Intelligence のモデルの定義](#)”を参照してください。

アナライザの使用方法的詳細は、“[アナライザの使用法](#)”を参照してください。

4. 必要に応じて、KPI を作成します。“[InterSystems Business Intelligence の上級モデリング](#)”を参照してください。
5. 必要に応じて、カスタム・アクションを作成します。“[カスタム・アクションの定義](#)”を参照してください。
6. キューブを最新の状態に維持するために必要な変更を加えます。その方法は、データがどの程度新しい必要があるのかと、パフォーマンスに関する考慮事項によって異なります。

“[キューブの最新状態の維持](#)”を参照してください。

7. ピボット・テーブルとダッシュボードを作成します。“[アナライザの使用法](#)” および “[ダッシュボードの作成](#)”を参照してください。
8. 配置を容易にするために、ピボット・テーブルとダッシュボードを InterSystems IRIS クラスにパッケージ化します。
“[クラスへの Business Intelligence 要素のパッケージ化](#)”を参照してください。
9. アプリケーションからダッシュボードへのリンクを作成します。“[アプリケーションからダッシュボードへのアクセス](#)”を参照してください。

このプロセスにおいて適切な時点で、以下の操作が必要となる場合もあります。

- ・ データ・コネクタの作成 – “[データ・コネクタの定義](#)”を参照してください。
- ・ 設定の構成 – “[設定の構成](#)”を参照してください。
- ・ ローカライズの実行 – “[ローカライズの実行](#)”を参照してください。
- ・ ダッシュボードで使用するカスタム・ポートレットの定義 – “[ダッシュボードで使用するポートレットの作成](#)”を参照してください。
- ・ その他の開発タスクの実行 – “[その他の開発作業](#)”を参照してください。
- ・ セキュリティの設定 – “[Business Intelligence のセキュリティの設定](#)”を参照してください。

1.5 実装ツール

実装プロセスでは、以下のツールを使用します。

- ・ 管理ポータル の Business Intelligence セクションから使用できるツール：
 - アーキテクト – これを使用して、キューブおよびサブジェクト領域を定義します。また、キューブのコンパイルや構築（およびサブジェクト領域のコンパイル）もできます。
 - アナライザ – これを使用して、モデルの検証時にキューブおよびサブジェクト領域を調べます。後で、ピボット・テーブルの作成に使用します。
 - ユーザ・ポータル – これを使用して、ダッシュボードを定義します。
 - MDX クエリツール – これを使用して、MDX クエリを作成し、それらのクエリ・プランを表示します。
 - フォルダマネージャ – これを使用して、主に、ピボット・テーブルとダッシュボードをエクスポートし、それらの定義を InterSystems IRIS クラスにパッケージ化できるようにします。
また、このツールを使用して、リソースをフォルダに関連付けることもできます。
 - [設定] オプション – これを使用して、ユーザ・ポータルの外観や動作を指定したり、ダッシュボードで使用可能な変数を定義します。
 - Business Intelligence ログ – これを使用して、そのネームスペースの Business Intelligence 構築ログを表示します。
- ・ ターミナル – これを使用して、キューブを再構築したり、メソッドをテストします。
- ・ MDX シェル (ターミナル内で動作) – これを使用して、キューブやサブジェクト領域を調べたり、カスタム MDX クエリを作成してその結果を表示します。

- ・ 管理ポータルのその他のセクション – これらを使用し、必要に応じて、グローバルのマップ、Business Intelligence で使用するリソース、ロール、およびユーザの定義、および Business Intelligence ファクト・テーブルの検証を行います。
- ・ ユーティリティ・メソッド：
 - `%DeepSee.Utils` には、キューブの構築、キューブの同期化、セル・キャッシュのクリア、およびその他のタスクに使用できるメソッドが含まれています。
 - `%DeepSee.UserLibrary.Utils` には、フォルダマネージャでサポートされるタスクをプログラムで実行する場合に使用できるメソッドが含まれています。
- ・ データ・コネクタ・クラス (`%DeepSee.DataConnector`) – これを使用して、キューブおよびリストで任意の SQL クエリを使用できるようにします。
- ・ 結果セット API (`%DeepSee.ResultSet`) – これを使用して、MDX クエリをプログラムによって実行し、その結果にアクセスします。

1.6 サンプルのアクセス方法

このドキュメントのほとんどの例は、Samples-BI サンプル (<https://github.com/intersystems/Samples-BI>) または Samples-Aviation サンプル (<https://github.com/intersystems/Samples-Aviation>) の一部です。

サンプルは、専用のネームスペース (`SAMPLES` など) を作成して、そのネームスペースにロードすることをお勧めします。一般的な手順は、“InterSystems IRIS で使用するサンプルのダウンロード” を参照してください。

これらのサンプルには、キューブ定義、サブジェクト領域、KPI、データ・コネクタ、およびプラグインが含まれています。また、ピボット・テーブルおよびダッシュボードのサンプルも含まれています。

2

Business Intelligence の初期設定の実行

ここでは、Business Intelligence の実装プロセスの最初に実行する設定作業について説明します。

2.1 Web アプリケーションの設定

InterSystems IRIS® Business Intelligence を Web アプリケーションで使用するには、その Web アプリケーションを Analytics に対応するように設定する必要があります。具体的には、アプリケーションの構成時に [Analytics を有効にする] チェック・ボックスにチェックを付けると、Web アプリケーションが Analytics 対応になります。Web アプリケーションの定義と構成の詳細は、“アプリケーション” を参照してください。

アプリケーション名は、アプリケーションへのアクセス方法に影響を与えます。以下の表を参照してください。

Web アプリケーションの構成	管理ポータルで、Business Intelligence メニューは、この Web アプリケーションにリンクしています
<ul style="list-style-type: none">・ 名前は /csp/namespace です・ ネームスペースは namespace です。・ [Analytics を有効にする] にチェックが付いています	はい (Business Intelligence メニューは常にこの Web アプリケーションにアクセスしようとします。別のアプリケーションが既定として設定されている場合でも、[ネームスペースの既定アプリケーション] オプションを使用して、この Web アプリケーションへのアクセスを試行します)
<ul style="list-style-type: none">・ 名前は、/csp/namespace 以外の名前です・ ネームスペースは namespace です。・ [Analytics を有効にする] にチェックが付いています	いいえ (ブラウザで URL を入力すると、Web アプリケーションにまだアクセスできます)

2.2 Business Intelligence グローバルの別のデータベースへの配置

特定のネームスペースで Business Intelligence を使用すると、そのネームスペースで使用するデータベースに格納されるデータの量が増加します。ソース・テーブルが巨大になると、システムは、それに相当する大量の独自データを格納するようになります。Business Intelligence のキャッシュが要求するストレージ容量はさらに増加します。そのため、通常は、

いくつかの Business Intelligence グローバルを別々のデータベースにマップすることが良策といえます。すべての Business Intelligence グローバルを 1 つのデータベースにマップすることも、複数のマッピングを定義することもできます。その一例として、以下の手順では、すべての Business Intelligence グローバルを 1 つの独立したデータベースに配置する方法について説明します。

1. データベースを作成します。

その際、実行時の拡張で生じるディスクの断片化を回避するために、データベースを事前に拡張（つまり、その初期サイズを設定）しておくことも検討します。

2. Business Intelligence で使用する予定のクラスが含まれているネームスペースにグローバル・マッピングを追加します。そのためには、以下の操作を実行します。

- ・ [グローバルデータベース位置] で、作成したデータベースを選択します。
- ・ [グローバル名] に、DeepSee.* と入力します。

より使用に適したマッピングについて、[次のセクション](#)も参照してください。

3. このネームスペース内のすべてのキューブ、サブジェクト領域、および KPI クラスをリコンパイルします。

また、すべてのキューブを再構築します。

データベースの作成およびグローバルのマッピングの詳細は、“データベースの構成” および “ネームスペースへのグローバル、ルーチン、およびパッケージ・マッピングの追加” を参照してください。

2.3 グローバルの代替マッピング

Business Intelligence と関連グローバルは、別々のデータベースにマップすることが必要になることがあります。以下のテーブルに主なグローバルを示します。

項目	グローバル	コメント
ファクト・テーブル、およびそれらのインデックス	<ul style="list-style-type: none"> ・ ^DeepSee.Fact ・ ^DeepSee.FactRelation ・ ^DeepSee.Index 	最初にキューブを構築する際、これらのグローバルを格納するデータベースのジャーナリングを無効にすることもできます。その後、データベースのジャーナリングを有効にしてください。
キューブとソース・テーブルの同期を維持するために使用するグローバル	<ul style="list-style-type: none"> ・ ^OBJ.DSTIME ・ ^DeepSee.Update 	“ キューブの最新状態の維持 ” を参照してください。
キューブ内部	<ul style="list-style-type: none"> ・ ^DeepSee.Cubes ・ ^DeepSee.Dimension ・ ^DeepSee.DimensionI 	

項目	グローバル	コメント
キューブ・マネージャ	<ul style="list-style-type: none"> • ^DeepSee.CubeManager • ^DeepSee.CubeManager.CubeEventD • ^DeepSee.CubeManager.CubeEventI • ^DeepSee.CubeManager.CubeRegistr 	“キューブの最新状態の維持”の“ キューブ・マネージャを使用する方法 ”を参照してください。
リスト・グループ	^DeepSee.ListingGroups	“InterSystems Business Intelligence のモデルの定義”の“ リスト・グループの定義 ”を参照してください。
結果キャッシュ（大量のデータ・セットの場合）	<ul style="list-style-type: none"> • ^DeepSee.BucketList • ^DeepSee.Cache.* • ^DeepSee.JoinIndex • ^DeepSee.UpdateCounter • ^DeepSee.Listing 	これらのグローバルを格納するデータベースのジャーナリングを無効にできます。結果キャッシュの詳細は、“ キューブの更新および結果キャッシュ ”を参照してください。
アナライザとダッシュボード・デザイナーで作成された項目	<ul style="list-style-type: none"> • ^DeepSee.Filters • ^DeepSee.Folder* • ^DeepSee.FolderItem* 	“ アナライザの使用法 ”および“ ダッシュボードの作成 ”を参照してください。
条件リスト	• ^DeepSee.TermList	“ InterSystems Business Intelligence の上級モデリング ”を参照してください。
品質メジャー	• ^DeepSee.QMsrs	“ InterSystems Business Intelligence の上級モデリング ”を参照してください。
ピボット変数	• ^DeepSee.Variables	“アナライザの使用法”の“ ピボット変数の定義と使用 ”を参照してください。
その他のポータル・オプション	<ul style="list-style-type: none"> • ^DeepSee.DashboardSettings（ユーザ固有のダッシュボード設定） • ^DeepSee.User.SendTo（ユーザ電子メール・アドレス） • ^DeepSee.User.Settings（実行時変数） • ^DeepSee.User.Icons（カスタム・アイコン） • ^DeepSee.UserPortalSettings（一般設定、およびワークリスト設定） • ^DeepSee.UserPreferences（ユーザごとの最近の項目） • ^DeepSee.PaperSizes（“用紙サイズの追加”を参照） 	これらの大半については、“ 設定の構成 ”を参照してください。

項目	グローバル	コメント
カスタム・コード	<ul style="list-style-type: none"> ・ ^DeepSee.InitCode ・ ^DeepSee.AuditCode 	“ その他の開発作業 ”を参照してください。
最近の履歴とログ	<ul style="list-style-type: none"> ・ ^DeepSee.AgentLog ・ ^DeepSee.Last* ・ ^DeepSee.PivotError ・ ^DeepSee.QueryLog ・ ^DeepSee.Session ・ ^DeepSee.SQLError 	
InterSystems IRIS NLP	<ul style="list-style-type: none"> ・ ^IRIS.IK.* 	
処理に使用される内部	<ul style="list-style-type: none"> ・ ^DeepSee.ActiveTasks ・ ^DeepSee.Build ・ ^DeepSee.Cancel ・ ^DeepSee.ComputedSQL ・ ^DeepSee.Functions ・ ^DeepSee.IDList ・ ^DeepSee.Pivot ・ ^DeepSee.Shell ・ ^DeepSee.TaskGroups ・ ^DeepSee.Tasks ・ ^DeepSee.UI.Charts 	

これは包括的なリストではありません。システムでは名前の先頭が ^DeepSee である別のグローバルが使用されます。一般的に、ここに記載されていないグローバルは、少量のデータのみを格納するものか、簡単に定義しただけのものです。

2.4 Web セッション・タイムアウト期間の調整

ユーザ・ポータルは、作業中のネームスペースの Web セッション・タイムアウト期間に従います。既定のセッション・タイムアウト期間は 15 分ですが、場合によっては十分でないこともあります。

Web タイムアウト期間を延長する手順は以下のとおりです。

1. 管理ポータルに移動します。
2. [システム] > [システム管理] > [セキュリティ] > [アプリケーション] > [Web アプリケーション] をクリックします。
3. Business Intelligence を使用するネームスペースの行にある **[編集]** をクリックします。

4. Web セッションの既定のタイムアウト期間 (秒単位) を指定する [セッションタイムアウト] の値を変更します。
5. [保存] をクリックします。

3

設定の構成

ここでは、[実装プロセス](#)の一環として、InterSystems IRIS® Business Intelligence の外観や動作に影響を及ぼすオプションを構成する方法を説明します。

3.1 Business Intelligence の設定へのアクセス

Business Intelligence の設定にアクセスするには、以下の操作を実行します。

1. [InterSystems ランチャー] をクリックし、[管理ポータル] をクリックします。
セキュリティの設定によっては、InterSystems IRIS® ユーザ名とパスワードを使用してログインするように求められます。
2. 以下のように、適切なネームスペースに切り替えます。
 - a. [変更] をクリックします。
 - b. ネームスペースをクリックします。
 - c. [OK] をクリックします。
3. [Analytics]→[管理]→[設定] をクリックします。

次のページが表示されます。

3.2 基本設定の指定

[一般] タブで、以下の設定を指定できます。

- ・ **[一般の配色]** – ユーザ・ポータル配色を選択します。
- ・ **[グラフの系列の配色]** – グラフの系列の配色を選択します。これは、既定の配色として使用されます。ダッシュボード・エディタを使用して、指定されたグラフに別の配色を適用できます。
- ・ **[ホーム・ページ・タイトル]** – ブラウザ・ページまたはタブのタイトルを指定します。
- ・ **[会社名]** – ユーザ・ポータル右上の領域に表示されるタイトルを選択します。
これを指定した場合は、**[会社のロゴ]** を指定しないでください。
- ・ **[会社のロゴ]** – 会社名の右側に表示されるイメージの URL を指定します。
`http://` で始まる完全 URL か、そのネームスペースに対して定義されている Web アプリケーションを基準とした相対 URL のいずれかを指定します。
これを指定した場合、**[会社名]** は無視されます。
- ・ **[会社のリンク]** – 右上に表示される会社のロゴまたは会社名をユーザがクリックすると開かれる URL を指定します。
`http://` で始まる完全 URL か、そのネームスペースに対して定義されている Web アプリケーションを基準とした相対 URL のいずれかを指定します。
- ・ **[Google Maps API キー]** – Google Maps API に使用するキーを指定します。Google では Google Maps ライブラリの使用に関するポリシーを変更し、すべての新規インストールで API キーが機能することが必要になります。詳細は、["Google Maps API に関するドキュメント"](#) を参照してください。
- ・ **[ダッシュボードの電子メール]** – [次のトピック](#) を参照してください。
- ・ **[デフォルト・リソース]** – ピボット・テーブルやダッシュボードを保護するために使用する既定のリソース。
["Business Intelligence 要素に対するセキュリティの追加"](#) を参照してください。
- ・ **[ダッシュボードのタイトルなし]** – このオプションを選択すると、ユーザ・ポータルおよびすべてのダッシュボードのタイトル領域が表示されなくなります。タイトル領域は以下の領域です。

このオプションは、NOTITLE URL パラメータと等価です。“[使用可能な URL パラメータ](#)”を参照してください。

- ・ **[ダッシュボードの境界線なし]** – このオプションを選択すると、ユーザ・ポータルおよびすべてのダッシュボードの境界線が表示されなくなります。このオプションは、NOBORDER URL パラメータと等価です。“[使用可能な URL パラメータ](#)”を参照してください。
- ・ **[フィルタに計算メンバを表示]** – このオプションが選択されている場合、既存のキューブ・ディメンジョンの一部である計算メンバがフィルタに表示されます。この設定は、計算メンバの定義によって作成された特殊なディメンジョンの一部である計算メンバには影響しません。
- ・ **[自動保存]** – これらのオプションを使用して、このネームスペース内で自動保存機能を有効または無効にします。**[アナライザ]** チェック・ボックスにチェックが付いている場合、ピボット・テーブルごとに、各ユーザのアナライザの状態が自動的に保存されます。この結果として、ユーザがアナライザでピボット・テーブルを開くと、そのピボット・テーブルは、そのユーザによって最後に表示されたときと同じ状態で表示されます。

同様に、**[ユーザ・ポータル設定]** チェック・ボックスにチェックが付いている場合は、ダッシュボードごとに、各ユーザのユーザ・ポータルの状態が自動的に保存されます。

アナライザとユーザ・ポータルの両方で、自動保存状態をクリアするためのオプションが用意されています。(すべての自動保存データをプログラムによって削除することもできます。%DeepSee.UserLibrary.Utils の %KillAutosaveFolders() メソッドを参照してください。)

このタブで何らかの変更を行った後は、**[保存]** をクリックします。

3.3 電子メールをサポートするための Business Intelligence の構成

[一般] タブで、ユーザが電子メールをダッシュボード内から送信できるように Business Intelligence を構成できます。そのためには、**[ダッシュボードの電子メール]** 設定を使用します。以下のいずれかを選択します。

- ・ **[クライアント側電子メールを使用]** – Business Intelligence で電子メールを有効にします。ユーザが電子メールを送信すると、システムは既定のクライアント側電子メール・システムにアクセスし、ユーザはこのシステムを使用してメッセージを送信します。メッセージにはダッシュボードへのリンクが記載され、ユーザはメッセージを編集できます。
- ・ **[サーバ側電子メールを使用]** – Business Intelligence で電子メールを有効にします。ユーザが電子メールを送信すると、電子メール・アドレスとオプションのコメントをユーザが入力するためのダイアログ・ボックスが表示され、入力内容が生成メッセージに追加されます。この既定メッセージにはダッシュボードへのリンクが記載されます。その後、その電子メールが SMTP サーバを介して送信されます。

これを選択すると、SMTP サーバを使用するように InterSystems IRIS を構成する必要もあります。“システム管理ガイド”の“[タスク・マネージャの電子メール設定の構成](#)”を参照してください。

- ・ **[無効]** – Business Intelligence 内での電子メールのサポートを無効にします。
これが既定値です。

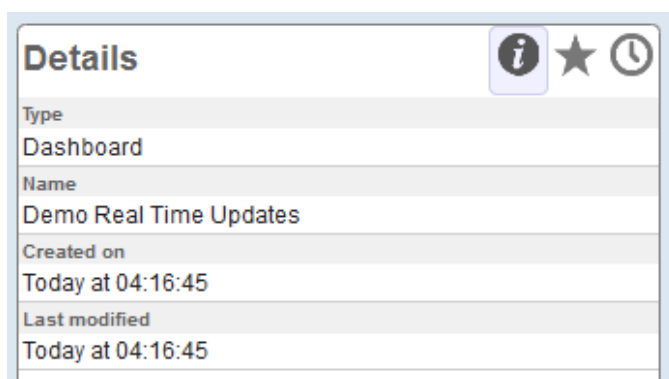
3.4 ワークリストのカスタマイズ

[ワークリスト] タブで、ワークリストの表示方法をカスタマイズできます。そのためには、[ワークリストのカスタマイズ] をクリックし、以下のグループでオプションを選択します。

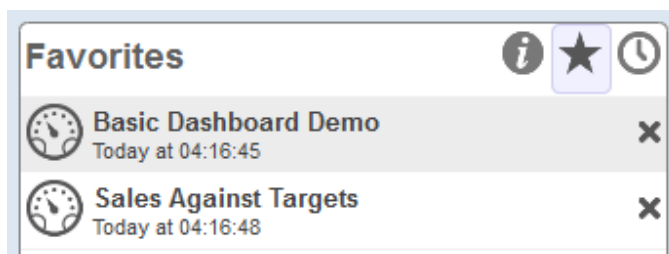
- ・ [ホーム・ページの上部パネル] および [ホーム・ページの下部パネル] オプションは、ユーザ・ポータルで使用可能なワークリストを指定します。ユーザ・ポータルの左側には、常に 2 個のワークリスト領域があります。
- ・ [ダッシュボード・ページの上部パネル] および [ダッシュボード・ページの下部パネル] オプションは、ダッシュボードで使用可能なワークリストを指定します。ダッシュボードの左側には、その構成に応じて、0 個、1 個、または 2 個のワークリスト領域があります。

このページの各セクションで、対応する領域で使用可能なワークリストを選択します。使用可能なワークリストは以下のとおりです。

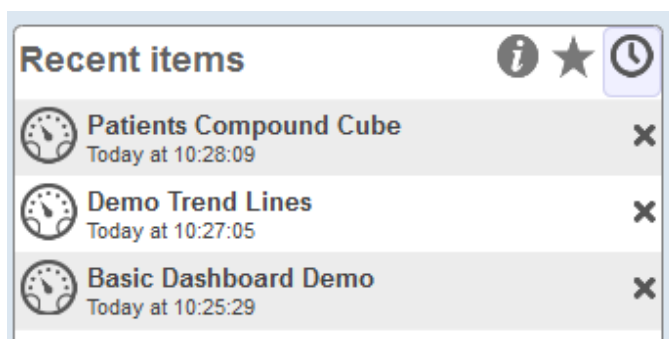
- ・ [詳細] ワークリストには、ユーザが選択したピボット・テーブルまたはダッシュボードの詳細が表示されます。以下はその例です。



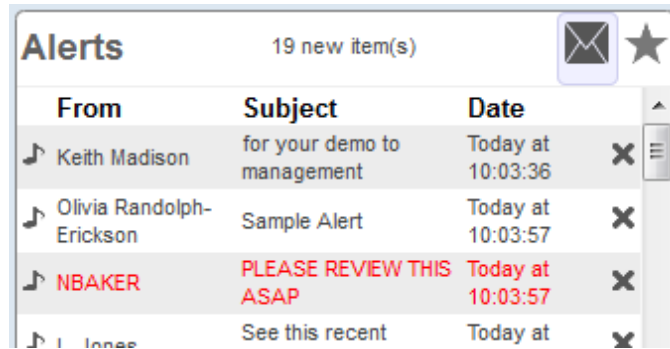
- ・ [お気に入り] ワークリストには、ユーザがお気に入りとしてマーク付けした項目がすべて表示されます。以下はその例です。



- ・ [最近の項目] ワークリストには、ユーザが最近アクセスした項目が表示されます。以下はその例です。



- ・ **[警告]** ワークリストには、最近ユーザに送信されたアラートが表示されます。以下はその例です。



From	Subject	Date	
Keith Madison	for your demo to management	Today at 10:03:36	X
Olivia Randolph-Erickson	Sample Alert	Today at 10:03:57	X
NBAKER	PLEASE REVIEW THIS ASAP	Today at 10:03:57	X
J. Jones	See this recent	Today at	X

- ・ **[フィルタ]** ワークリストには、ダッシュボード内のフィルタおよびその他のコントロールが表示されます。以下はその例です。



Filters

Home ZIP Code

Patient Group

Diagnoses

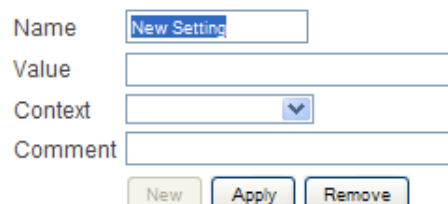
3.5 フィルタの既定値として使用する実行時変数の作成

[実行時変数] タブで、論理名と値（実行時に評価される ObjectScript 式）を持つ変数を定義できます。これらは、ダッシュボード内でフィルタの既定値に使用します。

設定を追加する手順は以下のとおりです。

1. **[新規作成]** をクリックします。

以下のようにページに表示されます。



Name

Value

Context

Comment

2. 以下の詳細を指定します。

- ・ **[名前]** – 変数の名前を指定します。
- ・ **[値]** – ObjectScript 式を指定します。

値には、任意の有効な ObjectScript 式を指定できます。例えば、クラス・メソッドまたはルーチンの呼び出しを指定できます。このメソッドやルーチンでは、\$USERNAME や \$ROLES などの特別な変数を使用できます。

指定可能な値の詳細は、“[指定可能なフィルタの既定値](#)”を参照してください。

- ・ **[コンテキスト]** – この式を使用するコンテキストを指定するには、**DefaultFilterValue** を選択します。これにより、ウィジェットにコントロールを追加すると、フィルタに使用できる既定値として、ウィジェット・エディタには、この設定がリストされます。

値 **Other** は、現在使用されていません。

- ・ **[説明]** – 必要に応じて、コメントを指定します。

3. [適用] をクリックします。

以下のように、変数がテーブルに追加されると共に、その現在の値も表示されます。

Name	Value	Context	Comment	Evaluates to	
DefaultPatGroup	##class(MyApp.Utils).GetDefaultPatGroup()	DefaultFilterValue	Uses \$ROLE	&[Group B]	✖
DefaultZIP	##class(MyApp.Utils).GetDefaultZIP()	DefaultFilterValue	Uses \$ROLE	&[34577]	✖

3.5.1 実行時変数の編集

実行時変数を編集する手順は以下のとおりです。

1. テーブルで目的の変数をクリックします。
2. テーブルの下にある領域で詳細を編集します。
3. **[適用]** をクリックします。

3.5.2 実行時変数の削除

実行時変数を削除するには、その変数の行にある [X] をクリックします。

変数が即座に削除されます。

3.6 指定可能なフィルタの既定値

以下のテーブルに、MDX ベースのデータ・ソースで使用する際にフィルタで指定可能な既定値を示します。フィルタの既定値として使用する実行時変数を定義する場合や、このドキュメントで説明されている他の方法でフィルタを指定する場合は、このテーブルの情報を使用してください。

シナリオ	目的の値を返す式
単一のメンバ	"&[keyval]"。keyval はメンバのキーです。“InterSystems MDX リファレンス”の“ キー値 ”を参照してください。
メンバの範囲	"&[keyval1]:&[keyval2]"
メンバのセット	"{&[keyval1],&[keyval2],&[keyval3]}"
指定された単一のメンバを除く、レベルのすべてのメンバ	"%NOT &[keyval]"
指定されたサブセットを除く、レベルのすべてのメンバ	"%NOT{&[keyval1],&[keyval2],&[keyval3]}"

MDX ベースのデータ・ソースの場合、フィルタ名とフィルタ値では大文字と小文字は区別されません (オプションの %NOT 文字列を除く)。

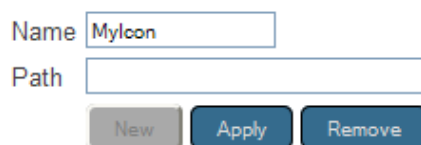
3.7 アイコンの作成

[**ユーザ定義アイコン**] タブで、論理名を持つ再使用可能なアイコンを定義できます。それらのアイコンは、条件付き書式を持つピボット・テーブル内、およびダッシュボードのウィジェット・コントロール内で使用できます。

アイコンを追加する手順は以下のとおりです。

1. [**新規作成**] をクリックします。

ページの下部領域に以下のように表示されます。



2. [**名前**] で、このアイコンを参照するために使用する名前を指定します。
3. [**パス**] で、アイコン・ファイルの場所を指定します。以下のいずれかを行います。

- ・ install-dir/CSP/broker/ を基準とした相対パスを指定します。
- ・ 完全 URL を指定します。

4. [**適用**] をクリックします。

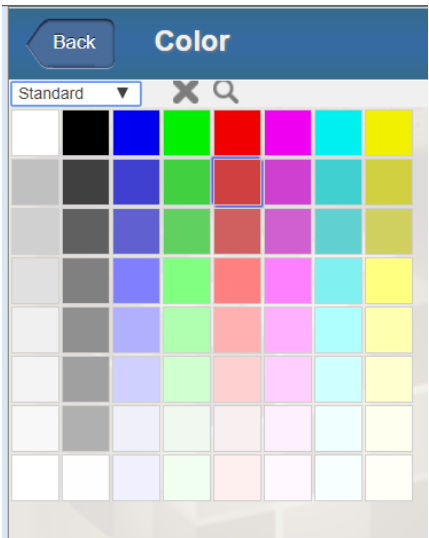
アイコンがテーブルに追加されると共に、プレビューも表示されます。

アイコンの編集または削除は、実行時変数の場合と同じ方法で実行することができます。詳細は、前のセクションを参照してください。

条件付き書式を設定したピボット・テーブルでのアイコンの使用法の詳細は、“アナライザの使用法”の“[条件付き書式の適用](#)”を参照してください。ウィジェット・コントロールの構成の詳細は、“ダッシュボードの作成”の“[コントロールの追加](#)”を参照してください。

3.8 カスタム・カラー・パレットの作成

ダッシュボード・エディタで使用するために、カスタム・カラー・パレットを作成することもできます。これによって、カラー・ピッカが提供されます。以下に、既定のカラー・パレットの 1 つを含むカラー・ピッカを示します。



カスタム・カラー・パレットを追加するには、以下のようにノードを `^DeepSee.UserPortalColorSets` グローバルに追加します。

ノード	値
<code>^DeepSee.UserPortalColorSets(n)</code> 。 <code>n</code> は整数で、前のノードからグローバルでインクリメントされます。	<p>以下の項目が以下の順序で含まれている <code>\$LISTBUILD</code> リストです。</p> <ol style="list-style-type: none">1. カラー・パレットの論理名。2. カラー・パレットの表示名。必要に応じて、<code>\$\$\$Text()</code> を使用して、この名前をローカライズ可能にします。3. セミコロンで区切られた CSS 色名のリスト。

以下に例を示します。

ObjectScript

```
set colorlist = "darkturquoise;greenyellow;hotpink;floralwhite;palevioletred;plum;"
set colorlist = colorlist _"powderblue;palegreen;plum;mediumaquamarine;linen;"
set colorlist = colorlist _"lightsteelblue;lightpink;oldlace;lightsalmon;gold;"
set mycolors=$LB("My Custom Colors","My Custom Colors",colorlist)
set ^DeepSee.UserPortalColorSets($I(^DeepSee.UserPortalColorSets)) = mycolors
```

ユーザがカラー・パレットを選択すると、グリッドに各色のサンプルが表示されます。最大 64 色まで指定できます。

4

データ・コネクタの定義

ここでは、Business Intelligence の実装プロセスの一環として、データ・コネクタを定義する方法を説明します。

4.1 データ・コネクタの概要

データ・コネクタは、キューブのソースとして、詳細なリストで、またはそれらの両方で使用可能なオブジェクトに、任意の SQL SELECT クエリの結果をマップします。(キューブおよびリストの定義の詳細は、“[InterSystems Business Intelligence のモデルの定義](#)”を参照してください。)

SQL クエリでは、以下の項目の組み合わせを使用できます。

- ・ InterSystems IRIS Business Intelligence を使用しているネームスペース内のローカル・テーブル。
- ・ 同じネームスペース内のビュー。
- ・ 同じネームスペース内のリンクされたテーブル。[リンク・テーブル・ウィザード](#)を使用して、リンクされたテーブルを定義します。このテーブルは、ネームスペース内にクラス定義を持ちますが、外部データベース内のテーブルにリンクされています。

重要 リンクされたテーブルを使用する際に、クエリで制約はありません。“InterSystems IRIS SQL ゲートウェイの使用法”の“[SQL ゲートウェイ・クエリに関する制限事項](#)”を参照してください。

キューブの更新をサポートするように、データ・コネクタを定義できます。このキューブを更新するには、キューブ全体を再構築するか、ProcessFact() を使用する必要があります。“[キューブの最新状態の維持](#)”を参照してください。

4.2 基本データ・コネクタの定義

データ・コネクタを定義するには、以下のようにクラスを作成します。

- ・ `%DeepSee.DataConnector` を拡張する必要があります。
- ・ クエリを指定する必要があります。クエリは、[最初のサブセクション](#)で説明するように、XData ブロックで指定できます。

また、実行時にクエリを作成するコールバックを実装するという方法もあります。これについては、このページで[後述](#)します。

- ・ 2 番目のサブセクションで説明するように、クエリの列をプロパティにマップする出力仕様を定義する必要があります。
- ・ このデータ・コネクタをリストに使用する必要がある場合、そのクラスでは SUPPORTSIDLIST クラス・パラメータに 1 を指定する必要があります。

Class Member

```
Parameter SUPPORTSIDLIST = 1;
```

- ・ このデータ・コネクタをキューブに使用する必要がある場合、キューブの更新をサポートする予定がある場合、そのクラスでは SUPPORTSSINGLE パラメータに 1 を指定する必要があります。

Class Member

```
Parameter SUPPORTSSINGLE = 1;
```

データ・コネクタをコンパイルすると、`packagename.classname.ResultSet` という名前のクラスが生成されます (`packagename.classname` はデータ・コネクタ・クラス自体の完全名です)。生成されたクラスを編集しないでください。

4.2.1 XData ブロックでのクエリの定義

XData ブロックでクエリを定義するには、以下のように、データ・コネクタ・クラスに要素を追加します。

Class Member

```
XData SourceQuery [ XMLNamespace = "http://www.intersystems.com/deepsee/connector/query" ]
{
  <sql>SELECT %ID, DateOfSale, Product->Name AS ProductName FROM HoleFoods.SalesTransaction</sql>
}
```

メモ：

- ・ データ・コネクタが詳細リストまたは更新をサポートする必要がある場合は、この方法を使用できません。その場合は、代わりに、このページで後述する [“実行時におけるクエリの定義”](#) を参照してください。
- ・ この XData ブロックの名前は必ず `SourceQuery` とします。
- ・ XMLNamespace パラメータは、`"http://www.intersystems.com/deepsee/connector/query"` である必要があります。
- ・ XData ブロックには、実行する SQL クエリを含む `<sql>` 要素を 1 つ含める必要があります。
- ・ クエリは、必要な他のフィールドに加えて、レコードの ID を返す必要があります。
- ・ 「より小さい」を意味するシンボル (`<`) をクエリにインクルードするには、`<` を使用します。

以下はその例です。

```
<sql>SELECT A,B,C FROM MyApp.MyTable WHERE A&lt;'50'</sql>
```

同様に、アンパサンド (`&`) をクエリにインクルードするには、`&` を使用します。

- ・ 矢印構文を使用してフィールドにアクセスする場合は、フィールドのエイリアスの指定も必要になることがあります。具体的には、データ・コネクタをキューブの基礎として使用する場合やキューブ要素の定義でフィールドを使用する場合に、エイリアスが必要です。

例えば、以下のクエリを考えてみます。

```
SELECT %ID, DateOfSale, Product->Name FROM HoleFoods.SalesTransaction
```

この場合、キューブ定義が `Product->Name` フィールドを参照する方法はありません。`Product->Name` または `Product.Name` を使用すると、ビルド・プロセスによってエラーがスローされます。したがって、このフィールドはレベルやメジャーの基礎としては使用できません。

一方で、以下のクエリを考えてみます。

```
SELECT %ID, DateOfSale, Product->Name AS ProductName FROM HoleFoods.SalesTransaction
```

この場合、`ProductName` はソース・クラスのプロパティとして扱うことができるため、これに基づいてレベルまたはメジャーを定義できます。

4.2.2 出力仕様の定義

以下の例のように、すべてのデータ・コネクタ・クラスに、クエリの列をプロパティにマップする XData ブロックを設定する必要があります。

Class Member

```
XData Output [ XMLNamespace = "http://www.intersystems.com/deepsee/connector/output" ]
{
<connector>
  <property name="Gender" sourceProperty="Gender" />
  <property name="Age" sourceProperty="Age" type="%ZEN.Datatype.integer"/>
  <property name="HomeCity" sourceProperty="HomeCity"/>
  <property name="PatientGroup" sourceProperty="PatientGroup"
    transform='$CASE(%val,"A":"Group A","B":"Group B",:%val)' />
  <property name="TestScore" sourceProperty="TestScore" type="%ZEN.Datatype.integer"/>
</connector>
}
```

各 `<property>` 要素は、データ・コネクタのプロパティであり、Business Intelligence で使用可能です。

メモ：

- ・ XData ブロックの名前は必ず `Output` とします。
- ・ XMLNamespace パラメータは、`"http://www.intersystems.com/deepsee/connector/output"` である必要があります。
- ・ この XData ブロックには、`<connector>` 要素を 1 つ含める必要があります。
- ・ `<connector>` 要素には、1 つ以上の `<property>` 要素が含まれている必要があります。
- ・ 各 `<property>` 要素では、以下の属性の一部またはすべてを指定する必要があります。

属性	目的
name	プロパティの名前。キューブ内またはキューブのソース式内のソース・プロパティとして、またはリスト内のフィールドとして使用されます。
sourceProperty	結果セットの対応する列の名前。
type	(オプション) プロパティのデータ型。既定は、 <code>%Library.String</code> です。
transform	(オプション) 入力として <code>%val</code> (現在の列値) を使用し、変換後の値を返す式。

- ・ このデータ・コネクタをリストに使用する場合は、該当する `<property>` 要素に `idkey` 属性も指定します。この属性は、指定されたプロパティがデータ・セットの `IdKey` を表すことを示します。

`idKey="true"` で複数のフィールドをマークした場合、データ・コネクタはこれらのフィールドを組み合わせます。

注釈 データ・コネクタに基づくキューブと、そのキューブ内に同じくデータ・コネクタに基づくリストがある場合、これらのすべてのデータ・コネクタは、`idkey="true"` とマークされた同じプロパティを持つ必要があります。これは、基礎となるメカニズムがいずれの場合も同じ ID 値を使用するためです。

以下に、`idkey` の使用例を示します。

Class Member

```
XData Output [ XMLNamespace = " http://www.intersystems.com/deepsee/connector/output" ]
{
<connector >
  <property name= "%ID" sourceProperty ="ID" displayName ="Record ID" idKey= "true"/>
  <property name= "Product" sourceProperty ="Product" displayName ="Product name"/>
  <property name= "AmountOfSale" sourceProperty ="AmountOfSale" displayName ="Amount of sale"/>
</connector >
}
```

4.3 クエリ結果のプレビュー

データ・コネクタをテストするために、クエリ結果を直接表示できます。データ・コネクタの結果を簡単に確認するには、ターミナルで `%Print()` クラス・メソッドを使用します。以下はその例です。

```
d ##class(BI.Model.PatientsQuery).%Print()
1      1      SUBJ_1003 M      27      Redwood
2      2      SUBJ_1003 M      41      Magnolia
3      3      SUBJ_1003 F      42      Elm Heigh
...
```

既定では、このメソッドは結果の最初の 100 個のレコードを出力します。

このメソッドには、以下のシグニチャがあります。

```
classmethod %Print(ByRef pParameters, pMaxRows As %Integer = 100) as %Status
```

`pParameters` は現在使用されていません。`pMaxRows` は表示する行数の最大値です。

4.4 実行時におけるクエリの定義

XData ブロックで **ハードコードされたクエリ** を定義する代わりに、実行時にクエリを作成できます。データ・コネクタが詳細リストまたは更新をサポートする必要がある場合は、この方法を使用する必要があります。

実行時にクエリを作成するには、`%OnGetSourceResultSet()` メソッドを実装します。このメソッドには、以下のシグニチャがあります。

```
Method %OnGetSourceResultSet(ByRef pParameters, Output pResultSet) As %Status
```

`pParameters` は現在使用されていません。`pResultSet` は結果セットです。

実装では、以下の手順を実行します。

- このデータ・コネクタを多目的に使用する場合、データ・コネクタのインスタンスの `%mode` プロパティを調べます。データ・コネクタのインスタンスが作成されるときに、このプロパティが自動的に設定されます。このプロパティには、以下の値のいずれかがあります。
 - "all" – キューブが構築されていること、またはすべてのメンバが表示されていることを示します。

- ・ "idlist" - リストが要求されていることを示します。
 - ・ "single" - %ProcessFact() が実行されたことを示します。
2. %SQL.Statement のインスタンスを作成します。クエリは、必要な他のフィールドに加えて、レコードの ID を返す必要があります。

このデータ・コネクタが作成されたモードに応じて、クエリの詳細は異なります。通常は、以下のとおりです。

 - ・ "all" モードで使用する基本クエリを定義します。
 - ・ モードが "single" の場合、制限を追加し、更新されている単一レコードを取得します。[最初のサブセクション](#)で詳細に説明します。
 - ・ モードが "idlist" の場合、別の制限を追加し、レコードのサブセットを取得します。[2 番目のサブセクション](#)で詳細に説明します。
 3. 必要に応じて実行時の値にパラメータとして渡し、その文を実行します。以降のサブセクションの説明のとおり、文インスタンスのプロパティとして、特定の実行時の値を使用できます。

この手順によって、%SQL.StatementResult のインスタンスを作成します。
 4. %SQL.StatementResult のインスタンスを出力パラメータとして返します。

4.4.1 更新が要求された場合のレコードの制限

ProcessFact() でキューブを更新するときには、更新するレコードの ID を指定します。キューブで使用するためのデータ・コネクタを作成するときには、そのクエリが指定された ID のみを使用するように、ロジックを追加する必要があります。この場合、データ・コネクタの %singleId プロパティを使用できます。このプロパティには、更新対象のレコードの ID が格納されています。以下はその例です。

```
//do this when constructing the SQL statement
if (..%mode="single") {
    set sql = sql _ " where %ID = ?"
}

...
//do this when executing the SQL statement
if (..%mode="single") {
    set pResultSet = tStatement.%Execute(..%singleId)
}
```

ProcessFact() の詳細は、“[キューブの最新状態の維持](#)” の項目を参照してください。

4.4.2 リストが要求された場合のレコードの制限

ユーザがリストを要求すると、システムは指定のコンテキストで使用するレコードの ID を取得して、それらの ID を後で使用するために格納します。既定のリストの場合は、それらの ID がリストの SQL クエリで自動的に使用されます。リストで使用するデータ・コネクタを作成するときには、それらの ID をクエリで使用するためのロジックを追加する必要があります。

この場合、システムがリスト用の ID を格納する方法を理解する必要があります。これらの ID はテーブル（このキューブのリスト・テーブル）に書き込まれます。このテーブルには、以下の列が含まれています。

- ・ _DSqueryKey - リストを識別します。
- ・ _DSsourceId - 元のソース・データと同じ ID。

以下に例を示します。

#	_DSListingId	_DSqueryKey	_DSsourceId
1	83616140 3970	83616140	3970
2	83616140 4151	83616140	4151
3	83616140 4188	83616140	4188
4	83616140 6245	83616140	6245
5	83616140 8685	83616140	8685
6	2139316107 1337	2139316107	1337
7	2139316107 7071	2139316107	7071

ここでは、最初の 5 行が _DSsourceId 列で得られる 5 つのレコードの ID を使用する、リスト 83616140 に関連付けられています。それに続く 2 行が 2 つのレコードの ID を使用する、リスト 2139316107 に関連付けられています。

リスト・テーブルを使用するようにデータ・コネクタ・クエリを変更する方法は 2 つあります。

- ・ クエリに IN 節を追加して、サブクエリ内でリスト・テーブルの該当する行を使用します。以下に例を示します。

SQL

```
SELECT A,B,C FROM MyApp.MyTable
WHERE (ID IN (SELECT _DSsourceId FROM listingtable WHERE
_DSqueryKey=somekey))
```

この場合は以下ようになります。

- listingtable は、キューブのリスト・テーブル名です。このテーブル名を取得するには、データ・コネクタの %listingTable プロパティを使用します。
- somekey は、現在のリストの一意キーです。このキーを取得するには、データ・コネクタの %listingKey プロパティを使用します。

この方法では、<MAXSTRING> エラーなどのサイズに関連する問題が発生することがあります。

- ・ 適切な WHERE 節を使用して、ソース・テーブルとリスト・テーブル間で JOIN を実行します。

キューブのソースおよびリストのソースとして使用されるデータ・コネクタの例を以下に示します。リストのキーがパラメータとしてクエリに渡されていることに注意してください。

Class Member

```
Method %OnGetSourceResultSet(ByRef pParameters, Output pResultSet) As %Status
{
    set tSC = $$$OK
    set pResultSet = ""
    Try {
        set sql = "SELECT %ID, fdate, fname, ftimestamp FROM TestTD.TimeDimensions"
        //when we're using this for a listing, add WHERE clause to restrict to
        //the appropriate IDs (in the table given by the %listingTable property)

        if (..%mode="idlist") {
            set sql = sql _ " where %ID in (select _DSsourceId from "
            _ ..%listingTable _ " where _DSqueryKey = ?)"
        }

        set tStatement = ##class(%SQL.Statement).%New()
        set tSC = tStatement.%Prepare(.sql)

        If $$$ISERR(tSC) {
            set ex = ##class(%Exception.StatusException).CreateFromStatus(tSC)
            throw ex
        }

        //if we're using this for a listing, pass in the listing key as a parameter
        if (..%mode="idlist") {
            set pResultSet = tStatement.%Execute(..%listingKey)
        } else {
            set pResultSet = tStatement.%Execute()
        }

        //check %SQLCODE and report if there's an error
    }
```



```

    If pResultSet.%SQLCODE {
        set sqlcode=pResultSet.%SQLCODE
        set message=pResultSet.%Message
        set ex = ##class(%Exception.SQL).CreateFromSQLCODE(sqlcode, message)
        throw ex
    }
}
Catch(ex) {
    Set tSC = ex.AsStatus()
}
Quit tSC
}

```

4.4.3 その他のコールバック

%DeepSee.DataConnector クラスには、エラーの処理、行での変換の実行、フィルタ処理の実行などを行うようにカスタマイズできる追加のコールバック・メソッドが用意されています。これらには %OnNextRecord() および %OnProcessRecord() があります。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。

4.5 プログラムによるデータ・コネクタの使用

プログラムを使用してデータ・コネクタを使用するには、以下の手順を実行します。

1. データ・コネクタのインスタンスを作成します。
2. 結果セットを返すその %Execute() メソッドを呼び出します。このメソッドは、参照によってステータスも返します。
3. 返されたステータスをチェックします。
4. ステータスがエラーでなければ、その結果セットを使用できます。これは、%SQL.StatementResult のインスタンスです。

以下はその例です。

ObjectScript

```

Set tConnector=.%New()
Set tRS=tConnector.%Execute(,tSC)
If $$$ISERR(tSC) {Quit}
//use tRS as needed ...

```


5

パフォーマンスのヒント

ここでは、InterSystems IRIS® Business Intelligence のパフォーマンスに関するヒントを紹介します。これは、[実装プロセス](#)の一環として確認してください。["Business Intelligence グローバルの別のデータベースへの配置"](#)も参照してください。

パフォーマンスおよびトラブルシューティング・オプションの詳細は、[InterSystems Developer Community](#) を参照してください。

5.1 結果キャッシュおよびキューブの更新

(既定で) 64,000 レコードより多く使用するキューブの場合、システムは、結果キャッシュを保持して使用します。同期または再構築することでキューブを更新した場合、あるいは[手動で更新した後にを明示的に呼び出した](#)場合、結果キャッシュの一部は無効と見なされ、クリアされます。詳細はキューブ定義のオプションに応じて異なります(このページで後述する["キャッシュのバケットとファクトの順序"](#)を参照)。したがって、通常、頻繁にキューブを更新することは望ましくありません。

結果キャッシュは以下のように動作します。ユーザが(例えばアナライザを使用して)クエリを実行するたびに、システムはそのクエリの結果をキャッシュします。次回そのクエリを任意のユーザが実行すると、システムはそのキャッシュがまだ有効かどうかをチェックします。有効な場合は、キャッシュの値が使用されます。それ以外の場合、システムはクエリを再実行し、新しい値を使用して、それらをキャッシュします。その実質的な効果は、より多くのユーザがより多くのクエリを実行するほど、経時的にパフォーマンスが向上することです。

5.2 キャッシュのバケットとファクトの順序

[前述](#)のとおり、大量のデータ・セットの場合、システムは結果キャッシュを保持して使用します。この場合、ファクト・テーブルの行の順序を制御すると役立つことがあります。この順序によって、システムがキャッシュをどのように作成して使用するかが変化するからです。そのためには、キューブの[\[初期ビルド順\]](#) オプションを指定します。["InterSystems Business Intelligence のモデルの定義"](#) の["他のキューブ・オプション"](#) を参照してください。

ユーザがピボット・テーブルを評価する場合、システムは可能な限り後で再使用する集約値を計算してキャッシュします。キャッシュが再使用できるかどうかを判断するために、システムは以下のロジックを使用します。

1. 特定のシナリオで使用されるレコードの ID を調べます(例えば、特定のピボット・テーブルのセルについて)。
2. それらの ID が属するバケットを確認します。バケットとは、ファクト・テーブル内で連続する多数のレコードのことです(詳細は、後述します)。

- ・ (バケット内の少なくとも 1 つの ID に変更が加えられたために) バケットが更新されると、システムは、そのバケットに関連付けられた該当のキャッシュを破棄して、結果を再生成します。
- ・ バケットが更新されていない場合、システムは適切なキャッシュを再使用します (利用可能なキャッシュがある場合)。または、結果を再生成します (利用可能なキャッシュがない場合)。

シナリオによっては、ソース・レコードに加えた変更 (および、キューブに対するそれに対応する更新) は、最新のソース・レコードで最初に発生します。このようなシナリオでは、確実に古いレコードが最初になるように、レコードの古さ順でファクト・テーブルを構築することが有効です。この方法では、データへの変更によってより古い行のキャッシュが無効になることがなくなります (これに対して、古い行と新しい行がファクト・テーブル全体に混在していると、新しいレコードに変更が発生したときに、すべてのキャッシュが無効になる可能性があります)。

詳細は、“[Analytics エンジンの仕組み](#)” を参照してください。

5.3 非アクティブのキャッシュ・バケットの削除

キャッシュ・バケットが無効になった場合 ([前のセクション](#)を参照)、非アクティブとしてマークされますが、削除はされません。非アクティブのキャッシュ・バケットを削除するには、%DeepSee.Utils の %PurgeObsoleteCache() メソッドを呼び出します。例：

```
d ##class(%DeepSee.Utils).%PurgeObsoleteCache("patients")
```

5.4 キューブ・セルの事前計算

前述のように、ユーザがピボット・テーブルを評価する場合、システムは可能な限り後で再使用する集約値を計算してキャッシュします。つまり、このキャッシュ処理は、Business Intelligence で作業するユーザが増えるほどシステムの動作が速くなることを意味します (詳細は、“[Analytics エンジンの仕組み](#)” を参照してください)。

初期パフォーマンスも高速化するには、ピボット・テーブルで使用される特定の集約値を事前計算してキャッシュします。これは特に、パフォーマンスが重要となる場合に有効です。この機能は、以下のように実行されます。

- ・ キューブ・クラス内で、事前計算とキャッシュが必要なキューブ・セルを指定する追加の XData ブロック (CellCache) を指定します。詳細は、最初のサブセクションを参照してください。
- ・ それらのキューブ・セルをプログラムを使用して事前計算するには、ユーティリティ・メソッドを使用します。2 番目のサブセクションを参照してください。

これは、キューブの構築後に実行する必要があります。

重要 より単純なオプションは、単にクエリを事前に (つまり、ユーザが作業する前に) 実行することです。

5.4.1 セル・キャッシュの定義

キューブ・クラスには、事前計算およびキャッシュが可能なキューブ・セルを指定する追加の XData ブロック (CellCache) を格納できます。これによって Business Intelligence の初期パフォーマンスが高速化されます。以下に例を示します。

```
/// This xml document defines aggregates to be precomputed.
XData CellCache [ XMLNamespace = " http://www.intersystems.com/deepsee/cellCache" ]
{
<cellCache xmlns= "http://www.intersystems.com/deepsee/cellCache" >
  <group name= "BS">
    <item>
```

```

        <element>[Measures].[Big Sale Count]</element>
    </item>
</group>
<group name= "G1">
    <item>
        <element>[UnitsPerTransaction].[H1].[UnitsSold]</element>
        <element>[Measures].[Amount Sold]</element>
    </item>
    <item>
        <fact>DxUnitsSold</fact>
        <element>[Measures].[Amount Sold]</element>
    </item>
</group>
</cellCache>
}

```

<cellCache> 要素は以下のとおりです。

- ・ ネームスペース "http://www.intersystems.com/deepsee/cellCache" 内に存在している必要があります。
- ・ ゼロ個以上の <group> 要素が含まれます。

各 <group> 要素は以下のとおりです。

- ・ name 属性があります。この属性は、後で事前計算するセル・グループを指定するときに使用します。
- ・ 1 つ以上の <item> 要素が含まれます。

各 <item> 要素は、キューブ・インデックスの組み合わせを表し、%SHOWPLAN によって返される情報に対応します。

<item> 要素は、1 つ以上の <element> 要素で構成されます。

<element> には、以下の構造のいずれかが、任意の組み合わせで 1 つ以上含まれます。

```
<fact>fact_table_field_name</fact>
```

または、以下のようになります。

```
<element>mdx_member_expression</element>
```

以下は、この指定の説明です。

- ・ fact_table_field_name は、レベルまたはメジャーのファクト・テーブルのフィールド名で、そのレベルまたはメジャーの factName 属性によって指定されています。
- ・ mdx_member_expression は、メンバとして評価する MDX 式です。これは、レベルのメンバの場合も、メジャー名の場合もあります（各メジャーは特殊な MEASURES ディメンジョンのメンバです）。

この式は、計算メンバであってはけません。

注釈 各グループは、一連の共通部分を定義します。グループ内の共通部分の数は、キューブ・セルを事前計算するときの処理速度に影響します。

5.4.2 キューブ・セルの事前計算

<group> で指定される集約値を事前計算するには、%DeepSee.Utils の %ComputeAggregateGroup() メソッドを使用します。このメソッドは、以下のとおりです。

```

classmethod %ComputeAggregateGroup(pCubeName As %String,
                                   pGroupName As %String,
                                   pVerbose As %Boolean = 1) as %Status

```

pCubeName はキューブの名前、pGroupName は <group> の名前、pVerbose はメソッドの実行中に進捗情報を書き込むかどうかを指定します。pGroupName については、"*" を使用してそのキューブのすべてのグループを事前計算することもできます。

このメソッドを使用する場合、最初にキューブを構築する必要があります。

このメソッドは、各グループを処理するために、ファクト・テーブルをループし、グループ内の項目によって定義されている共通部分を計算します。処理はグループ内の共通部分の数が少ないほど速くなります。この処理はシングルスレッドです。これによりフォアグラウンドでのクエリが可能になります。

5.5 インデックス圧縮ユーティリティの使用法

キューブが同期によって頻繁に更新されると、インデックスのためのストレージ容量のニーズが大幅に増加します。インデックスのストレージ要件を最小限に抑えるために、インターシステムズは %DeepSee.Utils の一部として %CompressIndices メソッドを提供しています。このメソッドは、以下のとおりです。

```
classmethod %CompressIndices(pCubeName As %String,  
pVerbose As %Boolean = 0) As %Status
```

pCubeName ではキューブ名、pVerbose ではメソッドの実行中に情報を書き込むかどうかを指定します。

5.6 バックグラウンド・タスクのワーカの割り当ての制限

ユーザは、%SetAgentCount メソッドを介して、バックグラウンド・タスクの特定のグループに割り当てられる %SYSTEM.WorkMgr エージェントの数を制限できます。このメソッドは、以下のとおりです。

```
classmethod %SetAgentCount(pNumAgents As %Integer = "", pType = "build", Output pStatus As %Status) As  
%Integer
```

pNumAgents は、特定のタイプのバックグラウンド・タスクに割り当てることができるエージェントの数で、pType は制限を適用するバックグラウンド・タスクのカテゴリです。pType は、既定で build タスクに設定されていますが、runTime に設定することもできます。各タイプの制限は別個に格納され、以下のコマンドを実行することで取得できます。

```
%DeepSee.Utils.%GetAgentCount(type)
```

type は、割り当て可能なエージェントの制限を確認するタスクのカテゴリです。

6

カスタム・アクションの定義

ここでは、Business Intelligence の実装プロセスの一環として、ダッシュボードで使用するカスタム・アクションを定義する方法を説明します。

6.1 概要

KPI クラス内でカスタム・アクションを定義します。次に、以下の操作を行います。

- ・ ウィジェットで指定の KPI を表示すると、カスタム・アクションを呼び出すそのウィジェットにコントロールを追加できます。“ダッシュボードの作成”の“[ウィジェット・コントロールの追加](#)”を参照してください。
- ・ KPI クラスを <cube> 要素の `actionClass` 属性として指定すると、このクラス内のすべてのアクションをそのキューブを使用するピボット・テーブルで使用できるようになります。つまり、それらのピボット・テーブルを表示するウィジェットにアクションをコントロールとして追加することができます。
- ・ KPI クラスを別の <kpi> 要素の `actionClass` 属性として指定すると、その KPI 内で定義されているすべてのアクションに加えて、このクラス内のすべてのアクションをその KPI で使用できるようになります。
- ・ アナライザ内からアクションを実行できます。この場合、クライアント側のコマンドのサブセット (`alert`、`navigate`、および `newWindow`) のみサポートされます。その他のコマンドは無視されます。

KPI の定義の詳細は、“[InterSystems Business Intelligence の上級モデリング](#)”を参照してください。

同じ操作の多くは、以下のように標準アクションまたはカスタム・アクションのいずれかで実行できます。

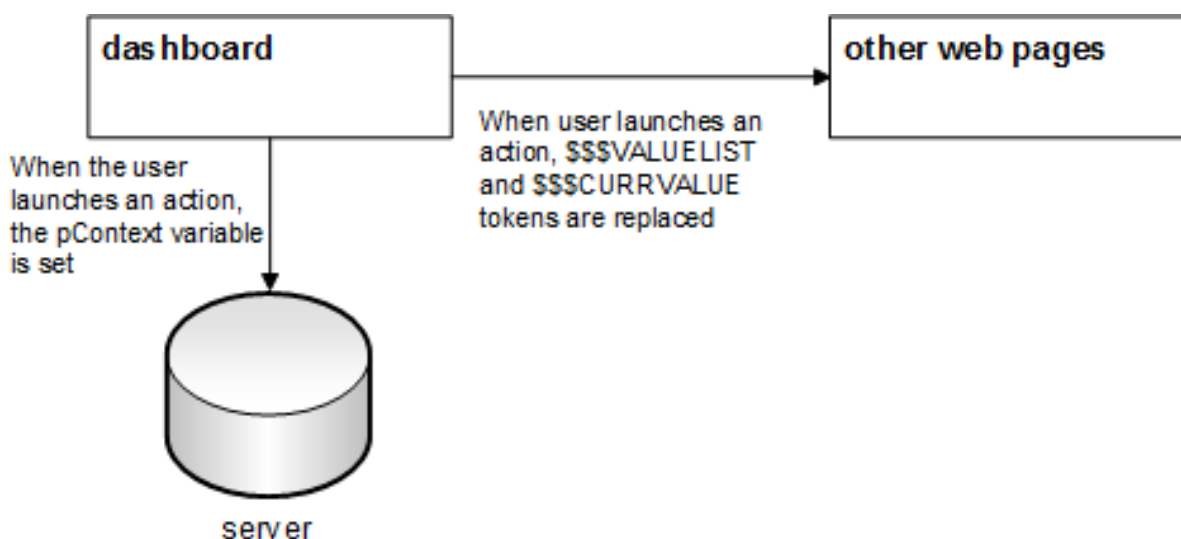
演算	標準アクションとして 使用可能か	カスタム・アクションで 実行可能か
フィルタの設定	はい	はい
フィルタの設定と表示の更新	はい	はい
ウィジェットの表示の更新	はい	はい
ダッシュボード全体の表示の更新	はい	いいえ
ピボット・テーブルの行または列の並べ替えの指定	はい	いいえ
ピボット・テーブルの行数または列数の指定	はい	いいえ
ピボット・テーブルのリストの表示	はい	いいえ

演算	標準アクションとして 使用可能か	カスタム・アクションで 実行可能か
別のダッシュボードの表示	はい	はい
同じページでの URL の表示	はい	はい
新しいページでの URL の表示	いいえ	はい
サーバでのコードの実行	いいえ	はい
ウィジェットのデータ・ソースの変更	はい	いいえ
ウィジェットの行仕様または列仕様の変更	はい	いいえ

標準アクションの詳細は、“ダッシュボードの作成”の“[ウィジェット・コントロールの追加](#)”を参照してください。

6.1.1 コンテキスト情報

システムは、2 つの異なるメカニズムによって、アクションでコンテキスト情報を利用できるようにします。ユーザがカスタム・アクションを起動すると、システムはコンテキスト情報を pContext 変数に書き込みます。この変数は、サーバ上のカスタム・コードで使用できます。カスタム・アクションが URL を開くとき、その URL に \$\$\$VALUELIST および \$\$\$CURRVALUE トークンが含まれていれば、システムはそれらのトークンを置換します。以下の図は、これらのメカニズムを示しています。



6.2 アクションの動作の定義

カスタム・アクションを定義するには、[アクションを宣言](#)して[動作を定義](#)する必要があります。

6.2.1 アクションの宣言

アクションを宣言するには、KPI クラスで以下のタスクのどちらかか両方を実行します。

- ・ <kpi> 要素内に、アクションごとに <action> 要素を 1 つずつ含めます。

この要素では、この KPI クラス内で使用可能なアクションの名前を指定します。ユーザ・インタフェースでは、この情報を使用して、ユーザが使用できるアクションの一覧が作成されます。以下はその例です。

```
<kpi xmlns="http://www.intersystems.com/deepsee/kpi"
  name="Holefoods Actions">

<action name="ActionA"/>
<action name="ActionB"/>
<action name="ActionC"/>
</kpi>
```

`<action>` の詳細は、“InterSystems Business Intelligence の上級モデリング” の “[KPI クラスおよびプラグイン・クラスのリファレンス情報](#)” を参照してください。

- ・ KPI クラスの `%OnGetActionList()` コールバック・メソッドをオーバーライドします。このメソッドには、以下のシグニチャがあります。

```
ClassMethod %OnGetActionList(ByRef pActions As %List, pDataSourceName As %String = "") As %Status
```

`pActions` は以下のノードの配列です。

ノード	値
<code>pActions</code>	アクションの数
<code>pActions(n)</code>	n 番目のアクションに関する詳細。これは、以下の項目で構成される \$LISTBUILD リストです。 <ul style="list-style-type: none"> – 論理アクション名に等しい文字列 – 対応する表示名に等しい文字列

`pDataSourceName` は、将来使用するためのものです。

以下はその例です。

```
ClassMethod %OnGetActionList(ByRef pActions As %List, pDataSourceName As %String = "") As %Status
{
  set newaction=$LB("New Action","New Action Display Name")
  set pActions($I(pFilters))=newaction
  quit $$$OK
}
```

6.2.2 アクションの動作の定義

アクションの動作を定義するには、KPI クラスの `%OnDashboardAction()` コールバック・メソッドをオーバーライドします。このメソッドには、以下のシグニチャがあります。

```
classmethod %OnDashboardAction(pAction As %String, pContext As %ZEN.proxyObject) as %Status
```

このコールバックは、ユーザがダッシュボードでアクションを呼び出したときに実行されます。`pAction` は、キューブの論理名です。`pContext` は、現在選択されているスコアカード行に関する[情報](#)を含むオブジェクトであり、メソッドがダッシュボードに[コマンド](#)を返す方法を提供します。以降のセクションで詳細を説明します。

簡単な例を以下に示します。

Class Member

```

ClassMethod %OnDashboardAction(pAction As %String, pContext As %ZEN.proxyObject) As %Status
{
    Set sc = $$$OK
    Try {
        If (pAction = "Action 1") {
            //this part defines Action 1
            //perform server-side actions
        }
        ElseIf (pAction="Action 2")
        {
            //this part defines Action 2
            //perform other server-side actions
        }
    }
    Catch(ex) {
        Set sc = ex.AsStatus()
    }
}
Quit sc
}

```

このメソッドでは、Action 1 と Action 2 の 2 つのアクションを定義します。

注釈 %OnDashboardAction() はクラス・メソッドであるため、このメソッドから %seriesNames や KPI クラスのその他のプロパティにはアクセスできません (クラス・インスタンスは、メソッドからは利用できません)。

6.3 使用可能なコンテキスト情報

アクションでは、コンテキスト情報 (アクションを起動する前にユーザが選択した行に基づくダッシュボードからの値) を使用できます。これらの値は、コンテキストに依存するデータベースに変更を加えたい場合に便利です。

%OnDashboardAction() はクラス・メソッドであるため、このメソッドから %seriesNames や KPI クラスのその他のプロパティにはアクセスできません。その代わりに、pContext 変数が用意されています。この変数は、当該操作で使用するための情報を提供するプロパティを持つオブジェクトです。詳細は、以下のシナリオごとに異なります。

- ・ [ピボット・テーブルをデータ・ソースとして使用するピボット・テーブル・ウィジェット](#)
- ・ [KPI をデータ・ソースとして使用するピボット・テーブル](#)
- ・ [ピボット・テーブルまたは KPI をデータ・ソースとして使用するスコアカード](#)

6.3.1 シナリオ : ピボット・テーブルをデータ・ソースとして使用するピボット・テーブル・ウィジェット

このシナリオでは、%OnDashboardAction() メソッド内の pContext オブジェクトに、以下の表で説明しているプロパティが格納されます。説明されているように、pContext オブジェクトのコンテンツは、ウィジェットにピボット・テーブル自体が表示される (ピボット・モード) か、リストが表示される (リスト・モード) かによって異なります。

プロパティ名	ピボット・モードのコンテンツ	リスト・モードのコンテンツ
currValue	最初に選択されているセルの値	リストの表示前に表示されていた最初の選択セルの値
currSeriesNo	列番号	リストの表示前に表示されていた最初の選択セルの列番号
currItemNo	行番号	NULL
currFilterSpec	現在のセル・コンテキストに適用されるフィルタ処理を表す 1 つまたは複数の MDX %FILTER 句。これには、すべてのフィルタ・コントロールの値、および行と列のコンテキストが含まれます。	NULL
valueList	NULL	リストの 1 列目の値のコンマ区切りリスト (これらの値にコンマが含まれてはいけません)
cubeName	このピボット・テーブルによるクエリの対象になるキューブ名	NULL
mdx	このピボット・テーブルで定義された MDX クエリ。	NULL
pivotVariables	ピボット変数ごとに 1 つのプロパティが含まれているプロキシ・オブジェクト。具体的には、pContext.pivotVariables.varname にはピボット変数 varname の値が含まれています。このプロキシ・オブジェクトでは、すべてのピボット変数名は小文字で表記されます。例えば、サーバ側で MyVar というピボット変数が定義されている場合、このピボット変数は pContext.pivotVariables.myvar として使用できます。	ピボット・モードと同じ
filters	現在アクティブなフィルタ・コントロールの現在値を示す配列。配列内の各ノードの添え字は、フィルタの MDX 式です。ノードの値は、それに対応する 1 つ以上のキーです。キーの形式については、“ 指定可能なフィルタの既定値 ”を参照してください。	ピボット・モードと同じ
dataSource	現在のデータ・ソースの名前。	現在のデータ・ソースの名前。

6.3.2 シナリオ : KPI をデータ・ソースとして使用するピボット・テーブル・ウィジェット

このシナリオでは、%OnDashboardAction() メソッド内の pContext オブジェクトに、以下のテーブルで説明しているプロパティが格納されます。説明されているように、pContext オブジェクトのコンテンツは、ウィジェットにピボット・テーブル自体が表示される (ピボット・モード) か、リストが表示される (リスト・モード) かによって異なります。

プロパティ名	ピボット・モードのコンテンツ	リスト・モードのコンテンツ
currValue	最初に選択されているセルの値	リストの表示前に表示されていた最初の選択セルの値
currSeriesNo	列番号	リストの表示前に表示されていた最初の選択セルの列番号
valueList	NULL	リストの 1 列目の値のコンマ区切りリスト (これらの値にコンマが含まれてはいけません)
pivotVariables	ピボット変数ごとに 1 つのプロパティが含まれているプロキシ・オブジェクト。具体的には、pContext.pivotVariables.varname にはピボット変数 varname の値が含まれています。このプロキシ・オブジェクトでは、すべてのピボット変数名は小文字で表記されます。例えば、サーバ側で MyVar というピボット変数が定義されている場合、このピボット変数は pContext.pivotVariables.myvar として使用できます。	ピボット・モードと同じ
filters	すべてのフィルタ・コントロールの現在値を示す配列。配列内の各ノードは、データ・ソースである KPI で定義されているいずれかのフィルタに対応します。配列内の各ノードの添え字は、KPI 定義クラスで指定されているフィルタの名前です。ノードの値は、そのフィルタで現在選択されている 1 つ以上のキーです。キーの形式については、“ 指定可能なフィルタの既定値 ”を参照してください。フィルタにキーが選択されていない場合、対応するノードの値は NULL です。	ピボット・モードと同じ
dataSource	現在のデータ・ソースの名前。	現在のデータ・ソースの名前。

6.3.3 シナリオ : ピボット・テーブルまたは KPI をデータ・ソースとして使用するスコアカード

スコアカードでは、%OnDashboardAction() メソッド内の pContext オブジェクトのコンテンツは、データ・ソースがピボット・テーブルであるか KPI であるかに関係なくほぼ同じです。以下の表は、スコアカードの pContext のコンテンツを示しており、データ・ソースによって差異がある点について説明しています。

プロパティ名	データ・ソースとしてピボット・テーブルを使用する場合のコンテンツ	データ・ソースとして KPI を使用する場合のコンテンツ
currValue	このスコアカード内で [値列] としてマークされているピボット列の値	このスコアカード内で [値列] としてマークされている KPI プロパティの値
currSeriesNo	行番号	データ・ソースがピボット・テーブルの場合と同じ
valueList	このスコアカード内で [値列] としてマークされているピボット列の値	このスコアカード内で [値列] としてマークされている KPI プロパティの値
pivotVariables	ピボット変数ごとに 1 つのプロパティが含まれているプロキシ・オブジェクト。具体的には、pContext.pivotVariables.varname にはピボット変数 varname の値が含まれています。このプロキシ・オブジェクトでは、すべてのピボット変数名は小文字で表記されます。例えば、サーバ側で MyVar というピボット変数が定義されている場合、このピボット変数は pContext.pivotVariables.myvar として使用できます。	データ・ソースがピボット・テーブルの場合と同じ
filters	現在アクティブなフィルタ・コントロールの現在値を示す配列。配列内の各ノードの添え字は、フィルタの MDX 式です。ノードの値は、それに対応する 1 つ以上のキーです。キーの形式については、“ 指定可能なフィルタの既定値 ”を参照してください。	すべてのフィルタ・コントロールの現在値を示す配列。配列内の各ノードは、データ・ソースである KPI で定義されているいずれかのフィルタに対応します。配列内の各ノードの添え字は、KPI 定義クラスで指定されているフィルタの名前です。ノードの値は、そのフィルタで現在選択されている 1 つ以上のキーです。キーの形式については、“ 指定可能なフィルタの既定値 ”を参照してください。フィルタにキーが選択されていない場合、対応するノードの値は NULL です。
dataSource	現在のデータ・ソースの名前。	データ・ソースがピボット・テーブルの場合と同じ

6.4 クライアント側コマンドの実行

アクションには、制御がダッシュボードに戻ったときに実行するコマンドを含めることができます。そのようなコマンドを含めるには、アクションの定義内に pContext.command プロパティを設定します。以下はその例です。

ObjectScript

```
//this part defines Action 1
//perform server-side actions
//on returning, refresh the widget that is using this KPI
Set pContext.command="refresh;"
```

pContext.command には、以下の形式の文字列を指定して、単一のコマンドを実行します。

```
command1
```

pContext.command には、セミコロンで区切ったコマンドのリストを指定します。

```
command1;command2;command3;...;
```

最後のセミコロンはオプションです。

一部のコマンドは、1 つ以上の引数を受け入れます。それらのコマンドには、command の代わりに command:arg1:arg2:... を使用します。

6.4.1 使用可能なコマンド

pContext.command 内では、以下のコマンドを使用できます。

alert

```
alert:message
```

メッセージがダイアログ・ボックスに表示されます。message は表示されるメッセージです。

以下はその例です。

ObjectScript

```
Set pContext.command = "alert:hello world!"
```

applyFilter

```
applyFilter:target:filterSpec
```

引数の詳細は、“[applyFilter](#) と [setFilter](#) の詳細”を参照してください。

このコマンドは、指定されたフィルタを設定し、ブラウザ・ウィンドウを更新します。

例えば、以下のように、ピボット・テーブルにフィルタを適用します。

ObjectScript

```
Set pContext.command = "applyFilter:samplepivot:[DateOfSale].[Actual].[YearSold]:&[2008]"
```

navigate

```
navigate:url
```

url は、表示する URL です。

このコマンドは、指定された URL を同じブラウザ・ウィンドウ内で開きます。

以下はその例です。

ObjectScript

```
Set pContext.command = "navigate:http://www.google.com"
```

newWindow

```
newWindow:url
```

url は、表示する URL です。

このコマンドは、指定された URL を新しいブラウザ・ウィンドウ内で開きます。

以下はその例です。

ObjectScript

```
Set pContext.command = "newWindow:http://www.google.com"
```

popup

```
popup:popupurl
```

popupurl は、ポップアップ・ウィンドウの相対 URL です。

このコマンドにより、指定されたポップアップ・ウィンドウが表示されます。以下はその例です。

ObjectScript

```
Set pContext.command = "popup:%ZEN.Dialog.fileSelect.cls"
```

refresh

```
refresh:widgetname
```

widgetname は更新するウィジェットのオプションの名前です。この引数を省略すると、コマンドはユーザがアクションを起動したウィジェットを更新します。

このコマンドは、ブラウザ・ウィンドウの更新に、フィルタの現在の設定を使用します。

以下はその例です。

ObjectScript

```
// Refresh the widget that fired this action and another named samplepivot.  
Set pContext.command = "refresh;refresh:samplepivot"
```

この例では複数のコマンドがセミコロンで区切られています。

setFilter

```
setFilter:target:filterSpec
```

引数の詳細は、“[applyFilter](#) と [setFilter](#) の詳細”を参照してください。

このコマンドは、指定されたフィルタを設定しますが、ブラウザ・ウィンドウは更新しません。

6.4.2 applyFilter と setFilter の詳細

applyFilter および setFilter コマンドはそれぞれ以下のとおりです。

```
applyFilter:target:filterSpec
```

および、

```
setFilter:target:filterSpec
```

以下は、この指定の説明です。

- target は、フィルタ処理するウィジェットです。アスタリスク (*) を使用すると、すべてのウィジェットにフィルタを適用できます。

- ・ filterSpec では、指定のターゲットに適用するフィルタ値を指定します。これは、以下の形式にする必要があります。

filter_name:filter_values

引数は、以下のようにターゲット・ウィジェットの詳細によって異なります。

シナリオ	filter_name	filter_values
ターゲット・ウィジェットでピボット・テーブルを表示する	[dimension].[hierarchy].[level]	“設定の構成”の“指定可能なフィルタの既定値”を参照してください。
ターゲット・ウィジェットで KPI を表示する	その KPI に定義されているフィルタ名	このフィルタに使用できる値のいずれか (KPI で定義されているものと同じ)

メモ：

- 複数のフィルタ仕様を使用できます。そのためには、チルダ(~)でそれぞれを区切ります。以下はその例です。

`FILTER:filterspec1~filterspec2`

- MDX クエリを使用する KPI やピボット・テーブルの場合、フィルタ名とフィルタ値では大文字と小文字は区別されません。
- このフィルタが作用するウィジェットは、同じフィルタを使用するフィルタ・コントロール (非表示の場合もあります) で構成されているウィジェットのみです。例えば、ウィジェットに Cities フィルタ・コントロールが含まれていて、その他のフィルタ・コントロールが含まれていないとします。アクションが市町村に対するフィルタ処理の場合、そのウィジェットは更新されます。アクションが郵便番号に対するフィルタ処理の場合、そのウィジェットは更新されません。

6.5 別のダッシュボードの表示

カスタム・アクションでは、navigate または newWindow を使用して別のダッシュボードを表示できます。これには、ダッシュボードの URL を使用します (“アプリケーションからダッシュボードへのアクセス”を参照してください)。この URL には、ダッシュボードの状態を初期化する `SETTINGS` キーワードを含めることができます。

6.6 キューブ・コンテキストからの SQL テーブルの生成

カスタム・アクションでは、キューブ・コンテキストから SQL テーブルを作成する %CreateTable API を使用できます。テーブルは、以下のいずれかから作成されます。

1. フィールド・リスト
2. キューブで定義されたリスト (フィールド・リストまたはカスタム SQL クエリ) の名前

詳細は、クラスリファレンスを参照してください。

7

アプリケーションからダッシュボードへのアクセス

ここでは、アプリケーションから InterSystems IRIS® Business Intelligence ダッシュボードとユーザ・ポータルにアクセスする方法を説明します。これらの接続は、Business Intelligence の [実装プロセス](#) の一環として確立します。

7.1 ダッシュボードへのアクセス

ダッシュボードにアクセスするには、以下の形式の URL を使用します。

```
http://localhost:52773/csp/samples/_DeepSee.UserPortal.DashboardViewer.zen?DASHBOARD=dashbdname.dashboard
```

localhost:52773 は InterSystems IRIS® が動作しているサーバおよびポート、samples はダッシュボードが定義されているネームスペース、dashbdname はダッシュボードの名前です (ダッシュボードが属しているフォルダがあれば、その名前を含みます)。

より一般的には、以下の形式の URL を使用します。

```
http://localhost:52773/csp/samples/_DeepSee.UserPortal.DashboardViewer.zen?parmstring&parmstring&parmstring...
```

parmstring はパラメータです。この後に等号記号と値が順に続きます。以下に例を示します。

```
DASHBOARD=Drill%20Options.dashboard
```

前に示したように、アンパサンド (&) を使用して、複数のパラメータ文字列を結合します。以下はその例です。

```
DASHBOARD=Drill%20Options.dashboard&NOMODIFY=1
```

7.1.1 URL のエンコード

特定の文字が URL を表すために予約されていますが、その他の文字は許可されていません。これに該当する文字を parmstring に含めるには、その文字を URL エンコードされたバージョンの文字に置き換えます (このエンコードはパーセント・エンコードと呼ぶこともあります)。以下の方法で、この処理を最も容易に実現できます。

1. 予約されている文字または許可されていない文字を含む可能性のあるすべての文字列を特定します。
2. これに該当する文字列ごとに、以下を順番に実行します。
 - a. UTF-8 エンコードへの変換

- b. URL エンコードされたバージョンの文字列を作成します。

これらの変換をサーバ上で実行している場合、\$ZCONVERT や \$TRANSLATE などの ObjectScript 関数を使用できます。以下に例を示します。

```
set UTF8db=$ZCONVERT(dashboardname,"O","UTF8")
set escapeddb=$ZCONVERT(UTF8db,"O","URL")
set url=baseurl_"DASHBOARD="_"escapeddb
```

これらの変換をクライアント上で実行している場合、適切なクライアント関数を使用してください。例えば、クライアントが JavaScript を使用する場合は、[encodeURIComponent](#) 関数を使用します。

または、\$TRANSLATE 関数などの他のロジックを使用します。頻繁に使用される文字のいくつかを以下に示します。

文字	URL エンコードされたバージョン
スペース文字	%20
&	%26
,	%2C

URL エンコードされた文字のリストはインターネット上で確認できます。1 つのリソースとして Wikipedia (<https://en.wikipedia.org/wiki/Percent-encoding>) が挙げられます。

7.2 使用可能な URL パラメータ

ダッシュボードの URL 内では、以下のパラメータ (大文字と小文字が区別されます) を使用できます。一部のパラメータには、平文のバージョンか、暗号化されたバージョンのどちらかを使用できます。例えば、ダッシュボードの URL には、暗号化されたバージョンのダッシュボード名を含めることができます。

DASHBOARD

```
DASHBOARD=dashbdname.dashboard
```

このパラメータは、表示するダッシュボードを指定します。このパラメータまたは XDASHBOARD パラメータを指定する必要があります。

dashbdname はダッシュボードの名前です (ダッシュボードが属しているフォルダがあれば、その名前を含みます)。以下はその例です。

```
DASHBOARD=Dashboards/Dashboard%20with%20Filters%20and%20Listing%20Button.dashboard
```

この %20 は、スペース文字を表します。前述の ["URL のエンコード"](#) を参照してください。

XDASHBOARD

```
XDASHBOARD=encryptedvalue
```

暗号化されたバージョンの DASHBOARD パラメータです。パラメータは、Web セッションのコンテキスト内でのみ使用できます。このパラメータまたは DASHBOARD パラメータを指定する必要があります。

encryptedvalue を作成するには、最初に、ダッシュボード名を用意します (このダッシュボードが属するフォルダがある場合は、そのフォルダを含む)。以下はその例です。

```
Dashboards/Dashboard with Filters and Listing Button.dashboard
```

URL エスケープ表記を含めてはいけません。例えば、スペースは、スペース文字のまま残します。

その後で、**%CSP.Page** の Encrypt() クラス・メソッドを使用して、この値を暗号化します。Encrypt() から返された値を、XDASHBOARD パラメータの値として使用します。

EMBED

```
EMBED=1
```

このパラメータが 1 の場合、ダッシュボードは埋め込みモードで表示されます。これは、NOTITLE=1、NOMODIFY=1、NOBORDER=1、および WORKLISTS=0 に設定する場合と同じです。

XEMBED

```
XEMBED=encryptedvalue
```

暗号化されたバージョンの EMBED パラメータです。パラメータは、Web セッションのコンテキスト内でのみ使用できます。

encryptedvalue を作成するには、最初に、EMBED に使用する値を用意します。その後で、**%CSP.Page** の Encrypt() クラス・メソッドを使用して、この値を暗号化します。Encrypt() から返された値を、XEMBED パラメータの値として使用します。

NOTITLE

```
NOTITLE=1
```

このパラメータが 1 の場合、ダッシュボードはタイトル領域なしで表示されます。タイトル領域は、以下の例のように最上部の領域です。

```
Menu           Home | Save | Logout           User: UnknownUser           Licensed to: InterSystems Sales Engineers           Patients Sample
```

NOMODIFY

```
NOMODIFY=1
```

このパラメータが 1 の場合、ダッシュボードは変更できません。このオプションによって、[メニュー] から項目が削除されます。また、ウィジェットの編集オプションも抑制されるため、ウィジェットには最小化、最大化、および削除のオプションのみが右上に表示されます。

NOBORDER

```
NOBORDER=1
```

このパラメータが 1 の場合、ダッシュボードは境界線なしで表示されます。

RESIZE

```
RESIZE=boolean
```

ウィジェットのサイズ変更と移動が可能かどうかを指定します。boolean が 1 (既定) の場合、ウィジェットのサイズ変更と移動が可能です。boolean が 0 の場合、これらは実行できません。

WORKLISTS

`WORKLISTS=n`

`n` は 0、1、または 2 です。このパラメータは、左側に表示するワークリスト領域の数を指定します。

XWORKLISTS

`XWORKLISTS=encryptedvalue`

暗号化されたバージョンの WORKLISTS パラメータです。パラメータは、Web セッションのコンテキスト内でのみ使用できます。

`encryptedvalue` を作成するには、最初に、WORKLISTS に使用する値を用意します。その後で、`%CSP.Page` の `Encrypt()` クラス・メソッドを使用して、この値を暗号化します。`Encrypt()` から返された値を、XWORKLISTS パラメータの値として使用します。

SCHEME

`SCHEME=schemename`

ダッシュボードの配色を指定します (既定値を使用しない場合)。`schemename` には、[設定] ページの [一般] タブにリストされている配色を指定します。“[基本設定の指定](#)”を参照してください。

SETTINGS

`SETTINGS=name1:value1;name2:value2;name3:value3;...;`

`name1`、`name2`、`name3` (以降同様) は、次のセクションで説明するように、ダッシュボードの設定の名前です。また、`value1`、`value2`、`value3` (以降同様) は、その設定の値です。

このパラメータは、URL に複数回含めることができます。

例えば、ダッシュボード内の特定のウィジェットに値を渡すには、以下のバリエーションを使用します。

`basic_dashboard_url&SETTINGS=TARGET:widgetname;name:value;name:value;name:value;...;`

ダッシュボード内のすべてのウィジェットに値を渡すには、以下の形式の URL を使用します。前の例で使った `TARGET` パラメータが除外されていることに注意してください。

`basic_dashboard_url&SETTINGS=name:value;name:value;name:value;...;`

ダッシュボード内の複数のウィジェットに値を渡すには、以下のバリエーションを使用します。

`basic_dashboard_url&SETTINGS=...;&SETTINGS=...;&SETTINGS=...;...;`

特定のウィジェットに対する設定は、常にすべてのウィジェットに対する設定よりも優先されます。それ以外の場合、設定は指定された順に適用されます。ある設定がもう 1 つの設定と矛盾する場合、後に指定された設定が適用されます。これらの設定がユーザ設定よりも優先されることはありません。

XSETTINGS

`XSETTINGS=encryptedvalue`

暗号化されたバージョンの SETTINGS パラメータです。パラメータは、Web セッションのコンテキスト内でのみ使用できます。

encryptedvalue を作成するには、最初に、SETTINGS に使用する値を用意します。その後で、%CSP.Page の Encrypt() クラス・メソッドを使用して、この値を暗号化します。Encrypt() から返された値を、XSETTINGS パラメータの値として使用します。

IRISUsername と IRISPassword

```
IRISUsername=myuser&IRISPassword=mypass
```

myuser は InterSystems IRIS のユーザ名、mypass は対応するパスワードです。ユーザがまだ InterSystems IRIS にログインしていない場合は、これらのパラメータを含めます。

AUTOSAVE

```
AUTOSAVE
```

ダッシュボードの自動保存されたバージョンを要求します。自動保存機能については、“[基本設定の指定](#)”を参照してください。

7.3 SETTINGS パラメータのオプション

SETTINGS URL パラメータには、以下のリストに記載された設定を使用できます。SETTINGS 文字列はすべてのウィジェットに適用されるか、特定のウィジェットに適用されます。SETTINGS 文字列は必要なだけ含めます。以下はその例です。

```
basic_dashboard_url&SETTINGS=...;&SETTINGS=...;&SETTINGS=...;...
```

注釈 InterSystems IRIS が SETTINGS パラメータを解析する際、すべてのセミコロンは 2 つの設定文字列の区切り記号と見なされます。セミコロンを区切り文字として扱われないように含めるには、そのセミコロンを %3B%3B に置き換えます (このシーケンスは、URL エンコードされた 2 つのセミコロンです。URL エンコードされた 2 つのセミコロンの使用は、解析処理の実行方法のために必要になります)。

TARGET

```
TARGET:widgetname
```

この一連の設定を適用するウィジェットを指定します。すべてのウィジェットに適用する設定が必要な場合、このパラメータは SETTINGS 文字列で省略します。

FILTER

```
FILTER:filter_name.filter_values
```

指定のウィジェットをフィルタ処理する方法を指定します。引数は、以下のようにターゲット・ウィジェットの詳細によって異なります。

シナリオ	filter_name	filter_values
ターゲット・ウィジェットでピボット・テーブルを表示する	[dimension].[hierarchy].[level] の URL エンコード・バージョン	“設定の構成”の“指定可能なフィルタの既定値”に示されている、指定可能なフィルタ値の URL エンコードされたバージョン
ターゲット・ウィジェットで KPI を表示する	対象の KPI に定義されているフィルタ名の URL エンコードされたバージョン	このフィルタに対して KPI で定義している指定可能値の URL エンコードされたバージョン

URL エンコードされた文字列の作成の詳細は、“[URL のエンコード](#)”を参照してください。

メモ：

- filter_name.filter_value の代わりに特殊なトークン \$\$\$FILTERS を使用できます。これは、URL をカスタム・ナビゲート・アクション (特定のダッシュボードから別のダッシュボードにアクセスします。“[別のダッシュボードの表示](#)”を参照してください) で使用する場合に便利です。このシナリオでは、\$\$\$FILTERS は元のダッシュボードの現在のフィルタ値によって置換されます。以下に例を示します。

```
FILTER:$$$FILTERS
```

ターゲット・ダッシュボードに同じフィルタが含まれます。

- 複数のフィルタ仕様を使用できます。そのためには、チルダ (~) でそれぞれを区切ります。以下はその例です。

```
FILTER:filterspec1~filterspec2
```

各 filterspec は filter_name.filter_values です。

- 同じフィルタの複数のメンバをまとめて使用するには、それらのメンバをリストするセット式を使用します。“[設定の構成](#)”の“[指定可能なフィルタの既定値](#)”を参照してください (SETTINGS 文字列に同じフィルタを複数回指定すると、最後に指定した値が使用されます。多くの場合、これは目的の動作ではありません)。

TARGET パラメータを指定せずに SETTINGS 文字列で FILTER パラメータを渡すと、“ディメンジョンが見つかりません”というエラーが発生する場合があります。これは、特定のウィジェットが、フィルタ処理の対象となるディメンジョンが欠如しているキューブに基づいているためです。

VARIABLE

```
VARIABLE:variable_name.variable_value
```

指定のピボット変数の値を指定します。ピボット変数の詳細は、“[アナライザの使用法](#)”の“[ピボット変数の定義と使用](#)”を参照してください。

variable_name.variable_value の代わりに特殊なトークン \$\$\$VARIABLES を使用できます。これは、URL をカスタム・ナビゲート・アクション (特定のダッシュボードから別のダッシュボードにアクセスします。“[別のダッシュボードの表示](#)”を参照してください) で使用する場合に便利です。このシナリオでは、\$\$\$VARIABLES は、元のダッシュボードで指定されている、指定されたピボット変数の現在の値に置換されます。以下はその例です。

```
VARIABLE:$$$VARIABLES
```

ROWCOUNT

```
ROWCOUNT:n
```

表示する行の最大数 (n) を指定します。これは、メンバが行として表示される場合にのみ適用されます。

COLCOUNT

COLCOUNT:n

表示する列の最大数 (n) を指定します。これは、メンバが列として表示される場合にのみ適用されます。

ROWSORT

ROWSORT:measure

行の並べ替えの基準にするメジャーを指定します。measure は、メジャーの MDX 識別子です。以下はその例です。

ROWSORT:[MEASURES].[mymeasure]

これらの識別子の角括弧は省略できません (Business Intelligence でのその他の MDX の使用法とは異なります)。

COLSORT

COLSORT:[MEASURES].[my measure]

列の並べ替えの基準にするメジャーを指定します。measure は、メジャーの MDX 識別子です。"ROWSORT" を参照してください。

ROWSORTDIR

ROWSORTDIR:sortkeyword

行の並べ替え方法を指定します。sortkeyword は以下のいずれかになります。

- ・ ASC – 昇順で並べ替えられますが、すべての階層は保持されます。
- ・ DESC – 降順で並べ替えられますが、すべての階層は保持されます。
- ・ BASC – 昇順で並べ替えられ、階層は保持されません。
- ・ BDESC – 降順で並べ替えられ、階層は保持されません。

COLSORTDIR

COLSORTDIR:sortkeyword

列の並べ替え方法を指定します。ROWSORTDIR を参照してください。

PORTLET

PORTLET:portlet_setting.value

ポートレット設定の値を指定します。これにより、その設定のすべての構成済み値が指定変更されます。他の SETTINGS オプションと同様に、この設定は、TARGET パラメータで指定されているすべてのウィジェットに適用されます (または、TARGET が指定されていない場合はすべてのポートレット・ウィジェットに適用されます)。

ここで、portlet_setting にはそのポートレットで定義されている設定の名前を指定し、value にはこの設定の許容値の URL エンコード・バージョンを指定する必要があります。URL エンコードされた文字列の作成の詳細は、このページで前述した "[URL のエンコード](#)" を参照してください。

複数のポートレット仕様を使用するには、それぞれの仕様をチルダ (~) で区切ります。次に例を示します。

```
PORTLET:portletspec1~portletspec2
```

各 portletspec は portlet_setting.value です。

ポートレットの定義については、“[ダッシュボードで使用するポートレットの作成](#)” を参照してください。

例を表示するには、[ウィジェットの例/カスタム・ポートレット] というダッシュボードを表示してから (丸い時計が表示されます)、次の文字列をブラウザ上の URL の末尾に追加します。

```
&SETTINGS=PORTLET:CIRCLE.0~SIZE.200
```

次に、**Enter** キーを押します。時計が四角形に変化して、以前よりもやや大きくなることを確認します。

例えば、以下はほとんどのウィジェットに対して列数を 3 に制限しますが、ウィジェット RegionVsYear に対しては列数を 1 に制限します。

```
&SETTINGS=TARGET:RegionVsYear;COLCOUNT:1;&SETTINGS=COLCOUNT:3;
```

注釈 これらの設定は、カスタム・ウィジェットやカスタム・コントロールではサポートされません。

7.4 アプリケーションから他の Business Intelligence ページへのアクセス

アプリケーションで、アナライザやユーザ・ポータルなどの他の Business Intelligence Web ページへの直接リンクを提供することもできます。

Business Intelligence Web ページの URL は、以下のような一般的な構造になります。

```
http://localhost:52773/csp/samples/_Package.Class
```

localhost:52773 は InterSystems IRIS が実行されているサーバとポート、samples は Business Intelligence が実行されているネームスペース、_Package.Class はページを定義するクラスとパッケージの名前で、パッケージ名の先頭はパーセント記号ではなくアンダースコアにします。アナライザや他の Business Intelligence ページにアクセスすると、この URL がツールバーまたはブラウザに表示されます。

これらのページで、対応する任意の URL パラメータを使用できます。このページで前述の“[使用可能な URL パラメータ](#)”を参照してください。アナライザの URL を使用する際は、表示するピボット・テーブルを指定する PIVOT URL パラメータも指定できます。次に例を示します。

```
http://localhost:52773/csp/samples/_DeepSee.UI.Analyzer.zen?PIVOT=Pivot%20Features%2FConditional%20Formatting.pivot
```

アナライザの URL を使用する場合に、AUTOSAVE URL パラメータを指定する一方で PIVOT パラメータを指定しない場合は、アナライザには一番最近に表示されたアイテムが表示されます。

8

キューブの最新状態の維持

ここでは、Business Intelligence の実装プロセスで必要に応じて、キューブ・マネージャなどのツールを使用してキューブを更新する方法を説明します。

“[キューブ・バージョンの使用](#)” も参照してください。

8.1 概要

一般的なフレーズ「キューブの更新」は、ソース・テーブルおよび関連テーブルの現在の内容をキューブに反映させるプロセスを表します。以下の 3 つの手法が使用できます。

- ・ キューブを再構築します (例えば、アーキテクトで **[構築]** オプションを使用)。このプロセスは時間がかかる場合があります。また、キューブの再構築中にはクエリを実行できません。
 - また、[選択的構築](#)を使用して、キューブの特定の要素を再構築することもできます。
- ・ キューブを同期します。キューブの同期機能 (DSTIME 機能とも呼ばれる) を使用すると、InterSystems IRIS Business Intelligence でデータへの変更を追跡できます。定期的にキューブを同期し、変更を組み込みます。
同期中にクエリを実行することは可能です。
キューブの実装および変更するデータによっては、この機能を使用できない場合があります。このページで後述する “[キューブの同期が可能なとき](#)” を参照してください。
- ・ キューブを手動で更新します。このプロセスは、**%ProcessFact()** メソッドおよび **%DeleteFact()** メソッドを使用します。他のオプションと異なり、この場合、ファクト・テーブルのどのレコードを更新または削除するかをコードが認識する必要があります。
手動更新中にクエリを実行することは可能です。

これらの方法を適切に組み合わせて使用できます。以下のテーブルでは、上記の方法を比較しています。

	再構築	同期	手動による更新	選択的構築
プロセスにかかる時間の比較	長い	短い	短い	長い
プロセス中にクエリを実行できるか	いいえ	はい	はい	はい (再構築中のキューブ要素はクエリに使用できない)
すべてのシナリオで使用可能か	はい	いいえ	はい	はい
変更するレコードを認識している必要があるか	いいえ	いいえ	はい	いいえ
結果キャッシュの一部が無効になるか	はい	はい	いいえ	はい
このオプションを提供するユーザ・インタフェース	キューブ・マネージャおよびアーキテクト	キューブ・マネージャ	なし	アーキテクト

キューブ・マネージャの詳細は、このページで後述する [“キューブ・マネージャを使用する方法”](#) を参照してください。

8.1.1 キューブの更新および関連キューブ

どのような更新でも、キューブ間のリレーションシップがある場合には必ず、特定の順序でキューブを更新する必要があります。特に、まず独立したキューブを更新します。次に、これに依存するキューブを更新します。これを行うために、キューブ・マネージャを使用できます。これは、リレーションシップを検索し、正しい更新順序を判断します。

または、キューブを適切な順序で構築するユーティリティ・メソッドまたはルーチンを作成して使用できます。

8.1.2 キューブの更新および結果キャッシュ

(既定で) 512,000 レコードより多く使用するキューブの場合、システムは、結果キャッシュを保持して使用します。更新方法とツールのどのような組み合わせでも、キューブの更新の頻度も注意深く考慮する必要があります。これは、どの更新でも結果キャッシュの一部が無効化する可能性があるためです。

大量のデータ・セットの場合、システムは、以下のように各キューブの結果キャッシュを保持して使用します。ユーザが (例えばアナライザを使用して) クエリを実行するたびに、システムはそのクエリの結果をキャッシュします。次回そのクエリを任意のユーザが実行すると、システムはそのキャッシュがまだ有効かどうかをチェックします。有効な場合は、キャッシュの値が使用されます。それ以外の場合、システムはクエリを再実行し、新しい値を使用して、それらをキャッシュします。その実質的な効果は、より多くのユーザがより多くのクエリを実行するほど、経時的にパフォーマンスが向上することです。

同期または再構築することでキューブを更新した場合、有効でなくなった一部の結果キャッシュがシステムによってクリアされます。詳細はキューブ定義のオプションに応じて異なります ([“キャッシュのバケットとファクトの順序”](#) を参照)。したがって、通常、頻繁に更新することは望ましくありません。

注釈 キューブを手動で更新した場合は、結果キャッシュが自動的に無効になることはありません。これは、InterSystems IRIS では、キャッシュされている結果が古くなっているとの判断が、`%ProcessFact()` メソッドによって更新されない `^OBJ.DSTIME` グローバル内のエントリに基づいているためです (`OBJ.DSTIME` は、キューブの更新の自動プロセスでバッファとして機能します。これについては、[次のセクションで説明します](#))。キューブのクエリがキャッシュされている古い結果を返すことのないように、手動更新の後に `%SetCubeDSTime()` メソッドを呼び出す必要があります (キューブ・クラスの `%OnAfterProcessFact()` メソッドでこのメソッドを呼び出すなど)。

8.2 キューブの同期が機能する方法

ここでは、キューブの同期が機能する方法について簡単に説明します。内部的に、この機能には 2 つのグローバル、`^OBJ.DSTIME` と `^DeepSee.Update` を使用しています。

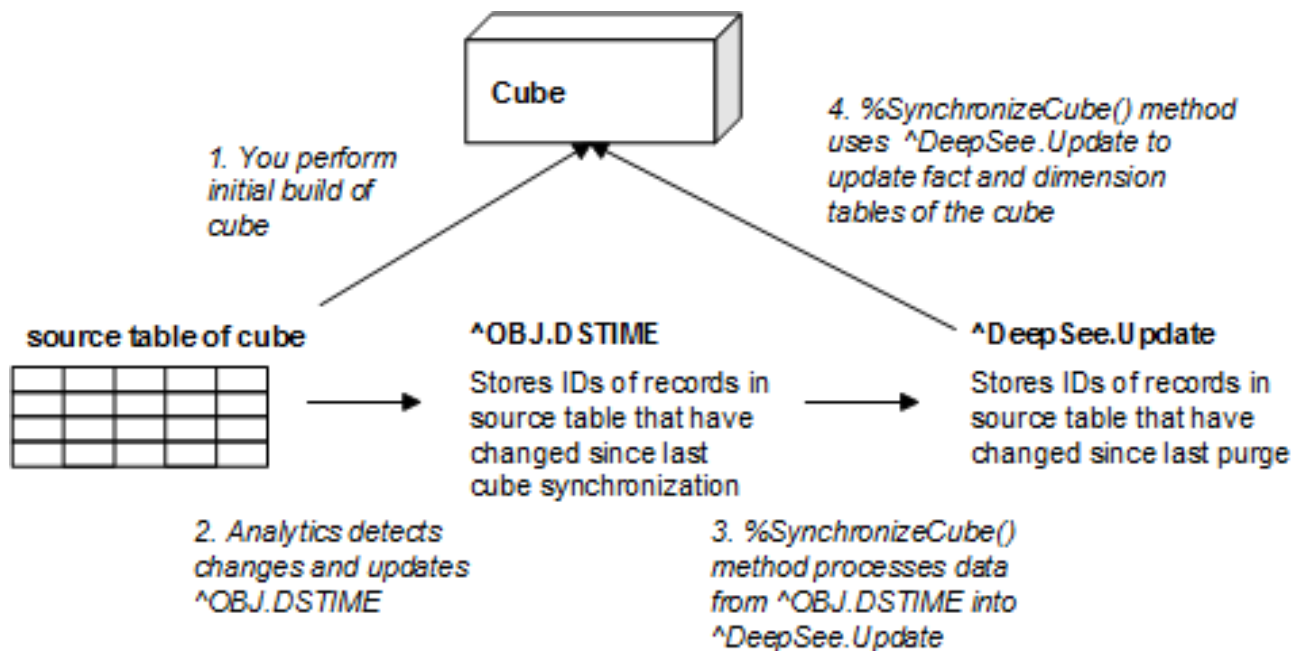
最初に、キューブの初期ビルドを実行する必要があります。

InterSystems IRIS® が、キューブによって使用されるソース・テーブル内で変更を検出すると、エントリを `^OBJ.DSTIME` グローバルに追加します。これらのエントリには、追加、変更、または削除が行われた ID を示す目的があります。

キューブを同期する場合 (このページで後述する `%SynchronizeCube()` を使用)、最初に InterSystems IRIS は `^OBJ.DSTIME` グローバルを読み取り、それを使用して `^DeepSee.Update` グローバルを更新します。ID が `^DeepSee.Update` グローバルに追加された後、InterSystems IRIS は `^OBJ.DSTIME` グローバルから同じ ID を削除します。(以前のバージョンでは、キューブの同期機能は 1 つのグローバルのみ使用していました。新しいシステムでは競合条件を回避します。)

次に、InterSystems IRIS は `^DeepSee.Update` グローバルを使用して、キューブのファクト・テーブルとディメンジョン・テーブルを更新します。これにより、キューブが最新状態になります。

以下の図は、このプロセス・フローを示しています。



サブセクションでは、以下の詳細について説明します。

- ・ キューブの同期機能を使用できるとき
- ・ キューブの同期機能を使用できないとき
- ・ ミラーリング環境におけるキューブの同期
- ・ キューブの同期のグローバルの構造

8.2.1 キューブの同期が可能なとき

キューブ同期機能は、以下のすべての項目に当てはまるシナリオで使用できます。

- ・ キューブのベース・クラスが永続クラスである (ただし、リンク・テーブルではないこと)。

- ・ 変更されたレコードがそのクラス内のレコードである。

8.2.2 キューブの同期が不可能なとき

キューブ同期機能は、以下のシナリオでは使用できません。

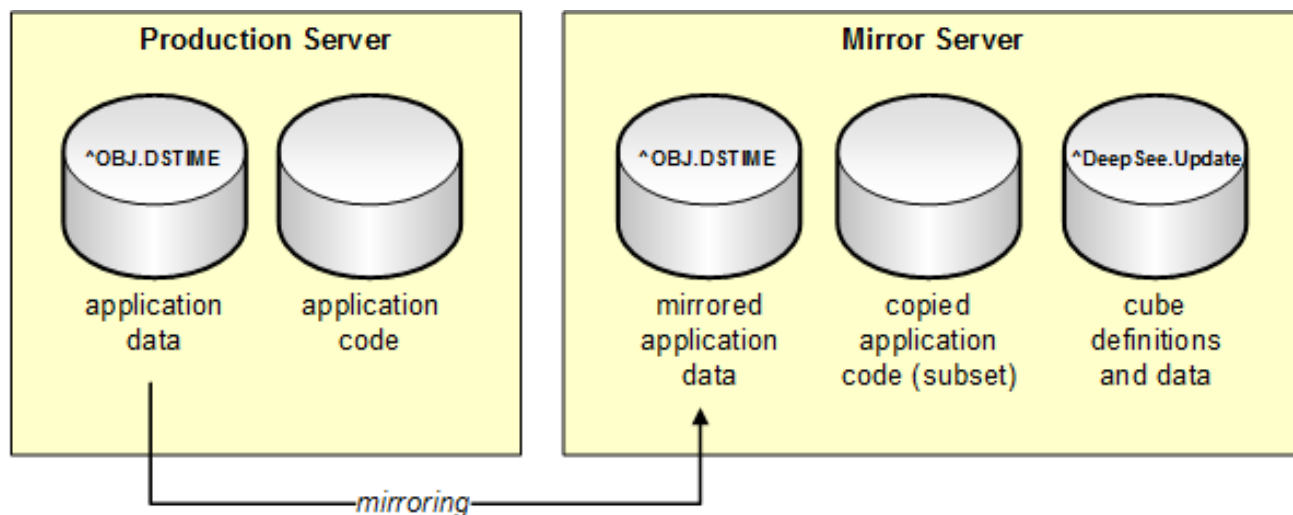
- ・ キューブのベース・クラスがデータ・コネクタである。([“データ・コネクタの定義”](#)を参照してください。)
- ・ キューブのベース・クラスがリンク・テーブルである。([“InterSystems IRIS SQL ゲートウェイの使用法”](#)の[“リンク・テーブル・ウィザード”](#)を参照してください。)
- ・ 変更されたレコードがキューブで使用されるベース・クラスのエクステント内にない。つまり、変更されたレコードが別のテーブルに属している。

これらのシナリオでは、キューブ同期機能によって変更を検出できず、[“キューブの手動更新”](#)の説明に従って、手動でキューブを更新する必要があります。

また、キューブ同期は年齢ディメンジョン（つまり、[\[ディメンジョン・タイプ\]](#)が[\[年齢\]](#)であるディメンジョン）に影響しません。

8.2.3 ミラーリング環境におけるキューブの同期

Business Intelligence をミラー・サーバで使用する場合、`^OBJ.DSTIME` グローバルはアプリケーション・データの一部であり、ミラーリングする必要があることに注意してください（例えば、異なるデータベースにマップされている場合は、そのデータベースをミラーリングする必要があります）。`^DeepSee.Update` グローバルは Business Intelligence コードによって生成されます。そのため、キューブ定義とデータが含まれるデータベースにのみ存在します。



重要 ミラー・サーバでは、`^OBJ.DSTIME` グローバルと `^DeepSee.Update` グローバルを格納しているデータベースを読み取り/書き込みに設定しておく必要があります。これらのグローバルは、同じデータベースに両方とも格納することもできますが、上記の図では個別のデータベースに格納しています。

ミラー・サーバでの Business Intelligence の使用に関する詳細は、[“推奨アーキテクチャ”](#)を参照してください。

8.2.4 キューブの同期のグローバルの構造

ここでは、キューブの同期のグローバルの構造について説明します。この情報は、キューブの同期を使用するのに必要ではありません。しかし、これらのグローバルを他の目的で使用する場合を考慮して提供されています。

8.2.4.1 ^OBJ.DSTIME

DSINTERVAL が設定されているかどうかによって、^OBJ.DSTIME グローバルには別の形式があります。

DSINTERVAL が設定されていない場合、このグローバルには以下のようなノードがあります。

ノード	値
<code>^OBJ.DSTIME(class increment ID)</code> 。ここで、class はソース・クラスの完全なパッケージおよびクラス名です。increment は 0 です。ID は、所定のクラスの新規レコード、変更されたレコード、または削除されたレコードの ID です。	以下の値のいずれかです。 <ul style="list-style-type: none"> ・ 0 (レコードが変更されたことを意味します) ・ 1 (レコードが追加されたことを意味します) ・ 2 (レコードが削除されたことを意味します)

`%SetDSTimeIndex()` メソッドを使用することで、対応するレコードをソース・クラスから削除することなく、ファクト・テーブルからファクトを手動で削除できることに注意してください。

DSINTERVAL が設定されている場合、このグローバルには以下のようなノードがあります。

ノード	値
<code>^OBJ.DSTIME(class timestamp ID)</code> 。ここで、class と ID は他のシナリオの場合と同じです。timestamp は、1840 年 12 月 31 日午前零時以降の秒数です	他のシナリオの場合と同じ

キューブの同期または再ビルドが行われるときに、不要なエントリは ^OBJ.DSTIME グローバルから削除されます。

8.2.4.2 ^DeepSee.Update

^DeepSee.Update グローバルには以下のようなノードがあります。

ノード	値
<code>^DeepSee.Update</code>	使用する次の increment の値を示す整数
<code>^DeepSee.Update(class increment ID)</code> 。ここで、class はソース・クラスの完全なパッケージおよびクラス名です。increment は 0 または正の整数です。ID は、所定のクラスの新規レコード、変更されたレコード、または削除されたレコードの ID です。キューブが同期されるたびに、システムは次に大きい increment 整数値を使用して、このグローバルにノードを更新します。例を参照してください。	^OBJ.DSTIME グローバルの場合と同じ
<code>^DeepSee.Update("cubes" cube "dstime")</code> 。ここで、cube はキューブの論理名です	特定のキューブに対する変更を記録するために、このグローバルでノードを作成するときに使用する increment の次の値を示す整数。
<code>^DeepSee.Update("cubes" cube "lastDataUpdate")</code> 。ここで、cube はキューブの論理名です	このキューブが最後に同期化されたときの日付と時間 (\$H 形式)。

次に例を示します。

```

^DeepSee.Update=3
^DeepSee.Update("DeepSee.Study.Patient",0,1)=0
^DeepSee.Update("DeepSee.Study.Patient",0,2)=0
^DeepSee.Update("DeepSee.Study.Patient",0,100)=0
^DeepSee.Update("DeepSee.Study.Patient",1,1)=2
^DeepSee.Update("DeepSee.Study.Patient",1,120)=0
^DeepSee.Update("DeepSee.Study.Patient",2,42)=0
^DeepSee.Update("DeepSee.Study.Patient",2,43)=0
^DeepSee.Update("DeepSee.Study.Patient",2,50)=0
^DeepSee.Update("DeepSee.Study.Patient",2,57)=0
^DeepSee.Update("cubes","PATIENTS","dstime")=3
^DeepSee.Update("cubes","PATIENTS","lastDataUpdate")="64211,63222.68"

```

`^DeepSee.Update("DeepSee.Study.Patient",0)` の下のノードは、最初の変更セットを表し、
`^DeepSee.Update("DeepSee.Study.Patient",1)` の下のノードは、2 番目の変更セットを表します。その後も同様に続きます。

InterSystems IRIS が `^DeepSee.Update` グローバルからノードを自動的に削除することはありません。このグローバルの削除の詳細は、["DSTIME の削除"](#) を参照してください。

8.3 キューブ同期の有効化

キューブを同期する前に、そのキューブのキューブ同期機能を有効化する必要があります。そのためには、以下の操作を実行します。

1. キューブ同期がシナリオで可能になっていることを確認します。このページで前述の ["キューブの同期が可能なき"](#) を参照してください。
2. 以下のように、DSTIME パラメータをそのキューブで使用されるベース・クラスに追加します。

Class Member

```
Parameter DSTIME = "value";
```

DSTIME パラメータは、3 つの文字列のいずれかをその value として受け入れます。"AUTO" が指定されている場合、`^OBJ.DSTIME` グローバルは、レコードの変更ごとに更新を受け取ります。これは、`%SynchronizeCube()` メソッド ([以下のセクションで説明](#)) を呼び出すと、すべての変更が `^DeepSee.Update` に転写され、対応するキューブと同期されることを意味します。DSTIME が "MANUAL" に設定されている場合、そのベース・クラスのレコードへの変更の自動ジャーナリングは無効になります。値 "CONDITIONAL" を使用すると常に、クラスの DSCONDITION パラメータを指定することによって、`^OBJ.DSTIME` が同期させる変更を記録する条件を動的に指定できます (以下を参照)。

3. DSTIME パラメータを "CONDITIONAL" に設定する場合、以下のパラメータをベース・クラスに追加します。

```
Parameter DSCONDITION = expression
```

指定された expression が TRUE と評価された場合、`^OBJ.DSTIME` は、同期のために、指定のクラスのレコードの更新を受け取ります。DSCONDITION は実行時に実行される式ですが、[通常のように](#)、そのパラメータ・タイプを COSEXPRESSION として明示的に指定する必要はありません。

4. 必要に応じてベース・クラスに以下のパラメータも追加できます。

Class Member

```
Parameter DSINTERVAL = 5;
```


このパラメータは主に、エントリが ^OBJ.DSTIME グローバルに保存される方法に影響を与えます。["キューブの同期のグローバルの構造"](#)を参照してください。^OBJ.DSTIME グローバルの形式は、キューブの同期メカニズムの動作に影響を与えません。

5. ベース・クラスと、そのベース・クラスが使用するすべてのキューブ・クラスをリコンパイルします。
6. それらのキューブをリビルドします。

8.4 ^OBJ.DSTIME グローバルのクリア

このセクションでは、^OBJ.DSTIME グローバルをクリアする方法について説明します。場合によっては、^OBJ.DSTIME グローバルを定期的にクリアできます。例えば、Business Intelligence でキューブを使用していない場合、^OBJ.DSTIME グローバルをクリアして領域を解放できます。

タスク・マネージャで、^OBJ.DSTIME グローバルを定期的にクリアするタスクを設定できます。そのためには、以下のよう
に、OnTask() メソッドで新規タスクを作成します。

```
Method OnTask() As %Status
{
    set classname=$ORDER(^OBJ.DSTIME(""))
    while (classname="") {

        //check to see if this classname is contained in ^DeepSee.Cubes("classes")
        set test=$DATA(^DeepSee.Cubes("classes",classname))

        if (test'=1) {
            kill ^OBJ.DSTIME(classname)
        }
        set classname=$ORDER(^OBJ.DSTIME(classname))
    }
    q $$$OK
}
```

このタスクは、^OBJ.DSTIME エントリが Business Intelligence のキューブによって使用されていない場合、それらのエントリをクリアします。[\[タスクスケジューラウィザード\]](#)を使用して、必要な頻度で実行するようにタスクをスケジュールします。

8.5 キューブ・マネージャを使用する方法

このセクションでは、キューブの更新の管理に役立つように設計されている、キューブ・マネージャにアクセスして使用する方法について説明します。キューブの更新方法と更新時期をキューブ・マネージャで指定します。キューブ・マネージャでは、選択したスケジュール日時にキューブの再構築や同期を実行するタスクを追加できます。このセクションでは、以下の項目について説明します。

- ・ [キューブ・マネージャの概要](#)
- ・ [更新計画の概要](#)
- ・ [キューブ・マネージャへのアクセス方法](#)
- ・ [レジストリ詳細の変更方法](#)
- ・ [キューブ・グループの登録方法](#)
- ・ [更新計画の指定方法](#)
- ・ [登録されたすべてのキューブを構築する方法](#)

- ・ [オンデマンド構築を実行する方法](#)
- ・ [キューブ・グループの登録解除方法](#)
- ・ [キューブ・マネージャ・イベントの表示](#)
- ・ [キューブ・マネージャ UI へのアクセスを制限する方法](#)

注釈 キューブ・マネージャのタスクはタスク・マネージャに表示できます。タスク・マネージャは、“システム管理ガイド”の“タスク・マネージャの使用”で説明しています。どのような方法でもこれらのタスクを変更しないことをお勧めします。

8.5.1 キューブ・マネージャの概要

キューブ・マネージャを使用すると、キューブ・レジストリを定義できます。これには、現在のネームスペース内のキューブに関する情報が含まれています。特に、構築方法または同期方法あるいはこれら両方に関する情報が含まれています。

キューブ・レジストリでは、一連のキューブ・グループが定義されます。キューブ・グループは、関連性があるまたは同時に更新することを選択したために、同時に更新する必要があるキューブの集合です。キューブ・マネージャに初めてアクセスすると、キューブ・グループの初期セットが表示されます。それぞれの初期キューブ・グループは、1 つのキューブまたは相互に関連付けられている（したがって、グループとして更新する必要がある）一連のキューブです。必要に応じて、これらの初期キューブ・グループをマージできます。ただし、どの初期キューブ・グループも分割できません。

各キューブ・グループは初期状態では登録されていません。つまり、キューブ・レジストリに組み込まれていません。キューブ・グループを登録（レジストリに配置）した後、更新計画を定義します。キューブ・マネージャによって、これらの更新計画を使用した自動タスクが作成されます。詳細は、[次のセクション](#)を参照してください。

8.5.2 更新計画の概要

キューブ・グループの更新計画によって、キューブの更新方法と更新時期が指定されます。各グループには、既定の計画があり、これを変更できます。グループ内の特定のキューブに対して異なる更新計画を指定することもできます。どちらの場合も、計画の選択肢は以下のとおりです。

- ・ **[構築と同期]** – 定期的に再構築します。既定では 1 週間に 1 回です。また、定期的に同期します。既定では 1 日 1 回です。

このオプションは、([このページで前述](#)したように) 指定されたキューブが同期をサポートしていない場合、そのキューブではサポートされません。

- ・ **[構築のみ]** – 定期的に再構築します。既定では 1 週間に 1 回です。
- ・ **[同期のみ]** – 定期的にキューブを同期します。既定では 1 日 1 回です。

このオプションは、([このページで前述](#)したように) 指定されたキューブが同期をサポートしていない場合、そのキューブではサポートされません。

重要 キューブをキューブ・マネージャから同期する前に、少なくとも一度、キューブ・マネージャからキューブを構築する必要があります。

- ・ **[手動]** – キューブ・マネージャから再構築または同期しません。

代わりに、他のツール（アーキテクトの **[構築]** オプションおよび %BuildCube()、%SynchronizeCube()、%ProcessFact()、%DeleteFact() の各メソッド）の適切な組み合わせを使用します。後者の 3 つのメソッドについては、このページの後半で説明します。

または、%DeepSee.CubeManager.Utils.BuildCube() の呼び出しによって、手動でキューブを再構築することもできます。さらに、%DeepSee.CubeManager.Utils.BuildCube() の呼び出しによって、手動でキューブを同期することもできます。

各計画 ([手動] 以外) では、スケジュールの詳細をカスタマイズできます。

任意のネームスペースに対して、キューブ・マネージャは、2 つのタスクを定義します。このネームスペースで要求されたキューブ構築アクティビティをすべて実行するタスクと、このネームスペースで要求されたキューブ同期アクティビティをすべて実行するタスクです。これらのタスクは両方とも、キューブ・レジストリで指定された手順に従います。両方のタスクは、リレーションシップによって要求される正しい順序で自動的にキューブを処理します。

キューブ・マネージャには、登録されたグループおよびキューブそれぞれに対して [除外] チェック・ボックスが用意されていて、これを使用して該当するグループまたはキューブをキューブ・マネージャによるアクティビティから除外できます。具体的には、キューブ・マネージャのタスクでは、除外されたグループおよびキューブは無視されます。最初、これらのチェック・ボックスは選択されています。これは、通常、更新の準備ができるまで、これを実行しないことが適切であるためです。

8.5.3 キューブ・マネージャへのアクセス

キューブ・マネージャにアクセスするには、管理ポータルで以下の手順を実行します。

1. 以下のように、適切なネームスペースに切り替えます。
 - a. [変更] をクリックします。
 - b. ネームスペースをクリックします。
 - c. [OK] をクリックします。
2. [アナリティクス]→[管理]→[キューブマネージャ] をクリックします。
3. このネームスペースでキューブ・マネージャを使用したことがない場合、キューブ・レジストリに関する情報の入力を求めるプロンプトが表示されます。この場合、以下の情報を指定します。
 - ・ [キューブ・レジストリのクラス名] – 完全なクラス名 (パッケージを含む) を指定します。このクラス定義は、このネームスペースのキューブ・レジストリになります。
 - ・ [無効] – 必要に応じて、これをクリックしてレジストリを無効にします。レジストリを無効にすると、キューブ・マネージャのタスクは一時停止されます。(まだキューブ・マネージャのタスクはないため、この時点でレジストリを無効にする必要はありません。)
 - ・ [グループの更新] – 相互にグループを更新する方法を指定します。[連続] を選択すると、タスクでは一度に 1 グループが更新されます。[並行] を選択すると、タスクでは並行して複数のグループが更新されます。
 - ・ [この時点以降に構築の開始を可能にする] – 構築可能な最も早い時間を指定します。

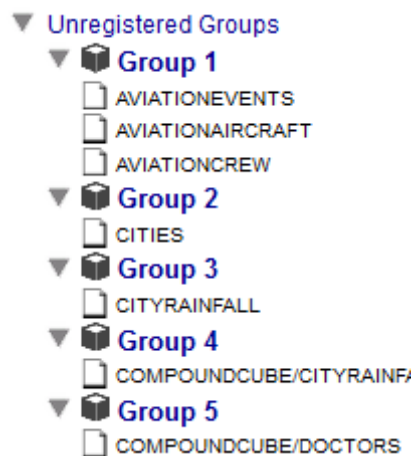
これらのすべての詳細は後で変更できます (クラス名を除く)。

そして、[OK] をクリックします。

[キューブ・レジストリ] ページが表示されます。このページは 2 つのモードで表示できます ([表示] ボタンを使用)。左の [表示] ボタンをクリックするとツリー表示になり、右の [表示] ボタンをクリックするとテーブル表示になります。

8.5.3.1 ツリー表示

ツリー表示では、キューブ・マネージャの左側の領域に登録されていないキューブ・グループのツリーが表示されます。以下はその例です。



中央の領域には、登録されたグループに関する情報が記載されたテーブル（初期状態では空）が表示されます。以下の例は、グループ登録後のテーブルの外観を示します。

Registered Groups	Build Frequency	Synch Frequency
▼ Group 14	1 Week (Sunday)	1 Day
RELATEDCUBES/CITIES	1 Week (Sunday)	
RELATEDCUBES/DOCTORS	1 Week (Sunday)	
RELATEDCUBES/PATIENTS	1 Week (Sunday)	1 Day
RELATEDCUBES/ALLERGIES	1 Week (Sunday)	
RELATEDCUBES/CITYRAINFALL	1 Week (Sunday)	1 Day

この領域は、以下のように色分けされます。

- ・ 背景が白 – このグループまたはキューブは包含されています。つまり、キューブ・マネージャのタスクはこれを更新します。このページで後述する“[更新計画の指定](#)”の[除外]オプションを参照してください。
- ・ 背景がグレー – このグループまたはキューブは除外されています。つまり、キューブ・マネージャのタスクはこれを無視します。

また、この領域は、以下のように指定されたキューブに基づいてサブジェクト領域を（イタリックで）リストします。

▼ Group 10	1 Week (Sunday)	1 Day
PATIENTS	1 Week (Sunday)	1 Day
<i>ASTHMAPATIENTS</i>		
<i>DEMOMDX</i>		
<i>YOUNGPATIENTS</i>		

キューブ内の更新内容は、そのキューブに基づいたすべてのサブジェクト領域で自動的に利用可能になるため、サブジェクト領域に対して更新計画を指定することはできません。（このため、サブジェクト領域のベースとなっているキューブとは独立して、そのサブジェクト領域を更新する必要も方法もありません。）

右側の領域では、[詳細] タブ（非表示）に現在の選択肢の詳細が表示されます。このタブは編集できます。[ツール] タブには、他のツールへのリンクが用意されています。

注釈 キューブ・マネージャがツリー表示されている場合は、中央領域に表示されるすべての登録済みグループの表示を展開したり折りたたんだりできます。そのためには、中央領域の上部にある **[すべて展開]** ボタンや **[すべて折りたたみ]** ボタンを適宜使用します。これらのボタンを使用しても、未登録グループが表示されるページの左側領域は変化しません。

8.5.3.2 テーブル表示

テーブル表示では、キューブ・マネージャに、現在のネームスペースのすべてのキューブが更新計画とともにリストされます。以下はその例です。

Filter:		Page size: 0	Max rows: 1000	Results: 20	Page: < << 1 >> > of 1				
Cube Name	Group Name	Registered	Exclude	Group Build Order	Update Plan	Supports Synchronize	Build Every	Synch Every	
CITIES	Group 2	Yes	Yes		1 Build Only	No	1 Week		
CITYRAINFALL	Group 3	Yes	Yes		1 Build and Synch	Yes	1 Week	1 Day	
HOLEFOODS	Group 8	Yes	No		1 Build Only	Yes	1 Week		
PATIENTS	Group 12	Yes	No		1 Build and Synch	Yes	1 Week	1 Day	
RELATEDCUBES/CITIES	Group 14	Yes	Yes		1 Build Only	No	1 Week		
RELATEDCUBES/DOCTORS	Group 14	Yes	Yes		2 Build Only	No	1 Week		
RELATEDCUBES/PATIENTS	Group 14	Yes	Yes		3 Build and Synch	Yes	1 Week	1 Day	
RELATEDCUBES/ALLERGIES	Group 14	Yes	Yes		4 Build Only	No	1 Week		
RELATEDCUBES/CITYRAINFALL	Group 14	Yes	No		5 Build Only	Yes	1 Week		
AVIATIONEVENTS	Group 1	No	Yes			No			
AVIATIONAIRCRAFT	Group 1	No	Yes			No			
AVIATIONCREW	Group 1	No	Yes			No			
COMPOUNDCUBES/CITYRAINFALL	Group 4	No	Yes			Yes			

このテーブルは、以下のように色分けされます。

- 背景が白 — このキューブは包含されています。つまり、キューブ・マネージャのタスクはこれを更新します。このページで後述する **“更新計画の指定”** の **[除外]** オプションを参照してください。
- 背景がグレー — このキューブは除外されています。つまり、キューブ・マネージャのタスクはこれを無視します。
- 背景がピンク — このキューブは登録されていないため、更新計画がありません。

[グループ名] フィールドは各キューブの所属先のグループを示し、[グループの構築順序] フィールドはグループ内で各キューブが構築または同期される順序を示します。キューブ・マネージャは、登録されたグループ内のキューブに対してのみこの順序を計算します。

右側の領域では、**[詳細]** タブ (非表示) に現在の選択肢の詳細が表示されます。このタブは編集できます。**[ツール]** タブには、他のツールへのリンクが用意されています。

8.5.4 レジストリの詳細の変更

キューブ・マネージャに初めてアクセスすると、初期情報の入力を求めるプロンプトが表示されます。これらの詳細を後で変更するには、以下の手順を実行します (レジストリのクラス名を除く。これは変更できません)。

1. **ツリー表示** でキューブ・マネージャを表示します。
2. 中央の領域で、**[登録されたグループ]** で始まる見出しをクリックします。
3. 右側の詳細を編集します。
オプションの詳細は、[前のセクション](#)を参照してください。
4. **[保存]** をクリックします。

8.5.5 キューブ・グループの登録

キューブ・グループを登録するには、以下の手順を実行します。

1. ツリー表示でキューブ・マネージャを表示します。
2. 左側の登録されていないキューブのリストを展開します。
3. この領域から中央の領域の[登録されたグループ]の見出しにグループをドラッグ・アンド・ドロップします。

または、テーブル表示でキューブ・マネージャを表示し、グループ内のキューブの行をクリックし、右側の領域の[登録されたグループ]をクリックします。

どちらの場合でも、変更内容は自動的に保存されます。

8.5.6 更新計画の指定

キューブ・グループおよびそのキューブの更新計画を指定するには、以下の手順を実行します。

1. ツリー表示でキューブ・マネージャを表示します。
2. 中央の領域でグループをクリックします。
3. 右側の[詳細] ペインで、以下の情報を指定します。

- ・ [名前] – このグループの一意の名前です。
- ・ [除外] – 生成されたタスクがこのグループ内のキューブに対して更新アクティビティを実行するかどうかを制御します。初期状態では、このオプションが選択されていて、グループが除外されています。

キューブ・マネージャでは、除外されているグループまたはキューブはグレーの背景で表示されます。

- ・ [更新計画] – 更新計画を選択します。
キューブ・マネージャでは、(このページで前述したように) キューブが同期をサポートしていない場合、同期の使用は許可されません。例えば、グループに[構築と同期] 計画を選択できますが、同期をサポートしていないキューブに対しては、キューブ・マネージャは、この更新計画を自動的に [Build] に設定します。

重要 キューブをキューブ・マネージャから同期する前に、少なくとも一度、キューブ・マネージャからキューブを構築する必要があります。

- ・ [構築間隔] – これらのフィールドを使用して、構築タスクのスケジュールを指定します (該当する場合)。
- ・ [同期間隔] – これらのフィールドを使用して、同期タスクのスケジュールを指定します (該当する場合)。
- ・ [同期してキューブを構築する] – これを選択すると、これらのキューブは同期的に構築されます (該当する場合)。このオプションがクリアされている場合は、それらが非同期で構築されます。

初期状態では、これらの詳細は、グループ内のすべてのキューブに適用されます。特定のキューブに対する詳細を編集し、後でグループの既定値を再適用する場合、[グループ内のすべてのキューブに適用する] をクリックします。

4. 必要に応じて、(中央の領域で) このグループ内のキューブをクリックし、右側の[詳細] ペインでそのキューブの情報を編集します。

オプションはグループ全体のものと似ていますが、キューブが同期をサポートしているかどうかに応じて、以下の追加オプションを含みます。

- ・ **[事後構築コード]**－このキューブを構築した直後に実行される ObjectScript の 1 行を指定します。以下はその例です。

```
do ##class(MyApp.Utils).MyPostBuildMethod("transactionscube")
```

- ・ **[事前同期コード]**－このキューブを同期する直前に実行される ObjectScript の 1 行を指定します。以下はその例です。

```
do ##class(MyApp.Utils).MyPresynchMethod("transactionscube")
```

必要に応じて、同期を中止するには、コードで以下を実行します。

```
set $$$ABORTSYNCH=1
```

- ・ **[事後同期コード]**－このキューブを同期した直後に実行される ObjectScript の 1 行を指定します。以下はその例です。

```
do ##class(MyApp.Utils).MyPostsynchMethod("transactionscube")
```

いずれの場合でも、コードによって、必要な任意のプロセスを実行できます。

必要に応じて、各キューブを変更します。

5. **[保存]** をクリックします。

これを行うと、キューブ・マネージャは、このネームスペースでキューブ・レジストリを作成または更新します。タスク・マネージャが必要なタスクをまだ組み込んでいない場合、キューブ・マネージャがこれを作成します。

8.5.7 グループのマージ

あるグループ (グループ A) を別のグループ (グループ B) にマージできます。具体的には、グループ A からグループ B にすべてのキューブを移動し、空になったグループ A を削除します。

あるグループを別のグループにマージするには、以下の手順を実行します。この手順では、グループ A はまだ登録されていない必要があり、グループ B は登録されている必要があります。

1. **ツリー表示**でキューブ・マネージャを表示します。
2. グループ A (移動するキューブが含まれているグループ) を左の領域から、中央の領域にあるグループ B (ターゲット・グループ) のグループ見出しにドラッグ・アンド・ドロップします。

システムからアクションの確認が求められます。

3. **[OK]** をクリックします。

新しく移動されたキューブの一部で使用できない更新計画がグループ B にある場合、これを示すダイアログ・ボックスが表示されます。**[OK]** をクリックします。このようなキューブの場合、キューブ・マネージャは、使用できる更新計画を選択します。

4. 新しく移動された各キューブに対する更新計画を確認し、必要に応じて変更します。
5. **[保存]** をクリックします。

または、以下の代替手順を実行してください。この手順では、両方のグループが登録されている必要があります。

1. **テーブル表示**でキューブ・マネージャを表示します。
2. 中央の領域で、グループ A (移動するキューブが含まれているグループ) のキューブの行をクリックします。
3. 右側で、**[別のグループにマージする]** をクリックし、ドロップダウン・リストからグループ B (ターゲット・グループ) を選択します。

4. [マージ] をクリックします。

システムからアクションの確認が求められます。

5. [OK] をクリックします。

新しく移動されたキューブの一部で使えない更新計画がグループ B にある場合、これを示すダイアログ・ボックスが表示されます。[OK] をクリックします。このようなキューブの場合、キューブ・マネージャは、使用できる更新計画を選択します。

6. 新しく移動された各キューブに対する更新計画を確認し、必要に応じて変更します。

7. [保存] をクリックします。

8.5.8 登録されているすべてのキューブの構築

システムには、登録されているキューブをすべて正しい順序で構築するために使用できるユーティリティ・メソッドが用意されています。このメソッドは、クラス `%DeepSee.CubeManager.Utils` の `BuildAllRegisteredGroups()` です。このメソッドでは、レジストリで指定されたスケジュールは無視されますが、レジストリで指定された構築順序が使用されます。

重要 キューブをキューブ・マネージャから同期する前に、少なくとも一度、キューブ・マネージャのユーザ・インタフェースからそれらのキューブを構築する必要があります。

8.5.9 オンデマンド構築の実行

キューブ・マネージャでは、オンデマンドで（つまり、スケジュールを無視して）キューブを構築するオプションも提供されます。この種の構築では、キューブ・マネージャは、要求されたキューブおよびこれに依存するすべてのキューブを再構築します。

オンデマンド構築を実行するには、以下の手順を実行します。

1. 変更内容をキューブ・レジストリに保存します。

重要 未保存の変更内容があると、構築オプションは無効になります。

2. 登録されているキューブを選択します。そのためには、以下のいずれかを実行します。

- ・ キューブ・マネージャを[ツリー表示](#)で表示し、中央の領域でキューブをクリックします。
- ・ キューブ・マネージャを[テーブル表示](#)で表示し、[Registered] 列に [はい] と表示されているキューブをクリックします。

3. 右側で、[除外] オプションをクリアします。

4. [依存関係リストを構築] をクリックします。

キューブ・マネージャで、構築ダイアログ・ボックスが表示されます。

5. [リストの構築] をクリックします。

ダイアログ・ボックスに、構築の進捗状況が表示されます。

6. 構築が完了したら、[OK] をクリックします。

オンデマンド構築を実行する方法は他にもあります。

- ・ [ツリー表示](#)でキューブ・マネージャを表示します。中央の領域でテーブルのヘッダをクリックします。次に、[すべての登録グループを構築] をクリックします。前述の説明に従って続行します。

- ・ **ツリー表示**でキューブ・マネージャを表示します。中央の領域でキューブ・グループをクリックします。次に、**[このグループを構築]**をクリックします。前述の説明に従って続行します。

8.5.10 キューブ・グループの登録解除

キューブ・グループを登録解除するには、以下の手順を実行します。

1. **ツリー表示**でキューブ・マネージャを表示します。
2. 中央の領域で、キューブ・グループの行で **[X]** をクリックします。
3. **[OK]** をクリックします。

8.5.11 キューブ・マネージャ・イベントの表示

特定のイベントでは、キューブ・マネージャはテーブルにログ・エントリを書き込みます。これは、SQL でクエリできます。テーブル名は、`%DeepSee_CubeManager.CubeEvent` です。CubeEvent フィールドは、キューブ・イベントのタイプを示します。このフィールドの可能な論理値には、以下が含まれます。

キューブ・イベントの値	キューブ・マネージャがこのログ・エントリを書き込む時期
register	キューブ・グループの登録の直後
update	キューブ・グループへの変更の保存の直後
unregister	キューブ・グループの登録解除の直後
build	キューブの構築時キューブ・マネージャは構築の開始直前に初期ログを生成し、構築の完了後にそのエントリを更新します。
synch	キューブの同期時キューブ・マネージャは同期の開始直前に初期ログを生成し、同期の完了後にそのエントリを更新します。
presynch	[事前同期コード] オプションで指定されたコードの実行直後
postsynch	[事後同期コード] オプションで指定されたコードの実行直後
postbuild	[事後構築コード] オプションで指定されたコードの実行直後
repair	[依存関係リストを構築] オプションの使用時(このオプションを指定すると、指定されたキューブおよび関連キューブのオンデマンド構築が実行されます)。キューブ・マネージャは構築の開始直前に初期ログを生成し、構築の完了後にそのエントリを更新します。

このテーブルの他のフィールドの詳細は、`%DeepSee.CubeManager.CubeEvent` のクラスリファレンスを参照してください。

8.5.12 キューブ・マネージャへのアクセスの制限

ユーザが[キューブ・レジストリ] ページを使用してキューブの更新スケジュールを変更できないようにして、そのスケジュールを管理することも考えられます。[キューブ・レジストリ] ページへのアクセスを制限するには、保存したキューブ・レジストリ内の **RegistryMap** オブジェクトまたは **RegistryMapGroup** オブジェクトで **UserUpdatesLocked** 属性を "true" に設定します。以下に例を示します。

```
<RegistryMap Disabled="false" IndependentSync="false" SerialUpdates="false" UserUpdatesLocked="true">
```

RegistryMap について **UserUpdatesLocked** を "true" に設定した場合、次のようになります。

- ・ **【詳細】** タブを使用してレジストリの **【無効】** 設定を変更することはできません。このタブにアクセスする方法については、“[レジストリの詳細の変更](#)”を参照してください。

RegistryMapGroup について UserUpdatesLocked を "true" に設定した場合、次のようになります。

- ・ 登録された各グループの **【除外】** チェック・ボックスは表示されますが、無効になります。
- ・ 登録された各キューブの **【除外】** チェック・ボックスは非表示になります。
- ・ 登録された各グループの **【更新計画】** は非表示になります。
- ・ 登録された各キューブの **【更新計画】** は非表示になります。
- ・ 登録されたグループを削除するための赤い [X] ボタンは削除されます。
- ・ **【構築頻度】** と **【同期頻度】** の列は空白のままです。
- ・ **【依存関係リストを構築】** をキューブで使用することはできますが、**【このグループを構築】** ボタンは無効になります。

8.6 %SynchronizeCube() の使用

注釈 キューブを同期する前に、このページで前述した “[キューブ同期の有効化](#)” の手順を実行します。

プログラムでキューブを同期するには（つまり、キューブ・マネージャを使用しない）、%DeepSee.Utils クラスの %SynchronizeCube() メソッドを呼び出します（これには以下のシグニチャがあります）。

```
classmethod %SynchronizeCube(pCubeName As %String, pVerbose As %Boolean = 1) as %Status
```

指定されているキューブ (pCubeName) に対して、このメソッドは前回の呼び出しより後に行われたすべての変更をソース・データから検出して適用します。

pVerbose が True の場合、メソッドはステータス情報をコンソールに書き込みます。このメソッドのその他の引数については、クラスリファレンスを参照してください。

%SynchronizeCube() は、以下の方法のいずれかで呼び出すことができます。

- ・ メソッドをベース・クラスのデータを変更するコードの一部から呼び出します。
これは、Patients サンプル内で使用されている方法です。
- ・ 定期的に、%SynchronizeCube() を繰り返しのタスクとして呼び出します。

%SynchronizeCube() によって、No changes detected というメッセージが表示された場合、これは、キューブを再構築したことがないことを示しています。

8.7 キューブの無効化

特定のシナリオでは、一時的にキューブを無効にすることができます。これにより、キューブの定義の編集中や既知のエラーの修正中にキューブを使用しようとした場合に、ユーザにエラーが表示されるのを防ぐことができます。キューブの削除とは異なり、キューブの無効化では、手で編集した内容はコードとは別に保持されます。無効化したキューブはキューブ・マネージャで非表示になるため、既にリレーションシップを確立したキューブを無効化しないことを強くお勧めします。

キューブを無効化するには、以下の手順を実行します。

1. 管理ポータルに管理者権限を持つユーザとしてログインします。
2. 希望のアナリティクス対応ネームスペースにいることを確認します。
3. [ホーム]→[アナリティクス] に移動し、[実行] をクリックします。
4. [開く] をクリックし、ポップアップ・ウィンドウから適切なキューブを選択します。
5. インタフェースの右側にある [詳細] ペインに、[無効] というラベルのチェックボックスが表示されます。キューブを無効にするには、このチェックボックスをクリックします。

実装したい変更内容を実装し終えたら、前述の [無効] ボックスのチェックを外すことによって、キューブを再度有効化できます。再度有効化する場合、キューブを再構築する必要があります。

8.8 DSTIME の削除

これまでの経緯と便宜上の理由から、DSTIME の削除という表現は、`^OBJ.DSTIME` グローバルから古いエンティティを削除することを表します。このグローバルは、かなり大きくなることがあるため、定期的に削除する必要があります。

特定のキューブの DSTIME を削除するには、以下の操作を実行します。

1. REST API の `/Data/GetDSTIME` を呼び出します。“InterSystems Business Intelligence のクライアント側 API” の “[GET /Data/GetDSTIME](#)” を参照してください。引数として、キューブのソース・クラスのフルネームを渡します。
この REST 呼び出しは、特定のサーバ上でそのソース・クラスに対して処理された最後の `^OBJ.DSTIME` タイムスタンプを返します。非同期ミラー設定の場合、この REST サービスから取得したタイムスタンプは、プライマリ・プロダクション・サーバで安全に削除できる直近のタイムスタンプになります。
2. 返されたタイムスタンプを引数として使用して、`%DeepSee.Utils` の `%PurgeUpdateBuffer()` メソッドを呼び出すことで、リモート・サーバで処理されたタイムスタンプ未満の `^OBJ.DSTIME` を削除します。このメソッドの既定の動作では、ローカルの `^OBJ.DSTIME` の最上位ノードが増分されます。これにより、Business Intelligence サーバに伝播される新しい同期ポイントが削除のたびに提供されるようになります。

8.9 キューブの手動更新

“[キューブの同期が不可能なとき](#)” で説明したように、キューブを手動で更新することが必要な場合があります。このような場合、アプリケーションで以下のことを実行する必要があります。

1. ベース・クラス内の影響を受けるレコードの ID を判断する。
2. `%DeepSee.Utils` の `%ProcessFact()` および `%DeleteFact()` メソッドを呼び出すことで、それらのレコードに対してキューブを更新する。

入力として、これらのメソッドには影響を受ける行の ID が必要です。

注釈 `%ProcessFact` を使用すると、開発者は DeepSee キューブへの単一 ID の挿入または更新を完全に制御できます。その機能を提供するために、`%ProcessFact` は、`%BuildCube` および `%SynchronizeCube` 内で提供される同時性保護をバイパスします。この保護は、複数のプロセスが同じ作業を試行することを防止するためのものです。

カスタム・コードに `%ProcessFact` を含める場合は、このコードで同じキューブ、ID ペアで複数の呼び出しを行わないようにすることを強くお勧めします。この保護がなければ、`%ProcessFact` が複数のプロセスの同じ ID で同時に呼び出された場合、重複する挿入が実行される可能性があることがわかっています。

以下のリストにこれらの方法に関する情報が用意されています。

%ProcessFact()

```
classmethod %ProcessFact(pCubeName As %String,
                          pSourceId As %String = "",
                          pVerbose As %Boolean = 0) as %Status
```

pCubeName はキューブの論理名、pSourceID はそのキューブで使用されるベース・クラス内のレコードの ID です。指定されたキューブに対して、このメソッドはファクト・テーブルの対応する行、関連付けられたインデックス、および影響を受ける場合はレベル・テーブルを更新します。

pVerbose が True の場合、メソッドはステータス情報をコンソールに書き込みます。

%DeleteFact()

```
classmethod %DeleteFact(pCubeName As %String,
                         pSourceId As %String = "",
                         pVerbose As %Boolean = 0) as %Status
```

pCubeName はキューブの論理名、pSourceID はそのキューブで使用されるベース・クラス内のレコードの ID です。指定されたキューブに対して、このメソッドはファクト・テーブルの対応する行を削除し、それに応じてインデックスを更新します。

pVerbose が True の場合、メソッドはステータス情報をコンソールに書き込みます。

8.10 その他のオプション

このセクションでは、その他の高度なオプションまたはあまり一般的でないオプションについて説明します。

- ・ [DSTIME=MANUAL を使用する方法](#)
- ・ [ファクト・テーブルにレコードを挿入する方法](#)
- ・ [ディメンジョン・テーブルを事前構築する方法](#)
- ・ [手動でディメンジョン・テーブルを更新する方法](#)

8.10.1 DSTIME=MANUAL を使用する方法

システムで ^OBJ.DSTIME グローバルを自動的に更新する代わりに、ユーザーが選択した時間にこのグローバルを更新することができます。そのためには、以下の操作を実行します。

1. DSTIME を、"AUTO" ではなく "MANUAL" に指定します。
2. その後、アプリケーション内で、%DeepSee.Utils クラスのオブジェクトを追加、変更、または削除するたびに、または ^OBJ.DSTIME グローバルを更新するときに、このクラスの %SetDSTimeIndex() メソッドを呼び出します。

このメソッドには、以下のシグニチャがあります。

```
ClassMethod %SetDSTimeIndex(pClassName As %String,
                            pObjectId As %String,
                            pAction As %Integer,
                            pInterval As %Integer = 0)
```

以下は、この指定の説明です。

- ・ pClassName は、追加、変更、または削除したオブジェクトの完全なパッケージとクラスの名前です。

- ・ pObjectId は、そのオブジェクトのオブジェクト ID です。
- ・ pAction は、オブジェクトを更新した場合は 0、オブジェクトを追加した場合は 1、オブジェクトを削除したか、オブジェクトを削除することなく対応するファクトをファクト・テーブルから削除したい場合は 2 です。pAction の値は、生成される ^OBJ.DSTIME ノードの値として使用されます。対応するレコードがソース・クラスに存在しない場合、または pAction に値 2 が指定されている場合、ファクトは同期中にキューブから削除されることに注意してください。
- ・ pInterval はオプションの整数です。これを正の整数として指定すると、^OBJ.DSTIME グローバルおよび ^DeepSee.Update グローバルでタイムスタンプ・サブスクリプトが使用されます。“[キューブ同期の有効化](#)”の DSINTERVAL パラメータの説明を参照してください。

次に、指定のキューブを更新する場合、前述のように %DeepSee.Utils クラスの %SynchronizeCube() メソッドを呼び出します。

8.10.2 ファクト・テーブルへのファクトの挿入

まれに、どのソース・レコードとも対応しないレコードを含むファクト・テーブルが必要なことがあります。その場合は、キューブ・クラスの %InjectFact() メソッドを使用します。

このメソッドには、以下のシグニチャがあります。

```
classmethod %InjectFact(ByRef pFactId As %String,
                        ByRef pValues As %String,
                        pDimensionsOnly As %Boolean = 0)
                        as %Status
```

以下は、この指定の説明です。

- ・ pFactId は、ファクトの ID です。これを "" (挿入を表します) に設定します。この引数が返されるときに、ファクトで使われる ID が格納されます。
- ・ pFactId は、ファクト値の多次元配列です。この配列では、添え字は sourceProperty 名 (大文字と小文字の区別あり) です。
- ・ pDimensionsOnly では、このメソッドがファクト・テーブルとディメンジョン・テーブルの両方に影響するか、ディメンジョン・テーブルのみに影響するかを制御します。この引数が True の場合、このメソッドはディメンジョン・テーブルにのみ影響します。この引数は、[次のセクション](#)の説明に従ってディメンジョンを事前構築する場合に使用します。

注意 ソース式に基づくレベルのディメンジョン・テーブルを更新する場合には、このメソッドを使用しないでください。これらのテーブルにレコードを追加するには、代わりに SQL UPDATE 文を使用します。

 ソース・プロパティに基づくレベルのディメンジョン・テーブルを更新する場合、%InjectFact() を使用できます。

8.10.3 ディメンジョン・テーブルの事前構築

既定では、ファクト・テーブルが構築されると同時に、ディメンジョン・テーブルにデータが入力されます。ファクト・テーブルより前にディメンジョン・テーブルにデータが入力されるように (何らかの理由でこれが必要な場合)、1 つまたは複数のディメンジョン・テーブルを事前構築できます。

1 つまたは複数のディメンジョン・テーブルを事前構築するには、以下の手順を実行します。

- ・ キューブ定義クラスに %OnBuildCube() コールバックを実装します。このメソッドには、以下のシグニチャがあります。

```
classmethod %OnBuildCube() as %Status
```

%BuildCube() メソッドは、古いキューブ・コンテンツを削除した直後、新しいコンテンツの処理を開始する前にこのメソッドを呼び出します。

- この実装では、キューブ・クラスの %InjectFact() メソッドを呼び出して、pDimensionsOnly 引数を True に指定します。
- このメソッドの詳細は、[前のセクション](#)を参照してください。

例えば、以下の部分実装では、HoleFoods サンプルの Cities ディメンジョンが事前定義されています。

Class Member

```
ClassMethod %OnBuildCube() As %Status
{
    // pre-build City dimension
    Set tVar("Outlet.Country.Region.Name") = "N. America"
    Set tVar("Outlet.Country.Name") = "USA"

    Set tVar("Outlet") = 1000
    Set tVar("Outlet.City") = "Cambridge"
    Do ..%InjectFact("",.tVar,1)

    Set tVar("Outlet") = 1001
    Set tVar("Outlet.City") = "Somerville"
    Do ..%InjectFact("",.tVar,1)

    Set tVar("Outlet") = 1002
    Set tVar("Outlet.City") = "Chelsea"
    Do ..%InjectFact("",.tVar,1)

    Quit $$$OK
}
```

メモ：

- メンバの名前のほかに一意の ID も指定する必要があります。
- 完全を期すために、このコードでは市の人口、経度、および緯度も指定する必要があります。これは、対応するディメンジョン・テーブルにはこれらの値が格納されているためです。
- また、より高レベルのメンバの値を指定する必要もあります。

8.10.4 手動でのディメンジョン・テーブルの更新

場合によっては、ベース・クラスに変更がなくても、レベルとして使用される検索テーブルに変更があることがあります。そのような場合、このページで前述した方法のいずれかでキューブを更新できます。ただし、1 つのディメンジョン・テーブルを変更するだけであれば、そのレベルのテーブルを直接更新の方が簡単です。これには、%DeepSee.Utils の %UpdateDimensionProperty() メソッドを使用します。

このメソッドには、以下のシグニチャがあります。

```
classmethod %UpdateDimensionProperty(pCubeName As %String,
                                     pSpec As %String,
                                     pValue As %String,
                                     pKey As %String)
                                     as %Status
```

以下は、この指定の説明です。

- pCubeName は、キューブの名前です。
- pSpec は、更新するレベル・メンバを参照する MDX メンバ式です。この式では、ディメンジョン、階層、およびレベルの識別子を使用する必要があります。例："[doccd].[h1].[doctor].&[61]"

バリエーションとして、pSpec はメンバのプロパティへの参照とすることができます。例：

"[homed].[h1].[city].&[Magnolia].Properties("Principal Export")"

システムは、この引数と pCubeName 引数を使用して、更新するテーブルと行を確定します。

- ・ pValue は、このメンバの新しい名前です (存在する場合)。
また、メンバのプロパティを指定した場合は、pValue がそのプロパティの新しい値として使用されます。
- ・ pKey は、このメンバの新しいキーです (存在する場合)。
この引数は、pSpec のメンバを指定した場合のみ指定します。

このメソッドを使用して、以下の 3 種類の変更を行うことができます。

- ・ メンバの新しいキーを指定する。以下はその例です。

```
Set tSC =
##class(%DeepSee.Utills).%UpdateDimensionProperty("patients","[docd].[hl].[doctor].&[186]",,"100000")
```

既定では、キーは名前としても使用されます。そのため、このアクションによって名前も変更されることがあります。

- ・ メンバの新しい名前を指定する。以下はその例です。

```
Set tSC =
##class(%DeepSee.Utills).%UpdateDimensionProperty("patients","[docd].[doctor].&[186]","Psmith,Alvin")
```

既定では、名前はキーです。そのため、このアクションによってキーも変更されることがあります。

- ・ 他のプロパティの新しい値を指定する (Name と Key はどちらもプロパティです)。以下はその例です。

```
Set memberprop="homed.hl.city.Pine.Properties("Principal Export")"
Set tSC = ##class(%DeepSee.Utills).%UpdateDimensionProperty("patients",memberprop,"Sandwiches")
```

8.11 例

Patients サンプルには、データを変更し、同期または手動更新を必要に応じて使用するユーティリティ・メソッドが含まれています。それらのメソッドを試すには、このサンプルで提供されているダッシュボードを使用できます。

1. **サンプル**をインストールしたネームスペースでユーザ・ポータルを開きます。
2. **Real Time Updates** ダッシュボードをクリックします。
3. 左上の領域にあるボタンをクリックします。これらのボタンのそれぞれが、このサンプル内のデータをランダムに変更するメソッドを実行する KPI アクションを実行します。アクションは、バックグラウンド・プロセスを開始する JOB を使用してメソッドを起動します。
 - ・ **[患者を追加]** では、患者が追加されます。
このアクションは、患者を追加するたびに %SynchronizeCube() を呼び出しながら 100 人の患者を追加するメソッドを呼び出します。
 - ・ **[患者グループの変更]** では、一部の患者に対する患者グループの割り当てが変更されます。
このアクションは、一定の割合の患者について患者グループの割り当てをランダムに変更し、その変更のたびに %SynchronizeCube() メソッドを呼び出すメソッドを呼び出します。
 - ・ **[一部の患者を削除]** では、一部の患者が削除されます。
このアクションは、患者を削除するたびに %SynchronizeCube() を呼び出しながら 1% の患者を削除するメソッドを呼び出します。

- ・ **【好きな色を変更】** では、一部の患者の好みの色が変更されます。

このアクションは、一定の割合の患者の好みの色をランダムに変更するメソッドを呼び出します。この場合、変更されたデータは **BI.Study.PatientDetails** テーブルに格納されますが、このテーブルは Patients キューブのベース・テーブルではありません。したがって、%SynchronizeCube() の代わりに %ProcessFact() を使用する必要があります。

このコード・ブロックは、データの変更によって影響を受けるすべての患者を返す SQL クエリを実行します。その後、それらの患者を繰り返し処理し、患者ごとに Patients キューブを更新します。

- ・ **【受診を追加】** では、一部の患者の受診が追加されます。

このアクションは、**BI.Study.PatientDetails** の場合と同様のロジックを含むメソッドを呼び出します（前の項目を参照してください）。

- ・ **【医者グループの変更】** では、一部の一次診療医で医者グループの割り当てが変更されます。

このアクションは、**BI.Study.PatientDetails** の場合と同様のロジックを含むメソッドを呼び出します。

Tip ヒン これらのメソッドは、ログの詳細を ^DeepSee.Study.Log グローバルに書き込みます。以下はその例です。

ト

```
^DeepSee.Study.Log(1)="13 May 2011 05:29:37PM Adding patients..."
```

```
^DeepSee.Study.Log(2)="13 May 2011 05:29:38PM Current patient count is 10200"
```

9

Business Intelligence クエリのプログラムによる実行

ここでは、InterSystems IRIS® Business Intelligence 結果セット API の使用方法と、MDX ファイルが含まれているファイルの実行方法を説明します。これらのオプションは、Business Intelligence の実装で必要な場合があります。

%ShowPlan() および %PrintStatistics() の詳細は、“[Analytics エンジンの仕組み](#)”を参照してください。

9.1 結果セット API の使用

%DeepSee.ResultSet クラスを使用すると、キューブに対して MDX クエリを実行し、その結果を表示して検証できます。このクラスを使用するには、以下の操作を実行します。

1. %DeepSee.ResultSet のインスタンスを作成します。

以下はその例です。

ObjectScript

```
set rset=##class(%DeepSee.ResultSet).%New()
```

2. オプションで、キャッシュの使用を無効にします。そのためには、そのインスタンスの %UseCache プロパティを 0 に設定します。以下はその例です。

ObjectScript

```
set rset.%UseCache=0
```

既定では、キャッシュは有効になっています。

3. オプションで、トレースを有効にします。準備フェーズで詳細トレースを有効にするには、結果セット・インスタンスの %Trace プロパティを 1 に設定します。クエリのすべてのフェーズでトレースを有効にするには、%dstrace 変数を 1 に設定します。次に例を示します。

ObjectScript

```
set rset.%Trace=1
set %dstrace=1
```

既定では、トレースは無効になっています。

4. MDX クエリを文字列として作成します。以下に例を示します。

ObjectScript

```
set query="SELECT MEASURES.[%COUNT] ON 0, diagd.MEMBERS ON 1 FROM patients"
```

Business Intelligence でサポートされている MDX 構文および関数の詳細は、“[InterSystems MDX の使用法](#)” および “[InterSystems MDX リファレンス](#)” を参照してください。

5. クエリを準備して実行します。通常、以下の操作を行います。

- a. クエリ文字列を引数として使用し、インスタンスの `%PrepareMDX()` メソッドを呼び出します。
- b. `%Execute()` または `%ExecuteAsynch()` を呼び出します。

これらの各メソッドはステータスを返します。処理を進める前に、このステータスを確認するコードを用意する必要があります。

または、`%ExecuteDirect()` を呼び出すことで、クエリを準備して実行することもできます。

あるいは、`%DeepSee.ResultSet` の下位レベルのメソッドを呼び出すこともできますが、ここではそれらのメソッドについては説明しません。

注釈 クエリでプラグインを使用する場合、保留中の結果がすべて完了するまで、`%Execute()` と `%ExecuteDirect()` は値を返しません。具体的には、Analytics エンジンがクエリで使用するプラグインの実行を終了するまで、これらは値を返しません。

6. `%ExecuteAsynch()` を使用する場合、クエリが完了したかどうかを定期的に確認します。クエリでプラグインを使用する場合、保留中の結果が完了したことも確認してください。保留中の結果はプラグインからの結果で、クエリとは別に実行されます。

クエリのステータスを判別するには、インスタンスの `%GetStatus()` メソッドを呼び出します。または、`%DeepSee.ResultSet` の `%GetQueryStatus()` クラス・メソッドを呼び出します。これらのメソッドはクエリのステータスを返し、さらに保留中の結果のステータスも (別々に) 返します。詳細は、クラス・ドキュメントを参照してください。

未完了のクエリをキャンセルするため、必要に応じて、`%CancelQuery()` クラス・メソッドを呼び出します。

7. これで、`%DeepSee.ResultSet` のインスタンスにクエリ結果が含まれます。ここで、そのインスタンスのメソッドを使用して、以下のようなタスクを実行することができます。

- ・ 結果の出力。
- ・ セル値の取得、結果セット内のセル数または軸数の取得、およびその他の結果の検証。
- ・ クエリ・プラン、リストに使用される SQL、クエリ内のセル範囲に使用される MDX など、クエリ自体のメタデータの取得。
- ・ クエリ統計の取得。

9.2 基本的な例

以下の例では、クエリを作成して準備し、これを実行して結果セットを出力として返し、結果を表示します。

Class Member

```
ClassMethod RunQuery1(Output result As %DeepSee.ResultSet) As %Status
{
  Set rset=##class(%DeepSee.ResultSet).%New()
  Set query="SELECT MEASURES.[%COUNT] ON 0, diagd.MEMBERS ON 1 FROM patients"
  Set status=rset.%PrepareMDX(query)
  If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

  Set status=rset.%Execute()
  If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

  Write !, "Full results are as follows *****",!
  Do rset.%Print()
  Quit $$$OK
}
```

ターミナルでこのメソッドを実行すると、以下のような結果が表示されます。

```
SAMPLES>do ##class(BI.APISamples).RunQuery1()

Full results are as follows *****
Patient Count
1 None 8,394
2 asthma 671
3 CHD 357
4 diabetes 563
5 osteoporosis 212
```

9.3 クエリの準備と実行

クエリを準備して実行する場合、通常以下のメソッドを使用します。

%PrepareMDX()

```
method %PrepareMDX(pMDX As %String) as %Status
```

クエリを解析し、実行時クエリ・オブジェクトに変換し、実行に向けて準備します。

%Execute()

```
method %Execute(ByRef pParms) as %Status
```

クエリを同期的に実行します。pParms 引数については、このリストの後で説明します。これは、クエリの準備が完了した後にのみ使用します。

%ExecuteAsynch()

```
method %ExecuteAsynch(Output pQueryKey As %String,
                      ByRef pParms,
                      pWait As %Boolean = 0) as %Status
```

クエリを非同期的に（または、pWait の値に応じて同期的に）実行します。引数については、このリストの後で説明します。これは、クエリの準備が完了した後にのみ使用します。

%ExecuteDirect()

```
classmethod %ExecuteDirect(pMDX As %String,
                          ByRef pParms,
                          Output pSC As %Status) as %DeepSee.ResultSet
```

クエリを準備して実行し、結果セットを返します。pSC はステータスであり、その内容をチェックする必要があります。その他の引数については、このリストの後の説明を参照してください。

以下はその説明です。

- ・ `pParms` — このクエリで使用する名前付きパラメータの値を指定します。これは、以下のように 1 つ以上のノードを持つ多次元配列です。

ノード	値
パラメータ名 (大文字と小文字の区別なし)	このパラメータの値

これらの値は、クエリ自体の本文内に指定されている同じパラメータの値をオーバーライドします。

- ・ `pQueryKey` — このクエリの一意のキーを返し、このクエリを後で (クエリのキャンセル、セル数の取得などの目的で) 参照する際に使用できるようにします。
- ・ `pWait` — クエリが完了するまで待ってから、このメソッド呼び出しから戻るかどうかを指定します。

`pWait` が `True` の場合、`%ExecuteAsynch()` は同期的に実行されます。

以下のサンプルでは、名前付きパラメータを含むクエリが使用されています。これは、インターシステムズによる MDX への拡張機能です。

Class Member

```
ClassMethod RunQuery2(city as %String = "Magnolia",Output result As %DeepSee.ResultSet) As %Status
{
  Set rset=##class(%DeepSee.ResultSet).%New()
  Set query="WITH %PARM c as 'value:Pine' "
    _"SELECT homed.[city].@c ON 0 FROM patients"
  Set status=rset.%PrepareMDX(query)
  If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

  Set myparms("c")=city
  Set status=rset.%Execute(.myparms)
  If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

  Write !, "Full results are as follows *****",!
  Do rset.%Print()
  Quit $$$OK
}
```

ターミナル・セッションの例を以下に示します。

```
d ##class(BI.APISamples).RunQuery2("Centerville")

Full results are as follows *****
                          Centerville
                             1,124
```

9.4 クエリ結果の出力

診断のためにクエリ結果を表示するには、以下のメソッドのいずれかを使用します。

`%Print()`

クエリ結果を出力し、ステータスを返します。使用例については、このページで前述した “[基本的な例](#)” および “[クエリの準備と実行](#)” を参照してください。

`%PrintListing()`

クエリで MDX DRILLTHROUGH 節が使用されている場合、このメソッドはクエリの最初のセルのドリルスルーを行い、結果を現在のデバイスに出力します。それ以外の場合は、エラーを出力します。

このメソッドは何も返しません。

重要 どちらのメソッドでも、各データ行の先頭に (すなわち列見出しの後に) 行番号が付加されます。この行番号は結果の一部ではありません。

以下は、%PrintListing() の例です。

Class Member

```
ClassMethod RunQuery3()
{
    Set rset=##class(%DeepSee.ResultSet).%New()

    Set query="DRILLTHROUGH SELECT gend,female ON 0,birthd.[1913] ON 1 "
        _"FROM patients RETURN PatientID,PrimaryCarePhysician->LastName"

    Set status=rset.%PrepareMDX(query)
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit}

    Set status=rset.%Execute()
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit}

    Write !, "Listing details for the first cell are as follows *****",!
    Do rset.%PrintListing()
}
```

これは、以下のようにターミナルで使用できます。

```
SAMPLES>d ##class(BI.APISamples).RunQuery3()

Listing details for the first cell are as follows *****
# PatientID      LastName
1: SUBJ_101317   Xiang
2: SUBJ_104971   North
3: SUBJ_105093   Klausner
4: SUBJ_109070   Quine
```

9.5 クエリ結果の検証

プログラムによってクエリ結果を操作するには、まずそれらの構成を理解する必要があります。結果セットは、一連の軸から成るセルのセットです。結果セットの構成が不明な場合は、%GetRowCount() および %GetColumnCount() を使用して **行数と列数** に関する情報を取得します。

次に、**指定したセルの値** にアクセスするには、%GetOrdinalValue() メソッドを使用します。または、**列ヘッダと行ヘッダのラベル** にアクセスするには、%GetOrdinalLabel() メソッドを使用します。または、セルで使用されているメンバに関する **詳細情報** を取得するには、%GetAxisMembers() メソッドを使用します。次のサブセクションで詳細を説明します。

注釈 DRILLTHROUGH クエリの結果を調べるための、さまざまなメソッドが用意されています。**次のセクション** を参照してください。

9.5.1 列数と行数の取得

結果セット内の列数を取得するには、%GetColumnCount() を使用します。

同様に、行数を取得するには %GetRowCount() を使用します。

例えば、以下のメソッドは特定の結果セットを出力し、前述のメソッドを使用して、この結果セットの軸について報告します。

Class Member

```
ClassMethod ShowRowAndColInfo(rset As %DeepSee.ResultSet)
{
    //print query results
    write !, "Result set for comparison",!
    do rset.%Print()

    set colCount=rset.%GetColumnCount()
    set rowCount=rset.%GetRowCount()
    write !, "This result set has ",colCount, " column(s)"
    write !, "This result set has ",rowCount, " row(s)"
}
```

以下は、このメソッドの出力例です。

```
Result set for comparison
Patient Count
1 None 844
2 asthma 55
3 CHD 38
4 diabetes 55
5 osteoporosis 26

This result set has 1 column(s)
This result set has 5 row(s)
```

以下は、異なる結果セットに基づいた出力例です。

```
Result set for comparison
1 0 to 29->Female 207
2 0 to 29->Male 192
3 30 to 59->Female 205
4 30 to 59->Male 209
5 60+>Female 115
6 60+>Male 72

This result set has 1 column(s)
This result set has 6 row(s)
```

前述のとおり、%Print() によって各データ行の先頭に行番号が付加されますが、この行番号は結果の一部ではありません。

9.5.2 指定のセルの値の取得

指定したセルの値を取得するには、%GetOrdinalValue() を使用します。このメソッドには、以下のシグニチャがあります。

```
method %GetOrdinalValue(colNumber,rowNumber) as %String
```

ここで、colNumber は列番号です (1 は最初の列を表します)。同様に、rowNumber は行番号です (1 は最初の行を表します)。該当するセルが結果セット内にない場合は、このメソッドは NULL を返します。

9.5.3 列または行のラベルの取得

列または行で使用するラベルを取得するには、インスタンスの %GetOrdinalLabel() メソッドを呼び出します。このメソッドには、以下のシグニチャがあります。

```
method %GetOrdinalLabel(Output pLabel As %String,
    pAxis As %Integer,
    pPosition As %Integer,
    Output pFormat As %String) as %Integer
```

以下は、この指定の説明です。

- ・ pLabel は、以下のようにラベルごとに 1 つのノードを持つ多次元配列です。

ノード	値
ラベル番号を表す整数。最初のラベルが 1 で、以降同様です。	ラベル

この配列では、最初のラベルは最も具体的な（最も内側の）ラベル、2 番目のラベルはその次に具体的なラベル（以降同様）です。例を参照してください。

この配列は出力パラメータとして返されます。

- ・ pAxis は、検証する軸です。列のラベルを取得するには 1 を指定し、行のラベルを取得するには 2 を指定します。
- ・ pPosition は、検証する軸に沿った位置です。最初の位置は 1 です。

このメソッドは、指定された軸上の指定された位置にあるラベルの数を返します。以下に例を示します。この例では、（軸が複数のラベルを持つように）CROSSJOIN クエリを実行して、ラベルと比較できるように結果を表示してから、この軸上のメンバを繰り返し処理し、それぞれのラベルを出力します。

Class Member

```
ClassMethod ShowRowLabels() As %Status
{
    Set rset=##class(%DeepSee.ResultSet).%New()
    Set query="SELECT CROSSJOIN(aged.[age group].MEMBERS,"
    Set query=query_"gend.gender.MEMBERS) ON 1 FROM patients"
    Set status=rset.%PrepareMDX(query)
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

    Set status=rset.%Execute()
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

    Write !, "Full results are as follows *****",!
    Do rset.%Print()

    Write !, "Labels used on the rows are as follows *****",!
    For j=1:1:rset.%GetRowCount() {
        Write !, "Row ",j
        Set labelcount=rset.%GetOrdinalLabel(.pLabel,2,j)
        For i=1:1:labelcount {
            Write !, "    label("_i_") is "_pLabel(i)
        }
    }

    Quit $$$OK
}
```

ターミナルで実行すると、このメソッドは以下のような出力を生成します。

```
SAMPLES>d ##class(BI.APISamples).ShowRowLabels()

Full results are as follows *****

1 0 to 29->Female                207
2 0 to 29->Male                  192
3 30 to 59->Female                205
4 30 to 59->Male                  209
5 60+>Female                     115
6 60+>Male                       72

Labels used on the rows are as follows *****

Row 1
label(1) is Female
label(2) is 0 to 29
Row 2
label(1) is Male
label(2) is 0 to 29
Row 3
label(1) is Female
label(2) is 30 to 59
Row 4
label(1) is Male
label(2) is 30 to 59
Row 5
label(1) is Female
label(2) is 60 +
```

```

Row 6
    label(1) is Male
    label(2) is 60 +
SAMPLES>

```

9.5.4 セル・コンテンツの詳細の取得

ここまで、このページではラベルとセル値を取得する手順のみを紹介してきました。場合によっては、指定したセルのコンテンツに関するより具体的な情報が必要になることがあります。

まず、いくつかのサンプル・クエリを参考として挙げて、概念を検証することが有用です。例えば、Business Intelligence シェルに表示された次のクエリ結果について考えてみてください。

	Patient Count
1 None	844
2 asthma	55
3 CHD	38
4 diabetes	55
5 osteoporosis	26

この例では、各行は診断ディメンジョンの 1 つのメンバに対応しています。列は、Measures ディメンジョンの 1 つのメンバ (Patient Count) に対応しています。以下にもう 1 つの例を示します。

	Patient Count
1 0 to 29->Female	207
2 0 to 29->Male	192
3 30 to 59->Female	205
4 30 to 59->Male	209
5 60+>Female	115
6 60+>Male	72

この例では、各行は 1 つのタプルに対応しています。このタプルは、年齢グループ・ディメンジョンの 1 つのメンバと性別ディメンジョンの 1 つのメンバを組み合わせたものです。(タプルは複数のメンバの共通部分です。)

一般に MDX 結果セット内では、各行は 1 つのタプルに対応しており、各列も 1 つのタプルに対応しています。これらの各タプルは、最初の例のように単一のシンプルメンバである場合も、2 つ目の例のように複数メンバの組み合わせである場合もあります。タプルには、メジャーが含まれる場合と含まれない場合があります。

任意の指定したセルについて、そのセルが属する列や行のタプルに関する情報が必要になることがあります。これらのタプルに関する情報を取得するには、以下の手順を実行します。

1. 結果セットの %GetAxisMembers() メソッドを実行します。

```

method %GetAxisMembers(pAxis As %Integer,
    Output pKey,
    pItemNo As %Integer = "") as %Status

```

要求された軸 (およびその軸上の要求されたオプションの項目) の情報を検索し、それをプロセス・プライベート・グローバルに書き込み、参照によってそのグローバルから情報を取得するために使用できるキーを返します。(システムは、この情報をプロセス・プライベート・グローバルに書き込みます。これは、この情報が大量の情報になる可能性があり、その構造を事前に特定することができないためです。)

pAxis では、対象の軸を必要に応じて指定します。

- ・ 0 を使用すると、スライサ軸 (WHERE 節) に関する情報が返されます。
- ・ 1 を使用すると、列に関する情報が返されます (これは、MDX の軸 0 です)。
- ・ 2 を使用すると、行に関する情報が返されます。

pKey は、出力パラメータとして返されます。これは、その情報に後でアクセスするために使用するキーです。

pItemNo では、情報が必要な軸に対するタプルを必要に応じて指定します。この引数を指定すると、このメソッドは、指定されたタプルのみのデータを書き込みます。この引数を省略すると、このメソッドはすべてのタプルのデータを書き込みます。軸上の最初のタプルには 1 を使用します。

2. pKey を使用して、プロセス・プライベート・グローバル ^||DeepSee.AxisMembers から適切なノードを取得します。%GetAxisMembers() メソッドは、データをノード ^||DeepSee.AxisMembers(pKey,pAxis,j,k) に書き込みます。以下はその説明です。
 - ・ pKey は、%GetAxisMembers() メソッドから返されるキーです。
 - ・ pAxis は、軸を指定する整数です。
 - ・ j は、対象のタプルを指定する整数です。軸上の最初のタプルには 0 を使用します。
 - ・ k は、対象のタプルのメンバを指定する整数です。タプルの最初のメンバには 1 を使用します。
3. これらの各ノードから、適切なリスト項目を取得します。^||DeepSee.AxisMembers の各ノードは、以下の形式の値を保持します。

```
$LB(nodeno,text,dimName,hierName,levelName,memberKey,dimNo,hierNo,levelNo)
```

以下は、この指定の説明です。

- ・ nodeno は、軸のこの部分のノード番号です。
 - ・ text は、軸のこの部分のテキストです。
 - ・ dimName、hierName、および levelName は、軸のこの部分に使用されるディメンジョン、階層、およびレベルの名前です。
 - ・ memberKey は、軸のこの部分に使用されるメンバのキーです。
 - ・ dimNo、hierNo、および levelNo は、軸のこの部分に使用されるディメンジョン、階層、およびレベルの数です。
4. プロセス・プライベート・グローバル ^||DeepSee.AxisMembers の生成されたノードを削除します。
 または、%GetAxisMembers() メソッドを使用している他のプロセスがないことが明らかな場合には、グローバル全体を削除します。
 このグローバルが自動的に削除されることはありません。

次のサンプル・メソッドは、指定された結果セットとセル位置に基づいて、指定されたセルの列と行のタプルの情報を出力します。

Class Member

```
ClassMethod ShowCellDetails(rset As %DeepSee.ResultSet, col As %Integer = 1, row As %Integer = 1)
{
    //print query results
    write !, "Result set for comparison",!
    do rset.%Print()

    //call %GetAxisMembers to build process-private global with info
    //for given result set and axis; return key of node that has this info
    Set status=rset.%GetAxisMembers(1,.columnkey)
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit}
    Set status=rset.%GetAxisMembers(2,.rowkey)
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit}

    write !, "We are looking at the cell ("_col_", "_row_")"
    write !, "The value in this cell is ", rset.%GetOrdinalValue(col,row)
    write !, "For this cell, the column is a tuple that combines the following member(s):"
    set i=0
    while (i <= " ") {
        write !, "    Member ",i
        set infolist=^||DeepSee.AxisMembers(columnkey,1,col,i)
        write:$LI(infolist,3)'=" " !, "        Dimension name: ",$LI(infolist,3)
        write:$LI(infolist,4)'=" " !, "        Hierarchy name: ",$LI(infolist,4)
        write:$LI(infolist,5)'=" " !, "        Level name: ",$LI(infolist,5)
        i=i+1
    }
}
```

```

        write:$LI(infolist,6)'=" !, "      Member key: ", $LI(infolist,6)
        set i=$ORDER( ^||DeepSee.AxisMembers(columnkey,1,col,i) )
    }

    write !, "For this cell, the row is a tuple that combines the following member(s):"
    set i=0
    while (i != "") {
        write !, "      Member ",i
        set infolist=^||DeepSee.AxisMembers(rowkey,2,row,i)
        write:$LI(infolist,3)'=" !, "      Dimension name: ", $LI(infolist,3)
        write:$LI(infolist,4)'=" !, "      Hierarchy name: ", $LI(infolist,4)
        write:$LI(infolist,5)'=" !, "      Level name: ", $LI(infolist,5)
        write:$LI(infolist,6)'=" !, "      Member key: ", $LI(infolist,6)
        set i=$ORDER( ^||DeepSee.AxisMembers(rowkey,2,row,i) )
    }
    Kill ^||DeepSee.AxisMembers(columnkey)
    Kill ^||DeepSee.AxisMembers(rowkey)
}

```

以下は、このメソッドの出力例です。

Result set for comparison

	0 to 29	30 to 59	60+
1 Female->None	189	184	62
2 Female->asthma	18	7	7
3 Female->CHD	*	4	14
4 Female->diabetes	*	11	23
5 Female->osteopor	*	*	23
6 Male->None	178	186	45
7 Male->asthma	14	7	2
8 Male->CHD	*	5	15
9 Male->diabetes	*	11	10
10 Male->osteoporos	*	*	3

We are looking at the cell (2,6)

The value in this cell is 186

For this cell, the column is a tuple that combines the following member(s):

Member 0

Dimension name: AgeD
Hierarchy name: H1
Level name: Age Group
Member key: 30 to 59

For this cell, the row is a tuple that combines the following member(s):

Member 0

Dimension name: GenD
Hierarchy name: H1
Level name: Gender
Member key: Male

Member 1

Dimension name: DiagD
Hierarchy name: H1
Level name: Diagnoses
Member key: <null>

9.6 DRILLTHROUGH クエリのクエリ結果の検証

クエリで MDX DRILLTHROUGH 文を使用する場合は、別の方法を使用して結果を検証します。

この場合、`%DeepSee.ResultSet` のインスタンスの以下のメソッドを使用します。

```
method %GetListingResultSet(Output pRS As %SQL.StatementResult, Output pFieldList As %List) as %Status
```

このメソッドは、以下の出力パラメータを返します。

- ・ pRS は、DRILLTHROUGH クエリからの結果を含む `%SQL.StatementResult` のインスタンスです。
- ・ pFieldList は、この結果セット内のフィールドの (\$LIST 形式の) 一覧です。

pRS は、`%SQL.StatementResult` の他のインスタンスを使用する場合と同じ方法で使用します。詳細は、クラスリファレンスを参照してください。

9.7 クエリ・メタデータの検証

%DeepSee.ResultSet のインスタンスのキューブ名やクエリ・テキストなどのメタデータを取得するには、以下のメソッドを使用します（クエリ・プランへのアクセスの詳細は、次のセクションを参照してください）。

%GetCubeName()

```
method %GetCubeName() as %String
```

クエリで使用するキューブの名前を返します。このメソッドを使用する前に、クエリを準備する必要があります。

%GetListingSQL()

```
method %GetListingSQL() as %String
```

クエリが DRILLTHROUGH クエリである場合は、ソース・データの表示に使用される SQL 文を返します。

%GetParameterInfo()

```
method %GetParameterInfo(Output pParms) as %Status
```

クエリで使用されるパラメータを含む多次元配列を、パラメータに使用される値と共に返します。この配列は、このページで前述した構造を持ちます。

%GetQueryText()

```
method %GetQueryText() as %String
```

結果セットの作成時に使用された MDX クエリを含む文字列を返します。

%GetSlicerForCellRange()

```
method %GetSlicerForCellRange(Output pSlicer As %String,  
                               pStartRow As %Integer, pStartCol As %Integer,  
                               pEndRow As %Integer, pEndCol As %Integer)  
as %Status
```

指定されたセルの範囲の MDX スライサ文を含む文字列を、参照によって返します。セルの範囲を指定するには、開始の行と列および終了の行と列から成る矩形を指定します。いずれの軸でも、最初のセルの位置は 1 です。

%IsDrillThrough()

```
method %IsDrillThrough() as %Boolean
```

クエリが DRILLTHROUGH クエリである場合は True を、それ以外の場合は False を返します。

例えば、以下のメソッドは基本のメタデータに関するレポートを生成します。

Class Member

```
ClassMethod ShowQueryMetadata(rset As %DeepSee.ResultSet) As %Status  
{  
    Set cubename=rset.%GetCubeName()  
    Write !, "This result set comes from the following cube: ",cubename,!  
  
    Set status=rset.%GetParameterInfo(.pParms)  
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}  
    If $DATA(pParms) {  
        Write "The query uses the following parameters:",!  
    }  
}
```

```

        Set p = $ORDER(pParms(""))
        While (p != "") {
            Write $$$UPPER(p), " = " , $GET(pParms(p,"VALUE")),!
            Set p = $ORDER(pParms(p))
        }
    }
Set query=rset.%GetQueryText()
Write "The query is as follows:",!, query,!

Set isdrill=rset.%IsDrillThrough()
If isdrill {
    Set listingsql=rset.%GetListingSQL()
    Write !!, "It uses the following SQL to drill into the source table:"
    Write !, listingsql
}
}

```

以下の例は、いくつかの結果セットのサンプルを使用して、このメソッドからの出力を示しています（見やすくするために改行が追加されています）。最初の例では、[サンプル](#)・クラス **BI.APISamples** の `GetResultSet1()` を使用します。

```

SAMPLES>set rs1=##class(BI.APISamples).GetResultSet1()

SAMPLES>d ##class(BI.APISamples).ShowQueryMetadata(rs1)

This result set comes from the following cube: patients
The query is as follows:
SELECT {[MEASURES].[AVG TEST SCORE],[MEASURES].[%COUNT]} ON 0,
[DIAGD].[H1].[DIAGNOSES].MEMBERS ON 1 FROM [PATIENTS]

```

次の例では、名前付きパラメータを含むクエリを使用する `GetResultSet2()` を使用します。

```

SAMPLES>set rs2=##class(BI.APISamples).GetResultSet2()

SAMPLES>d ##class(BI.APISamples).ShowQueryMetadata(rs2)

This result set comes from the following cube: patients
The query uses the following parameters:
C = Magnolia
The query is as follows:
SELECT [HOMED].[H1].[CITY].MAGNOLIA ON 0,%SEARCH ON 1 FROM [PATIENTS]

```

次の例では、ドリルスルーを行うクエリを使用する `GetResultSet3()` を使用します。

```

SAMPLES>set rs3=##class(BI.APISamples).GetResultSet3()

SAMPLES>d ##class(BI.APISamples).ShowQueryMetadata(rs3)

This result set comes from the following cube: patients
The query is as follows:
DRILLTHROUGH SELECT [GEND].[H1].[GENDER].[FEMALE] ON 0,[BIRTHD].[H1].[YEAR].[1913] ON 1
FROM [PATIENTS] RETURN PatientID, PrimaryCarePhysician-> LastName

It uses the following SQL to drill into the source table:
SELECT TOP 1000 PatientID,PrimaryCarePhysician-> LastName FROM
BI_Study.Patient source WHERE source.%ID IN (SELECT _DSsourceId FROM
BI_Model_PatientsCube.Listing WHERE _DSqueryKey = '1858160995')

```

以下のメソッドの例は、指定の結果セット内で指定のセルの範囲の MDX スライサを示すレポートを生成します。

Class Member

```

ClassMethod ShowSlicerStatement(rset As %DeepSee.ResultSet, Row1 As %Integer = 1,
Coll As %Integer = 1, Row2 As %Integer, Col2 As %Integer) As %Status
{
    If '$DATA(Row2) {Set Row2=Row1}
    If '$DATA(Col2) {Set Col2=Coll}

    Set status=rset.%GetSlicerForCellRange(.slicer,Row1,Coll,Row2,Col2)
    If $$$ISERR(status) {Do $System.Status.DisplayError(status) Quit status}

    Write !, "The requested cell range:"
    Write !, "    Columns ",Coll, " through ", Col2
    Write !, "    Rows    ",Row1, " through ", Row2

    Write !, "The slicer statement for the given cell range is as follows:"
    Write !, slicer
}

```

```

If 'rset.%IsDrillThrough(){
    Write !!, "For comparison, the query results are as follows:",!
    Do rset.%Print()
}
Else {
    Write !!, "This is a drillthrough query and %Print "
    _-"does not provide a useful basis of comparison"
}
}

```

このメソッドを試すには、**BI.APISamples** の `GetResultSet4()` を使用します。このメソッドでは、以下のように行と列に異なるレベルを持つクエリを使用します。

```

SAMPLES>d ##class(BI.APISamples).ShowSlicerStatement(rs4)

The requested cell range:
  Columns 1 through 1
  Rows    1 through 1
The slicer statement for the given cell range is as follows:
CROSSJOIN({[AgeD].[H1].[Age Bucket].&[0 to 9]},{[GenD].[H1].[Gender].&[Female]})

For comparison, the query results are as follows:

```

	Female	Male
1 0 to 9	689	724
2 10 to 19	672	722
3 20 to 29	654	699
4 30 to 39	837	778
5 40 to 49	742	788
6 50 to 59	551	515
7 60 to 69	384	322
8 70 to 79	338	268
9 80+	204	113

9.8 その他のメソッド

%DeepSee.ResultSet クラスでは、以下のような追加のメソッドも提供されています。

- ・ `%GetCellCount()`
- ・ `%FormatNumber()`
- ・ `%GetOrdinalLabel()`
- ・ `%GetOrdinalKey()`
- ・ `%GetQueryKey()`
- ・ `%GetRowTotal()`
- ・ `%GetColumnTotal()`
- ・ `%GetGrandTotal()`

全リストと詳細については、クラスリファレンスを参照してください。

9.9 クエリ・ファイルの実行

システムには、ファイルに保存されている MDX クエリを実行するためのツールが用意されています。出力は、現在のデバイスに書き込むことも、ファイルに書き込むこともできます。出力結果には、クエリの実行に関する統計が含まれます。

このツールは簡単なテストで役立ちます。

9.9.1 クエリ・ファイルについて

クエリ・ファイルは、以下のような ASCII ファイルである必要があります。

- ・ ファイル内の改行は無視されます。
- ・ 行内の 2 つ以上の空白は、1 つの空白として処理されます。
- ・ ファイルには、任意の数 (0 個以上) の MDX クエリを含めることができます。
- ・ クエリにはコメントを含めることができますが、コメントを入れ子にすることはできません。MDX コメントの形式は以下のとおりです。

```
/* comment here */
```

コメントは独自の行に配置される場合とそうでない場合があります。

- ・ 行で GO コマンドを単独で使用して、クエリを実行します。クエリは、前の GO コマンド (またはファイルの先頭) から指定した GO コマンドまで (ただし、この GO コマンドを除く) のすべてのテキストで構成されます。

その行の GO コマンドの前にスペースを入れないでください。

以下はその例です。

```
/* First query in this file*/
SELECT MEASURES.%COUNT ON 0,
homed.[home zip].[34577].CHILDREN
ON 1 FROM patients
GO

/* Second query in the file*/
SELECT MEASURES.%COUNT ON 0,
homed.[home city].MEMBERS ON 1 /*ignore this comment*/FROM patients
GO
```

9.9.2 クエリ・ファイルの実行

クエリ・ファイルを実行するには、%DeepSee.Shell の以下のクラス・メソッドを使用します。

```
ClassMethod %RunQueryFile(pQueryFile As %String, pResultFile As %String = "") As %Status
```

以下は、この指定の説明です。

- ・ pQueryFile は、クエリ・ファイルの名前です。
 - ・ pResultFile は、クエリ統計の書き込み先のファイルの名前です。
- この引数が NULL の場合、メソッドは現在のデバイスにクエリ統計を書き込みます。

いずれの場合でも、メソッドは現在のデバイスにクエリ結果を書き込みます。

以下はその例です。

```
d ##class(%DeepSee.Shell).%RunQueryFile("c:\mdxtest.txt")
-----
Query 1:
/* First query in this file*/SELECT MEASURES.%COUNT ON 0, homed.[home zip].[34577].CHILDREN ON 1 FROM
patients
Count
1 Cypress 1,091
2 Magnolia 1,087
3 Pine 1,121
Query Statistics:
```

```
Results Cache:          1
Computations:           0
Cache Hits:             0
Cells:                  0
Expressions:            0
```

```
Prepare:                0.261 ms
Execute Axes:           0.026 ms
Execute Cells:          0.000 ms
Consolidate:            0.000 ms
Total Time:             0.287 ms
```

```
ResultSet Statistics:
Cells:                  3
Parse:                  3.553 ms
Display:                0.361 ms
Total Time:             3.914 ms
```

```
-----
Query 2:
/* Query 2*/SELECT MEASURES.%COUNT ON 0, homed.[home city].MEMBERS ON 1 /*ignore this comment*/FROM
patients
```

```
Count
1 Cedar Falls      1,119
...
```

クエリ統計の詳細は、"[Analytics エンジンの仕組み](#)"を参照してください。

10

Business Intelligence のローカライズの実行

ここでは、InterSystems IRIS® Business Intelligence の実装プロセスの一環として、Business Intelligence 内の文字列をローカライズする方法を説明します。

10.1 Business Intelligence のローカライズの概要

このセクションでは、InterSystems IRIS Business Intelligence で文字列のローカライズがどのようにサポートされているかについて概要を説明します。

10.1.1 モデルのローカライズ

システムには、レベル、メジャー、およびその他のモデル要素の名前をローカライズするための簡単なメカニズムがあります。

Business Intelligence モデル内の各要素は、論理値と表示値を持ちます。その論理値、元の表示値、および他の言語ロケールで使用するための代替りの表示値を指定します。次に、以下の操作を行います。

- ・ MDX クエリでは、必ず論理値を使用します。
- ・ ユーザ・インタフェースでは、使用可能であれば、適切な表示値が使用されます。ユーザは、優先言語が使用されるようにブラウザを構成します。ブラウザからサーバに要求が送信されると、それらの要求は、使用可能であればその優先言語を使用するように指示します。サーバは、その優先言語に基づいて、一連の適切な文字列を含む応答を送信します。

10.1.2 フォルダ項目のローカライズ

同様に、ダッシュボード、ピボット・テーブル、およびその他のフォルダ項目内にある以下の文字列の特定のセットをローカライズできます。これらの文字列について、元の表示値と、他の言語ロケールで使用するための代替りの表示値を指定します。

ユーザ・ポータルとダッシュボード・ビューワでは、使用可能であれば、適切な表示値が使用されます。ユーザは、優先言語が使用されるようにブラウザを構成します。ブラウザからサーバに要求が送信されると、それらの要求は、使用可能であればその優先言語を使用するように指示します。サーバは、その優先言語に基づいて、一連の適切な文字列を含む応答を送信します。

10.2 モデルのローカライズの準備

Business Intelligence モデル内の文字列のローカライズを準備するには、以下の手順を実行します。

- 各キューブ、サブジェクト領域、および KPI クラス内で、DOMAIN クラス・パラメータを指定します。
以下はその例です。

Class Member


```
Parameter DOMAIN = "PATIENTSAMPLE";
```

Patients サンプル内のクラスでは、すべて DOMAIN に同じ値が使用されていますが、これは必須ではありません。クラスごとに異なる値を指定することもできます。

- Business Intelligence 要素ごとに displayName 属性の値を指定します。

アーキテクトでは、名前を指定すると、[表示名] フィールドが同じ名前で初期化されます。一方、スタジオで作業する場合は、name 属性 (必須) に加えて、displayName 属性 (オプション) を指定することに注意してください。

クラスをコンパイルすると、そのネームスペース内の `IRIS.Msg グローバルに値が追加されます。それらの値は以下のように表示されます。

Global Search Mask: ^IRIS.Msg		
Search History: ^IRIS.Msg ▼ 		Maximum Rows: 100
1:	^IRIS.Msg("HOLEFOODS")	= "en"
2:	^IRIS.Msg("HOLEFOODS", "en", 14218931)	= "Date Of Sale"
3:	^IRIS.Msg("HOLEFOODS", "en", 32956064)	= "Channel Name"
4:	^IRIS.Msg("HOLEFOODS", "en", 41399927)	= "%Search"
5:	^IRIS.Msg("HOLEFOODS", "en", 65437166)	= "DaySold"
6:	^IRIS.Msg("HOLEFOODS", "en", 79524168)	= "Listing"
7:	^IRIS.Msg("HOLEFOODS", "en", 118549960)	= "Longitude"
8:	^IRIS.Msg("HOLEFOODS", "en", 125554797)	= "HoleFoods Budget"
9:	^IRIS.Msg("HOLEFOODS", "en", 147780672)	= "Region"
10:	^IRIS.Msg("HOLEFOODS", "en", 212821625)	= "Product Category"
11:	^IRIS.Msg("HOLEFOODS", "en", 273309567)	= "Product Name"

このグローバル (メッセージ・ディクショナリといいます) には、そのネームスペースで定義されているメッセージが含まれます。Business Intelligence では、各メッセージはモデル要素の名前に対応します。

DOMAIN パラメータを定義するキューブ、サブジェクト領域、または KPI クラスをコンパイルすると、そのクラスで定義されているメッセージが既定の言語で含まれるように、このグローバルが更新されます。各メッセージは、数値識別子を使用し、既定の言語に適用される文字列値を含みます。

期待する一連の文字列が表示されない場合は、クラスで DOMAIN パラメータが定義されていること、displayName の値を指定済みであること、およびクラスをコンパイル済みであることを確認してください。

10.3 フォルダ項目のローカライズの準備

このセクションでは、ダッシュボード、ピボット・テーブル、およびその他のフォルダ項目にある文字列のローカライズを準備する方法について説明します。

10.3.1 既定のドメイン

DeepSeeUser は、ダッシュボード内のローカライズされた文字列を検索するときに、既定で使用されるドメインです。詳細は、以下のセクションを参照してください。

10.3.2 メッセージ・ディクショナリへの文字列の追加

コンパイル時にメッセージ・ディクショナリに一連のエントリを生成するクラスを作成します。そのクラスで以下の操作を行います。

- ・ **%RegisteredObject** または標準のシステム・マクロへのアクセスを提供するその他のクラスを拡張します。
- ・ DOMAIN クラス・パラメータを指定します。以下はその例です。

Class Member

```
Parameter DOMAIN = "DeepSeeUser";
```

DeepSeeUser ドメインは、[既定のドメイン](#)であるため、最も便利な選択肢です。

- ・ `$$$Text (Localizable String)` を使用して、指定のドメインに含まれている必要のある文字列を参照するメソッドを定義します。Localizable String は、評価結果がこのドメインの文字列となる式です。

このメソッドには任意の名前を指定できます。引数を取る必要も、値を返す必要もありません。以下に例を示します。

```
ClassMethod DefineL18N()
{
    set x=$$$Text("Dashboard Title")
    set x=$$$Text("Dashboard Description")
    set x=$$$Text("KeywordA")
    set x=$$$Text("KeywordB")

    set x=$$$Text("Control Label")
    set x=$$$Text("Tooltip")
    set x=$$$Text("Widget Title")
    set x=$$$Text("Chart Title")
}
```

または、`$$$Text (Localizable String)` の代わりに、`$$$Text (@MessageID@)` を使用します。MessageID は、指定したドメインで一意的な数値 ID です。

このクラスのコンパイル時に、コンパイラは `$$$Text` マクロの各インスタンスを検索し、このネームスペース内の `^IRIS.Msg` グローバルに値を追加します。

10.3.3 ダッシュボード、ピボット・テーブル、または他のフォルダ項目でのローカライズ可能な文字列の使用

ダッシュボード、ピボット・テーブル、または他のフォルダ項目の定義では、表示する実際の文字列の代わりに、以下のいずれかの値を使用します。

- ・ `$$$Localizable String`

Localizable String は、既定のドメインに定義されている文字列です。

例を以下に示します。

Description	Keywords
\$\$\$Dashboard Description	\$\$\$KeywordA \$\$\$KeywordB

Used to help find items. One keyword per line.

別の例を示します。

Control Label or Icon	\$\$\$Control Label
Label displayed for this control; U	
Control Tooltip	\$\$\$Tooltip
Tooltip displayed for this control	

- \$\$\$Localizable String/OtherDomain

Localizable String は、OtherDomain で指定されたドメインに定義されている文字列です。

例を以下に示します。

Description	Keywords
\$\$\$Dashboard Description1/myDomain	\$\$\$KeywordA1/myDomain \$\$\$KeywordB1/myDomain

Used to help find items. One keyword per line.

/OtherDomain の部分を指定していない場合は、既定のドメインでこの文字列が検索されます。

重要 フォルダまたはフォルダ・アイテムの名前の場合、\$\$\$Localizable String#OtherDomain のバリエーションを使用します。

例えば、\$\$\$My Folder#MyDeepSeeDomain をフォルダ名として使用します。

- \$\$\$@MessageID

MessageID は、既定のドメインに定義されている数値メッセージ ID です。

- \$\$\$@MessageID/OtherDomain

MessageID は、OtherDomain で指定されているドメインに定義されている数値メッセージ ID です。

/OtherDomain の部分を指定していない場合は、既定のドメインでこの文字列が検索されます。

フォルダ項目定義で以下のいずれかの文字列に対して、これらの値を使用します。

- フォルダ名
- フォルダ項目名
- (ダッシュボードの場合) ダッシュボードのタイトル (指定していれば、ダッシュボード名の代わりに表示されます)
- 項目の説明
- キーワード

- ・ ダッシュボード・コントロールのラベル
- ・ ダッシュボード・コントロールのツールのヒント
- ・ ウィジェットのタイトル (ウィジェットの論理名ではありません)。
- ・ グラフを表示するダッシュボード・ウィジェット内にあるグラフのタイトル

10.4 文字列のローカライズ

文字列をローカライズする手順は以下のとおりです。

1. メッセージ・ディクショナリを 1 つ以上の XML ファイルにエクスポートします。そのためには、ターミナルで以下の操作を実行します。
 - a. Business Intelligence を使用しているネームスペースに変更します。
 - b. 出力ファイルとその場所を特定します。

ObjectScript

```
SET file="C:\myLocation\Messages.xml"
```

指定したディレクトリは既に存在している必要があります。システムによって作成されることはありません。

- c. export コマンドを実行します。
 - ・ 特定のドメインにあるメッセージのみをエクスポートすることが実際的な場合があります。この場合は次のように指定します。

ObjectScript

```
DO ##class(%Library.MessageDictionary).ExportDomainList(file,"myDomain")
```

ドメイン名では大文字と小文字が区別されます。

- ・ ネームスペースにあるすべてのメッセージをエクスポートするには次のように指定します。

ObjectScript

```
DO ##class(%Library.MessageDictionary).Export(file)
```

2. 希望する言語ごとに、メッセージ・ファイルのコピーを作成します。
3. 各メッセージ・ファイルを以下のように編集します。
 - a. ルート要素の Language 属性を以下のように編集します。

```
<MsgFile Language="en">
```

これを希望する言語の言語名に変更します。

言語名は、[RFC1766](#) に準拠した、すべて小文字の言語タグである必要があります (これにより、ユーザはブラウザ内で標準セットから優先言語を選択できます)。このタグは、1 つ以上の部分で構成されます。つまり、主言語タグ (en や ja) の後に、オプションでハイフン (-) で区切られた 2 番目の言語タグを (結果として en-gb や ja-jp の形式になるように) 記述します。

以下はその例です。

```
<MsgFile Language="es">
```

- b. ファイルをスキャンして、適切なドメインに対応する <MsgDomain> 要素を検索します。

```
<MsgDomain Domain="myDomain">
```

エクスポートしたドメインが 1 つだけであれば、ファイルには <MsgDomain> 要素が 1 つだけ含まれます。

- c. そのセクション内で、各メッセージの値を編集します。例：変更前

```
<Message Id="2372513034">City</Message>
```

変更後：

```
<Message Id="2372513034">Ciudad</Message>
```

4. 編集したメッセージ・ファイルをインポートします。そのためには、以下の操作を実行します。

- 1 つのファイルをインポートするには、以下の操作を実行します。

ObjectScript

```
SET file="C:\myLocation\myfile.xml"
DO ##class(%Library.MessageDictionary).Import(file)
```

- 同じディレクトリ内にすべてのファイルをインポートするには、以下の操作を実行します。

ObjectScript

```
SET myFiles="C:\myLocation"
DO ##class(%Library.MessageDictionary).ImportDir(myFiles,"d")
```

5. 必要に応じて、管理ポータルを使用して、メッセージ・ディクショナリが更新されたことを確認します。そのためには、適切なネームスペースに切り替えて、[システム・エクスプローラ]→[グローバル]を選択して、^IRIS.Msg グローバルの [グローバル表示] をクリックします。

このグローバル内に、追加した言語に対応する一連の新しい添え字が表示されます。

6. ブラウザで、ローカライズされたページで使用するように要求する言語を制御する設定を見つけます。その設定を、編集したメッセージ・ファイルで指定した言語に変更します。

ブラウザによっては、ブラウザ・キャッシュのクリア、ブラウザの再起動、またはその両方が必要となる場合があります。

7. アナライザにアクセスして、変換された文字列が表示されることを確認します。

%Library.MessageDictionary のユーティリティ・メソッドの詳細は、そのクラスのクラス・リファレンスを参照するか、“文字列のローカライズとメッセージ・ディクショナリ” の技術文書を参照してください。

11

クラスへの Business Intelligence 要素のパッケージ化

Business Intelligence の実装では、ほとんどの場合、テスト・システム上でアプリケーションの要素を開発し、それらをプロダクション・システムにコピーします。ここでは、InterSystems IRIS® Business Intelligence 要素をパッケージ化して、それらを別のシステムにコピーする方法を説明します。

注釈 このページの内容は、ユーザがスタジオに対するエクスポートとインポートのプロセスを理解していることを前提としています。

[“その他のエクスポート/インポート・オプション”](#) も参照してください。

11.1 概要

Business Intelligence の実装には、以下の要素の一部またはすべてを含めることができます。

- ・ キューブ・クラス定義
- ・ サブジェクト領域クラス定義
- ・ KPI クラス定義
- ・ Business Intelligence フォルダ項目。クラスとして定義されていないすべての項目が含まれます。これには、ピボット・テーブル、ダッシュボード、ピボット変数などがあります。

これらの項目を別のシステム（ターゲット・システムと呼ばれる）に移動させるには、以下の手順を実行します。

1. 次のセクションの説明に従って、すべてのフォルダ項目を 1 つ以上の Business Intelligence コンテナ・クラスにエクスポートします。

Business Intelligence コンテナ・クラスには、任意の数の Business Intelligence フォルダ項目の XML 表現が含まれます。

2. キューブ、サブジェクト領域、および KPI クラス定義をエクスポートします。

Business Intelligence のクラス定義とフォルダ項目をすべて含むプロジェクトを作成できます。その後で、このプロジェクトを InterSystems IRIS® からエクスポートして、このプロジェクトが必要になる別の InterSystems IRIS インスタンスにインポートできます。スタジオのエクスポート・オプションやインポート・オプションを使用することも、%SYSTEM.OBJ の通常のクラス・メソッドを使用することもできます。

3. エクスポートしたフォルダ項目定義を調べて、[移植のための編集](#)が行えるようにします。

4. すべてのクラス定義をターゲット・システムにインポートします。

コンテナ・クラスをコンパイルするときに、システムはそれらのクラスに含まれるすべてのフォルダ項目に対して処理を繰り返し、それぞれの項目をターゲット・システムで作成または上書きします。

11.2 コンテナ・クラスへのフォルダ項目のエクスポート

Business Intelligence フォルダ項目をコンテナ・クラスにエクスポートするには、その項目が含まれるコンテナ・クラスを定義したファイルを生成するメソッドを使用します。メソッドは %ExportContainer() です。これはクラス %DeepSee.UserLibrary.Utils にあります。このメソッドは、以下のとおりです。

```
classmethod %ExportContainer(ByRef pItemList As %String,  
                             pFileName As %String,  
                             pContainerClassName As %String = "") as %Status
```

以下はその説明です。

- ・ pItemList は、以下の形式のノードが含まれる多次元配列です。

ノード	ノード値
pItemList(itemidentifier)	" "

各 itemidentifier には、以下のいずれかの文字列を使用します。

- dashboardname.dashboard。ここで、dashboardname はダッシュボードの名前です。すべてのダッシュボードを表すために、ワイルドカード * を使用できます。ワイルドカードは他のタイプの項目でも使用できます。
- pivotname.pivot。ここで、pivotname はピボット・テーブルの名前です (または * を使用)。
%ExportContainer() メソッドは、エクスポートされるダッシュボードによって使用されるすべてのピボット・テーブルを識別することに注意してください。明示的にエクスポートする必要があるピボット・テーブルだけが、ダッシュボードによって使用されないピボット・テーブルです。
- namedfiltername.namedFilter。ここで、namedfiltername は名前付きフィルタの名前です (または * を使用)。
- sharedcalcmembername.sharedCalcMember。ここで、sharedcalcmembername は共有される計算メンバです (または * を使用)。
- listinggroupname.listingGroup。ここで、listinggroupname はリスト・グループの名前です (または * を使用)。
- pivotvarname.pivotVariable。ここで、pivotvarname はピボット変数の名前です (または * を使用)。
- settingname.userSetting。ここで、settingname はユーザ設定の名前です (* を使用)。
- termlistname.termList。ここで、termlistname は用語リストの名前です (または * を使用)。
- themename.theme。ここで、themename はダッシュボードのテーマの名前です (または * を使用)。
- widgettemplatename.widgetTemplate。ここで、widgettemplatename はウィジェット・テンプレートの名前です (または * を使用)。
- linkname.link。ここで、linkname はダッシュボード・リンクの名前です (または * を使用)。
- reportname.report。ここで、reportname はダッシュボード・レポートの名前です (または * を使用)。
- ・ pFileName は、生成するファイルの名前です。

- ・ `pContainerClassName` は、生成するコンテナ・クラスのフルネームです (パッケージを含む)。

11.3 移植のための Business Intelligence のフォルダ項目の編集

Business Intelligence フォルダ項目を別のシステムにコピーする場合は、エクスポートした XML を検証して、必要な編集を実行することが重要になります。これについては、後続のサブセクションで説明します。

また、以下の点についても注意してください。

- ・ ダッシュボードをエクスポートするときに、システムは、そのダッシュボードが使用するピボット・テーブルを自動的にエクスポートしません。ユーザが、それらのピボット・テーブルも特定してエクスポートする必要があります。
- ・ Business Intelligence 要素間の参照 (ダッシュボードからピボット・テーブルへの参照など) は、名前に基づいて行われます。

11.3.1 <filterState> 要素の削除

以前のリリースで保存されている場合、サポートされなくなった `<filterState>` 要素がフォルダ項目の定義に含まれている場合があります。この場合、これらを削除する必要があります。つまり、開始タグの `<filterState>` と、それに対応する終了タグ `</filterState>` の両方を削除します。

11.3.2 ローカル・データの削除

フォルダ項目の定義には、そのシステムにローカルで他のシステムでは使用できない情報が含まれていることがあります (システム間でパッケージ化および共有する要素の内容に応じて異なります)。以下の項目について、XML を確認してください。

localDataSource 属性

これは、エクスポートしたダッシュボードの `<widget>` 要素にあります。

この属性には、ミニ・アナライザで実行されたローカルのオーバーライドが含まれています。別のシステムでエクスポートした XML を使用する場合は、これを必ずクリアしてください。例：変更前

```
localDataSource="$LOCAL/Basic Dashboard Demo/SamSmith/590125613.pivot"
```

変更後：

```
localDataSource=""
```

または、`localDataSource` 属性を削除します。

owner 属性

これは、すべてのフォルダ項目にあります。

この要素には、この項目を所有するユーザの名前が含まれています。指定されたユーザがターゲット・システムに存在しない場合は、この属性を編集します。この属性は、NULL に設定できます。例：変更前

```
owner="DevUser"
```

変更後：

```
owner=""
```


または、この属性を削除できます。

resource 属性

これは、すべてのフォルダ項目にあります。

この要素には、この項目をセキュリティ保護するために使用されたリソースの名前が含まれています (存在する場合)。このリソースがターゲット・システムに存在しない場合は、この属性を編集します。この属性は、NULL に設定することも、削除することもできます。

createdBy 属性

これは、すべてのフォルダ項目にあります。

この要素には、この項目を作成したユーザの名前が含まれています。この属性は、NULL に設定することも、削除することもできます。これを実行すると、XML のインポート時に (または、コンテナ・クラスのコンパイル時に)、createdBy が現在のユーザに設定されます。

timeCreated 属性

これは、すべてのフォルダ項目にあります。

この要素には、この項目を作成したユーザの名前が含まれています。この属性は、NULL に設定することも、削除することもできます。これを実行すると、XML のインポート時に (または、コンテナ・クラスのコンパイル時に)、timeCreated が現在のタイム・スタンプに設定されます。

11.4 エクスポートしたコンテナ・クラスのインポート

エクスポートしたコンテナ・クラスをインポートするには、%ImportContainer() メソッドを使用します。これはクラス %DeepSee.UserLibrary.Utils にあります。このメソッドは、以下のとおりです。

```
ClassMethod %ImportContainer(pFileName As %String = "", pReplace As %Boolean = 1) As %Status
```

以下はその説明です。

- ・ pFileName は、生成するファイルの名前です。
- ・ pReplace は、既存のクラスを置き換えるかどうかを指定します。

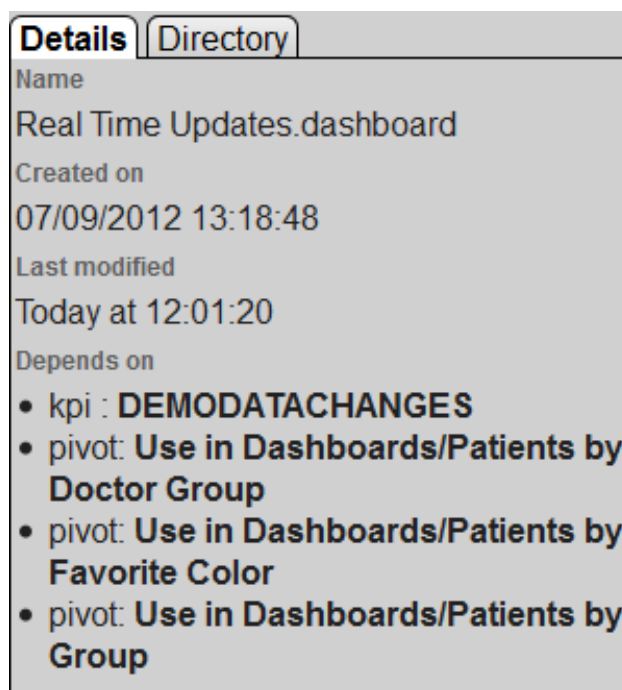
%ImportContainer() は、%OnLoad() メソッドを自動的に呼び出します (これがコンテナ・クラスで定義されている場合)。

11.5 フォルダマネージャの使用法

このセクションでは、フォルダマネージャを使用して、[項目の依存関係の表示](#)、[項目のエクスポート](#)、および[項目のインポート](#)を実行する方法について説明します。また、このページで後述するように、スタジオで[エクスポート](#)および[インポート](#)を実行することもできます。

11.5.1 フォルダ項目の依存関係の表示

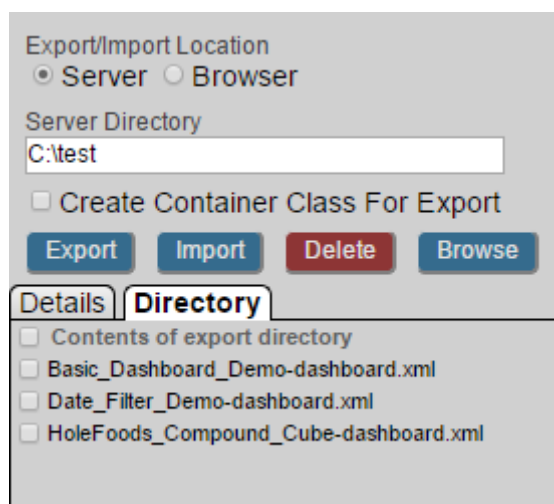
1 つの項目のチェック・ボックスにチェックを付けた場合は、フォルダマネージャの左側の領域に、その項目の詳細 (その項目が依存する項目の一覧など) が表示されます。



11.5.2 サーバへの Business Intelligence フォルダ項目のエクスポート

Business Intelligence のフォルダ項目をサーバ上のファイルにエクスポートするには、以下の手順を実行します。

1. [InterSystems ランチャー] をクリックし、[管理ポータル] をクリックします。
セキュリティの設定によっては、InterSystems IRIS ユーザ名とパスワードを使用してログインするように求められます。
2. 以下のように、適切なネームスペースに切り替えます。
 - a. [変更] をクリックします。
 - b. ネームスペースをクリックします。
 - c. [OK] をクリックします。
3. [Analytics]→[管理]→[フォルダマネージャ] の順に選択します。
4. [サーバ] を選択します。
5. [サーバ・ディレクトリ] に、項目のエクスポート先ディレクトリのフル・パスを入力します。または、そのネームスペースの既定のデータベースが含まれているディレクトリを基準とした相対ディレクトリ名を入力します。または、[参照] ボタンを使用します。
このディレクトリは既に存在している必要があります。
6. エクスポートする各項目の横にあるチェック・ボックスにチェックを付けます。
また、すべての項目を選択する場合は、チェック・ボックス列の一番上にあるチェック・ボックスにチェックを付けます。
7. [エクスポート] をクリックします。
8. 必要に応じて、[ディレクトリ] タブをクリックします。このタブに、指定したディレクトリ内のファイルが表示されます。



11.5.2.1 バリエーション: コンテナ・クラスのエクスポート

代わりに特定のフォルダ項目が含まれるコンテナ・クラスで構成される単一ファイルをエクスポートするには、以下の手順を実行します。

1. 前述した手順のとおり、[サーバ] と [サーバ・ディレクトリ] を指定します。
2. エクスポートする項目を選択します。
3. [エクスポートするコンテナ・クラスの作成] オプションを選択します。
4. 必要に応じて、[関連サポート項目のエクスポート] を選択します。これにより、選択したフォルダ項目の配置に必要とされる可能性のあるサポート項目をすべてエクスポートします。サポート項目の例として、ピボット変数、名前付きフィルタ、共有される計算メンバなどが挙げられます。
5. [コンテナ・クラス名] には、必要に応じて完全修飾クラス名 (パッケージとクラス) を指定します。[コンテナ・クラス名] の指定がない場合は、名前の生成にコンテナ・クラスとエクスポート・ファイルの両方が使用されるようになります。
6. [エクスポート] をクリックします。

コンテナ・クラスに関する詳細は、"[クラスへの Business Intelligence フォルダ項目のパッケージ化](#)" を参照してください。

11.5.3 ブラウザへの Business Intelligence フォルダ項目のエクスポート

Business Intelligence のフォルダ項目をブラウザのダウンロード・ディレクトリにエクスポートするには、以下の手順を実行します。

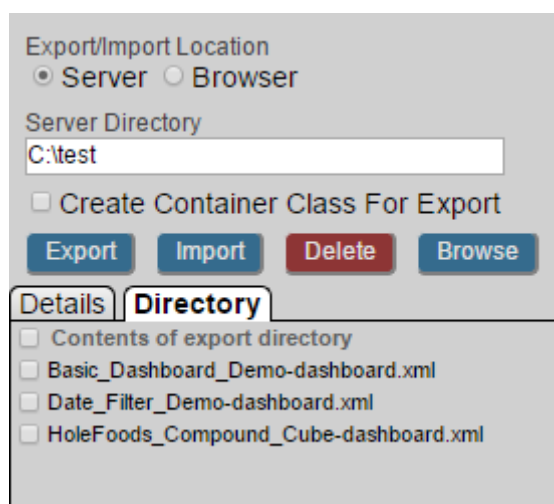
1. [InterSystems ランチャー] をクリックし、[管理ポータル] をクリックします。
セキュリティの設定によっては、InterSystems IRIS ユーザ名とパスワードを使用してログインするように求められます。
2. 以下のように、適切なネームスペースに切り替えます。
 - a. [変更] をクリックします。
 - b. ネームスペースをクリックします。
 - c. [OK] をクリックします。
3. [Analytics] → [管理] → [フォルダマネージャ] の順に選択します。
4. [ブラウザ] を選択します。
5. エクスポートする項目を選択します。

- 必要に応じて、[関連サポート項目のエクスポート] を選択します。
- [コンテナ・クラス名] には、必要に応じて完全修飾クラス名 (パッケージとクラス) を指定します。[コンテナ・クラス名] の指定がない場合は、名前の生成にコンテナ・クラスとエクスポート・ファイルの両方が使用されるようになります。
- [エクスポート] をクリックします。

11.5.4 Business Intelligence フォルダ項目のインポート

エクスポート済みのフォルダ項目をインポートするには、以下の手順を実行します。

- [Analytics]→[管理] をクリックして、[フォルダマネージャ] をクリックします。
- [サーバ・ディレクトリ] に、エクスポート済みの項目を格納しているディレクトリのフル・パスを入力します。または、そのネームスペースの既定のデータベースが含まれているディレクトリを基準とした相対ディレクトリ名を入力します。
- [ディレクトリ] タブをクリックします。このタブに、指定したディレクトリ内の項目のファイル名が表示されます。



- インポートする各ファイルの横にあるチェック・ボックスにチェックを付けます。
また、すべての項目を選択する場合は、チェック・ボックス列の一番上にあるチェック・ボックスにチェックを付けます。
- [インポート] をクリックします。
- プロンプトで [OK] をクリックして次に進みます。または、[キャンセル] をクリックします。

注釈 ファイルのインポート時に作成される項目については、InterSystems サービスを実行するユーザ名 (SYSTEM など) が所有者になります。

11.5.4.1 バリエーション : ローカル・ファイルのサーバへのインポート

ローカル・ファイルをサーバにインポートするには、以下の手順を実行します。

- [Analytics]→[管理] をクリックして、[フォルダマネージャ] をクリックします。
- [ブラウザ] を選択します。
- [ディレクトリ] タブをクリックして、[ファイルの選択] をクリックします。
- インポートするファイルを選択します。
- [インポート] をクリックします。
- プロンプトで [OK] をクリックして次に進みます。または、[キャンセル] をクリックします。

12

ダッシュボードで使用するポートレットの作成

ここでは、Business Intelligence の[実装](#)で使用するために、ダッシュボードにウィジェットとして追加できるポートレットを作成する方法を説明します。

12.1 ポートレットの基本

ポートレットを定義するには、以下のようにクラスを作成してコンパイルします。

- ・ `%DeepSee.Component.Portlet.abstractPortlet` をスーパークラスとして使用します。
- ・ `%DrawHTML()` メソッドを実装します。このメソッドは、ポートレットの本文を HTML として描画します。
このメソッドには、以下のシグニチャがあります。

```
method %DrawHTML()
```

追加のオプションについては [“設定の使用”](#) も参照してください。

- ・ 必要に応じて、`%OnGetPortletName()` メソッドを実装します。このメソッドは、[ウィジェット・ビルダ] ダイアログ・ボックスに表示するポートレットのローカライズ名を返します。

それ以外の場合は、短いクラス名がポートレット名となります。

このメソッドには、以下のシグニチャがあります。

```
classmethod %OnGetPortletName() as %String
```

- ・ 必要に応じて、`%OnGetPortletIcon()` メソッドを実装します。このメソッドは、[ウィジェット・ビルダ] ダイアログ・ボックスに表示するポートレットのアイコンの URL を返します。

それ以外の場合は、システムによって汎用アイコンが使用されます。

このメソッドには、以下のシグニチャがあります。

```
classmethod %OnGetPortletIcon() as %String
```

- ・ 必要に応じて、`%OnGetPortletSettings()` メソッドを実装します。このメソッドは、1 つ以上の構成可能な設定を返します。[“設定の定義と使用”](#) を参照してください。

それ以外の場合は、ポートレットに設定はありません。

- ・ 必要に応じて、adjustContentSize() メソッドを実装します。このメソッドは、ポートレットを含むウィジェットをロードまたは変更するたびに呼び出されます。このメソッドには、以下のシグニチャがあります。

```
ClientMethod adjustContentSize(load, width, height) [ Language = javascript ]
```

- ・ 必要に応じて、onApplyFilters() メソッドを実装します。このメソッドは、フィルタの変更をウィジェットに送信するたびに呼び出されます。このメソッドには、以下のシグニチャがあります。

```
ClientMethod onApplyFilters(refresh) [ Language = javascript ]
```

12.2 設定の定義と使用

構成設定を提供するポートレットは簡単に定義できます。そのためには、ポートレット・クラスの %OnGetPortletSettings() メソッドを実装します。このメソッドには次の 2 つの役割があります。

- ・ ダッシュボード・デザイナーで、このウィジェットの **[設定]** メニューに表示される設定を定義すること
- ・ ダッシュボードの URL を介してこれらの設定の値を受け取ることURL を介して値を渡す方法については、“[アプリケーションからダッシュボードへのアクセス](#)” を参照してください。

%OnGetPortletSettings() メソッドには、以下のシグニチャがあります。

```
classmethod %OnGetPortletSettings(Output pInfo As %List, ByRef pSettings) as %Status
```

pInfo は、以下のノードが含まれる多次元配列である必要があります。

ノード	値
pInfo(integer)	<p>以下のような、\$LISTBUILD によって返される一覧：</p> <pre>\$LB(name,default,type,caption,tooltip)</pre> <ul style="list-style-type: none"> ・ name は設定の論理名です。 ・ default は設定の既定値です。 ・ type は設定のタイプです。この後のサブセクションを参照してください。 ・ caption は設定のローカライズされたキャプションです。 ・ tooltip はオプションのツールチップです。

pSettings はこのメソッドに渡される多次元配列であり、URL を介して渡される設定の値を格納しています。詳細は、[2 番目のサブセクション](#)を参照してください。

12.2.1 設定のタイプ

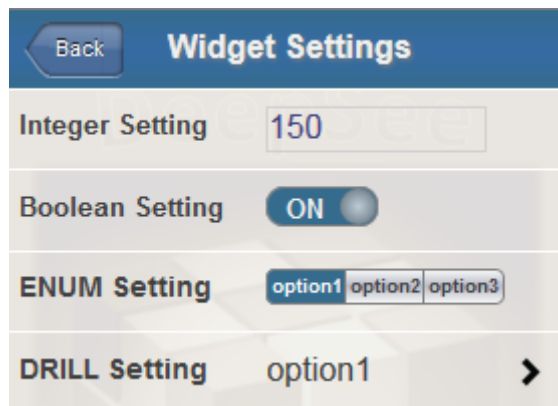
%OnGetPortletSettings() の pInfo 引数では、各設定のタイプを指定できます。これにより、ダッシュボード・デザイナーでその設定がどのように表示されるのが制御されます。以下のいずれかを使用します。

- ・ "%Integer"
- ・ "%Boolean"
- ・ "ENUM^caption1:value1,caption2:value2" または類似の形式。この文字列で、caption1 と caption2 はダッシュボード・デザイナーに表示するラベルであり、value1 と value2 は実際に使用される対応する値です。実際

には、このタイプの設定は 2 ～ 3 個のオプションしか提供できず、ダッシュボード・デザイナーではスペース不足のためにそれ以上のオプションを表示できなくなります。次の項目を参照してください。

- ・ "DRILL^caption1:value1,caption2:value2" または類似の形式。この文字列で、caption1 と caption2 はダッシュボード・デザイナーに表示するラベルであり、value1 と value2 は実際に使用される対応する値です。

次の図は、これらの各設定タイプの例を示しています。



以下の %OnGetPortletSettings() の実装は、これらの設定がどのように定義されたのかを示しています。

Class Member

```
ClassMethod %OnGetPortletSettings(Output pInfo As %List, ByRef pSettings) As %Status
{
  Kill pInfo
  set pInfo($I(pInfo)) = $LB("INTEGERSETTING","150","%Integer","Integer Setting","Sample integer setting")

  set pInfo($I(pInfo)) = $LB("BOOLEANSETTING","1","%Boolean","Boolean Setting","Sample boolean setting")

  set pInfo($I(pInfo)) = $LB("ENUMSETTING","150","ENUM^option1:150,option2:200,option3:200",
    "ENUM Setting","Sample ENUM setting")

  set pInfo($I(pInfo)) = $LB("DRILLSETTING","150",
    "DRILL^option1:150,option2:200,option3:200,option4:200,option5:200,option6:200,option7:200",
    "DRILL Setting","Sample DRILL setting")

  Quit pInfo
}
```

12.2.2 URL を介して渡される設定の受け取り

ダッシュボードの URL を介して、そのダッシュボード上の一部またはすべてのウィジェットに値を渡すことができます (任意のポートレット設定の値を含む)。これらの値を受け取るには、%OnGetPortletSettings() を実装する際に pSettings 引数を使用します。この引数は、URL を介して渡された設定の値が格納されている多次元配列です。この配列の構造は以下のとおりです。

ノード	値
pSettings("setting")。ここで、setting は設定の名前です。	その設定の値

1 つの方法として、**\$GET**(pSettings("setting")) を各設定の既定値として使用できます。次に例を示します。

```
ClassMethod %OnGetPortletSettings(Output pInfo As %List, ByRef pSettings) As %Status
{
    Kill pInfo
    Set pInfo($I(pInfo)) = $LB("LOGO",$G(pSettings("LOGO")),",","Clock logo","Logo displayed on top of clock")

    Set pInfo($I(pInfo)) = $LB("STEP",$G(pSettings("STEP"),"10"),"%Integer",
    "Second hand redraw interval (msec)","milliseconds steps of second hand")

    Set pInfo($I(pInfo)) = $LB("OFFSET",$G(pSettings("OFFSET"),"0"),"%Integer",
    "Offset from base time (min)","minutes difference from base time (Local or UTC)")

    Set pInfo($I(pInfo)) = $LB("UTC",$G(pSettings("UTC"),"0"),"%Boolean","UTC","Time Base: local (default) or UTC")

    Set pInfo($I(pInfo)) = $LB("CIRCLE",$G(pSettings("CIRCLE"),"1"),"%Boolean",
    "Circle","Shape: square (default) or circle")

    Set pInfo($I(pInfo)) = $LB("SIZE",$G(pSettings("SIZE"),"150"),"%Integer","Size","Size of the clock")

    Quit pInfo
}
```

12.2.3 設定の使用

ポートレット内の設定を使用するには、%DrawHTML() メソッドがポートレットの **settings** プロパティから設定値を抽出してから、それらの値をユーザーのニーズに応じた形で使用するように、このメソッドを定義します。ポートレットの **settings** プロパティは、以下の構造の多次元配列です。

ノード	値
settings ("setting")。ここで、setting は設定の名前です。	その設定の値

簡単な例として、%DrawHTML() は SIZE と呼ばれる設定の抽出を格納できます。

```
set size=$G(..settings("SIZE"))
```

その後このメソッドは、この値を使用してポートレットのサイズを設定できます。

12.3 例

以下に簡単な例を示します。

```
Class BI.Model.Custom.MyPortlet Extends %DeepSee.Component.Portlet.abstractPortlet
{
    /// Static HTML display method: draw the BODY of this component as HTML.
    Method %DrawHTML()
    {
        &html<<div class="portletDiv" style="overflow:hidden;">>
        &html<<div style="font-size:16px; border-bottom:1px solid gray;">My Widget</div>>

        Set tInfo(1) = $LB("Sales","UP","12")
        Set tInfo(2) = $LB("Costs","DOWN","-8")
        Set tInfo(3) = $LB("Profits","UP","18")

        &html<<table width="100%" cellpadding="0" border="0">>
        Set n = $O(tInfo(""))
        While (n!="") {
            Set tName = $LG(tInfo(n),1)
            Set tDir = $LG(tInfo(n),2)
            Set tPct = $LG(tInfo(n),3)
            Set clr = $S(tPct<0:"red",1:"black")
            Set bg = $S(n#2:"#FFFFFF",1:"white")
```



```

Set tPct = tPct _ "%"
&html<tr style="font-size:24px; background:#{bg}#;color:#{clr}#;">
  <td style="padding:4px;">#{tName}#</td>
  <td style="padding:4px;">#{tDir}#</td>
  <td style="padding:4px;text-align:right;">#{tPct}#</td></tr>>
Set n = $O(tInfo(n))
}
&html<</table>>
&html<</div>>
}
}

```

ウィジェットとして使用される場合、このウィジェットには以下のコンテンツがあります。

My Widget		
Sales	UP	12%
Costs	DOWN	-8%
Profits	UP	18%

この例では静的なデータが表示されていますが、ポートレットではリアルタイム・データを表示することも可能です。

設定も定義する、より複雑な例については、[サンプル](#)・クラス `BI.Model.PortletDemo.ClockPortlet` を参照してください。

13

Business Intelligence のその他の開発作業

ユーザのニーズやビジネス要件によっては、Business Intelligence の実装の一環として、ここで説明する開発作業の一部またはすべてを追加で行う必要がある場合があります。

13.1 用紙サイズの追加

ダッシュボード・ウィジェットを PDF ファイルに出力するときに、既定で用意されている一連の用紙サイズが提示され、ユーザはその中からサイズを選択できます。この一連のサイズを拡張するには、必要に応じて、以下のように `^DeepSee.PaperSizes` グローバルにノードを追加します。

ノード	値
<code>^DeepSee.PaperSizes(n)</code> 。ここで、 <code>n</code> は整数です。	<code>\$LISTBUILD(sizename,dimensions)</code> 。ここで、 <code>sizename</code> はサイズの名前です。 <code>dimensions</code> ではディメンジョンを指定します。 <code>dimensions</code> は以下のいずれかの形式である必要があります。 <code>widthxheight in</code> <code>widthxheight mm</code> <code>height</code> と単位名の間にはスペースを 1 つ記述する必要があります。

以下はその例です。

```
Set ^DeepSee.PaperSizes(1) = $LB("My Sticker","100x100 mm")
```

新しいサイズは直ちに使用可能になります。

13.2 ユーザ・アクティビティの監査

ユーザがクエリを実行したり、ダッシュボードにアクセスしたりするたびに、監査ログへの書き込みなど、カスタム・コードを実行することができます。

ユーザがクエリを実行したときに実行されるカスタム・コードを追加するには、以下の 1 回限りの設定手順を実行します。

- ・ カスタム・コードを含むクラス・メソッド、ルーチン、またはサブルーチンを記述します。最初のサブセクションには要件とオプションに関する詳細が記載されており、2 番目のサブセクションには例が示されています。

- ・ **^DeepSee.AuditQueryCode** を、そのメソッド、ルーチン、またはサブルーチンを実行する有効な ObjectScript 文を含む文字列と等しく設定します。

例えば、ターミナルで以下の操作を実行します。

```
set ^DeepSee.AuditCode="do ^MyBIAuditCode"
```

このネームスペースでクエリを実行するたびに、**^DeepSee.AuditQueryCode** に指定したコードが実行され、それによってルーチンまたはクラス・メソッドが呼び出されます。

同様に、ユーザがダッシュボードにアクセスした際に実行されるカスタム・コードを追加するには、以下のようにします。

- ・ カスタム・コードを含むクラス・メソッド、ルーチン、またはサブルーチンを記述します。
- ・ **^DeepSee.AuditCode** を、そのメソッド、ルーチン、またはサブルーチンを実行する有効な ObjectScript 文を含む文字列と等しく設定します。

このネームスペースでダッシュボードにアクセスするたびに、**^DeepSee.AuditCode** に指定したコードが実行されます。

13.2.1 監査コードの要件とオプション

いずれかのシナリオの監査コードを定義する際、コードによって現在のデバイスに出力が書き込まないようにします。また、コードによって InterSystems IRIS® で必要な % 変数が削除されないようにします。

コードでは、以下の変数を使用できます。

- ・ \$USERNAME — 現在のユーザの名前。
- ・ \$ROLES — 現在のユーザのロール。
- ・ %dsQueryText — 現在のクエリのテキスト。
- ・ %dsCubeName — 現在のクエリで使用するキューブの論理名。
- ・ %dsResultSet — %DeepSee.ResultSet の現在のインスタンス。必要に応じて、他の情報にアクセスするために使用できます。**%DeepSee.ResultSet** の使用方法の詳細は、“[Business Intelligence クエリのプログラムによる実行](#)”を参照してください。
- ・ %dsDashboard — アクセス中のダッシュボードの名前（該当する場合）。

一般的に、監査コードは出力をファイルまたはグローバルに書き込みます。

%dsQueryText、%dsCubeName、および %dsResultSet は、**^DeepSee.AuditQueryCode** を使用する監査ルーチンのみで使用できます。%dsDashboard は、**^DeepSee.AuditCode** を使用するルーチンのみで使用できます。

13.2.2 例

以下に簡単な監査ルーチンの例を示します。**^DeepSee.AuditQueryCode** で使用するサブルーチンが 1 つあり、**^DeepSee.AuditCode** で使用するサブルーチンが別に 1 つあります。

```
; this is the routine DeepSeeAudit
quit

dashboard
set auditentry="At "_$ZDT($H,3)_" , " _$USERNAME_" accessed dashboard: "_dsDashboard
set ^MyBIAuditLog($INCREMENT(^MyBIAuditLog))=auditentry
quit

query
set auditentry="At "_$ZDT($H,3)_" , " _$USERNAME_" ran query: "_dsQueryText
set ^MyBIAuditLog($INCREMENT(^MyBIAuditLog))=auditentry
quit
```

このルーチンを使用するには、以下の 2 行をターミナルで入力します。

```
SAMPLES>set ^DeepSee.AuditQueryCode="do query^DeepSeeAudit"
SAMPLES>set ^DeepSee.AuditCode="do dashboard^DeepSeeAudit"
```

監査ログを表示するには、ZWRITE を使用できます。以下に結果の例を示します (見やすくするために改行が追加されています)。

```
SAMPLES>zw ^MyBIAuditLog
^MyBIAuditLog=2
^MyBIAuditLog(1)="At 2014-06-20 16:26:38, SamSmith accessed dashboard: User Defined Listing.dashboard"
^MyBIAuditLog(2)="At 2014-06-20 16:26:38, SamSmith ran query: SELECT NON EMPTY {[MEASURES].[AMOUNT
SOLD],
[MEASURES].[UNITS SOLD]} ON 0,NON EMPTY [DATEOFSALE].[ACTUAL].[YEARSOLD].MEMBERS ON 1 FROM [HOLEFOODS]"
```

13.3 サーバ初期化コードの定義

サーバ初期化コードを定義する手順は以下のとおりです。

- ・ 有効な ObjectScript 文を ^DeepSee.InitCode グローバルに配置します。
例えば、ターミナルで以下の操作を実行します。

```
set ^DeepSee.InitCode="do ^myroutine"
```
- ・ コードによって現在のデバイスに出力が書き込まないようにします。
- ・ また、コードによって InterSystems IRIS で必要な % 変数が削除されないようにします。

このコードは、%DeepSee.Utils の %RunServerInitCode() メソッドによって呼び出されます。このメソッドは、InterSystems IRIS Business Intelligence セッションが作成されるたびに呼び出されます。

14

Business Intelligence のセキュリティの設定

Business Intelligence の実装プロジェクトでは、機能および Business Intelligence 項目へのアクセスを定義する必要があります。InterSystems IRIS® Business Intelligence には、基盤となるインターシステムズのセキュリティ・フレームワークに基づいた正規のメカニズムが備わっています。

この章では、ユーザが“承認ガイド”で説明されているインターシステムズのセキュリティを理解していることを前提としています。特に、リソース、ロール、およびユーザ間のリレーションシップを理解しているものとします。

注釈 InterSystems IRIS® を [最小のセキュリティ] オプションでインストールした場合（また、その後セキュリティを強化しなかった場合）、UnknownUser ユーザは **%All** ロールに属し、Business Intelligence のすべての部分にアクセスできます。その場合、このページは無視してください。

重要 また、Business Intelligence は Web アプリケーション内から使用する点にも注意してください。既定では、Web アプリケーションは、インターシステムズ・クラスのサブセットにアクセスできますが、これには **%DeepSee** クラスは含まれません。Web アプリケーションで Business Intelligence を使用するには、Analytics へのアクセスを明示的に有効化する必要があります。詳細は、“[Web アプリケーションの設定](#)”を参照してください。

14.1 セキュリティの概要

以下のテーブルは、Business Intelligence 内の要素の保護方法をまとめたものです。

要素	保護方法
Business Intelligence ユーザ・ポータル	%DeepSee_Portal および %DeepSee_PortalEdit リソース
アナライザ	%DeepSee_Portal、%DeepSee_Analyzer、および %DeepSee_AnalyzerEdit リソース
アーキテクト	%DeepSee_Portal、%DeepSee_Architect、および %DeepSee_ArchitectEdit リソース
フォルダ・マネージャとキューブ・マネージャ	%DeepSee_Portal および %DeepSee_Admin リソース
[MDX クエリツール] ページおよび [設定] ページ	%DeepSee_Portal、%DeepSee_Admin、および %Development リソース
[条件リスト・マネージャ] ページおよび [品質メジャー・マネージャ] ページ	%DeepSee_Portal および %DeepSee_PortalEdit リソース
@リスト・グループ・マネージャ	%DeepSee_ListingGroup、%DeepSee_ListingGroupEdit、および %DeepSee_ListingGroupSQL リソース
キューブ、サブジェクト領域、リスト、リスト・フィールド、リスト・グループ、KPI、フォルダ、その他のフォルダ項目（ダッシュボードやピボット・テーブルなど）	カスタム・リソース（オプション）
品質メジャー	品質メジャーのパブリッシュ先であるキューブのユーザのみがアクセス可能（追加のセキュリティなし）
条件リスト	セキュリティ・オプションなし

詳細は、このページで後述する “[Business Intelligence の一般的なタスクに関するセキュリティ要件](#)” を参照してください。

14.2 基本要件

ユーザが Business Intelligence を使用する場合、このページの残りの部分で示す他の要件に加えて、以下の要件に当てはまる必要があります。

- ・ ユーザは、Business Intelligence が使用されるデータベースにアクセスできる必要があります。
既定では、データベースを作成すると、InterSystems IRIS によって以下の操作が行われます。
 - データベース名 (%DB_database_name) に基づいた名前を持つリソースが作成されます。
 - そのリソースが新しいデータベースへのアクセスを制御するように設定されます。
 - そのリソースと同じ名前のロールが作成されます。そのロールは、リソースに対する読み取り/書き込み特権を持ちます。
読み取り/書き込み特権がパブリックであるかどうかを指定できます。これらの特権は既定でパブリックになっていません。

例えば、Business Intelligence で使用するために **MyApp** というデータベースを作成し、ここで説明したように InterSystems IRIS によってリソースとロールが作成されるようにするとします。また、読み取り/書き込み特権がパブリック

でないとしてします。その場合、Business Intelligence ユーザは、**%DB_MyApp** リソースに対する読み取り/書き込み特権を持つ **%DB_MyApp** ロールに属する必要があります。

- ・ DeepSee グローバルが別のデータベースからマップされている場合には、ユーザはそれらのグローバルが含まれているデータベースへもアクセスできる必要があります。

14.3 Business Intelligence の一般的なタスクに関するセキュリティ要件

以下のテーブルでは、[前のセクション](#)の項目に加え、一般的なタスクに関するセキュリティ要件を示します。

タスク	このタスクに対してユーザに付与されている必要がある特権*
ユーザ・ポータル(アナライザまたはミニ・アナライザ以外)の表示(ダッシュボードの作成は不可)	%DeepSee_Portal リソースに対する USE 許可
ユーザ・ポータル(アナライザまたはミニ・アナライザ以外)の表示(ダッシュボードの新規作成が可能)	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_PortalEdit リソースに対する USE 許可
ダッシュボードの表示(Excel へのエクスポートおよび PDF への出力を含む)	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ ダッシュボードに関連付けられたリソース(存在する場合)に対する USE 許可 (“モデル要素に対するセキュリティの追加” を参照) ・ ダッシュボードで使用されるピボット・テーブルに関連付けられたリソース(存在する場合)に対する USE 許可 ・ ダッシュボードおよびピボット・テーブルが格納されているフォルダに関連付けられたリソース(存在する場合)に対する USE 許可 ・ ピボット・テーブルで使用されるキューブまたはサブジェクト領域**に関連付けられたリソース(存在する場合)に対する USE 許可 ・ ダッシュボードで使用される KPI に関連付けられたリソース(存在する場合)に対する USE 許可 ・ KPI のクエリによって使用されるすべてのテーブルに対する SQL SELECT 特権 <p>ユーザが許可を有するすべてのウィジェットが表示されます。つまり、ダッシュボードは、ユーザが表示できない部分があるとしても表示されます。</p>
アナライザまたはミニ・アナライザへの読み取り専用アクセス	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_Analyzer リソースに対する USE 許可
アナライザまたはミニ・アナライザへの完全なアクセス	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_AnalyzerEdit リソースに対する USE 許可

タスク	このタスクに対してユーザに付与されている必要がある特権*
リストの表示	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ リストに関連付けられたリソース (存在する場合) に対する USE 許可 ・ リストで使用されるすべてのソース・テーブルに対する SQL SELECT 権限と、そのキューブに対して生成された CubeClass.Listing テーブルに対する SELECT 権限カスタム・リストで \$\$\$RESTRICT トークンを使用する場合、CubeClass.Listing テーブルに対する SELECT 権限が必要です。
アナライザでの既存のピボット・テーブルの変更	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_AnalyzerEdit リソースに対する USE 許可 ・ 指定ピボット・テーブルに関連付けられたリソース (存在する場合) に対する USE 許可および WRITE 許可 ・ ピボット・テーブルがあるフォルダに関連付けられたリソース (存在する場合) に対する USE 許可 ・ ピボット・テーブルで使用されるキューブ**またはサブジェクト領域に関連付けられたリソース (存在する場合) に対する USE 許可
ダッシュボードの新規作成	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_PortalEdit リソースに対する USE 許可 ・ ダッシュボードがあるフォルダに関連付けられたリソース (存在する場合) に対する USE 許可
既存のダッシュボードの変更	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_PortalEdit リソースに対する USE 許可 ・ 指定ダッシュボードに関連付けられたリソース (存在する場合) に対する USE 許可および WRITE 許可 ・ ダッシュボードがあるフォルダに関連付けられたリソース (存在する場合) に対する USE 許可
アーキテクトへの読み取り専用アクセス	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_Architect リソースに対する USE 許可
アーキテクトでのキューブまたはサブジェクト領域の新規作成	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_ArchitectEdit リソースに対する USE 許可
アーキテクトでの既存のキューブまたはサブジェクト領域の変更	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_ArchitectEdit リソースに対する USE 許可 ・ 指定したキューブまたはサブジェクト領域に関連付けられたリソース (存在する場合) に対する USE および WRITE 許可 (“モデル要素に対するセキュリティの追加” を参照)

タスク	このタスクに対してユーザに付与されている必要がある特権*
<ul style="list-style-type: none"> ・ [フォルダマネージャ] ページ ・ [MDX クエリツール] ページ ・ [設定] ページ 	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_Admin リソースに対する USE 許可、または %Development リソースに対する USE 許可
<ul style="list-style-type: none"> ・ [条件リスト・マネージャ] ページ ・ [品質メジャー] ページ 	<ul style="list-style-type: none"> ・ %DeepSee_Portal リソースに対する USE 許可 ・ %DeepSee_PortalEdit リソースに対する USE 許可
リスト・グループ・マネージャ(読み取り専用アクセス)	%DeepSee_ListingGroup リソースに対する USE 許可
リスト・グループ・マネージャ(編集アクセス、カスタム SQL クエリ・オプションを除く)	%DeepSee_ListingGroupEdit リソースに対する USE 許可
リスト・グループ・マネージャ(編集アクセス、カスタム SQL クエリ・オプションを含む)	<ul style="list-style-type: none"> ・ %DeepSee_ListingGroupEdit リソースに対する USE 許可 ・ %DeepSee_ListingGroupSQL リソースに対する USE 許可

*[前のセクション](#)も参照してください。リソース定義で、これらの許可の一部をパブリックにできます。例えば、最小セキュリティ・インストールでは、既定で、USE 許可はすべての Business Intelligence リソースに対してパブリックになります。

**あるキューブに他のキューブへのリレーションシップがある場合、それらのキューブは個別に保護されます。リレーションシップを使用するには、ユーザにそれらのすべてに対する USE 許可が必要です。同様に、複合キューブは複数のキューブで構成され、それぞれ個別に保護されます。

14.4 モデル要素に対するセキュリティの追加

キューブ、サブジェクト領域、KPI、ピボット・テーブル、ダッシュボード、リスト、またはリスト・フィールドに対してセキュリティを追加する手順は以下のとおりです。

1. 管理ポータルでリソースを作成します。[リソース] ページを使用します ([システム管理] > [セキュリティ] > [リソース] を選択します)。
2. 管理ポータルでロールを作成します。[ロール] ページを使用します ([システム管理] > [セキュリティ] > [ロール] を選択します)。このロールは、作成したリソースに対する USE および WRITE 許可を持つ必要があります。

または、USE および WRITE 許可を持つ 1 つのロールと、USE 許可のみを持つもう 1 つのロールを作成することもできます。

3. 以下のように、リソースを Business Intelligence 項目に関連付けます。
 - ・ ダッシュボードまたはピボット・テーブルの場合は、その項目を保存する際に、該当するリソースの名前を [リソースへのアクセス] フィールドに入力します。

“[ダッシュボードまたはピボット・テーブルのリソースの指定](#)” も参照してください。

ダッシュボードまたはピボット・テーブルを保存するには、適切な Business Intelligence ユーザ・インタフェース・コンポーネントに対する USE 特権および WRITE 特権が付与されていることも必要です。詳細は、[前の見出し](#)を参照してください。

- ・ キューブ、サブジェクト領域、またはリスト・フィールドの場合は、アーキテクトを使用してその項目を保護するリソースを指定します。
 - ・ キューブ定義で定義されたリストの場合は、アーキテクトを使用してその項目を保護するリソースを指定します。
 - ・ リスト・グループの場合やリスト・グループで定義されたリストの場合は、リスト・グループ・マネージャを使用してその項目を保護するリソースを指定します。
 - ・ KPI の場合は、スタジオでクラス定義を編集します。適用可能なリソースの名前を、RESOURCE クラス・パラメータの値として使用します。
4. 必要に応じて、ユーザをロールに割り当てます。

14.5 ダッシュボードまたはピボット・テーブルのリソースの指定

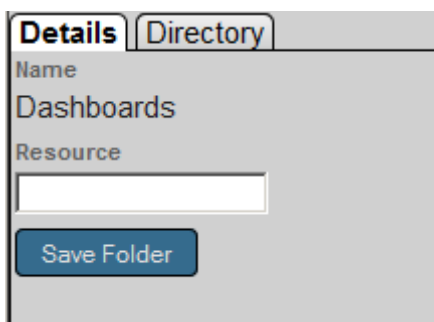
ダッシュボードまたはピボット・テーブルのリソースを指定するには、その項目を保存するときに [リソースへのアクセス] フィールドを指定します。以下のいずれかの場合に、これを実行できます。

- ・ 項目に所有者が存在しない (所有者は [所有者] フィールドで指定されています)。
- ・ ユーザ自身がその項目の所有者である。
- ・ ユーザが **%DeepSee_Admin** リソースに対する USE 許可を持っている。

14.6 フォルダのリソースの指定

フォルダのリソースを指定する手順は以下のとおりです。

1. [InterSystems ランチャー] をクリックし、[管理ポータル] をクリックします。
セキュリティの設定によっては、InterSystems IRIS ユーザ名とパスワードを使用してログインするように求められます。
2. 以下のように、適切なネームスペースに切り替えます。
 - a. [変更] をクリックします。
 - b. ネームスペースをクリックします。
 - c. [OK] をクリックします。
3. [Analytics]→[管理]→[フォルダマネージャ] をクリックします。
4. フォルダの横にあるチェック・ボックスにチェックを付けます。
5. 左の領域で [詳細] タブをクリックします。



The screenshot shows a web interface with two tabs: 'Details' and 'Directory'. The 'Details' tab is active. It contains a 'Name' label followed by the text 'Dashboards'. Below this is a 'Resource' label followed by an empty text input field. At the bottom of the form is a blue button labeled 'Save Folder'.

6. リソースの名前を入力します。
7. [フォルダを保存] をクリックします。

A

キューブ・バージョンの使用

ここでは、キューブ・バージョン機能の使用法を説明します。この機能を使用すると、実行中クエリのわずかな中断しか伴わずに、キューブ定義の変更、構築、ユーザへの提供を行うことができます。キューブ・バージョンは、Business Intelligence の実装で利用できるオプションの機能です。

この機能を使用するには、キューブあたり 2 倍のディスク領域が必要です。また、スタジオでキューブ・クラスを編集することも必要になります。

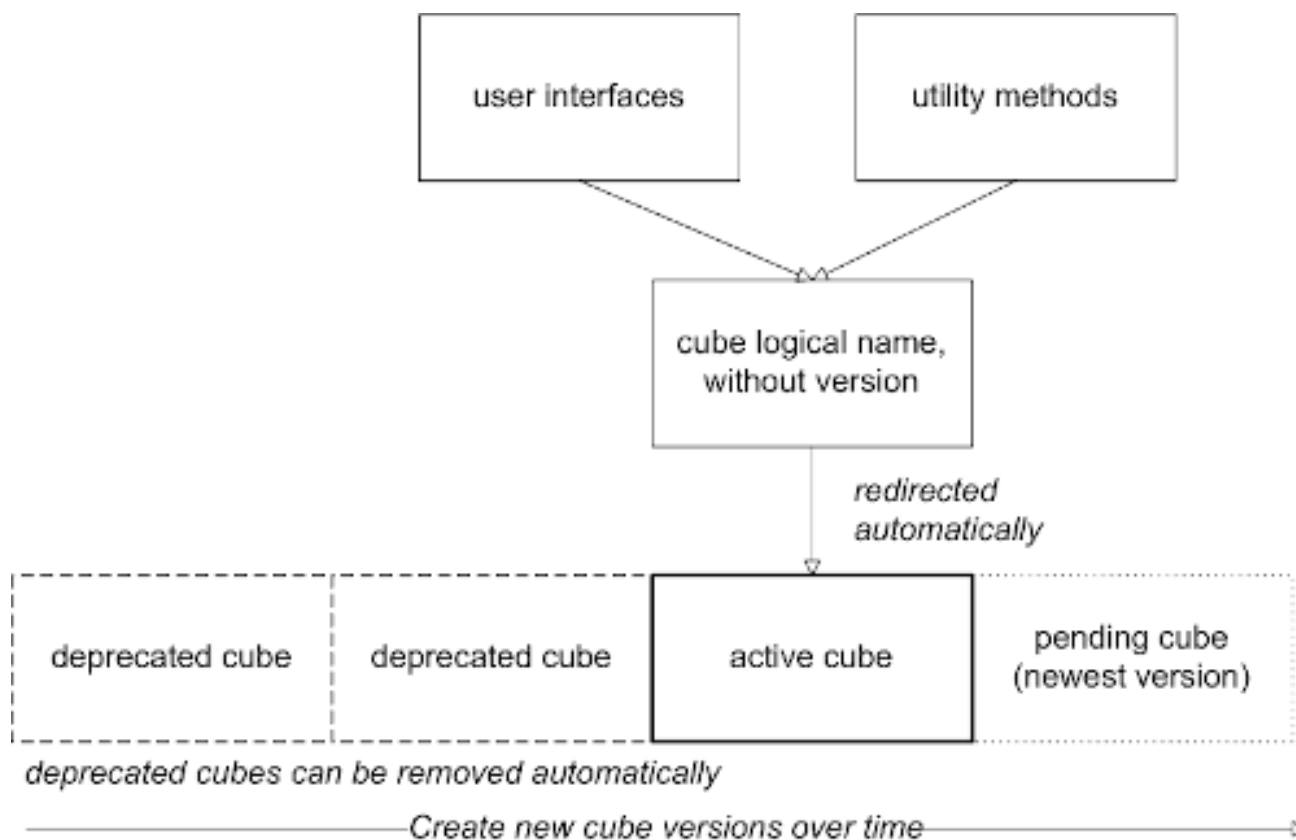
注釈 形式的に共有されるディメンジョンを定義するキューブの場合、キューブ・バージョン機能はサポートされません。また、単方向リレーションシップを定義するキューブの場合もサポートされません。この機能は、双方向リレーションシップを定義するキューブでは使用できます。

A.1 キューブ・バージョン機能の概要

キューブ・バージョン機能を使用すると、実行中クエリのわずかな中断しか伴わずに、キューブ定義の変更、構築、ユーザへの提供を行うことができます。この機能は、以下のように実行されます。

- ・ 指定したキューブ定義にバージョンを付与できます。
- ・ システムは、キューブ・バージョンごとにバージョン固有のファクト・テーブルとディメンジョン・テーブルを生成します。
- ・ どの時点でも、1 つのキューブ・バージョンのみがアクティブになります。ユーザ・インタフェースとすべての生成されたクエリではこのバージョンが使用されます。
- ・ 最新のキューブ・バージョンを使用可能にするには、そのバージョンをアクティブ化する必要があります。アクティブ化した時点で、システムは一時的にどのクエリも実行できないようにしてから、最新のバージョンに切り替えます。

以下の図は、このプロセス全体を示しています。



キューブの論理名はアクティブ・キューブに自動的にリダイレクトされます。アナライザと他のユーザ・インタフェースではキューブの論理名のみが使用されるため、アクティブ・キューブのみが認識されます。同様に、%DeepSee.Utils 内のメソッドを使用するときに、バージョン番号なしでキューブの論理名を指定すると、そのメソッドはアクティブ・キューブに対して実行されます。

キューブ・バージョン番号を(スタジオで)更新して、リコンパイルすると、保留キューブが作成されます。その後ユーザはこのキューブを構築できます。準備ができたなら、ユーティリティ・メソッドを使用してこのキューブをアクティブ化します。これにより、保留キューブがアクティブになり、直前までアクティブであったキューブが使用されなくなります。

既定では、このアクティブ化プロセスによってその使用されなくなったキューブが自動的に削除されます。キューブ・バージョン機能の目的は、バージョン間の切り替えを支援することではありません。

最適な方法は、ソース・コントロールを使用することです。キューブ・バージョン機能はソース・コントロールの代替となるものではありませんが、ソース・コントロールと組み合わせると役に立つことがあります。

A.1.1 キューブの最新状態の維持

あるキューブでキューブ・バージョン機能が使用されている場合、そのキューブのアクティブ・バージョンを構築することはできません。すなわち、\$SYSTEM.DeepSee.BuildCube() メソッドはアクティブ・バージョンには適用されず、代わりにエラーが返されます。アーキテクトの **[ビルド]** オプションも同じ動作を示します。これらの操作はブロックされます。その理由は、これらの操作によって実行中のクエリが長時間にわたって中断されますが、この機能の目的はそのような中断を防止することだからです。

そのキューブを同期できます。

A.1.2 クエリ中断の原因となり得るモデルの変更

キューブ・バージョン機能は、アクティブ・キューブ上で正常に機能しているクエリが保留キューブ上でも正常に機能するかどうかをチェックしません。例えば、アクティブ・キューブで定義されたモデル要素が保留キューブに含まれなくなった

場合、保留キューブをアクティブ化すると、その要素を使用するクエリはすべて機能しなくなります。このため、ユーザー側で、中断の原因となる可能性のあるモデル変更を特定して、そのような変更を適切に扱う必要があります。

A.2 バージョンをサポートするためのキューブの変更

キューブでキューブ・バージョン機能がサポートされるようにキューブを変更するには(および初期バージョンを作成してアクティブ化するには)、以下の手順を実行します。

重要 キューブ・バージョンに移行しようとしており、かつこの機能を使用しない既存のキューブがあり、かつどのクエリも中断させたくない場合は、この注意事項をお読みください。

キューブ・バージョンに移行する場合、そのためのプロセスは最初のキューブ・バージョンでは異なります。具体的には、最初のキューブ・バージョンは現在使用されているキューブ(バージョン管理されていないキューブ定義)とランタイム上の互換性を備えている必要があります。すなわち、最初のキューブ・バージョンによって、バージョン管理されていないキューブ定義と比較して、どのメジャーやレベルも削除および再定義されてはいけません。要素を追加することはできます。そのことは既存のクエリに影響を与えないからです。

1. キューブ・クラスに以下のパラメータを追加します。

```
Parameter USECUBEVERSIONS=1;
```

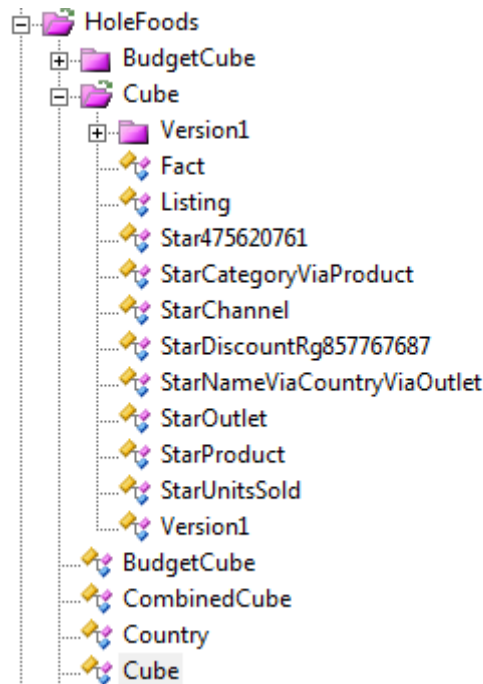
この変更と次の変更を加えるには、スタジオを使用する必要があります。

2. 以下の属性を <cube> 要素に追加して、クラスを保存します。

```
version="versionnum"
```

ここで、versionnum は整数です。

3. クラスをコンパイルします。この時点で、システムによってこのキューブ用に生成されたパッケージには、新しいサブパッケージ (Versionversionnum という名前) が含まれています。例を以下に示します。



この例では、新しいパッケージは **HoleFoods.Cube.Version1** です。

HoleFoods.Cube.Fact、**HoleFoods.Cube.Listing**、**HoleFoods.Cube.Star475620761** などのクラスは以前は存在していましたが、これらは USECUBEVERSIONS の追加前にこのキューブ用に生成されました。キューブ・バージョン・ユーティリティではこれらのキューブ定義は扱われません。

4. 必要に応じてキューブ定義に変更を加えます。このセクションの冒頭にある重要な注意事項を読んで、どのような変更を加えるのかを決定します。変更内容を保存します。
5. キューブを構築します。この手順は、実行中のクエリには影響を与えません（このセクション冒頭の重要な注意事項に記載されたガイドラインに従っている場合は、これより前の手順も実行中のクエリには影響を与えません）。

ターミナルでキューブを構築する場合は、少し異なる出力が表示されて、特定のキューブ・バージョンを構築していることが示されます。次に例を示します。

```
Building cube [HOLEFOODS:1]
```

6. ターミナルで、**%DeepSee.CubeVersion.Utils** クラスの **%ActivatePendingCubeVersion()** メソッドを実行します。このメソッドは、構築するキューブの名前である 1 つの引数を取ります（バージョン番号なしで）。次に例を示します。

ObjectScript

```
d ##class(%DeepSee.CubeVersion.Utils).%ActivatePendingCubeVersion("holefoods")
```

このメソッドは、以下のような出力を表示します。

```
Pending version for holefoods: 1
Pending version synchronized: HOLEFOODS:1
Queries locked for cube: holefoods
Killing active tasks for cube: holefoods
Cube version activated: HOLEFOODS:1
Removing non-versioned cube data
```

このメソッドの 1 つのステップが原因で、このキューブに対してクエリを一時的に実行できなくなりますが、ほとんどの場合はユーザ側では実際の遅延は感じられません。

この時点で、すべてのユーザに新しいバージョンのキューブが表示されます。

7. キューブ・マネージャを使用してこのキューブを更新する場合は、キューブの更新計画が **[同期のみ]** または **[手動]** であることを確認します。“[キューブの最新状態の維持](#)”を参照してください。

A.2.1 キューブのバージョンとリレーションシップ

リレーションシップに含まれるキューブでキューブ・バージョン機能を使用できます。ルールは以下のとおりです。

- ・ すべてのリレーションシップは、一方向ではなく双方向である必要があります。
- ・ それぞれの関連キューブでもキューブ・バージョンを指定する必要があります。
- ・ バージョンを更新して、新しいバージョンを構築して、いずれかのキューブの新しいバージョンをアクティブ化する際は、すべての関連キューブに対して同じ操作を実行する必要があります。
- ・ 関連キューブをアクティブ化する順序は、それらのキューブを構築する順序と同じである必要があります。“[InterSystems Business Intelligence の上級モデリング](#)”の“[関連キューブの構築順序の決定](#)”を参照してください。

A.2.2 %ActivatePendingCubeVersion() の詳細

%ActivatePendingCubeVersion() メソッドには、以下のシグニチャがあります。

```
ClassMethod %ActivatePendingCubeVersion(pCubeGenericName As %String,  
                                         pRemoveDeprecated As %Boolean = 1,  
                                         pVerbose As %Boolean = 1) As %Status
```

以下は、この指定の説明です。

- ・ pCubeGenericName は、キューブの名前です（バージョン番号は含まない）。この引数では大文字と小文字は区別されません。
- ・ pRemoveDeprecated は、このメソッドによって、現在は使用されていないキューブ・バージョンも削除するかどうかを指定します。この引数の値が 1 の場合は、このメソッドによって、現在は使用されていないキューブ・バージョンのファクト・テーブルとそのデータ、ディメンジョン・テーブルとそのデータ、すべてのキャッシュされたデータ、およびすべての内部使用メタデータが削除されます。

このメソッドを初めて使用する際は、バージョン管理されていないキューブからの移行時に、そのバージョン管理されていないキューブでファクト・テーブルなどに格納されているデータが削除されます。バージョン管理されていない生成済みクラスは、システムで必要なため削除されません。

- ・ pVerbose は、このメソッドの処理ステージを示すメッセージを表示するかどうかを指定します。

A.3 キューブ・バージョンの更新

重要

最初のキューブ・バージョンをまだアクティブ化していない場合は、[前のセクション](#)を参照してください。最初のキューブ・バージョンをコンパイルする際は、そのキューブに何らかの変更を加えると、そのキューブをアクティブ化する前であっても、実行中のクエリが影響を受けます。したがって、バージョン管理されていないキューブとランタイム上の互換性があるいずれかのバージョンのキューブをコンパイル、構築、およびアクティブ化する必要があります。このことの意味については、[前のセクション](#)を参照してください。

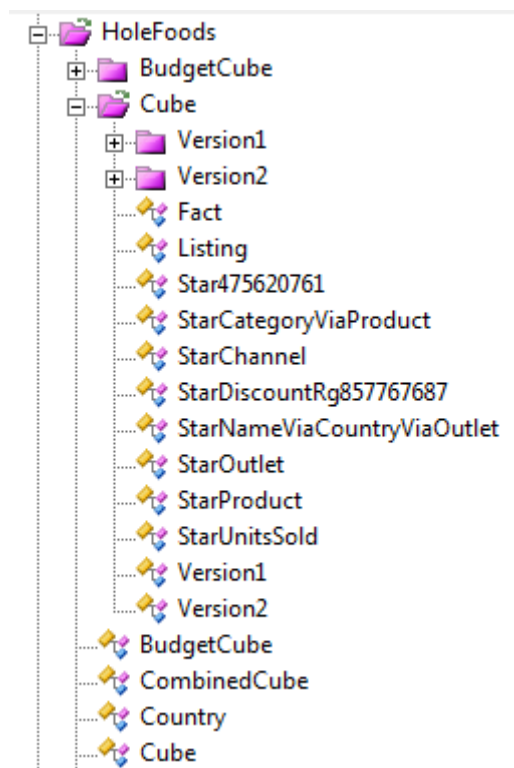
キューブを既に変更して初期バージョンを作成した場合は、以下の手順に従ってそのキューブを更新してください。

1. まず、<cube> 要素内で、新しいバージョン番号を使用するようにキューブ・クラスを変更します。これにより、キューブに対する変更が本来より早い段階で表示されることが防止されます。（表示名などの一部のキューブ変更内容は、キューブをコンパイルした時点で有効になります。“InterSystems Business Intelligence のモデルの定義”の["リコンパイルおよび再構築のタイミング"](#)を参照してください。）
2. キューブ・クラスを保存します。
3. 希望に応じてキューブに変更を加えて、それらの変更内容を保存します。

実働システムの場合は、まず別のシステム上でこれらの変更内容をテストしてください。

4. キューブをコンパイルします。

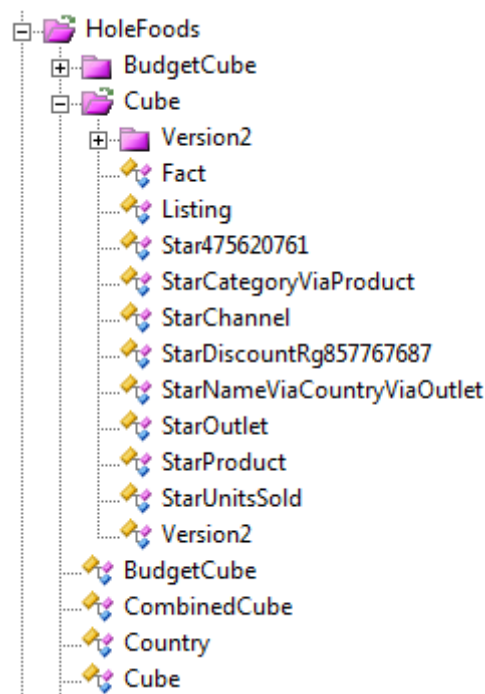
この時点で、システムによってこのキューブ用に生成されたパッケージには、新しいバージョン番号を持つ別の新しいサブパッケージが含まれています。次に例を示します。



5. キューブを構築します。
6. ターミナルで、`%DeepSee.CubeVersion.Utils` クラスの `%ActivatePendingCubeVersion()` メソッドを実行します。この場合は、このメソッドは以下のような出力を表示します。

```
Pending version for holefoods: 2
Pending version synchronized: HOLEFOODS:2
Queries locked for cube: holefoods
Killing active tasks for cube: holefoods
Cube version activated: HOLEFOODS:2
Deprecating previously active version: HOLEFOODS:1
Removing previously active version: HOLEFOODS:1
```

この時点で、システムによってこのキューブ用に生成されたパッケージには、新しいバージョン番号を持つサブパッケージのみが含まれています。次に例を示します。



この時点で、すべてのユーザに新しいキューブが表示されます。

注釈 バージョニング機能を使用するキューブに基づいて、サブジェクト領域を定義できます。ベース・キューブでの変更と同様に、キューブ・バージョンを変更する際には、正しく機能するようにサブジェクト領域をリコンパイルする必要があります。

A.4 操作対象のキューブの指定

キューブ・バージョンを使用する場合は、以下の方法で操作対象のキューブを指定できます。

- アナライザまたは MDX クエリ・ツールで手動クエリを作成する際は、以下のいずれかの形式のキューブ名を使用できます。
 - 論理キューブ名。この場合は、そのクエリではアクティブ・バージョンのキューブが使用されます。
 - `cubename:versionnum` という形式。ここで、`cubename` は論理キューブ名であり、`versionnum` はバージョン番号です。この場合は、そのクエリでは指定されたバージョンが使用されます。

- アナライザやキューブ・マネージャなどのユーザ・インタフェースでは、アクティブ・バージョンのみを操作できます（ただし例外として上の項目で述べたケースは除きます）。

これらのユーザ・インタフェースでは、バージョン情報が含まれていないキューブ・キャプションのみが表示されます。

また、変更内容を保存すると、手動クエリにバージョン番号を入力した場合を除いて、保存されたデータには論理キューブ名のみが含まれます（バージョン番号は含まれない）。既定では、ピボット・テーブルとリスト・グループの定義にはバージョン番号は含まれません。

- キューブ名を引数として指定できる `%DeepSee.Utils` 内のメソッドを使用する場合は、論理キューブ名または `cubename:versionnum` という形式のキューブ名を使用できます。
- MDX シェルでは、論理キューブ名または `cubename:versionnum` という形式のキューブ名を使用できます。シェル内でトレースが有効になっている場合は、シェルにはキューブ・バージョン番号が表示されます。

A.5 追加のオプション

%DeepSee.CubeVersion.Utils クラスでは、デバッグ目的に使用できる追加のメソッドが提供されています。以下はその概要です。

- ・ %GetVersionedCubeName()
- ・ %DeprecateCubeVersion()
- ・ %SetPendingCubeVersion()
- ・ %RemoveCubeVersion()

詳細は、%DeepSee.CubeVersion.Utils のクラスリファレンスを参照してください。

また、%DeepSee.Utils の %BuildCube() は、参照によって、アクティブ・バージョン番号が含まれたキューブ名を返すことができます。以下はその例です。

```
SAMPLES>set cubename="patients"

SAMPLES>set status=##class(%DeepSee.Utils).%BuildCube(.cubename)

Building cube [PATIENTS:1]
Existing cube deleted.
Fact table built:          1,000 fact(s) (2 core(s) used)
Fact indices built:        1,000 fact(s) (2 core(s) used)

Complete
Elapsed time:              0.461454s
Source expression time:    0.298187s

SAMPLES>w cubename
PATIENTS:1
```

\$SYSTEM.DeepSee.BuildCube() メソッドにはこのオプションはありません。

A.5.1 キューブ・バージョン機能の無効化

指定したキューブのバージョンを無効にする手順は以下のとおりです。

1. キューブ・クラスを編集して、USECUBEVERSIONS の値として 0 を指定します。
2. クラスを保存して、コンパイルします。
3. キューブを構築します。
4. 希望に応じて、不要になったキューブ・バージョンを削除します。ターミナルで以下のコマンドを実行します。

ObjectScript

```
set status=##class(%DeepSee.CubeVersion.Utils).%RemoveCubeVersion(cubename,version)
```

ここで、cubename は論理キューブ名であり、versionnum はバージョン番号です。

アクティブ・バージョンを削除しようとした場合は、このメソッドはエラーを返します。

この時点から、このキューブはバージョン管理されていないキューブと同じ動作を示します。

B

Analytics エンジンの仕組み

ここでは、Analytics エンジンが MDX クエリを実行する仕組みについて説明します。この情報は、Business Intelligence の実装時または実装後に、クエリ・プランを表示する場合や問題を診断する場合に役立ちます。

重要 ここでは、内部的に使用されるグローバルに関する情報を提供します。この情報はデモンストレーションを目的としたものであり、これらのグローバルを直接使用することはできません。これらのグローバルの編成は、予告なしに変更される場合があります。

B.1 概要

このセクションでは、基本概念について説明します。詳細は、[次のセクション](#)を参照してください。

B.1.1 ビットマップ・インデックスの使用

キューブ・クラスのコmpイル時に、Analytics エンジンは、ファクト・テーブル・クラスを作成して、そのテーブルを使用します。このクラスは、このエンジンが必要とするすべてのビットマップ・インデックスを定義します。これらのビットマップ・インデックスは、`^DeepSee.Index` グローバルに格納されます。キューブのビルド時または同期時に、これらのインデックスが必要に応じて更新されます。ファクト・テーブルでレコードを検索する必要がある場合、このエンジンは必要に応じてこれらのビットマップ・インデックスを組み合わせ使用します。

例えば、1 つのビットマップ・インデックスが、Product Category レベルの Snack メンバに関係するすべてのレコードへのアクセスを提供するとします。もう 1 つのビットマップ・インデックスが、City レベルの Madrid メンバに関係するすべてのレコードへのアクセスを提供するとします。さらにもう 1 つのビットマップ・インデックスが、YearSold レベルの 2012 メンバに関係するすべてのレコードへのアクセスを提供するとします。Snack、Madrid、および 2012 に関係するすべてのレコードを探し出すために、エンジンはこれらのビットマップ・インデックスを組み合わせ、得られたインデックスを使用してレコードを取得します。

B.1.2 キャッシュ処理

(既定で) 512,000 レコードより多く使用するキューブの場合、Analytics エンジンは、結果キャッシュを保持して使用します。この場合、エンジンは、MDX クエリを実行するたびに結果キャッシュを更新して、その後で可能な限りこの結果キャッシュを使用します。結果キャッシュには以下のグローバルが含まれています。

- ・ `^DeepSee.Cache.Results` には、指定のキューブに対して以前に実行された各クエリの値が格納されます。このグローバルには、これらのクエリに関するメタ情報も格納されます。このメタ情報を使用してクエリを迅速に再実行

できます。クエリの情報を取得するために、エンジンはキューブ名とクエリ・キー (正規化されたクエリ・テキストのハッシュ) を使用します。

指定されたキューブ名およびクエリ・キーについて、このグローバルには最終値と中間値が格納されたサブノードのセットが含まれています。これらのサブノードは、バケット番号別に分類され、さらに結果セル別に分類されています (バケットは、ソース・テーブル内の連続するレコードのセットです。[次のサブセクション](#)を参照してください)。

以下に例を示します。

```
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",-1,2,3)=67693.46
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",-1,2,4)=425998.02
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",-1,2,5)=212148.68
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",0,2,3)=301083.77
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",0,2,4)=1815190.08
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",0,2,5)=910314.95
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",1,2,3)=78219.74
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",1,2,4)=463165.12
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",1,2,5)=233031.39
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",2,2,3)=79153.44
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",2,2,4)=461472.97
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",2,2,5)=233584.42
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",3,2,3)=76017.13
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",3,2,4)=464553.97
^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",3,2,5)=231550.46
```

この例では、"data" に続く最初の添え字はバケット番号を示しています。バケット -1 および 0 は特殊なバケットです。-1 バケットはアクティブ・バケット (最新のレコードを表します) で、0 バケットはすべてのバケットにわたる統合結果です。

最後の 2 つの添え字は、結果セルをその位置で示しています。このノードの値は、指定された結果セルの値です。

例えば、`^DeepSee.Cache.Results("HOLEFOODS","en2475861404","data",0,2,3)` には、すべてのバケットにわたるセル (2,3) の統合値が格納されています。この数値は、他のノードに格納されているこのセルの中間値の合計です。

- ・ `^DeepSee.Cache.Axis` には、以前に実行したクエリの軸に関するメタデータが含まれています。エンジンは、指定されたクエリの軸を反復処理する必要があるときはいつでも、この情報を使用します。キャッシュしたデータは格納されません。
- ・ `^DeepSee.Cache.Cells` (以前に実行したクエリで返されたセルのキャッシュされたメジャー値が格納されています)。セルは、メジャーではない任意の数のメンバが交差する位置にある共通部分です (Madrid、Snack、および 2012 の共通部分など)。このグローバルでは、各セルはセル仕様で表されます。セル仕様は、専用のコンパクトな内部使用の式です。以下は、部分的な例です。

```
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:1:1",1)=$1b(1460.05)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:1:2",1)=$1b(606.22)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:1:3",1)=$1b(40.17)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:1:4",1)=$1b(63.72)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:2:1",1)=$1b(3778)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:2:2",1)=$1b(1406.08)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:2:3",1)=$1b(117.31)
^DeepSee.Cache.Cells("HOLEFOODS",1,":::2012:::::1:1:2:4",1)=$1b(412.24)
```

最初の添え字はキューブ名、2 番目の添え字はバケット番号、3 番目の添え字はセル仕様 (例えば "`:::2012:::::1:1:1:1`") です。最後の添え字はメジャーを示しています。指定されたノードの値は、指定されたキューブ、セル、およびバケットで指定されたメジャーの集約値です。この場合、内部処理の都合上、結果は \$LISTBUILD 形式で表現されます。このグローバルはクエリ・キーを使用しません。これは、複数のまったく異なるクエリによって同じセルが簡単に生成されるためです。

このグローバルは、セル・キャッシュと呼ばれ、キャッシュでバケットを使用する場合にのみ生成されます。

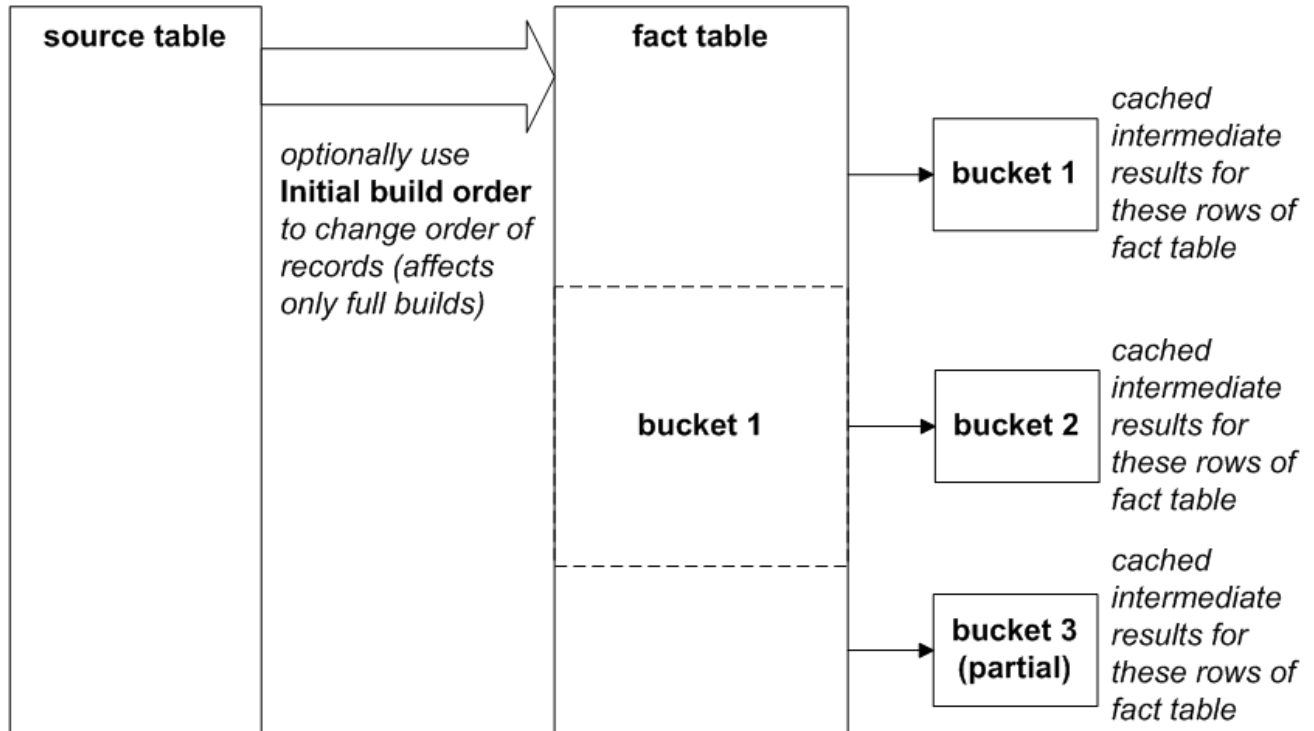
セル・キャッシュには、アクティブ・バケットの値は含まれていません。また、(すべてのバケットにわたって統合された) 0 バケットの値も含まれていません。

ユーザがクエリを実行しない限り、これらのグローバルには値が生成されません。クエリの繰り返し実行に伴って、キャッシュの内容は充実していきます。その結果、クエリを再実行する代わりにキャッシュを使用できるようになるため、パフォーマンスが向上します。

キャッシュには、`isReference="true"` と定義されているプロパティの値は含まれていません。これらの値は常に、実行時に取得されます。

B.1.3 バケット

(既定で) 512,000 レコードより多く使用するキューブの場合は、キャッシュがバケットにまとめられます。各バケットは、以下の図のように、ファクト・テーブル内で連続する大量のレコードに対応しています。



最後のバケット (部分的なバケット) はアクティブなバケットであり、セル・キャッシュには示されません。

既定で、ファクト・テーブルには、ソース・テーブルと同じ順序でレコードが格納されます。キューブの [初期ビルド順] を指定して、エンジンがキューブの完全ビルドを実行するときにソース・テーブルのレコードを検証する順序を制御できます。“InterSystems Business Intelligence のモデルの定義” の “他のキューブ・オプション” を参照してください。

同期または再構築することでキューブを更新した場合、あるいは **手動で更新した後に明示的に呼び出した場合**、エンジンにより必要に応じてキャッシュの一部が破棄されます。より具体的には、ファクト・テーブルにあるバケットのうち、影響を受ける部分のレコードを使用するバケットが無効になります。その他のバケットはそのままです。クエリの実行時には、有効なバケットに対してのみキャッシュされたデータが使用されます。キャッシュされた有効な結果が存在しないレコードの場合、エンジンはビットマップ・インデックスを使用して必要な中間値を再計算します。クエリ実行の最後のフェーズとして、エンジンは結果を統合します。したがって、エンジンは、キャッシュされたデータと新規データまたは変更されたデータとの組み合わせから得られた結果を提供できます。また、エンジン動作の中にはバケット単位で分割できるものがあるので、エンジンは一部の処理を並行して実行できます。

B.1.3.1 既定のバケット・サイズ

既定では、バケットは 512,000 個のレコードです。バケット・サイズは、`bucketSize` オプションによって制御されます。このオプションでは、バケット・サイズはレコードのグループの数を表す整数値で示されます。グループは、64,000 個の連続するレコードで構成されています。既定の `bucketSize` は 8 です。このため、既定のバケットは、8 x 64,000 レコー

ド、つまり 512,000 レコードです。bucketSize の詳細は、“InterSystems Business Intelligence のモデルの定義” の “<cube>” を参照してください。

B.2 エンジン・ステップ

MDX クエリを処理するために、Analytics エンジンには以下のステップを実行します。

1. 準備 (プロセス内で実行されます。つまりこのステップはバックグラウンド・プロセスとしては起動しません)。このフェーズで実行される手順は以下のとおりです。
 - a. エンジンがクエリを解析して、それをオブジェクト表現 (解析ツリー) に変換します。
この解析ツリーでは、クエリの各軸が独立して表現されます。1 つの軸がクエリの全般的なフィルタを表します。
 - b. エンジンが、解析ツリーを正規化クエリ・テキストに変換します。
この正規化したクエリ・テキストでは、例えば、すべての **%FILTER** 節が、同等な 1 つの **WHERE** 節にまとめられています。
 - c. エンジンは、正規化されたクエリ・テキストに基づいてハッシュを生成し、このハッシュ値をクエリ・キーとして使用します。クエリ・キーにより、エンジンは、このページで説明したグローバルで、このクエリの結果を検索できます。
 - d. ^DeepSee.Cache.Results にある以前に実行したクエリ結果の中に再利用できるものがある場合、エンジンはその結果を再利用し、以降のステップを省略します。
2. 軸の実行 (これもプロセス内で実行されます)。このフェーズで実行される手順は以下のとおりです。
 - a. エンジンは、サブクエリを実行します。
 - b. エンジンはスライサ軸 (**WHERE** 節) を検証して、関連フィルタ処理 (例えば、サブジェクト領域のフィルタ) をマージして、^DeepSee.Cache.Axis をこの軸に関する情報で更新します。
 - c. 残りの各軸も検証して、^DeepSee.Cache.Axis を更新します。
3. セルの実行 (複数の並行処理によって、バックグラウンドで実行します)。このフェーズで、エンジンは以下のようにバケットごとに結果の各セルの中間値を取得します。
 - a. 最初に、指定されたバケットのセルの値が ^DeepSee.Cache.Cell に格納されているかどうかを確認します。
格納されている場合は、その値を使用します。
 - b. 格納されていない場合は、^DeepSee.Index の該当ノードを使用して、必要なビットマップ・インデックスを取得します。これらのビットマップ・インデックスを組み合わせ得られたインデックスを使用して、ソース・テーブルの中で該当するレコードを検索します。

キャッシュがバケットを使用している場合、エンジンは今後のクエリで使用するために ^DeepSee.Cache.Cell にノードを追加します。
4. 統合 (プロセス内で実行されます)。このフェーズで実行される手順は以下のとおりです。
 - a. スライサ軸ごとに、エンジンがその軸の結果セルをそれぞれ検証します。

結果セルごとに、^DeepSee.Cache.Cell の中でそのセルの値が格納されているすべてのノードを検索します。

次に、それらの値を結合します。
 - b. 結果セルごとに、スライサ軸全体にわたって結果を結合して単一の値を取得します。

詳細は、[次のセクション](#)を参照してください。

エンジンは、統合フェーズ中に **CURRENTMEMBER** 関数を評価します。その他の関数については、この処理のより早い段階で評価します。

B.3 軸のたたみ込み

統合フェーズで、複数のスライサ軸が存在すると、Analytics エンジンでは結果セルごとにそれらの軸全体にわたる結果を統合します。このステップを、軸のたたみ込みといいます。

重要 軸のたたみ込みを実行すると、どのスライサ軸にも NULL の結果がないソース・レコードは複数回カウントされます。

軸のたたみ込みが必要かどうかを判断するために、エンジンは、すべてのソース (サブジェクト領域、ピボット・テーブル、およびダッシュボード) から、クエリに適用したすべてのフィルタを検討します。以下のように、これらのフィルタの最終的な組み合わせから、軸のたたみ込みが必要かどうかが決まります。軸のたたみ込みが必要になる状況を以下のテーブルに示します。

フィルタの形式	軸のたたみ込みの実行
単一のメンバ。例: [PRODUCT].[P1].[PRODUCT CATEGORY].&[Candy]	なし
単一のメジャー。例: [MEASURES].[Units Sold]	なし
タプル (複数のメンバの組み合わせ、または複数のメンバと 1 つのメジャーの組み合わせ)。例: ([Outlet].[H1].[City].&[7],[PRODUCT].[P1].[PRODUCT CATEGORY].&[Candy])	なし
%TIMERANGE 機能 CROSSJOIN(%TIMERANGE([BirthD].[H1].[Date]&[1000],[BirthD].[H1].[Date]&[5000]),%TIMERANGE([BirthD].[H1].[Date]&[4000],[BirthD].[H1].[Date]&[NOW])) でラップされたメンバを使用する交差結合	可
その他の交差結合。例: NONEMPTYCROSSJOIN([Outlet].[H1].[City].&[7],[PRODUCT].[P1].[PRODUCT CATEGORY].&[Candy])	なし
%OR 関数。複数のメンバをリストするセット式のラップです。例:%OR({[Product].[P1].[Product Category].&[Candy],[Product].[P1].[Product Category].&[Snack]})	なし
複数のメンバをリストするが、%OR は使用しないセット式。例:{[Product].[P1].[Product Category].&[Candy],[Channel].[H1].[Channel Name].&[2]}	あり

アナライザでこれらの式をフィルタとして作成するには、**[フィルタ]** ボックスに項目をドラッグ・アンド・ドロップする方法が一般的です。最後の 2 つの行にセット式を作成するには、高度なフィルタ・エディタを使用する必要があります。エンジンは、可能な場合、自動的に %OR 関数を使用します。高度なフィルタ・エディタでは、これはオプションとしては表示されません。

B.4 クエリ・プラン

MDX クエリツールでクエリを実行する場合はクエリ・プランを表示できます。同様に、(“Business Intelligence クエリのプログラムによる実行”で説明したように) プログラムによってクエリを実行する場合は、結果セットの %ShowPlan() メソッドを呼び出すことができます。以下はその例です。

```
SAMPLES>do rsl.%ShowPlan()
----- Query Plan -----
**SELECT {[MEASURES].[AVG TEST SCORE],[MEASURES].[%COUNT]} ON 0,[AGED].[AGE
BUCKET].MEMBERS ON 1,[GEND].[GENDER].MEMBERS ON 2 FROM [PATIENTS]****
DIMENSION QUERY (%GetMembers): SELECT %ID,DxAgeBucket MKEY, DxAgeBucket
FROM BI_Model_PatientsCube.DxAgeBucket ORDER BY DxAgeBucket**
**DIMENSION QUERY (%GetMembers): SELECT %ID,DxGender MKEY, DxGender
FROM BI_Model_PatientsCube.DxGender ORDER BY DxGender**
**EXECUTE: 1x1 task(s) **
**CONSOLIDATE**
----- End of Plan -----
```

ここでは、ドキュメントを PDF バージョンに適した形式にするために、改行とスペースが追加されています。

B.5 クエリ統計

(“Business Intelligence クエリのプログラムによる実行”で説明したように) プログラムによってクエリを実行する場合は、結果セットの %PrintStatistics() メソッドを呼び出すことができます。以下はその例です。

```
SAMPLES>do rsl.%PrintStatistics()
Query Statistics:
Results Cache:                0
Query Tasks:                  1
Computations:                 15
Cache Hits:                   0
Cells:                        10
Slices:                       0
Expressions:                  0

Prepare:                      0.874 ms
Execute Axes:                 145.762 ms
Columns:                      0.385 ms
Rows:                         144.768 ms
Members:                     134.157 ms
Execute Cells:                6.600 ms
Consolidate:                  1.625 ms
Total Time:                   154.861 ms

ResultSet Statistics:
Cells:                        0
Parse:                        3.652 ms
Display:                      0.000 ms
Total Time:                   3.652 ms
```

ここに表示された値の説明を以下に示します。

- ・ Query Statistics — この統計のグループでは、結果セットを返したクエリに関する情報が得られます。その結果セットを使用するために実行された内容についての情報は含まれません。
 - Results Cache は、結果キャッシュが使用された場合に 1 になります。それ以外の場合は 0 になります。
 - Query Tasks は、このクエリがいくつかのタスクに分割されたかのカウントです。
 - Computations は、中間計算 (集約オプションに従ってメジャーを集約するなど) を実行するために費やした時間を示します。これには、MDX 式の評価は含まれません。
 - Cache Hits は、中間キャッシュが何回使用されたかのカウントです。

- Cells は、結果セットのすべてのセルと、計算された中間セルのカウントです。
- Slices は、クエリ内のキューブ・スライス数のカウントです。このカウントは、WHERE 節での項目数を示します。
- Expressions は、MDX の評価に費やした時間を示します。

キャッシュが使用された場合は、Computations、Cache Hits、Cells、および Expressions のすべてがゼロになります。

- Prepare、Execute Axes、Execute Cells、および Consolidate は、クエリ処理の各部分に費やされた時間を示します。これらの部分は、順序どおりにリストされます。
- Total Time は、上記の部分の合計です。

キャッシュが使用された場合は、Execute Cells と Consolidate の両方がゼロになります。これは、それらの処理部分が実行されないためです。

- ・ ResultSet Statistics - この統計グループでは、結果セットで返された後で、結果セットを使用するために実行された内容に関する情報が得られます。この値は以下のとおりです。

- Cells は、結果セット内のセル数のカウントです。
- Parse は、結果セットの解析に費やした時間を示します。
- Display は、表示に費やした時間を示します。
- Total Time は、これらの時間の合計です。

C

MDX パフォーマンス・ユーティリティの使用

システムには、ツール `%DeepSee.Diagnostic.MDXUtils` クラスが用意されていて、クエリ統計と下位レベルのパフォーマンス統計を同時に収集できます。Business Intelligence の実装時または実装後にこのツールを使用できます。

このクラスは、`%Run()` メソッドを提供します。

```
classmethod %Run(pMDX As %String = "",  
                 pBaseDir As %String = "",  
                 pVerbose As %Boolean = 0,  
                 ByRef pParms="",  
                 Output pOutFile="") as %Status
```

MDX クエリを指定すると、このメソッドは、クエリを準備して実行し、そのクエリに関する診断情報を含むファイルを生成します。引数は以下のとおりです。

- ・ `pMDX` – MDX クエリを指定します。
- ・ `pBaseDir` – 出力ディレクトリ (`MDXPerf`) の書き込み先のベース・ディレクトリを指定します。既定のベース・ディレクトリはインストール・ディレクトリです。
- ・ `pVerbose` – ルーチンを詳細モードで呼び出すかどうかを指定します。はいの場合には 1、いいえの場合には 0 (既定値) を使用します。
- ・ `pParms` – 多次元配列のパラメータを指定します。この配列には以下のノードを使用できます。
 - `pParms("CubeStats")` – キューブ統計を生成するかどうかを指定します。はいの場合には 1 (既定値)、いいえの場合には 0 を使用します。
 - `pParms("TimePERFMON")` – `^PERFMON` を使用してデータを収集する時間を秒単位で指定します。正の整数を指定します。既定値は 15 です。詳細は、“監視ガイド”の“`^PERFMON`を使用したパフォーマンスの監視”を参照してください。
 - `pParms("pButtonsOn")` – `^SystemPerformance` レポートも生成するかどうかを指定します。はいの場合には 1、いいえの場合には 0 (既定値) を使用します。
 - `pParms("pButtonsProfile")` – 使用する `^SystemPerformance` プロファイルの名前を指定します。詳細は、“監視ガイド”の“`^SystemPerformance`を使用したパフォーマンスの監視”を参照してください。
- ・ `pOutFile` – 出力パラメータとして返されます。この引数は、このメソッドで生成されるメイン・レポートの HTML ファイルの名前を指定します。

`%Run()` メソッドは以下のファイルを生成します。

- ・ `MDXPerf_nnnnnn_nnnnnn.html` – メイン HTML レポート・ファイルです。これには、クエリ統計、クエリ・プランなどが含まれます。

- ・ `cubename.xml` – 指定したキューブの定義です。
 - ・ **Cached_MDXPerf_cubename_nnnnnn_nnnnnn.html** – 結果キャッシュを使用するときにクエリを実行する `PERFMON タイム・コレクション・レポート
- 詳細は、“監視ガイド” の “`PERFMON を使用したパフォーマンスの監視” を参照してください。
- ・ **Uncached_MDXPerf_cubename_nnnnnn_nnnnnn.html** – 結果キャッシュを使用しないときにクエリを実行する `PERFMON タイム・コレクション・レポート
- (既定で) 512,000 レコードより多く使用するキューブの場合のみ、エンジンは結果キャッシュを作成するため、このレポートは **Cached_MDXPerf_cubename_nnnnnn_nnnnnn.html** と同じ数値を使用する可能性があります。
- ・ `hostname_date_time.html` – `SystemPerformance レポート
- 詳細は、“監視ガイド” の “`SystemPerformance を使用したパフォーマンスの監視” を参照してください。
- ・ `SystemPerformance によって生成された他のファイル。これらはオペレーティング・システムによって異なります。

D

InterSystems Business Intelligence の診断

DeepSeeButtons は、Business Intelligence 環境についての診断レポートを生成するために使用するツールです。Business Intelligence の実装時または実装後にこのツールを使用できます。

HTML 形式のレポートによって、システムの以下の側面に関する情報を提供します。

- ・ 設定パラメータ
- ・ サーバの詳細
- ・ キューブとそのプロパティのリスト
- ・ キューブごとに、ディメンジョンとそのプロパティのリスト
- ・ キューブごとに、ピボット変数、名前付きセット、およびリスト・フィールドなどのその他の要素のリスト
- ・ Business Intelligence ログ
- ・ iris.cpf ファイルのコンテンツ
- ・ messages.log ファイルのコンテンツ

このレポートを生成するには、%SYS ネームスペースにいることを確認し、以下のコードを実行して、ターミナルから DeepSeeButtons ツールを起動することができます。

```
Do ^DeepSeeButtons
```

後続のプロンプトに従って、レポートを生成してください。生成された HTML は、Chrome または Firefox で表示することをお勧めします。

E

Business Intelligence のその他のエクスポート/ インポート・オプション

ここでは、Business Intelligence の要素をエクスポートおよびインポートするためのその他のオプションについて説明します。この情報は、[実装](#)プロセスの手順の一つとして、“[クラスへの Business Intelligence 要素のパッケージ化](#)”を補足するものです。

注釈 このページの内容は、ユーザがスタジオに対するエクスポートとインポートのプロセスを理解していることを前提としています。

E.1 Business Intelligence コンテナ・クラスの作成

“[クラスへの Business Intelligence 要素のパッケージ化](#)”で説明したように、ピボット・テーブルなどのフォルダ項目を InterSystems IRIS® のクラスにパッケージ化できます。要素を必要な数だけ 1 つのクラスにパッケージ化できるので、多数の個別のファイルよりもエクスポートおよびインポートしやすくなります。

そのようなクラスを作成する手順は以下のとおりです。

- ・ クラスは `%DeepSee.UserLibrary.Container` を拡張したものである必要があります。
- ・ クラスには、`Contents` という名前の XData ブロックを含める必要があります。その XData ブロックに対して、以下のように XML ネームスペースを指定する必要があります。

```
[ XMLNamespace = "http://www.intersystems.com/deepsee/library" ]
```

- ・ この XData ブロック内の最上位の要素は、`<items>` でなければなりません。

`<items>` 要素内に XML 定義を必要な数だけ含めます。この定義は、スタジオでコピーすることも、エクスポート済みの XML ファイルからコピーすることもできます。[次のセクション](#)も参照してください。必要になる編集についての説明があります。

また、ファイルの先頭にある XML 宣言ではなく、定義部分のみをコピーして貼り付けるようにしてください。つまり、以下の行は XData ブロックにコピーしないでください。

```
<?xml version="1.0" encoding="UTF-8"?>
```

以下はその例です。

```
Class BI.Model.DashboardItems Extends %DeepSee.UserLibrary.Container
{
XData Contents [ XMLNamespace = "http://www.intersystems.com/deepsee/library" ]
{
<items>
<dashboard dashboard definition here ...
</dashboard>
<dashboard another dashboard definition here ...
</dashboard>
<pivot pivot definition here ...
</pivot>
<pivot another pivot definition here ...
</pivot>
<pivot yet another pivot definition here ...
</pivot>
</items>
}
}
```

このクラスのコンパイル時や、このクラスの %Process() インスタンス・メソッドの呼び出し時に、システムは XData ブロックに定義された項目を作成します。具体的には、それらの定義をユーザ・ポータルで使用する内部グローバルにインポートします。

同じクラスでは、%OnLoad() コールバックを定義することもできます。このコールバックは、それらの項目の設定時に必要な追加コードを実行できます。

クラス定義にパッケージ化されるピボット・テーブルとダッシュボードのサンプルは、[サンプル](#)・クラス `BI.DashboardsEtc` および `HoleFoods.DashboardsEtc` を参照してください。

コンテナ・クラスを削除しても、現在存在するピボット・テーブルやダッシュボードには影響しません。

E.2 フォルダ項目のエクスポートとインポート

ここでは、フォルダ項目を[エクスポート](#)および[インポート](#)するための古い API について説明します。

E.2.1 プログラムによるフォルダ項目のエクスポート

フォルダ項目をプログラムでエクスポートするには、以下のコマンドを使用します。

```
Do ##class(%DeepSee.UserLibrary.Utils).%Export(itemname,filename)
```

以下は、この指定の説明です。

- ・ itemname は項目の完全名です (項目が属しているフォルダの名前を含みます)。
 - ピボット・テーブルでは拡張子 `.pivot` を追加します。
 - ダッシュボードでは拡張子 `.dashboard` を追加します。
 - ウィジェットでは拡張子 `.widget` を追加します。
 - テーマでは拡張子 `.theme` を追加します。
- ・ filename は、作成するファイルのフル・パスとファイル名です。ファイルは XML ファイルであるため、ファイル名の末尾に `.xml` を使用することをお勧めします。

以下はその例です。

ObjectScript

```
set DFIname="Chart Demos/Area Chart.pivot"
set filename="c:/test/Chart-Demos-Area-Chart-pivot.xml"
do ##class(%DeepSee.UserLibrary.Utils).%Export(DFIname,filename)

set DFIname="KPIs & Plugins/KPI with Listing.dashboard"
set filename="c:/test/KPIs-Plugins-KPI-with-Listing-dashboard.xml"
do ##class(%DeepSee.UserLibrary.Utils).%Export(DFIname,filename)
```

E.2.1.1 代替手法 (複数の項目のエクスポート)

複数の項目をプログラムによって 1 つの XML ファイルにエクスポートするには、\$system.OBJ.Export() メソッドを使用します。このメソッドの最初の引数と 2 番目の引数を以下に示します。

- items は以下のような多次元配列です。

配列ノード	ノード値
items ("full-folder-item-name.DFI")。ここで、items は配列名で、full-folder-item-name.DFI はフォルダ・アイテムの完全名です。大文字と小文字の区別も含めて、スタジオに表示されるとおりになります。	" "

この引数は多次元配列なので、\$system.OBJ.Export() メソッドを使用する際、その前にピリオドを付ける必要があります。

- filename は、作成するファイルのフル・パスとファイル名です。ファイルは XML ファイルであるため、ファイル名の末尾に .xml を使用することをお勧めします。

以下に例を示します。

ObjectScript

```
set items("Chart Demos-Area Chart.pivot.DFI")=""
set items("Chart Demos-Bar Chart.pivot.DFI")=""
set items("Chart Demos-Bubble Chart.pivot.DFI")=""
set filename="c:/test/Chart-Samples.xml"
do $system.OBJ.Export(.items,filename)
```

このメソッドを使用して、クラスなどの他の項目をエクスポートすることもできます。詳細は、%SYSTEM.OBJ のクラス・リファレンスを参照してください。

E.2.2 プログラムによるフォルダ項目のインポート

フォルダ項目をプログラムでインポートするには、以下のコマンドを使用します。

ObjectScript

```
Do ##class(%DeepSee.UserLibrary.Utils).%Import(pFile, pReplace, pVerbose)
```

以下は、この指定の説明です。

- pFile は、インポートするファイルのフル・パスとファイル名です。
- pReplace が True の場合、同じ名前を持つ既存の項目が置換されます。既定値は False です。
- pVerbose が True の場合、ステータスがコンソールに書き込まれます。既定値は True です。

以下はその例です。

ObjectScript

```
set filename="c:/test/Chart-Demos-Area-Chart-pivot.xml"  
do ##class(%DeepSee.UserLibrary.Utils).%Import(filename,1,1)
```

F

Business Intelligence と災害復旧

ここでは、Business Intelligence を使用して非同期ミラー・メンバのコピー済みソース・データを書き込み保護する際の推奨手順を説明します。

F.1 構成

このセクションでは、必須の初期構成タスクについて説明します。

1. すべてのソース・データ・データベースと新しくマップされた[^]OBJ.DSTIME のデータベースを備えた災害復旧 (DR) 非同期として、非同期ミラーを設定します。これにより、復旧時ではなく構成時にシステムの詳細な検証を実行して、ISCAgent などの問題をプッシュします。このモードでは、読み込み/書き込みデータベースは考慮されない点に注意してください。
2. 構成の完了後に、DR を読み取り専用非同期メンバに切り替えます。
3. 読み取り専用非同期では、特定のデータベースごとに ReadOnly フラグがあります。このフラグは、書き込みを許可するためにクリアできます。この操作は、[^]OBJ.DSTIME を格納しているデータベースに対して実行します。

ソース・データは書き込み保護され、キューブは適切に同期できるようになります。

F.2 災害復旧

このセクションでは、災害復旧時に実行する手順について説明します。

1. ミラー構成から、[^]OBJ.DSTIME を格納しているデータベースを削除します。このデータベースは、引き続き使用できます。
2. 非同期メンバを DR メンバに切り替えます。
3. このメンバをプライマリに昇格させます。
4. キューブを同期します。

[^]OBJ.DSTIME バッファは、このバッファに現在依存している別のシステムでは期限切れとして扱う必要があります。これは、そのデータが別の非同期メンバと同期されなくなるためです。[^]OBJ.DSTIME を格納しているデータベースは、復旧手順の一環として、ミラー・セットに追加し直す必要があります。

