

CA Business Service Insight

実装ガイド

8.2.5



このドキュメント（組み込みヘルプシステムおよび電子的に配布される資料を含む、以下「本ドキュメント」）は、お客様への情報提供のみを目的としたもので、日本 CA 株式会社（以下「CA」）により随時、変更または撤回されることがあります。

CA の事前の書面による承諾を受けずに本ドキュメントの全部または一部を複写、譲渡、開示、変更、複本することはできません。本ドキュメントは、CA が知的財産権を有する機密情報です。ユーザは本ドキュメントを開示したり、

(i) 本ドキュメントが関係する CA ソフトウェアの使用について CA とユーザとの間で別途締結される契約または (ii) CA とユーザとの間で別途締結される機密保持契約により許可された目的以外に、本ドキュメントを使用することはできません。

上記にかかわらず、本ドキュメントで言及されている CA ソフトウェア製品のライセンスを受けたユーザは、社内でユーザおよび従業員が使用する場合に限り、当該ソフトウェアに関連する本ドキュメントのコピーを妥当な部数だけ作成できます。ただし CA のすべての著作権表示およびその説明を当該複製に添付することを条件とします。

本ドキュメントを印刷するまたはコピーを作成する上記の権利は、当該ソフトウェアのライセンスが完全に有効となっている期間内に限定されます。いかなる理由であれ、上記のライセンスが終了した場合には、お客様は本ドキュメントの全部または一部と、それらを複製したコピーのすべてを破棄したことを、CA に文書で証明する責任を負います。

準拠法により認められる限り、CA は本ドキュメントを現状有姿のまま提供し、商品性、特定の使用目的に対する適合性、他者の権利に対して侵害のないことについて、黙示の保証も含めいかなる保証もしません。また、本ドキュメントの使用に起因して、逸失利益、投資損失、業務の中断、営業権の喪失、情報の喪失等、いかなる損害（直接損害か間接損害かを問いません）が発生しても、CA はお客様または第三者に対し責任を負いません。CA がかかる損害の発生の可能性について事前に明示に通告されていた場合も同様とします。

本ドキュメントで参照されているすべてのソフトウェア製品の使用には、該当するライセンス契約が適用され、当該ライセンス契約はこの通知の条件によっていかなる変更も行われません。

本ドキュメントの制作者は CA です。

「制限された権利」のもとでの提供: アメリカ合衆国政府が使用、複製、開示する場合は、FAR Sections 12.212、52.227-14 及び 52.227-19(c)(1)及び(2)、ならびに DFARS Section 252.227-7014(b)(3) または、これらの後継の条項に規定される該当する制限に従うものとします。

Copyright © 2013 CA. All rights reserved. 本書に記載された全ての製品名、サービス名、商号およびロゴは各社のそれぞれの商標またはサービスマークです。

CA への連絡先

テクニカルサポートの詳細については、弊社テクニカルサポートの **Web** サイト (<http://www.ca.com/jp/support/>) をご覧ください。

目次

第 1 章: 概要	9
役割.....	9
サービス カatalog マネージャ	10
契約マネージャ.....	10
ビジネス ロジック エキスパート	11
データ ソース エキスパート	11
システム管理者.....	12
ソリューション プロセス	13
計画	14
設計	15
実装	17
インストールおよび展開.....	17
 第 2 章: 計画立案および設計	 19
要件収集セッション（すべての役割）	20
サンプルデータの収集（データ ソース エキスパート）	22
設計.....	23
設計概要（契約マネージャ、ビジネス ロジック エキスパート、データ ソース エキスパート）	24
契約モデリング（契約マネージャ）	25
契約モデリング段階の出力（契約マネージャ、データ ソース エキスパート）	47
データ モデリング（データ ソース エキスパート、ビジネス ロジック エキスパート）	49
データ モデリング段階の出力（データ ソース エキスパートおよびビジネス ロジック エキスパート）	67
 第 3 章: 実装	 69
実装 - 概要.....	70
フレームワークのセットアップ（契約マネージャ）	73
テンプレート ライブラリのセットアップ（契約マネージャ）	73
契約の作成方法（契約マネージャ）	74
サービスからの契約の作成.....	75
サービス レベル テンプレートの作成.....	77
契約ライフサイクルおよびバージョン管理.....	77
データ収集（データ ソース エキスパート）	79

アダプタ機能.....	80
アダプタ環境.....	81
メイン ファイル.....	84
ワーク ファイル.....	85
アダプタ通信.....	91
アダプタ レジストリ 設定.....	95
アダプタの実行モード.....	97
設定ツールおよびメンテナンス ツール.....	99
アダプタおよび変換の設定方法.....	99
変換スクリプトによる自動変換.....	139
アダプタ機能の詳細トピック.....	140
ビジネス ロジック スクリプティング (ビジネス ロジック エキスパート)	148
ビジネス ロジック スクリプティングのワークフロー	150
ビジネス ロジック モジュール.....	151
ビジネス ロジック の内部.....	152
契約をアクティブ化する方法 (契約マネージャ)	196
契約メトリックの完全な再計算.....	199
成果物の作成 (契約マネージャ)	200
セキュリティ設定の定義 (管理者)	200
レポートの作成方法.....	201
ダッシュボード ページの設定.....	214
サービス レベル アラート プロファイルの追加	217

第 4 章: インストールおよび展開 221

概要.....	222
準備.....	224
インストール.....	226
環境間でのインポート/エクスポート方法 (データ ソース エキスパート)	227
統合 - メール サーバの設定 (システム管理者)	229
システム プリファレンスの設定 (システム管理者)	230
セキュリティ設定 (システム管理者)	230
SSA 同期設定の指定.....	231

付録 A: サービスドメインおよびドメイン カテゴリの例 233

付録 B: ケース スタディ例 235

契約モデリング例.....	235
ケース スタディ 1: システム可用性	236

ケース スタディ 2 : システム可用性 2	237
ケース スタディ 3 : システム応答時間	239
ケース スタディ 4 : ヘルプデスク パフォーマンス	246
ケース スタディ 5 : システム バックアップ	247
会計メトリック モデリング例	248
ケース スタディ 6 : 契約/サービスの会計条件のモデリング	248
データ モデリングの例	256
ケース スタディ 7 : サーバ パフォーマンス	256
ケース スタディ 8 : ヘルプデスクの能力	260
カスタム属性の使用例	268
ケース スタディ 9 : 動的な複数ターゲット	268
変換スクリプトの例	272
ケース スタディ 10 : 基本的な自動変換	272
ケース スタディ 11 : リソース モデルの更新	275
ビジネス ロジック スクリプティングの例	278
ケース スタディ 12 : カウンタ ロジックの障害数の計算への使用	279
ケース スタディ 13 : 動的コンポーネント グループの処理	283
ケース スタディ 14 : 累積時間クロックの処理	288
効果的なビジネス ロジック例の作成	294
ケース スタディ 15 : クラスタ化メトリック	297
ケース スタディ 16 : ビジネス ロジックの設計パターン	301
ケース スタディ 17 : 組み込み機能	306
ケース スタディ 18 : 登録	309
ケース スタディ 19 : ファイル ベースのデータ ソースの場合のアダプタ ウィザード	312
アダプタの作成	314
アダプタ展開	324
アダプタのスケジュール	325
ケース スタディ 21 : LDAP との統合の例	328
ケース スタディ 22 : PERL を使用してレポートを生成する	335

付録 C: アダプタ設定の指定 339

全体の構造	339
General セクション	340
CA Business Service Insight Interface セクション	346
DataSourceInterface セクション	349
SQL インターフェース セクション	352
InputFormatCollection セクション	356
TranslationTableCollection セクション	361
TranslatorCollection セクション	363

付録 D: ビジネス ロジック計算式の定義(ビジネス ロジック エキスパート)	367
ビジネス ロジック計算式の作成時に避けるべき事項	367
クラスタ化メトリックおよびリソースの有効性	367
用語集	371

第 1 章：概要

本書では、CA Business Service Insight の計画と設計、実装、インストールと展開に伴う実装プロセスと関連する役割、責任、インタラクションについて説明します。

このセクションには、以下のトピックが含まれています。

[役割](#) (P. 9)

[ソリューションプロセス](#) (P. 13)

役割

役割とは、典型的な実装中に実行されるタスクの共通のセットを実行するユーザーのことです。また「役割」という用語は、タスクセット自体を指す場合にも使用されます。CA Business Service Insight には多くの事前定義済みの役割があり、システム管理者はそれらを編集できます。また、新しい役割や特定のアクセス権を作成することもできます。

以下の役割が実装に必要です。

- サービス カタログ マネージャ
- 契約マネージャ
- ビジネス ロジック エキスパート
- データ ソース エキスパート
- システム管理者

各役割の責任および資格について、以下に説明します。

注: システム管理者の定義するとおり、同一人物が複数の役割を果たせます。

サービス カタログ マネージャ

責任：

- 組織のサービス カタログの管理
- 組織の提供サービス カタログの定義
- 契約の定義およびレポートの要件の管理

必要資格

- 組織の E2E サービス デリバリ プロセスを熟知していること
- CA Business Service Insight の概念を熟知していること

契約マネージャ

責任：

- エンド ユーザとのインタラクション
- 定義された要件に基づくサービス カタログの作成
- 新規契約の作成と既存契約の保守
- （任意、推奨）特定のカスタマ SLA の実装

資格：

- メトリックの原理とロジックを理解していること
- CA Business Service Insight の GUI ベースの機能のエキスパート ユーザであること

ビジネス ロジック エキスパート

責任：

- メトリックの実装
- ビジネス ロジック スクリプトの作成と既存ビジネス ロジック スクリプトの保守

資格：

- 基本的な開発経験、および VB-Script などのスクリプト言語の豊富な実務知識
- CA Business Service Insight データ フローについての十分な理解
- CA Business Service Insight ビジネス ロジック スクリプティングの専門家
- CA Business Service Insight アーキテクチャおよびコンポーネントについての十分な理解
- CA Business Service Insight の GUI ベースの機能のエキスパート ユーザであること

データ ソース エキスパート

責任：

- CA Business Service Insight のインフラストラクチャ設計
- 新しいアダプタの作成、既存アダプタの保守、オペレーショナル インフラストラクチャとインターフェースをとって測定値を取得
- CA Business Service Insight インフラストラクチャの構築および保守

資格：

- カスタマ データ ソース環境および構造を理解していること
- データベース、XML およびスクリプト言語についての実務知識
- CA Business Service Insight データ フローおよびデータ収集プロセスを理解していること
- CA Business Service Insight アダプタの専門知識
- CA Business Service Insight アーキテクチャおよびコンポーネントを理解していること
- CA Business Service Insight の GUI ベースの機能のエキスパート ユーザであること
- 開発経験があるのは有利

システム管理者

責任：

- インストールとアップグレードの管理
- ハードウェアおよびソフトウェア システムの要件を満たすこと
- セキュリティ設定の管理： ユーザおよび役割の定義
- システムのモニタおよび管理

資格：

- カスタマイズインフラストラクチャおよびネットワークを理解していること
- CA Business Service Insight アーキテクチャおよびコンポーネントを理解していること
- DB、XML、およびスクリプト言語についての実務知識
- CA Business Service Insight データ フローを理解していること
- CA Business Service Insight の GUI ベースの機能のユーザであること

ソリューション プロセス

ソリューション プロセスには、以下の 3 つの基本的な段階が含まれます。

- 計画立案および設計
- 実装
- インストールおよび展開

前述の役割は、それらの専門に従って実装プロセスの異なる段階にそれぞれ参加します。これらの役割は相互に連携して、最初から最後まで（つまり、契約上の定義から出力レポートまで）の全工程を完了します。

このガイドは、実装プロセスを構築する一般的な段階フローに基づいて指向された役割およびプロセスとして構成されています。このガイドでは、各役割の作用および役割間の連携について説明します。

以下のパラグラフでは、実装プロセスおよび各役割のアクティビティに含まれる段階の概要を示します。

このガイドの残りの部分は段階別に構成されており、段階に参加する役割を示します。各章では、各アクティビティについて、それを担当する役割を示します。

各段階の基本的なタスクおよび各役割の機能について、簡単な説明を以下に示します。詳細については、各章で確認することができます。

計画

計画立案段階の目的は、実装の一部として対処する必要のある要素をすべて特定および数値化することです。

この段階では、成功した実装を踏まえて **CA Business Service Insight** からの期待度を理解するために、契約マネージャがサービス カタログ マネージャからビジネス要件を収集します。この段階では、現時点での要件と将来計画の両方を理解し、容易かつ円滑に成長、発展できる実装であることを確認することが重要になります。

定義された実装要件に従って、契約マネージャは既存のプロセスの入出力を確認することにより、既存の関連プロセス（存在する場合）を検証する必要があります。必要な情報ソースをすべて特定すること、および既存の出力の計算に使用されるサンプル、契約、出力レポート、入力ソースを取得することが、この段階では必要になります。この情報を注意深く指定して、契約マネージャが必要な入力をすべて特定する必要があります。そうすることにより、必要な出力が引き出されます。

この段階では、契約マネージャは契約を検証して、実装の一部として提供されるサービス カタログとテンプレートが現在および将来のニーズに対応すること、また現在の実装が契約領域をすべて網羅していることを確認します。

契約マネージャはレポートおよびその形式を検証し、定義されたすべての測定が行われ、既存の実装で作成できることを確認します。

その後、データ ソース エキスパートは、契約マネージャによって提供される必要な測定値および計算について入力データ サンプルを検証します。これにより、データ ソース エキスパートは対処する必要のある入力の問題（カスタム形式や不具合など）を特定し、追加のデータ ソースが必要かどうかを判断することができます。

この検証の目的は、必要なメトリックを計算するために必要な情報および取得プロセスに関する初期情報（データ ソースとの通信方法やデータ ソース構造など）がソースに含まれるようにすることです。

設計

設計段階では、収集された入出力要件は、契約、メトリック、およびリソースで構成される **CA Business Service Insight** モデル構造に変換されます。これは、**CA Business Service Insight** フレームワークに適合するように実環境データを変換し、それを概念化することを意味します。

システム設計には以下のものが含まれます。

契約モデリング

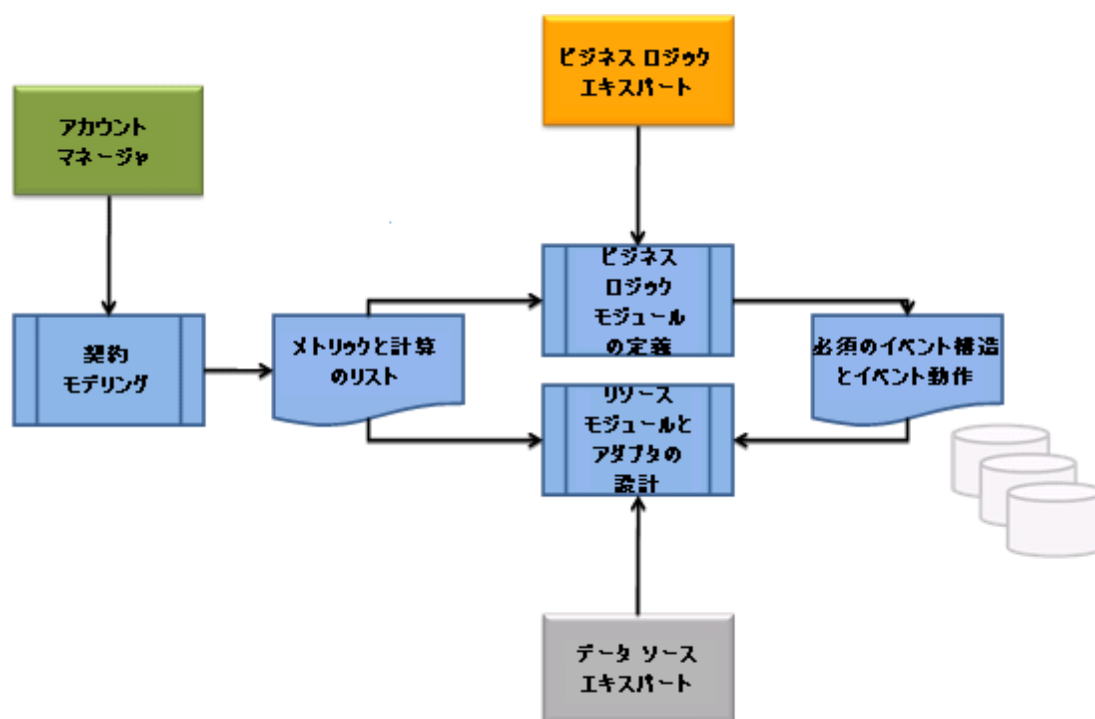
顧客の SLA が **CA Business Service Insight** 契約として解釈され、サービスカタログ エンティティ (テンプレート フォルダなど) が定義されます。これは、主として契約マネージャによって実行されます。

データ モデリング

リソース データが検証され、**CA Business Service Insight** リソース モデルにモデル化されます。これは、データ ソース エキスパートおよびビジネス ロジック エキスパートによって実行されます。

契約またはデータ モデリングのいずれかで利用可能な方法がいくつかある場合があります。けれども、たいていは最適なプラクティスがあり、問題を解決するだけでなく、柔軟性または生産性を向上させます。これにより、さらに成長するための強力なフレームワークが提供されます。

最適な方法を選択する際にデータ ソース エキスパートを支援するためには、ケース スタディがソリューションの提案と共に使用されます。



上記のワークフロー図に示すように、契約モデリングプロセスの結果、契約マネージャは、設定する必要があるメトリックのリストおよびそれらの計算概要定義を出力として提供します。

例:

顧客 A、CNP システムの平均応答時間

メトリック リストは、必要な計算スクリプトの定義を可能にする必要なイベント構造とイベント動作を定義するビジネスロジックエキスパートの入力として提供されます。

メトリック リストおよびイベント構造と動作要件は、データソースエキスパートによるリソースモデルとデータアダプタの設計の入力になります。

実装

実装段階では、CA Business Service Insight システムは設計段階の結果に基づいて設定されます。実装段階では、理論的な設計段階の結果を取得し、それらの結果を使用して CA Business Service Insight の運用要件を構築する必要があります。

作成または対処する必要がある項目には以下のものが含まれます。

契約マネージャ

- サービス セットアップ
- 契約作成
- レポートおよびダッシュボード セットアップ

データソース エキスパート

- アダプタ設定
- インフラストラクチャ セットアップ

ビジネス ロジック エキスパート

- ビジネス ロジック モジュール

インストールおよび展開

インストールおよび展開の段階は、実稼働システムのインストールおよび統合に関係しており、そのパフォーマンスのテストと監視、およびユーザのトレーニングを行います。システム管理者およびデータ ソース エキスパートは、この段階でほとんどのアクティビティを実行します。

第 2 章：計画立案および設計

このセクションには、以下のトピックが含まれています。

[要件収集セッション（すべての役割）](#) (P. 20)

[サンプルデータの収集（データ ソース エキスパート）](#) (P. 22)

[設計](#) (P. 23)

要件収集セッション(すべての役割)

要件収集セッションは CA Business Service Insight の実装で最初に重要な手順で、実装に関与する主要人員はすべて参加する必要があります。

このセッションに必要な情報を提供する者は、SLA ビジネス ロジック、現在の SLA 管理法、SLA 用に作成されるレポートの種類、将来予想されるフル レポート要件の内容など、選ばれた SLA に精通している必要があります。

以下の者が参加する必要があります。

- サービス カタログ マネージャ
- 契約マネージャ
- データ ソース エキスパート
- ビジネス ロジック/スクリプトのエキスパート
- 選択された SLA に精通している他の関連スタッフ
- プロジェクト マネージャ (指定された場合)

CA の推奨事項は以下のとおりです。

- 選択された SLA の確認中に生じる可能性のある不明瞭な定義について、決定が下せる人物がいることを保証するのは重要です。
- この会議に先立って、実装チームがこのプロジェクト用に選択された SLA のコピーを受け取ることをお勧めします。出席者は、選択された SLA に関連するデータ ソースすべてに精通し、それらのデータ ダンプを提供することができ、必要に応じてその内部構造を説明できる必要があります。

計画セッションの主目標は、次の項目を定義し指定することです。

- 計画された作業の範囲
- 成功基準
- 関連するすべての当事者の役割および責任
- 実装プロセスおよびスケジュール
- 必要なハードウェア/ソフトウェアの要件

これは次のものによって実行されます。

- 実装で使用する SLA の確認
- 選択された SLA 内の各目的の「ビジネス ロジック」の確認
- 選択された SLA に関連するレポート、アラートおよびダッシュボード要件の確認

- 実装で使用する関係データ ソースの確認
- 関連するインフラストラクチャ トポロジの確認
- 関連するデータ ソースから関連データ ダンプを収集
- 鍵となる人物とその責任の確認
- 実装の継続期間、レビュー会議、Q&A プロセスなどについてのコミュニケーション パスの定義
- 実装のプロセスおよびスケジュールの確認
- 業務の現状、すなわち、これらの SLA の管理方法や不足物の確認
- 実装の成功条件の定義
- 必要なハードウェア/ソフトウェア要件の定義
- 実装スケジュールの定義

サンプルデータの収集(データソース エキスパート)

オフサイトでアダプタの初期設定を実行するには、データ収集の目的で過去のデータのサンプルを取得することが重要です。これらのサンプルは、CA Business Service Insight が必要とするデータの取得元となる実際のシステムから、後で受信される見込みのデータと同じ構造である必要があります。サンプルに加えて、データソース エキスパートは以下の問題を明らかにすることができるように、データソースの所有者からデータソースに関する知識を得る必要があります。

タイプ:

データベース、ファイル、ストリームなど。

場所:

どこにあるか? そこにはどのように行くか(接続詳細)? セキュリティ障害は? プラットフォームは?

命名規則:

ファイル名はどのように付けられるか?(ファイルベースのデータソースか?) テーブル名はどのように付けられるか? テーブルのフィールド名は?

可用性:

いつアクセスできるか? いつアクセスする必要があるか(ロードの考慮事項)? 連続的な更新か、またはX時間ごとの更新か? 持続期間はどれくらいか?

動作:

データは累積しているか? データは上書きされるか? アーカイブは保持されるか?

スパン:

利用可能な履歴データの量はどれくらいか?

ボリューム:

現在のデータのボリュームは? 成長率は? 予測は?

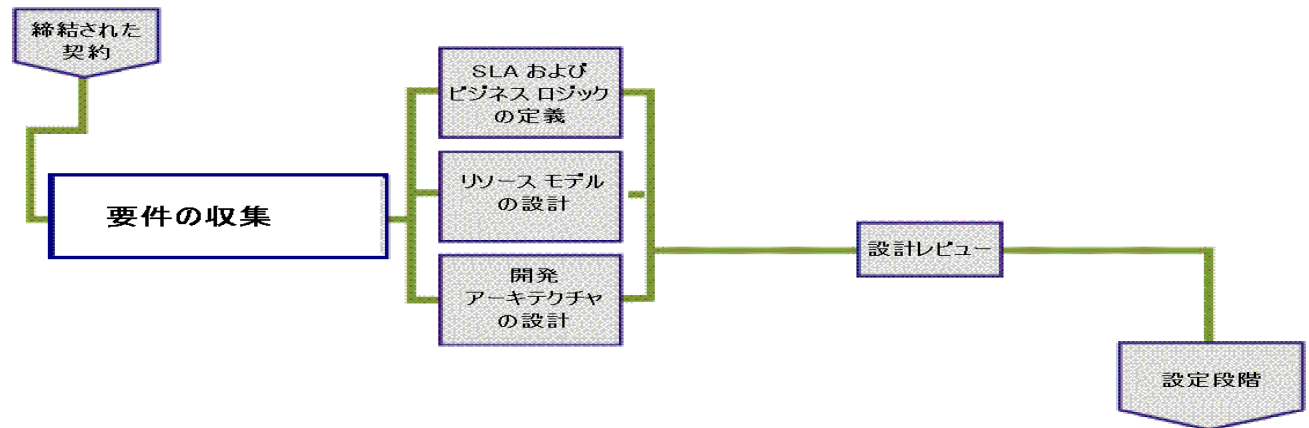
構造および形式:

データはデータソース内でどのように構成されるか? データフィールドは何か? テーブル名は何か? データフィールドを区切るものは何か?

ストリーミング:

アダプタはデータを取得するか、またはデータはアダプタによって収集されるか?

設計



設計概要(契約マネージャ、ビジネス ロジック エキスパート、データ ソース エキスパート)

このセクションでは、ソリューション プロセスの設計段階の背後にあるプロセスおよび推論について説明します。計画立案段階の後に設計段階が続き、その後に以下の章の実装段階が続きます。

設計段階の目的は、実装チームが実際の契約、それらの条項、および既存のパフォーマンス データと **CA Business Service Insight** システムとのマッピングを完了できるようにすることです。

このプロセスを開始する前に、実装チームは、すべての要件を考慮するだけでなく、同時に将来の成長や変化を考慮しながら、結果としての設計を可能な限り最適化するために、利用可能なさまざまなメソッドやオプションについて理解する必要があります。

設計プロセスには、以下の手順を実行する実装チームが関与します。

- 契約を検証し、必要な **CA Business Service Insight** オブジェクトに変換します。これらのオブジェクトは、このガイドで「契約モデリング」（契約マネージャの責任）と呼ばれています。
- 既存のデータ セットを取得し、望ましい結果を考慮して抽出すべきもの、および関連項目を抽出する方法を決定します。望ましい結果は、このガイドで「データ モデリング」（データ ソース エキスパートおよびビジネス ロジック エキスパートの責任）と呼ばれています。

契約モデリング セクションでは、以下について説明します。

- 使用される用語（正しい実装に必須です）
- ビジネス ロジック テンプレートおよびモジュール
- サービス レベル テンプレートおよびその使用方法
- 強力なサービス カタログを作成する重要性
- 顧客契約に適用される会計管理メトリック（ペナルティ、インセンティブ、コスト、例）およびその他の契約上のメトリック

データ モデリング セクションでは、以下について説明します。

- **CA Business Service Insight** に適用されるイベントおよび **CA Business Service Insight** システムを介したそれらのイベントのフロー
- メトリックおよびその登録方法
- **CA Business Service Insight** リソースおよびその特定方法
- これらのリソースの収集および定義を自動化する方法の提案

上記のすべてのポイントについて、以下の各セクションでさらに詳しく説明します。

この段階での不適切な選択は CA Business Service Insight の運用に悪影響を及ぼす可能性があり、後の段階での変更が困難または不可能になる場合があることに留意することが重要です。

契約モデリング(契約マネージャ)

契約モデリング用の以下タスクは契約マネージャによって実行されます。

契約用語

契約は目標の集合です。目標は、契約上/運用上の目標（メトリック）または会計上の目標（インセンティブ）のいずれかです。契約モデリングのプロセスでは、実装スコープに含まれる契約内容全体を把握し、それを CA Business Service Insight モデルに変換する必要があります。

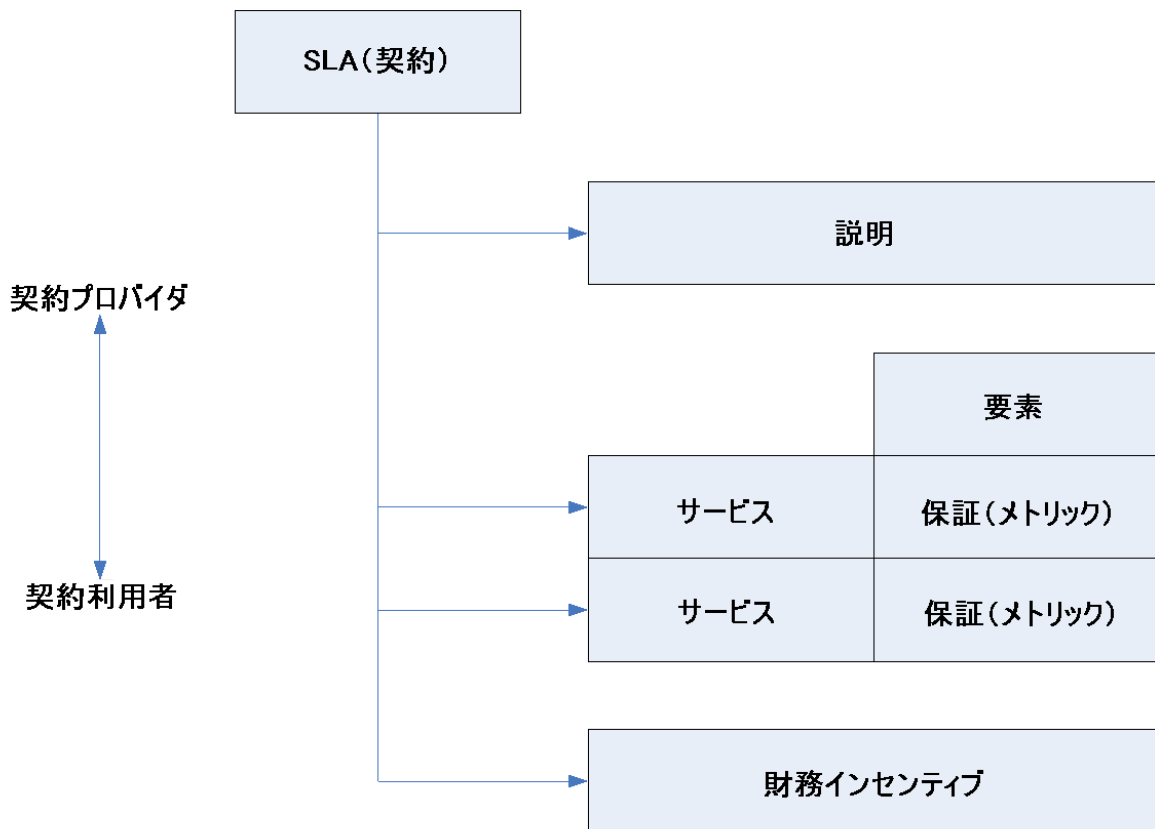
以下の図はこのプロセスを示します。

契約プロバイダ

SLA (契約)

- 提供/利用するサービスの説明。
- 提供/利用するサービスのターゲット サービス レベル。
- サービスの測定方法の詳細。
- サービスの計算方法の詳細。
- サービス レベルのレポート方法の詳細。

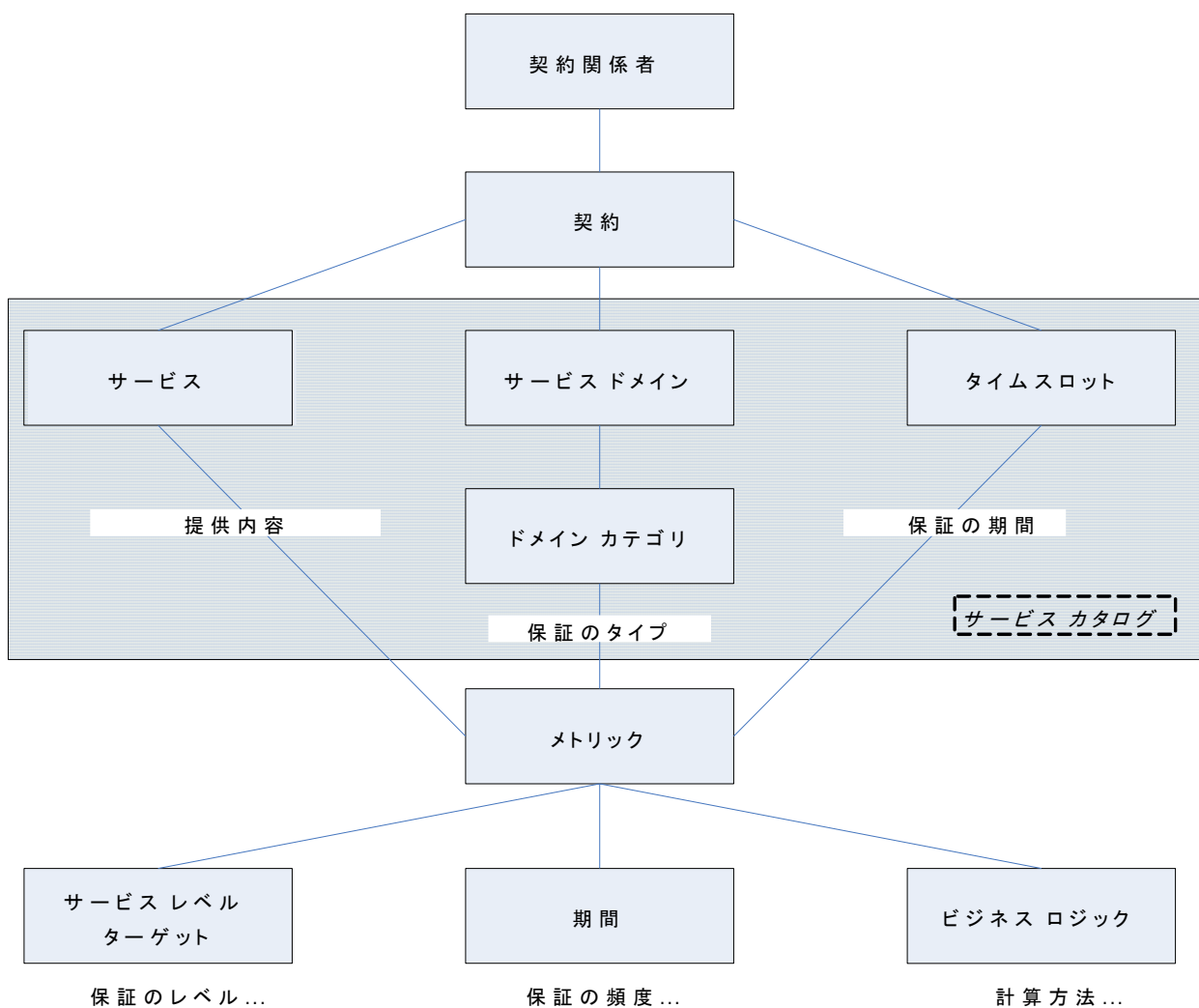
契約利用者



注: 実装スコープに含まれる要素のみをモデル化するとしても、将来考えられるシステムの計画を考慮する必要があります。スコープには、将来の開発を包含する広範で十分なモデルを設計するために、組織の一般的な契約管理要件を含める必要があります。これにより、将来必要となる変更が最小化および簡略化されます。

顧客契約を CA Business Service Insight モデルに変換することは、契約マネージャが、共通の範囲で提供される保証（メトリック）を論理グループ、共通サービス コンポーネント、契約要素などにグループ化するプロセスです。この論理的なグループ化により、非常に柔軟なサービス レベル管理が可能になります。

CA Business Service Insight 契約モデルは、以下の図に示すエンティティで構成されます。



〔契約〕

メトリックの条件および集合を定義します。契約は、契約関係者間の関係に応じて、3つの異なるタイプのいずれかになります。契約は、SLA（組織とその顧客の間のサービス レベル アグリーメント）、OLA（組織内の部門間のオペレーショナル レベル アグリーメント）、または UC（組織とそのサプライヤの間の請負契約。一般に UC は、SLA を介して組織が別の顧客に供給しているサービスに関するものです）のいずれかになります。

契約関係者

提供されるサービスの顧客、および契約に調印する相手の顧客を定義します。単一の契約関係者を複数の契約に添付することができます。契約内では、その契約に関連するサービスのプロバイダおよび利用者を定義できることに注意してください。

メトリック

単一のサービス レベル目標または保証を定義します。各メトリックは、レポートが実行される実際のサービス レベルの結果を生成する測定値のリクエストです。

メトリックには以下の項目が含まれます。

タイプ

メトリックは以下の 8 つのタイプのいずれかになります。

SLO	サービス レベル目標。
情報	ターゲットのないサービス レベル目標。
KPI	さまざまな業界の概念をサポートするために使用されるキー パフォーマンス インジケータ。電気通信会社では KPI=SLO は顧客の義務を意味し、IT では技術的な義務を意味します。
KQI	キー クオリティ インジケータ。さまざまなインジケータから集約された結果を測定するための全体的なメトリックです。
中間	他のメトリックで使用する中間測定値。これらのメトリック結果はレポートできません。
消費	サービス/リソース消費を計算するために会計メトリックによって使用されます。価格アイテム メトリックと共に使用されます。
インセンティブ	インセンティブ（ペナルティに対応する肯定的な用語）の計算に使用される会計メトリック。

価格アイテム 消費に基づいて結果を計算するために使用される会計メトリック。期間ごとまたは単位数ごとに異なる価格。

トラッキング期間

実際に計算されたサービス レベルの結果がターゲットと比較される契約上のレポート期間または期間。CA Business Service Insight のトラッキング期間は、時間、日、週、月、四半期、または年の粒度で定義できます。根本原因解析のために、トラッキング期間に加えて、メトリックも他の 6 つの粒度で計算されます。しかし、これらの期間の結果はオペレーション結果のみとしてマークされます。

注: これが実行されるのは、これらの計算粒度をメトリックに対して指定した場合のみです。

タイムスロット

特定の保証または義務が適用されるトラッキング期間内の時間。例外的なタイムスロット定義（予測されるメンテナンス期間、標準的な銀行休業日など）が含まれます。

ビジネス ロジック

サービスの要素に対して実際のサービス レベル値を計算するために Raw データの評価方法を定義する計算式、および計算で考慮する必要のある特定の前提条件。設計段階では、計算の概要のみが特定されます。設定段階では、より詳細に定義されます。

ターゲット

契約上のサービス レベル義務。ターゲットは、該当のメトリックのタイプに応じて、必須である場合と必須でない場合があります。ターゲットが定義されない場合、メトリックはレポートのみの目的で結果を提供し、契約上の義務または保証と比較することはできません。ターゲットは静的または動的になります。

注: ここまでに示したいくつかの概念について理解するには、[付録 B 「Case Studies」](#) (P. 235)を参照してください。

各メトリックは、システム テンプレート フォルダに含まれる以下のシステム全体のエンティティに関連付けられています。

サービス コンポーネント

提供する義務があるものを説明します。

サービス コンポーネント グループ

サービス コンポーネントの集合。単一のサービス コンポーネントを複数のグループに含めることができます。サービス コンポーネントグループはオプションのエンティティです。これを使用すると、レポート機能の柔軟性が非常に高くなります。

サービスドメイン

サービス レベルのモニタリングの目的で測定する必要のあるサービス コンポーネントの特定要素（たとえば、パフォーマンスや可用性）。

ドメイン カテゴリ

特定の測定単位。これはデフォルトの測定単位を定義し、またターゲット目標が最小値か最大値かを定義します。本質的に、これは測定されるサービス ドメインの特定ディメンション（つまり、利用可能時間の %、停止回数、平均スループット率など）です。

上記のエンティティ、および CA Business Service Insight で監視されるすべてのサービス レベル目標とこれらのエンティティとの関係は、契約モデリング段階で特定する必要があります。

モデリング プロセスの機能

モデリング プロセスでは、検討中の契約およびレポート要件に基づいて、システムで設定する必要のあるすべてのメトリックをそれらの関連エンティティと共に定義する必要があります。

この段階以前またはこの段階中に、システムが明確かつ整然と見えるように、またシステムのナビゲートが容易になるように、メトリック名に使用する命名基準を決定することをお勧めします。さらに、メトリックの「目標ステートメント」セクション内で使用できるメトリックの説明についても検討してください。

契約分析プロセスには以下の手順が含まれます。

1. 契約上の目的を特定します。

各目的について、以下のものを特定します。

- 適切な名前（命名基準に留意してください）
- ターゲット（オプション）
- トラッキング期間
- 測定単位
- サービス コンポーネント
- サービス ドメイン
- ドメイン カテゴリ（および測定単位）
- タイムスロット（期間：24 x 7 か?、業務時間のみか?）
- 計算概要（必要な変数およびサービス レベルの計算方法）

注: これらの一部は、最初のチェックからは明確にならない場合がありますが、後でサービス カタログを絞り込むことで明確にすることができます。

2. 契約から、会計上の目標をすべて特定し、会計上の目標ごとに以下のことを判断します。
 - その目標はペナルティ、インセンティブ、またはコストの目標か?
 - その目標をトリガする条件は何か?
 - その目標は、どのサービス コンポーネントに対して発生するか?
 - サービス ドメイン
 - ドメイン カテゴリ（ここでの単位は、通貨やコストの金額、支払の割合などの場合があります）

すべての目標が特定および定義された後は、メトリックの完全なリストを確認し、カタログの最適化/正規化を試みることをお勧めします。

この手順では、サービス コンポーネント、サービス ドメイン、およびドメイン カテゴリがそれぞれ適切に定義され、それらが、システム全体にわたって提供されるものに関する明確で簡潔なカタログを形成できるようにすることが重要です。このカタログには、システムのメトリックおよび契約のすべてが含まれます。これにより、システム内に非常に強力なサービス カatalogが構築され、システムの将来の成長が可能になります。

サービス ドメインおよびドメイン カテゴリは、極端に低い粒度までは定義しないでください。それらは、わかりやすくする必要はありますが、メトリックよりも高い粒度にしてください。

たとえば、以下の3つのメトリックがあるとします。

- 期限内に配布された停止レポートの割合
- 期限内に配布された例外レポートの割合
- 期限内に提供された管理レポートの割合

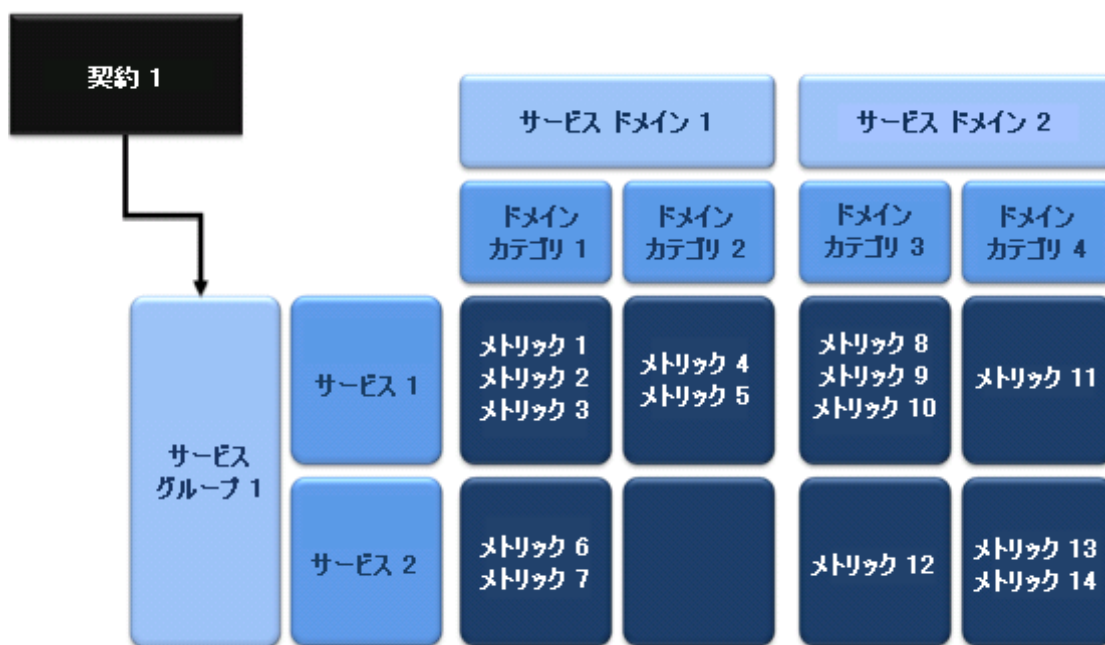
3つのメトリックすべてが分類されると考えられる最適なカテゴリは、パフォーマンスのサービス ドメインおよび「期限内に配布されたレポートの割合」のドメイン カテゴリです。ドメイン カテゴリにはレポートのタイプを含めないでください。これら3つのメトリックは同じ計算方法を持ち、同じビジネス ロジックを使用すると考えられます（ただし、異なるレポート タイプを識別するためのパラメータは除きます。このパラメータは1つであると考えられます）。

適切なサービス ドメインおよびドメイン カテゴリは、ITIL (Information Technology Infrastructure Library) 標準から取得することができます。ただし、これらは提示例にすぎません。個別の組織ごとに独自の標準が定義されている場合があります。提示されるサービス ドメインおよびドメイン カテゴリの一部については、[付録A「サンプルのサービス ドメインとドメイン カテゴリ」](#) (P. 233)を参照してください。

注: 主要なすべての利害関係者との会議を開いて、それらの利害関係者の現在および将来のニーズに対応できるように、選択されたカタログについて互いに賛同を得ることが、この時点では役立つ場合があります。

重要なポイントを実証する詳細な例が付録に記載されています。これらの例では、単一の契約目標をそのモデリングと共に説明しています。現状をモデル化するには、すべての目標を認識して、カタログエンティティが広範かつ包括的な方法ですべての目標を表すようにする必要があります。

すべてのメトリックおよびそれらの関連エンティティを特定するプロセスが完了すると、契約マネージャはすべてのメトリックを示す以下の図のようなマトリックスを作成できます。





モデリングプロセスで考慮する必要のあるその他の問題について、以下の各セクションで説明します。

契約マネージャに対する質問

すべての要素が確実に考慮されるように、契約マネージャが検討する必要がある質問を以下に示します。

正しいサービス コンポーネントを選択したことをどのように確認するか？

サービス コンポーネントを複数の契約に適用でき、複数の要素で測定できる場合、そのサービス コンポーネントは正しく定義されています。

たとえば、システム X は複数の顧客に提供されるシステムであり、その可用性、信頼性、パフォーマンスなどによって測定できます。

正しいサービス ドメインを選択したことをどのように確認するか？

複数の方法でサービス ドメインを測定および計算できる場合、そのサービス ドメインは、提供されたサービスの一般的な要素を示しているので、正しいサービス ドメインです。

たとえば、可用性はいくつかの方法で測定できます。そのうちの 1 つは利用可能時間の割合です。その他の方法としては、業務時間内または業務時間外の可用性の割合、失敗の数、MTBF（Mean Time Between Failures）、MTTR（Mean Time To Repair）、最大ダウンタイム、平均ダウンタイム、合計ダウンタイムなどがあります。これらはすべて特定のシステムの可用性を評価する方法です。

モデリングプロセスで考慮する必要があるケース

モデリングプロセスで考慮する必要がある概念を説明するために、やや一般的なケースともう少し具体的なケースについて、いくつかの例を以下に示します。これらの概念は、メトリックのより正確な定義および安定したフレームワークで発生する場合があります。

ターゲットのないメトリック

メトリック定義内のターゲット フィールドが必須ではないので、そのフィールドが定義されない場合は、サービス レベル レポートをメトリックで利用できます。ただし、ターゲットと偏差に対するサービス レベル レポートは利用できません（実際に計算されたサービス レベルと比較するターゲットがないため）。これらのタイプのメトリックは、実際の契約上の義務に含まれていない情報についてレポートを必要とする場合に定義されます。

このタイプのメトリックを定義すると、レポートで利用可能なすべてのドリルダウン機能がユーザに提供され、また今後任意の段階で測定値をターゲットに適用するオプションがサービス レベル マネージャに提供されます。

例:

契約上の保証は、**99%** のネットワーク可用性および毎月のダウンタイム数に関するレポートを提供することです。

2つのメトリックが定義されます。一方のメトリックは、可用性 **99%** のターゲットで定義されます。もう一方のメトリックは、毎月のダウンタイム数をカウントするためにターゲットなしで定義されます。これらのメトリックは両方ともレポート可能ですが、最初のメトリックにのみ、その契約上の義務により偏差計算があります。

注: この種の状況に対処するには、ビジネス ロジック出力およびこのデータに関する自由形式レポートを使用するという方法も考えられます。ただし、この方法では、データに関するレポートのドリルダウン機能、および簡易レポートウィザードを使用するオプションが失われます。これに対し、ビジネス ロジック出力を使用する利点は、メトリックの数を減らすことによりエンジンの能力を節約できることです。

この方法の詳細については、「[出力 - ユーザ テーブル](#) (P. 173)」のケース スタディを参照してください。

ターゲットがあるメトリック

メトリックに対してターゲットが定義される場合は、ターゲットを指定する 2 つの方法が考えられます。静的ターゲットまたは動的ターゲットとして指定できます。静的ターゲットは最もよく見られるシナリオです。静的ターゲットでは、ターゲットを契約期間にわたり有効な合意値にすることができます。

例:

ネットワーク可用性は、毎月 98% 以上にする必要があります。

この場合のターゲットは 98% です。

あるいは、ターゲットは前月のパフォーマンスに依存するか、年間を通じてその値を単純に変更することもできます。ここで発生する可能性のある別の状況は多数ありますが、一般にそれらはすべて計算式によって実装されます。CA Business Service Insight は、標準的なビジネス ロジック テンプレートからの追加の関数呼び出しによってこの機能をサポートします。目的関数はビジネス ロジックのコンテキストから他のパラメータにアクセスでき、考えられる必要なシナリオをすべてサポートできます。

たとえば、ヘルプデスク ロードに依存するヘルプデスク内のチケットの解決時間について考えます。同じ月内に 1000 チケットしかない場合、優先度の高いチケットの平均解決時間は 1 日です。月内にヘルプデスクで発行されるチケットが 1000 を超える場合、優先度の高いチケットの平均解決時間は 2 日です。

この場合、その月に発行されるチケット数に応じて、ビジネス ロジック スクリプト内で評価される動的ターゲットを持つものとしてメトリックが定義されます。

注: 動的ターゲットの実装方法の詳細については、「動的ターゲットの実装」のケース スタディを参照してください。

メトリック パラメータ

メトリック パラメータは、メトリックのビジネス ロジック内から扱える値であり、また実際のコードを変更することなく、メトリック定義から簡単に変更できる値です。この値はハードコードされた値の代わりに使用され、簡単に変更できます。

メトリック パラメータを特定してビジネス ロジック モジュールを簡単に特定できるようにすること、および再利用可能なコンテンツを作成することが重要です。さらに、エンドユーザが簡単に変更を実行できるようにする契約ウィザードを使用して、メトリック パラメータにアクセスできます。

例:

- 重大度 1 のインシデントは 24 時間以内に解決する必要があります。

上記の義務では、ターゲットは 24 時間の解決率であり、重大度レベル（重大度 1）はパラメータとして定義できます。

- 月内のダウンタイム数は3回を超えないようにしてください。ダウンタイムとは、5分超にわたりシステムが利用できなかった時間です。

上記の義務では、ダウンタイムとして定義する必要があると考えられる時間はパラメータとして定義できます。

契約パラメータ

契約パラメータは契約内のすべてのメトリックで扱える値です。契約パラメータはメトリックパラメータと同じ方法でメトリック内で使用できますが、動的パラメータとして定義されます。

複数のメトリックで同じ値を使用する必要がある場合は、契約パラメータの使用をお勧めします。契約パラメータを使用する別の重要なインセンティブは契約管理を容易にすることです。パラメータは変更されることが多く、システム内で更新する必要があるので、契約内の各メトリックにアクセスしてメトリックレベルでパラメータ値を変更するよりも、契約内の1つの場所にアクセスしてすべてのパラメータ値を同時に変更する方が簡単です。

したがって、最も推奨されるモデリングは、契約レベルのパラメータを契約パラメータとして定義し、メトリックレベルの動的パラメータを使用して、それらの値にアクセスすることです。

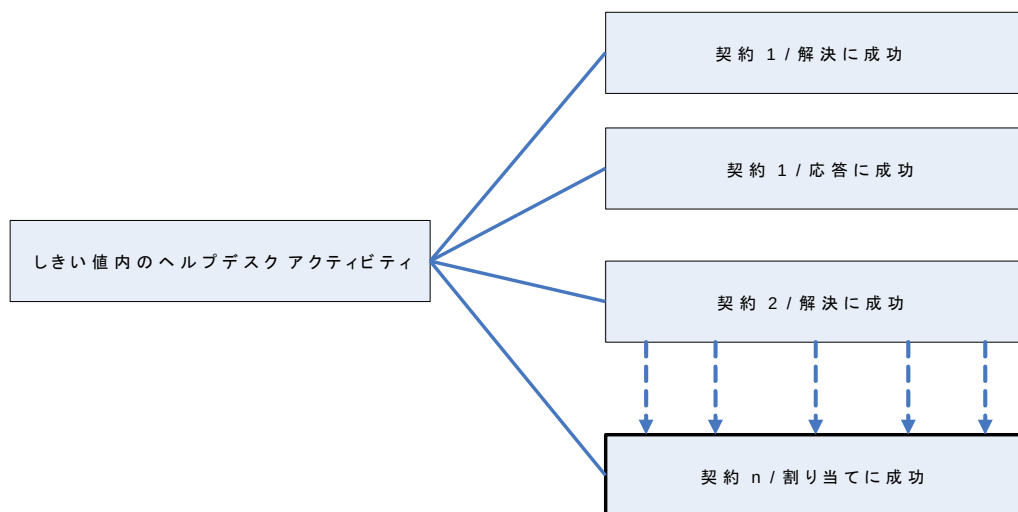
例については、「[ヘルプデスクのパフォーマンス \(P. 246\)](#)」のケーススタディを参照してください。

ビジネス ロジック テンプレートおよびモジュール

ビジネス ロジック テンプレートはメトリックの計算方法を格納する簡単な方法です。フル ビジネス ロジック テンプレートは十分なビジネス ロジック コンポーネントであり、他のビジネス ロジック コンポーネントのベースラインを作成する便利な方法です。テンプレートから作成された新規ビジネス ロジック コンポーネントはコードをコピーし、その新規インスタンスを作成します。ただし、一般的に、テンプレートを使用した場合は柔軟性が非常に低いため、可能な限りビジネス ロジック モジュールを使用する必要があります。

ビジネス ロジック モジュールは、他のビジネス ロジック による同じコード ベースの再利用が可能な独立したコード コンポーネントです。モジュールには他のモジュールを含めることもできるため、階層レベルが複数になることがあります。モジュールを使用する場合、コードは1つの場所に含まれており、それにリンクしている他の各コンポーネントによって再利用されます。このコードセクションの再利用により、コードの重複を排除し、システム全体のロジック変更を迅速かつ簡単に適用できるようにすることで、メンテナンスが容易になります。

設計段階では、主要なビジネス ロジック モジュールおよびそれらの関連パラメータを特定する必要があります。契約モデリングが完了し、使用するロジックが契約マネージャに明確に見えるようになると、契約マネージャが共有し、個別のモジュールに定義できる計算を特定することが可能になります。



上記の図は、指定されたしきい値内の目標を実現するために、ヘルプデスク アクティビティの成功の割合を計算するモジュールを示します。このようなモジュールを実装するには、メトリック パラメータと呼ばれる 2 つのパラメータを定義する必要があります。つまり、ヘルプデスク アクティビティのタイプを定義するパラメータ、および比較に使用するしきい値のパラメータです（「[モデリングプロセスで考慮する必要があるケース \(P. 36\)](#)」のメトリック パラメータの定義を参照してください）。

システムに実装される計算のタイプを慎重に検討すると、コードの 1 つの小さなセクションの変更、および計算タイプ間の「スイッチ」として動作するパラメータの使用により、多くの類似タイプが実行される場合があることがわかります。このように、作成する必要があるコードの量を最小化し、コード再利用の量を最大化できます。

サービス レベル テンプレート

サービス レベル テンプレートは、サービス コンポーネントおよびそれらを測定するために定義された関連メトリックのパッケージされた集合です。これらのサービス レベル テンプレートは、必要に応じて作成することができ、通常は最も一般的に測定されるサービス要素に従って定義されます。

サービス レベル テンプレートを定義する際の重要な問題は、メトリックの動作を変更するために使用できると考えられるすべてのパラメータを特定および公開することです。これにより、システムに最大限の柔軟性が提供され、システムのユーザに関する設定が容易になります。サービス レベル テンプレートを使用して新規メトリックを作成する場合、各設定パラメータはウィザードインターフェースに公開されています。つまり、契約をアクティブにするために、追加のカスタマイズはほとんど必要ありません。ウィザードでユーザに公開されるパラメータは、目標ステートメント内にあります。そのため、メトリックの目標ステートメントでは、エンドユーザが変更できるようにするパラメータをすべて検討します。

サービス レベル テンプレートで最大効率を実現するには、前述のモジュール機能によってすべてのビジネス ロジックを実行することを目指す必要があります。

サービス レベル テンプレートを使用するシナリオ例を以下に示します。このシナリオ例では、顧客の支払金額に応じて、アプリケーション ホスティング サービス コンポーネントが3つのレベル（ブロンズ、シルバー、ゴールド）で顧客に提供されます。追加メトリックは、より厳密なターゲット共に、より高い各ホスティング レベルで提供できます。以下のサンプルシナリオに示すように、これらの各レベルはサービス レベル テンプレートの作成に適した候補になる場合があります。

- アプリケーション ホスティング - ブロンズ（5 メトリック）
- アプリケーション ホスティング - シルバー（8 メトリック）
- アプリケーション ホスティング - ゴールド（12 のメトリック）

これで、新しい顧客をシステムに追加する際に、顧客の要求に応じて必要な定義を簡単に選択できるようになります。ウィザード インターフェースを使用すると、ウィザード インターフェース内の各メトリックを特定の顧客向けにカスタマイズできます。また、特別なカスタマイズが必要な場合は、定義から一部のメトリックのみを選択したり、2つの異なる定義から新しい顧客の契約にメトリックを追加したりすることもできます。

強力なサービス カatalogの重要性

前述のように、サービス カatalogはシステムの重要なコンポーネントであり、構造化された方法で設定する必要があります。これにより、すべてのユーザがシステムを有効に活用できるようになり、混乱が回避されます。また、システムが組織と共に成長できるようになり、既存の機能に対する影響を最小限に抑えながら将来の機能拡張や機能追加にも対処できます。

このようなシステムでは、サービス コンポーネントおよび SLA のカatalogを作成、管理する際の柔軟性が高くなります。ただし、優れたシステムになるかどうかは設計次第なので、設計の初期段階で将来の計画に時間をかけることが非常に重要です。

構造化された効率的な方法で CA Business Service Insight サービス カatalogを定義すると、システムではサービスおよびドメインをサービス カatalogに後で柔軟に追加できるようになります。また、将来的に、契約、メトリック、アラート、およびレポートを追加することもできます。さらに、ビジネス ロジックに対する構造化された手法が強化され、ビジネス データおよび関連する計算を処理する際に、標準化されたモジュールとテンプレートを使用できるようになります。

システムに定義されたサービス レベル テンプレートはカatalogと連携して、必要なデータに関する基本的なレベルの知識がほとんどなくても、新規契約を非常に簡単に作成できる優れた方法を契約マネージャに提供します。この方法を効果的にセットアップすると、単にサービス レベル テンプレートのパラメータを変更するだけで顧客の新規契約を設定することができます。ただし、これは、カatalogおよび定義が最も効果的にセットアップされているかどうか完全に依存します。これらのすべてのパラメータは、サービス レベル テンプレートのメトリックごとに目標ステートメントを介して公開され、目標ステートメントで変更することも、定義を使用する際にウィザードから変更することもできます。

例:

サービス レベル テンプレートを使用して一部のカスタマ サポート メトリックを契約に展開する場合に、既存の定義から必要なメトリックを選択することができます。

サービス

サービス

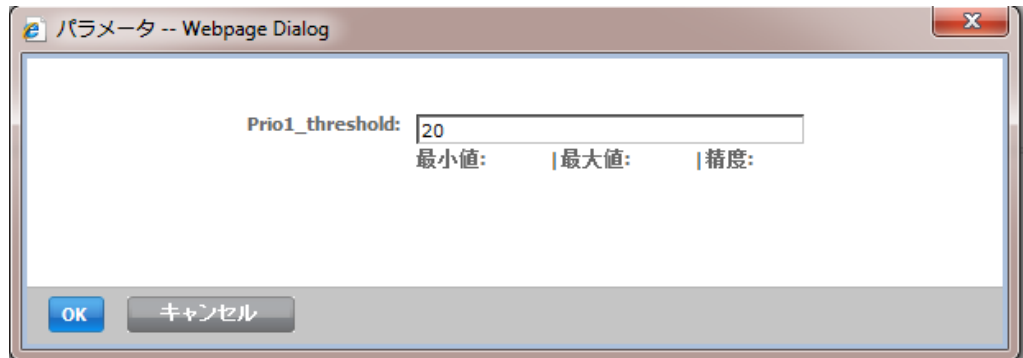
2件中 1 - 2 件を表示します。

名前
Gmail App
SHI

このサービス レベル テンプレートの内部には、[期限内に配布されたインシデントの割合]と呼ばれるメトリックがあります。[期限内に]のコンポーネントは必ずしも明確ではないので、このメトリックには主観性のレベルが存在することがわかります。以下の例では、このメトリックで行われる測定について説明します。

全般	詳細	メモ	クラスifiz	しきい値	関連リンク	精度	目標ステートメント
名前: *	% Inc responded on-time						
説明:	This percentage of incidents responded on time. The response time is represent the time that is needed before the organisation start the resolution process of the incident.						
メトリック タイプ:	Service Level Objective						
目標ステートメント:	次の値以上 98 % of all Incidents will be responded on time during Always per 1 月, with "on-time" being defined as in less then 20 Minutes for Priority1; and less then 2 Hours for all other priorities.						

メトリックの〔全般〕タブの下部（または〔目標ステートメント〕タブ）に表示される目標ステートメントは、このメトリックで公開されるすべてのパラメータを示します。上記の図では、〔期限内に〕の定義は20分と指定されています。これは、この事前定義済みメトリックに関して独自に再解釈できるようにするためのカスタマイズ可能なパラメータです。この値を変更するには、〔20分〕パラメータリンクをクリックします。



このように、サービス レベル テンプレートから作成される新規メトリックは、メトリックの基本的なビジネス ロジックを変更する必要なしにカスタマイズできます。これは、サービス レベル計算にこれらのパラメータを組み込むような方法でビジネス ロジックが書き込まれることを前提としていることに注意してください。

この単純な例は、システム カタログによってサービス レベル テンプレートから将来の契約を活用できるようにするために、強力な柔軟なサービス レベル テンプレートのセットを作成することの重要性を明確に示しています。

会計管理(ペナルティ、インセンティブ、およびコスト)

CA Business Service Insight の以前のバージョンでは、Excel のような数式を使用して実装されたペナルティとして知られている契約エンティティがありました。ペナルティの結果は単に契約のメトリックからの入力に基づいていました。また、ペナルティは結果としてのペナルティ額の計算を基本機能に依存していました。バージョン 4.0 以降では、これらはユーザが作成できる会計メトリックに置き換えられ、柔軟性が非常に高くなっています。これらの会計メトリックは、契約に関するインセンティブ情報またはコスト情報を提供するために使用できます。

注: インセンティブは、CA Business Service Insight 3.0 以前の古いペナルティの用語に代わるものであり、パフォーマンスに応じてポジティブまたはネガティブになります。ただし、負のインセンティブはペナルティと基本的にまったく同じです。また、ペナルティをインセンティブ メトリック タイプと共に実装している場合は、Result() 関数から必ず負の値が返されるようにする必要がありますことに注意してください。これにより、異なるメトリック結果を組み合わせる可能性のある要約関数では、正しい方向に合計を調整できるようになります。つまり、インセンティブが値を増やすと、ペナルティは値を減らします。

また、CA Business Service Insight バージョン 4.0 では、サービス コンポーネントおよびリソースの使用状況を測定する消費メトリックを作成し、そのサービスまたはリソースのコストを決定する価格アイテム メトリックと消費メトリックを組み合わせる機能を提供します。強化された予測機能とこれらを組み合わせることにより、いくつかの非常に包括的な会計管理メトリックを作成することが可能になります。

会計メトリックでは、他の契約上のメトリックからの出力を取得し、それらの契約上のメトリックのパフォーマンスに基づいて、関連するペナルティまたはインセンティブの値を決定することもできます。また、会計メトリックは他のタイプの情報（たとえば、単価情報、レポート機能（予測コストと実コストなど）を可能にする予測モデル）と連携して、それらの結果を決定することもできます。

コスト項目の例:

特定のリスク アプリケーションには、システムの同時ユーザ数に基づく関連コストがあります。これは月単位で計算され、このアプリケーションに提供される予測値があります。このアプリケーションの単価（同時ユーザ当たりのコスト）を以下の表に示します（このアプリケーションはインデックス 1 に分類されるとします）。

Index	Oct-01-2006	Nov-01-2006	Dec-01-2006	Jan-01-2007	Feb-01-2007	Mar-01-2007	Apr-01-2007	May-01-2007	Jun-01-2007
1	1.0	1.0	1.0	0.75	0.75	0.75	0.6	0.6	0.6
2	0.9	0.9	0.9	0.7	0.7	0.7	0.5	0.5	0.5

この期間の予測される同時ユーザ数もあります（同様にインデックス 1）。

Index	Oct-01-2006	Nov-01-2006	Dec-01-2006	Jan-01-2007	Feb-01-2007	Mar-01-2007	Apr-01-2007	May-01-2007	Jun-01-2007
1	3000	5000	8000	5000	5000	5000	5000	5000	5000
2	2500	2500	6000	4000	4000	4000	4000	4000	4000
3	1000	2000	3000	1000	1000	1000	1000	1000	1000

これらのコスト テーブルを使用して、このコスト メトリックをモデル化することにより、実際の同時ユーザ数に基づいて、月単位のアプリケーション コストを決定することができます。この情報はデータ ソースから取得され、上記の単価額を乗算してコスト額を求めることができます。また、この情報を予測値と比較して、期待原価と実際原価の分析を行うこともできます。この場合、ビジネス ロジックでは、期間中に発生した実際の同時ユーザ数を特定し、それを単位当たりコストの値で乗算する必要があります。さらに、ビジネス ロジック内の予測機能は予測情報を参照します。これはコスト項目メトリックを適用する例です。

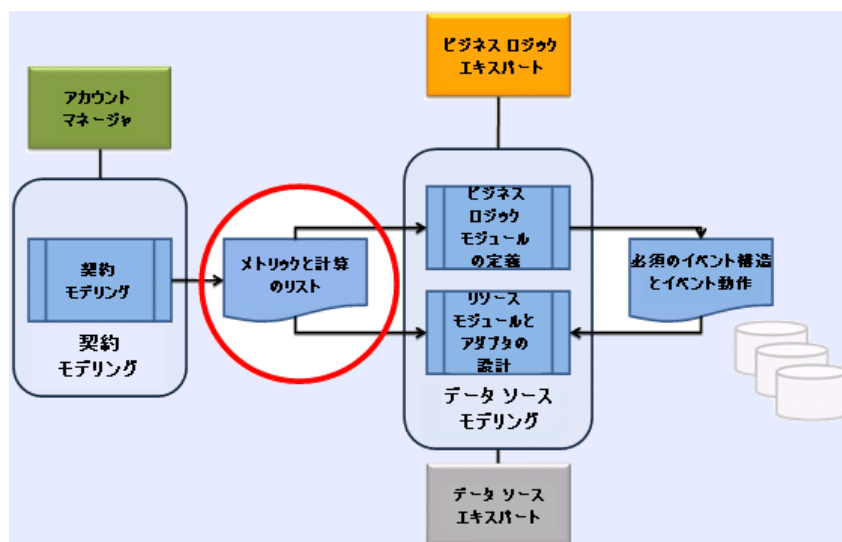
ペナルティ シナリオ例:

所定月の業務時間における 98% のネットワーク可用性を保証するために、顧客の SLA には不履行条項が含まれています。月単位のサービス レベルがこれを下回る場合、計算式（ターゲット（Penalty = ターゲットを下回る全割合につき \$1000（つまり、 $96.5\% = (98 - \text{Round}(96.5)) * 1000 = (98 - 97) * 1000 = \$1000$ ））に基づいてペナルティの支払が課されます。

このペナルティ条件を実装するために、既存のメトリック（ネットワーク可用性 $\geq 98\%$ ）から入力を取得する財務インセンティブ メトリックを作成できます。このメトリックの登録プロセスは RegisterByMetric() プロセスを使用して、比較を行うためにそのメトリックからサービス レベル値を受信します。これは、そのメトリックのトラッキング期間に対するサービス レベル値をこの会計メトリックにイベントとして送信します。その後、これらのイベントはシナリオからの計算式を使用して、その同じ期間のペナルティ額を決定する計算の一部として使用されます。

注: 詳細なケース スタディについては、「[Financial Metric Modeling Case Study \(P. 248\)](#)」を参照してください。

契約モデリング段階の出力(契約マネージャ、データソース エキスパート)



上記の図に示すように、ソリューションのデータ モデリング段階に進むために、契約モデリング段階の最後に、契約マネージャはメトリックのリストおよびそれらの必要な計算を提供する必要があります。リストには以下の情報が含まれています。

- 完了したサービス カタログ定義。これは以下のもので構成されます。
 - 提供されるサービス コンポーネントの完全なリスト
 - サービス ドメインおよびドメイン カテゴリの内訳
 - 各メトリックに必要なすべてのタイムスロットの定義
 - 各タイプの計算を処理するビジネス ロジック モジュールのリスト（それらを実装するために必要なパラメータを含みます）
- 実装されるメトリックの完了したリスト。これには以下のものが含まれます。
 - メトリックに関する明確な命名基準
 - 事前定義済みサービス カタログ コンポーネントに基づく各メトリックの分類

このドキュメントは、契約に関するすべての重要な定義が行われたこと、およびそのすべてのメトリックが完全に定義されたことを確認するための有用なツールです。

このスプレッドシート内の情報は、データ ソース エキスパートが次の段階（データ モデリング）に進むために必要な入力です。

これらの項目が完了した後、次の段階に進むことができます。次の段階では、ソースからの実際のデータでモデル化を開始することができます。完了した契約モデルがない場合、契約上の義務を果たすためにデータ ソースから何が必要かを正確に認識することは非常に困難です。

データ モデリング (データ ソース エキスパート、ビジネス ロジック エキスパート)

データ モデリング セクションでは、設計プロセスの 2 番目の部分について説明します。データ モデリング段階は、カスタマ ソースからデータを取得し、必要なデータの特定のコンポーネントを確認し、このデータの正規化方法を決定し、関連するデータを対応するメトリックに割り当てる方法を評価する過程です。

この段階には以下のタスクが含まれます。

■ ビジネス ロジック エキスパート

- 後にイベント タイプとして定義される、計算に必要な入力イベント構造の識別および定義
- 必要なフィールドをイベント タイプに組み込み、使用されるすべての計算に必要なデータを提供
- イベント フローを最大化するためのメトリック登録プロセスの定義
- 利用可能なデータからリソース モデルを作成し、すべてのレポート要件とロジック要件を満たす方法の決定

■ データ ソース エキスパート

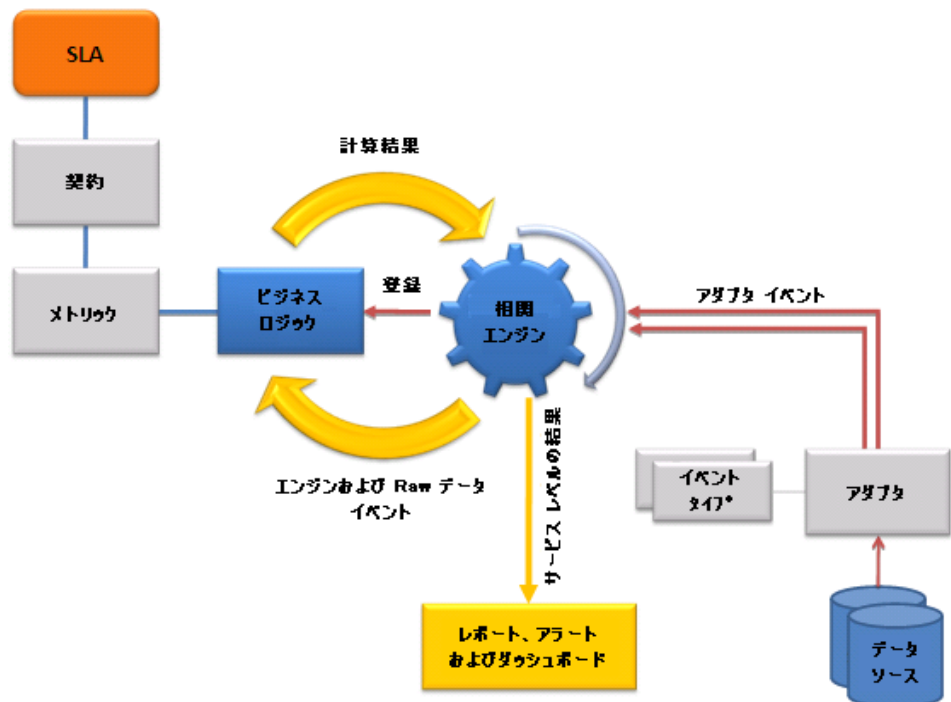
- データ ソース タイプの確認およびアダプタによる定義済みイベント タイプへの正規化および CA Business Service Insight データベースへの供給方法の決定
- データに添付される必要のあるイベント タイプ識別子の決定

イベントおよびイベントフロー

システム内のデータフローはイベントの形式になっています。イベントはソースデータに基づいてアダプタによって作成された情報メッセージであり、CA Business Service Insight によってそのサービスレベル計算に使用できる形式になっています。Raw データは常にイベントで構成されます。

したがって、設計ではシステム内のこのイベントフローに注力する必要があります。

データ要件をモデル化する前に、ビジネス ロジック エキスパートおよびデータ ソース エキスパートは、CA Business Service Insight システム内のイベントとイベントフローについて確実に理解しておく必要があります。以下の図は、この基本的なイベントフローの概略を示しています。



上記の図は、アダプタがデータ ソースからイベントを取得する方法、およびイベント タイプとして定義された標準的なイベント構造にイベントを正規化する方法を示します。これらのイベントはアダプタによって **CA Business Service Insight** に送信されます。これらのイベントは **Raw** データ イベントとして参照されます。

各メトリックのビジネス ロジック計算は **Raw** データ イベントのサブセットに基づいています。したがって、ビジネス ロジックは登録を実行して、このサブセットをリクエストします。

登録ステートメントに基づいて、関連エンジンはビジネス ロジック計算に関連する **Raw** データ イベントのみを送信します。

ビジネス ロジックに送信される他のタイプのイベントはエンジン イベントです。このプロセスに関するすべての概念については、この章で詳しく説明します。

このセクションでは、図の以下の部分を中心に説明します。

- データ ソース
- アダプタ
- イベント タイプ
- メトリック登録

CA Business Service Insight データ モデルは、システム内のデータのこのストリームの効率を最大化するように設計されています。

多くの場合、**CA Business Service Insight** は 2 つのレイヤー（インフラストラクチャ レイヤーおよびビジネス モデル レイヤー）で機能します。単純化された内訳として、インフラストラクチャ レイヤーにはアダプタ、リソース、およびイベント タイプ オブジェクトが含まれ、ビジネス レイヤーには契約、メトリック、およびサービス オブジェクトが含まれています。2 つのレイヤーの間には、関連レイヤーと呼ばれる仮想シム レイヤーがあります。

1 つのイベント識別子はイベント タイプ オブジェクトです。イベント タイプによって、イベントの定義方法および **CA Business Service Insight** へのイベントのレポート方法が決まります。また、処理中にビジネス ロジックによって解釈できるように、イベント データ フィールドの構造も定義されます。

もう1つのイベント識別子は計算で使用する最小エンティティであるリソースです。たとえば、サーバ可用性を計算するとき、レポートする必要がある最小エンティティの論理的な定義は特定のサーバになる場合があります、または、その顧客のチケット処理についてレポートする際には顧客になる場合があります。リソースは、データソースおよび計算要件の両方から派生する **CA Business Service Insight** エンティティの定義です。各リソースには、リソース識別子であるリソースタイプが与えられます。リソースタイプによって、どのような「タイプ」のリソースが定義されるかが正確に決まります。各リソースにはリソースタイプを関連付ける必要があります。これにより、追加のカスタム属性も各リソースに関連付けることができます。これらの属性の詳細については、「[リソースおよびリソース管理 \(P. 60\)](#)」を参照してください。

相関は受信アダプタ イベントと契約メトリックの間に発生します。この相関プロセスの中心はリソース配置およびメトリック登録です。

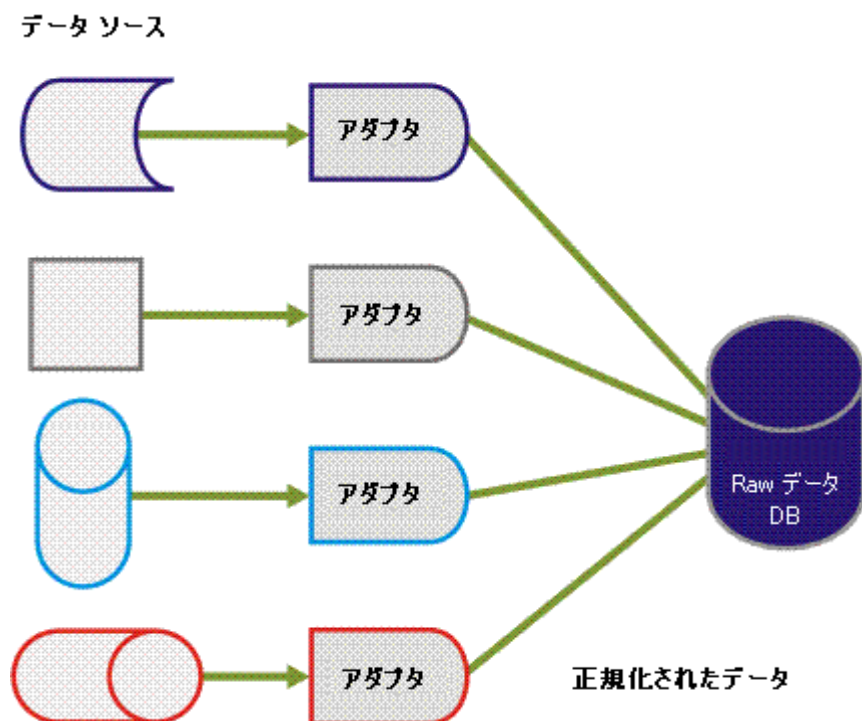
リソース配置およびメトリック登録では、どのリソース イベント ストリームがどのメトリックによって測定されるかを指定します。

メトリック登録では、あるメトリックの出力を別のメトリックの入力として使用できるので、他のメトリックとの間である程度の再利用および相互依存が考えられることに注意してください。同様に、メトリックの出力としてではなく、後で他のメトリックによって使用できる中間の計算手順として、サービス レベルの測定に使用される中間イベントがあります。

データモデル - 概要

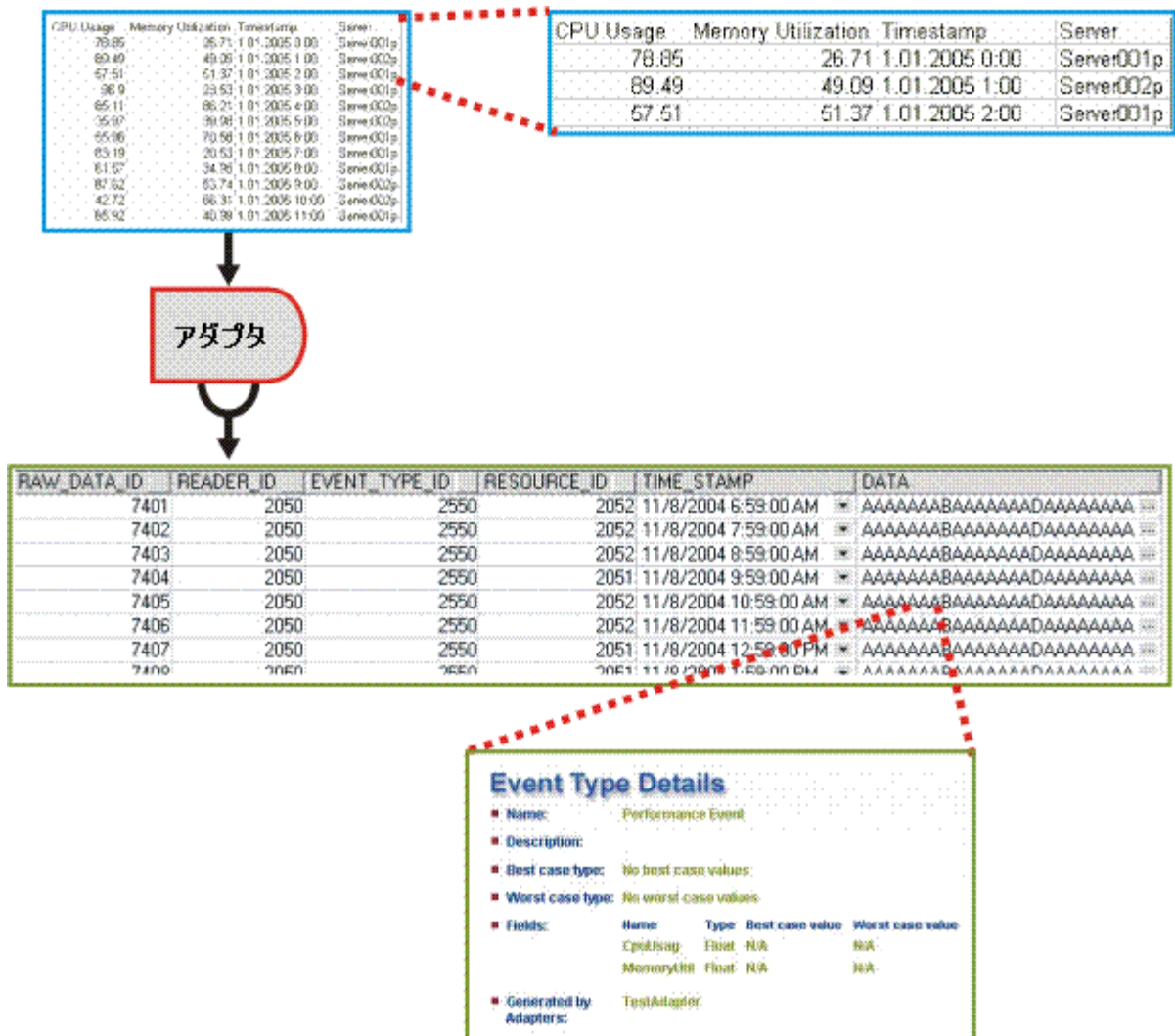
CA Business Service Insight データモデルは、以下の課題に対処してこれを克服するように設計されています。

Raw データは、まったく異なる各種データソースからアダプタによって取得され、さまざまな形式で保持されます。これらの多様なデータを取得し、単一のデータベーステーブルに均質化する必要があります。したがって、以下の図に示すように、アダプタは統合データモデルにデータを読み込み、正規化する必要があります。



このプロセスの一環として、すべてのデータ フィールドが同じデータベース テーブル フィールドに挿入されますが、それらは暗号化されます。CA Business Service Insight データベースに挿入された各行にはイベント タイプ識別子が添付されます。イベント タイプ定義にはデータ フィールドの説明が含まれます。イベント タイプ定義により、相関エンジンはデータ フィールドを正しく解釈し、それらのデータ フィールドがビジネス ロジックでの計算に必要とされる時期を特定することができます。

以下の図は、このプロセスのデータ検索およびデータベース入力のセクションをわかりやすく示しています。また、拡大されたセクションでは、Raw データがどのように表示されるかではなく、実際の期間でデータが何を表しているかを示しています。



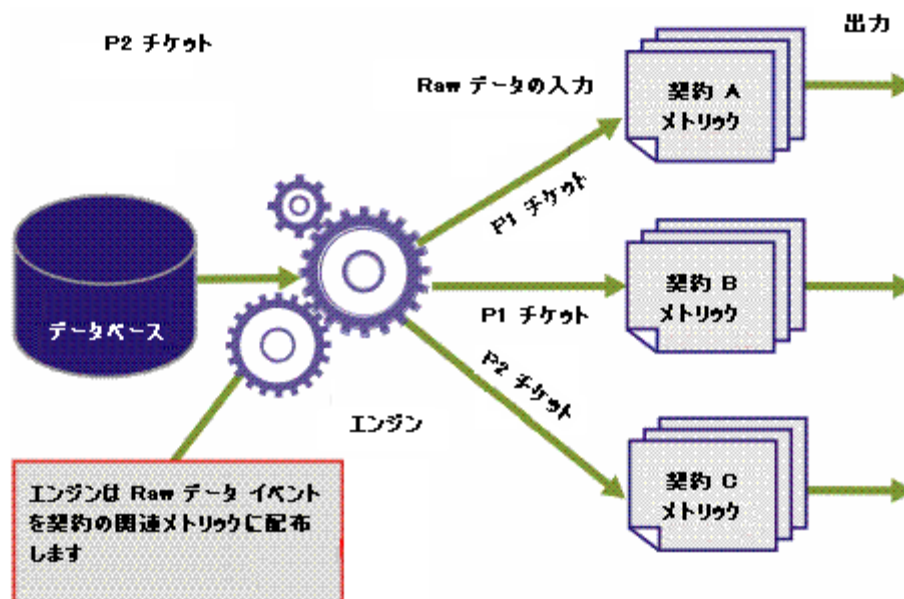
CA Business Service Insight システムには、結果としてのサービス レベルパフォーマンス情報を生成するために、Raw データに対する評価を必要とする契約およびメトリックもすべて含まれています。各メトリックは、その計算に関連するデータのサブセットのみを受信する必要があります。Raw データはさまざまなタイプのレコードを大量に保持している可能性があります。メトリックを使用して関連イベントをそれらの値でフィルタすることは、非常に効率の悪い方法です。そのため、CA Business Service Insight エンジンに関連する Raw データを特定の各メトリックに配布します。

例:

契約に以下の 2 つのメトリックがあるとしてします。

- 優先度 1 (P1) チケット平均解決時間
- 優先度 2 (P2) チケット平均解決時間

最初のメトリックは優先度 1 のチケットのみを評価し、2 番目のメトリックは優先度 2 のチケットのみを評価する必要があります。このため、エンジンはそれに応じてレコードを配布する必要があります。また、最初の契約では、契約関係者 A に公開される P1 チケットに対して解決時間が計算されます。同時に、2 番目の契約では契約関係者 B の P1 チケットに対して計算され、3 番目の契約では契約関係者 C の P2 チケットに対して計算されます。したがって、以下の図に示すように、エンジンはチケット タイプおよびそれを表示された顧客を選択する必要があります。



前述のように、**Raw** データ レコードは、各メトリックのビジネス ロジックに関連するレコードおよびイベントをエンジンが特定できるようにする識別子を添付しています。2つの識別子はイベント タイプとリソースです。

アダプタの変換および正規化

アダプタの機能は、データ ソースからデータを読み取り、それをイベントの形式に正規化することです。CA Business Service Insight 内の各イベントには以下のフィールドが含まれます。

- リソース ID
- イベント タイプ ID
- タイムスタンプ
- イベント タイプに応じた値フィールド

アダプタはデータ ソースの元のフィールドを対応する CA Business Service Insight リソース フィールドにリンクする必要があります。このため、アダプタは、データ ソースおよび対応する CA Business Service Insight リソース ID で見つかった値が含まれる変換テーブルを使用します。

リソース ID およびイベント タイプ ID を関連するデータ ソース値に添付するプロセスは、**アダプタ変換**と呼ばれます。このプロセス中に、変換テーブルは一致する値で作成されます。このテーブルは、プロセスが作成しているイベントレコード内の関連するイベント タイプ ID およびリソース ID に値を入力するためにアダプタによって使用されます。リソースおよびイベント タイプを変換するために、独立したテーブルが作成されます。

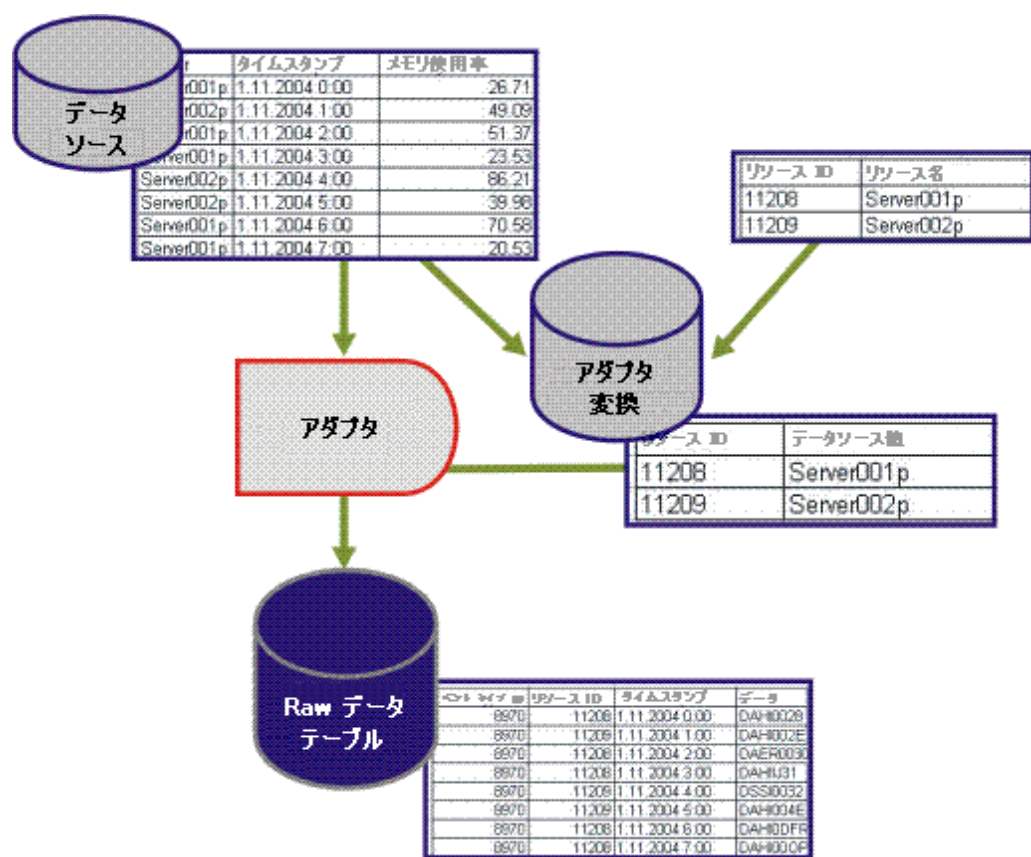
前述のように、リソース ID およびイベント タイプ ID は登録の目的で使用され、データ フィールドおよびタイム スタンプの値は実際の計算に使用されます。

計算用に送信されたイベントの順序およびタイミングを決定するために、タイムスタンプ フィールドもエンジンによって使用されます。

イベント タイプ定義は、入力データ ソースおよび必要な出力に基づいて CA Business Service Insight 内で手動で実行されます。

注: リソース定義は、手動または変換スクリプトを使用して自動で実行できます（詳細については、「[リソースおよびリソース管理 \(P. 60\)](#)」を参照してください）。

以下の図は、データ ソース、アダプタ変換テーブル、アダプタおよび CA Business Service Insight Raw データ テーブルの間の連携を示します。



メトリック登録

リクエストされるデータを相関エンジンが認識できるように、メトリックはその存在と要件を相関エンジンに登録する必要があります。

メトリック登録は、イベント（計算に含める必要のあるイベントのみ）を受信するためのメトリックによるリクエストです。このリクエストは、イベント識別子のイベントタイプおよびリソースの状態によって実行されます。

登録は単一のリソースまたはリソースのグループに基づいて実行することができます。

例:

情報メトリック「サーバXのダウンタイム数」の場合、および、データソースがサーバのアップ/ダウン時に通知を行い、その通知が特定の時間におけるサーバのアップ/ダウンの状態を示すことを前提とした場合、登録は以下のとおりです。

イベントタイプ：サーバアップ/ダウンイベント

リソース：サーバX

上記の登録に基づいて、リソースフィールドにサーバXがあり、さらにイベントタイプフィールドにアップ/ダウンの定義もあるすべてのイベントをエンジンは除外することになります。

契約が CA Business Service Insight システムでアクティブになると、すべてのメトリックはそれらの計算に必要な関連イベントを登録します。これらのリクエストに基づいて、相関エンジンは各ビジネスロジックに関連するイベントをマークします。計算が開始されると、関連するイベントが計算のために各メトリックに送信されます。

リソースおよびリソース管理

動的な登録がを可能にするために、リソースは、その一意の名前/識別子によって個別に配置するか、または論理グループとの関係によって配置することができます。

例:

メトリック「データ センター サーバの全体的なダウンタイム数」の場合、登録は以下のとおりです。

イベント タイプ: アップ/ダウン イベント

リソース: データ センター サーバのもとにタグ付けされるすべてのリソース (これはリソース グループになると考えられます)。

リソース ライフ サイクルについて

リソースは、時間の経過に伴ってその特性を変更できる物理的または論理的なエンティティです。リソースは、ある時点で、特定のサービス コンポーネント、契約関係者などに配置され、その後、将来ある時点で再配置される場合があります。任意の時点で計算を実行できるようにするために、その正確な時間のリソースの設定および配置に基づいて、これらの変更または再配置はそれぞれ **CA Business Service Insight** によってキャプチャされます。

リソースの変更またはその配置はいつでも行なうことができますが、そのリソースの新バージョンを作成する必要があります。新しいバージョンにはそれぞれ有効日（変更が発生する日）を設定する必要があります。その後、変更は将来に引き継がれます。ただし、その同じリソースの後続バージョンで追加の変更が発生した場合はこの限りではありません。この新しいバージョンがアクティブにされて有効になった後は、計算エンジンのみがすべての変更を認識および利用することができます。このプロセスは、リソースの「コミット」と呼ばれます。

CA Business Service Insight では、1つの手順で複数のリソース配置を処理する方法もあります。この方法では「変更セット」が使用されます。変更セットにより、トランザクションデータベースが動作する方法と同様に、単一の「トランザクション」で大量のリソース変更を行うことができます。変更セットに配置されているすべてのリソースに対してすべての変更を行なうには、変更セット全体に対する操作を実行した後に、1つの手順で変更セットをコミットします。

リソースおよびそれらの変更を処理する際に、計算エンジンについて以下の点を考慮することをお勧めします。

- 特定のリソースまたはリソースのグループ（変更セット）に対する変更をアクティブにする（コミットする）ときは、システムで影響を受けるものについて検討してください。リソース モジュールの変更により再計算がトリガされる場合があるので、リソースの有効化日付および1つの操作でアクティブにされる変更の数を最適化することが重要です。
- **一括更新**：多くのリソースに対して同じ変更を適用（一括更新）することができます。リソース モジュールを変更すると再計算が発生する場合があるので、この操作を最適化することが重要です。

上記の例はリソースを直接扱うものではなく、リソースの機能や場所（上記の例ではリソースの機能（データ センター サーバ））への論理的な配置を通じてリソースを扱っています。

データ センターに保持された個別のサーバごとにイベントがリクエストされる場合、登録リクエストは非常に煩雑になる可能性があります。1つの問題は、参照されるリソースの数です。もう1つの問題は、データ センターのインフラストラクチャが定期的に変更されることです。その結果、データ センターに含まれていたサーバが現在はそこに存在しなくなったり、新しいサーバが追加されていたりする場合があります。そのため、動的なリストにする必要があります。

上記の例に基づいて、論理的なエンティティ（論理グループ）を介してリソースを扱うことができるように、論理グループにリソースを添付する必要があることは明らかです。さらに、論理グループが常に変更されている場合は、論理グループ自体の管理が必要になることもあります。

リソース配置は、リソースのタグ付けに関する CA Business Service Insight の方法です。リソースは、複数のグループ、リソース タイプ、契約関係者、またはサービスに配置することができます。リソース配置は CA Business Service Insight バージョン コントロールによって管理されます。

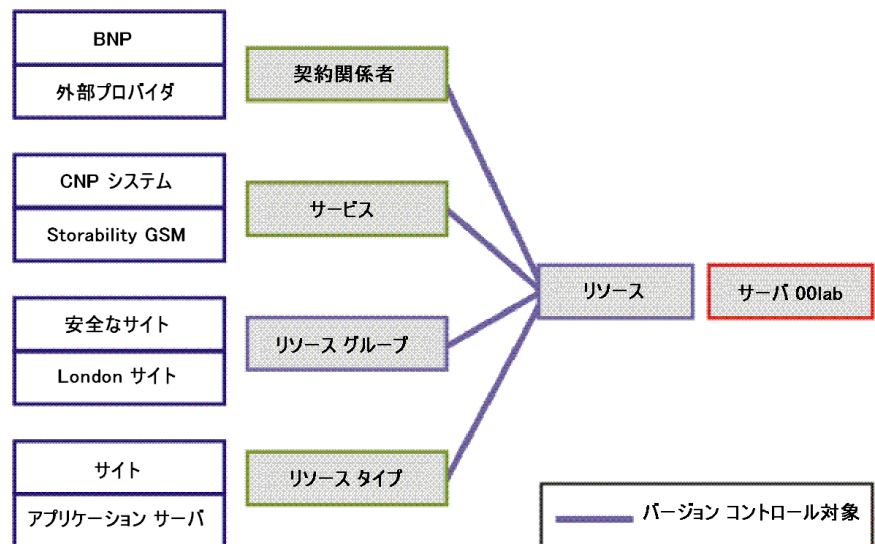
計算に含めるために利用できるリソースは、システム内で現在有効になっているリソースによって決まります（その時点で計算されている時間範囲に関連します）。

ここで上記の例に戻ります。

データ センター サーバの全体的なダウンタイム数

システムでは、データ センターはデータ センター内のすべてのサーバが配置されるサービスとして表すことができます。また、「データ センター サーバ」と呼ばれるリソース グループとして定義することもできます。これらは、この特定のケースでリソース配置のために選択できる 2 つの代替方法ですが、利用可能なオプションはほかにもあります。

以下の図は、リソースが添付される可能性のあるエンティティ、およびそれらの論理的な用途を示します。



リソース グループは、計算に必要なリソースの要素（リソースの場所やリソースに含まれる技術など）をすべて反映できます。

これらのエンティティにリソースを配置する主な目的は、計算要件を満たすこと、およびモデルを可能な限り動的な状態に維持することです。

カスタム属性

リソースに対して利用可能な別の機能は、カスタム属性を提供する機能です。それぞれのリソース タイプにはカスタム属性を追加でき、逆にその特定のリソース タイプとして定義されているそれぞれのリソースは、これらの属性を継承しています。

以前の例を使用して、データ センタ サーバのそれぞれに対してその IP アドレスも関連付けます。データ センタ サーバのそれぞれが 'Data Centre Server' リソースで定義される場合、'IP_Address' と呼ばれるカスタム属性がデータ センタ サーバのリソース タイプに追加されることを確認します。このようにして、各リソース（サーバ）は、カスタム属性を通じて IP アドレスと関連付けられます。

注： 詳細およびサンプルのケース スタディについては、「[カスタム属性を使用したケース スタディ \(P. 268\)](#)」を参照してください。

クラスタ化メトリックについて

クラスタ化メトリックは、リソースのグループに関するサービス レベルを計算するために設定されるメトリックです。その計算は、リソースごとに個別に行われます。この場合、異なるリソースの登録のたびにメトリックを複製する必要はありません。

クラスタ化は、ターゲット サービス レベルが添付されていないか、または同じターゲット サービス レベルを持つリソースのグループに対してのみ実行することができます。たとえば、すべてのアプリケーション サーバの可用性はサーバ当たり 99.9% 以上にする必要があります。この場合、メトリックは、すべてのアプリケーション サーバが含まれるリソースのグループに対してクラスタ化されます。エンジンはサーバごとに個別のサービス レベルの結果を計算します。各サーバのターゲットは 99.9% です。

このタイプのクラスタ化に加えて、CA Business Service Insight は「ロールアップ」タイプのクラスタ化をサポートし、単一のメトリックによる複数レベルのリソース（およびグループ）のレポートを可能にします。これにより、複数レベルのリソース階層でサービス レベルを計算できるため、このリソース構造の関連エンティティ間のドリルアップ/ダウンが可能になります。[メトリックのクラスタ化] タブから、このページの関連するオプションを選択することにより、このオプションを有効にすることができます。

☐ このメトリックはクラスタ化されません。

☒ このメトリックは以下のとおりクラスタ化されます：

クラスタ化対象のアイテム:

☒ 動的

☐ 第 1 レベルのみ: 直接メンバ

☐ すべてのレベル: リソースのみを含む

☒ すべてのレベル: リソースおよびリソース グループを含む

☐ 静的: 以下のアイテムのみが含まれます：

システムのデータ モデルの構築

データ モデリング プロセスの一環として、必要なコンポーネントがデータ ソースおよび計算要件に基づいて特定されます。

データ モデリング プロセスおよびそれらの定義で特定する必要のあるコンポーネントのリストを以下に示します。

イベント名

CA Business Service Insight に表示されるイベントの名前。可能な限りわかりやすい名前にする必要があります。

イベント動作	指定されたイベントの動作。データ ソースから受信される時期、条件などです。
タイムスタンプフィールド	イベント タイムスタンプ] として使用されるデータ ソースのフィールド。
イベント タイプ フィールド	イベント タイプに変換されるデータ ソースでのフィールド。レポートのタイプを示します。 イベント タイプの定義は手動であり、1 回で完了するのが理想的であるため、異なるイベント タイプの数は可能な限り最小化することが重要です。
データ フィールド	データ フィールドとして取得されるデータ ソースのフィールド。
リソース フィールド	リソースに変換されるデータ ソースのフィールド。ライフサイクルが比較的固定されているレポートに必要なエンティティが含まれます。リソースはライフサイクルが定義されたエンティティです。システム内で変更を動的に管理することができます。 リソースのライフサイクルを参照すると、前述のように、リソースが追加されたり、異なるサービスや他のエンティティにリソースの配置が変更されたりする頻度がわかります。 CA Business Service Insight 内のリソースとして変換されるフィールドは、配置変更をほとんど行わないようにする必要があり、その他の変更も制限するようにしてください。

最後に、上記のすべての定義に基づくコンポーネントを以下に示します。

登録基準	登録基準を定義します。登録基準は、メトリックが登録するイベントタイプおよびリソースを定義します。 リソースに対するリクエストは、リソースでの登録によって直接実行するか、または配置エンティティ（サービス、契約関係者、リソース グループ、リソース タイプなど）によって実行することができます。 この定義は登録機能によって決定されます。 もう 1 つの方法はメトリックによる登録です。この場合、現在のメトリックは別のメトリックから出力（サービス レベルの結果）を取得し、それらを入力として使用します。 また、複数のメトリックからの結果を入力として使用することもできます。

登録を定義するためのガイドライン

登録を定義するには、以下のガイドラインを使用します。

- イベント タイプのみで登録を設定しないでください。 計算要件にリソース フィルタリングが必要ない場合でも、少なくともリソース タイプにはリソース フィルタリングを追加します。

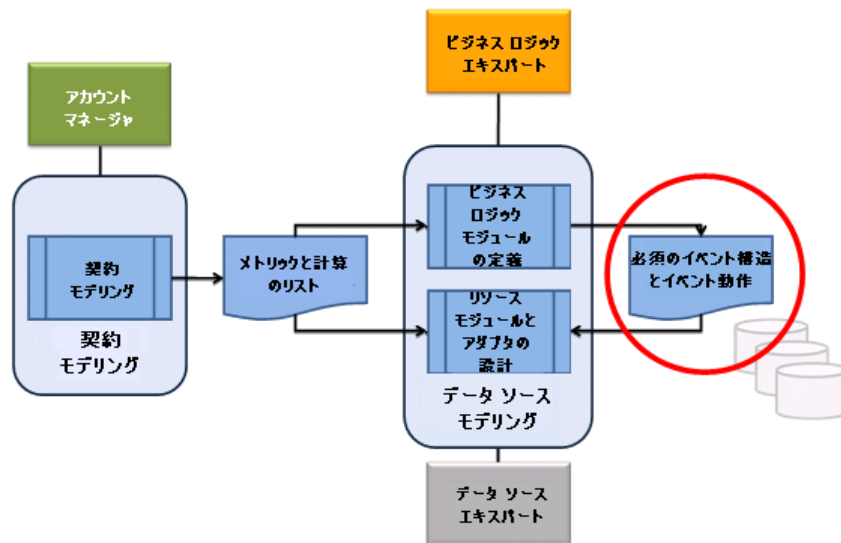
たとえば、アプリケーション サーバの全体的な平均応答時間を計算する際に、応答時間イベントがそれらのアプリケーション サーバに関連付けられている場合にのみ、応答時間イベントをレポートする必要があります（つまり、イベントに関連付けられたリソースのリソース タイプが「アプリケーション サーバ」の場合です）。このような場合、他のタイプのリソース（同じイベント タイプを使用してデータを送信し、イベント タイプを識別するためのサイトやルータなど）がシステムに存在する可能性があります。 レポートされるリソース タイプ（「アプリケーション サーバ」）は、3 番目のパラメータとして登録 コマンドに追加する必要があります。

この理由は、リソースの変更が発生したとき、エンジンが、このリソースに関連付けられたメトリックを「再計算が必要」としてマークするためです。メトリックがイベント タイプのみで登録する場合、エンジンはすべてのリソースにメトリックが登録されたと見なすので、各リソースのアクティブ化での再計算が必要なため、そのメトリックをマークします。これを回避するには、リソース タイプの「オプションの」3 番目のパラメータを使用する必要があります。

- 登録するための最も効率的なメソッドは、契約関係者およびサービスです。この方法でリソースを整理すると、システムでのデータ レイヤとビジネス レイヤの間の論理関係を表現することができます。これらのエンティティによってリソースを登録する場合、別の契約で使用される計算式、または別のサービスに使用される計算式を変更する必要はありません。メトリックのコンテキスト契約およびサービスは、関連する契約関係者およびサービスを定義します。このタイプの登録に定義されたビジネス ロジック計算式は簡単に再利用できます。それらの計算式は登録の変更が必要ないためです。

注: 登録は、各メトリック内の[登録]タブでも処理できます。このインターフェースでは、選択されたメトリックに関する必要な手順を示すウィザードが提供されます。

データモデリング段階の出力(データソースエキスパートおよびビジネスロジックエキスパート)



ソリューションの実装段階に進むために、データソースエキスパートは以下の点を確認する必要があります。

- 以下のものを含む、関連するデータソースについての十分な知識
 - サンプル履歴データ
 - 通信方法
 - リソースモデルのキーコンポーネントに使用される各データソースのキーフィールドについての知識
- 以下の定義を含む、各データソースに関する構造と動作を示すイベントタイプの固有のリスト
 - イベント名
 - イベント動作
 - タイムスタンプフィールド
 - イベントタイプフィールド
 - データフィールド
 - リソースフィールド

注: 取得する情報のサマリドキュメントを作成する必要があります。

第 3 章：実装

このセクションには、以下のトピックが含まれています。

[実装 - 概要 \(P. 70\)](#)

[フレームワークのセットアップ \(契約マネージャ\) \(P. 73\)](#)

[テンプレートライブラリのセットアップ \(契約マネージャ\) \(P. 73\)](#)

[契約の作成方法 \(契約マネージャ\) \(P. 74\)](#)

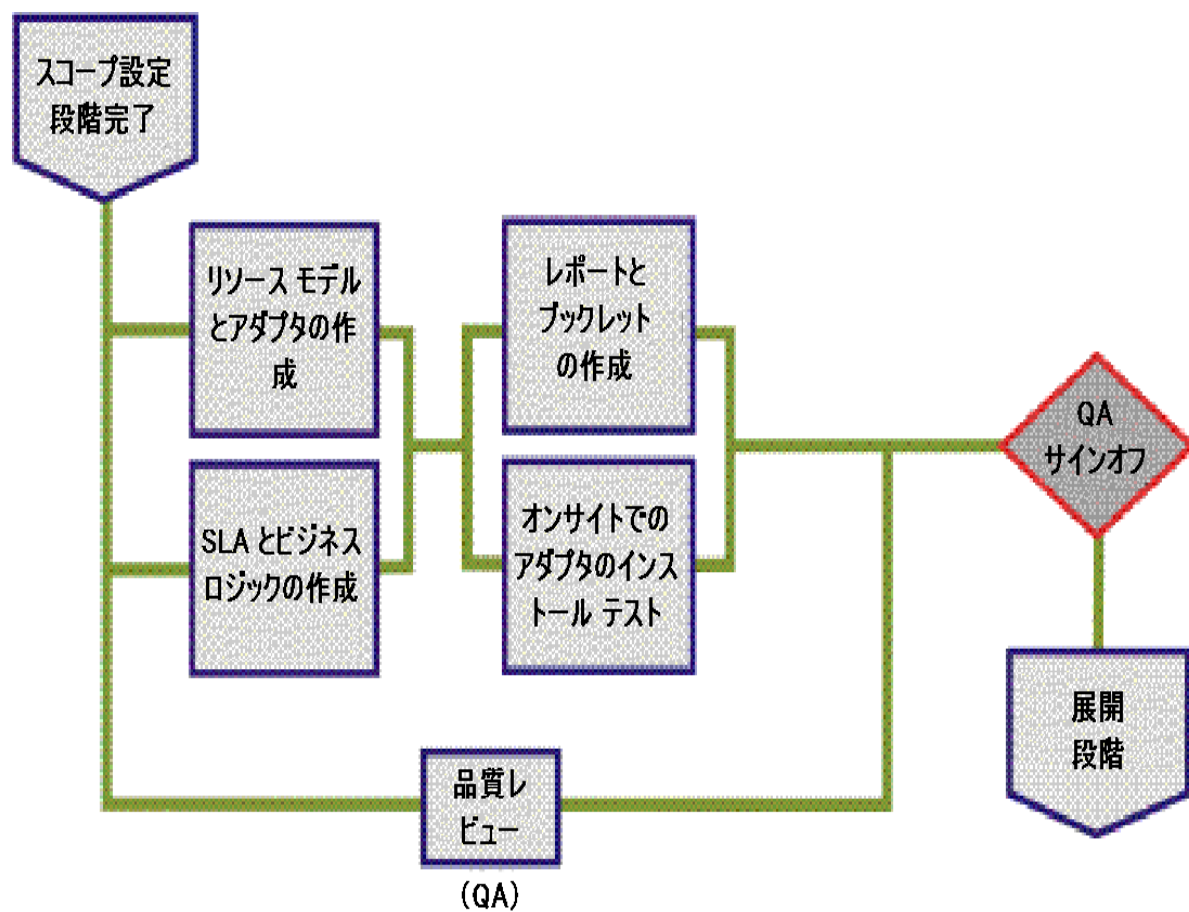
[データ収集 \(データ ソース エキスパート\) \(P. 79\)](#)

[ビジネス ロジック スクリプティング \(ビジネス ロジック エキスパート\) \(P. 148\)](#)

[契約をアクティブ化する方法 \(契約マネージャ\) \(P. 196\)](#)

[成果物の作成 \(契約マネージャ\) \(P. 200\)](#)

実装 - 概要



この章では、プロジェクトの実装段階の背後にあるプロセスおよび推論について説明します。上記の図に示すように、設計段階の後に実装段階が続き、その後にインストール段階および展開段階が続きます。

実装段階の目的は、設計段階で定義された、CA Business Service Insight システムのすべての項目およびオブジェクトの実際の作成を契約マネージャが完了できるようにすることです。実装段階では、チームは完全な展開およびインストールに向けてシステムを準備する必要があります。

この段階を開始するには、設計段階をすべて完了し、必要な目標をすべて考慮して、正しく定義しておく必要があります。設計段階が正しく完了していないか、または、すべての契約、メトリック、アダプタなどが明確に定義されていない場合、システムに関する問題が発生したり、実装段階で実装する必要のあるデータが欠落したりします。設計レビューが終了しない限り、実装段階を開始しないでください。

また、システムのインストールおよび展開に進む前に、実装段階を正しく完了することも重要です。品質レビューが終了しない限り、次の段階には進まないでください。

設定プロセスには、以下の手順を実行する契約マネージャが関与します。

- サービス カタログのセットアップ
- 契約の作成
- 契約のアクティブ化
- セキュリティの設定
- レポート/アラート/ダッシュボードの作成

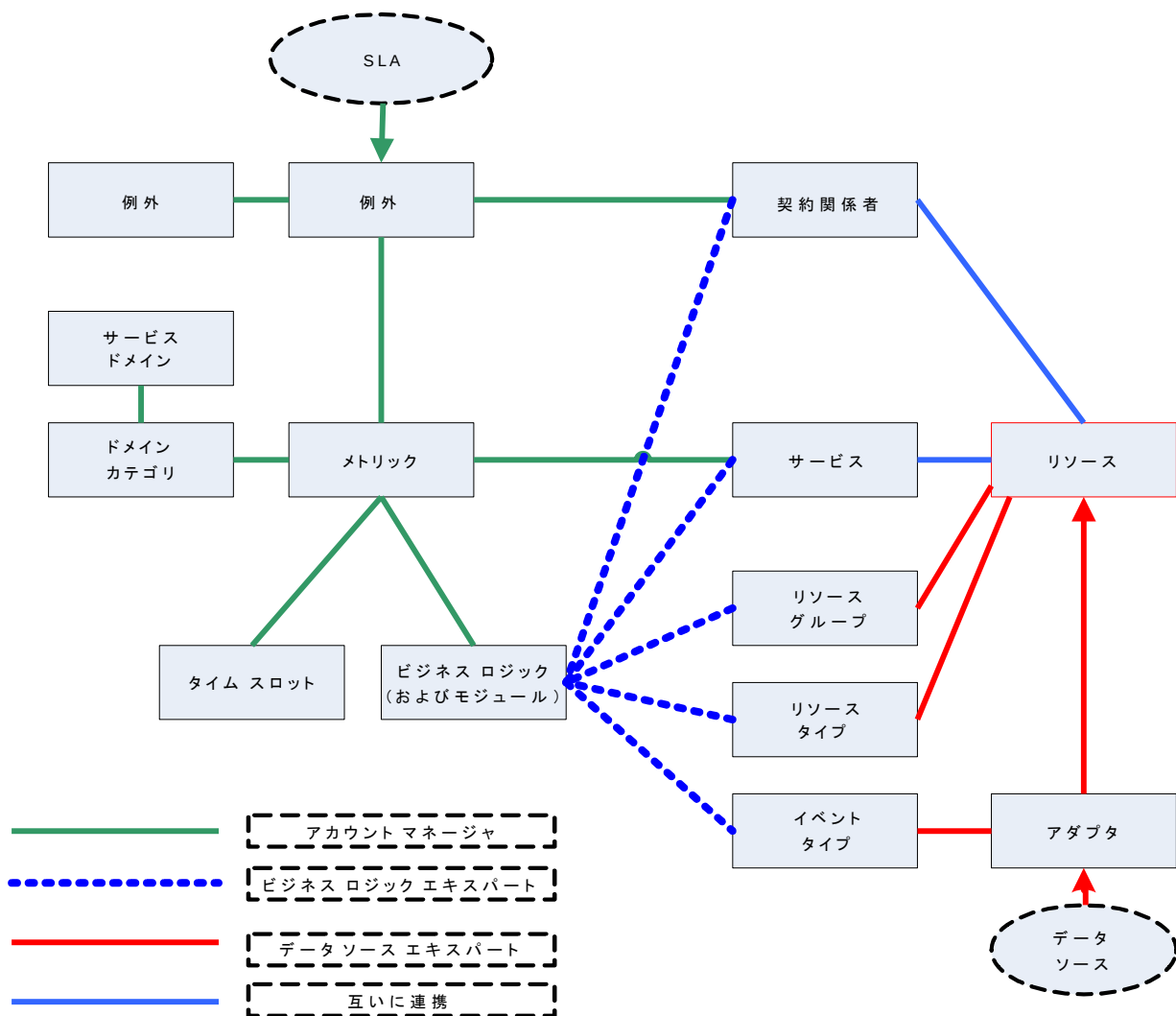
データ ソース エキスパートは、アダプタ設定およびデータ ソースとの統合を行う必要があります。また、データ ソース エキスパートは、リソース変換手順を実行し、データ ソース構造を CA システム内に定義されたエンティティにリンクする必要があります。この手順は全工程を通じて重要であり、契約マネージャの指導が必要になる場合もあります。

これに加えて、ビジネス ロジック エキスパートは、設計段階からの計画に従って、すべてのメトリックのビジネス ロジックを作成する必要があります。これには、期待される計算機能性を実現するために、必要なすべてのモジュールの作成および関連パラメータの設定が含まれる場合もあります。

上記のすべてのポイントについて、この章の各セクションでさらに詳しく説明します。

注: この段階での不適切な選択は CA Business Service Insight の運用に悪影響を及ぼす可能性があり、後の段階での変更が困難または不可能になる場合があることに留意することが、契約マネージャにとって重要です。

以下の図は、全体的な論理ワークフローを示します。



フレームワークのセットアップ(契約マネージャ)

フレームワークでは、以下のことが可能です。

- サービス、サービス グループ、サービス ドメイン、および単位の定義
- テンプレート(ビジネス ロジック テンプレートとタイムスロット テンプレート) およびビジネス ロジック モジュールの作成とメンテナンス
- リソース タイプのカスタム属性の管理

この段階では、設計段階で特定されたシステム全体のエンティティはすべて、アプリケーションのフレームワーク セクションで作成されます。これらのフレームワーク エンティティがシステムに含まれる場合に限り、契約およびそれらの関連メトリックの作成に進むことができます。

フレームワークの構築では、以下の新しい項目を追加する必要があります。

- サービス
- サービス グループ
- サービス ドメインおよびドメイン カテゴリ
- 測定単位
- タイムスロット テンプレート
- 契約関係者
- カスタム属性

上記の各項目の詳細については、オンライン ヘルプを参照してください。

テンプレート ライブラリのセットアップ(契約マネージャ)

テンプレート ライブラリでは、以下のものを定義および管理することが可能です。

- テンプレート ライブラリ
- テンプレート フォルダ
- サービス レベル テンプレート
- 契約テンプレート
- ユーザ アクセス権限のセキュリティ設定

上記の各項目の詳細については、オンライン ヘルプを参照してください。

契約の作成方法 (契約マネージャ)

この段階では、設計段階で定義された契約およびそれらの関連エンティティがシステムで作成されます。

以下の手順に従います。

1. 新規契約を追加し、契約の一般詳細を適用します。
2. 契約ごとに、そのメトリックを定義し、それらの一般詳細を適用します。

この段階では、契約内容の一般詳細のみが適用され、契約のメトリックのビジネスロジックおよびクラスタ化は含まれません。

これらの手順に関する以下の説明では、この段階で考慮する必要のあるいくつかの重要な点を強調しています。オンラインヘルプでは、これらの手順に関する全詳細が記載されています。

手順 1：新規契約を追加し、契約の一般詳細を適用します。

契約の定義には以下のことを含める必要があります。

- 契約の名前の設定。
- 関連する契約関係者の選定。
- 関連サービスの添付。
- 契約の発効日の設定。契約の発効日は、関連エンジンがこの契約のサービスレベルを計算する日付範囲です。レポートの結果は発効日にのみ利用できます。日付を設定する際には、契約関連のレポートの可用性に関する要件、および利用可能な **Raw** データを考慮することが重要です。
- 契約のタイムゾーンおよび通貨の設定。この定義はレポートを目的とし、この契約に関するレポートが関連タイムゾーンに従うようにします。通貨定義により、レポートエンジンは、ペナルティ計算式のペナルティ結果を表示する通貨を決定できます。

手順 2：メトリックの一般詳細を追加します。

契約の準備が整ったら、契約内にメトリックを作成します。メトリックを定義するプロセスでは、以下の手順を実行する必要があります。

- メトリック名の設定。
- メトリック詳細（サービス、ドメイン、単位、タイムスロット、タイムゾーンなど）の適用。
- ダッシュボードしきい値の設定（「[ダッシュボードページの設定](#) (P. 214)」を参照）。

- 関連メトリック（該当する場合）およびそれらの関係の添付。
- メトリックがエンジンによって計算される粒度の定義。
- 目標ステートメントとパラメータの設定。

この段階では、メトリックの定義には以下の項目がまだありません。

- ビジネス ロジック 計算式/モジュールおよび登録
- クラスタ化の定義

システム インフラストラクチャが構築され、ビジネス ロジック 計算式が開発およびテストされた後にのみ、これらの項目は契約のメトリックに含まれます。

注: 上記の手法に対する代替方法は、すぐに契約を定義するのではなく、システム内のサービス レベル テンプレートを最初に開発することです。これにより、後で追加の契約を作成するために使用できるテンプレートの作成が可能になります。場合によっては、既存のサービス レベル テンプレートを別の **CA** インスタンスからシステムにインポートできます。これにより、最初から契約を作成する場合に比べて、有利なスタートを切れる場合があります。独自のサービス レベル テンプレートを作成する方法の詳細については、「[サービス レベル テンプレートの作成](#) (P. 77)」を参照してください。

場合によっては、最初にサンプルの契約を作成して、すべて予想どおりに正しく動作するかどうかをテストすることをお勧めします。この契約からレベル テンプレートを作成し、それをサービス カタログに格納することで、システム内のすべての契約に対して確実な開始点を提供することができます。

サービスからの契約の作成

サービス カタログを使用して、（複数のサービス レベル テンプレートに基づいて）テンプレートから、またはテンプレートなしで、システムに契約を作成すると、非常に効率が高くなり、高度な再利用が可能になります。これは数多くのさまざまな契約に適用することができます。一般的に、サービス レベル テンプレートには、特定のサービス コンポーネントに適用可能な事前定義済みのメトリック群が含まれています。必要に応じて 1 つのサービス レベル テンプレートに 1 つ以上のサービス（および関連付けられたメトリック）を持つこともできます。通常、サービス レベル テンプレートのコンテンツはその用途に応じて定義し、要件に従って変更することもできます。

例として組織によって提供されたアプリケーション ホスティング サービスの場合を考察します。組織は、パッケージに含まれる内容に応じて「ブロンズ」、「シルバー」、および「ゴールド」のような 3 つの異なるサービス パッケージをカスタマに提供します。パッケージごとにサービス レベル テンプレートを作成するのは、サービス レベル テンプレートの良い使用例です。

一度定義すると、これらの定義を使用して新規のカスタマ契約を効率良く作成することができます。たとえば、カスタマ ABC がゴールドのアプリケーションホスティングパッケージを契約することを決めたとします。ユーザは、以下のとおりにサービスレベルテンプレートをを使用して、この新規契約をシステムディレクトリに作成することができます。

[契約] ページで、[新規追加] - [サービス カatalogの使用] をクリックし、[準拠するテンプレート] または [テンプレートなし] のどちらかを選択します。次に、[契約ウィザード] にしたがって契約の作成を完了します。[準拠するテンプレート] を選択する場合は、テンプレート設定を指定する必要があります。

[契約ウィザード] にサービスレベルテンプレートのリストが表示され、ユーザは契約に含めるサービスレベルテンプレートを指定できます。複数のサービスレベルテンプレートから特定のメトリックを選択する、または定義全体を選択して含めることができるので留意しておいてください。この例では、ゴールドホスティングパッケージのすべてのメトリックになります。トップレベルを選択すると、子ノードが自動的にすべて選択されるので注意してください。また、必要に応じて異なるサービスにメトリックを割り当てることができる点も留意しておいてください。ただし、デフォルトでは、定義にあるように同じサービスコンポーネントに割り当てておくことになっています。

必要なすべてのメトリックの選択が完了したら、[次へ] ボタンをクリックし、新規契約テンプレートへメトリックを転送します。契約名および一般詳細を入力するように指示されます。[保存して次へ] をクリックして契約を作成します。

完了後は以下のオプションがあります。

- [契約ウィザード] を続行してパラメータを定義し、[メトリック ウィザード] を実行します。これを実行すると、[ウィザード] インターフェースが開き、契約メトリックのカスタマイズが可能になります。ウィザードでは各メトリックを参照することができ、また、目標ステートメントのメトリックパラメータ、メトリック名、タイムスロット、説明などの利用可能なフィールドを変更することで修正ができます。各メトリックを完了すると、ウィザードは[契約メトリック] ページに戻り、ユーザは新規契約を終了および保存することができます。
- [契約] ページを開き、契約を表示、または編集します。

サービス レベル テンプレートの作成

サービス レベル テンプレートの作成はとても簡単なプロセスです。既存の契約（「契約の詳細」ページ）から含めるメトリックをすべて選択し、「サービス レベル テンプレートとして保存」をクリックします。

次のウィンドウでは、サービス レベル テンプレート名の入力が必要です。その後、サービス レベル テンプレートを保存できます。保存すると、選択したメトリックに関連付けられたすべてのタイムスロットが、タイムスロット タブに含まれます。この状態から、柔軟性を持たせるために、サービス レベル テンプレートをさらにカスタマイズすることもあります。この場合には、パラメータをメトリックに追加し、これらのパラメータを目標ステートメントを経由して各メトリックへ公開することが含まれます。また、一部またはすべてのメトリックが参照できるサービス レベル テンプレート パラメータ（契約パラメータに類似）の作成も含まれる可能性があります。完了すると、サービス レベル テンプレートをほかの契約に展開することが可能になります。

契約ライフサイクルおよびバージョン管理

契約の設定が完了したら、コミットを行う必要があります。このアクションは、計算エンジンが契約内のすべてのメトリックのサービス レベルの計算を開始することを許可します。契約をコミットすると、ステータスが「暫定」（編集可）から「有効」（編集不可）に変更されます。この契約にさらに変更が必要な場合は、契約の新規バージョンを作成する必要があります。新規バージョンに同じ発効日を選択した場合、変更を完了して新規バージョンをコミットした後、旧バージョンが完全に上書きされます。これは、エンジンをトリガすることにもなり、旧バージョンと異なるメトリックの再計算が行われます。有効バージョンは部分的にオーバーラップすることもできます。たとえば、発効日の途中で契約内の一部のメトリックのターゲットが変更された場合です。この場合、2 番目のバージョンの発効日がアクティブになるまで、旧バージョンが使用されます。このとき、2 番目のバージョンは、計算の有効ステータスを推測します。

以下の表に、再計算の影響範囲および期間に対するユーザ変更を示します。変更は契約固有のバージョン内の特定メトリックに影響し、また、複数の契約間のメトリックおよび契約バージョンのメトリックにも影響します。

変更	影響範囲	影響期間
メトリック内で行われた変更内容		
メトリック詳細の変更 - ビジネス ロジック 計算式	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
メトリック詳細の変更 - 目標値	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから

メトリック詳細の変更 - トラッキング期間	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
メトリック詳細の変更 - メトリックのパラメータ	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
メトリック詳細の変更 - タイムゾーン	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
メトリック詳細の変更 - クラスタ化	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
契約の詳細の変更 - 発効日	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
契約の詳細の変更 - タイムスロット	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
契約レベルパラメータの変更	契約固有のバージョン内のすべてのメトリックに影響	契約の有効バージョンの始めから
SLALOM モジュールの変更	すべての契約および契約バージョン内の、変更されたスラロームモジュールに添付されたすべてのメトリックに影響	契約の有効バージョンの始めから

一般的な操作

メトリック リソース モデル または変更セットの変更 (CA Business Service Insight 4.0)	すべての契約および契約バージョン内の、リソースを登録するすべてのメトリックに影響	リソースの変更が発生した日付以前の最も近い状態から
メトリックの遅延イベントを取得 過去のタイムスタンプを持つイベント (Raw データまたは再利用イベント)	すべての契約および契約バージョン内の、リソースを使用するすべてのメトリックに影響	イベントの変更が発生した日付以前の最も近い状態から
メトリック修正データの追加	すべての契約および契約バージョン内の、リソースを使用するすべてのメトリックに影響	修正の変更が発生した日付以前の最も近い状態から

メトリック例外時間の変更、更新、または削除（アクティブ化およびアクティブ化解除）	例外設定および特定の実装によって、契約固有のバージョンのメトリックに影響、または複数の契約間のメトリックおよび契約バージョンのメトリックに影響	例外時間に可能な限り近い
カスタム属性の更新	すべての契約および契約バージョン内の、リソースを登録するすべてのメトリックに影響	リソースの変更が発生した日付以前の最も近い状態から

最後に契約バージョンについて留意すべきキーポイントをいくつか示します。

- 新規バージョンに同じ発効日がある場合、変更されたメトリックのみが再計算され、バージョンの始めから再計算されます。
- 新規バージョンが別の日付の場合、新規バージョン内のメトリックはすべてそのバージョンの始めから計算され、新規バージョンが有効になるまで、旧バージョン内のメトリックはすべてそのバージョンのある時点から再計算されます。再計算の総量は状態の設定によります。
- 発効日を1年にした契約を作成し、期限が切れたら更新することをお勧めします。これにより、1年より長い再計算期間を防ぐことができます。
- 契約の無効バージョン（現在の日付が契約の発効日の終了日を過ぎている）は計算されます（したがって、レポート用に無効バージョンに関連付けられているサービス レベル データを計算するため、システム内ではアクティブなメトリックとみなされます）。

メトリック内のグローバル変数と関連付けられた値は、契約バージョン間で持ち越されません（すなわち、各契約バージョンの最初にビジネス ロジック内の OnLoad ルーチンが呼び出されます）。

注: 段階的シナリオおよびケース スタディについては「[契約モデリングのケーススタディ](#) (P. 235)」を参照してください。

データ収集(データソース エキスパート)

実装プロセスのデータ収集段階では、アダプタを使用します。以下のトピックでプロセスの概要について説明します。

アダプタ機能

アダプタは、データ ソースからデータを収集し、CA Business Service Insight システムへ渡すことを管理するモジュールです。アダプタは、データ ソースから来るデータをフィルタし、操作するため、データがシステムに到達するときには、サービス レベル計算に必要な正しい構造の情報のみが含まれています。

アダプタ プラットフォームは、以下に対する柔軟性を提供します。

- オンラインまたはオフラインで、必要な任意の頻度でデータを受信する
- さまざまなレベル (raw、計算済み、または集約済み) でデータを受信する
- 広範囲のツール タイプからデータを受信する

基本的に、すべてのアダプタは 2 つのコンポーネントで構成されます。

- **汎用アダプタ コンポーネント：**

汎用アダプタ コンポーネントには、ASCII ファイル アダプタ コンポーネントおよび ODBC ベースの SQL アダプタ コンポーネントの 2 つのタイプがあります。これらのコンポーネントはデータ ソースへ接続し、ASCII ファイルとして構文解析を行うか、または SQL クエリを実行します。

- **アダプタ設定ファイル：**

接続先や接続方法、受信対象、汎用 CA トランザクションおよびイベントへのデータ変換方法を明らかにするため、どのアダプタにも設定ファイルが必要です。CA Business Service Insight は、接続するデータ ソースを考慮してカスタマの特徴に合わせた調整ができるデフォルト XML 設定テンプレートを備えた、汎用アダプタ タイプをすべて提供します。XML 設定ファイルは、取得する必要があるフィールド、その識別方法、システム標準化したデータベースへの変換方法などを定義します。

注：[アダプタ ウィザード] は、ユーザ インターフェースに組み込まれており、オンラインで XML テンプレートの基本的なカスタマイズが可能です。アダプタの XML 設定ファイルの作成を目的として使用することもできます。この機能の詳細については、この章の後半で説明します。

アダプタ プラットフォームには再起動/回復メカニズムが組み込まれており、サードパーティ ツールから受信されたデータの問題を処理できます。具体的には、ツール クラッシュ、ネットワークトラブル、欠落データ、重複データ、不要データ、データ間の食い違い、データの妥当性などの問題を処理できます。すべてのアダプタは、データの完全性とすべてのアダプタ メッセージの完全なトラッキングおよびログを提供します。詳細については後述します。

CA Business Service Insight アダプタはサービスまたはアプリケーション (可視または不可視) として実行することができます。CA Business Service Insight アダプタ技術は、暗号化、ハンドシェイク、認証プロセスなどの高度なセキュリティ メカニズムをサポートしています。

ここでは、アダプタ ウィザードが以降のプロセスおよびタスクを自動化するメカニズムであることに留意することが重要です。ウィザード駆動のアプローチを使用する場合、前述の特定のエレメントは常に不可視である可能性があります、これらはすべてウィザードインターフェースの「裏側」に存在します。

アダプタ環境

以下のエンティティは、アダプタとその設定および実行パラメータに関連します。

データソース:

アダプタの接続先およびオリジナル形式データの受信元であるデータソース。

ワークファイル:

アダプタによって生成され、プロセス内に書き込まれている出力ファイル（詳細については、「[ワーク ファイル](#) (P. 85)」を参照してください）。

CA Business Service Insight アダプタ リスナ:

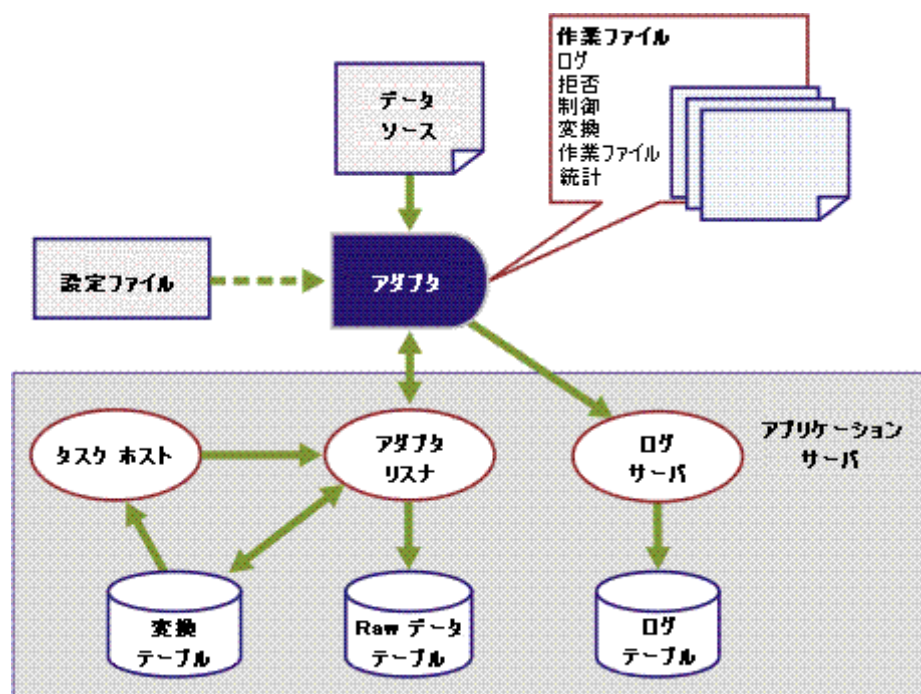
3つのメッセージタイプがアダプタとアダプタ リスナの間で転送されます。

- **コントロール** : リスナによってアダプタに送信された開始/停止/一時停止のメッセージを、アダプタのステータスが変更された場合に送り返します。
- **変換** : アダプタは変換テーブル コンテンツおよび特定の変換値のリクエストを送信します。リスナはテーブルおよび変換値を返します。アダプタ リスナは、変換エントリが変換済みというタスク ホストからの表示を受信します。その後、アダプタにメッセージを送信します。
- **Raw データ** : アダプタによって送信された Raw データ イベントを統一したもの。これらのイベントはパケットで送信され、確認応答が含まれます。

CA Business Service Insight ログ サーバ

アダプタを、メッセージのログをシステム ログに送信するように設定でき、同様にローカル ファイルに書き込むようにも設定できます。ログサーバのポートおよび IP アドレスがアダプタのレジストリ設定に指定および設定されている場合、アダプタはログサーバにもメッセージを送信します。

以下の図は、アダプタ プロセスを各エンティティとの相互作用関係で説明しています。



これらのエンティティと相互作用するアダプタ プロセスの説明については、以下のとおりです。

設定ファイル:

アダプタの設定パラメータの一部またはすべての設定が含まれます。アダプタは、イベント出力を作成するため、アダプタおよびメトリックによって解析に使用される接続メソッドを設定ファイルを使用して決定します。これは XML ファイルで、形式には 6 つの基本的なエレメントが含まれています。

General :

多岐にわたるアダプタの属性 (作業ディレクトリ、出力ファイル、デバッグ フラグなど)。

OblicoreInterface :

CA Business Service Insight サーバとの接続用の属性。

DataSourceInterface :

データ ソースとの接続用の属性 (ファイル パスおよびパターン、接続文字列、SQL クエリなど)

InputFormatCollection :

メトリックを解析し、ソース データの分析と操作を行う。

TranslatorCollection :

解析および操作されたデータ フィールドで構成された統合イベントを構築するためのメトリック。

TranslationTableCollection :

オリジナル データと CA Business Service Insight エンティティ間でデータをマッピングするためのメトリック。

これら 6 つセクションにはそれぞれ、アダプタによるデータ ソースへの接続、必要な情報の取得、CA Business Service Insight 統合イベント構造への解析、および CA Business Service Insight Raw データ テーブルへの格納を可能にする関連情報がすべて含まれています。

メイン ファイル

アダプタは、実行可能ファイルと設定ファイルの 2 つのメイン ファイルから構成されます。実行可能ファイルは一般的なファイルです。そのような実行可能ファイルには、SQL アダプタおよびファイル アダプタの 2 つがあります。

XML 設定ファイルは、特定のデータ ソース要件を格納するために各アダプタ専用になっています。設定ファイルは、データ ソース（名前、場所、接続方法および構造体）、およびアダプタによって生成される出力イベントの構造に関連する情報を指定します。

設定ファイルには、事前定義済み構造化 XML ファイル内の属性用に設定されたパラメータおよび値が含まれます。

新規アダプタを作成するとき、（ターゲットデータ ソース タイプ、フラット ファイルデータ ソース用のファイル、データベース データ ソース用の SQL に応じて）既存の関連する実行可能ファイルを使用し、必要に応じて設定ファイルを変更することが必要です。2 つの構造には、テキストまたは SQL アダプタ用のわずかに異なる構成要素が含まれています。通常、これはアダプタ マネージャ ユーティリティを使用してアダプタを作成するとき、自動でセットアップされます。

アダプタに関連する他のファイルは、CA Business Service Insight システムからデータ ソースを読み取り、またイベントを書き込む過程でアダプタによって作成されたワーク ファイルです。

注: 設定ファイルの変更の詳細については、「[Adapter Configuration Specifications \(アダプタ設定仕様\)](#) (P. 339)」を参照してください。

ワーク ファイル

アダプタ ワーク ファイルは、アダプタが初めて実行されるときに作成され、アダプタの実行時に常に更新されます。

各アダプタには独自のワーク ファイルがあります。ワークファイルの名前はアダプタ設定ファイルで設定する（オプション）、またはデフォルトの名前のままにすることもできます。ワークファイルの場所は「作業フォルダ」によって設定され、設定ファイル内にも設定できます。指定されたパスは、アダプタが存在するカレント ディレクトリ内に関連している場合があります。アダプタが正常に動作するためには、指定されたパスが既存のものである必要があります（またはユーザが作成する必要があります）。

注: パスが存在しない場合、フォルダは自動で作成されません。

設定ファイル内の関連パラメータはすべて **General** セクションに格納されます。ログ ファイルの場所のみがレジストリに設定されるか、またはコマンドラインを介して渡されます。

AdapterStatistics.txt

アダプタは 1 分間隔でこのファイルに統計情報を書き込みます。アダプタの停止時に最終行が書き込まれます。ファイルの各行には次の数が含まれています。

- 受信したイベント
- 無視したイベント
- エラーのあるイベント
- 送信したメッセージ
- 送信したパッケージ

アダプタが実行されるたびに統計を開始します。

RejectedEvents.txt

このファイルには、アダプタが **CA Business Service Insight** への送信に失敗したイベントがすべて含まれ、変換テーブル内に一致する ID のない 1 つの変換要求として定義されています。これは、関連する変換が実行されなかったことを意味します。少なくとも 1 つの値を持つ変換待機中のイベントが、それぞれ **rejectedEvents** ファイルに書き込まれています。

アダプタは実行を開始するたびに、まず **CA Business Service Insight** へ **rejectedEvents** ファイルのイベントを送信すると同時に、変換テーブル内の関連する値と一致する ID の検索を試みます。値が見つかった場合、アダプタはイベントを送信し、ファイルから削除します。一致する値が見つからない場合、イベントは **rejectedEvents** ファイル内に残ります。

アダプタ設定ファイルの **RejectedEventsUpperLimit** パラメータを設定することで、拒否されたイベント数の上限を設定することができます。上限に達すると、アダプタが新規レコードの読み取りを停止し、「ブロック」ステータスを入力します。これは、デバッグ出力がアダプタの実行中に画面に表示されるときに見ることができます。大文字 **B** が連続する文字列があれば、アダプタがブロックされ、さらにデータをロードする前に変換が必要な保留中エントリがあることを示しています。

保留中イベントは **XML** 形式でファイルに書き込まれています。ファイルのイベント例は以下のとおりです。

```
<rejectedEvent
  createDate="1062330841"
  translator="Translator">
  <event
    inputFormat="InputFormat">
      <field name="resource" type="3" value="Server333p"/>
      <field name="timestamp" type="4" value="1036108800"/>
      <field name="memory_utilization" type="2" value="26.71"/>
      <field name="cpu_utilization" type="2" value="78.85"/>
    </event>
  </rejectedEvent>
```

アダプタ ログ

アダプタ ログはアダプタがメッセージのログを書き込むファイルです。

アダプタ ログ ファイルの参照にはログ ブラウザ ユーティリティの使用をお勧めします。

アダプタ設定ファイルの **LogDebugMode** パラメータを変更することで、このログ ファイルにレポートするレベルの設定が可能です。「yes」に設定された場合、アダプタはログに正常な表示メッセージを書き込み、また同様に元のレコード、結果解析、および指定イベントも書き込みます。

このパラメータは通常、アダプタのテストおよびモニタリング中に「yes」に設定します。

デフォルトでは、ファイル サイズの制限数は **1MB** です。上限に達するとアダプタが「_old」を付加してファイル名を変更し、新しいログ ファイルを作成します。アダプタは、**2MB** 以内のメッセージのログの格納を見込んでいます（旧ファイルに **1MB**、現在のファイルに **1MB**）。

ログ ファイルのサイズ制限は各アダプタのレジストリのエントリとして設定することができ、最大 **10MB** です。レジストリのエントリは **LogFileMaxSize** と命名され、特定のアダプタ下に **KB** の倍数の値で定義されます。

DataSourceControl.xml

DataSourceControl.xml ファイルはアダプタによってデータ ソースへのアクセス制御に使用され、実行時に常に最終地点からの読み取りを続行するようにします。

ファイル アダプタは、最後に読み取りを行ったファイルの名前、最終行、およびアクセスしたファイル内の位置を保持します。次にアダプタが実行されるときには、アダプタが DataSourceControl.xml ファイルの情報を使用して、その場所からファイルにアクセスします。このメカニズムを使用すると、アダプタは実行時ごとに新規レコードのみを読み取ることができます。

アダプタは、ソース ファイル上で直接作業せず、最初に現在のファイルをワーク ファイルにコピーします。したがって、同じ情報がワーク ファイルとソース ファイルで維持されます。ソース ファイルに追加があった場合は、新規レコードのみがワーク ファイルにコピーされます。

設定ファイルのパラメータ DeleteAfterProcessing を「yes」に設定して、アダプタを一度処理したソース ファイルを削除する設定にした場合、アダプタはソース ファイルの情報を保存しません。完了すると、アダプタは設定ファイル内で定義されたファイル パターンと一致する作業フォルダ内の新しいファイルを読み取ります。

DeleteAfterProcessing を「no」に設定した場合のみ、前回のファイルの新規レコードをチェックします。何もない場合は、辞書式順序に従って次のファイルに移動します。そのため、ソース データ ファイルを命名するときは、正しい順序で読み取られるように順番に文字を増やす命名パターンになるようにします。これを実現するためには、たとえば逆の日付値 (yyyymmdd-hhmmss) をファイルに付加します。例：

- DataSourceABC20070517-14:00:00.csv
- DataSourceABC20070517-15:30:00.csv
- DataSourceABC20070517-17:00:00.csv 以下同様

ファイル アダプタ用の DataSourceControl.xml の例を以下に示します。

```
<AdapterControl Save="end" LastSaveTime="2005/05/20 13:06:39">
<Data><WorkData><LineNumber>0</LineNumber>
<FilePattern>c:\adapters\callsadapter\*adapterpca.csv</FilePattern>
<file name></FileName>
<BasicPosition>0</BasicPosition>
</WorkData>
<NonDeletedFiles>
<File NamePattern="c:\adapters\callsadapter\*adapterpca.csv">
<file name>2005adapterpca.csv</FileName>
<LastLine>25/04/2005,5925,NN4B,12,12,0,10,0,11</LastLine>
<LastPosition>15427</LastPosition></File>
</NonDeletedFiles>
</Data>
```


</AdapterControl>

SQL アダプタは、各クエリが実行されるたびにクエリ キー フィールドの前の値を保持します。キー フィールドは、送信先のデータベース テーブル内のレコードに使用する一意の識別子です。アダプタは、次のクエリを構築する場合にこれらの値を使用します。これにより、アダプタは新規レコードのみを読み取ることができます。

たとえば、以下の SQL ステートメントを、あるトラブル チケット データの取得に使用するとします。

```
Select ticket_id, status, organization, open_date, respond_date, resolved_date,
record_modified_date from t_ticket_data;
```

この例では、データ ソースから最新のレコードをすべてキャプチャするためにクエリ キー フィールド **record_modified_date** が選択されており、確実に最新情報を取得します。これは、このフィールドが前回のアダプタ実行時以降に発生した新規チケットを生成し、また既存のチケットを更新するためです。クエリ キー フィールドとしてこのフィールドを選択することにより、アダプタは実行時に次のセクションをクエリの最後に自動で付加します。

```
where record_modified_date > :previous_value order by record_modified_date asc
```

したがって、新規レコードのみが取得されます。クエリ キー フィールドを選択する際には考慮すべき事項が多くあり、これは常にデータ ソースの動作および取得したデータで何を行うかに依拠しています。また、前述の例で選択されているフィールドが必ずしもすべての状況に対する最適な選択だとは限らないことに注意してください。

SQL アダプタ用の DataSourceControl.xml ファイルの例を以下に示します。

```
<AdapterControl Save="end" LastSaveTime="2005/05/20 15:59:02">
```

```
<Data><QueryCollection>
```

```
<Query QueryName="ChangeManagementOpenQuery">
```

```
<KeyField Name="Incident Ref"><LastValue>32357</LastValue></KeyField>
```

```
<KeyField Name="Date Logged"><LastValue>18/04/2005 16:56:26</LastValue></KeyField>
```

```
<LastFileName/>
```

```
</Query>
```

```
<Query QueryName="ChangeManagementPendingQuery">
```

```
<KeyField Name="Incident Ref"><LastValue>0</LastValue></KeyField>
```

```
<KeyField Name="Date
```

```
Resolved"><LastValue>1900-01-01</LastValue></KeyField><LastFileName/>
```

```
</Query>
```

send.txt

構築され、CA Business Service Insight に送信される準備ができていてすべてのイベントは、最初に send ファイルに書き込まれます。

SendControl.xml

sendcontrol.xml ファイルには、送信済みで CA によって確認された行がすべて含まれます。

このファイルは、アダプタがスライディング ウィンドウ確認プロトコルに従って CA ヘデータを転送することを許可します。このメカニズムの詳細については、「[アダプタ通信 \(P. 91\)](#)」を参照してください。

```
<SendControl  
PacketMaxSize="50"  
LastAckSequence="47"  
LastAckIndex="-1"/>
```

IllegalEvents 出力ファイル(.txt)

アダプタは、読み取り済みで解析の結果エラーがあったすべてのレコードを IllegalEvents 出力ファイルに書き込みます。これは、通常アダプタ設定ファイルに入力された検証ロジックによって行われます。設定ファイル内のパラメータ SaveIllegalEvents を「yes」に設定した場合、アダプタはこれらのレコードを保存します。このオプションのパスは「IllegalEventsDirectoryName」を使用して設定することに注意してください。フォルダは自動で作成されないため、既存のフォルダが必要です。フォルダが存在しない場合、アダプタは実行時にエラーを返します。

ファイルアダプタでは、ファイルにソース ファイルと同じ名前のエラー レコードが含まれ、SQL アダプタではクエリの名前になります。

Translations.xml

translation.xml ファイルにはアダプタ変換テーブルが格納されます。

アダプタがオンライン モードで実行されると、ファイルにはデータベースから変換テーブルのコピーが格納されます。変換テーブルがリモートに設定されている場合は、アダプタがデータベースからこのファイルに変換テーブルをロードして置き換えます。スタンドアロンの場合はローカル ファイルを読み取ります。

アダプタがオフライン モードで実行される場合、アダプタはファイルを変換テーブルとしてのみ使用します（オンライン/オフライン モードの詳細については、「[アダプタの実行モード](#) (P. 97)」を参照してください）。

```
<TranslationTableCollection>
```

```
<AssystResourceTable>
```

```
<Entry TranslationStatus="Ignored" resource="Authority"/>
```

```
<Entry TranslationStatus="Translated" resource="LONDON" TranslationTo="1006"/>
```

```
</AssystResourceTable>
```

```
<AssystSupplierEventTypeTable>
```

```
<Entry TranslationStatus="Translated" severity="1" TranslationTo="1545"/>
```

```
<Entry TranslationStatus="Translated" severity="2" TranslationTo="1550"/>
```

```
<Entry TranslationStatus="Translated" severity="3" TranslationTo="1551"/>
```

```
</AssystSupplierEventTypeTable></TranslationTableCollection>
```

アダプタ通信

以下の図に示すように、アダプタはデータ ソースと相互に動作する一方で、CA Business Service Insight アダプタ リスナおよびログ サーバとも動作します。



アダプタはデータ ソースと通信し、ODBC 接続を使用してデータを取得します。また、アダプタが ODBC 接続を確立できる間は、アダプタはデータ ソースに対してローカルまたはリモートのどちらにも位置することができます。

アダプタは、TCP/IP プロトコルを使用して CA Business Service Insight アプリケーション サーバと通信します。したがって、アダプタが TCP/IP プロトコル接続を確立できる間は、サーバに対してローカルまたはリモートに位置することができます。

アダプタは、アダプタ リスナ用およびログ サーバ用に 2 つのポートを開放する必要があります。アダプタ リスナポートはアダプタごとに固有である必要があり、このポートを使用している可能性があるほかのネットワーク処理またはアプリケーションと競合しないようにします。たとえば、普段ポート 1521 がデータベース通信用に Oracle TNS プロトコルによって使用されている場合、このポートを使用できません。また、このトラフィックをブロックする可能性があるローカルファイアウォールを考慮する必要があります。

注: 利用可能なポートがわからない場合や、この通信のためにポートの開放を要求する場合は、ローカル管理者に確認してください。

アダプタ リスナのポートおよびアドレスはアダプタ設定ファイルに設定されます。ログ サーバのポートおよび IP アドレスは、レジストリ内のアダプタのエントリを介して設定されます。

アダプタ リスナに関するクライアント/サーバ運用は設定することができ、アダプタをクライアントまたはサーバとして運用することができます。クライアント/サーバ運用の設定はアダプタ側で行い、設定ファイル内のパラメータで設定します。これを行うには、ポート、アドレス、および接続イニシエータ変数を適宜設定する必要があります。

接続イニシエータがアダプタに設定されている場合は宛先ポートのみが必要です。CA Business Service Insight に設定されている場合、CA Business Service Insight 上のアダプタ リスナのポートおよび IP アドレスが必要です。デフォルトでは、サーバはアダプタに設定されています。これはファイアウォール ルールのトリガを有効にする場合の重要な機能です（ポート トリガとして知られている機能です）。メッセージが同じポートのファイアウォールの「内部」から送信された場合、ファイアウォールはポートに対する内向きのリクエストのみ許可することがあります。その場合、ファイアウォールをトリガして通信を許可します。

注: アダプタ通信に影響する可能性があるローカル条件の詳細については、ネットワーク管理者にお問い合わせください。

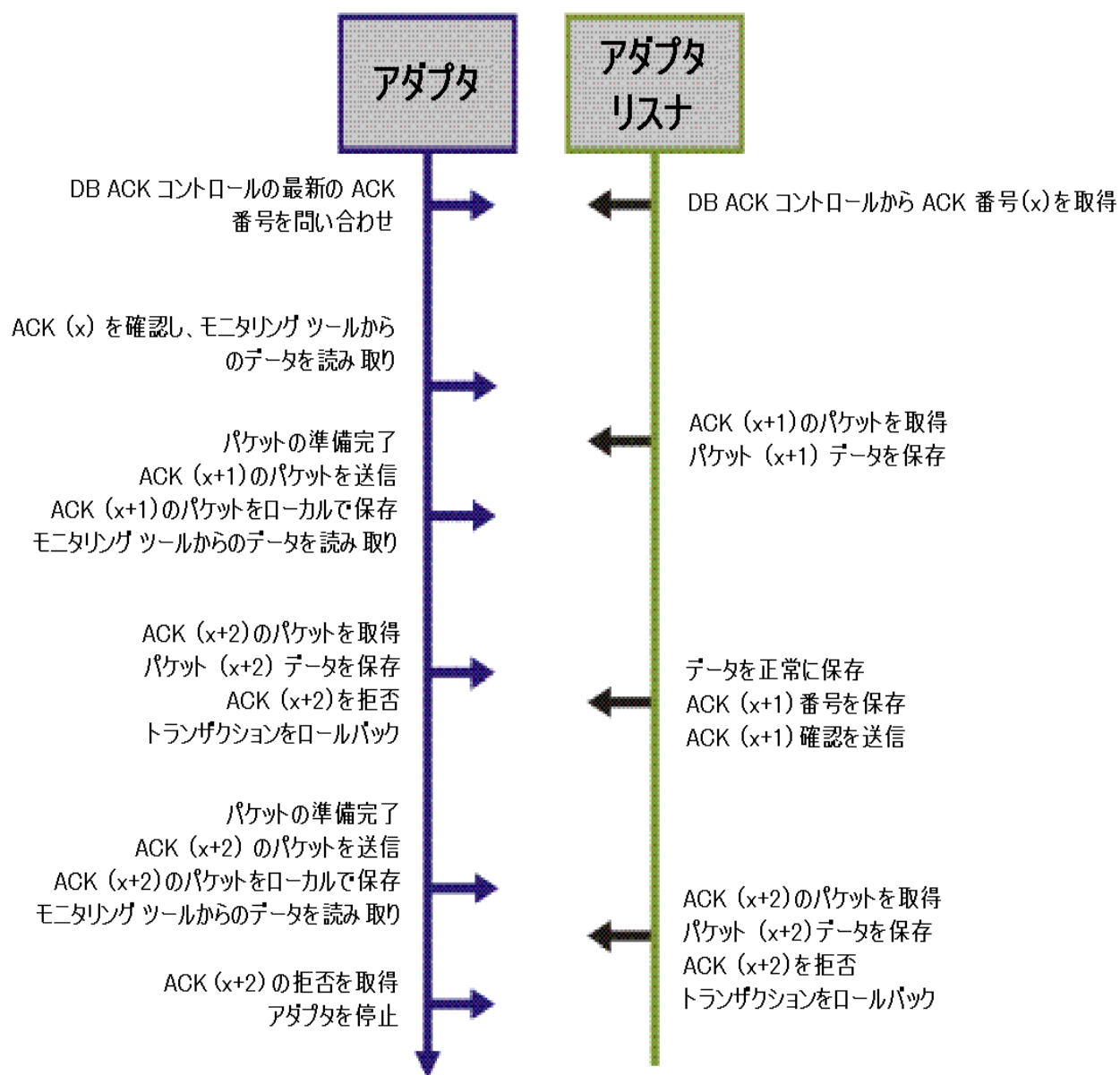
セキュリティの観点から、テストおよび実稼働が複数展開環境で動作する際のイベントの宛先を確認するため、アダプタはクライアントに設定することをお勧めします。

アダプタから **CA Business Service Insight** アダプタ リスナへのデータ レコード転送の成功を検証するには、アダプタに **TCP/IP** レイヤを利用した確認応答およびスライディング ウィンドウ アルゴリズムを組み込みます。このアルゴリズムは基本的にパケットでデータを送信し、次のパケットに移行する前にアダプタ リスナからの確認応答を待ちます。各パケットには **Raw** データ メッセージがいくつか含まれます。パケットのメッセージ数はパケット サイズ パラメータを設定することにより設定できます。パケットにはそれぞれ確認応答メッセージに含まれているシーケンスがあります。プロセスの制御に関連するパラメータはすべて設定ファイルの **CA Business Service Insight** インターフェース セクション内に格納されています。通常はこれらのパラメータを変更する必要はありません。

アダプタ リスナは、単一トランザクションのパケット内の **Raw** データを書き込みます。

注: 確認応答処理は、**CA Business Service Insight** に送信された **Raw** データ メッセージにのみ行うことができます。

以下の図にアダプタ通信プロセスを示します。



アダプタ レジストリ設定

コマンドラインから情報が失われている場合、アダプタは、アダプタがインストールされたサーバのシステム レジストリに格納された定義の一部を使用します。

アダプタ マネージャ ユーティリティがアダプタのインストールに使用された場合、レジストリ エントリはアダプタ マネージャ ユーティリティによって書き込まれています。それがアダプタのインストールに使用されなかった場合、これらのエントリは手動でレジストリに追加できます。

注: UNIX 環境内にアダプタをインストールする場合、この環境用のアダプタ マネージャがないので、これらのエントリは手動で追加する必要があります。

以下にリスト表示されているのは、アダプタおよびアダプタ マネージャ ユーティリティで使用されるレジストリ エントリです。

一般的なサーバ エントリ

以下のエントリは `¥HKEY_LOCAL_MACHINE¥SOFTWARE¥Oblicore¥Adapters` レジストリに書かれています。

利用可能なプロパティ:

名前	タイプ	説明
AdaptersDir	文字列	すべてのアダプタのルート ディレクトリ。*
FileAdapterConfTemplate	文字列	ファイル アダプタ設定テンプレートのパス。* アダプタ マネージャは、アダプタ作成プロセスの一部として指定された場合に、この情報を使用して設定テンプレートを新規アダプタのフォルダにコピーします。
GenericFileAdapter	文字列	ファイル アダプタの実行可能ファイル。* アダプタ マネージャは、アダプタ作成プロセスの一部として指定された場合に、実行可能ファイルへのショートカットを作成するか、実行可能ファイルを新規アダプタのフォルダにコピーします。
GenericSqlAdapter	文字列	SQL アダプタの実行可能ファイル。* アダプタ マネージャは、アダプタ作成プロセスの一部として指定された場合に、実行可能ファイルへのショートカットを作成するか、実行可能ファイルを新規アダプタのフォルダにコピーします。

名前	タイプ	説明
LogServerAddress	文字列	ログ サーバのネットワーク アドレス。(オプション) ** ログ サーバのポート - 通常は 4040 。(オプション) ** これらのパラメータが設定されると、アダプタは CA Business Service Insight ログ サーバにメッセージのログをレポートします。
LogServerPort	文字列	
SqlAdapterConfTemplate	文字列	SQL アダプタ設定テンプレートのパス。* アダプタ マネージャは、アダプタ作成プロセスの一部として指定された場合に、この情報を使用して設定テンプレートを新規アダプタのフォルダにコピーします。

* アダプタ マネージャ ユーティリティによってのみ使用されます。

* アダプタによって使用されます。

個別のアダプタ エントリ

以下のエントリは `¥HKEY_LOCAL_MACHINE¥SOFTWARE¥Oblicore¥Adapters¥<アダプタ名>` レジストリに書かれています。

利用可能なプロパティ：

名前	タイプ	説明
ConfigurationFileName	文字列	アダプタ設定ファイルの名前。 **
Directory	文字列	アダプタ ディレクトリ *
LogFileName	文字列	アダプタ ログ ファイルの名前。 **
Path	文字列	アダプタの実行可能パス *
RunAs	数字	実行モード。 * サービス/コンソール アプリケーション/Windows アプリケーション
Type	数字	アダプタ タイプ。 * File/SQL
LogFileMaxSize	数字	値はキロバイト数です。 ** 許可されている範囲は 1,000-100,000 で、デフォルト値は 1000 です。

- * アダプタ マネージャ ユーティリティによってのみ使用されます。
- * アダプタによって使用されます。

アダプタの実行モード

アダプタは以下のどちらかで実行できます。

サービス:

アダプタは標準の Windows サービスとしてインストールできます。これによって、標準の Windows サービスと同様にアダプタのステータス（実行、一時停止、停止、自動）を制御できます。

Windows サービスとしてアダプタをインストールするには、アダプタの実行可能ファイルをコマンドラインで実行します。**-i**はサービスとしてインストールし、**-u**はアンインストールします。

アプリケーション:

アダプタの実行可能ファイルをコマンドラインで実行します。アダプタ コマンドラインを実行する方法は以下のとおりです。

コマンドライン オプション:

```
TextFileAdapter.exe -i | -u | -v | -d [-t] [-f configurationFileName] [-l logFileName] [-n serviceName] [-a OblicoreAddress] [-p OblicorePort] [-la LogGerheaDed] [-lp LogServerPort]
```

パラメータ	機能
-i	サービスのインストール
-u	サービスの削除
-v	バージョンの表示
-d	コンソール アプリケーションとしてアダプタを実行
-t	チェックのみ - 設定ファイルを確認して停止
-f	設定ファイル名の設定
-l	ログ ファイル名の設定
-n	サービス名の設定
-a	アプリケーション サーバ アドレスの設定
-p	アプリケーション サーバのポート番号（1024-49151）の設定

パラメータ	機能
-la	ログ サーバ アドレスの設定
-lp	ログ サーバのポート番号 (1024-49151) の設定

このタイプの実行は、一般的にプロジェクトに使用されます。これは、.bat ファイルを介したアダプタの実行を許可し、また、アダプタ実行のタイミングを制御するために Windows スケジューラの使用を可能にします。Windows スケジューラを使用してアダプタをスケジュールするには、実行モードを Run Once に設定する必要があります。

RunOnce : (オプション [yes/no])。設定ファイルで「no」に設定した場合、一度実行したアダプタは連続して実行されます。「yes」に設定されたファイルアダプタが実行されると、レコードを読み取り、新規レコードがない場合は自動で停止します。ファイルアダプタはファイル全体を読み取り、次に数秒間待機した後、新規レコードの読み取りを試みます (SleepTime の設定によります) 。新規レコードがない場合、アダプタは停止します。SQL アダプタは、クエリをそれぞれ 1 回のみ実行します。RepeatUntilDry が「no」に設定されている場合はすぐに停止します。RepeatUntilDry が「yes」に設定されている場合は待機します (SleepTime の設定によります) 。アダプタは再度クエリの実行を試み (クエリのスリープ時間に応じて) 、新規レコードがない場合は停止します。

SleepTime 属性および RepeatUntilDry 属性の詳細については、「[Adapter Configuration Specifications](#) (P. 339)」を参照してください。

設定ファイルの CA Business Service Insight インターフェース セクションは、CA Business Service Insight への 2 つの接続モード (オンラインおよびオフライン) を指定する属性で構成されます。

オンラインモードでは、CA Business Service Insight にアダプタが接続して CA Business Service Insight から変換テーブルおよび制御コマンドを取得し、次にイベント、ステータスおよび変換リクエストを CA Business Service Insight に送信します。オフラインモードでは、アダプタはローカルの変換テーブル ファイルで動作し、出力ファイルにイベントを書き込みます。

通常、オフラインモードは最初のアダプタ開発およびテスト時に使用します。

「yes」に設定した ConsoleDebugMode を使用すると、デバッグ メッセージをコンソールに表示できます。

インジケータについての詳細については、「[Adapter Configuration Specifications](#) (P. 339)」の ConsoleDebugMode 属性を参照してください。

設定ツールおよびメンテナンス ツール

アダプタの設定およびメンテナンスのプロセスの一部として必要なツールは、主に簡単な Windows 実行可能ファイルである CA Business Service Insight ユーティリティなどのスタンドアロンユーティリティです。

注: UNIX 環境でアダプタを設定する場合はこれらのユーティリティを利用できないため、手動で設定を行う必要があります。

CA Business Service Insight アプリケーション サーバで作業する場合、ユーティリティはアプリケーション インストールの一部として Program Files\CA\Cloud Insight\Utilities フォルダにインストールされています。

インストールの一部として、ショートカットがスタート メニューに作成されています。ユーティリティの実行には、このショートカットを使用することをお勧めします。

CA Business Service Insight アプリケーション サーバで作業しない状況では、これらのユーティリティを任意の Windows コンピュータ標準ファイルとしてインストールするか、または提供されている CA Business Service Insight パッケージのカスタムインストールを選択してインストールすることができます。このインストール ルーチンは必須ではなく、通常は .exe ファイルをワークステーション上の利用しやすい場所に置くことで十分に対処できます。ただし、このオプションを使用すると一部の .dll ファイルが欠落する可能性があるため、可能であれば、このファイルをサーバからワークステーションのローカル フォルダへコピーする必要があります。

アダプタおよび変換の設定方法

以下の手順を行います。

1. アダプタ ウィザードを使用してアダプタを設定、または手動で XML 設定ファイルを編集します（以降の章で説明）。
2. アダプタを展開します。
3. アダプタをテストします。
4. 変換を実行します。
5. 変換スクリプトを記述して自動変換プロセスをサポートします(オプション)。

注: このタスク処理のため、アプリケーション サービス上で背景サービスとして実行される新規アダプタ展開サービスとしてアダプタ ウィザードを使用する場合は、アダプタの展開が自動で行われます。

新規アダプタの展開(アダプタ ウィザード)

アダプタ ウィザードを使用して新規アダプタを作成する場合、「アダプタ リスト」および「アダプタ展開」サービスが実行されている必要があります。

新規アダプタの展開(手動)

新規アダプタを作成するための前提条件

新規アダプタの作成を始める前に、以下の条件をすべて整えておく必要があります。

- **アダプタのルート フォルダ**：サーバに CA Business Service Insight がインストールされている場合、このフォルダは Program Files¥CA¥Cloud Insight root folder に存在します。存在しない場合は作成する必要があります。
- **個別のアダプタ フォルダ**：アダプタルート フォルダの下に、特定のアダプタ用のフォルダを作成します。

注：ユーザがアダプタ マネージャ ユーティリティを使用している場合、新規アダプタを追加する際にユーティリティによって自動でフォルダが作成されます。

- **アダプタの実行可能ファイル**：TextFileAdapter.exe（テキスト ファイル アダプタ用のテキスト ファイル アダプタ実行可能ファイル）および SQLAdapter.exe（SQL アダプタ用の SQL ファイル アダプタ実行可能ファイル）。これらは、アプリケーション サーバの Program Files¥CA¥Cloud Insight¥Adapters フォルダ下にあります。

注：アダプタ マネージャ ユーティリティを介したアダプタ作成プロセス中に、できる限りコピーの作成ではなく、実行可能ファイルのショートカットを作成するオプションを選択します。これを行うと、アップグレードまたはサービス リリース（SR）が CA Business Service Insight に適用された場合に、確実にすべてのバイナリ コンポーネントが正しくアップグレードされます。

- **設定テンプレート**：アダプタ設定ファイルのテンプレート。アダプタのルート フォルダ内にファイルを置きます。これらは、アプリケーション サーバの Program Files¥CA¥Cloud Insight¥Adapters フォルダ下にあります。設定テンプレートは設定ファイルの作成に使用され、新規作成することを回避できます。また、この目的で既存のアダプタ設定ファイルを使用することも一般的です。
- **アダプタ マネージャ ユーティリティ**：スタンドアロンの実行可能ファイル。アプリケーションサーバの Program Files¥CA¥Cloud Insight¥Utilities フォルダ下のユーティリティ フォルダから AdaptersManager.exe のコピーを作成すれば十分です。このユーティリティを使用してアダプタを作成することは必須ではありません。このユーティリティは Windows サーバ上でのみ使用できます。
- **2つの TCP/IP 空きポート**：1つはアプリケーションサーバ上のアダプタリスナ用、もう1つはログサーバ用。ログサーバのポートは通常 4040 です。
- 実行されている CA Business Service Insight アプリケーション サービス コンポーネントを確認します。アダプタを実行するために、以下のサービス コンポーネントが実行されている必要があります。

– AdapterListener

- TaskHost
- LogServer

以下の手順に従います。

1. アダプタ マネージャ ユーティリティを実行します（上記のアダプタ マネージャ ユーティリティのセクションを参照してください）。
2. 上記の前提条件をすべて準備し、実行可能コマンドラインでバッチ ファイルを作成します（「[アダプタの実行モード](#) (P. 97)」を参照してください）。

アダプタ設定ファイルの変更

アダプタを作成する際の作業の大半は設定ファイルの編集です。

この作業では XML ファイル内の属性を設定して、アダプタが必要な動作をするように制御します。設定ファイルは XML ファイルで、内部ワーク フローの手順に対応している各セクションがあります。

セクションは以下のとおりです。

- **General セクション**：多岐にわたるアダプタの属性（作業ディレクトリ、出力ファイル、デバッグ フラグなど）。

Oblicore:

- **OblicoreInterface セクション**：CA Business Service Insight サーバとの接続の属性（TCP/IP ポート、セキュリティ モードなど）。
- **DatasourceInterface セクション**：データ ソースとの接続の属性（ファイルパスおよびパターン、接続文字列、SQL クエリなど）。
- **InputFormatCollection セクション**：オリジナルデータ形式を分析し、操作するための解析ルール（区切り文字、フィールド タイプ、データ順序、正規表現など）。
- **TranslatorCollection セクション**：解析および操作されたデータ フィールドで構成された CA Business Service Insight 統合イベントのルール。
- **TranslationTableCollection セクション**：オリジナルデータ用語と CA Business Service Insight データ エンティティ間のデータ マッピングのルール。

これらのセクションについては、「[Adapter Configuration Specifications](#) (P. 339)」で詳しく説明しています。

注: 各セクションの XML ノードの順序は重要ではありません。

ファイル アダプタ

ファイル アダプタは、FileAdapter 汎用コンポーネントの総括的なコンポーネント（実行可能ファイル）および ASCII ファイルを解析する設定ファイルを使用します。

ファイル アダプタ ワークフロー：

- ソース ファイルをワークファイルにコピー/名前変更します。
- 論理レコードを読み取ります。
- 区切り文字または正規表現に従ってレコードを解析します。
- 正しい inputFormat を検索します。
- イベント レコードを作成します。
- イベント レコードを変換します。
- 制御ファイルを更新します。

設定ファイルの例

以下の例では、単純な ASCII ファイル データ ソースをイベント出力要件と共に使用し、そのアダプタ設定ファイルに必要な設定を確認します。

設定ファイルは XMLPad ユーティリティを使用して表示、編集できます。

設定ファイルの構造およびコンテンツの概要については、関連するセクションを参照してください。

注： 確認された設定は主要で最も重要な属性設定のみです。完全な属性仕様については、[アダプタ設定仕様](#) (P. 339)を参照してください。

ファイル アダプタのケース スタディ

以下に示す構造に基づいた情報を持つログ ファイルを生成するサーバ モニタリング システムについて考察します。

サーバ	タイムスタンプ	メモリ 使用率	CPU 使用率
Server001 p	1.11.2004 0:00	26.71	78.85
Server002 p	1.11.2004 1:00	49.09	89.49
Server001 p	1.11.2004 2:00	51.37	57.51
Server001 p	1.11.2004 3:00	23.53	98.9
Server002 p	1.11.2004 4:00	86.21	85.11
Server002 p	1.11.2004 5:00	39.98	35.97
Server001 p	1.11.2004 6:00	70.58	55.96

ファイルは CA Business Service Insight アダプタ フォルダの「データ」という名前のサブ フォルダに配布されます。すべてのファイル名の先頭には「ServerData」が付き、後ろに日時が付いています。また、ファイルは .csv の拡張子が付いた CSV ファイルです。

以下の出力イベントが必要です。

- リソース変換フィールド： Server
- タイムスタンプ フィールド： Timestamp
- データ フィールド： Memory Utilization、CPU Usage

また、データ ソースはベルギーにあるものと仮定します（+1 タイムゾーン オフセットのある CET タイムゾーンで、夏季には夏時間期間の 1 時間が追加されます）。

設定ファイルの General セクション

- **MajorVersion And MinorVersion** : デフォルトではアプリケーションバージョンに設定する必要があります。
- **WorkingDirectoryName (オプション)** : アダプタ出力ファイル (データソース制御、変換、送信) のデフォルトパスを指定します。このケースでは「output」と設定したため、アダプタのメインフォルダ下にこのフォルダが作成されます。

以下のインジケータは、アダプタがレコードを読み取り、変換する方法を制御します。

- **RunOnce** : 「yes」に設定するとアダプタが1回実行され、利用可能なデータをすべて読み取ってから停止します。
- **DataSourceDifferenceFromUTC** : UTC と (協定世界時、常にオフセット時間がゼロである。GMT に同じ) データソースのタイムゾーンとの差を示します。データソースのタイムゾーンは、時間フィールドが指定されているタイムゾーンです。この属性は、アダプタがすべての日付を UTC に標準化するために必要です。すべての日付を UTC にするとアプリケーションに柔軟性を持たせることができ、ユーザ要件に応じて時間を表示することができます。以下の属性は、入力から UTC へ時間フィールドを変換する方法の詳細をアダプタに提供します。
 - **DefaultOffset** : 夏時間でない場合の UTC からのオフセット。このケースでは、中央ヨーロッパ時間 (CET) であるため「1」を設定します。
 - **TimeFormat** : 夏時間の詳細 (次に詳述) を解析する形式。CA Business Service Insight 形式の仕様では「%d/%m/%Y %H:%M:%S」と設定しますが、ヨーロッパの時間形式は「dd/mm/yyyy hh:mi:ss」です。
 - **DayLightSaving** : データソースタイムゾーンの1期間の夏時間期間。このエレメントはオプションで (選択しない場合は夏時間がないという意味です)、1回以上存在することができます (夏時間期間の入力ごとに1回)。いくつかのエレメントが存在する場合は、必ず時間を並べ替え基準にし、各期間はオーバーラップしないようにします。通常は、アダプタを設定する際にこのエレメントを数年先まで指定します。このケースでは例として1年のみ設定します。
- **From** : 期間の開始日。上記の時間形式セット「25/03/2007 02:00:00」を使用して指定します。
- **To** : 期間の終了日付。上記の時間形式セット「28/10/2007 03:00:00」を使用して指定します。
- **Shift** : 夏時間期間内の DefaultOffset に追加された時間シフト。指定した2つの日付の間の夏時間期間中は時間が1時間進むため、「1」と入力します。

設定ファイルの OblicoreInterface セクション

このセクションでは、CA Business Service Insight サーバとの接続を定義します。

- **Mode** - オンライン モードでは、CA Business Service Insight にアダプタが接続して CA Business Service Insight から変換テーブルおよび制御コマンドを取得し、次にイベント、ステータス、および変換リクエストを CA Business Service Insight に送信します。このモードでアダプタを設定し、「online」に値を設定します。
- **Port** - アダプタが CA Business Service Insight サーバ（アダプタ リスナが存在する場所）との通信に使用する TCP/IP ポート番号です。

この設定に問題がないと仮定し、アダプタをポート「5555」（任意に選択）を使用するように設定します。また、通信を有効にするため、サーバ上のアダプタの GUI でもこれを指定する必要があります。

設定ファイルの DataSourceInterface セクション

DataSourceInterface セクションは、接続を指定する属性およびアダプタとデータソース間の接続タイプを指定する属性で構成されます。 インターフェースにはファイルと SQL の 2 種類があります。 2 つの主な違いは、ファイルにはファイル収集が必要であり、SQL にはクエリ収集が必要なことです。

DataSourceInterface セクションでは、アダプタがソース ファイルを管理する方法（オリジナル ファイルがアダプタ用にのみ作成されている場合にファイルを削除するかどうか、またほかの目的で使用するためにデータを保持するかどうかなど）も定義します。

ファイルアダプタでは、ASCII ファイルの読み取りおよび解析を行うため、ファイルインターフェースを以下の図のように使用します。以下のように値を選択して設定します。

DataSource Interface ノード下のファイル セクションはデータ ソースとの接続に関連します。 以下の属性を設定します。

注: このセクションは SQL アダプタでは異なります。

- **DeleteFileAfterProcessing**: アダプタがソース ファイルを処理する方法の設定、および新規レコードのみを読み取るために、読み取りを制御する方法を決定します。 このケースでは、ソース ファイルをサーバに保持し、値を「no」に設定します。

ファイルがアダプタ用にのみ作成されており、処理後に削除することができる場合は、「yes」に設定します。 これを行うとファイルの名前が変更され、処理後に削除されます。

「no」に設定した場合、ファイルがコピーされ、コピーされたファイルで処理が行われます。 このファイルに新規レコードが追加された場合、アダプタは次のサイクル中にこれらの新規レコードをワーク ファイルにコピーします。 このファイルに新規レコードが追加されていない場合、アダプタは、現在のファイルと同じパターンおよび名前（辞書式順序）を持つ次のファイルを検索します。 アダプタはそのようなファイルを検出した場合、そのファイルで作業を進めます。 新規レコードが追加されても、アダプタは前のファイルに戻りません。

ファイルを追加する際にソース ファイルの完全性を維持する必要がある場合は「no」を選択します。

- **InputFormat**: InputFormat エlement に与えられた名前を、このデータソースの接続から得たデータを処理する次の InputFormatCollection で参照します。 入力フォーマットは、データ ソースから来るアダプタによる解析済みの入力データのフィールド構造です。 区切り文字属性に指定されている解析メソッドについては、以下で説明します。 異なるインターフェースの形式を使用して 1 つ以上の接続を処理する場合、このフィールドはより重要なものになり、各データ ソースのデータを処理する入力フォーマットの構造を決定します。

- **Path** : データ ソース ファイルの物理的な場所。例 : C:\Program Files\CA\Cloud Insight\Adapters\ServersAdapter\data
- **NamePattern** : データ ソースのファイル名を指定します。1 つ以上のファイルが同じ入力フォーマットを使用する場合は、ワイルドカードを使用することもできます。ワイルドカードを使用せずにファイル名を指定すると、アダプタはこの名前のファイルのみを検索します。ワイルドカードを使用すると、アダプタはパターンに対応するファイルをすべて検索し、辞書式順序で並べ替えてから順番に読み取ります。次回の実行では、次に進む前に前回のファイルの新規レコードを検索します。

この例ではワイルドカード文字「*」を使用し、属性値は「ServerData*.csv」となります。(アダプタは、ServerData から名前が始まり、拡張子が.csv であるすべてのファイルを読み取ります。)

重要 : 確実にファイルを並べ替えて正しい順序で 1 つも飛ばさずに処理するために、日付および時間をファイル名の最後に YYYYMMDD-HHMISS の形式を追加することをお勧めします。曜日ごとに作成されている複数のファイルがある場合にも、時間の部分を追加できます。

- **Delimiters** : ファイルの解析方法を決定するメソッド。フィールドに解析されるデータ行にしたがって、1 つ以上の文字を区切り文字として設定することができます。指定しない場合のデフォルト値は「/t」です。

この例のデータ ソース ファイルは CSV (カンマ区切り) ファイルです。このようなファイルを解析する最も簡単な方法は、区切り文字としてカンマを指定することです。

解析に利用可能なその他のメソッドは以下のとおりです。

- **RegexForParser** : 解析ルールの設定に正規表現を使用します。
- **LogicLineDefinition** : ファイル内の行が複数行で構成されている場合に使用します。
- **TitleExists** (オプション) : データ ソース ファイルの最初の空白以外の行がタイトル行であるかどうかを指定します。タイトルは、データの解析時にアダプタが使用することができます。この例では各データ ファイルにタイトル行が含まれているため、この属性は「yes」に指定する必要があります。

設定ファイルの InputFormatCollection セクション

このセクションでは、データ行をフィールドに分割する方法およびフィールドタイプや形式など、データソースから取得したデータの構造を指定します。初期データフィルタリングおよびデータ操作は、InputFormatSwitch フィールドおよびCompound フィールドをそれぞれ使用して、このセクションで実行される場合があります。

このセクションでは、InputFormatValidation および ValidationCase を使用してレコードが正しいかどうかを判断する入力レコードの検証メトリックを定義することができます。

InputFormatCollection ノードには 1 つ以上の InputFormat ノードが含まれていることがあります。

このセクションの、アダプタによる一般的なワークフローは以下のとおりです。

- データ行は DataSourceInterface に指定された InputFormat と照合されます。
- データは InputFormat の仕様と一致するフィールドに分割されます。InputFormatFields の順序はデータソースで解析されたフィールドの順序と一致する必要があります。
- Compound フィールドは、事前に指定したデータフィールドを結合または分割することで割り当てられた値です。Compound フィールドの定義は通常のフィールド定義の後ろにする必要があります。
- 処理されたデータは、トランスレータスイッチ条件と照合され、入力レコードから出力イベントを構築するトランスレータが決定されます。
- 処理されたデータは、一致したトランスレータに送信されるか、または無視されます。

以下のパラメータを設定します。

- **InputFormatName**: この形式の名前は DataSourceInterface セクションによって参照されます。
- **InputFormatFields**: InputFormatFields には、入力フィールドのデータソース数により 1 つ以上のフィールドノードが含まれる場合があります。
- **InputFormatField**: オリジナルのデータ行のデータフィールドまたは Compound フィールドを 1 つ指定します。

```
<InputFormatField Name="timestamp" Type="time"
```

```
TimeFormat="%d/%m/%Y %H:%M:%S"/>
```

- **名前**: このフィールドの名前はソースフィールドとして、Compound エlement または TranslatorFields などのほかの Element によって参照されます。

- **Type** : フィールドのデータ タイプ - string/integer/real/time。
- **Source** : このフィールドのソース値で、デフォルト値は「event」です。
設定できる値は以下のとおりです。
 - **event** : フィールド値はデータ ソースから来るイベントから得られます。フィールドの値はデータ ソースから来るイベントの順番で取得されます。
 - **compound** : フィールド値は、ほかのフィールドの値または定数の操作を基準にして、ほかのフィールドで構築されます。操作されたフィールドは、定義済みである必要があります。
 - **title** : フィールド値はタイトル フィールドの名前から取得されます。参照されたフィールドは、定義済みである必要があります。
 - **filename** : フィールド値はデータ ソースのファイル名から取得されます。テキスト ファイルアダプタ専用です。
 - **constant** : フィールド値は ConstantValue から取得され、隣にプロパティが表示されます。
- **TranslatorSwitch** : データ行を統合イベントに変換するために使用するトランスレータを決定します。
- **DefaultTranslator** : 基準なしに一致する場合に使用するトランスレータ。値を「Ignore」に設定する場合はトランスレータは使用されず、その行は無視されて CA Business Service Insight に送信されません。

設定ファイルの TranslatorCollection セクション

TranslatorCollection セクションは、前のセクションで抽出、解析、操作されたデータソースレコードを CA Business Service Insight イベントに変換する方法を定義します。

このセクションでは、重複するイベントを処理する方法およびイベント固有性メカニズムを使用する方法も定義します(詳細については「[イベント固有性 \(P. 141\)](#)」を参照してください)。

インターフェース モードをオンラインに設定する場合、CA Business Service Insight イベントは以下のフィールドを含む統合構造を持ちます。

- **Timestamp** : イベントの発生時間。
- **ResourceId** : イベントに関連付けられたリソース ID (そのイベントで測定されたリソース)。
- **EventTypeId** : イベントと関連付けられたイベントタイプであり、イベントタイプの説明(リソースの測定値のタイプ、チケットアクションのタイプなど)。
- **DataSourceId** (オプション) : 任意の値です。Raw データ イベントに追加するフィルタ基準です。
- **Value** (複数可) : イベントの値(測定結果、チケット番号など)。多くの場合、このフィールドは1回以上出現します。

トランスレータの構造は、CA Business Service Insight のイベントタイプ、および以下の図に示すようにイベントを格納するデータベース テーブル T_RAW_DATA に対応しています。

The screenshot displays the XML configuration for the TranslatorCollection. The left pane shows the 'Structure' tree with the following hierarchy:

- Translator
 - TranslatorName
 - TranslatorFields
 - TranslatorField
 - Name
 - SourceType
 - SourceName
 - TranslatorField
 - Name
 - SourceType
 - SourceName
 - TranslatorField
 - Name
 - SourceType
 - SourceName
 - TranslatorField
 - Name
 - SourceType
 - SourceName
 - TranslatorField
 - Name
 - SourceType
 - SourceName

The right pane shows the 'Values' section with the following fields:

- RMTranslator
- ResourceId
- table
- ResourceTable
- EventTypeId
- table
- EventTypeTable
- DataSourceId
- Field
- FILENAME
- Timestamp
- Field
- RaisedDateTime
- Value
- Field
- RAISEDATE

Arrows point from the 'ResourceId', 'EventTypeId', 'DataSourceId', and 'Value' fields in the XML to the corresponding columns in the T_RAW_DATA table schema shown on the right.

Name	Type	Nullable
RAW_DATA_ID	NUMBER	
READER_ID	NUMBER	Y
EVENT_TYPE_ID	NUMBER	
RESOURCE_ID	NUMBER	
TIME_STAMP	DATE	
READER_TIME_STAMP	DATE	Y
CREATED_BY	NUMBER	
CORRECTED_BY	NUMBER	
DATA	VARCHAR2(1000)	
CREATE_DATE	DATE	
MODIFY_DATE	DATE	
DATA_SOURCE_ID	VARCHAR2(60)	Y
HASH_DATA_KEY	VARCHAR2(116)	Y

- **Translator** : 取得するフィールドセットを出力イベントに変換する方法を説明します。
- **TranslatorName** : このトランスレータへフィールドセットを送信するために InputFormat によって使用される名前。この例では「Translator1」を使用します。このシナリオで使用できる値については、上の図を参照してください。
- **TranslatorFields** : TranslatorField エレメントのリストが含まれており、それぞれが以下の属性を持ちます。
 - **Name** : フィールド名。オンライン インターフェースでは、Timestamp、ResourceId、EventTypeId、ResourceId または Value にする必要があります。
 - **SourceType** : フィールド値のソースを指定します。以下のいずれかになります。
 - **Field** : このフィールドの値は入力フォーマットのフィールドから取得されます。SourceName 属性には InputFormat フィールド名が格納されます。
 - **Table** : フィールドの値が変換テーブルから取得されます。SourceName 属性には、次の TranslationTableCollection セクションに定義されている名前を参照する変換テーブルの名前が格納されます。このタイプは、イベントのリソースおよびイベント タイプに変換する値に使用します。
 - **Lookup** : このフィールドの値は変換テーブルから取得されます。SourceName 属性にはテーブル名が含まれます。入力フォーマットではなく LookupValue 属性から変換される値です。通常、変換に定数が必要な場合に使用します。
 - **Constant** : フィールドの値は定数で、値は ConstantValue 属性にあります。定数を使用する場合は、Type 属性を使用してフィールド タイプを指定する必要があります。
 - **SourceName** : フィールド名または変換テーブル名が格納されます。
 - **LookupValue** : SourceType="lookup" を設定している場合に検索値を格納します。
 - **ConstantValue** : SourceType=constant を設定している場合に定数値を格納します。フィールドタイプが時間であるとき、定数値は TimeFormat (アダプタの General セクションで設定) に従ってフォーマットされた文字列、または「Now」か「NowUtc」になります。「Now」はデータ ソース ロケールの現在の時刻、「NowUtc」は UTC の現在の時刻です。

このセクションには、データ ソース値の CA Business Service Insight イベント フィールドへのマッピングを定義するマッピング テーブルも含まれており、変換する参照データ ソース値と共にテーブルの定義が保持されます。

- **LoadingMode** : オンライン インターフェースのデフォルトは「remote」、オフライン インターフェースは「standalone」です。

変換テーブルのローディング メソッドは以下のとおりです。

- **Standalone** : アダプタは変換テーブルをローカルにロードします。 変換に関しては CA Business Service Insight サーバと接続しません。 変換テーブルの変更はローカル ファイル内にのみ保持されます。
- **Remote** : アダプタは CA Business Service Insight サーバからすべてのテーブルをロードするリクエストを送信します。 変換テーブルの変更は同じくローカルに格納されます。
- **TranslationTable** : イベント値をマッピング テーブルにリンクします。
 - **Name** : トランスレータによって使用され参照される変換テーブルの名前。
 - **DestinationType** : [resource、event_type、contract_party、service、time_zone、value]。 変換テーブルのタイプを保持します。 この例では、データ ソース ファイルの **Servers** 列が CA Business Service Insight リソースに変換されます。 そのため、変換テーブル タイプは Resource になり、CA Business Service Insight のリソース ID へ変換値を保持します。
 - **TranslationField** : 入力フォーマット フィールドに取得される変換元前のフィールド名。 最大で 5 つのフィールドを格納することができます。

設定ファイルに定義された変換テーブルは、それぞれ CA Business Service Insight ユーザ インターフェース内に対応する定義が必要です。

サンプル設定ファイルの XML 表現は以下のとおりです。

```
<?xml version="1.0" encoding="utf-8"?>
<AdapterConfiguration>
  <General MajorVersion="4" MinorVersion="0" RunOnce="yes" LogDebugMode="yes"
ConsoleDebugMode="yes" WorkingDirectoryName="output"
RejectedEventsUpperLimit="10000">
    <DataSourceDifferenceFromUTC DefaultOffset="0" TimeFormat="%d/%m/%Y %H:%M">
      <DaylightSaving From="20/04/2001 00:00" To="15/10/2001 00:00" Shift="1"/>
    </DataSourceDifferenceFromUTC>
  </General>
  <OblicoreInterface Mode="online">
    <OnlineInterface Port="5555" SecurityLevel="none"/>
  </OblicoreInterface>
  <DataSourceInterface>
    <Files>
      <File DeleteFileAfterProcessing="no" InputFormat="InputFormat1"
NamePattern="servers*.csv" Path=" C:¥Program
Files¥Oblicore¥Adapters¥ServersAdapter¥data¥" TitleExists="yes" SleepTime="60"
Delimiters=", "/>
    </Files>
  </DataSourceInterface>
  <InputFormatCollection>
    <InputFormat InputFormatName="InputFormat1">
      <InputFormatFields>
        <InputFormatField Name="resource" Type="string"/>
      </InputFormatFields>
    </InputFormat>
  </InputFormatCollection>
</AdapterConfiguration>
```

```
<InputFormatField Name="timestamp" Type="time"
TimeFormat="%d.%m.%Y %H:%M"/>
    <InputFormatField Name="memory_util" Type="real"/>
    <InputFormatField Name="cpu_util" Type="real"/>
</InputFormatFields>
<TranslatorSwitch DefaultTranslator="Translator1"/>
</InputFormat>
</InputFormatCollection>
<TranslatorCollection>
<Translator TranslatorName="Translator">
    <TranslatorFields>
        <TranslatorField Name="ResourceId" SourceType="table"
SourceName="ResourceTable"/>
        <TranslatorField Name="EventTypeId" SourceType="lookup"
SourceName="EventTable" LookupValue="PerformanceEvent"/>
        <TranslatorField Name="Timestamp" SourceType="field"
SourceName="timestamp"/>
        <TranslatorField Name="Value" SourceType="field"
SourceName="memory_util"/>
        <TranslatorField Name="Value" SourceType="field"
SourceName="cpu_util"/>
    </TranslatorFields>
</Translator>
</TranslatorCollection>
<TranslationTableCollection LoadingMode="remote">
    <TranslationTable Name="ResourceTable" DestinationType="resource">
        <TranslationField>resource</TranslationField>
    </TranslationTable>
    <TranslationTable Name="EventTable" DestinationType="event_type">
        <TranslationField>resource</TranslationField>
    </TranslationTable>
</TranslationTableCollection>
</AdapterConfiguration>
```

SQL アダプタ

SQL アダプタは、基本的に SQL アダプタの標準コンポーネント（SQL アダプタ実行可能ファイル）を設定ファイルの適切な設定で使⽤します。SQL アダプタは、ODBC および OLEDB をサポートするすべてのデータ ソースに接続できます。接続は接続文字列を介して確⽴されます。アダプタがインストールされるサーバに適⽫なドライバをインストールする必要があります。

データ ソース例は以下のとおりです。

- Oracle
- SQL Server
- Access
- Excel
- テキスト ファイル、CSV ファイル（これらも TEXT アダプタを介して接続することができる場合がありますが、ODBC 接続は追加のフィルタリング/クエリ機能を提供します）。

SQL アダプタ ワークフローは以下のとおりです。

- 接続を開始します。
- ローカル変数を最後のキーフィールド値に置換します。
- クエリの WHERE 句をオートコンプリートを利用して作成し、クエリの最後に連結します。

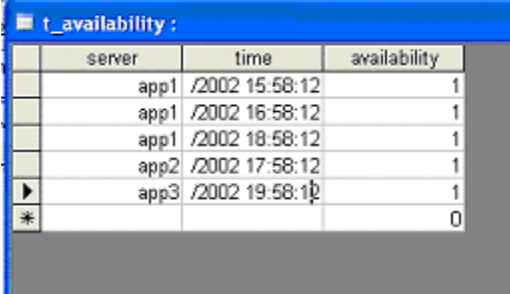
クエリを実行し、recordset を取得します。

クエリによって返された recordset 中の各レコードについて以下を実行します。

- 正しい InputFormat を検索します。
- イベント レコードを作成します。
- レコードを変換します。
- 制御ファイル内のキーフィールド値を更新します。
- 接続を終了します。
- スリープに移動します。

SQL アダプタの設定ファイル例

以下のテーブルと MS Access (.mdb) データベースが与えられています。



	server	time	availability
	app1	/2002 15:58:12	1
	app1	/2002 16:58:12	1
	app1	/2002 18:58:12	1
	app2	/2002 17:58:12	1
	app3	/2002 19:58:12	1
*			0

SQL アダプタの設定ファイルとファイルアダプタの設定ファイルの違いは、**DatasourceInterface** セクションのみです。

ファイルアダプタの **DatasourceInterface** セクションはファイルコレクションを格納しますが、SQL アダプタ ファイルには **ConnectionString** と **QueryCollection** があります。

2 つの設定ファイルの主な違いは、取得と解析メソッドにあります。その他は同じです。

SQL インターフェースは、データベースへの接続、およびデータの取得に使用されるクエリを定義します。

詳細については以下のとおりです。

注: このセクションでは、上記のデータ ソース データベースに基づいて定義します。

接続文字列エレメント

ConnectionString : ソース データベースにアクセスするための接続文字列。

ConnectionString は **DataSourceInterface** エレメントおよび (または) **Query** エレメントに定義できます。**DataSourceInterface** エレメントの **ConnectionString** 定義はデフォルトであり、**ConnectionString** 定義がないクエリにのみ使用されます。これにより、各クエリが独自の接続文字列を持つことができ、アダプタは複数のデータベースに接続することができます。接続文字列のメカニズムの詳細については、以下のセクションを参照してください。

設定ファイルの QueryCollection セクション

Query : クエリ属性を指定します。

- **QueryName** : クエリ名。
- **InputFormat** : このクエリに関連付けられている **InputFormat**。アダプタは **InputFormat** を使用してソースからデータを抽出します。

注: **InputFormat** フィールドの順序は、クエリで選択されたフィールドの順序に対応する必要があります。
- **SleepTime** : アダプタが新規データの到着を待つ間アイドル状態にいる時間 (秒)。
- **SelectStatement** : ソース データベース上で実行される、選択されたステートメントを格納します。

注: ステートメントで最初に選択された値としてクエリ キー フィールドに入力する必要があります。

 - **AutoCompleteQuery** : 「yes」に設定すると、アダプタが以下のように自動で **WHERE** ステートメントを指定したクエリに連結します。
 - キー フィールドを基準にして新しい値のみを取得する **WHERE** ステートメントを作成。
 - キー フィールドを基準にしてステートメントの結果を並べる。
- **QueryKeyFields** : 次のデータ取得を開始するためにフィールドを定義します。
 - **KeyField** :
 - **Name** : クエリのフィールドから取得されるフィールドの名前。
 - **Sort** : データの並べ替え順 (昇順/降順)。

- **ValueLeftWrapper** : フィールド値の先頭に文字を連結します。デフォルトは' (アポストロフィ) です。
- **ValueRightWrapper** : フィールド値の後に文字を連結します。デフォルトは' (アポストロフィ) です。
- **Critical** : 特定のクエリが失敗した場合に、ほかのクエリを停止します。
- **SleepTime** : 必須操作間のスリープ時間。デフォルトは' (アポストロフィ) です。

注: 「Date」フィールドには、日付自体を囲むために特殊文字を使用します。データソースによっては、フィールドが日付フィールドであることを認識するためにこの文字が必要になります。このセクションの最初に図に示したように、# 文字は Excel の値フィールドのラップに使用できます。ただし、ほかのデータソースには別の方法が必要になります。詳細については、「[Adapter Configuration Specifications](#) (P. 339)」を参照してください。

- **SelectInitialValues** : 最初の WHERE ステートメントのキーフィールドに初期値を提供する SELECT ステートメント (コントロールファイルが空の場合)。

AutoCompleteQuery="yes" のときに構築されるクエリ (Excel ベースの ODBC) の例は以下のとおりです。

```
SELECT INC_CLOSE_DATE, INCIDENT_REF, Severity, `Resolve Time`, `Date Logged`,
`Date Resolved` FROM `AllCalls$` WHERE INC_CLOSE_DATE>CDate('7/03/2005
13:06:21') order by INC_CLOSE_DATE
```

この SELECT ステートメントは、クエリが実行されているターゲット DB 上で実行可能である必要があります。これは、ソースと接続に使用する ODBC ドライバ間で異なる可能性があります。たとえば、Oracle では特殊な「dual」テーブルから値を選択 (「aaa」および「1-jan-1970」を dual から選択する) することができますが、Excel ではテーブルを使用せずに直接値を選択します。(「aaa」を選択)

以下は XML レイアウトの完全な設定ファイルです。

```
<?xml version="1.0" encoding="utf-8"?>
<AdapterConfiguration>
  <General MajorVersion="3" MinorVersion="0" RunOnce="yes" LogDebugMode="yes"
ConsoleDebugMode="yes" WorkingDirectoryName="d:\Oblicore\Training
Kit\Exercises\Adapters\SQL Adapters\Ex1\Solution">
    <DataSourceDifferenceFromUTC DefaultOffset="1"
TimeFormat="%Y/%m/%d-%H:%M:%S"/>
  </General>
  <OblicoreInterface Mode="online">
    <OnlineInterface Port="2000" SecurityLevel="none"/>
  </OblicoreInterface>
  <DataSourceInterface>
    <ConnectionString>
```

```
Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\¥Oblicore¥Training
Kit¥Exercises¥Adapters¥SQL Adapters¥Ex1¥db1.mdb;

</ConnectionString>
<QueryCollection>
  <Query QueryName="Query" InputFormat="InputFormat" SleepTime="5">
    <SelectStatement AutoCompleteQuery="yes">

      select time,server,availability from t_availability

    </SelectStatement>
    <QueryKeyFields>
      <KeyField Name="time" Sort="asc" ValueLeftWrapper="#"
ValueRightWrapper="#" />
      <KeyField Name="server" Sort="asc" />
      <SelectInitialValues>

        select 'AAA', '1/1/1970'

      </SelectInitialValues>
    </QueryKeyFields>
  </Query>
</QueryCollection>
</DataSourceInterface>
<InputFormatCollection>
  <InputFormat InputFormatName="InputFormat">
    <InputFormatFields>
      <InputFormatField Name="timestamp" Type="time"
TimeFormat="%m/%d/%Y %I:%M:%S %p" />
      <InputFormatField Name="server" Type="string" />
      <InputFormatField Name="status" Type="integer" />
    </InputFormatFields>
    <TranslatorSwitch DefaultTranslator="Translator" />
  </InputFormat>
</InputFormatCollection>
<TranslatorCollection>
  <Translator TranslatorName="Translator">
    <TranslatorFields>
      <TranslatorField Name="ResourceId" SourceType="table"
SourceName="ResourceTable" />
      <TranslatorField Name="EventTypeId" SourceType="constant"
ConstantValue="1500" />
      <TranslatorField Name="Timestamp" SourceType="field"
SourceName="timestamp" />
      <TranslatorField Name="Value" SourceType="field" SourceName="status" />
    </TranslatorFields>
  </Translator>
</TranslatorCollection>
<TranslationTableCollection LoadingMode="remote">
```



```
<TranslationTable Name="ResourceTable" DestinationType="resource">
  <TranslationField>server</TranslationField>
</TranslationTable>
</TranslationTableCollection>
</AdapterConfiguration>
```

SQL アダプタの接続文字列

SQL アダプタの接続文字列のメカニズムは以下の機能を提供するために設計されています。

- 同じアダプタで複数の接続文字列を使用する。
- 設定ファイルを変更せずに、ファイル名を変更できる接続文字列テンプレートを使用する。
- アダプタが読み取りを完了したら、ファイル データ ソースを削除する。
- アダプタが読み取りを完了したら、ファイル データ ソースを別の場所に移動する。

接続文字列を簡単に定義すると **ConnectionString** エlement 内の文字列ですが、説明を付け加えると、接続文字列は、接続文字列に連結された指定値を保持するセグメントによって定義することができます。これにより、アダプタは接続文字列を動的に構築することができます。

セグメントには 2 つのタイプがあります。1 つはテキストで、接続文字列にそのまま連結されるテキストを含んでいます。2 つ目はファイルで、ワイルドカードがある（またはない）ファイル名を含んでいます。ファイルセグメントは、1 回のみ使用することができます。ファイルセグメントには、読み取りファイルで実行する必要があるその他の属性が含まれています。

ConnectionString は **QueryCollection** Element および（または）**Query** Element に定義できます。**QueryCollection** Element の **ConnectionString** 定義はデフォルトであり、明確な **ConnectionString** 定義がないクエリにのみ使用されます。

エレメントと属性

ConnectionString エレメント

このエレメントには **Segment** 子エレメントを格納することができます。少なくとも 1 つ以上の **Segment** エレメントが含まれる場合は、接続文字列はそのエレメントと連結されます。その他の場合、接続文字列は **ConnectionString** エレメントテキスト（現在のバージョンに含まれる）から取得されます。

Segment エレメント

このエレメントには、**text** および **file** の 2 つの有効な値を持つ **Type** と呼ばれる必須属性が格納されます。**Type="file"** の場合、**Segment** エレメントには少なくとも 1 つの **File** エレメントを格納する必要があります。**File** エレメントはそれぞれ別のクエリと考えます。

File エレメント

このエレメントには、接続文字列で使用する必要があるファイルおよびアダプタによる読み取り完了後のファイル処理を定義する属性が格納されます。

- **Path** : ファイルパス（ディレクトリ）を定義します。
- **NamePattern** : 既知のパスでファイル名を定義します。これにはワイルドカードが含まれる場合があります。
- **UsePath** : 有効値は **yes** または **no** です。デフォルトは「**yes**」です。「**yes**」に設定した場合、ファイルパスが接続文字列に連結されます。
- **UseFileName** : 有効値は **yes** または **no** です。デフォルトは「**yes**」です。「**yes**」に設定した場合、ファイル名が接続文字列に連結されます（Excel ファイルに関して必要に応じて）。
- **UpdateSchemaFile** : 有効値は **yes** または **no** です。デフォルトは「**no**」です。「**yes**」に設定した場合、アダプタが **schema.ini** ファイルを現在のファイル名に更新します。

注: アダプタによる **schema.ini** ファイルの変更を許可する場合のみこの属性を使用します。

内部変数

追加用の 2 つの内部変数が追加されており、**SelectStatement** エレメントおよび **SelectInitialValues** エレメントに使用できます。内部変数については以下のとおりです。

- **:file_name** : 現在のファイル名および拡張子に置き換えられます。
- **:file_name_no_ext** : 拡張子なしで現在のファイル名に置き換えられます。

例

例 1 : Oracle 用の簡単な接続文字列

```
<ConnectionString> Provider=msdsql;dsn=0; uid=0; pwd=0 </ConnectionString>
```

または

```
<ConnectionString>
  <Segment Type="text" Text="Provider=msdsql;"/>
  <Segment Type="text" Text="dsn=0; "/>
  <Segment Type="text" Text="uid=0; "/>
  <Segment Type="text" Text="pwd=0; "/>
</ConnectionString>
```

例 2 : Excel 用の簡単な接続文字列 (単一ファイル)

```
<ConnectionString>Driver={Microsoft Excel Driver (*.xls)}; DriverId=790;
                    Dbq=d:\oblicore\Availability_2003_01.XLS
</ConnectionString>
```

または

```
<ConnectionString>
  <Segment Type="text" Text=" Driver={Microsoft Excel Driver (*.xls)};"/>
  <Segment Type="text" Text=" DriverId=790;"/>
  <Segment Type="text" Text=" Dbq="/>
  <Segment Type="File">
    <File Path="d:\oblicore " NamePattern="Availability_2003_01.XLS">
  </Segment>
</ConnectionString>
```

例 3 : Excel 用の簡単な接続文字列 (複数ファイル)

```
<ConnectionString>
  <Segment Type="text" Text=" Driver={Microsoft Excel Driver (*.xls)};"/>
  <Segment Type="text" Text=" DriverId=790;"/>
  <Segment Type="text" Text=" Dbq="/>
  <Segment Type="File">
    <File Path="d:\oblicore ",NamePattern="Availability_*.XLS"/>
  </Segment>
</ConnectionString>
```

例 4 : ODBC 標準の DSN エントリの使用

ODBC 標準の DSN エントリを使用すると、アプリケーション サーバの ODBC マネージャで作成された DSN エントリを持つすべてのソースに接続することができます。ODBC 標準の DSN エントリは、サーバの「管理ツール」セクションにあります。

```
<ConnectionString>dsn=SampleDataSource;usr=scott;pwd=tiger;</ConnectionString>
```

ファイル データ ソースからレコードを読み取る

データ ソースと接続する ODBC インターフェースがあれば、SQL アダプタを設定してファイルのクエリを行うことができます。アダプタを複数のファイルを読み取るように設定するには、**ConnectionString** エLEMENT の **Segment** ELEMENT を使用する必要があります。例については、接続文字列について説明している前のセクションを参照してください。

SQL アダプタがファイル进行处理する方法については以下のとおりです。

1. 各クエリでは、アダプタはレコードが取得できなくなるまでファイルの読み取りを試みます。
2. その後、アダプタは次のファイルに移動し、そのファイルの読み取りを試みます。
3. アダプタはそれ以上ファイルがなくなると、**SleepTime** 属性に従ってこのクエリに対してスリープ状態になります。

Schema.ini ファイル

テキスト ODBC ドライバが展開される場合、スキーマ情報ファイル (**schema.ini**) によってテキスト ファイルの形式が決定されます。このスキーマ情報ファイルはテキスト データ ソースと同じディレクトリ内に配置する必要があります。

Schema.ini ファイルは複数のエントリで構成され、各エントリは単一のテキスト ファイルの構造を説明しています。各エントリの先頭行は、角かっこで囲まれたテキスト ソース ファイルの名前です。

アダプタがワイルドカードを使用して定義されたファイル进行处理の際、ファイル名は絶えず変更されます。**schema.ini** ファイル内の名前にはワイルドカードを含めることができないため、アダプタはデータベースに接続する前に **schema.ini** ファイルを変更する必要があります。

そのため、ユーザはファイル エントリ行の前にインジケータ行を追加する必要があります。このインジケータ行には、角かっこで囲まれた、アダプタ設定ファイルの接続文字列 ELEMENT の名前パターンが含まれます。アダプタは、次の行を角かっこで囲んだ新規ファイル名で置き換えます。

注: **schema.ini** ファイルには複数のエントリを格納できます。ユーザの責任で正しい位置に行 (複数) を追加してください。

Schema.ini の例

```
[sqltes*.txt]
[sqltest2.txt]
ColNameHeader = False
CharacterSet = ANSI
```

```

Format = TabDelimited
Col1=id Char Width 30
Col2=idname Char Width 30
-----
[report_200*.txt]
[report_2003_09.txt]
ColNameHeader = False
CharacterSet = ANSI
Format = TabDelimited
Col1=date Char Width 30
Col2=service Char Width 30
Col3=responseTime Char Width 30
-----

```

DataSourceControl ファイル

各クエリのデータ ソース制御ファイルには、ファイル名のパターンおよびアダプタが再起動する際に同じファイルから続行するための現在の入力ファイルの名前が格納されます。

XML 形式での制御ファイルの例を以下に示します。

```

<AdapterControl Save="end" LastSaveTime="2005/03/23 09:16:15">
  <Data>
    <QueryCollection>
      <Query QueryName="TroubleTickets (on D:¥Data¥Incidents*.xls)">
        <KeyField Name="INC_CLOSE_DATE">
          <LastValue>7/03/2005 13:06:21</LastValue>
        </KeyField>
        <LastFileName>IncidentsMarch.xls</LastFileName>
      </Query>
    </QueryCollection>
  </Data>

```

入力ファイルの保持

アダプタは、現在のファイルの読み取りを完了すると、次のファイルを検索します。次のファイルの読み取りは、名前パターンおよびファイル名が前のファイルより大きい（辞書式順序で）という条件に当てはまった最初のファイルに対して行われます。新規レコードが追加されても、アダプタは前のファイルに戻りません。

これらの 2 つの属性が「no」であり、制御ファイルに最後のファイル名が含まれていない場合に限り、アダプタは InitialFileName 属性を使用します。

クエリの確認

アダプタは、定義ファイルが存在する場合のみ、接続文字列およびクエリを確認します。これらがワイルドカードを使用して定義されている場合、アダプタは最初のファイルに対してのみ確認を行います。

アダプタが新規ファイルから読み取りを行おうとすると、エラーが発生します。この場合、クリティカル属性が「yes」のときはアダプタがただちに停止します。「no」のときは、アダプタはこのクエリの実行を止めますが、ほかのクエリを続行します。または、アダプタ現在のファイルを保持して次のファイルへ移動します。

アダプタ内部変数

ファイル名の現在のコンテキストを参照するために設定ファイル内に使用できる 2 つの内部変数があります。これらの内部変数は `:file_name` と `:file_name_no_ext` で、それぞれ現在のファイル名と、現在のファイル名からファイルの拡張子を除いた参照しています。

これらの変数は `SelectStatement` 要素、`SelectInitialValues` 要素、および `QueryKeyField¥Function` 属性内で使用できます。

アダプタは、変数をクエリ内のファイル名に置換します。

例:

- `select date, service, value from ":filename"`
- `select id and name from :file_name_no_ext`

CA Business Service Insight ユーザ インターフェースを使用したアダプタの作成

CA Business Service Insight 環境内で実行されるように設定される各アダプタは、レジストリ内に定義するだけでなく、ユーザ インターフェース内に登録する必要があります。この手順の背景にある理由は、主に、アダプタ リスナ側の設定を確立して、アダプタからのイベントを受信する準備ができていないようにすることです。この手順の間に、変換テーブルやイベント タイプなどのアダプタのすべての設定が定義されます。

以下の手順に従います。

1. アダプタを作成します。
2. リソース/アダプタを選択します。
3. リスト内で既存のアダプタをチェックして、自身のアダプタと同じポート、つまりディレクトリ `OblicoreInterface¥OnlineInterface¥Port` でアダプタの設定ファイル内に定義されているのと同じポート上に何も定義されていないことを確認します。
4. [新規追加] をクリックし、アダプタの作成で使用するメソッドを選択します。ここではいくつかのオプションがあります。
 - a. **手動で作成する**：定義した（またはこれから定義する）アダプタに手動で接続するように、アダプタ リスナを設定します。
 - b. **ウィザードを使用する**：画面ごとのウィザード インターフェースを使用してアダプタを作成します。詳細については、次のセクションを参照してください。
 - c. **テンプレートから**
 - d. **既存の設定ファイルから**：あらかじめ設定されているアダプタ テンプレートをアップロードできます。このテンプレートは、アダプタ ウィザードのフィールドに、指定された設定ファイルのオプション セットを自動的に挿入します。
 - e. 結果の画面は、選択したオプションによって異なります。
5. フィールドに値を入力します。

[ネットワーク アドレス] では、アダプタの IP アドレスを入力します。アプリケーション サーバに対してローカルな場合はローカルホストを、それ以外の場合は、定義されているポートを入力します。
6. [保存] をクリックします。

必要な変換テーブルを作成する方法

注：これらの手順は、設定ファイル内で定義される各変換テーブルについて実行する必要があります。

1. [設計] メニューで、[変換] - [変換テーブル] をクリックし、[新規追加] ボタンをクリックします。
2. すべての変換テーブルの設定は、アダプタ設定ファイル内の相当テーブルの定義に対応している必要があります。
 - 名前: 設定ファイルの変換テーブル、名前の名前属性と一致している必要があります。
 - ソース フィールド: 変換テーブルの **TranslationField** のすべてのフィールドを持っている必要があります。すべてのフィールドを追加します。変換値を構成する 2 つ以上のフィールドの組み合わせがある場合があります。これがどのように見えるかを示す例は以下のとおりです。

TranslationTable	
Name	BatchFileResourceTable
DestinationType	resource
TranslationField	jobname
TranslationField	site

3. デスティネーションタイプ: 設定ファイル内の変換テーブル定義の **DestinationType** 属性に一致する必要があります。(resource、event_type など)。
4. 登録済みアダプタ: この変換テーブルを使用するアダプタを追加します。複数のアダプタで、1 つの変換テーブルを使用することができます。
5. [保存] をクリックします。
6. 各イベントタイプのイベントタイプフィールドの定義をインポートします。
 特定のアダプタ設定ファイルからフィールド定義をインポートできるようにするには、このアダプタを少なくとも 1 回は実行し、CA Business Service Insight に接続しておく必要があります。アダプタは CA Business Service Insight に接続するときに設定ファイルを CA Business Service Insight に送信します。これによりシステムは、設定ファイルからフィールド定義を使用できるようになります。
 注: イベントタイプに対してフィールド定義を手動で指定することもできます。
7. アダプタをアクティブにします。
 - a. [設計] メニューで、[データ取得] - [アダプタ] をクリックします。
 - b. アダプタの [スタート] ボタンをクリックします。

以下のテーブルには、アダプタのさまざまなステータスが含まれています。

ステータス	説明
非アクティブ	初期状態。アダプタは非アクティブで、まだ開始されていません。

リスナ非アクティブ	アダプタ リスナ（ディスパッチャ）サービスは開始されていません。
開始中	アダプタを開始しています。
開始済み	アダプタは開始されています。
停止中	停止中です。
一時停止中	一時停止中です。
一時停止	一時停止。
停止中	アダプタは、アダプタのコンピュータ上で実行されていません。
エラー	アダプタの設定ファイルにエラーがあり、アダプタを開始できません。
接続エラー	アダプタの接続（不正なホスト/ポート）エラー、またはアダプタを最初に実行する前に信号が検出されませんでした。 アダプタを最初に実行するときのステータス。
ブロック	拒否されたイベントが最大数になりました。

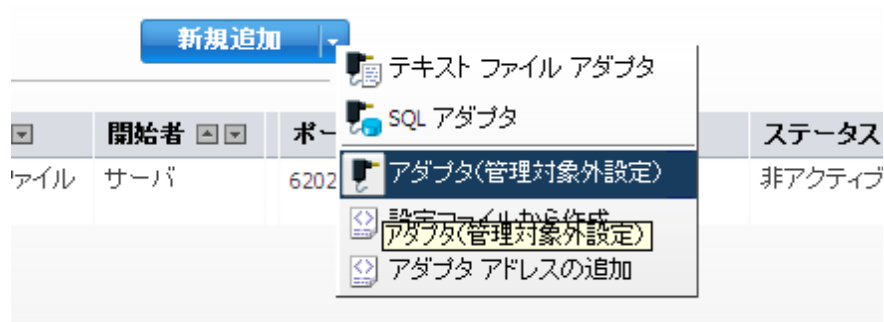
アダプタ ウィザードを使用したアダプタの作成

アダプタ ウィザードは CA Business Service Insight ユーザ インターフェースの新しい機能で、XML エディタよりも直観的なインターフェースを使用して、新しいアダプタを作成するうえでガイダンスを提供します。ウィザードは、アダプタの作成に必要なすべての情報が含まれている一連のタブを介してガイダンスを提供し、最終的には、コンパイルされた XML の設定ファイルのコピーをダウンロードすることができます。ただし、ウィザードで実行できる内容にはいくつかの制限があります。

現在、ユーザは次の処理を実行できません。

- 異なるデータ ソース インターフェースから同じ入力構造（入力フォーマット）を参照すること
- 異なる入力から同じ出力構造（トランスレータ）を参照すること
- 入力フォーマット スイッチを設定し、それを使用して、データ ソース インターフェースからどの入力フォーマットを使用するかを決定すること
- 出力構造内でデータ ソース ID フィールドを定義すること
- 接続文字列に複数のファイルの定義を指定して、同一クエリでテキスト ファイルまたは Excel ファイルをクエリすること
- UTC または UtcNow の時間制約を使用すること
- 先頭または末尾がスペースの値を指定すること

ウィザードを介して新しいアダプタを作成する場合には、選択肢として以下の図に示すような 4 つのオプションがあります。



最初の2つのオプションでは、ウィザードインターフェースを使用してテキストファイルアダプタまたはSQLアダプタを使用することができます。次のオプション [アダプタ (管理対象外設定)] は、XMLエディタを使用して構築した、設定済みのアダプタを追加するときに選択するオプションです。このオプションでは、後でウィザードを使用して、このアダプタに対する設定を編集することはできません。最後のオプション [設定ファイルから作成] では、あらかじめ設定されているアダプタの設定をアップロードし、後で編集するために、システムでその設定をウィザードインターフェースにインポートすることができます。このオプションでは、特定の設定において、前述のウィザードの制約が実施されないようにする必要があります。

オプションの選択が修了すると、アダプタウィザードの設定オプションでは、手動のアダプタ設定で説明したものと同一機能が提供されます。ここでの目的は、設定を編集するための、より簡単でわかりやすいインターフェースを提供することです。手動の場合と同じ機能および原理が適用されるため、詳細については該当するセクションを参照してください。

アダプタの実行およびテスト

アダプタの設定ファイルの設定は通常、1回のサイクルでは完了しません。設定において、アダプタの実行を反復し、その結果をチェックしてアダプタの設定が正しいことを確認することが必要な場合があります。

修正が必要な一般的な問題のうち、いくつかを以下に示します。

- 接続の問題（データソースとアダプタの間、またはアダプタとリスナの間）
- 次の設定ファイルの誤り
 - 不正な構造
 - 属性の誤用
 - 大文字/小文字の相違（アダプタでは大文字/小文字が区別されるため、'Yes'と'yes'が異なる、など）
 - 不正な値の割り当て
- データ操作の誤り（出力のイベント構造、不正なイベント値、クエリ内の誤りなど）

以下の手順に従います。

1. アダプタで `RunOnce = "yes"`、および `LogDebugMode = "yes"` に設定し、`RejectedEventsUpperLimit` の設定を適当な数に設定します（「[アダプタの実行モード](#)」(P. 97)）を参照してください）。
テストに必要な設定は以下の図のとおりです。
2. また、設定ファイル設定を設定するために、オフライン モードを使用することもできます。
3. 設定ファイルを正常にロードしたら、設定をオンラインに戻します（「アダプタの実行モード」を参照してください）。
4. それぞれの反復には、以下の手順が含まれます。
 - a. アダプタの設定ファイルを更新/修正します。
 - b. アダプタのすべての出力ファイルを削除します。
 - c. **Adapter** 実行可能ファイル、または作成された **.bat** ファイルへのショートカットをダブルクリックし、アダプタを実行します。
 - d. ログブラウザ（CA Business Service Insight ユーティリティ内にある **Log Browser.exe** ユーティリティ）を使用してアダプタのログ ファイルを開き、エラー メッセージがないことを確認します。
5. 最初の手順は、正しい設定ファイルを取得し、アダプタが設定ファイルを正常にロードした状態にすることです。CA Business Service Insight およびデータソースに正常に接続し、変換に対するイベントおよび要求を拒否するまでこの手順を繰り返します。

6. この段階を完了するには、以下の内容を確認します。

- アダプタのログ ファイルにエラー メッセージがない。
- アダプタが CA Business Service Insight およびデータ ソースに正常に接続している。
- アダプタが、変換の要求、および拒否されたイベントを送信している。

アダプタが拒否するすべてのレコードについて、コンソールに "R" の文字が表示されることが予想されます。必要なすべての変換が完了するまで、拒否されたイベントが要求されることを覚えておいてください。

7. rejectedEvents ファイルにレコードが存在し、空ではないことを確認します。



8. CA Business Service Insight にログインして「変換エントリ」ページに移動し、保留中の変換エントリをアダプタから検索します。いくつかのエントリがあることが予想されますが、これらのエントリは、各変換要求に対して 1 つ、アダプタが送信したものです。

警告： アダプタの出力ファイルを削除することは非常に危険です。この段階でアダプタの出力ファイルを削除することは、テストの目的でのみ行うようにします。たとえば制御ファイルを削除すると、アダプタは自身が読み込んだファイルのトラックを失うため、データを失ってファイルを再読み込みすることがあります。機能的な結果を伴わずに運用モード中に削除できるファイルは、ログ ファイルだけです。

ログ ブラウザを使用してエラー メッセージを表示する方法

エラー メッセージがある場合は、そのメッセージをダブルクリックして読みます。これは通常、設定ファイルのエラーによって生じます。

リソースとイベントの変換

前の手順で、アダプタの実行中に作成され、拒否されたイベントが多数ありました。これらの拒否されたイベントは **RejectedEvents.txt** ファイルに格納されましたが、保留中の変換エントリとして **CA Business Service Insight** データベースにも格納されました。**Raw** データをシステムにロードする過程の次の手順は、測定されたものの変換を提供し、必要に応じて **CA Business Service Insight** がこのデータを使用できるようにすることです。

Raw データ イベントは、イベント タイプ、またはリソースがシステム内で定義されていないために拒否されることがあります。保留中のイベントは、それらのイベントが生じた変換テーブルのタイプによって定義されます。最も一般的な例は、ある変換テーブルからイベント タイプが生じて、別の変換テーブルからリソースが生じている場合です。

アダプタで新しいリソースが検出されると（たとえば、ネットワーク モニタリング ツールに新しいサーバが追加され、このモニタリング ツールから、データ ソースに新しいエントリとして表示される場合）、これはシステムのリソース モデルに追加することができます。この新しいリソースを、**CA Business Service Insight** でレポート可能にするためには、2つの手順が必要です。

最初に、**CA Business Service Insight** のエンティティとしてリソースを作成し、次に変換を入力する必要があります。これにより、データ ソース内で見られる文字列の表現と、**CA Business Service Insight** 内でリソースとして定義されたエンティティの間にリンクが提供されます。この 2つの手順のプロセスは、「追加および変換」と呼ばれる 1つのプロセスでユーザ インターフェースを介して 1つのアクションで実行することができます。これにより、新しいリソースおよび必要な変換エントリを 1つの手順で作成できます。追加および変換する場合には、エントリの割り当て設定が同じであれば、複数のエントリを選択することができます。変換の実行は、データ ソースの値と **CA Business Service Insight** エンティティ間の一致を作成するプロセスです。変換を作成すると、一致する値を持った変換テーブルに対して 1つのエントリが追加されます。その後、データ ソースに対してクエリが発行された場合、アダプタはこの新しい値をどのように処理するかを自動的に認識します。

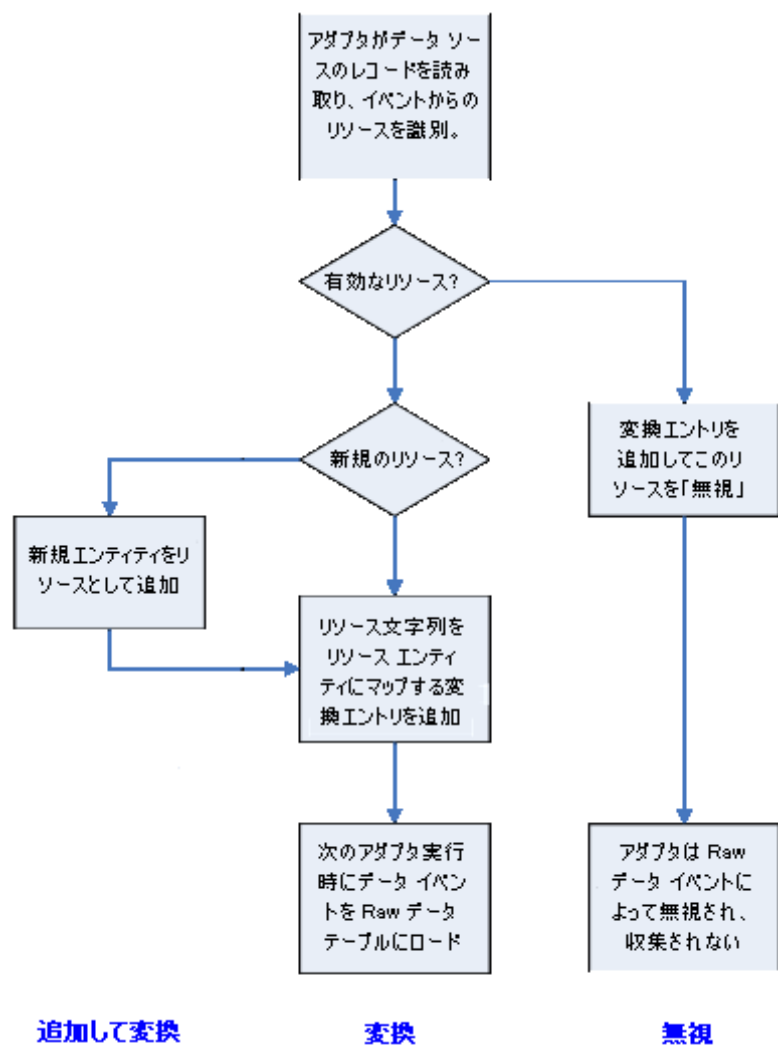
この段階で、アダプタは、変換フィールド内の各値について、変換の要求をすでに実行し、送信しています。これらの値と関連付けられたイベントは拒否され、変換が完了すると、**CA Business Service Insight** に送信されるのを待機します。変換は、手動で実行することも、変換スクリプトを使用して自動で実行することもできます。

保留中の変換エントリに対して、実行できる変換アクションは以下のとおりです。

変換： データ ソース値と、該当する CA Business Service Insight エンティティの間に一致を作成し、変換テーブルにエントリを作成することができます。変換が行なわれる CA Business Service Insight エンティティは、あらかじめ存在している必要があります。(わかりやすい例として、データ ソース内のリソースのスペリングが間違っている場合があります。この場合、実際には同じ論理エンティティを参照しているのに、データソース内では異なる名前が指定されている可能性があります。たとえば Server001 と SERVER001 などです。CA Business Service Insight リソースは大文字と小文字を区別します)。

- **追加および変換：** CA Business Service Insight に新しいエンティティを作成し、同時に変換テーブルでそのエンティティに対して変換エントリを追加できます。CA Business Service Insight でインフラストラクチャを構築するために変換メカニズムが使用されるため、これは保留中の変換エントリで実行される、最も一般的なアクションです。
- **無視：** あるエントリを無視すると、関連付けられているすべてのイベントは無視され、CA Business Service Insight の Raw データ テーブルに送信されません。無視されたイベントは失われます。たとえば、データ ソースには、あるデータ センターのすべてのサーバに関する情報が含まれていて、サービス レベルの計算ではアプリケーション サーバのデータのみが必要な場合、すべてのサーバが変換用のアダプタに到達しますが、アプリケーション サーバのみが変換されます。CA Business Service Insight では、不要なイベントも必ず無視されるため、他のすべてのものは無視されます。必要な場合は、無視されたエントリを後で変換することもできますが、このデータはそれ以降、単純にキャプチャします。
- **削除：** 変換エントリが削除され、関連する拒否されたイベントも削除されます。同じ値が後でデータ ソースによって再送信される場合、新しい保留中エントリが作られます。

これらのオプションを使用するケースに関するまとめは以下のとおりです。



手動変換

CA Business Service Insight 内にすでにエンティティが存在している場合は、手動変換が必要です。手動変換は、多くの状況で発生することがあります。たとえば、外部ソースからリソースを自動で作成する変換スクリプトが実行された（ただし変換は自動では行われなかった）場合。または、データソースに不明瞭なエントリがあり、1つのリソースが別の方法で2回定義されている場合（Server001p と server001P など）。これは、手動で作成されたリソースが原因である場合もあります。

リソースがすでに存在する場合に手動変換を実行する方法

1. [設計] メニューで、[変換] - [変換エントリ] をクリックします。
2. デフォルトで、保留中のすべてのエントリが表示されます。
3. 保留中の変換エントリの中で、変換するものを選択するには、対象のエントリの隣にあるチェック ボックスをオンにします。
4. [変換] をクリックします。
5. リストから、該当するエンティティ（[リソース]、[イベントタイプ] など）を選択します。（リストに項目が表示されない場合は、ボックス内のデフォルトの検索条件を変更しなければならないことがあります）。
6. 項目が含まれている行をクリックし、表示されているエンティティのリストから、[リソース] / [イベントタイプ] を選択します。選択されたものは、強調表示された状態になります。
7. [変換] をクリックします。ここで変換エントリは、システム内に格納されます。

リソースが存在しない場合に手動変換を実行する方法

1. [設計] メニューで、[変換] - [変換エントリ] をクリックします。
2. デフォルトで、保留中のすべてのエントリが表示されます。
3. 保留中の変換エントリの中で、**CA Business Service Insight** のリソースにして変換するものを選択するには、対象のエントリの隣にあるチェック ボックスをオンにします。
4. [追加および変換] をクリックします。
5. 意図したとおりにリソース名が指定されていることを確認します。 表示名フィールドをカスタマイズして、リソースがレポート上でどのように示されるかを変更することもできます。 このリソースを「変更セット」の一部として管理する場合は、特定の変更セットもここで指定する必要があります。
6. リソースの[発行日] は、システム上でこのリソースがレポート可能になる日付として設定する必要があります。 ここで指定された日付より前には、このリソースはレポートに表示されないことに注意してください。
7. [詳細] タブをクリックし、リソースの関連付けについて、[有効] (true/false)、[リソース タイプ]、[リソース グループ メンバシップ]、[サービスおよび契約] の関連付けのオプションから選択します。
8. [保存 & 変換] をクリックします。 リソースはリソース サービス カタログに追加され、変換エントリもシステムに格納されます。 保留中のエントリがすべて処理されたら、データがシステムにロードされたことを確認できます。
9. **CA Business Service Insight** 内のリソースを参照し、新しいリソースが作成されたことを確認します。
10. アダプタをもう一度実行します。
11. 拒否されたイベント ファイルが空であることを確認し、コンテンツとサイズを確認します。
12. 送信されたイベント ファイルが空であることを確認し、コンテンツとサイズを確認します。
13. **Raw** データ ツールを使用して、**Raw** データに追加されたイベントを確認します。

インフラストラクチャのセットアップ

インフラストラクチャのセットアップには、設計フェーズで識別されるような、データ モデリング エンティティの定義が含まれています。

この段階では、すべてのインフラストラクチャのエンティティの定義は含まれていません。すべての定義は、アダプタの設定が完了したときに完全になります。

この段階では、以下のエンティティが **CA Business Service Insight** システムに提供されます。

- イベント タイプ
- リソース タイプ
- リソース & リソース割り当て
- リソース グループ

注: 一度アダプタが正常に実行されると、イベント タイプ フィールドを定義するときに自動インポート機能を使用することができます。

変換スクリプトによる自動変換

自動変換は、変換アクションを実行するスクリプトを使用して、外部データ ソースに基づいて、作成と変換のプロセスおよびインフラストラクチャを自動化します。

自動変換は、変換スクリプトによって実行されます。変換スクリプトにより、新しいIT およびビジネス リソースを **CA Business Service Insight** へマッピングするプロセスが高速化されます。変換スクリプトは、新しい変換エントリが受信されたタイミングを自動的に認識し、リソースを変換して、迅速で効率的なリソースマッピングを実現します。自動化は、**CMDB** に対するインターフェースをサポートし、システムは、設定済みの定義に基づいてリソースを識別することができます。自動変換には、変換のメンテナンスを容易にしてエラーを防ぐことなど、多くの利点があります。変換スクリプトを使用して新しいリソースを作成し、変更を割り当てることができます。

また、変換スクリプトを使用して以下の処理を行うことができます。

- 既存の **CA Business Service Insight** オブジェクトへエントリを変換する。
- 新しい **CA Business Service Insight** オブジェクトを追加し、既存の変換エントリに基づいてそれらを変換する。
- オブジェクトを作成し、**CA Business Service Insight** 外部のテーブル（外部の **CMS** のリソース テーブルなど）に基づいてエントリを変換する。
- オブジェクトが存在するかどうかを確認する。
- リソースを作成し、リソースの割り当てを実行する（リソース タイプ、リソース グループ、契約関係者、サービス コンポーネントなど）。
- リソースを変更セットに対して割り当て/割り当て解除する。

ユーザ インターフェースでは、変換プロセスも手動の手順としてサポートされているため、どの変換プロセスを選択するかを決定する必要があります。この決定を行う場合には、自動変換に関する以下の長所と短所も考慮します。

- プロジェクトの複雑さが増加した。変換スクリプトでは適切なスキルと、スクリプトを開発する労力が必要になる。
- 開発時間、およびテストのための品質保証時間が増えた。
- 変換プロセスが単純に 1 回だけ行われる場合は、実装する必要がない。
- 第 2 フェーズのアプローチの一部として、実装をスケジュールすることができる。
- 継続的なメンテナンスが減少した。
- 人的な変換エラーが減少した。
- 変換スクリプトの作成および実行の詳細については、**SDK** のガイドで「**CMDB Integration**」の章を参照してください。

アダプタ機能の詳細トピック

以下のトピックでは、アダプタ機能の詳細なトピックについて説明します。

イベント固有性

イベント固有性はアダプタのメカニズムで、**T_Raw_Data** テーブルに対して重複する **Raw** データの挿入を回避するためのプロセスを提供します。

イベント固有性を使用しない場合、データ ソースに対してアダプタを実行し、データベースにイベントをロードしても、同一のイベントに対して検証が行なわれません。イベント固有性の機能は、挿入前に、イベントの固有性をチェックするかどうかを指定し、重複したイベントが検出された場合はどのようにするかを指定することによって、この問題を解決します。ただしこの検証プロセスは、アダプタのパフォーマンスにマイナスの影響を与えることがあります。

このソリューションでは、ユーザが、イベントのフィールドに基づいてキーを定義できるようにします。このようなキーはイベントの一意性を表しており、2つのイベントが同じキーを持っていれば、それらのイベントは同じイベントであることを意味します。

また、データベース内に重複するキーが見つかった場合に実行するアクションを指定することもできます。これらのアクションについては、以下で説明します。

注: キーは、トランスレータのいくつかのフィールドの組み合わせとして定義することができます。

イベント固有性を備えたアダプタ設定ファイル

TranslatorCollection/Translator/OnDuplication

このフィールドは、データベース内にイベントがすでに存在している場合に、何を行うかを決定します。

有効な値は以下のとおりです。

- **Add** : (追加の) イベントをデータベースに挿入します。
- **Update** : (場合によっては) 既存のイベントを削除し、イベントが変更されたことを検証した後で新しいイベントを挿入します。
- **updateAlways** : (場合によっては) 既存のイベントを削除し、変更を確認せずに新しいイベントを挿入します。
- **Ignore** : データベースに新しいイベントを挿入しません。

デフォルト値は **Add** です。

Ignore と Add

Ignore モードでアダプタを実行すると、パフォーマンスがわずかに低下することがあります。ただし、**Add** モードでは、イベントが重複している場合は常にシステムで再計算がトリガされます。

Update と updateAlways

Update オプションを使用するとアダプタのパフォーマンスが低下します。**updateAlways** を使用すると、すべてのケースで再計算をトリガするうえに、計算のパフォーマンスが低下します。

TranslatorCollection > Translator > TranslatorFields > TranslatorField > IsKey

この属性は、自身が属しているフィールドが、**Raw** データの一意キーに対して値を提供する必要があるかどうかを決定します。値が **"yes"** の場合は含めるようにし、値が **"no"** の場合は含まれません。標準フィールド (**ResourceId**、**EventTypeId** および **Timestamp**) に対するデフォルト値は **"yes"** で、その他のすべてのフィールドのデフォルト値は **"no"** です。**Raw** データが必要な完全性を保持して、計算が正確に行われるよう、キーは注意して選択する必要があります。

イベント固有性を備えたアダプタリスナの動作

アダプタから新しいイベントを受け取ると、リスナは **OnDuplication** フィールドの値を確認します。値が "add" の場合、標準の挿入プロセスが実行されます。値が "add" でない場合、リスナは、データベース内の同じ **UniqueKey** および同じリーダ ID を持つイベントの存在を確認します。データベース内にこのようなイベントがすでに含まれている場合、**OnDuplication** の値が "ignore" であれば、新しいイベントはデータベースに挿入されません。

OnDuplication の値が "update" の場合、イベント内の変更に対してチェックが実行されます。すべてのフィールドが同じ場合、新しいイベントはデータベースに挿入されません。

OnDuplication の値が "updateAlways" の場合、前述のチェックはスキップされ、更新が行なわれます。

update モードと **updateAlways** モードの両方で、以下の手順が行なわれます。

- **ResourceId**、**EventTypeId** および **Timestamp** が変更されている場合、古いイベントの計算式を使用するすべてのメトリックに対して再計算が行なわれます。
- イベントは適切な値で更新されます。

トランザクション データの集約

トランザクション データは、しきい値と照合するため、または期間内の最終的な成功率を計算できるようにするために、頻繁に収集されます。たとえば、システムに対して 5 分ごとに仮想トランザクションが実行され、結果（ミリ秒単位の応答時間）が以下のように格納されます。

```
[...]
1/1/2004 00:00 432
1/1/2004 00:05 560
1/1/2004 00:10 329
1/1/2004 00:15 250
1/1/2004 00:20 275
1/1/2004 00:25 2860
1/1/2004 00:30 140
[...]
```

仮想トランザクションを使用しないような他の状況では、システム内で実際のトランザクションへのアクセスが生じることがあります。これらの場合には、1 時間に何百、または何千のトランザクションが実行される可能性があります。

これらの両方のケースでは、このようなボリュームの情報を **CA Business Service Insight** にロードすることは、可能であれば回避する必要があります。

データ量を減らすための最適な方法は、一定期間ごとにデータを集約することです。成功を測定する対象となるしきい値が修正されると、アダプタで、成功した集約期間内のトランザクション数をカウントすることによって、集約を実行することができます。たとえば、前の例で成功のしきい値が 500 ミリ秒に設定されている場合、表示されている期間では、7 個のトランザクションのうち 5 個だけが成功しています。このアプローチでの問題は、修正されたしきい値です。もし、後でしきい値を変更したい場合にはどうしますか? このような場合には、**Raw** データを再読み込みし、新しいしきい値に対するアダプタでテストする必要があります。

したがって、アダプタは、重要なデータを失わずに柔軟な方法でトランザクション データを最適に集約する必要があります。

制約のあるソリューションでは、アダプタで、いくつかのしきい値に対してトランザクションをテストします。このテストを行うには、以下の 2 つの方法があります。

- 1 つのイベントと複数のテスト: `Event Type = {TotalTransactions, TransBelow250, TransBelow500, TransBelow750, [...]}`
- 複数のイベントと 1 つのテスト: `Event Type = {TotalTransactions, Threshold, TransBelowThreshold}`.

これらの 2 つのオプションには同じ問題があります。つまり、しきい値は将来的に、あらかじめ定義された小さな値セット内でしか変更できないということです。

推奨されるソリューション

前提：考えられるすべてのしきい値は、特定の値の倍数とします。たとえば、すべてのしきい値（ミリ秒）は 250 の倍数となっており、つまり 0、250、500、1750 および 3000 ミリ秒はすべて有効なしきい値です。

この前提に基づくと、推奨されるソリューションは、すべての値を公倍数の概数で表し、概数化されたそれぞれの値に何個の値が該当するかをカウントすることによって、トランザクションを集約することです。Event Type = {RangeFrom, RangeTo, TransactionCount}

たとえば、上記に表示されるデータを集約するために以下のイベントが生成されます。ここで公倍数は 250 ミリ秒です。

```
{1,      250,    2}
{251,    500,    3}
{501,    750,    1}
{751,   1000,    1}
```

コメント：

それらのイベントのタイムスタンプは同じになります（たとえば、集約されたすべてのイベントは 1/1/2007 00:00 に発生する可能性があり、1 時間ごとに集約すると仮定すると、1/1/2007 01:00 に次のサンプルについて、もうひとつのイベントセットが存在する可能性があります）。

RangeTo は、トランザクションを公倍数の概数にすることによって計算されます（以下を参照してください）。

RangeFrom は、RangeTo から倍数を減算し、1 を加えた値です。これは明確さの理由のみで指定するもので、省略することもできます。

例：1 時間ごとの集約は以下のように示されます（MULTIPLE を倍数の値に置き換えます）。

```
select trunc (time_stamp, 'hh') "TimeStamp",
       round (response_time/MULTIPLE, 0)*MULTIPLE-MULTIPLE+1 "RangeFrom",
       round (response_time/MULTIPLE, 0)*MULTIPLE "RangeTo",
       count (*) "TransactionCount"
from   t_log
group by trunc (time_stamp, 'hh'),
         round (response_time/MULTIPLE, 0)*MULTIPLE
```

ビジネス ロジックでは、イベントに対して以下の条件が適用されることがあります。

```
If eventDetails("RangeTo")<=Threshold Then
    SuccessfulTransactions=SuccessfulTransactions+eventDetails("TransactionCount")
End If
```

最終的な洞察：

- 通常、トランザクションは分散する傾向にあるため、集約イベントの数は比較的少なくなります。
- 大きい公倍数を選択するほど、集約イベントの数が少なくなります。
- 集約イベントのボリュームは、常に **Raw** データのボリューム以下でなければなりません。

ファイアウォールの背後でのアダプタの実行

ファイアウォールの背後でアダプタを実行できるようにするには、以下のソリューションが推奨されます。

2つのポートを開きます。1つ目のポートは **CA Business Service Insight** 内のアダプタに割り当てられ、2つ目のポート（これは任意ですが推奨されます）はログサーバポートです。ログサーバポートは、デフォルトではポート **4040** として定義されます。ログサーバポートを開くと、アダプタは **CA Business Service Insight** のログに対してレポートできるようになり、アラートも生成することができます。アダプタは常にローカルログファイルに対してレポートするため、これはオプションです。

これらの2つのポートに対して、使用されるプロトコルは **TCP/IP** です。

複数のディレクトリからのデータのロード

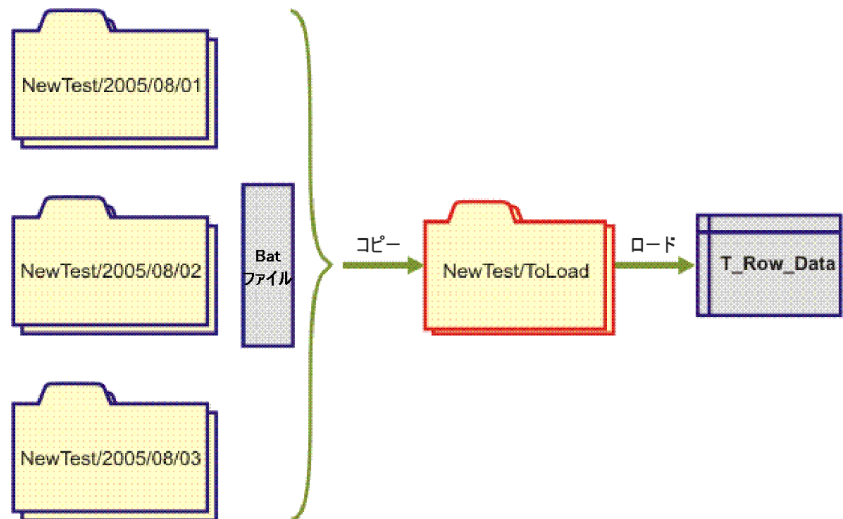
このセクションでは、データソースファイル（CA Business Service Insight アダプタに対する入力）が毎日、またはすべての設定期間において、（特定の命名規則に従って）異なるディレクトリに配置される場合に、実装できるソリューションについて説明します。

たとえば、ディレクトリ構造が `c:\¥NewData¥YYYY¥MM¥DD` であるとします。この場合には、日付ごとに新しいディレクトリが作成され、その日のファイルは対応するディレクトリに配置されます。

CA Business Service Insight アダプタは、最新のディレクトリを介して参照し、新しいファイルをロードする必要があります。

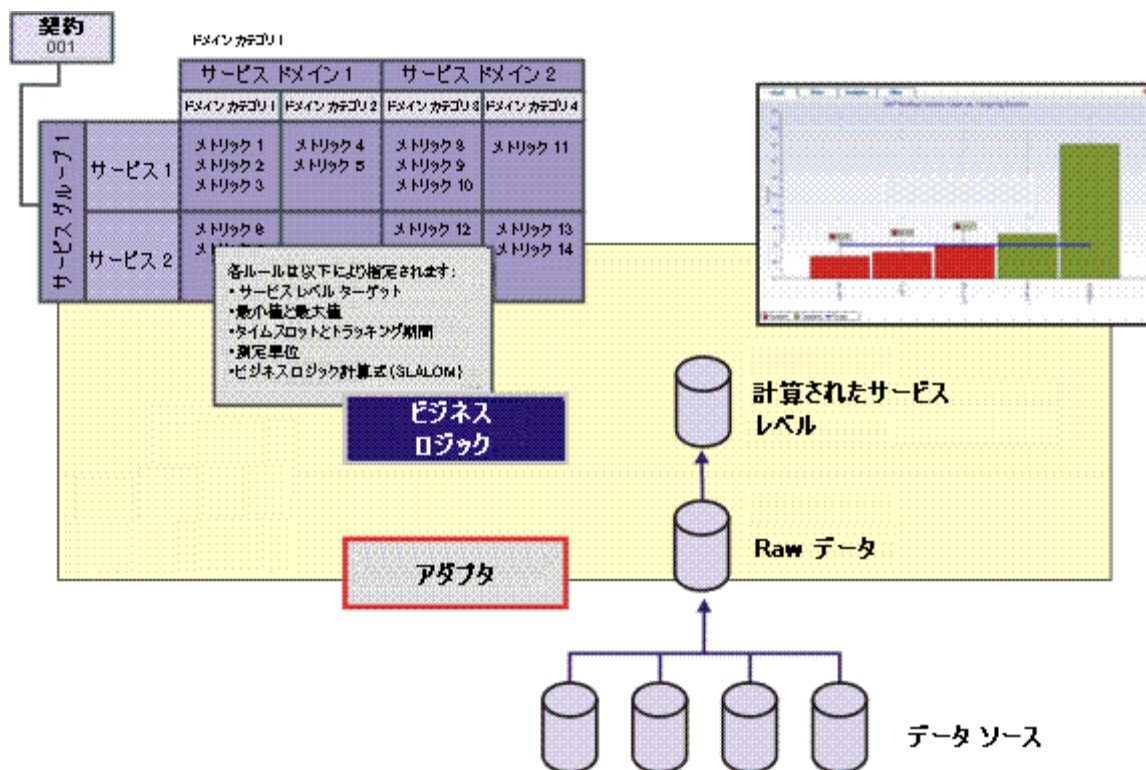
有効な対策として、アダプタを実行する `bat` ファイルの先頭にいくつかのコマンドを追加する方法があります。これらのコマンドは、（規則に従って）特定のフォルダから、この目的のために明示的に作成された専用の 1 つのフォルダにロードする必要があるデータソースをコピーします。アダプタは、情報を常にこのフォルダからロードします。

以下の図は、このソリューションについて説明しています。



ビジネス ロジック スクリプティング (ビジネス ロジック エキスパート)

CA Business Service Insight 内のビジネス ロジックの位置は以下の図のとおりです。
ビジネス ロジックは、契約内の各メトリックに分類されます。



実装のこの段階では、対象のアダプタが設定され、**Raw** データ レコードは、**CA Business Service Insight** の **T_RAW_DATA** テーブル内ですでに利用できるようになっています。ここでは、各メトリックについて実際のサービス レベル結果を生成できるよう、イベントに対してビジネス ロジックを適用することができます。

ビジネス ロジック スクリプティングは、サービス レベルを計算するために、**Raw** データ (アダプタによって送信される **Raw** データ) 上で論理的に稼動するコードを記述するプロセスです。

システム内の多くのメトリックには通常、共通した **1** つのロジックがあり、**Raw** データ イベントの複数のセットにこれを適用することができますが、各メトリックには、独自のビジネス ロジック計算式があり、これを使用して実際のサービス レベルを計算します。

たとえば、**Severity 1** のチケット解決時間を計算するメトリック、および **Severity 2** のチケット解決時間を計算する別のメトリックは別のレコードセットを評価します。つまり最初のメトリックは **Severity 1** チケットのみを使用し、**2** 番目のメトリックは **Severity 2** のチケットのみを使用します。ただし、解決時間を計算するためには、両方のメトリックで多くの場合、同じメソッドを適用します。解決時間スクリプトは開発され、**1** 回テストされ、ビジネス ロジック モジュールとして定義されます。また、このモジュールをメトリック ビジネス ロジックのエリアに含めることによって、両方のメトリックで使用されます。

したがって通常は、ビジネス ロジック スクリプトを開発する場合、メイン モジュールまたはテンプレートは、システム内のすべてのメトリックで使えるように開発されます。また、各ドメイン カテゴリでは通常、別の測定タイプを反映するため、各ドメイン カテゴリは一般的に、別のビジネス ロジック モジュールまたはテンプレートに従います。

ビジネス ロジック スクリプティングのワークフロー

ビジネス ロジック スクリプティングの段階には、以下の手順の実行が含まれています。

- 計算式の定義

設計フェーズで定義された計算の要件に基づいて計算式を作成します。定義された計算式はすべて、契約メトリック内のさまざまな組み合わせにおいて、それぞれビジネス ロジック モジュールとして使用するための一意の計算式です。

たとえば、契約で平均のチケット解決時間を計算するために3つのメトリックを保持しており、各チケットの優先度に対して1つのメトリックがある場合は、チケットの解決時間を計算するために1つの計算式が開発され、その計算式のパラメータとしてチケットの優先度を備えています。この計算式は1回テストされ、1つのモジュールとして定義され、関連するすべてのメトリックに割り当てられます。

- 計算式のテスト

計算式が正しく、エラーがないように定義されており、計算の結果が予想どおりになることを確認するために、テストが実行されます。テストの一部として、極端な限界のケース、および境界条件がすべて含まれていることが重要です。ビジネス ロジック スコープは、テストの目的のために計算式が実行される範囲です。計算式は、最初に定義されたときに全体がテストされます。その後、すべてのメトリックに対していったん、1つのモジュールとして適用されます。スコープ内で、少なくとも1回は各メトリックを実行し、イベントを受信すること（つまり登録が正しいこと）、および適切な結果が生成されることを確認することが重要です。

- 関連付けられている SLALOM モジュールの定義

各モジュールは一意のビジネス ロジックの計算であり、パラメータの定義を使用して、関連するすべてのメトリックに適用されます。モジュールを定義するときに、モジュールを完全にテストし、詳しく文書化する、つまりモジュールが何を行うのか（計算の説明）、どのパラメータが要求されるのか（名前、意味、使用など）を文書化することが重要です。

- 対象のビジネス ロジック モジュールに対するすべてのメトリックの割り当て

定義された契約内の各メトリックについて、対象のビジネス ロジック モジュールに対して1つのリンクを定義する必要があります。このリンクはビジネス ロジック スコープ内で実行し、リンクが正しく実装されていること、および登録が正しく機能していることを確認する必要があります。これらのことを確認するには、対象のイベントを受け取り、予想どおりの結果になることを確認します。

ビジネス ロジック モジュール

1つのメトリック内で複数のモジュールを使用する場合には特に、ビジネス ロジック用のモジュールの作成において、考慮すべき重要事項がたくさんあります。

- モジュールの使用を明確にするためには、対象のメトリックのビジネス ロジックの先頭にコメントを追加する必要があります。
- メトリックのビジネス ロジック スペース内で使用しているコードが少量で、そのコードの中に1つ以上のモジュールが含まれている場合は、いずれかのデフォルト イベント ハンドラまたはサブルーチンについて、メインのメトリック ビジネス ロジックからそのコードセクションを削除する必要があります。特定のメトリックによって使用されているそれぞれのモジュールにおいて、各サブルーチンおよびイベント ハンドラが1回のみ定義されていることを確認する必要があります。これは、計算エンジンの混乱を回避して、予想どおりの結果を生成するためです。

注: たとえば、モジュール内に `OnPeriodStart()` 関数を定義し、その中に特別なコードを挿入します。また、メトリックのメイン ビジネス ロジックの画面で、デフォルトの `OnPeriodStart()` にはコードを挿入せず、そのままにすると、実行時にエンジンは、どのサブルーチンを実行するのか認識しません。このため、実行させようと意図しているコードが実行されない場合があります。

- **OnRegistration** サブルーチンをパラメータ化する場合は、慎重にする必要があります。場合によっては、**OnRegistration** サブルーチンをパラメータ化すると、新しい **Raw** データの追加に基づいてメトリックを再計算するために、システム内に構築された自動トリガが壊れることがあります。

ビジネス ロジックの内部

ビジネス ロジックはすべてイベント駆動型スクリプト内にあり、VBScript の構文を基準としています。ビジネス ロジックに到達するイベントはそれぞれイベントハンドラのトリガになります。

以下のタイプのイベントはエンジンによって送信され、ビジネス ロジックによって評価されます。

- **Raw** データ イベント。 ビジネス ロジックがその結果のベースとする実際の **Raw** データ入力。 エンジンは、数式の登録を基準とする、関連する **Raw** データ イベントのみを送信します。
- エンジン イベント。 エンジンによって暗黙的に送信されたイベントで、タイムスロット定義やトラッキング期間など、メトリック定義のプロパティを反映するもの。
- 中間イベント。他のビジネス ロジック スクリプトによって生成されたイベントで、他の内部で使用できます。

メトリック定義内に含まれていたビジネス ロジック計算式はイベントを評価するもので、レポートが基づくサービス レベル結果を出力します。これらのサービス レベル結果およびドメイン定義に応じて、エンジンはまた偏差結果（サービス レベル目標値が **Metric** に適用されている場合）を作成します。作成された結果は、**T_PSL** と呼ばれるデータベース テーブルに格納されます。 レポート作成時、レポートウィザードが問い合わせるのはこのテーブルです。従って、レポートデータはレポートのパフォーマンスが必ず最大になるよう、すべて事前に計算されます。

イベントフロー

前述のように、ビジネス ロジックに対する入力、エンジン イベントと Raw データ イベントです。

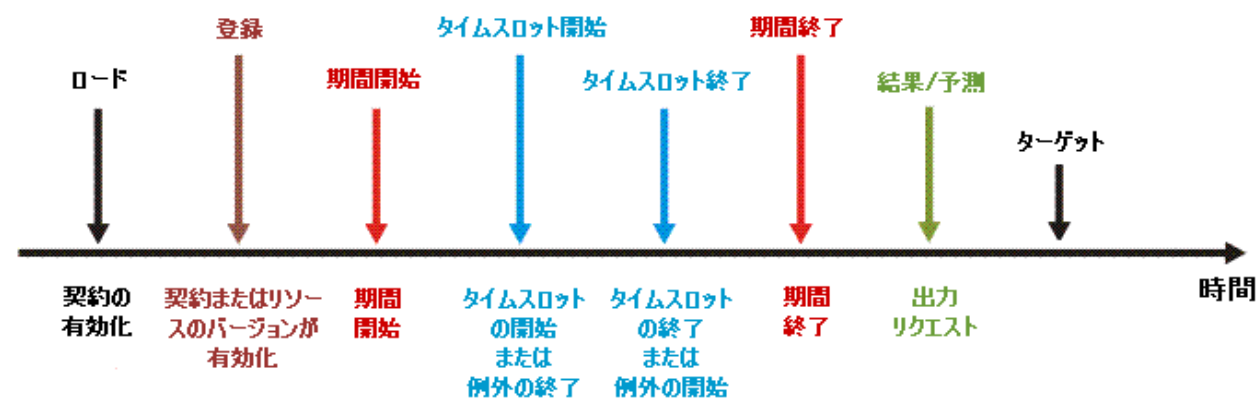
ビジネス ロジックによって受信された Raw データ イベントは、登録関数によって決定されます。この関数内でコードは、イベント タイプおよびリソース識別子によって定義されている、Raw データ イベントの特別なセットを要求します。

ビジネス ロジックにおける登録では、イベントが受信されたときに Raw データ イベントを処理するために実行する、ユーザ定義サブルーチンも関連付けます。(デフォルトでは `OnXXXEvent` という名前ですが、これはもっとわかりやすい名前に変更する必要があります)。

エンジン イベントは、関連する契約およびメトリックの定義に従って、エンジンによってトリガされます。エンジン イベントがトリガされ、受信されると、エンジンは適切なイベント ハンドラを実行します。それぞれのエンジン イベントには、暗黙的なイベント ハンドラが 1 つあります。これらのイベント ハンドラは `VBScript` の先頭に定義されている関数とプロシージャです。登録および「結果」関数を処理するイベント ハンドラは、コード内で実装するために両方とも必須です。他のすべてのイベント ハンドラは任意です。ただしビジネス ロジックは、イベント ハンドラが実装されないエンジン イベントは処理しません。そのため、将来的に拡張できるように、(使用されなくても) それらのものを同じ場所に残しておくのは良い方法です。

注: ビジネス ロジック スクリプトを記述する場合には、すべてのエンジン イベントを実装して、考えられるすべての可能性を網羅できるようにすることが重要です。たとえば、実装の最初の段階でタイムスロットの定義が必要でなかったとしても、将来的に必要な場合には、すべての計算式を修正する必要があります。このため、ビジネス ロジック エキスパートは、最初が必要でなくても後でこの動作が必要になった場合に、必要なシステム変更が少量で済むように、`"in timeslot"` と `"out of timeslot"` の期間の動作を詳細に定義しておくことをお勧めします。

以下に、さまざまなエンジン イベントとそのイベント ハンドラを示します。



- **OnLoad (Time)** : (任意)。契約がアクティブになると、計算の開始時に 1 回コールされます。グローバル変数を初期化するために使用できます。
- **OnRegistration (Dispatcher)** : (必須)。対象の Raw データ イベントを要求し、これらの Raw データ イベントを、処理されるユーザ定義プロシージャに関連付けるためのプロシージャ。ディスパッチャ オブジェクトのメソッドを使用することにより、イベントが要求され、プロシージャに関連付けられます。
OnRegistration はメトリックの計算の最初に、計算エンジンによって 1 回コールされ、メトリックに登録されているリソースが有効になるたびに再計算されます。その後エンジンは、そのリソースに対して行われた変更セットを評価しますが、これは計算式に分類されるイベント セットに影響を与えることがあります。エンジンにはイベント要求の定義があります。これは、イベント タイプおよびリソース識別子によって、リソース（または複数のリソースが含まれている変更セット）が、このセットに関連する何らかのものを変更した場合に定義されるものです。これがいったん有効になると、OnRegistration イベント ハンドラがトリガされます。
- **OnPeriodStart (Time)** : (任意)。(時間単位に従って設定される) エージェント期間の最初にコールされます。最初の OnPeriodStart がいったんトリガされると契約が有効になります。ここで、残りの期間は、1 時間すべての時間単位で開始されます。このイベント ハンドラは通常、変数を定期的に初期化するために使用されます。この変数とは、計算期間中に保留になっているインシデントのうち、各期間の最初にゼロに初期化する必要があるインシデントの数を保持するカウンタなどがあります。
- **OnPeriodEnd (Time,IsCompleteRecord)** : (任意)。(時間単位に従って設定される) エージェント期間の最後にコールされます。これは常に、概数化された時間単位の最後 (24:00 など) にコールされます。(アプリケーションサーバの実際の時間に従って) メトリックの期間が終了した場合、IsCompleteRecord は True になり、中間の計算を作成する場合は False になります。このイベント ハンドラは通常、Result 関数によって提供される最終的な結果の根拠を用意するために、終了期間に対する最後の計算で使用されます。
- **OnTimeslotEnter (Time)** : (任意)。関連付けられているメトリックの定義に基づいて、TimeSlot 期間に入ったときにコールされます。たとえば、メトリックが営業時間のタイムスロット定義に関連付けられており、毎朝午前 8 時がタイムスロットの開始である場合、毎朝午前 8 時にこのイベント ハンドラがトリガされます。
- **OnTimeSlotExit (Time)** : (任意)。関連付けられているメトリックの定義に基づいて、TimeSlot 期間を終了するときにコールされます。たとえば、メトリックが営業時間のタイムスロット定義に関連付けられており、毎日午後 5 時に終了する場合、この時間にイベント ハンドラがトリガされます。

- **Target()** : (任意)。メトリックが動的ターゲットを持つよう指定されている場合にコールされます。これにより、ビジネス ロジックの実行時に、メトリックのサービス レベル目標値を決定することができます。このようなターゲットには、サービス コンポーネントの消費やインフラストラクチャの変更が含まれます。これには **Result** 関数と同じように、各モードに対して 1 つずつ、合計 4 つの値があります。この関数は、通常の実行時に **Result** 関数の後で実行されます。
- **Forecast()** : (任意)。計算エンジンが、予測モードで契約をいったん計算することによって、契約バージョンのコミットを実行するときに一度コールされます。予測モードは、契約における完全な計算エンジン サイクル (契約バージョンの開始日から終了日まで) を実行しています。このサイクルの目的は単に予測値を計算することです。(T_RAW_DATA テーブルでは計算は行われません)。関数は、この実行モードのときに、**Result** 関数の代わりにコールされます。
- **Result()** : (必須)。特定の期間についての計算結果を取得するために、計算エンジンによってコールされます。**Result** 関数は常に、OnPeriodEnd イベントハンドラの後にコールされます。

単一のビジネス ロジック計算式を処理するための、計算メカニズム (PslWriter サービス) の後の手順は以下のとおりです。

- PslWriter は計算式をメモリにロードし、宣言セクションにあるすべてのステートメントを実行します (宣言セクションは、関数またはサブルーチンの外部にあるすべてのコードです)。割り当てられているすべてのモジュールおよびライブラリが含まれており、実行用のこのシングルコードモジュールにコンパイルされることに注意してください。
- PslWriter が OnLoad イベントハンドラをコールします。
- PslWriter が OnRegistration をコールします。
- OnPeriodStart のコールによって、期間の計算が開始されます。
- 期間の範囲内にある OnRegistration で登録されたそれぞれの Raw データ イベントについて、PslWriter は、そのイベントに関連付けられているユーザ定義のイベントハンドラをコールします。
- タイムスロットの開始または終了が対象期間の範囲内にある場合、OnTimeslotEnter または OnTimeSlotExit がコールされます。
- 対象期間の範囲内で関連するインフラストラクチャが変更された場合、OnRegistration がコールされます。
- OnPeriodEnd のコールによって、期間の計算が修了します。
- 動的ターゲットが指定されている場合は、この関数がコールされます。
- 期間計算の最終結果を取得するために、PslWriter が **Result** 関数をコールします。

注: 契約バージョンが最初にコミットされ、予測が選択されると、**Result** 関数の代わりに **Forecast** 関数がコールされます。ただし、これは各バージョンに対してこの 1 回だけ発生します。

エンジンはそれぞれの計算サイクルで、計算期間に基づいて、エンジン イベントおよび **Raw** データ イベントは何かを評価します。この評価は最初、時間で並べ替えられ、次に、計算のために計算式へ送信されます。**Raw** データ イベントの時間はそのイベントのタイムスタンプで、エンジン イベントの時間はそれがトリガされた時間です。これらの 2 つのタイプのイベントは時系列に組み合わせられ、計算用に送信されます。

イベントのタイミングは関連付けられているローカル メトリックによって決まりますが、イベント ハンドラの **Time** パラメータ (**OnPeriodStart (Time)** など)、および **Raw** データ イベントのタイムスタンプは **UTC** の値に従っています。エンジンは、参照で一定のポイントを使用できるように、**UTC** におけるそれぞれの値に従って比較を行います。

例:

あるメトリックで、**UTC** とのタイム ゾーンの差が 2 時間 (つまり **GMT+2**) の場合、インシデントの開始イベントのタイムスタンプが午前 10 時で、エンジンの中でイベント ハンドラが使用するタイムスタンプは適宜シフトされ、実際には **8 : 00am UTC** に開始される、ということがあります。アダプタがこれと同じタイム ゾーンを使用するよう設定されているとすると、**Raw** データ イベントも、データベース内で **UTC** に合わせて 2 時間前に戻されます。イベントがビジネス ロジックに渡されるとき、午前 10 時に始まる期間についてイベントの計算を行う計算エージェントは、実際にはそのイベントに対して **UTC** 時間を使用します (これは午前 8 時です)。ただし、タイムスタンプを印刷するためにコード内で **out.log** メッセージを使用している場合は、(メトリックに従って) 指定された期間が **10am** であっても、**UTC** にシフトしたタイムスタンプ、つまり **8:00 AM** が表示されます。

以下の計算の要件では、使用する前にイベントのタイムスタンプを変換することが重要です。

メトリックが、開始と終了が同じ日付であるようなインシデントの数を計算する場合、それぞれのインシデントの開始時間と終了時間を比較する必要があります。インシデントの開始時間と終了時間が同じ日 (および定義されているタイムスロット内の範囲内) になる場合、このインシデントはカウントされます。

インシデントを、元のローカル時間から UTC にシフトしている途中で、日付が変わってしまう（つまりもう一度 GMT+2 を使用する）ことがあります。午前 1 時に開始されたインシデントは、UTC では前日の午後 11 時にシフトされます。このようなインシデントは、（シフト前はカウントの対象であっても）カウントされません。これは、最初に時間をローカル メトリックの時間に戻して変換し、その後で単純に比較しなければならない場合です。このような場合には、**GetLocaleTime(utcTime)** メソッドを使用します。このメソッドは、UTC タイムゾーンで指定された時間を、ローカル メトリックのタイム ゾーンに変換します。

イベント ハンドラのコードは以下のとおりです。

```
Sub OnIncidentEvent(eventDetails)
    If dateDiff("d",Tools.GetLocaleTime(eventDetails.time),_
        Tools.GetLocaleTime(eventDetails("ClosedAt")))=0 then
        CountIncidents=CountIncidents+1
    End If
End Sub
```

登録

登録は、計算の一部となる **Raw** データ イベント セットについて、ビジネス ロジックが要求を計算エンジンへサブミットするプロセスです。

登録プロセスは 2 つの方法で処理できます。つまり、登録ウィザードを使用する方法と、ビジネス ロジック内でディスパッチャ オブジェクトを使用して手動で処理する方法です。

登録ウィザードを使用する方法は、使用できるオプションから選択する単純なプロセスです。パラメータを使用することはできませんが、登録を手動で行う場合と同じオプションをすべて使用できます。パラメータを使用する必要がある場合は、登録を手動で実行してください。ウィザードの基本的なフローでは、実行する登録のタイプを最初に決定し、次に登録を実行するリソース タイプおよびイベントを設定し、最後に、収集したイベントの処理でどのイベント ハンドラを使用するかを最後に決定します。

登録が完了すると、メトリックの [登録] タブに、登録した内容が表示されます。1 つのメトリックに対して複数の登録ステートメントを持つことも可能です。

実際には、登録ウィザードでは手動の登録と同じ機能を使用しています。これらのすべてのオプションについては、次のセクションで説明します。

ビジネス ロジック内で手動で登録を実行する場合、計算式の登録は **OnRegistration** イベント ハンドラによって処理されます。これは、計算式の中で実装され、登録エンジン イベントがトリガされるたびにトリガされるようにする必要があります。登録イベントは、契約がアクティブになったときに 1 回トリガされ、対象のリソースまたは変更セットがアクティブになるたびにトリガされます。リソースが、メトリックが受け取る予定のイベントに影響を与える場合は、影響を受けるリソースの変更が関連すると考えられます。たとえば、契約関係者 (**RegisterByContractParty**) によって登録が行なわれる場合は、メトリックの契約関係者に割り当てられているリソースを持つ、定義済みのタイプのすべてのイベントが計算の一部になることを意味します。このような場合、新しいリソースが割り当てられる、またはその契約関係者に対して割り当て解除されるたびに、変更のエンジンに通知するためにメソッドがトリガされます。

登録のメソッドは、**OnRegistration** に引数として渡される **Dispatcher** オブジェクトによって提供されます。いくつかのメソッドは、イベントタイプの定義に基づいて、フィルタリング基準、およびリソースの割り当て基準 (リソース グループのリソース、特定タイプのリソースなど) を定義するためのさまざまな方法を提供します。

契約関係者およびサービスの登録メソッドを使用すると、ビジネス ロジックをモジュールまたはテンプレートとして簡単に使用できるため、この登録メソッドを使用することが推奨されます。この場合、対象の契約関係者およびサービスは関連するメトリック定義から取得され、異なる契約やサービス コンポーネントに対して計算式を再利用する場合は、登録を変更する必要がありません。

もうひとつの一般的な登録メソッドは **RegisterByResourceGroup** です。これは論理的にグループ化されているけれども、必ずしも契約関係者またはサービスに関連付けられていないリソースを操作するうえで便利です。ここでは、グループに対するリソースの割り当てはリソース カタログによって（個別に、または変更セットを介して）管理され、変換スクリプトによって自動的に更新することもできます。

一般的に、登録メソッドは設計フェーズで決定され、定義されたデータ モデルによって直接駆動されます。

注: ディスパッチャ オブジェクトが正しく使用されているかどうかをチェックするために、**SLALOM** の構文チェックのときに **OnRegistration** 関数もコールされます。この理由により、**OnRegistration** 関数の前に **OnLoad** を実行することを前提にしないようにします。また、**"TimeUnit"**、**"IntervalLength"**、**"IsPeriod"**、**"CorrectionsApply"**、**"ExceptionsApply"** などのコンテキスト オブジェクトのいくつかのプロパティを **OnRegistration** イベントハンドラで使用してはいけません。

登録メソッドは、イベントのタイムスタンプに従ってトリガされるプロシージャに対してイベントを関連付けます。定義されたプロシージャはどのような名前にすることもできますが、常にパラメータとして **eventDetails** オブジェクトを持っています。 **eventDetails** オブジェクトは、受け取った **Raw** データ イベントを反映し、以下の登録で見られるように、すべてのイベントの詳細を保持します。

```
Sub OnRegistration(dispatcher)
dispatcher.RegisterByContractPartyAndService "OnMemUseEvent", "MemUse", "Server"
'the parameters of the method are: <procedure name>, <Event Type>, <Resource Type>
End Sub
```

上記の登録ステートメントは、**'MemUse'** イベント タイプのすべての **Raw** データ イベント、および **'Server'** リソース タイプに関連付けられているすべての **Raw** データ イベントが、ビジネス ロジック内で **'OnMemUseEvent'** イベント ハンドラに送信されることを示しています。

以下のプロシージャも、ビジネス ロジック内にあらかじめ定義しておく必要があります。

```
Sub OnMemUseEvent(eventDetails)
If InTimeSlot And eventDetails("MemoryUsage")>MaxMemoryUse Then
    MaxMemoryUse = eventDetails("MemoryUsage")
End If
End Sub
```


`eventDetails` オブジェクトを参照し、'`MemoryUsage`' パラメータを使用することによって、ステートメントは、関数に渡されたイベントから `MemoryUsage` フィールドの値を抽出します。これらのフィールドはイベント タイプで定義されたものと同じで、フィールド名は大文字/小文字が区別されます。

クラスタ化メトリックの登録

クラスタ化メトリックでは、リソース グループの個々のメンバに対して 1 つのメトリックの定義を実行して、一連のアイテムに同じ定義およびロジックを適用することができます。クラスタ化は、事前に定義されたリソース セットに静的に設定することもできますが、グループを長期にわたって変更し、グループに対してメンバを対象、または対象外にしたりして、リソース グループに動的に設定することもできます。

注: 詳細な説明については、付録 E「Defining Business Logic Formulas (Business Logic Expert)」を参照してください。

リソースのグループ内で各アイテムに対してサービス レベルの結果を計算する必要がある場合は、クラスタ化メトリックが使用されます。リソース グループのアイテムはリソースまたは他のリソース グループにすることができます。このため、クラスタ化メトリックのビジネス ロジック スクリプト内の登録メソッドは、**RegisterByResourceGroup** または **RegisterByResource** でなければなりません。ここで指定されるリソース名またはリソース グループ名は、クラスタ内でアイテムとして定義されます。これは、現行のクラスタ アイテムの名前を保持しているコンテキスト オブジェクトの 'ClusterItem' プロパティを使用して行われます。

例:

```
dispatcher.RegisterByResource "<ProcedureName>", "<Event Type name>",  
Context.ClusterItem
```

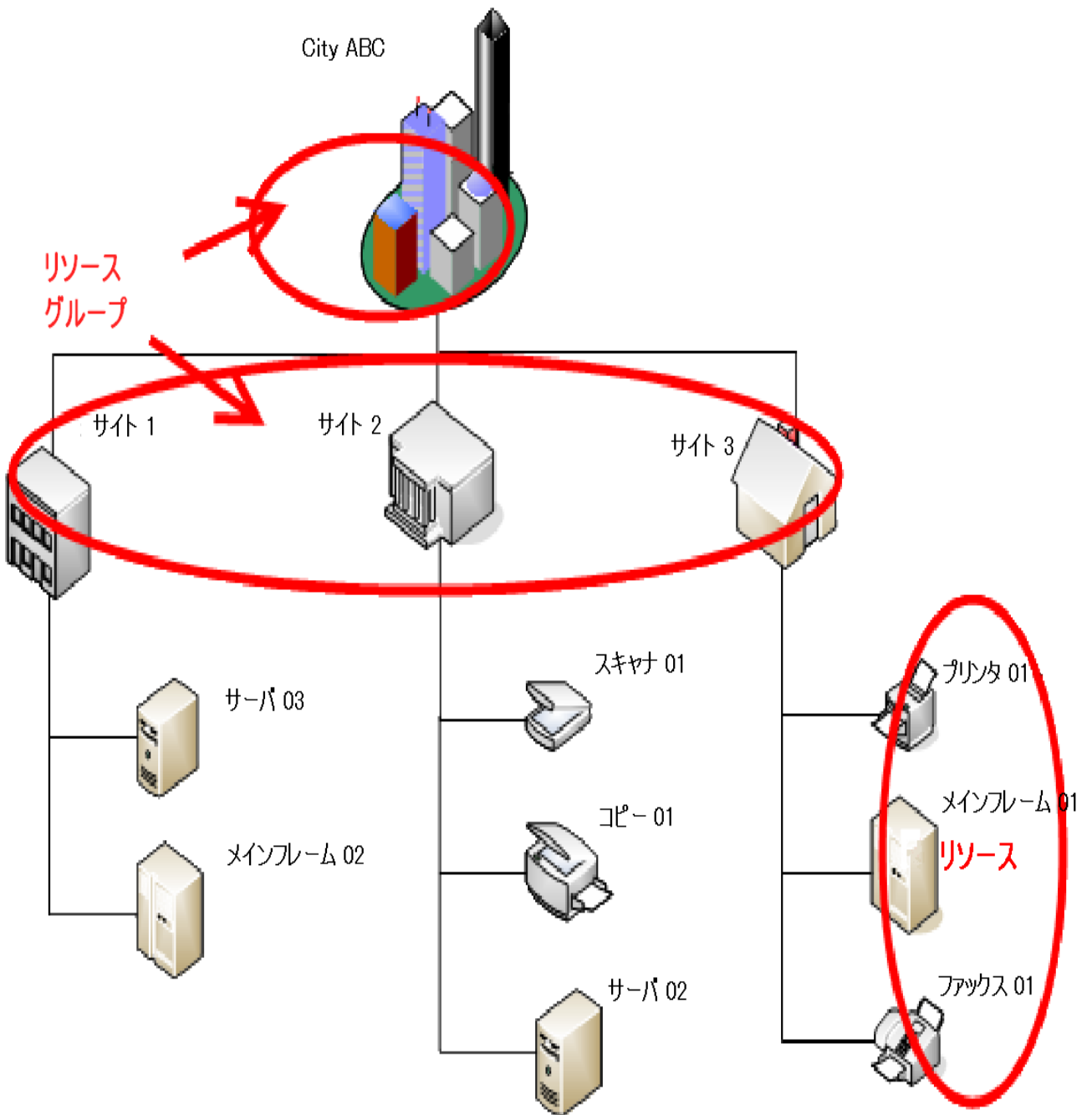
この登録メソッドが使用される場合、メトリックは、自身がクラスタ化されているリソース グループ内の各リソースについて結果を計算します。

または

```
dispatcher.RegisterByResourceGroup "<ProcedureName>", "<Event Type name>",  
Context.ClusterItem
```

この登録メソッドが使用される場合、メトリックは、自身がクラスタ化されているリソース グループに含まれている各リソース グループについて結果を計算します。

クラスタ化は、リソース モデルがどのように作成されているかによって、さまざまなレベルで生じることがあります。組織では、さまざまなグループ化のレイヤを表現したいことがよくあります。たとえば、ある都市に多数のサイトがあり、それぞれのサイトに多数のインフラストラクチャ装置が存在していることがあります。(プリンタ、スキャナ、サーバなど) このようなタイプのグループ化を使用すると、複数のレベル、およびこれらのハードウェア アイテムのグループ化が含まれている、定義済みのリソース階層を構造化することができます。インフラストラクチャ装置が「リソース」であると仮定すると、ここで説明した構造は次のように見ることができます。



この図からわかるように、複数のグループ レイヤがあります。最上位レベルの 'City ABC' グループには異なる 3 つのサイトが含まれています (これらのサイトも リソース グループです)。リソース グループ 'Site 3 Resources' には異なる 3 つのリソースが含まれています。したがって前の例から、異なる 3 つのサイト間でメトリックをクラスタ化するには、以下の登録を使用します。

```
dispatcher.RegisterByResourceGroup "<ProcedureName>", "<Event Type name>",
Context.ClusterItem
```

この場合、Context.ClusterItem は 'City ABC Sites' というリソース グループを参照しており、この中には 'Site 01 Resources'、'Site 2 Resources' などの他の 3 つのリソース グループが含まれています。これはメトリックの [クラスタ化] タブで次のように表示されます。

全般	詳細	メモ	クラスタ化	しきい値	関連メトリック
☐ このメトリックはクラスタ化されません。 ☒ このメトリックは以下のとおりクラスタ化されます: クラスタ化対象のアイテム: RG_RUI ☒ 動的 ☒ 第 1 レベルのみ: 直接メンバ ☐ すべてのレベル: リソースのみを含む ☐ すべてのレベル: リソースおよびリソース グループを含む ☐ 静的: 以下のアイテムのみが含まれます: 追加 削除					

グループで何らかの変更が発生した場合は、その変更が自動的にクラスタ化に含まれるため、クラスタ化は動的に設定されていることに注意してください。静的なクラスタ化は、リソース グループのサブセットに対して有用です。また、長期間クラスタ化を変更したくない場合にも有用です。

Site 3 グループのリソースについてレポートするメトリックを作成するには、以下の登録ステートメントを使用します。

```
dispatcher.RegisterByResource "<ProcedureName>", "<Event Type name>",  
Context.ClusterItem
```

ここで、Context.ClusterItem は個別のリソースを参照するため、リソースによってのみ登録します。メトリックの [クラスタ化] タブには、'Sites 03 Resources' グループへの参照が含まれています。

1 つのメトリック内の別の階層レベルで、クラスタ化を動作させるよう設定することができます。たとえば、前述の例で説明した状況を前提とし、'City ABC Sites' グループでこのメトリックをもう一度クラスタ化するとします。1 つのメトリック内に、階層の他のレベルのリソース メンバを含めることが可能です。ここでは、このグループ化にどのリソースを含めるかについて、以下の 3 つのオプションがあります。

- **第 1 レベルのみ: 直接メンバ**： 前述の古いクラスタ メソッドと同じです。
- **すべてのレベル: リソースのみを含む**： 1 つのレベルの 3 つのサイトのリソース グループに含まれているすべてのリソースを含みます。また、それぞれについてサービス レベルを計算します。
- **すべてのレベル: リソースおよびリソース グループを含む**： 3 つのサイトのリソース グループ、および 3 つのリソース グループに含まれているすべてのリソースを含みます。また、それぞれについてサービス レベルを計算します。

Agents

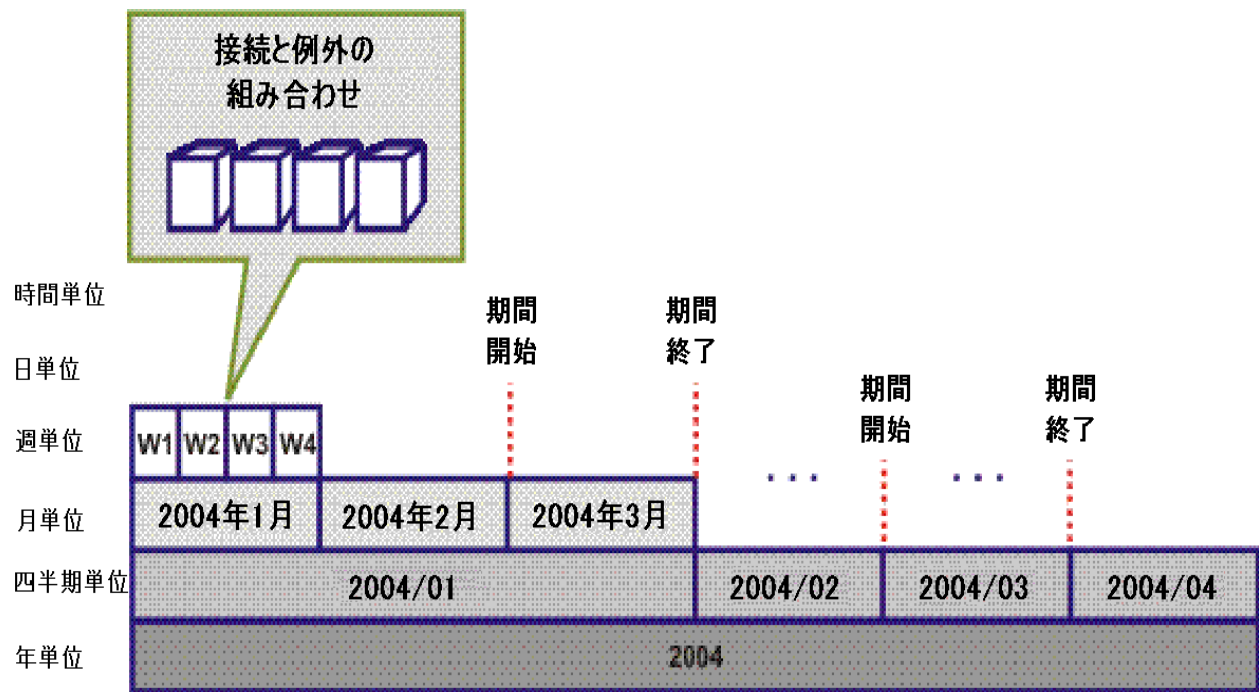
各メトリックには、トラッキング期間の定義が 1 つあります。トラッキング期間は、メトリックがサービス レベルの結果を出力しなければならない期間で、その結果は、定義されているターゲットと比較する必要があります。トラッキング期間に生成された結果はビジネス結果であり、これは言い換えると、特定のターゲットのサービス レベルを提供するためにプロバイダがコミットした契約上の結果です。各期間のビジネス ロジックのインスタンスはそれぞれ 1 つのエージェントと呼ばれるもので、各メトリックに関連付けられている粒度に直接関係しています。

たとえば、メトリックに 1 か月のトラッキング期間が定義されている場合、このメトリックは月単位の結果を提供するために実行されます。

月単位の結果について、さらに日単位の結果を参照できるようにするドリルダウン機能を提供するために、メトリックでは、追加の時間単位の定義が必要になります。この時間単位でメトリックを計算する必要があります。時間単位は、[粒度] タブの [メトリック] の [一般詳細] セクションで定義されます。

メトリックの [粒度] タブで、トラッキング期間に対して定義されたそれぞれの時間単位について、別のビジネス ロジック インスタンスがエンジンによって実行されます。これらの各インスタンスは同じビジネス ロジック コードを実行しますが、これらの各インスタンスに対して **OnPeriodStart** と **OnPeriodEnd** は別にトリガされます。日単位のインスタンスについては、1 日の始めと 1 日の終わりにトリガされます。各時間単位については、その時間単位の開始と終了のポイントに従ってトリガされます。

ビジネス ロジックの各インスタンスは、対象の時間単位の結果を生成するために実行されます。また、各期間では、あらゆる例外を考慮した結果を備えている必要があります。（例外が定義されている場合）、このような結果を保持していない期間は、ユーザが、考慮された例外あり、または例外なしのどちらでサービス レベルの結果を見るかを選択できるようにする必要があります。この要件があるために、各メトリックは、エンジンによって実行される 14 個のエージェント（インスタンス）を潜在的に備えています。2 つのエージェントはビジネス結果用で、12 個のエージェントは追加の 6 個の運用期間用です。



ビジネス ロジックのタイム スケール

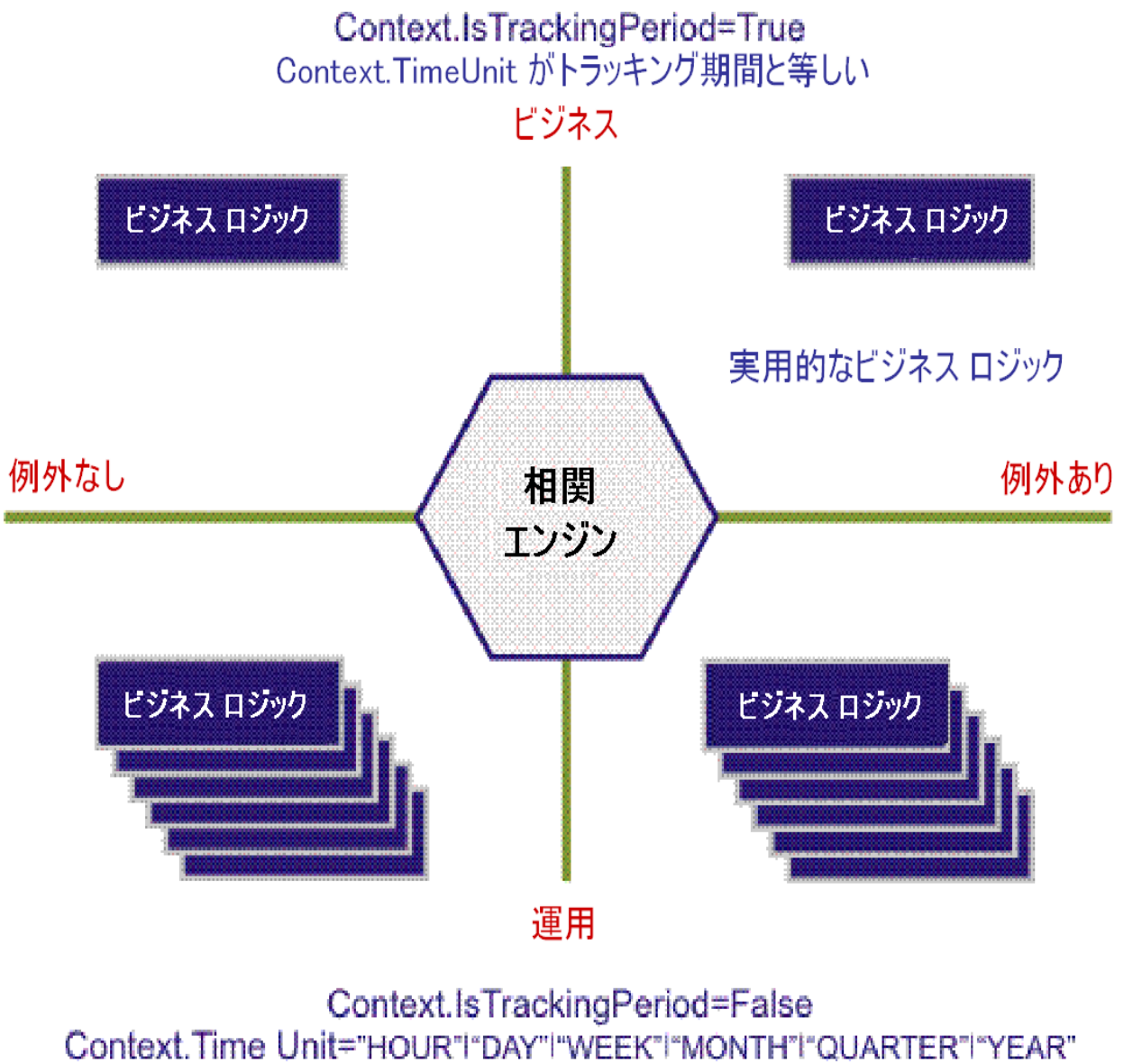
これは、別のエージェントに対してはそれぞれの期間が計算されるため、粒度の定義が計算エンジンのパフォーマンスに多大な影響を与えることを意味します。したがって、全体または部分的にドリル ダウン機能が必要でない場合には、いくつかのエージェントをオフにすることが推奨されます。これは、時間単位の結果のように、粒度が低い場合には特に大きな影響があります。また、エンジンは、発生したそれぞれのクラスタアイテムに対して上記のすべての計算を行うため、これはクラスタ化メトリックに対しても多大な影響があります。実際には、エンジンは各クラスタアイテムを新しいメトリックとして扱います。たとえば、50 のアイテムのリソース グループでメトリックを計算する場合には、同様の非クラスタ化メトリックに比べて、エンジンが行なう処理の 49 倍機能します。

たとえば、メトリックが数日間、解決時間を計算するよう設定された場合は、時間単位の結果は意味がありません。また、エンジンが不要な計算を実行しないように、粒度のタブでチェックを解除しておく必要があります。

Context オブジェクトの **TimeUnit** 属性（つまりビジネス ロジックの **context.TimeUnit**）は、実行中のエージェントの時間単位を返します。ここで返される値として可能性のあるものは、**HOURL**、**DAY**、**WEEK**、**MONTH**、**QUARTER**、**YEAR** です。

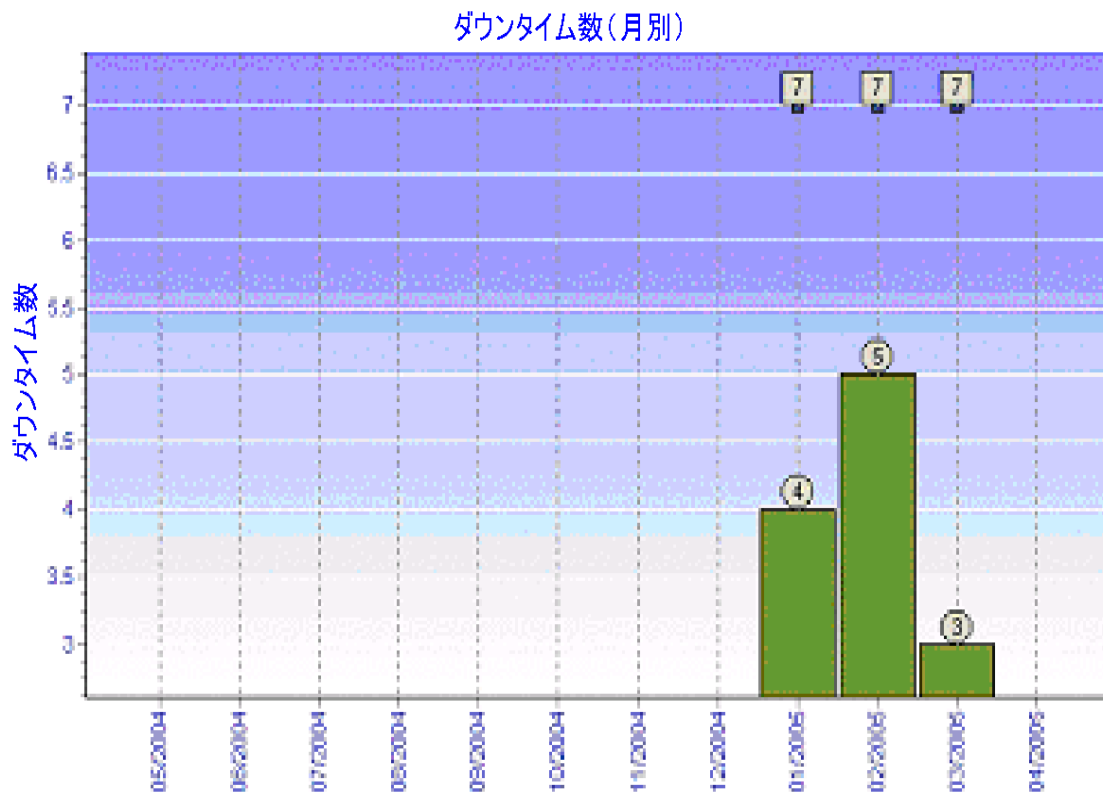
たとえば、日単位のエージェントの場合、**Context.TimeUnit** は **"DAY"** を返します。

Context オブジェクトの **IsTrackingPeriod** 属性は、トラッキング期間の時間単位を計算するエージェントに対して **True** を返します。また、メトリックの [粒度] タブに示されているエージェントは、メトリックのトラッキング期間のエージェントに付加されたものであることに注意してください。このため、月単位のトラッキング期間を持つメトリックに対しても、月単位のエージェントをオフにしたうえで、月単位のサービス レベルを（ビジネス結果として）計算することができます（非オペレーション結果）。

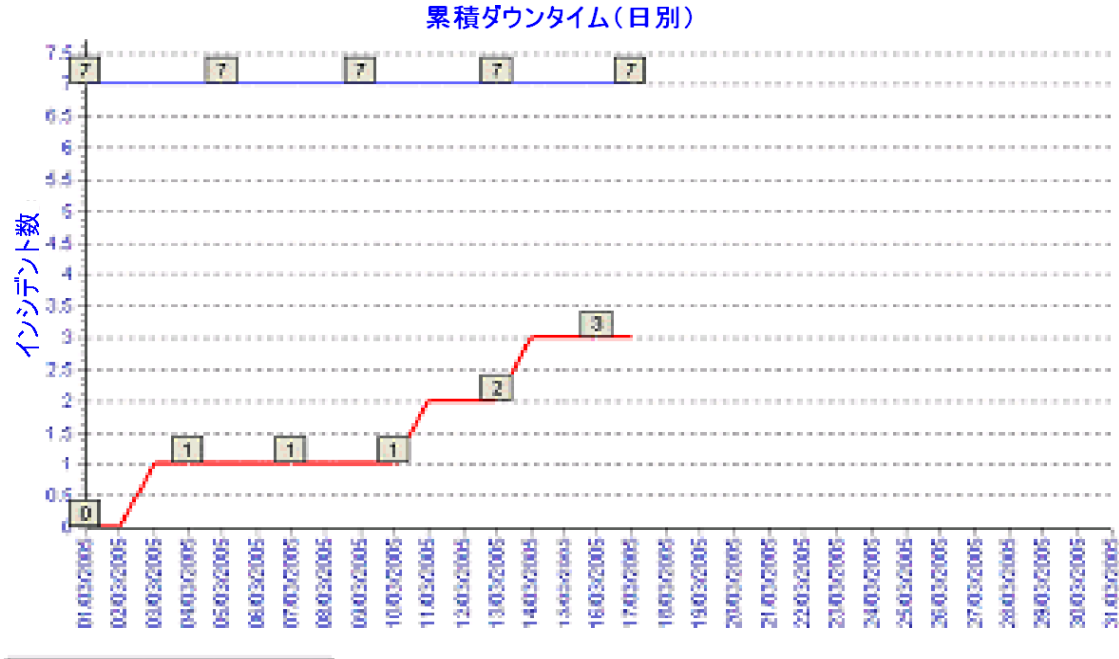


上記で説明するように、各種のエージェントは通常同じビジネス ロジック コードを実行しますが、わずかに異なるロジックを適用しなければならない場合があります。

たとえば、月単位の場合には、以下に示すように、結果として各月でそれぞれ何回か停止時間がありました。



日単位でドリルダウンする場合には、累積した停止時間を参照できるようにして、それぞれの日が、月の始めからの日数の合計になるようにする必要があります。このメソッドは、以下に示すように 1 月以下のすべての時間単位に適用されます。



2つの時間単位の違いは、停止時間が発生するトラッキング期間を計算しているエージェントは、各期間の最初に0に初期化されますが、日単位のエージェントでは、月の最初の日だけカウンタが初期化される、ということです。

OnPeriodStart イベント ハンドラは以下のとおりです。

```
Sub OnPeriodStart(time)
    If InStr("MONTH,QUARTER,YEAR", Context.TimeUnit) > 0 _
    Or (Context.TimeUnit = "DAY" And Day(time) = 1) _
    Or (Context.TimeUnit = "HOUR" And Day(time) = 1 And Hour(time) = 0) Then
        DownTimeCounter = 0
    End If
End Sub
```

再計算とは

関連付けエンジンが、メトリックの最終期間の結果が有効ではなく、結果を再計算しなければならないことが認識された場合に、再計算が実行されます。

再計算は以下の事項によって生じることがあります。

- 過去に、実際に発生した新しいイベントを受信した場合。（エンジンが、メトリックに対して最終計算をする前）
- メトリックに登録されているリソースが変更された（リソースに対して新しい日付と変更が適用された）場合。
- 新しい契約バージョンがコミットされたが、以前に計算した期間と重複しており、いくつかのメトリックに変更が加えられている場合（変更されたメトリックのみ再計算される）。

登録するための最も効率的なメソッドは、契約関係者およびサービスです。契約関係者およびサービスのリソースを整理すると、システムでのデータレイヤとビジネスレイヤの間の論理関係を表現することができます。これらのエンティティを介してリソースを登録すると、別の契約で使用されている場合、または別のサービスに対して使用されている場合でも計算式に対する変更が不要になります。メトリックの現在のコンテキストは契約およびサービスを特定しましたが、これは対象の契約関係者、および関連するサービスを定義します。このタイプの登録で定義されているビジネスロジックの計算式は、登録において変更が必要ないため、簡単に再利用できます。

出力 - ユーザ テーブル

標準のビジネス ロジック スクリプトは、外部の出力テーブルに対するアクセス権がありません。出力先は以下の 2 つだけになります。

- **PSL テーブル (T_PSL)**。エンジンはこのテーブルに対してサービス レベルの結果を自動的に書き込みます。また、結果の関数で指定された結果は、このテーブルに書き込まれます。このテーブルに書き込まれるサービス レベルの値は、数値タイプのみです。T_PSL テーブルに書き込まれる結果は、レポート ウィザードがレポートする結果です。これらの結果が書き込まれる構造またはメソッドについて制御することはできません。これは、関連付けエンジンによって自動的に行われます。
- **SLALOM 出力テーブル (T_SLALOM_OUTPUTS)**。このテーブルへの書き込みは、ビジネス ロジック 計算式の Tools オブジェクトによって提供されるメソッドを使用して行われます。このテーブルへ出力する場合、ユーザ テーブルとして参照される論理テーブル名が提供されます。これらのテーブルを使用して、サービスレベル計算の際に情報を出力します。後でこの情報を使用して、自由形式のレポートを生成することができます。多くのユーザ テーブルが存在する場合があります。

定期的なサービス レベル結果のほかに追加の出力が必要な場合、別のメトリックを追加しても、これ以上の出力が得られない場合、または別のメトリックを追加すると、同じレコードセットを通過して異なる出力を提供するだけで、計算のパフォーマンスが低下する場合には、T_SLALOM_OUTPUTS 外部テーブルの使用が必要になります。

たとえば、1 日以内に解決されたチケットの割合を計算するようメトリックが設定されており、1 日以内に解決できなかったすべてのチケット リストを示したレポートを作成する必要がある場合について考えてみると、計算式で、外部テーブルに対する障害とみなされた各チケットを出力し、それを計算の統計に追加する必要があります。

前述の要件では、標準の出力サービス レベル テーブルはこの出力を提供できません。理由は以下のとおりです。

- サービス結果はすべて数値である
- 各期間に対しては、単一のサービス レベル結果しか使用できない

メトリックのトラッキング期間に実行されているエージェント、および例外と修正を計算しているエージェントに対してのみ、ユーザの出力テーブルにレコードが書き込まれます。たとえば、月単位のトラッキング期間を持つようメトリックが定義されている場合、エンジンが他の粒度 (HOUR、DAY、WEEK、QUARTER、YEAR など) について計算式を実行しても、出力ステートメント (Tools.SaveRecord and Tools.SaveFields) は実行されません。

このテーブルを外部で保持していることによる他のメリットは、複数のレポート要件で使用できることです。他の典型的なレポート要件は、これらのテーブル、および契約要件から生成することができます。たとえば、1 か月に「クローズしたチケット数」を計算するメトリックは、チケットの解決時間を計算することも、この情報をすべて出力テーブルに出力することも可能です。これにより、期間中にクローズした個々のチケットについて、より詳細な分析が可能になり、別のレポート要件で必要とされる可能性のある、より詳細な情報が得られます。

これらのテーブルの情報も、エンジンの再計算メカニズムによって管理されます。この再計算プロセス中に、レコードは非アクティブとしてマークされ、代わりに新しい行が書き込まれます。すべての自由形式レポートの SQL ステートメントでは、T_SLALOM_OUTPUTS テーブルの IS_ACTIVE フィールドについてのチェック (is_active = 1) を含める必要があるため、これは重要なポイントです。これらのレコードのみが最新であるためです。

注: 計算式のデバッグプロセス中にビジネス ロジック スコープを実行すると、メッセージは、T_SLALOM_OUTPUTS テーブルの代わりに、T_DEBUG_SLALOM_OUTPUTS テーブルに書き込まれます。

T_SLALOM_OUTPUTS を使用してデータを文書化する場合、挿入されるデータは常にテキストです (T_SLALOM_OUTPUTS のフィールドがすべて varchar2 であるため)。したがって、オペレーティング システムのフォーマットを適用することにより、日付の値はテキストに変換されます。オペレーティング システムのフォーマットは、アプリケーションのライフサイクル中に変わることがあります。これにより、T_SLALOM_OUTPUTS は日付のフォーマットの整合性が保持できないことがあります。また、ビジネス ロジックは UTC の日付を処理しますが、あるユーザは T_SLALOM_OUTPUTS でローカル タイムスタンプが保持されることを期待しています。このため、このような状況に対応するために、変換関数 *Tools.GetLocaleDate(date)* の使用が必要になることがあります。

以下の関数は、日付をローカル時間に変換し、日付を dd/mm/yyyy hh24:mi:ss 形式にフォーマットすることによって、日付フォーマットの整合性を保持します。

```
Function FormatDate(time)
    Dim LocalTime
    LocalTime=Tools.GetLocaleTime(time)
    FormatDate=Day(LocalTime) & "/" & Month(LocalTime) & "/" &
    Year(LocalTime) & " " & _
    Hour(LocalTime) & ":" & Minute(LocalTime) & ":" & Second(LocalTime)
End Function
```

ビジネス ロジック 計算式から外部テーブルへ書き込むには、以下の 2 つのメソッドを使用できます。

- Tools.SaveRecord <parameter list>
- Tools.SaveFields <parameter list>

Tools オブジェクトのこれらのメソッドについては、以下で詳しく説明します。

Tools.SaveRecord tableName, key,[val1],[val2],...

このメソッドは、**T_SLALOM_OUTPUTS** というテーブルにレコードを保存します。**tableName** パラメータは、**T_SLALOM_OUTPUTS** 内の（仮想）テーブルを指定します。このテーブルに情報が書き込まれます。ユーザテーブルの各レコードには、情報の書き込み先であるレコードを指定する一意のキーがあります。また、各レコードには、最大 20 文字の文字列タイプの値フィールドがあります。**SaveRecord** メソッドは、ユーザテーブル名とキーを受信します。また、ユーザテーブル内のすべての値フィールドを承認します（これらの値パラメータはオプションで、省略される場合があります）。同じキーを持つレコードがすでに存在している場合、そのレコードは更新されます（パラメータとして転送される値フィールドのみが更新されます）。このキーを持つレコードが存在しない場合は、レコードが作成されます。

```
Tools.SaveFields tableName, key, [fieldName1,fieldVal1], [fieldName2,fieldVal2]
```

このメソッドは **SaveRecord** に似ていますが、すべての値を列挙する代わりに、フィールド名と関連するフィールド値のペアを指定する点が異なります。フィールド名の代わりにフィールド数を使用することができます。フィールド名は、**T_SO_FIELD_NAMES** テーブルの中にあらかじめ手動で定義しておく必要があります。このテーブルは、出力テーブルの構造を記録するために使用します。

T_SLALOM_OUTPUTS へ書き込む前に、ビジネス ロジック エキスパートで出力テーブルの構造を定義しておくことをお勧めします。これは、この時点で構造およびフィールドの意味を明確にして、クエリをより簡潔にするためです。

テーブルは次のようになります。

- **table_name** : 各論理テーブルには一意の名前があります。
- **field_name** : テーブル内の各フィールドは一意です。
- **field_id** : 各フィールドは、1 から始まるシリアル番号でナンバリングされています。

SaveFields メソッドは、挿入された値の構造および意味について文書を保持するため、このメソッドを使用することが推奨されます。

例:

メトリックの結果によって、しきい値以内の時間で解決されたチケットの割合を計算するだけでなく、解決時間が定義されたしきい値を超えてしまったすべてのインシデントのリストを生成する必要がある場合について考えてみます。出力テーブルへの書き込みは、しきい値の検証の後で、**OnXXXEvent** イベント ハンドラ プロシージャで行われます。

これは、以下の例のように示されます。

```
Sub OnIncidentEvent(eventDetails)
```

```
    If eventDetails("RESOLUTION_TIME")<=SThreshold Then
        CountSuccessIncident s= CountSuccessIncidents+1
```

```

Else
building the record unique key
    KeyString= eventDetails("IncidentID") &eventDetails.Time

    Tools.SaveFields "IncidentsTable", KeyString, "CASE_ID",
eventDetails("CASE_ID"),_
    "CUSTOMER_REF",eventDetails("CUSTOMER_REF"),_
    "TICKET_CREATOR",eventDetails("TICKET_CREATOR"),_
    "CREATION_TIME",eventDetails("CREATION_TIME"),_
    "SEVERITY",eventDetails("SEVERITY"),_
    " RESOLUTION_TIME ",eventDetails("RESOLUTION_TIME "),_
    "CLOSE_TIME",eventDetails("CLOSE_TIME")
End If
End Sub

```

T_SLALOM_OUTPUTS テーブルの使用に関して、いくつかの提案を示します。

- 出力テーブルには、必要なデータのみを書き込むことをお勧めします（それ以上のものは書き込みません）。
- 必要なメトリックの粒度についてのみ出力テーブルへ書き込みます。
- 値フィールドのデフォルト サイズは 50 文字です（ただし列 VAL_1 は例外で 512 文字です）。フィールドで 50 文字を超える値は、このデータ サイズに合わせて切り捨てする必要があります。このようにしないと、ビジネス ロジックで実行時エラーが表示されます。
- データに利用可能な列は 20 個の列（VAL_1... VAL_20）のみです。

注: テーブルへの書き込みではメモリ内での計算に比べて計算量が非常に多いため、出力テーブルへの書き込みがエンジンのパフォーマンスに影響を与える可能性があります。このため、この機能の使用が必要かどうかと、この使用がどの部分で必要かを入念に検討して、このテーブルがアクセスされる回数を最小限に抑えます。

ユーザ定義テーブルからの情報に関するレポート

標準の CA Business Service Insight のレポート ウィザードは、出力テーブルに書き込まれた情報についてレポートを作成するためには使用できません。この情報に関するレポートを取得するには、自由形式レポートを作成する必要があります。これは、実質的にはこのテーブル上に SQL クエリを作成することを意味しています。このテーブルには各種計算式によって書き込まれる多数の論理テーブルが含まれているため、SQL に精通している担当者（データ ソース エキスパート）が T_SLALOM_OUTPUTS に対するビューを作成して、このテーブル内に含まれる各論理テーブルを区別したり、情報が計画どおりに取得されることを確認したりすることを容易にすることをお勧めします。

ビュー定義には、関連の情報タイプに対する文字列フィールドタイプのキャスト機能がすでにすべてあり、また必要なフィルタリング（論理テーブル、アクティブレコード、関連するメトリックなど）も保持されています。

ビューの作成例を以下に示します。

```
create or replace view kpi_view as
select
    to_date(t...) as fieldName,
    to_number(t..), ...
from t_slalom_outputs t,
    t_rules r,
    t_sla_versions sv,
    t_slas s,
where table_name = 'TableName'
    and is_active = 1
    and t.rule_id = r.psl_rule_id
    and r.sla_version_id = sv.sla_version_id
    and sv.sla_id = s.sla_id
    and sv.status = 'EFFECTIVE'
```

ビジネス ロジック モジュールの作成

ユーザは、複数のサービス レベル目標 (メトリック) による使用が可能な、独立したビジネス ロジック モジュールを定義できます。ビジネス ロジックに対してシステム全体にわたる変更を適用するために、ユーザは「ベース」ロジック コンポーネントを変更します。このコンポーネントは、この後、1回のクリックで関連する SLA すべてに適用できます。

ビジネス ロジック モジュールとは、内部でビジネス ロジックの定義と容易なメンテナンスが可能で、また冗長性も低減される 1つのコード コンポーネントです。単一のモジュールは複数のメトリックで使用できます。

設定段階中に、計算式はメインのビジネス ロジック モジュールを定義するように設定されます (「設計」の章の「ビジネス ロジック テンプレートおよびモジュール」を参照してください)。

ビジネス ロジック モジュールには、以下の 3 種類があります。

- **[フル ビジネス ロジック]** : 計算式全体を 1つのモジュールとして使用する必要がある場合に使用します。 **Result** メソッドと **OnRegistration** メソッドは、**Module** スクリプト内に実装する必要があります。
- **[クラス]** モジュール : 単一の VB クラスの定義が含まれています。
- **[ライブラリ]** モジュール : プロシージャ、関数、およびクラスのすべてのコレクションが含まれます。

各モジュールは、以下のうちのいずれかの内部から使用できます。

- メトリック
- 他のモジュール
- ビジネス ロジック テンプレート
- 契約テンプレートおよびサービス レベル テンプレート内のメトリック。

モジュールは、メトリックのコンテキスト **Parameters("ParamName")** から駆動されるパラメータを使用できます。

注: ランタイム エラーを回避するには、[ビジネス ロジック モジュール] 内のパラメータの使用時にデフォルト値を常に設定します。計算式により、存在しないパラメータに対してログのエラー メッセージが出力されます。

```
If Parameters.IsExist("ReportedDowntimesNum") Then
    maxBufferSize = Parameters("ReportedDowntimesNum")
Else
    maxBufferSize = 3
    out.log "ReportedDowntimesNum parameter not set", "E"
End If
```

ビジネス ロジック モジュールの例

「しきい値内のヘルプデスク アクティビティ」と表現されたサービス レベル目標があります。以下に説明するヘルプデスク システムには、次のステータスを持つチケット ライフサイクルがあります。

- Open
- Assigned
- Closed

ヘルプデスクのパフォーマンスを説明するために定義できるメトリックのうち、2つを以下に示します。

- [メトリック名] - 時間内のチケットの解決に成功。
[目標ステートメント]- チケットの 99% 以上を 4 時間以内に解決する必要がある。
[ビジネス ロジック]- 解決は Open から Closed までの間で計算する必要がある。
- [メトリック名] - 時間内のチケットの割り当てに成功。
[目標ステートメント] - チケットの 99% 以上を 30 分以内に割り当てる必要がある。
[ビジネス ロジック] - 割り当ては、Open から Assigned までの間で計算する必要がある。

これらのメトリックの両方で同じロジックを識別できるため、モジュールを両方のメトリックの要求に応じるように作成できます。

モジュールには、メトリック コンテキスト内に以下のパラメータが必要です。

- 第 1 ステータス
- 第 2 ステータス
- しきい値

ビジネス ロジック モジュールを作成すると、定義済みのメトリックでは、あるモジュールをその定義の一部として使用することができます。

次に、ロジックは変更することができます。たとえば、新規ステータスの「保留中のお客様」を考慮する必要があるとします。「保留中のお客様」は、ヘルプデスクがお客様からのチケットに関する追加情報を待っている状態のときに、チケットに設定されます。ビジネス ロジック内では、チケットが「保留中のお客様」の状態にある間、このチケットを計算対象の一部と見なすべきではありません。したがって、ビジネス ロジック モジュールのみをこの新規ステータスとロジックを考慮するように変更する必要があります。この新規ロジックが組み込まれた新しいバージョンのモジュールが作成されます。

変更をコミットする際、そのモジュールを使用するメトリックのリストが表示されます。変更は、すべてのメトリックに一括して適用でき、またリスト内の特定のメトリックのみに変更を適用することを選択することもできます。

リストから特定のメトリックのみを選択した場合、選択したメトリック用の新しいモジュールを作成するように求められます。選択したメトリックによって使用される古いモジュールは、新規ビジネス ロジック モジュールと置換されます。また、新しいロジックを使用して再計算が実行されます。

パラメータの作成

パラメータは、ビジネス ユーザに計算式のスクリプトを編集することなく、グラフィカル ユーザ インターフェース (GUI) を使用してそれらの値を表示して変更する、迅速で直観的な方法を提供します。

ビジネス ロジック内でのパラメータの使用により、システム全体で広く利用できる一般的な計算式の作成が可能になり、またモジュールの使用が最適化されます。

パラメータは、契約またはメトリックのレベルで定義できます。メトリックレベルのパラメータは、[メトリック詳細] の [目標ステートメント] タブに表示され設定されます。ビジネス ロジックは、メトリックレベルのパラメータにのみアクセスできます。このため、メトリック内から契約パラメータにアクセスするために、別のタイプのパラメータがメトリック内のローカルに作成されます。このパラメータは動的パラメータと呼ばれます。また、これは契約レベルのパラメータからの参照としてその値を取ります。動的パラメータで許可される参照値は、メトリックの親契約内で定義された参照値のみです。

パラメータ タイプ

- **Date** - 日付形式 (Date + Time)。
- **Number** - 最大サイズ 15 桁 (最大有効桁数 5 桁)。
- **Text** - 最大サイズ 100 文字。
- **Table** - 最大サイズ 120 エントリ。

計算式のコードからのパラメータの値にアクセスするには、**Parameters** オブジェクトを使用して、パラメータ名を参照する必要があります。

例:

`Parameters("Threshold")`

(これは値を呼び出す簡略的な方法であることに注意してください。通常、これは `Parameters.Item("Threshold")` として行なわれます。)

また、テーブルタイプのパラメータは以下のとおりです。

`Parameters("Table")("Word")("Price")`

(ここで、**"Word"** と **"Price"** の値は、それぞれテーブル化されたパラメータの行名と列名です)

テーブルパラメータは、以下のキーポイントに従ってのみ使用してください。

1. グローバル変数 (たとえば `dim myTableParam`) を定義します
2. 関数の **OnLoad** では、パラメータオブジェクトからの変数を設定します (たとえば `Set myTableParam = Parameters("MyParametersTable")`)。

- この後は、作成されたオブジェクトの **myTableParam** のみを使用します。このパラメータ自体は、**OnLoad** 関数の外部で使用しないでください。また、このパラメータから作成したグローバル変数オブジェクトのみを参照してください。
(たとえば「`dim myVal: myVal = myTableParam ("myRow")("myColumn")`」)。

これにより、パラメータで使用されたメモリが解放されます。またエンジンで各パラメータの呼び出し時、および「**OnXXXEvent**」(これは 1 つのメトリックにつき何千回も呼び出される可能性があります。) などのパラメータのマッピング作成が防止されます。

以下のコードでは、テーブル パラメータの正しい使い方を示しています。

Option Explicit

Dim sum

Dim myParamTable

Sub OnLoad(TIME)

Set myParamTable = Parameters("MyTableParam")

End Sub

Sub OnRegistration(dispatcher)

dispatcher.RegisterByResource " OnEvent", "EventType", "ResourceType"

End Sub

Sub OnPeriodStart(TIME)

sum = 0

End Sub

Sub OnEvent(eventDetails)

If Context.IsWithinTimeslot Then

sum = sum + 1 * myParamTable("firstRow")("secondColumn")

End If

End Sub

Function Result

Result = (sum * myParamTable("secondRow")("thirdColumn"))

End Function

以下のメソッドは、コード内で作成されたパラメータ オブジェクトごとに利用可能です。

パラメータ

説明

IsExist

param が存在するか。契約パラメータでは機能しません。

IsNumeric

Number タイプの param である。

IsText	Text タイプの param である。
IsDate	Date タイプの param である。
IsTable	Table タイプの param である。
IsEntryExist	Table 内にエントリが存在するか。
IsEntryNumeric	Table 内のエントリが Numeric タイプである。
IsEntryText	Table 内のエントリが Text タイプである。
IsEntryDate	Table 内のエントリが Date タイプである。
Dump	すべてのパラメータのリストを返す。
Item	パラメータの値を参照する。

動的ターゲットの実装

動的ターゲットは、標準的なビジネス ロジックのスクリプト内のイベントハンドラを使用して、ビジネス ロジックによって処理されます。動的ターゲットは、メトリックからサービス レベルの値を返すために使用される `Result()` 関数に似ています。動的ターゲットは、以下に示すように [メトリック] の [詳細] タブ上で指定する必要があります。

全般	詳細	メモ	クラス名	しきい値	関連メトリック	拉度	目標スタート/ストップ	登録
メイン インジケータ:								
サービス:								
サービス ドメイン: *	Service Availability							
ドメイン カテゴリ: *	% of time available							
測定単位: *	Percent							
タイムスロット: *	Business Hours (Weekly)							
タイムゾーン: *	GMT - Sun							
サービス レベル目標値: 目標値 - 次の値以下								
トラッキング期間: *	1 週							
測定ステータス: *	アクティブ							

動的ターゲット: ☒

動的ターゲットを指定すると、目標は [メトリック] の [詳細] タブで指定された静的な値ではなく、ビジネス ロジック内の **Target()** 関数から取得されます。

Target 関数は以下ようになります。

```
Function Target
  'TODO: ここにコードを追加して動的ターゲット計算を処理する
  Target = Null
End Function
```

この関数は、ある特定の期間に必要な目標値を返すようにメトリックの要件に基づいて実装する必要があります。この関数は、ビジネス ロジックで期間に割り当てることができるすべての数値を返すことができます。

動的ターゲットの実例

コール センターの場合、「*Avg Call Pickup time (平均の受話器を取るまでの時間)*」を測定するメトリックの目標は、受ける電話の量に依存する場合があります。たとえば、0 から 800 回までの電話がかかる場合目標は 15 秒未満、801 から 1500 回までの場合目標は 20 秒未満、1500 回以上の場合目標は 25 秒未満である必要があります。この場合、以下のように実装できます (**TotalCalls** が受けた呼び出しイベントごとに 1 ずつ増やされるカウンタです。また **TotalCalls** は 0 未満にはならないと想定します)

```
Function Target
  If TotalCalls >0 and TotalCalls <= 800 Then
    Target = 15
  ElseIf Total Calls > 800 and TotalCalls <= 1500 Then
    Target = 20
  Else
    Target = 25
  End If
End Function
```

動的ターゲットの使用法の別の例

メトリックの目標が計算の粒度に応じて変わる場合がある状況を考えてみてください。このような状況としては、一群のサーバに対する可用性の日単位の目標が 98% だが、月単位の目標は 99.5% であるというような場合があります。この状況に対する解決策では、**Context.TimeUnit** の関数呼び出しと共に動的ターゲット関数を使用して、計算中の現在のエージェントを判定することが必要になります。したがって、目標をそれに応じて調整できます。

```
Function Target
  If Context.TimeUnit = "DAY" Then
    Target = 98
  ElseIf
    Target = 99.5
  End If
End Function
```


状態のバックアップ

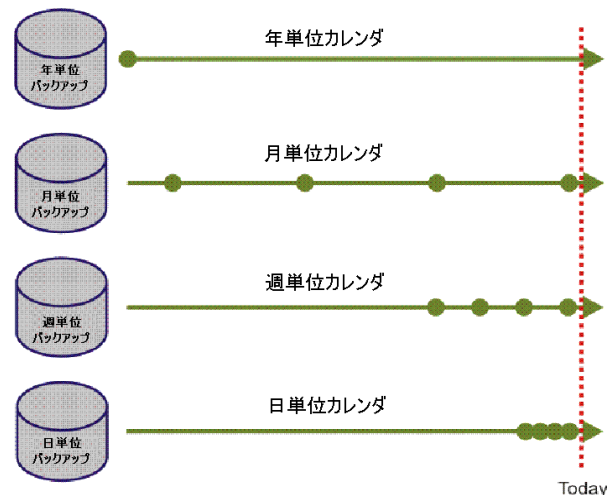
メトリックごとのサービス レベルを計算する進行中の処理中に、エンジンがまだ完了していない期間の部分的な計算の実行を強制されることがよくあります。新しいデータがやがて到着した際に、エンジンが計算の開始時点に戻る必要がないようにするために、エンジンでは、その次の計算タスクに移る前に、その現在の「状態」のある種のバックアップを行います。現時点では、エンジンにより計算でのその時点の現在の変数および値のスナップショットが作成され、その「状態」がデータベースに保存されます。

ビジネス ロジックのバックアップ処理とは、変数の値を含むビジネス ロジックのコードが、バイナリ ストリームにエンコードされて、データベース内に保存されるメカニズムです。このメカニズムは、再計算の場合に計算エンジンのパフォーマンスを向上させるためにも必要です。この状態は随時バックアップされ、再計算で計算を続行するための効率的な手段として使用されます。

たとえば、再計算が 1 か月間さかのぼることが必要な場合、契約の開始時点からではなく、再計算日付よりも前の最も近いバックアップの状態からの結果の再計算が使用され、その計算はその状態から以降で実行されます。

計算エンジンでは、事前定義済みのヒューリスティックス（発見的/試行錯誤的手法）を使用してバックアップがいつ必要かを判断し、またバックアップ機能を使用してエンコードされた状態をデータベースに格納します。

以下の図では、赤い点がある状態のバックアップを表しています。考慮に入れる期間の開始時点が前にさかのぼるほど、考慮されるバックアップ状態の数は少なくなります。このメカニズムの背後にあるロジックは、再計算が通常 1 月前よりも近い期間に必要とされるという想定です。



システムを再計算用に最適化する

サービス レベルの計算処理には、CPU リソース、メモリ、およびストレージが大量に必要です。マシンの負荷を軽減でき、また計算速度を向上させることができる推奨事項のリストを以下に示します。

注: これらの推奨事項の一部では、ユーザ インターフェイスで利用できない内部設定が必要になります。このような場合、詳細と手順について、CA テクニカル サポートにお問い合わせください。

■ 状態の保存設定を変更する

アダプタの既知の遅延に合わせて、状態パラメータを要件により適するように設定できます。つまり、状態の保存時点の回数と粒度を変更することができます。

■ システムを本当に必要な時間単位のみを計算するように設定する

メトリックには、7つまでの異なる粒度の期間、つまり年、四半期、月、週、日、時間、およびトラッキング期間を指定できます。一部の実装では、不要な粒度もあります。コミットされていない契約用の不要な粒度は、ユーザ インターフェイスを介してオフにできます。各メトリックの粒度用のタブを参照してください。コミットされた契約メトリックの粒度の変更については、CA テクニカル サポートにお問い合わせください。

注: トラッキング期間とほぼ同じ運用上の期間の計算は行なわないでください。

■ 「修正なし」、「例外なし」の値を計算しない

デフォルトでは、計算エンジンでは4つの異なる値、つまり、指定した値、修正のある指定した値、例外のある指定した値、または修正および例外のある指定した値を計算します。これは、指定した値のみを計算するように変更できます。

注: 詳細については、CA テクニカル サポートまでお問い合わせください。

■ 計算の順序を変更する

計算をより重要なメトリックから開始して、その後他のメトリックを続行するために、メトリックに対する PSL 計算の順序を優先順位付けすることができます。

注: 詳細については、CA テクニカル サポートまでお問い合わせください。

■ 複数の PslWriter インスタンスを作成する

複数の PSL インスタンス（計算エンジンの）を作成することによって、ワークロードを分割し、計算速度を向上させることができます。

注: 詳細については、CA テクニカル サポートまでお問い合わせください。

■ 再計算時間を短縮するための実装を計画する

再計算時間を最適化するために、次のことを行なえます。

- イベントの遅延を減らして、エンジンが処理するデータのバックログが大量にならないようにするために、アダプタをより頻繁に実行する
- [メトリック] の [粒度] タブで未使用のエージェントをオフにする。
- メトリックを複製し、同じメトリックを使用して別のエージェントを計算して、計算の負荷のバランスを取る
- 中間のメトリックを使用して共通の計算を実行し、同じデータが必要な他のすべてのメトリックとその結果を共有する。

■ **データ量を減らすための実装を計画する**

データの量を最適化するには、アダプタを使用して集約済み (処理済み) データのみを読み込みます。データ ソース情報を集約してから **CA Business Service Insight** の Raw データ テーブルへ送信することにより、PSL の入力読み取り効率が向上します。

■ **CA Business Service Insight の PSL 設定の推奨事項に従う**

PSL エンジンは、特定のアプリケーションの環境と要件に従って、パフォーマンスをより向上させるように再設定することができます。パラメータの中には、ユーザ インターフェースから設定できるものや、データベース システムの設定テーブルからしか設定できないものがあります。

注: PSL 設定に対するサポートの推奨事項を調べてください。

ログおよびアラート

ログに報告するか、またはアラート メッセージをトリガするために、ビジネス ロジックが必要になる場合があります。これは、メッセージをイベント処理に基づいて送信すべき場合に必要になります。計算の処理中に収集され、役立つ可能性がある情報は、すべてアラートとして送信することができます。たとえば、特定のイベントが指定された解決時間のしきい値の条件のもとにあるか、またはトレンド分析で連続した失敗が特定の回数に達した場合に、アラート メッセージを送信できます。

「Out」とは、計算式でログ メッセージと同様にアラートも送信できるグローバルのビジネス ロジックのオブジェクトです。このオブジェクトには、以下の各形式の関連付けられた 2 つのメソッドがあります。

Alert(<イベント タイプ>, <リソース名>, <値 1, 値 2>, ...<値 16>)

このコマンドでは、指定されたイベント タイプのアラートが送信されます。ただし、このアラートのためには、そのイベント タイプを手動で作成する必要があります。値の数およびそれらのタイプは、[イベント タイプ] の定義と一致している必要があります。

Log(<Message>, <Level>)

このコマンドでは、システム ログにメッセージが送信されます。1 番目のパラメータは報告される情報メッセージで、テキストを自由な形式で指定できます。また、メッセージに文脈上の意味を与えるために、この文字列に変数の値を追加できます。「レベル」パラメータは、以下の値のうちのいずれかを取ることができます。

説明

警告メッセージが報告されます。

エラー メッセージが報告されます。

[ビジネス ロジック スコープ] で実行される場合にのみ、情報メッセージが報告されます。PslWriter で実行される場合は、メッセージは報告されません。これがデフォルトのレベルです。これは、通常デバッグの目的で使用されます。

例:

以下の例は、イベントのインフラストラクチャ情報が実際のインシデントの詳細の前に期待されたケースから得られたものです。アラート機能のメカニズムは、この条件の管理者に通知して問題の解決を促すように設定されました。

Out.Alert "Site Unknown Alert", Context.ClusterItem, Context.Rule

```
Out.Log("Fault Event Received for a Site with no infrastructure details: " &  
Context.ClusterItem)
```

違反根本原因コメントおよびイベント コメント

違反根本原因コメントは、サービス レベルの結果について説明するために、ビジネス ロジック内に設定できます。違反根本原因コメントは、メトリックに関連付けられます。

また、メトリックではなく **Raw** データ イベントと関連付けられたコメントである、イベント注釈を生成することもできます。これらの両方のタイプのコメントは、レポート データから表示できます。

ビジネス ロジックの「Tools」オブジェクト内の以下の 2 つのメソッドを使用すれば、違反根本原因レコードとイベント注釈レコードの作成が可能になります。

- **Tools.AddRootCauseComment (Text, Timestamp, [resourceId])**

- 根本原因コメントが保存されます。この情報は、後で生成されるレポートで役立てることができます。保存された根本原因コメントには、ビジネス ロジック 計算式の実行中の特定の瞬間の特定の状況が説明されています。情報パラメータでは、コメントを記述することを指定します。このメソッドでは、コメントと共に保存されるタイムスタンプが受信されます。また、このメソッドでは、メソッドのコンテキストに関連付けられたリソースを指定する **ResourceId** パラメータを受け付けます。（このパラメータはオプションで省略される場合があります。）

- **Tools.AddEventAnnotation (EventId, Text)**

- これらのメソッドは、ビジネス ロジック内の任意の箇所で使用できます。また、それらが適用される箇所のコンテキストを考慮する必要があります。サービス レベルが計算期間の間に予想されるものよりも低いと理由が分っている場合、根本原因コメントの追加が、計算期間の最後により関連している場合があります。たとえば、1 か月の期間中に 4 回の停止が発生し、それらすべてが 1 つのデバイスによって引き起こされたと仮定します。この場合、根本原因コメントは、この停止に関する情報のうちのいくつかから編集される可能性があります。これは、この期間中のレポートが表示される際に、この期間中のすべてのサービス レベル違反の原因となったものを確認できるようにするためです。 **AddRootCauseComment** コマンドは、**OnPeriodEnd** イベントハンドラルーチン、または計算中の期間の終わりに実行される他の同様の関数に最適です。
- ただし、イベント注釈の追加は、実際の **Raw** データ イベント処理により適しています。また **OnXXXEvent**（登録ステートメントで指定されたカスタム イベントハンドラ）に対する使用に有利に機能します。このイベントハンドラ内では、実際のイベントに固有のフィールドはすべて **eventDetails** オブジェクトを介して利用できます。
- これらのメソッドをビジネス ロジック内で使用する方法的例を以下に示します。

```
Sub OnPeriodEnd(Time)
    pctAvailable = (TotalTime-OutageTime) / TotalTime
```

```
        Tools.AddRootCauseComment "Violations caused by the
following items: " & ViolationCollection, Time
End Sub
...
...
Sub OnIncidentEvent(eventDetails)
    OutageTime = OutageTime + eventDetails("TimeToResolve")

    If eventDetails("TimeToResolve") > 6 Then
        ViolationCollection = ViolationCollection &
eventDetails("HardwareId")
        Tools.AddEventAnnotation eventDetails.EventId, "Incident " & _
            eventDetails("IncidentId") & " not resolved within target
time 6 hours"
    End If
End Sub
```

例外をタイムスロットから分離する

CA Business Service Insight のビジネス ロジックでは例外イベントを受け取りません。ビジネス ロジックが受け取るものは、例外期間の開始時は **OnTimeslotExit**、例外期間の終了時は **OnTimeslotEnter** です。このため、ビジネス ロジックでは例外時間とタイムスロット外の時間を区別できません。さらに、ビジネス ロジックでは例外タイプを区別できません。この結果、例外時間の動作と「タイムスロット外」の動作に別のロジックを実装することができません。

特別の例外（つまり「タイムスロット外」期間として動作しない例外）を実装する 1 つの方法は、例外を処理するための CA Business Service Insight の組み込みのメカニズムを使用するのではなく、専用のイベント タイプを定義することです。これらのイベントは、ある 1 つのアダプタを使用して、専用のデータ ソースからのイベント タイプを読み取ることで生成されます。

Excel のスプレッドシート（または他の任意のデータ ソース）では、これらの例外を格納できます。またこの後に、アダプタでこのデータをロードし、応答の **Exception Enter**（例外に入る）と **Exception Exit**（例外を出す）イベントを生成できます。また、例外は修正の使用によって追加される場合があります。この修正に加えて、ダミーのリソースを定義し、登録用のこれらのイベントと関連付ける必要もあります。このリソースは、コマンドで必要になるため、プレースホルダ以外の目的には役立ちません。

これらの専用のイベントによって報告される例外時間を処理できるように、ビジネス ロジック 計算式を、これらの例外イベントに、通常必要な、計算で使用するための **Raw** データ イベントの登録に加えて登録する必要があります。

特別な各種例外についての異なる処理を考慮して、ビジネス ロジック エキスパートがイベント タイプに例外タイプ用のフィールドを組み込むことをお勧めします。

この方法には以下の特性があります。

- この方法では、標準と特別という 2 つの例外セットが分離されます。
- 特別の例外には、メンテナンス用のユーザ インターフェースがありません。
- アダプタによってイベントとして生成された例外に基づくレポートでは、それらのイベントの存在に関してコメントしません。修正メカニズムが使用されたケースでは、コメントがあります。システムによって生成された結果の完全性を維持するために、修正メソッドの使用をお勧めします。
- 「例外あり/なし」の指定では、それらの例外を考慮しません。
- 修正メカニズムが使用されている場合、修正あり/なしが適用されます。

特別な例外を一度実装したら、ビジネス ロジック エキスパートがシステム メトリックのすべてにこのロジックを適用することをお勧めします。

必要な場合、例外を単一のリソースに適用する別の方法があります。このメソッドでは、リソースの「有効」ステータスの使用を伴います。リソースのステータスを「無効」に設定することは、対象の期間中に、計算エンジンで、そのリソースに対して送信される **Raw** データ イベントがすべて無視されることを意味します。新バージョンのリソースの作成によって、リソースが有効でない期間の設定によって、例外期間の開始時に 1 つ、例外期間の終了時にまた別の 1 つ。

ただし、リソースがクラスタ化メトリックの一部で、リソースが有効になる予定で、また同じ計算時期内に無効になる予定の場合は、前述のようにリソースが有効だった最後の期間のみが結果内で考慮されます。この場合、カスタム属性機能を使用することをお勧めします。リソースのステータスについて記述するリソース用の追加の属性は管理できます。また、ビジネス ロジック 計算式では、スクリプト内の関連する場所ごとにリソースのステータスのクエリを実行します。

メモリとパフォーマンスの考慮事項

ビジネス ロジックによる解決策を設計する際に、考慮する必要がある各種状況を以下に示します。記述された各状況は、計算エンジンのパフォーマンスが悪影響を受けるおそれがあるケースです。

■ パラメータ

パラメータの値がコード内で必要な場合、パラメータ値を割り当てるグローバル変数を作成することをお勧めします。また、パラメータの値が必要な場合は、代わりにグローバル変数を必ず使用してください。これにより、エンジンがパラメータの呼び出しごとにパラメータ マップを作成する状況がなくなります。

■ クラスタ化メトリック内でのマップの使用

クラスタ化メトリック用のビジネス ロジック内の大規模なグローバル マップオブジェクトは、慎重に検討した後で使用してください。エンジンでクラスタ化メトリックを計算している間、エンジンは前の状態からのグローバル変数をクラスタ内の項目ごとに別々に読み込むことでビジー状態になります。

■ ビジネス ロジックの登録

Raw データ イベントは、登録メソッドのみによってフィルタすることをお勧めします。「if」ステートメントを使用した内部フィルタリングをコード内に追加すると、処理時間が増加します。さらに重要なことには、エンジンで不要な Raw データのレコードの取得および処理のために余分なオーバーヘッドが必要になることです。

■ Dispatcher.RegisterByEventType の使用の回避

パフォーマンスが向上します。この登録メソッドを使用することは、この特定のタイプのイベントを持つリソースだけでなく、システム内のすべてのリソースに対して登録することを意味します。したがって、リソースのすべての変化はメトリックの計算に影響を与えます。この登録メソッドの使用による別のデメリットは、このメソッドが Raw データにアクセスする際のメトリックの実行時に発生します。この登録メソッドは、次に、Raw データから特定のイベント タイプのイベントのみをフィルタし、他のイベントを無視する必要があります。

■ Dispatcher.Register

Dispatcher.Register を使用する場合は、³番目のパラメータを指定していることを必ず確認します。³番目のパラメータなしでの登録は、イベント タイプ (Dispatcher.RegisterByEventType) による登録を実行していることとまったく同じです。つまり、最初の 2 つの隣に少なくとも 1 つの他のパラメータを使用していることを確認してください。

■ 計算粒度

計算とドリルダウンのために必要なエージェントのみをオンにすることが重要です。エージェントの時間単位のすべての計算は、プロセッサを大幅に使用します。

契約をアクティブ化する方法 (契約マネージャ)

契約の有効化はそのコミットにより実行されます。（詳細については、オンラインヘルプを参照してください。）

契約の有効化により、契約のメトリックの計算を開始し、また契約の結果の生成を開始するめるために、エンジンがトリガされます。

契約をアクティブ化する前に、以下の条件がすべて満たされているかどうかを確認してください。

- すべてのメトリックにビジネス ロジックが定義され（メトリック内、またはリンクされたモジュールとして）ており、またテスト済みで、エラーが含まれていない必要があります。
- すべてのメトリックに定義済みのダッシュボードしきい値があります。（詳細については、オンラインヘルプを参照してください。）計算処理中にダッシュボードでメトリックの限度を評価できるように、しきい値がすでに定義されていることが重要です。
- 発効日が正しい期間に従っており、また利用可能な **Raw** データ レコードが存在していること。また結果がビジネス ロジック スコープを使用した想定されたものであることを確認します。

契約が 1 回コミットされている場合、有効化が正常に開始されているかどうか、およびその計算が予想どおりに進捗していることを確認します。

以下の手順に従います。

1. CA Business Service Insight のサービス コンポーネントのすべてが開始されている（特に計算エンジン。これには **PSLWriter** と **PenaltyWriter** の両方が含まれています）ことを確認します。計算が必要な場合は、必ずサービス コンポーネントをすべて実行することをお勧めします。
2. 契約を診断します。契約診断機能では、すべての契約メトリック（および使用される場合はペナルティの計算式）に対して実行された一連のテスト結果が表示されます。テスト結果の重大度が提示される解決手順と共に提供されます。契約をアクティブ化した場合は、その契約の計算の完了後と同様に、必ず診断を実行することをお勧めします。
3. [計算ステータス] の自由形式レポートを生成します。このレポートは、CA Business Service Insight の初期インストールの一部としてバンドルされ、「Admin Reports」バンドル フォルダ内にあります。このレポートは、計算の進捗状況に関する情報を提供し、PSL エンジンが進捗しているかどうか、計算が完了しているかどうかを確認するために、この時点で使用できます。このレポートを確認して、計算になんらかの潜在的な問題が存在するかどうかを評価します。

このレポートには、以下の列フィールドがあります。

フィールド	説明
契約	契約の名前が保持されます。リストには、有効または無効な契約の名前が含まれます。
メトリック	[契約] 内のメトリックの名前が保持されます。リストには、各 [契約] 内に含まれているメトリックがすべて含まれます。
トラッキング期間	メトリックの計算期間が保持されます。リストには、アクティブで、メトリックの粒度定義に基づくエージェントに従ったメトリックの計算時間単位ごとのエントリが表示されます。計算期間がトラッキング期間の場合では、これが注目されます。
表示日付までに更新	前回の結果の更新時間が保持されます。これは、特定のメトリックの結果がこの日付まで利用可能であることを示します。たとえば、 01/01/2006 が表示されている場合、これは、その時間単位のそのメトリックの結果がすべてこの日付までに更新されること、およびレポートがこの日付までその結果に利用可能であることを示します。
最後のサイクル開始時刻	メトリックの結果が更新された計算が開始した時の時間とサイクルが保持されます。
再計算開始が必要になる日付	エンジンで再計算用のあるメトリックをタグ付けし、それがまだ更新されていないケースでは、結果が関連せず、そのため更新が必要になる時間を指定した、生成された日付がここに表示されます。これは、再計算のいずれのケースでも発生する可能性があります。
[前回の更新時刻]	エンジンで前回の結果を持つレコードを更新した時間。
処理したエンジン番号	特定のメトリックの計算の処理を行なうために割り当てられたエンジンの番号が保持されます。

このレポートでは、利用可能な **Raw** データに基づいた以下の情報も提供できます。

- 単一のメトリックを計算するためにエンジンを使用した時間。単一のサイクルで計算されたメトリックをその更新された時刻ですべて並べ替えることによって、各メトリックを更新するために、エンジンを使用した時間を確認することができます。同じ [前回のサイクル開始時刻] の値を持つレコードは、すべて同じサイクル中に計算されます。また更新時刻は、[前回の更新時刻] の時刻です。時間単位の各々と同様にその基になるエージェントの時間単位を持つ全メトリックを計算するために、エンジンを使用した時間を評価することが可能です。
- 完全な契約を計算するためにエンジンを使用した時間。これは、契約の最初のメトリックの更新時刻、およびその契約の最後のメトリックの最後の更新時刻の検査および中間の時刻の計算により実行されます。

契約メトリックの完全な再計算

現在のコンテキストでの再計算の処理は、データ ソース エキスパートまたはビジネス ロジック エキスパートのいずれかによる完全なシステムの再計算の開始です。これは通常の計算処理中に実行されるエンジンの再計算ではありません。さまざまな誤動作が確認されている場合、この種のアクションが、通常契約設定の処理中またはその処理後に実行されます。

システムが完全な再計算を使用して現場で稼働するようになる前に、システムが一度安定状態になった（つまり、システムの構築中ではなく）場合にのみ、完全な再計算を開始することをお勧めします。

再計算は、現在データベースに対して SQL スクリプトを実行することにより実行されます。このスクリプトでは、PSL テーブル、および計算処理の一部である関連の支援テーブルのすべてがクリーンアップされます。時々発生するシステムに対する構造変更が、データベース スキーマおよび（または）依存オブジェクトへの変更を結果的にもたらす可能性があるため、このスクリプトは CA サポート チームによって承認され検証される必要があります。

注: スクリプトを実行する前に、すべてのサービス コンポーネントを停止する必要があります。またこの実行の後に、サービス コンポーネントが、計算を再スタートするために再起動される場合があります。

以下の状況が、完全な再計算の必要性を示している場合があります。

- [契約] 内の定義に関する問題 - この段階にある場合、メトリックの目標値が誤って定義されているか、またはメトリックのしきい値が誤っていることが見つかった。
- 契約におけるエラー。
- ダッシュボードしきい値を再評価するか、または SLA アラートを再生成する要件。

注: 必要な場合、CA サポートに連絡して、このオプションについて話し、またインストールされているバージョンの CA Business Service Insight 用の関連スクリプトのコピーも取得します。

成果物の作成(契約マネージャ)

この段階では、すべてのシステムの出力を要件を満たすように設定します（すべてのレポートの作成およびフォーマットなど）。

成果物の設定には、以下のものが含まれます。

- セキュリティ設定（ユーザ権限とグループ）の定義。
- 保存済みレポートの作成。
- ブックレットの作成。
- 契約エクスポートテンプレートの作成。
- サービス デリバリ ナビゲータ（SDN）ビューの作成
- ダッシュボードマップの設定
- サービス レベル アラート プロファイルの作成。

セキュリティ設定の定義(管理者)

セキュリティ設定を定義する際、CA ユーザおよびそれらの関連付けられた権限を設定する必要があります。これらの設定には、ユーザがアクセスできる必要がある情報（ユーザが表示または編集できるシステム内のエンティティ）の定義が含まれます。これらの権限は、ユーザ グループ、役割、またはユーザごとに個別にと、さまざまなレベルで設定できます。アクセス可能な情報は契約関係者に関連して定義され、ユーザに直接設定でき、またユーザが属するユーザ グループから継承することもできます。

主な役割を設定しそれとユーザ グループを関連付けるのは、この段階です。この結果、新規ユーザを追加する際に、そのユーザをあるグループに関連付けて関連の設定を継承する必要があるだけです。

許可されるアクションは役割に設定され、ユーザをある役割と直接関連付けるか、またはユーザが属するユーザ グループによってユーザに発行されます。許可されるダッシュボード関連のアクションも、関連付けられる役割に定義されます。

定義する必要があるユーザ グループと役割、またユーザの容易な追加が可能な構造を設定するための必要な権限を、管理者が決定することをお勧めします。

レポートの作成方法

以下のプロセスを使用して、レポートを作成します。

1. レポートの必須フォルダをすべて作成する - 新しい個々の保存済みレポートを保存する際に利用可能なように、すべてのフォルダを事前に作成する必要があります。各契約には、通常高位レベル レポート用の経営者フォルダなどの、それ自体のフォルダが存在します。
2. [レポート ウィザード] を使用してレポート フィルタ条件を定義する - 各保存済みレポートの作成は、[レポート ウィザード] を使用してレポートを生成することで始めます。[レポート ウィザード] では、必要なフィルタ条件が選択されており、レポートが生成されます。IT 管理者は、レポート ユーザ/ビューアによって入力されるユーザ定義のフィールドを指定して、そのユーザの関心に固有のレポート結果の生成を可能にするレポート パラメータを設定できます。

レポートを保存済みレポートとして設定する目的で生成する場合、フィルタ条件をできるだけ柔軟に定義することが非常に重要です。これは、システムが成長し進化するにつれて、不必要な処理の増加を防ぎます。この目的は、各レポートが最新の更新された情報を常に表示することを保証するためです。たとえば、レポートが現在 3 つのサービス コンポーネントを表示していて、今後新しいサービス コンポーネントを追加する場合、この新しいサービスがレポートに自動的に追加され、新規レポートの設定を必要としないことが重要です。また、月ごとにレポートしていて、レポートが現在 3 か月分を表示している場合、来月にはそのレポートは先月を含めた 4 か月分を表示する必要があります。多くの場合、レポートには常に最後の 6 か月分のデータを表示する必要があります。これらの「スライドする期間」タイプのレポートは、一度作成すれば余分な注意を必要としないため、固定期間レポートとは対照的に非常に有用です。

保存済みレポートに対するフィルタ条件の設定についてのいくつかヒントを以下に示します。

[基準] タブ

- [ビジネス データのみ] オプションを使用すると、表示されるデータがメトリックのトラッキング期間に関連するもののみ限定されます。
 - レポートする特定のメトリックを選択するのではなく、常に設定をグループ化オプションまたはごとのレポート オプションを指定します。
 - 動的時間範囲を設定します。これが固定時間範囲に設定されていると、レポートには常に同じ結果が表示されます (つまり、「過去 3 か月」は動的範囲です)。

[その他] タブ

- [未完了期間] をレポート内に表示する必要があるかどうかを確認します。表示する必要がある場合、このオプションをリストから選択します。通常レポートの設定時には、未完了期間はそれが最終結果ではないため除外されます。このためビジネス結果（業績）ではありません。
- レポート レイアウトを設計する - レポートを生成すると、[レポート ページ] の [設計ツール] を使用して、そのレイアウトを設計できます。設計には、レポートの意図されている受信者に応じて特定の要件が存在する場合があります。想定されるシナリオごとにいくつかの設計レイアウトを作成して、これらの設計を生成される残りのレポートに適用することをお勧めします。したがって、レポートの基準を定義する際に、[表示] タブで [基にするデザイン] を選択します。

注: 編集ツールを使用してレポートのレイアウトを設計する方法のヒントについては、次のセクションを参照してください。

[全般] タブ

- レポートを保存する - レポートの生成と設計が完了すると、それを関連するフォルダに保存できます。
- レポートを保存する処理中に、レポートはそのレポートにアクセス権を持つユーザと関連付けられます。このため、レポートをユーザと関連付けることができるように、ユーザ グループをすでに定義していることが重要です。正しい権限を持つユーザは、後の段階でフォルダのオプションを使用してレポートに関連付けることができます。
- 類似するか、または相互に関係のあるレポート間のナビゲーションがそれらのレポートのユーザに対して簡略化されるように、関連するレポートを関連付けます。
- [レポート フォルダ] ページでは、レポート、レポート グループ、複合レポート、自由形式レポート、ブックレット、ショートカット、およびフォルダを作成できます。また、レポートを検索することもできます。

レポート メニューで[レポート フォルダ]をクリックします。[レポート フォルダ] ページが表示され、保存済みレポートのリストが示されます。

偏差レポート

偏差値は、目標値のあるメトリックに対して CA Business Service Insight エンジンによって自動的に計算されます。この値は、実際のサービス レベルと目標値の間の差を表します。CA によって自動的に計算される偏差計算の計算式は、サービス レベル定義によって変わります。つまり、サービス レベル目標値が最大値（実際のサービス レベルが「次の値以下」の場合）か、最小値（実際のサービス レベルが「次の値以上」の場合）かによって変わります。計算式がどのように変わるかを示す以下の例を参照してください。

ターゲット ステートメント サービスレベルのしき 偏差の計算式 い値

サービスは、スケジュールされた時間のうち少なくとも 99% で利用可能である必要がある。

目標値は最小のしき
い値

$$Service_{Dev} = (-1) \times \frac{Service_{SL} - Service_{SLT}}{Service_{SLT}}$$

平均 MTTR は、1 か月当たり 4 時間を超えてはならない。

目標値は最大のしき
い値

$$Service_{Dev} = \frac{Service_{SL} - Service_{SLT}}{Service_{SLT}}$$

ここで $Service_{Dev}$ は、サービスの偏差

ここで $Service_{SLT}$ は、期待されるサービスのパフォーマンス

ここで $Service_{SL}$ は、実際のサービスのパフォーマンス

以下の例では、最小の偏差計算を示しています。

サービスは、スケジュールされたタイムスロットのうち少なくとも 99% で利用可能である必要がある。実際のサービス レベルは、スケジュールされたタイムスロットで 98% です。

$$Service_{Dev} = (-1) \times \frac{98\% - 99\%}{99\%} = 1.0101\%$$

偏差レポートは、ある外部性（別の種類の計算）の高位レベル表示の保証に使用されます。また共通の基盤を備えた 1 つのバーに偏差を集約するためにも使用されます

たとえば、以下のマトリックスが契約に存在するとします。

サービス - ヘルプ デスク	サービスドメイン チケット管理 優先度 1	サービスドメイン チケット管理 優先度 2	サービスドメイン チケット管理 優先度 3
	P1 の平均解決時間	P2 の平均解決時間	P3 の平均解決時間
	T1 内で解決されたチケットの P1 の %	T1 内で解決されたチケットの P2 の %	T1 内で解決されたチケットの P3 の %
	T1 内で応答されたチケットの P1 の %	T1 内で応答されたチケットの P2 の %	T1 内で応答されたチケットの P3 の %
	P1 の平均応答時間	P2 の平均応答時間	P3 の平均応答時間

ヘルプデスク サービスによってフィルタされた [サービス ドメイン] レポートを生成した結果は、以下のグラフに示されたとおりです。

上記のレポートでは、サービス レベル マネージャは、ヘルプデスクが契約または義務の種類に関係なく各種優先度に基づいてどの程度順調に実施されているかを確認することができます。

ヘルプデスクのメトリックは、すべてそれらの優先度に基づいて 1 つのバーに集約されます。

たとえば、「優先度 1」のバーには、メトリック内に定義された 3 つのメトリックが表示され、それらの偏差が 1 つの値に集約されます。（集約メソッドでは、平均/最小/最大にするようにレポート ウィザード内で選択できます。）

このようなレポートを使用すれば、たとえばマネージャは、「優先度 1」のチケットにさらに多くのリソースを投入するか、またはそれらに関連する契約を変更する必要があることを判断できます。

この例では、モデリングによって契約が履行されたか違反されたかを示す 1 つの義務に関するレポートだけでなく、サービス レベル マネージャが自分のリソースをより効率よく管理でき、その結果自分のサービス コンポーネントを向上させることもできる、より広範な管理レポートも提供することを示しています。

自由形式レポート

自由形式レポートを使用すれば、ユーザは CA Business Service Insight のデータベースまたは CA Business Service Insight サーバから接続を介してアクセスできる他の外部データ ソースから SQL クエリに基づいたレポートを生成できます。また、これには、Excel、Access、Lotus Notes、テキスト ファイルなどの ODBC を介してアクセスできる他の種類のデータ ソースも含まれます。自由形式レポートは、ビジネス ロジックのコマンド Tools.SaveRecord と Tools.SaveFields によって作成されたデータに基づいた統計レポートを設定するために一般的に使用されます。

自由形式レポートでは、選択したデータベースに接続文字列を介して接続します。また、クエリ文字列を使用して、データベースに対して SQL クエリを実行します。パラメータを両方の文字列に追加して、動的レポートを生成することができます。これによってユーザは、データベースに接続するためのユーザ名やパスワードなどの、クエリに含める特定の値を入力または選択できます。

自由形式レポートは、[レポート ウィザード] を使用して生成されたレポートと同様に、[グラフ]、[データ]、および[フィルタ] タブ内に表示されます。

注: 自由形式レポートには、列が最初の列を除いてすべて数値の場合にのみグラフを含めることができます。最初の列のデータは X 軸タイトルに使用されます。列名は他のタイトルに使用されます。

自由形式レポートは、データベースおよび開いている SQL クエリへの直接アクセスを使用するという理由で、メンテナンスに問題があります。自由形式レポートのソースとして役立つ基となるデータに影響を与えないように十分に注意してください。レポートが外部データ ソースから生成される場合、それらのデータソースが、自由形式のデータ レポートに責任を負う契約マネージャにまず意見を求めずに変更されないことを保証するために、通知プロセスを設定することをお勧めします。

自由形式レポートの作成時に留意する必要がある全般的な情報。

- 国際的な日付形式の問題のために、ビジネス ロジック内にカスタムの計算式を作成することによって、日付の正確な形式を指定することが多くの場合役立ちます。これにより、日付が常に同じ形式の文字列に変換され、また米国と EU の日付形式の問題が回避されます。これは、Tools.SaveFields コマンドまたは Tools.SaveRecord コマンドを介して T_SLALOM_OUTPUTS テーブルに書き込まれる日付に対して使用してください。サンプルの計算式を以下に示します。

Function FormatDate(DateField)

```
If DateField = "" Then
    FormatDate = ""
Else
    Dim PeriodYear, PeriodMonth, PeriodDay, PeriodHour,
```

```

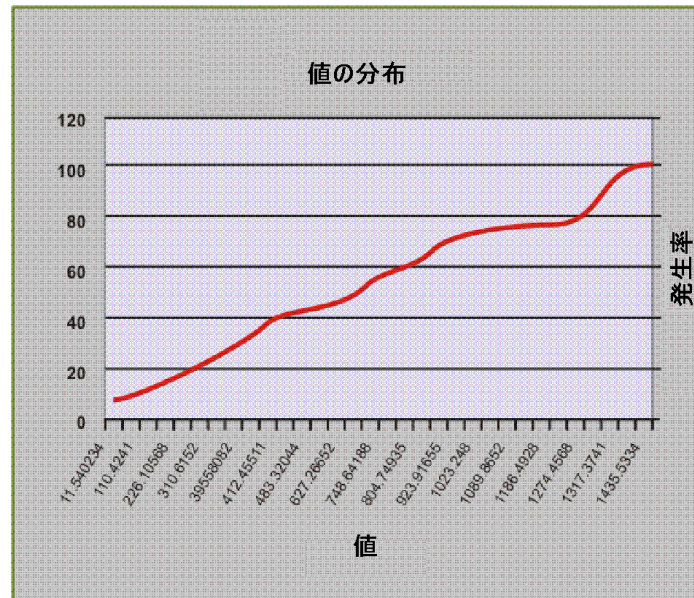
PeriodMinute, Periodsecond
    PeriodYear    = DatePart("yyyy",DateField)
    PeriodMonth   = DatePart("m",DateField)
    PeriodDay     = DatePart("d",DateField)
    PeriodHour    = DatePart("h",DateField)
    PeriodMinute  = DatePart("n",DateField)
    Periodsecond  = DatePart("s",DateField)
    FormatDate    = PeriodDay&"/"&PeriodMonth&"/"&PeriodYear& _
" "&PeriodHour&":"&PeriodMinute&":"&Periodsecond
End If
End Function

```

- ビジネス ロジックのイベントハンドラからの「time」パラメータ、または eventDetails オブジェクトからのタイム スタンプを使用する場合は、常にタイム ゾーンのシフトの影響に注意してください。テーブルに書き込まれた際の日付がローカル時間ではなく UTC 時間である場合があることに注意してください。これを修正するために、Tools.GetLocaleTime() 関数を使用することが必要になる場合があります。
- 外観をカスタマイズするには、これまでどおりレポート デザイナのユーティリティ（自由形式レポートでグラフを生成する場合）を使用できます。
- 自由形式レポートの PDF エクスポート オプションは、自由形式の設定画面の [レポート パラメータ] セクションを使用してカスタマイズ可能です。ページ レイアウト（横、縦）などのもの
- HTML をレポート内に埋め込んで、ハイパーリンクの追加、または列と行の色 / フォントや表示設定の変更などのような別の機能を実行することができます。
- データベース ビュー（Oracle の機能）を T_SLALOM_OUTPUTS テーブル上に作成して、レポートに必要な SQL を簡略化できます。
- レポートにパラメータを指定する場合、それらを次のものを含むいくつかの方法で設定することができます：自由形式テキスト、数値（検証）、パスワード（非表示文字 - DB 接続へのパスワードに役立つ）、日付（提供される日付ピッカ ユーティリティを使用）およびリスト（リストに入力するために SQL ステートメントを指定して作成）
- 自由形式レポートの SQL セクションに指定されたパラメータ値は、「@」記号が前に付いたパラメータ名と置換する必要があります。つまり @PARAM1。値が文字列の場合、このパラメータ値の前後に引用符を使用することが必要になる場合があります。

汎用のヒストグラム自由形式レポート

以下で述べるクエリを、下のグラフに示したようにテーブル内の値の分布をパーセンテージで示すために、自由形式レポート内で使用することができます。



前に示したグラフでは、各値のどの割合（パーセンテージで）が 11.5 未満（0%）、804.74 未満（～50%）、および 1435.53 未満（100%）であるかを確認できます。

SLA が目標を「値の x% が y 未満であるべき」と指定した場合、この自由形式の結果は SLA への適合を保証する x 値と y 値を見つけることに役立ちます。

以下のパラメータがクエリ内で使用されます。

- **@Query** - 次の形式のクエリの select ステートメント： `select some_value val from some_table`。エイリアス（別名）の `val` は必須です。このクエリにより、分布が分析される各値が提供されます。
- **@Buckets** - x 軸上の値の数。ソース値はこれらの数に丸められます。たとえば、`@Buckets = 100` を指定すると、ソースデータの値は 100 のグループに分割されます。また、クエリによりどれだけの値が各グループの範囲内に入るかが計算されます。`@Buckets` の値を大きくすると結果はより正確になりますが、グラフの視覚的な訴求力は低下します。
- **@Relation** - 累積の方向。可能な値は、「<」（目標が「次の値以上」の場合）または「>=」（目標がそれ以外の場合）です。

クエリは、最適な結果を得るために、データ ソースまたは `T_SLALOM_OUTPUTS` に対して実行できます。

以下のクエリによって、前に示したグラフが生成されます。

```
select val,100*records/(select count(*) from (@Query))
from
(
  select x.bucket_val val,
    sum(y.records) records
  from
  (
    select round(val/bucket_size,0)*bucket_size bucket_val,
      count(*) records
    from
    (
      select (max(val)-min(val))/@Buckets bucket_size
      from
      (
        @Query
      )
    ) params,
    (
      @Query
    ) source
  group by round(val/bucket_size,0)*bucket_size
  order by round(val/bucket_size,0)*bucket_size
) x,
(
```



```

select round(val/bucket_size,0)*bucket_size bucket_val,
       count(*) records
from
(
  select (max(val)-min(val))/@Buckets bucket_size
  from
  (
    @Query
  )
) params,
(
  @Query
) source
group by round(val/bucket_size,0)*bucket_size
order by round(val/bucket_size,0)*bucket_size
) y
where y.bucket_val @Relation x.bucket_val
group by x.bucket_val
order by x.bucket_val
)

```

使用可能なサンプルのパラメータ リスト (XML 形式) を以下に示します。

```

<custom>
  <connection>
    <params/>
  </connection>
  <query>
    <params>
      <param name="@Query" disp_name="Data Type" type="LIST">
        <value>PDP Context Activation Success</value>
        <list>
          <item>
            <value>select success_rate as val from
PDP_Context_Activation_Success.CSV</value>
            <text>PDP Context Activation Success</text>
          </item>
          <item>
            <value>select throughput as val from [gprs throughput
volume by apn.csv]</value>
            <text>Throughput of a Single APN</text>
          </item>
          <item>
            <value>select throughput as val from [Generic GPRS
Throughput.CSV]</value>
            <text>Generic Throughput</text>
          </item>
        </list>
      </param>
      <param name="@Buckets" disp_name="X Axis Values" type="LIST">

```

```

        <value>100</value>
        <list>
            <item>
                <value>25</value>
                <text>25</text>
            </item>
            <item>
                <value>50</value>
                <text>50</text>
            </item>
            <item>
                <value>100</value>
                <text>100</text>
            </item>
            <item>
                <value>250</value>
                <text>250</text>
            </item>
            <item>
                <value>500</value>
                <text>500</text>
            </item>
            <item>
                <value>1000</value>
                <text>1000</text>
            </item>
        </list>
    </param>
    <param name="@Relation" disp_name="Violation of threshold means"
type="LIST">
        <value>providing too little</value>
        <list>
            <item>
                <value>&gt;=</value>
                <text>providing too little</text>
            </item>
            <item>
                <value>&lt;=</value>
                <text>providing too much</text>
            </item>
        </list>
    </param>
</params>
</query>
</custom>

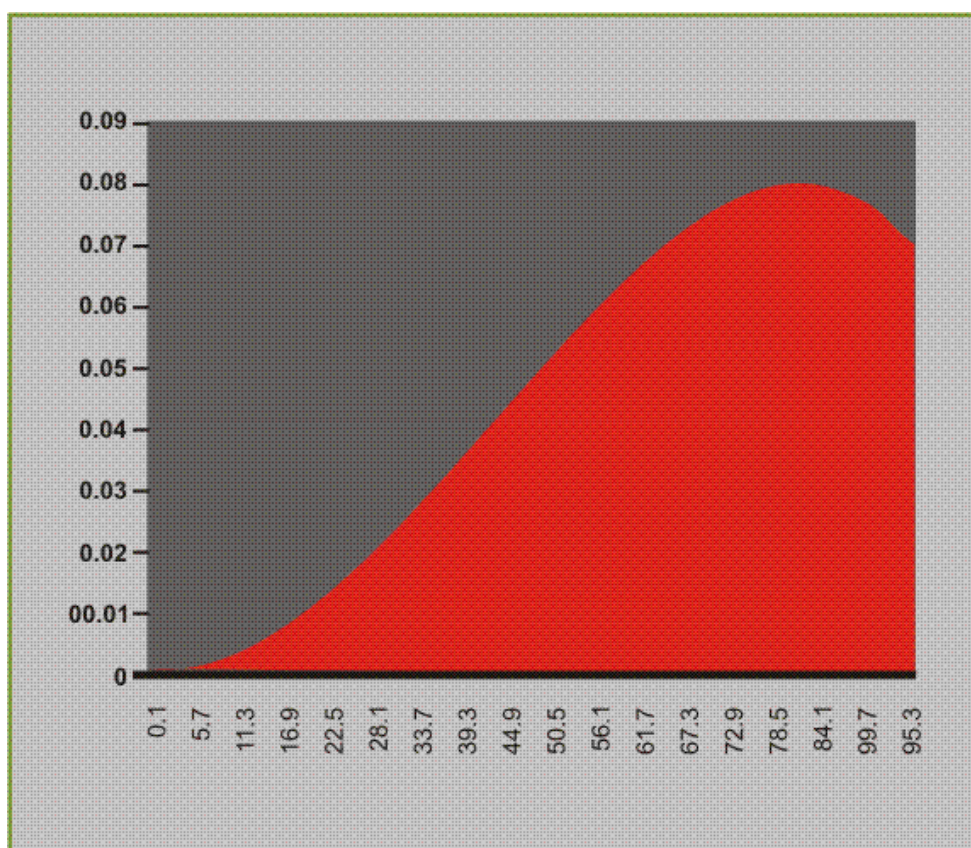
コメント

```

- このクエリは、Oracle と SQL Server 向けに最適化されています。他の ODBC データソースについては、列別名の前に「as」を追加し、また他の変更を適用することが必要になる場合があります。
- 結果を Excel にエクスポートする場合は、ユーザで XY (散布図) レポートを生成してそれを表すことをお勧めします。グラフを点で描ける場合は、値の散乱を示すことも可能です。

汎用の正規分布(ガウス)自由形式レポート

以下で述べるクエリを、下のグラフに示したようにテーブル内の値の正規分布を示すために、自由形式レポート内で使用することができます。



以下のパラメータがクエリ内で使用されます。

- @Query - 以下の形式の select ステートメント

```
select min (something) min_val,
       max (something) max_val,
       avg (something) avg_val,
       stddev (something) stddev_val
from table_name
```

x_val は必須です。このクエリにより、分布が分析される各値が提供されます。

- @Buckets - x 軸上の値の数。ソース値はこれらの数に丸められます。たとえば、@Buckets = 100 を指定すると、ソース データの値は 100 のグループに分割されます。また、クエリによりグループごとの正規分布の値が示されます。@Buckets の値を大きくすると結果はより正確になりますが、計算にかかる時間がより長くなります。100 が @Buckets に対する適した選択です。

このレポートの場合も、クエリが最適な結果を得るために、ソース データまたは T_SLALOM_OUTPUTS に対して実行できます。

以下のクエリによって、前に示したグラフが生成されます。

```
select (max_val-min_val)/@Buckets*serial,
       (1/stddev_val*sqrt (2*3.14159265359))*exp ( -1/(2*power (stddev_val, 2))*power
       (((max_val-min_val)/@Buckets*serial-avg_val), 2))
from   (
        @Query
      ),
      (
        select rownum serial
        from   t_psl
        where  rownum < @Buckets
      )
order by serial
```

これに対応する XML パラメータを以下に示します。

```
<custom>
  <connection>
    <params/>
  </connection>
  <query>
    <params>
      <param name="@Query" disp_name="Data Type" type="LIST">
        <value>PDP Context Activation Success</value>
        <list>
          <item>
            <value>
select min(success_rate) min_val,max(success_rate) max_val,avg(success_rate)
avg_val,stddev(success_rate) stddev_val
from PDP_Context_Activation_Success.CSV
</value>
```

```

        <text>PDP Context Activation Success</text>
      </item>
      <item>
        <value>
select min(throughput) min_val,max(throughput) max_val,avg(throughput)
avg_val,stddev(throughput) stddev_val
  from [gprs throughput volume by apn.csv]
</value>
        <text>Throughput in Kb</text>
      </item>
    </list>
  </param>
  <param name="@Buckets" disp_name="X Axis Values" type="LIST">
    <value>100</value>
    <list>
      <item>
        <value>25</value>
        <text>25</text>
      </item>
      <item>
        <value>50</value>
        <text>50</text>
      </item>
      <item>
        <value>100</value>
        <text>100</text>
      </item>
      <item>
        <value>250</value>
        <text>250</text>
      </item>
      <item>
        <value>500</value>
        <text>500</text>
      </item>
      <item>
        <value>1000</value>
        <text>1000</text>
      </item>
    </list>
  </param>
</params>
</query>
</custom>

```

コメント

- このクエリは、Oracle と SQL Server 向けに最適化されています。他の ODBC データ ソースについては、列別名の前に「as」を追加し、また他の変更を適用することが必要になる場合があります。
- 結果を Excel にエクスポートする場合は、ユーザで XY（散布図）または面レポートを生成してそれを表すことをお勧めします。

ダッシュボード ページの設定

以下のセクションには、CA Business Service Insight ユーザ用に設定されるコンテンツに関する推奨事項が記載されています。この推奨事項は高位レベルにあり、特定の顧客要件を考慮に入れてください。これらのページは、組織における役割を介して説明されるある特定のユーザの文脈で以下に説明されています。

経営責任者のページ

経営責任者が部門、国、顧客などにわたる高位レベル ビューに関心があるという想定です。経営責任者は通常業務を行わないため、戦略に関して決定を下すための情報を経営責任者に提供するビューが必要です。このため、現在のステータスではなく契約上のステータスが、経営責任者マップおよび集約的な「ダッシュビュー」への表示により関連する可能性があります。

たとえば、以下の各ダッシュビューが「概要」ページに組み込まれる場合があります。

ダッシュビュー	説明
クリティカルな顧客	慎重な扱いを要するものとしてマークされた契約関係者がすべて含まれています。経営責任者は、自分の観点から慎重な扱いを要するものとみなした契約または契約関係者を選択します。
全体的業績	全体的品質のカスタム ウィジェット、顧客の KQI のすべてが含まれた集約的なビューのウィジェットが組み込まれます。
部門	組織図を備えた背景画像が使用され、また関連する部門に関するウィジェットが配置されています。通常、サービス コンポーネント グループが、ある部門が組織内で表しているものに応じて、このダッシュビューでは役立ちます。
地勢	地理的な地図を備えた背景画像が使用され、また関連する場所に関する契約関係者グループ ウィジェットが配置されます。
会計実績	会計メトリックについての集約的な情報が含まれるウィジェットが組み込まれています。
URL	たとえば、企業ポータルからのページや販売部内のリードなどが組み込まれています。

レポート

偏差レポート

今月の会計報告

ページに推奨されるレポート

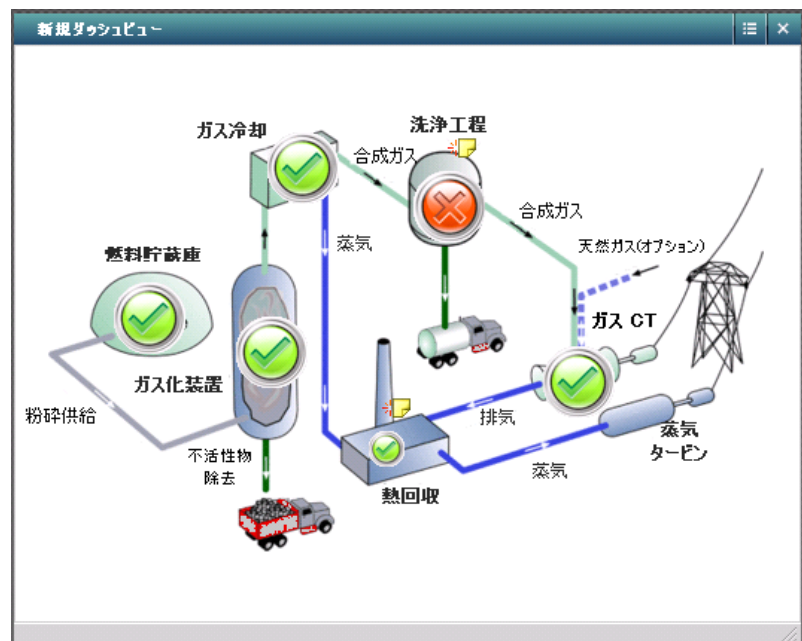
説明

一定の期間のワースト 10 の契約で、サービス デリバリの点で最悪のパフォーマンスにある領域に関する情報が提供されます。

時間によって集約された会計状態が要約されます。

プロセス ページ

このページには、下の例内のようなプロセス内の各チェーンを表す、ウィジェットを使用したプロセス図を表示するダッシュビューが含まれている必要があります。



契約マネージャのページ

契約マネージャに設定されたページには、契約マネージャが管理している顧客ごとにどの程度サービスが順調に提供されているかが表示されます。契約マネージャの責任下にある契約の義務に対して提供された現在のサービス レベルに関する管理者を更新するための、関連する契約のダッシュビュー。また、このようなページには、契約に含まれているサービス コンポーネントごとの利用可能なレポートが表示されます。

概要ページには以下のダッシュビューが含まれます。

ダッシュビュー	説明
マイ アカウント	契約マネージャの責任下にある、すべての契約または契約関係者のウィジェット。
マイ サービス	契約マネージャによって管理される顧客すべてにわたるサービス コンポーネント。
会計実績	管理対象顧客に関する会計メトリックについての集約的な情報が含まれるウィジェット。
URL	CRM システムなどのポータル

特定の顧客義務に関するビューを提供する管理対象顧客ごとの顧客ページを作成します。アカウント マネージャに先を見越して迅速に行動する能力を提供するために、現在のステータス計算のビューを契約上のステータスのわきに組み合わせることをお勧めします。例：

顧客ページ	説明
顧客義務	契約に含まれたメトリックが組み込まれます。
会計実績	管理対象顧客に関する会計メトリックについての集約的な情報が含まれるウィジェット。

サービス マネージャ ページ

特定のサービスまたはドメイン担当のマネージャ用に構成されるページ。さまざまな契約にわたるサービス目標、およびその目標の状態に関する詳細ビューを表示します。さらに、そのようなページには測定されるサービスの重要なサービスレベルパラメータを強調表示するレポートが含まれています。

サービス マネージャのページに含まれるダッシュビューはアカウント マネージャ ページと類似しており、情報は契約または契約関係者レベルではなく、サービス レベルでのみ表示されます。

ホーム ページ

ダッシュボード ページは、情報とアクションに迅速なアクセスを提供して、システムとの対話用の、カスタマイズ可能なゲートウェイを可能にするように、ホーム ページとして割り当てることができます。

サービス レベル アラート プロファイルの追加

アラート プロファイルを使用すれば、指定したデバイスを使用して、アラート通知が事前定義済みの受信者に送信される条件の定義ができます。

プロファイルを作成する前に、通知を必要とする条件のどれが十分に重要か、また目的の受信者を誰にすべきかを検討する価値があります。あらゆる問題や発生したサービスの問題を効果的に管理するために、契約マネージャとシステム管理者を支援できるプロファイルを定義することが重要です。通知はすべて、最も緊急な問題に対処して違反の状況を防ぐために必要なリソースを割り当てる際に具体的で明確である必要があります。

サービス レベル アラートを追加する場合、メトリックのステータスを判定してアラート条件をトリガするかどうかを決定するために使用できるフィールドが多数あります。標準的なメトリック詳細のうちのいずれか1つを、フィルタリングと条件の基準(契約、サービス、メトリック名、目標値、タイムスロット、サービスレベル結果など)に使用できます、サービス レベル予測ではアラートも通知できます。この場合、いくつかの追加の値(最良/最悪のケース偏差など)が提供され、またサービス レベルも見積もることができます。これらは、サービス レベル違反が一定の期間内に回復できるかどうかを判断するために非常に役立ちます。

CA Business Service Insight の [システム ログ] セクションおよび [全般] セクションに基づくアラートは、別の非常に役立つ機能です。これにより、アラートをシステム内で発生するすべてのアクション（これはシステム ログに書き込まれます）に基づいて送信することが可能になります。アクションがほとんどすべてログに記録されるため、これによりアラート プロファイルに非常に広い範囲が提供されます。共通のシステム ログのアラート プロファイルは、以下のものから構成されます。

- データ ソース エキスパートに対してアラートを実行するアダプタの通知。
- すべてのログに登録された「エラー」条件、システム管理者に送信された電子メールのアラート。
- データ ソースからの特定の条件に関連するビジネス ロジックによって作成されたカスタム エラー。このエラーでは、組織内のサービスを担当するグループおよび契約マネージャに電子メール アラートを生成できます。（つまり、高優先度インシデントが期間内に解決されていない場合、問題をエスカレートするためにヘルプデスクにアラートを送信する）
- 繰り返された無効なログインの試行。これは、誰かがシステムにハッキングしようとしていることを調査するように、システム管理者に電子メールを送信できます。
- アダプタから到着した新しい変換エントリ（つまり、新しいリソース エントリがデータ ソース内で見つかっています。これは、SLA が正しく計算されることを可能にするためにシステムでの変換を必要とする可能性があります。）。

' [アラート プロファイル詳細] ウィンドウを以下に示します。これには、「ヘルプデスク チケット オープン イベント」のカスタム イベントタイプに関してレポートするカスタム アラート プロファイルの設定が示されています。基準が一致する場合クリティカルアラートがこのチケットが効率よく管理されるようにヘルプデスクのコーディネータに送信されます。これは、アプリケーションが将来監視下にあり、極めて特別なレベルの注意が必要な状況に役立つ可能性があります。

アラート プロファイル詳細

名前: *	Critical Fidessa outage	
説明:	All servrity 1 incidents raised for fidessa application are sent to Helpdesk Coordinator for analysis and possible escalation	
タイプ: *	Helpdesk Ticket Open Event	リソース: * Enquity Traders-Trans E
条件計算式:	<input #severity='1"/' &&="" fidessa"="" type="text" value="#ID= "/>	計算式エディタ
メッセージ件名: *	Critical Fidessa Outage	
メッセージ:	Critical Fidessa Outage - Equity Traders Down	メッセージエディタ
受信者: *	含む Helpdesk Coordinator 利用可能 Demo1 Oblicore Support Sadmin	詳細の表示 アラート受信者の追加
重大度: *	クリティカル	
ステータス: *	有効	
アラート間の最小時間: *	10 分	

以下に SLA を依然満たすことができるかどうかに基づいたサービス レベル違反アラートの実装例を示します。この例では、現在のサービス レベルとメトリックのビジネス トラッキング期間内の残りの時間が考慮されています。

例: 取り消し可能な SLA 違反アラート

このアラートは、目標（メトリック）が現在違反されている場合にトリガされます。しかし、最良のシナリオでは目標への準拠を意味するため、より適切なサービス レベルが今後提供された場合、この違反は取り消すことができます。

タイプ: サービス レベル予測

条件計算式

```
#Deviation>0 And #BestCaseDeviation<=0 And  
#BestCaseServiceLevel<>#WorstCaseServiceLevel And #ClusterItem = ""
```

メッセージ

契約の「#契約」の目標「#メトリック」が現在違反されているが、違反は取り消し可能であることを注意してください。

サービス レベル目標値が「#ターゲット」の間、以下のサービス レベルが予測されます。

- 現在までのサービス レベル: #サービスレベル (偏差: #偏差%)。
- 最悪のシナリオのサービス レベル: #最悪ケースサービスレベル (偏差: #最悪ケース偏差%)。
- 最良のシナリオのサービス レベル: #最良ケースサービスレベル (偏差: #最良ケース偏差%)。

第 4 章：インストールおよび展開

このセクションには、以下のトピックが含まれています。

[概要](#) (P. 222)

[準備](#) (P. 224)

[インストール](#) (P. 226)

概要

設定段階が完了し、システムの計算が正しいことが確認された場合は、システムを実稼働環境に展開する準備が整っています。この章では、この段階に必要な可能性がある手順すべてを網羅することを目指しています。これらの手順はインストール環境ごとに異なる可能性があります。ただし、実際の実稼働環境にシステムを統合するための準備がすべて行なわれたかどうかを確認するために、以下の項目をチェックしてください。

- 実稼働のハードウェアおよびソフトウェアが利用可能でインストールの準備ができています。
- メール サーバが社外/社内へのメール送信をサポートするように設定されている。
- FTP エージェントが存在する（データ ファイル転送に必要な場合）。
- データ ソース フィールドが利用可能で動作している。

インストールには次の 2 つの段階があります。第 1 段階（インストール段階）では、システム管理者が実稼働環境に **CA Business Service Insight** コンポーネントをインストールして統合します。第 2 段階では、システム機能が検査されます。また、システム プロセスおよび UI 機能がすべて正しく動作することを保証するために、システムをモニタ対象ステータス状態にする必要があります。

インストール処理中に、リモート接続ツールを介して実稼働サーバ上で作業することが必要になる場合があります。このリモート接続ツールは、完全に安全なインストールを可能にするために、アプリケーションと **Web** サーバにインストールしてください。

設定が正しく実行され、設定されている場合がある企業要件に適合することを保証するために、可能な場合、**Oracle DBA** のエキスパートが **Oracle** データベースのインストールを実行する必要があります。また、**CA Business Service Insight** のインストールで提供されるデータ作成ユーティリティを使用したデータベースの作成をシステム管理者に依頼することができます。この場合、データベースのインストールが正常に完了していることを保証するために、**DBA** がこのインストールを確認する必要があります。システムが開発システム上ですでに設定されていて、移行する必要がある場合は、そのデータベースの内容をこの新しい実稼働データベースに移行する必要があります。

前に完全なシステムがこの新しい実稼働環境に移行されている場合があります。このような場合、このシステム内に保存された計算済みデータおよび **Raw** データのすべてを削除し、システムを一から再ロードする（なお、サービス カタログ、リソース、および変換のすべてを適切に維持したままにしておきます）ことをお勧めします。これは、実稼働環境内のシステム全体の操作をテストするための優れた方法です。

要約すると、以下の手順がこの段階内には含まれています。

- システムをインストールし接続する。
- 必要な場合は、設定されたシステム データベースをロードする。
- 接続を電子メール、Exchange、SMS などに統合する。
- 機能をテストし、パフォーマンスを調整する。
- リモート接続のインストール、設定、およびテストを行なう。
- システムを再初期化して「本番」運用の準備を完了させる。
- データ アダプタをアクティブ化して、データ ソースのポーリングと Raw データのロードを開始する。
- CA Business Service Insight エンジンのスイッチを入れて、計算の開始を可能にする。
- 必要なドキュメント（システム管理、データベース、その他の保守の各手順など）すべての作成を完了する。
- トレーニングがすべてユーザに提供されたことを確認する。

準備

インストールを効率的に行なえるように、以下のことを事前に準備しておくことが重要です。

1. **開発環境のデータベースのエクスポート。** このデータベースのエクスポートは、CA 承認の各スクリプトを使用して実行する必要があります。これらのスクリプトでは、抽出（ダンプ）ファイルを作成して、実稼働システム上のソフトウェアにそのファイルをインポートし戻すことができます。

注: このエクスポートは十分な権限を持つデータベース ユーザが実行すること、およびこの同じユーザがデータベースへのインポート時にアクセス可能であることが重要です。このためには、「**oblicore**」アカウントを使用でき（このアカウントは各システムに必ず作成されるため）、また代わりに「**sys**」アカウントを使用することもできます。ただし、「**sys**」のパスワードがインポート処理の実行を可能にするターゲットのデータベースで利用可能であることを確認してください。

2. **データベース スクリプト - DBA** がデータベース作成スクリプトを変更したい場合、変更したスクリプトの CA の DBA による確認を事前に行なってください。インストールを順調に行なえるように、これらのスクリプトは事前に確認して準備しておく必要があります。データベース設定がローカル ポリシーに準拠することを保証するために、CA のデータベース管理者によって承認される必要があるコメントおよび変更を、多数の組織が保持していることはよくあることです。

- **サーバの接続性およびアクセシビリティ** - アプリケーションと Web サーバの両方にリモート接続クライアントをインストールしてください。これは、外部アクセシビリティ（インストール時または今後のサポートの問題のために物理アクセスが不可能でない場合）をサポートするためです。外部アクセシビリティは、設定期間中にシステムのモニタリングを継続的行なうことを可能にするためにも重要です。外部リモートアクセスが必要な場合、これは余分なセキュリティの影響のためにその構築に時間が長くなる場合があります。従って、この構築は事前に行なっておいてください。

リモート接続は、以下のツールのいずれかを使用して確立できます。

- PcAnywhere
- Proxy Host / Master
- Microsoft Terminal Server
- Remote Desktop
- VNC
- 組織によってサポートされる他の任意のリモート接続ツール。

注:

1. Microsoft Terminal Server は、PcAnywhere、VNC、Proxy、およびその他のツールとは異なり、サーバ ログインをシミュレートすることによりサーバに接続します。
2. 一部の Oracle ソフトウェアは、Microsoft Terminal Server を介してインストールできないため、ローカルでインストールしてください。
3. **データソースのアクセシビリティ** - マニュアル操作でデータソースにアクセスできる必要があります。また、アダプタがデータソースに接続できるように ODBC 接続を設定します。
4. **サーバのセキュリティ** - アプリケーションと Web サーバの両方で、ローカル管理者権限を持つユーザプロファイルが必要です。データベースプラットフォームが標準の Windows OS の場合、ローカル管理者権限を持つユーザプロファイルも必要です。UNIX プラットフォームについては、データベース管理者に、サーバ上にデータベースを作成するための適切な権限が必要です。
5. **インターネットへのアクセス** - これにより、システム管理者は、必要な場合に任意のオペレーティングシステムまたはアプリケーションの更新のために、インターネットに接続することができます。
6. **標準のユーザデスクトップ PC (アプリケーション機能のテスト用)** アプリケーションでは、インストールされるいくつかの ActiveX コントロールの使用が必要になることに注意してください。ActiveX コントロールは、多くの場合組織のセキュリティポリシーによって制限されています。
7. **関連スタッフのトレーニングのスケジュール** - スタッフがユーザ受け入れトレーニングを実施して、システムの使用を開始できるようにする導入用トレーニング

インストール

インストール処理は、「インストール ガイド」に詳細に述べられていますが、以下アクティビティが含まれます。

1. データベースのインストール

データベースのインストールは、データ ソース エキスパートまたは **Oracle DBA** が担当し、特殊な状況では **CA** のエンジニアの指示のもとに行なってください。データベースのインストールには、以下の手順が含まれます。

- **Oracle** のデータベース サーバへのインストール。
- **Oracle** パッチのデータベース サーバへのインストール（必要な場合 - **Oracle** データベース ソフトウェアの最新のサポート サービス リリースを必ず使用してください）。
- **Oracle** クライアントのアプリケーション/**Web** サーバへのインストール。
- **Oracle** クライアントのパッチのアプリケーション/**Web** サーバへのインストール（必要な場合 - バージョンがサーバと一致していることを必ず確認してください）。

2. **CA Business Service Insight** アプリケーションのインストール

アプリケーションのインストールは、システム管理者が実行します。インストールが統合およびアダプタのテスト中にテスト環境（設定段階時）ですでに実行されていた場合は、初期化状態またはデータベースのインポートのみが必要です。この後、実稼働環境内でのインストールが必要になる場合があります。

アプリケーションのインストールには、以下の手順が含まれます。

- **CA Business Service Insight** データベースの作成。
- アプリケーション サーバのインストール。
- **Web** サーバのインストール。
- アダプタのインストール。

最新の **CA** サービス リリースをアプリケーション/**Web** サーバに必ずインストールしてください。アプリケーションとサービス リリースのインストール中に、データベースがそのリリース用の正しい構成にアップグレードされることを保証するために、**SQL** スクリプトが組み込まれます。将来データベースを再度更新することが必要になった場合に備えて、これらは、すべてアプリケーションのインストールディレクトリ内に格納されます。（たとえば、データベースを前のサービス リリース上で構築されたバックアップからインポートする必要がある場合。この場合、最新のサービス リリース更新スクリプトを実行する必要があります。これは、データベースの構成がこの同じサービス リリースの一部としてインストールされるバイナリ コンポーネントに合わせることを保証するためです。

また、下位データの操作が必要となった場合に役立つ PLSQL Developer や代替 SQL ユーティリティなどの支援ツールを別途インストールすることもできます。

環境間でのインポート/エクスポート方法(データ ソース エキスパート)

ここでは、開発/テスト環境での設定とテストが終了したアダプタを実際の環境または実稼働環境に移行させる方法を説明します。実稼働環境で標準 CA Business Service Insight インストールがすでに終了しており、データベースが用意されていることが前提条件となります。

移行に伴う手順を以下に示します。

データベースをエクスポートして新しい環境にインポートする方法

1. 開発システム上での作業をすべて終了し、開発システムでエクスポート処理を実行するための論理ポイントを選択して実稼働システムに使用できるようにします。CA Business Service Insight サービス コンポーネントと CA Business Service Insight COM+ コンポーネントをすべてシャットダウンします。CA Business Service Insight 標準システム エクスポート スクリプトを使用してエクスポートを実行します。(必要な場合は、CA までお問い合わせください。)
2. 抽出されたデータベースをコピーを取り、インポートの準備が整っている実稼働サーバにそのコピーを貼り付けます。必ず開発システム上のデータベースと実稼働システム上のデータベースの Oracle バージョンが一致するようにしてください。標準システム インポート スクリプト (エクスポート スクリプトと併せて提供されているもの) を使用してデータベースをインポートします。
3. 完了したら、インポート上の問題がないかを確認します。何も問題がない場合は、最新サービス リリースの SQL スクリプトを実行していることを確認します (必要な場合)。
4. [リトル ウィザード] ショートカットを実行し、新しい実稼働システムの設定に合わせてデータベースを設定します。
5. CA Business Service Insight サービス コンポーネントをすべて起動し、実稼働システムにログインして、インポートが正しく実行されていることを確認します。

アダプタを移行させる方法

1. インポートするアダプタと同じ設定になっている **AdapterManager** ユーティリティを使用してアダプタをインストールします（名前が完全に一致していないと、次のステップが正しく実行されません）。
2. 開発システムからアダプタ設定ファイルをコピーして実稼働システムの [新規アダプタ] フォルダに貼り付けます。デフォルトで用意されている設定を上書きします。（ファイルが上書きされていることを確認する必要があります。上書きされていないと、名前が一致しなくなるため、実行中に問題が発生します。）
3. アダプタ設定ファイルを更新します。新しい環境に合わせて、**CA Business Service Insight** サーバおよびデータ ソースとの通信方法を更新する必要があります。正しいアダプタ ポートを反映するように **OblicoreInterface** セクションを更新します。必要な場合は、正しい **ConnectionString**、ファイル名パターン、またはパスを反映するように **DataSourceInterface** セクションを更新します。
4. 必要な **ODBC DSN** (**Data Source Name**、データ ソース名) がすべて設定されており、新しいマシン上で有効になっていることを確認します。
5. アダプタの接続性をテストします。
6. アダプタの実行をテストします。
7. **変換** - 変換スクリプトが用意されている場合は、これらのスクリプトをアクティブにしておく必要があります。これらのスクリプトがアダプタと同期し、変換が正常に実行されることを確認します。手動で変換を行っており、まだ完了していない場合は、それ以降の変換をこの段階で実行する必要があります。
8. **アダプタのスケジュール** - アダプタがスケジュールどおりに実行されるように設定します。アダプタが **RunOnce** モードでアプリケーションとして定義されている場合は、**Windows** のスケジューラを使用して、そのアダプタのスケジュールを設定できます。これを表示するには、[コントロール パネル] - [タスク] を選択します。詳細については、「[アダプタの実行モード \(P. 97\)](#)」を参照してください。

統合 - メール サーバの設定(システム管理者)

システムの通知機能を有効にするためには、どのメール サーバとメールボックスが **CA Business Service Insight** 電子メールの送信に使用されているかを認識しておく必要があります。メールは **CA Business Service Insight** サーバからこのメールサーバに **SMTP** として送信されるため、指定のアカウントを使用して、このメールサーバを中継点としてメールを送信できるようにしておく必要があります。メールサーバのセットアップを完了した後、**CA Business Service Insight** のアラートおよび契約承認機能で電子メール機能を使用できます。

管理者メニューをクリックし、[サイト設定]、[アラート] を選択します。アラート設定セクションで、電子メールサーバ、アドレスと送信者名の送信、SMS ゲートウェイを使用するための **SMS** プロバイダ情報などの電子メール定義を構成します。

統合テストの一環として、アプリケーションサーバが組織のメールサーバに到達可能であり、このメールサーバを使用して **CA Business Service Insight** アラートを送信できることを確認する必要があります。

メールサーバとアプリケーションサーバの間の接続をテストするには、アプリケーションサーバで以下のコマンドを実行します。

```
C:\Documents and Settings>telnet ORGANIZATION-MAIL 25
```

ORGANIZATION-MAIL

電子メールクライアントで定義したメールサーバを定義します。確認できない場合は、システム管理者に問い合わせ、この情報を入手してください。

Outlook クライアント メール サーバを確認する方法

1. **Microsoft Outlook** で [ツール] - [電子メールアカウント] を選択した後、[既存の電子メールアカウントの表示と変更] を選択します。
2. [変更] をクリックします。
3. **Microsoft Exchange** サーバをコピーします。このサーバはユーザの組織のメールサーバです。

このコマンドに対してサーバからの応答があった場合は、接続が正常に確立されていることを意味します。以下のような応答が表示されます。

これ以外のメッセージが表示された場合は、2 台のサーバ間に接続が確立されていないことを意味します。システム管理者にお問い合わせください。

システム プリファレンスの設定(システム管理者)

エンジン設定

アダプタ リスナ:	☐ 参考になるメッセージを表示
関連エンジン: ACE インスタンス数:	1
サービス レベル計算間隔: *	1440 分
計算完了時の動作: *	続行
スケジュール設定方法: *	標準
☒ 計算間隔を制限する	
最大連続計算期間: *	100 日
☒ 関連エンジンによって使用されるメモリの量を制限する	
メモリの解放を開始する上限値: *	800 MB.
メモリの解放を停止する下限値: *	400 MB.
ペナルティ ライタ:	
ペナルティの計算間隔 *	60 分
システム ログ消去:	
☐ タスク パラメータを初期化	
デバッグ メッセージの保持期間: *	3 日
情報メッセージの保持期間: *	15 日
警告メッセージの保持期間: *	15 日
エラー メッセージの保持期間: *	30 日
前回の有効化日付:	

システム プリファレンスの設定では、関連する値をシステム変数に適用します。
 [管理] メニューの [サイト設定] - [エンジン] をクリックして [エンジン プリファレンス] ダイアログ ボックスを開きます。

各パラメータの詳しい推奨事項については、「管理者ガイド」を参照してください。

セキュリティ設定(システム管理者)

セキュリティ設定では、ユーザ、ユーザ グループ、および役割を作成します。デフォルトですべてのユーザはアプリケーションのインストール処理時に指定された組織に関連付けられますが、必要に応じて別の組織を作成することもできます。

必要な定義のほとんどは設定段階ですでに完了しているため、通常は、それ以降に追加された設定を定義するための微調整だけが必要となります。

セキュリティ設定の詳細については、オンライン ヘルプまたは管理者ガイドを参照してください。

SSA 同期設定の指定

CA Spectrum Service Assurance (SSA) を使用して CA Business Service Insight 用のサービスを検出する場合には、自動同期を有効にする設定を行うことができます。自動化機能を使用すると、サービスのリストとサービス データを最新の状態に維持できます。

注: これらの設定を編集するには、SSA で restful API にアクセスできる必要があります。

以下の手順に従います。

1. [管理者] メニューの [サイト設定] - [SSA 設定] をクリックします。
[SSA 設定] ダイアログ ボックスが表示されます。
2. [SSA ユーザ認証] 領域に以下の情報を入力します。

サーバ URL

ターゲット SSA サーバの URL を指定します。

ユーザ ID

SSA サーバ用のユーザ ID を指定します。

パスワード

SSA サーバのユーザ ID に対するパスワードを指定します。

3. [同期オプション] 領域に以下の情報を入力します。

自動同期

同期の頻度に従って、同期が自動的に行われるようにします（次の段落を参照）。

同期頻度の設定

新しいサービスを検索するための頻度を指定します。時間単位または日単位で値を指定できます。

手動同期

ダイアログ ボックスから、手動による同期を有効にします。

4. [デフォルト データ オプション] 領域に以下の情報を入力します。

デフォルト サービス

検出されたサービスをデフォルトで管理対象にするか、管理対象外にするかどうかを設定できます。

比較メトリックの計算の有効化

SSA サービスの比較メトリックを設定するためのデフォルトアクションの設定を有効にします。

5. [OK] をクリックします。

SSA 設定がアクティブになります。

付録 A: サービスドメインおよびドメイン カテゴリの例

以下のテーブルは、一般的に使用される共通サービス ドメインおよびドメイン カテゴリをまとめたものです。

サービスドメイン	ドメイン カテゴリ	コメント
システム可用性	利用可能な時間の割合(%)	
	停止/ダウンタイムの回数	
	MTTR（平均修理時間）	
	MTBF（平均故障間隔）	
サービス可用性	ダウンタイム（分）	
	切断日数	
会計管理	サービス コスト	
プロセス性能	時間どおりに完了したプロセスの割合（%）	
	完了したプロセス サイクル数	
インシデント管理	解決したインシデントの割合（%） <T1	
	応答済みインシデントの割合（%） <T1	
	インシデント数	
	問題に転換されたインシデントの割合（%）	
顧客満足度	顧客満足度（%）	
	平均の CSI スコア	
ヘルプデスクの能力	破棄されたコールの割合	
	平均コール ピックアップ時間	

サービスドメイン	ドメイン カテゴリ	コメント
	FLLR の割合 (%)	(First Level Resolution Rate : 第一水準の解決の割合)
データ品質	正確性の割合 (%)	
	適時性の割合 (%)	
	エラー/欠陥の数	

付録 B: ケース スタディ例

このセクションには、以下のトピックが含まれています。

[契約モデリング例](#) (P. 235)

[会計メトリック モデリング例](#) (P. 248)

[データ モデリングの例](#) (P. 256)

[カスタム属性の使用例](#) (P. 268)

[変換スクリプトの例](#) (P. 272)

[ビジネス ロジック スクリプティングの例](#) (P. 278)

[効果的なビジネス ロジック例の作成](#) (P. 294)

[ケース スタディ 19: ファイル ベースのデータ ソースの場合のアダプタ ウィザード](#) (P. 312)

[ケース スタディ 21: LDAP との統合の例](#) (P. 328)

[ケース スタディ 22: PERL を使用してレポートを生成する](#) (P. 335)

契約モデリング例

以下のケース スタディのそれぞれで、説明された目的に対し以下の分類項目を使用します。

- サービス
- サービス ドメイン
- ドメイン カテゴリ
- タイムスロット
- ターゲット
- トラッキング期間
- 測定単位
- 計算アウトライン
- 契約/メトリック パラメータ

ケース スタディ 1: システム可用性

以下の契約上の保証内容を検討してみましょう。

「システム X は営業時間中、月の少なくとも 99.6% の間利用可能である必要があります。」

これは以下の CA Business Service Insight エンティティを使用して説明できます。

- **メトリック名**：システム X 利用可能時間の割合 (%)
- **トラッキング期間**：1 か月
- **タイムスロット**：業務時間（詳細な定義が後で必要）
- **ビジネス ロジック**：（利用可能な時間/総時間）*100

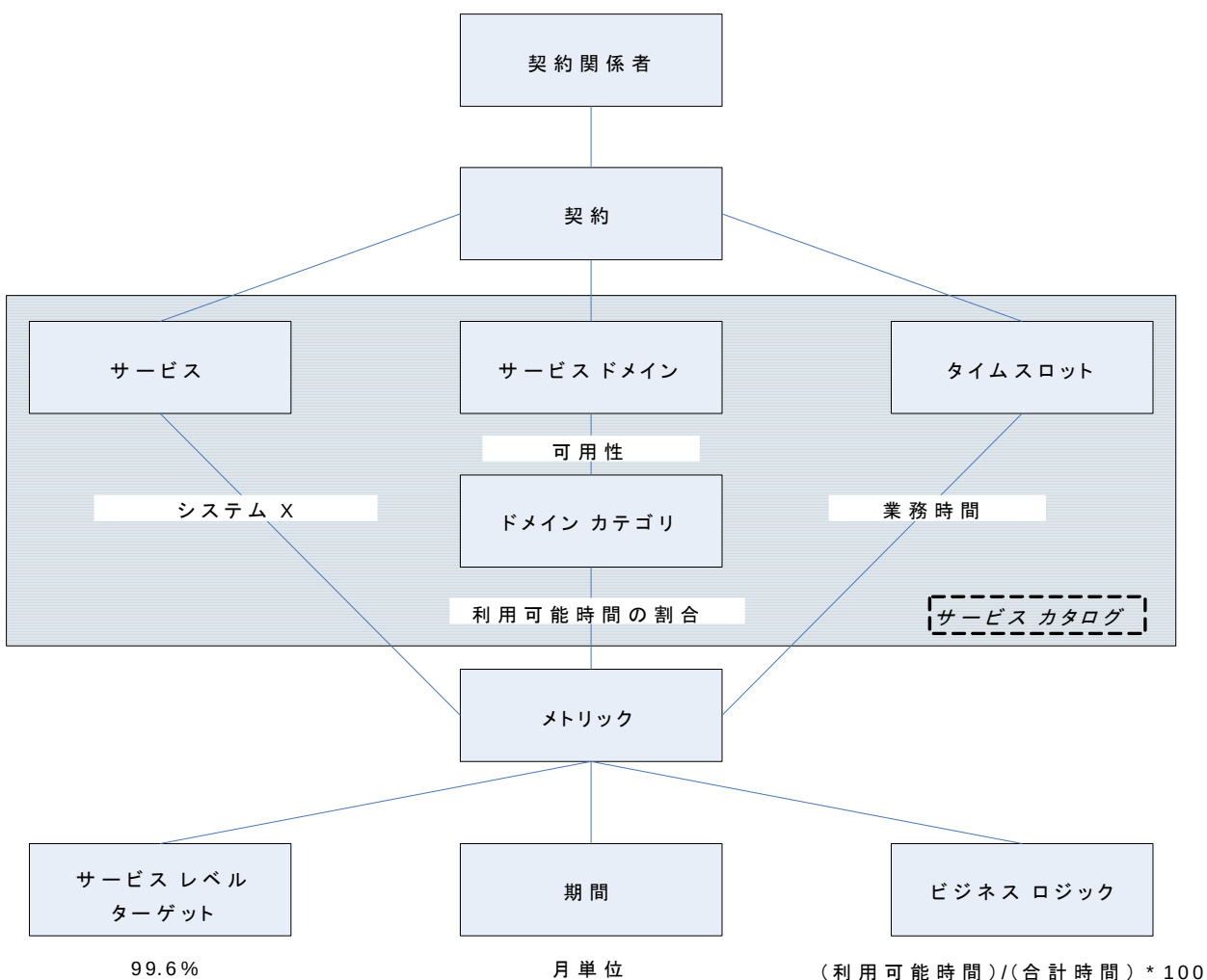
注: このケース スタディでは、モニタリングが業務時間内（メトリックのタイムスロット）のどこになるかのみに関係しています。

- **ターゲット**：99.6%

前のメトリック情報に加えて、システム サービス カタログからの以下の項目も上記の説明から推定できます。

- **サービス**：システム X。
- **サービス ドメイン**：可用性。
- **ドメイン カテゴリ**：利用可能時間の割合。
- **サービス グループ**：監視がある可能性がある複数のシステムを識別するすべてのグループ。この段階では、適切なグループを作成できるかどうか確定するのは困難です。

したがって、図中のこれらのエンティティを示す本ドキュメントの契約モデリングのセクションの図を再現できるようになりました。



ケース スタディ 2: システム可用性 2

CNP システム可用性は以下のレベルを維持する必要があります。

環境	平日	週末
運用	99.9%	98.9%
開発	90%	NA
試験/QA	保証なし	NA
ネットワーク	99.9%	NA

推奨ソリューション

メトリック	平日の CNP システム運用平均
ターゲット	99.9
トラッキング期間	1 か月
測定単位	%
サービス	CNP システム運用
サービス ドメイン	利用可能時間
ドメイン カテゴリ	アプリケーション可用性
タイムスロット	平日
メトリック	週末の CNP システム運用平均
ターゲット	98.9
トラッキング期間	1 か月
測定単位	%
サービス	CNP システム運用
サービス ドメイン	利用可能時間
ドメイン カテゴリ	アプリケーション可用性
タイムスロット	週末
メトリック	平日の CNP システム開発平均
ターゲット	90
トラッキング期間	1 か月
測定単位	%
サービス	CNP システム開発
サービス ドメイン	利用可能時間
ドメイン カテゴリ	アプリケーション可用性
タイムスロット	平日
メトリック	平日の CNP システム テスト/QA 平均
ターゲット	なし
トラッキング期間	1 か月

測定単位	%
サービス	CNP システムテスト Q/A
サービス ドメイン	利用可能時間
ドメイン カテゴリ	アプリケーション可用性
タイムスロット	平日
メトリック	CNP ネットワーク可用性
ターゲット	99.9
トラッキング期間	1 か月
測定単位	%
サービス	CNP システム ネットワーク
サービス ドメイン	利用可能時間
ドメイン カテゴリ	ネットワーク可用性
タイムスロット	常時

ケース スタディ 3: システム応答時間

以下のケース スタディでは、応答時間メトリックを説明します。1つの契約を複数の方法でモデル化できます。それぞれの方法に独自の利点があります。

以下の例では、さまざまなモデル化方法を確認します。

示唆されたモデリング ソリューション A	最大値
CNP システムのトランザクション応答時間	1 月当たり 750 ミリ秒を超えないこと
メトリック名	最長トランザクション応答時間
ターゲット	750
トラッキング期間	1 か月
測定単位	ミリ秒
タイムスロット	常時
サービス	CNP システム
サービス ドメイン	パフォーマンス

ドメイン カテゴリ

最長トランザクション応答時間

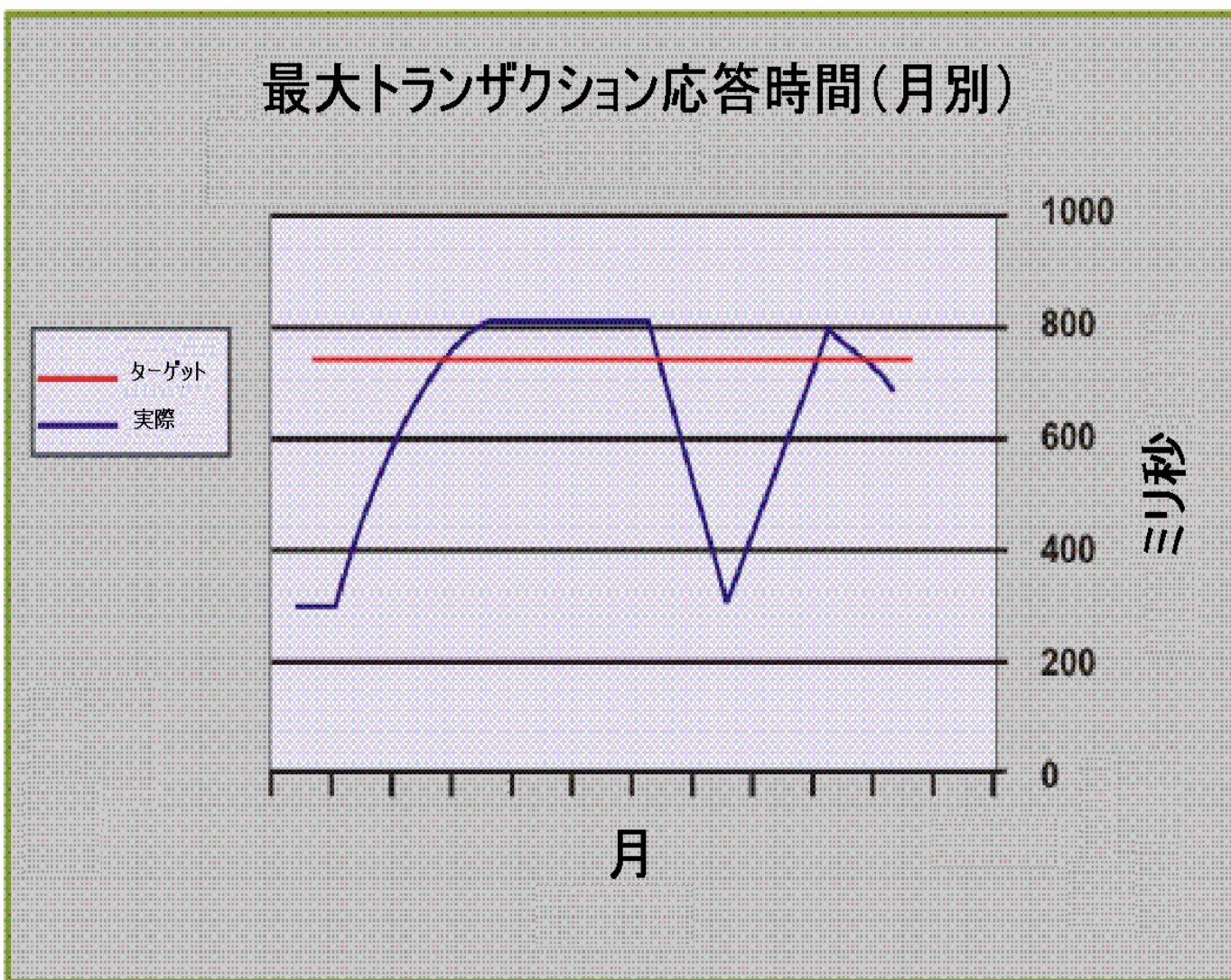
上記のマトリックスに基づいた場合、実際のサービス レベルはどのように算出されますか。

ドメイン カテゴリ定義に基づくと、実際のサービス レベルが最大値として算出されることが考えられます。つまり、1 か月間に実行されたすべてのトランザクションのうち、最大値が示されたトランザクションがキャプチャされ、この値が目標と比較されます。

注: サービス レベルの計算は、一定の期間内に収集された **Raw** データの総量に基づいて行われます。期間ごとにメトリックから **1** 件の結果が返されます。メトリックの目標は、単一のトランザクションと比較されるのではなく、その期間内に実行されたすべてのトランザクションを定期的に集計した月次結果と比較されます。契約マネージャは、この結果に契約が反映されるようにするだけでなく、サービス品質も反映されるようにする必要があります。

応答時間を最大値として測定することは、非常に厳重な義務であり、実際に達成することが極めて困難である点に注意してください。最大レベルの測定は、1 か月間に実行された **100** 万件のトランザクションのうち、**751** ミリ秒の単一のトランザクションが契約違反を引き起こすのに十分であることを意味します。このため、レポート内のバーはすべて赤色になり、提供されたサービスの実際の品質が反映されません。

このような状況における一般的なレポートを以下の図に示します。



目標を超えているトランザクションはすべて契約違反と見なされますが、サービス品質が劣化している場合に実際のサービス品質を把握するための基準になります。この理由は、単一のトランザクションしか反映されず、それ以外のトランザクションに関しては、単一の障害であるか傾向であるかなどの情報が何もわからないためです。これが単独のインシデントでない場合、1 か月間に何回トランザクションに失敗しましたか。また、1 か月間に実行されたトランザクションの総数のうち、失敗したトランザクションの占める割合はどのくらいですか。このような状況が発生したために契約違反となった月が複数ある場合、その傾向は何か。この状況は改善されていますか、それともさらに悪化していますか。このような質問はいずれもサービス レベル マネージャから寄せられる可能性があります、レポートは回答を提示できる必要があります。

注: メトリックとそれに関連した計算の概略を定義する場合は、レポートに結果がどのように表示されるかを想定しておくことが非常に重要となります。このレポートには、以下の 2 つの重要な項目を記載する必要があります。

- 契約違反の有無
- 失敗の根本原因を調査するのに十分なデータの透過性と能力。これ以外にも、サービス コンポーネントを完全に理解するための必須ツールもマネージャに提供する必要があります。

示唆されたモデリング ソリューション B	平均レスポンス時間
CNP システムのトランザクション応答時間	1 月当たり 750 ミリ秒以下であること
メトリック	平均トランザクション応答時間
ターゲット	750
トラッキング期間	1 か月
測定単位	ミリ秒
ドメイン カテゴリ	平均トランザクション応答時間

平均応答時間の計算では、毎月のサービス品質を十分に把握できます。また同時に、極端な応答時間や契約違反の応答時間が示された月を特定することもできます。

示唆されたモデリング ソリューション C	しきい値内で正常終了したトランザクションの割合
CNP システムのトランザクション応答時間	1 月当たり 750 ミリ秒以下であること
メトリック	正常なトランザクション応答時間
ターゲット	100
トラッキング期間	1 か月

測定単位	成功率
メトリック パラメータ	750 ミリ秒
サービス	CNP システム
サービス ドメイン	パフォーマンス
ドメイン カテゴリ	正常なトランザクション応答時間
タイムスロット	常時

この方法を使用すると、一定の期間内に 750 ミリ秒のしきい値を超えることなく正常に完了したトランザクションの割合が算出されます。この計算には、以下の数式が使用されます。

$$((750 \text{ ミリ秒未満のトランザクションの数}) / (\text{トランザクションの総数})) * 100$$

また、保証を成功率で表すことにより、厳重な保証（目標 100%）を維持できる一方で、サービス品質の良好度/劣悪度を示す実際の値を確保することもできます。

この方法を使用すると、目標が上限の 750 ミリ秒を超えないことなく、成功率を維持することになります。厳重な保証の維持が要求されるケースでは、たった 1 回の失敗の余地もない 100% の目標を設定する必要があります。このケースに対応するための変数が新規に導入されています（メトリック パラメータ）。必要に応じて簡単な変更ができるように、このパラメータをメトリック パラメータとして実装する必要があります。

この方法の代替モデルは、強制的にエスカレーション タイプ モデルになる場合があります。

以下のソリューションには、上記のソリューションのように 1 つのメトリックではなく、3 つのメトリックが定義されています。

メトリック	正常なトランザクション応答時間
ターゲット	95
トラッキング期間	1 か月
測定単位	成功率
メトリック パラメータ	750 ミリ秒
メトリック	正常なトランザクション応答時間
ターゲット	99
トラッキング期間	1 か月

測定単位	成功率
メトリック パラメータ	850 ミリ秒
メトリック	正常なトランザクション応答時間
ターゲット	100
トラッキング期間	1 か月
測定単位	成功率
メトリック パラメータ	1000 ミリ秒

契約上の義務だけでなく、750 ミリ秒のしきい値を超えているトランザクションの数も報告することが要求されているケースでは、失敗したトランザクションの数をカウントするメトリックを追加定義する必要があります。

注: 一定の期間内に各メトリックから生成される結果は1件です。トランザクションの割合を算出するように設定されている場合は、トランザクションの数も同時に算出することができません。

メトリックから追加レポートを生成するただ1つの方法は、ビジネス ロジックの出力を使用することです。(ビジネス ロジックから結果を出力する方法が記載されている「[出力 - ユーザテーブル](#) (P. 173)」を参照してください。)

メトリック	失敗したトランザクションの数
ターゲット	ターゲットなし
トラッキング期間	1 か月
測定単位	トランザクション数
メトリック パラメータ	750 ミリ秒
サービス	CNP システム
サービス ドメイン	パフォーマンス
ドメイン カテゴリ	トランザクション数
タイムスロット	常時

ケース スタディ 4: ヘルプデスク パフォーマンス

ヘルプデスクの状況を説明するケース スタディ

ヘルプデスクは、以下のすべてで **100%** の達成率を実現する必要があります。

チケット タイプ:	解決時間
優先度 1	1 時間
優先度 2	2 時間
優先度 3	4 時間

推奨モデリング、ソリューション A

メトリック	優先度 1 用の解決時間
ターゲット	100
トラッキング期間	1 か月
測定単位	成功率
契約パラメータ	解決時間マトリックス
サービス	ヘルプデスク
サービス ドメイン	ヘルプデスクの能力
ドメイン カテゴリ	チケット解決時間
タイムスロット	常時

上記のマトリックスは 3 つのメトリックに適用されます。各優先度については、同一カテゴリ内のすべての優先度について独自のメトリックが定義されます。

推奨モデリング、ソリューション B

Metric 定義は、ソリューション A に示された内容と同じです。

オプション 1:

サービス	ヘルプデスク
サービス ドメイン	チケット管理優先度 3
ドメイン カテゴリ	チケット解決時間
ドメイン カテゴリ	チケット応答時間
タイムスロット	常時

オプション 2:

サービス	ヘルプデスク
サービス ドメイン	チケット管理
ドメイン カテゴリ	優先度 3 チケット解決時間
ドメイン カテゴリ	チケット応答時間
タイムスロット	常時

ケース スタディ 5: システム バックアップ

バックアップは以下のように実行されます。

時間枠	バックアップの数
週単位	6
月単位	27
年 1 回	350

推奨ソリューション

メトリック	週ごとのバックアップ数
ターゲット	6
トラッキング期間	1 週間
測定単位	バックアップ
サービス	バックアップ
サービス ドメイン	バックアップ
ドメイン カテゴリ	週ごとのバックアップの数
タイムスロット	常時
メトリック	月ごとのバックアップ数
ターゲット	27
トラッキング期間	1 か月
測定単位	バックアップ
サービス	バックアップ
サービス ドメイン	バックアップ

ドメイン カテゴリ	週ごとのバックアップの数
タイムスロット	常時
メトリック	年ごとのバックアップ数
ターゲット	350
トラッキング期間	1 年
測定単位	バックアップ
サービス	バックアップ
サービス ドメイン	バックアップ
ドメイン カテゴリ	週ごとのバックアップの数
タイムスロット	常時

会計メトリック モデリング例

以下のケース スタディでは、会計モデリングの例を示します。

ケース スタディ 6: 契約/サービスの会計条件のモデリング

サービスまたは契約の会計条件のモデリングに使用されるメトリックには一般的に 3 種類あります。内部変数については以下のとおりです。

- 修正された価格コスト
- 消費コスト
- ペナルティ/インセンティブ チャージ

次の例を検討します。

新会社が発足し、電子メール サービスやそれに伴うメールボックスの設定と保守が必要になります。新しいスタッフが雇用されるので、メールボックスの数は明らかに増加します。契約 1 件あたりの電子メール サービスには、固定費 \$1000 と、1 月あたりに課金されるメールボックスごとの追加費用が発生します。メールボックス 1 つ当たりのこのコストは、以下のように段階的になっています。

メールボックスの数	メールボックス当たりのコスト
1-1,000	\$1.00
1,001 - 5,000	\$0.80

5,001+ \$0.50

したがって、追加されるメールボックスが多くなれば、追加に要するコストは低くなります（たとえば、1500 個のメールボックス作成には $(1000 \times \$1) + (500 \times \$0.80) = 1400$ ドル必要です）。このモデルを使用し、2 つのメトリックを作成して契約に反映できます。

- 電子メール サービス コスト（固定）
- メールボックス使用コスト（変動/従量制）

さらに、経営陣による、通年（2007）のスタッフ数の推計は以下のとおりです。この動向は、会社設立時の雇用と、それに続く他地域でのオフィス新設によるものです。

1 月	2 月	3 月	4 月	5 月	6 月	7 月	8 月	9 月	10 月	11 月	12 月
50	100	500	900	1600	1700	1800	2500	2600	3500	3600	5800

これらのメトリックをモデル化するには、以下の手順に従います。

以下の詳細を使用して、契約の固定費メトリック（価格アイテム タイプを使用）を作成します。

全般

詳細

メモ

クラスタ化

しきい値

関連メトリック

精度

目標ステータス

登録

ビジネスロジック

メイン インジケータ:

サービス:

サービス ドメイン: *

ドメイン カテゴリ: *

測定単位: *

タイムスロット: *

タイム ゾーン: *

サービス レベル目標値: 目標値 - 次の値以下

動的ターゲット: ☐

トラッキング期間: *

消費の単位:

単価:

予測計算:

予測:

測定ステータス: *

Email

Service Consumption

Number of Mailboxes

US Dollars

Always (週単位)

GMT

新規追加

情報の表示

1

月

Mailboxes

新規追加

新規追加

アクティブ

このサービスの契約への固定費を明記するためには、これをビジネス ロジック
(固定費は **Result** 関数から返される必要あり)へのパラメータとして実装します。
すると、このパラメータは、メトリックの「目標ステートメント」を通じて参照
できます。

全般	詳細	メモ	クラスタ化	しきい値	関連メトリック	精度	目標ステートメント	登録	ビジネスロジック

The [Email Service Consumption](#) Fixed Cost is [US Dollars](#) per [1 月](#)

ファイル名
パラメータ

パラメータの追加

パラメータの更新

SERVICE_COST:

最小値:

最大値:

精度:

この **Metric** に対するパラメータ値を返すことは、単に、**Result** 関数を通じてサー
ビス コスト値を返すことになります。

全般	詳細	メモ	クラスタ化	しきい値	関連メトリック	精度	目標ステートメント	登録	ビジネスロジック

```

Forecast = Null
End Function

Function Target
    Target= Null
End Function

Function Result
    If Parameters.IsExist("SERVICE_COST") Then
        Result = Parameters("SERVICE_COST")
    Else
        Result = Null
    End If
End Function
    
```

次に、変数価格メトリック（ここでも価格アイテム タイプを使用）を作成し、使用するメールボックス数の消費コストを確認します。このメトリックを'メールボックス消費コスト'と指定し、詳細を以下のように指定して作成します。

全般

詳細

メモ

クラス化

しきい値

関連メトリック

精度

目標ステートメント

登録

ビジネスロジック

メイン インジケータ:

サービス:

Email

サービス ドメイン: *

Service Consumption

ドメイン カテゴリ: *

Number of Mailboxes

測定単位: *

US Dollars

タイムスロット: *

Always (週単位)

タイム ゾーン: *

GMT

サービス レベル目標値: 目標値 - 次の値以下

\$

動的ターゲット: ☐

トラッキング期間: *

1

月

消費の単位:

Mailboxes

単価:

予測計算:

☒

予測:

測定ステータス: *

アクティブ

新規追加

証明の表示

このインスタンスでは、消費パラメータをメトリック詳細に入力する必要があります。これらは単価テーブルに入ります。メールボックス数と費用の上記テーブルのモデルを作成するには、メールボックスと単価の1つの上限に対する列を作成します。

列の追加

	名前	タイプ	インデックス	デフォルト
 	Mailboxes	10 進数	はい	
 	UnitCost	10 進数	いいえ	

次に、各段階に対する値を入力します。この場合、メールボックスの上限により、それに関連するコストブラケットが決定されます。3段階あるので、それに応じてテーブルに追加されます。

行の追加

	Mailboxes	UnitCost
 	1000	1.00
 	5000	0.80
 	99999999	0.5

これに加え、メールボックス消費の予測関数を実装します。具体的には、プリセットされた月次レイアウトを使用して予測テーブルを作成します。

列の追加

	名前	タイプ	インデックス	デフォルト
 	Consumption	10 進数	はい	
 	Jan-01-2011	10 進数	いいえ	
 	Feb-01-2011	10 進数	いいえ	
 	Mar-01-2011	10 進数	いいえ	
 	Apr-01-2011	10 進数	いいえ	
 	May-01-2011	10 進数	いいえ	
 	Jun-01-2011	10 進数	いいえ	
 	Jul-01-2011	10 進数	いいえ	
 	Aug-01-2011	10 進数	いいえ	
 	Sep-01-2011	10 進数	いいえ	
 	Oct-01-2011	10 進数	いいえ	
 	Nov-01-2011	10 進数	いいえ	
 	Dec-01-2011	10 進数	いいえ	

次に、これは、シナリオ説明で与えられたテーブルの値で満たされます。

	消費	01/01 /2007	02/01 /2007	03/01 /2007	04/01 /2007	05/01 /2007	06/01 /2007	07/01 /2007	08/01 /2007	09/01 /2007	10/01 /2007	11/01 /2007	12/01 /2007
1		50	100	500	900	1600	1700	1800	2500	2600	3500	3600	5800

これで、メトリック用の目標ステートメントを追加できます。この場合、単価テーブルおよび予測テーブルが派生元なので、パラメータ値は不要です。

全般	詳細	メモ	クラスタ化	しきい値	関連メトリック	格差	目標ステートメント	登録	ビジネスロジック

The [Email Service Consumption](#) cost is based upon the [Price Per Unit](#) table, and is measured in [US Dollars](#) per [1 月](#). The cost is accumulative as more mailboxes are added.

フィールド	測定単位:	US Dollars	↑
パラメータ	単位記号:	\$	↑
	タイムスロット:	Always (週単位)	↑
	タイムゾーン:	GMT	↑

最後に、以下のようにビジネス ロジックを完了します。

Option Explicit

```
Dim PPUmap1, PPUmap2, PPUmap3, PPUkey, FCmap, periodFC, TierPPU
Dim currentMonth, TotalMailboxes, MailboxesThisPeriod, TotalPrice
```

```
Sub OnRegistration(dispatcher)
    ' サンプル登録のみ
    dispatcher.RegisterByMetric "OnMailboxAddedEvent", "NewMailboxEventType", _
        "MailboxResource", "MONTH", "MetricName", "MetricContract", _
        "MetricContractParty"
End Sub
```

```
Sub OnLoad(TIME)
    ' 価格段階マップおよび予測マップの初期化
    Set PPUmap1 = Context.Field ("Price Per Unit")(1)
    Set PPUmap2 = Context.Field ("Price Per Unit")(2)
    Set PPUmap3 = Context.Field ("Price Per Unit")(3)
    Set FCmap = Context.Field ("Forecast")(1)
End Sub
```

```
Sub OnPeriodStart(TIME)
    ' 処理内容：ここにコードを追加して期間開始イベントを処理する
    currentMonth = GetMonth (time)
    If Context.IsInForecast Then
        periodFC = getForecastValue (currentMonth)
    End If
    MailboxesThisPeriod = 0
    TotalPrice = 0
End Sub
```

```
Sub OnPeriodEnd(TIME, isComplete)
    ' 段階的価格設定モデルを使用して、
    ' メールボックスすべての現在価格を計算
    ' 総コストを決定するために各段階を通過するので、
    ' これは累積的手法を使用しています。
    TotalPrice = getMailboxCost (TotalMailboxes)
End Sub
```

```
Sub OnTimeslotEnter(TIME)
End Sub
```

```
Sub OnTimeslotExit(TIME)
End Sub
```

```
Sub OnMailboxAddedEvent(eventDetails)
    MailboxesThisPeriod = MailboxesThisPeriod + 1
    TotalMailboxes = TotalMailBoxes + 1
End Sub
```

```

Function Forecast
    Forecast = getMailboxCost (periodFC)
End Function

Function Target
    Target = Null
End Function

Function Result
    result = TotalPrice
End Function

Function getforecastvalue(q)
    getforecastvalue = FCmap (q)
End Function

Function getmonth(time)
    ' この関数で月を取得
    Dim lTime
    lTime = Tools.GetLocaleTime(time)
    getmonth = monthname (datepart ("m", lTime), True) & _
        "-0" & datepart ("d", lTime) & "-" & datepart ("yyyy", lTime)
End Function

Function getMailboxCost(num_boxes)
    ' 関数は、段階的価格モデルを使用してメールボックスのコストを計算します
    Dim returnValue
    If num_boxes <= PPUMap1 ("Mailboxes") Then
        ' 最初の段階
        returnValue = num_boxes * PPUMap1 ("UnitCost")
        ' Out.Log "Tier1: " & num_boxes
    Else If num_boxes > PPUMap1 ("Mailboxes") And num_boxes <= PPUMap2 ("Mailboxes")
    Then
        ' 第 2 段階のみ
        returnValue = (PPUMap1 ("Mailboxes") * PPUMap1 ("UnitCost")) + _
            ((num_boxes - PPUMap1 ("Mailboxes")) * PPUMap2 ("UnitCost"))
        'Out.Log "Tier2: " & num_boxes
    Else If num_boxes > PPUMap2 ("Mailboxes") Then
        ' 第 3 段階
        returnValue = (PPUMap1 ("Mailboxes") * PPUMap1 ("UnitCost")) + _
            ((PPUMap2 ("Mailboxes") - PPUMap1 ("Mailboxes")) * PPUMap2 ("UnitCost")) +
            -
            ((num_boxes - PPUMap2 ("Mailboxes")) * PPUMap3 ("UnitCost"))
        'Out.Log "Tier3: " & num_boxes
    End If
    getMailboxCost = returnValue
    'Out.Log "Cost is: " & returnValue
End Function

```

注: このビジネス ロジック スクリプトは、（'予測'テーブルを使用した）予測計算および会計消費コスト結果の両方を処理します。どちらも、このメトリック用に定義された、'単価'に基づく段階的価格を計算する式 `getMailboxCost()` を使用しています。

データ モデリングの例

以下の 2 つのケースでは、基本的なデータ モデリング プロセスと、関連する場合がある細目のいくつかについて説明しています。

データ モデリング プロセスは契約モデリングプロセスの後に実行されるので、契約の保証内容を派生元とする計算要件は既知であり、契約モデリング プロセスの一部として確認されています。

データ モデルは、計算要件をすべて考慮に入れる必要があります。

以下の 2 つのケース スタディでは、データ モデリング プロセスを示すために単一または選ばれた少数の要件が選択されます。

ケース スタディ 7: サーバ パフォーマンス

これはサーバの性能用の典型的なケース スタディです。

以下のデータ ソース構造を考えます。

表示	サーバ	測定	タイムスタンプ
利用可能時間	Appserv01	1	03/01/2001 07:44
応答時間	Appserv01	354.6	03/01/2001 09:58
CPU 負荷	Dbserve02	83%	03/01/2001 12:12
利用可能時間	Appserv01	0	03/01/2001 14:26
CPU 負荷	Dbserve02	94.30%	03/01/2001 16:40
キャパシティ	Firewall01	10%	03/01/2001 18:54
応答時間	Dbserve02	476.89	03/01/2001 21:08
利用可能時間	Appserv02	1	03/01/2001 21:24
応答時間	Appserv01	774.4	03/01/2001 21:48
CPU 負荷	Dbserve01	83%	03/01/2001 21:52

上記に加えて、以下の計算要件があります。

各アプリケーション サーバの可用性を % で計算します。

各サーバの可用性は別々に計算される必要があります。そのため、単一のサーバの可用性を計算するには、この特定のサーバに対するイベントのみを受信する必要があります。さらに、データ ソースには、可用性の計算（応答時間、容量など）に関連しない他のパフォーマンス インジケータが含まれます。したがって、関連するサーバと同様に可用性インジケータもフィルタする必要があります。

CA Business Service Insight 内のフィルタ条件はイベント タイプとリソースなので、データ ソース値からのフィルタ条件をリソースとイベント タイプに変換します。

この場合、インジケータはイベント タイプを論理的に説明しているので、CA Business Service Insight 内のイベント タイプとして変換するには理想的な値です。可用性、応答時間、容量、CPU 負荷など、タイプ数は限られています。これは、サーバの可用性を計算するメトリックについては、登録は可用性のイベント タイプ用であることを意味します。

このように、サーバが多数ありそれぞれの可用性を計算する必要がある場合は、サーバごとにリソースとして定義される必要があることになります。その後、リソース グループ内部でグループ化する必要があります。また、メトリックはそのリソース グループの上にクラスタ化されます。

推奨モデリング

イベント名	可用性イベント。
動作	0 または 1 のステータスの変更としてレポート。
タイムスタンプフィールド	タイムスタンプ（データ ソース中ただ 1 つの時間フィールド）。
リソース フィールド	サーバ（データ ソース内に表示されるすべてのサーバは CA Business Service Insight リソースに変換されます）。
イベント タイプ フィールド	インジケータ（このフィールドのそれぞれの値は CA Business Service Insight 内のイベント タイプに変換されます）。4 つのイベント タイプがあります。
データ フィールド	測定値は 0 または 1（可用性レコード用のみ）です。

以下のリソース配置が定義される必要があります。

リソース タイプ属性	アプリケーション サーバ

契約関係者への割り当て アプリケーション サーバがそれぞれ契約関係者に割り当てられます。そこでは、関連するサーバはそのアプリケーションを実行します。これにより、すべてのサーバを取得する契約関係者に関する登録が可能になります。

サービスへの割り当て 上記と同じ。

リソース グループへの割り当て オプションです。クラスタ化が必要なところで、リソースをグループ化することが通常必要です。

最後に、上記のすべての定義に基づき、以下のようになります。

登録基準 クラスタ化メトリックについては、各サーバの可用性を個別に計算して、登録はリソース別に行われます。

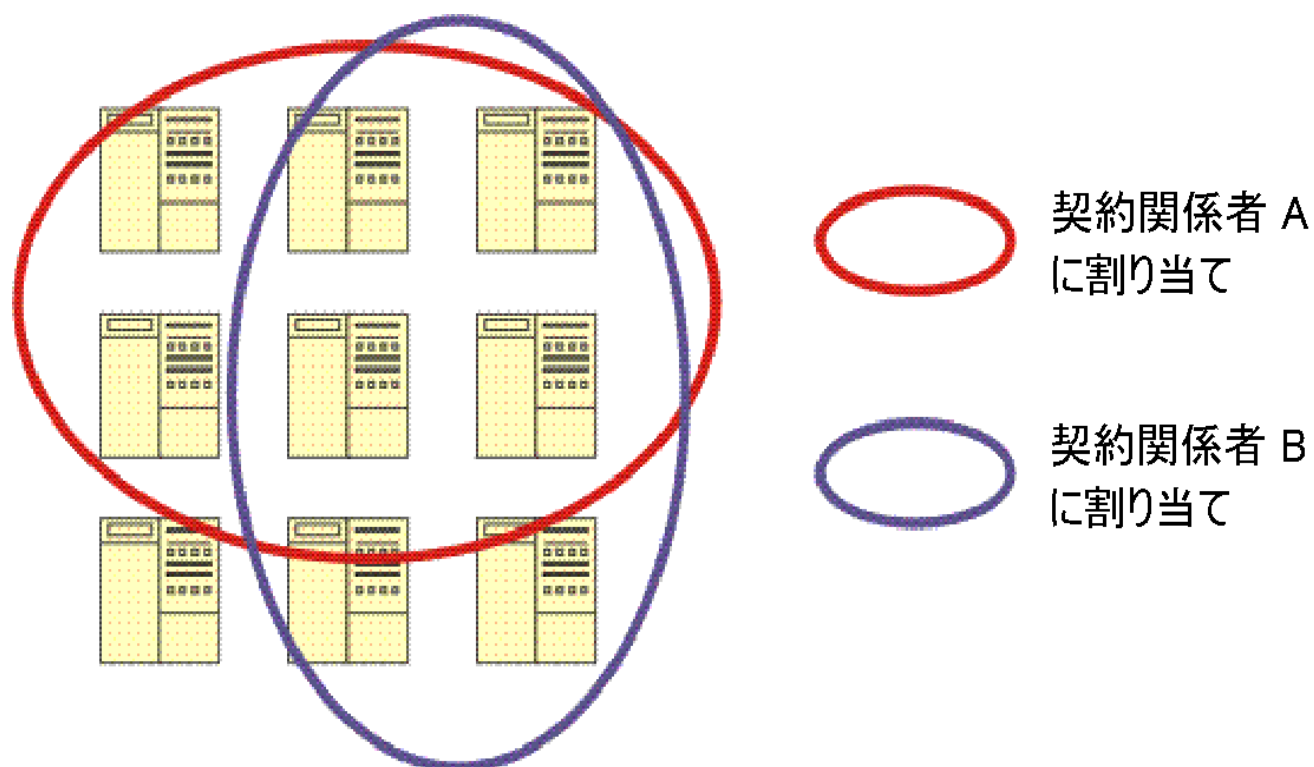
上記の要件を満たせるように、以下の基準が追加されます。

アプリケーション サーバの各契約関係者ごとの平均応答時間を別々に計算します。

この要件については、すべてのアプリケーション サーバに対する応答時間のイベントを受信する必要があります。このアプリケーション サーバは、特定の契約関係者向けにアプリケーションを実行するサーバのグループの一部です。応答時間イベントの受信は、"応答時間" 値を備えた表示フィールドから作成されたイベントタイプを登録することにより実行されます。これにより、サーバの応答時間についてレポートするイベントのみを受信することを保証します。

特定の契約関係者に関連するサーバについてレポートするイベントのみを受信するために、契約関係者割り当てによってリソースを登録する必要があります。

リソースは複数の契約関係者、サービス、またはグループに割り当てられます。そのため、契約関係者 A の契約の一部として応答時間の計算用に送信されたイベントが契約関係者 B 用の計算の一部でもあるといったことも起こります。



ケース スタディ 8: ヘルプデスクの能力

これはヘルプデスクの能力の典型的なケース スタディです。

データ ソースが以下のように与えられているとします。

TK 番号	TK. 優先 度	発生日時	終了日時	解決日時	Cust Ref	場所
3800968	1	19/12/2003 07:51	05/01/2004 11:31	22/12/2003 12:00	CM3	London
38000509	1	18/12/2003 09:21	05/01/2004 11:33	22/12/2003 12:00	CM1	Ipswich

TK 番号	TK. 優先 度	発生日時	終了日時	解決日時	Cust Ref	場所
38084199	2	07/01/2004 11:20	14/01/2004 09:10	09/01/2004 12:00	CM2	Ipswich
38188329	3	21/01/2004 09:27	27/01/2004 09:09	24/01/2004 12:00	CM3	Leeds
37964069	3	12/12/2003 14:04	05/01/2004 11:35	19/12/2003 12:00	CM286	Birmingham
3796448	1	12/12/2003 14:18	05/01/2004 11:39	19/12/2003 12:00	CM263	Luton
37965039	2	12/12/2003 14:57	14/01/2004 15:05	18/12/2003 12:00	CM264	Leeds
37970699	2	15/12/2003 09:26	05/01/2004 11:22	22/12/2003 12:00	CM288	London
37997259	1	17/12/2003 15:58	05/01/2004 11:27	23/12/2003 12:00	CM302	Ipswich
38000259	1	18/12/2003 09:11	06/01/2004 14:44	29/12/2003 12:00	CM340	London
38021049	1	22/12/2003 09:32	06/01/2004 14:28	31/12/2003 12:00	CM341	London

上記のデータ ソースには、顧客がサービスを提供するさまざまなロケーションで各顧客用に管理されるヘルプデスク チケットの詳細がリスト表示されています。単一のロケーションが顧客間で共有される場合もあります。

このデータ ソースを使用してレポートするためには以下の 3 つの要件が必要です。

- 顧客 CM3 に対し 4 時間以内に解決した優先度 1 チケットの割合 (%)。
- 各場所の顧客 CM3 に対し 4 時間以内に解決した優先度 1 チケットの割合 (%)。
- 各場所の顧客 CM3 に対し 1 日以内に終了した優先度 1 チケットの割合 (%)。

上記の要件は、イベントが次の項目によってフィルタされる必要があることを示しています。

- 優先度
- 顧客
- 場所

これらのフィルタ基準のどれがイベント タイプとして変換され、どれが関連リソースとして変換されるのか、指定する必要があります。

イベント タイプの選択方法

必要とされる 3 つのフィルタ条件のうち、以下のような理由で、イベント タイプに翻訳されることが最も適切なのはチケット優先度です。

- 処理されている（優先度 1 チケット）イベントの種類を説明していること。
- 存在する優先度の数が比較的少ない（優先度 1、2、3）こと。
- イベント タイプ自体は比較的一定（ヘルプデスクはチケットを処理する優先度をめったに変更しません）なこと。

リソースの選択方法

必要な他の 2 つのフィルタ基準は顧客と場所で、これらはレポートが必要な最も小さなエンティティです。そのため、顧客と場所の組み合わせはリソースです。

顧客と場所は比較的固定されたエンティティで、明確なライフ サイクルを持っています。それによって新しい顧客または新しいサイトが追加される場合があります。さらに、サイトと顧客の関係は変化する場合があります。

変換のために、データ ソースから複数のフィールドを使用することが可能です。前のケース スタディでは、サーバ フィールドは **CA Business Service Insight** リソースに翻訳されましたが、この場合、リソースは 2 つのフィールドの組み合わせを使用して構築されます。そのため、それぞれの並べ替えは新規リソースを作成します。

リソース リストを以下に示します。

カスタマ フィールド	サイト フィールド	出力リソース
CM3	London	CM3@London
CM1	Ipswich	CM1@Ipswich
CM2	Ipswich	CM2@Ipswich
CM3	Leeds	CM3@Leeds
CM286	Birmingham	CM286@Birmigham
CM263	Luton	CM263@Luton
CM264	Leeds	CM264@Leeds
CM288	London	CM288@London
CM302	Ipswich	CM302@Ipswich
CM340	London	CM340@London
CM341	London	CM341@London

出力リソースは、顧客フィールドとサイト フィールドを '+' 記号で組み合わせたものです。データ ソースから抽出され、後にレポートに表示されるので、リソースの名前を意識しておくことは重要です。レポート機能は期待に応える必要があります。

注: ヘルプデスクまたは任意のチケットまたはインシデント システムをモデル化するときによく見られる誤りは、単一のインシデントをリソースとして定義することです。

以下の正しくない仮定が誤りに結びつくことがよくあります。

1. 単一のインシデントは、レポートされるインシデントです。単一のインシデントが顧客サイトの計算の基礎とならないように、エンティティを、計算が生成される対象のエンティティとしてレポートされるように設定しません。一般的に、SLA は総合的業績に基づいており、個々がチケットを処理する能力に基づくものではありません。
2. 保証はチケット レベル別です。保証は定期的なものなので、これは誤りです。計算されるのは、時間経過に伴う総計です。

リソースに対する配置の設定

最初の計算要件として、以下の内容があります。

1. 顧客 CM3 に対し 4 時間以内に解決した優先度 1 チケットの割合 (%) 。
 - 特定の顧客のせいにされるチケットのイベントを受診する必要があります。顧客はこのリソースの一部で、また顧客サイトを示しています。そのため、リソースに関連し、その顧客に起因するイベントすべてをキャプチャするには、2つのオプションがあります。
 - データ ソース中の顧客が契約関係者を代表する場合は、そのリソースは関連する契約関係者に添付できます。これにより、契約関係者に関する登録が可能になります。可能な限り、これは常に好ましいことです。
 - データ ソース中の顧客それぞれに対するリソース グループを作成し、その内部の関連リソースすべてをグループ化します。この方法を使用して、登録はリソース グループ別に行われます。

次の 2 つの計算要件について検討します。

2. 各場所の顧客 CM3 に対し 4 時間以内に解決した優先度 1 チケットの割合 (%)。
3. 各場所の顧客 CM3 に対し 1 日以内に終了した優先度 1 チケットの割合 (%)。

これらの要件については、リソースをリソース グループに集めます。サイトごとに別々に計算が必要だという前提で、メトリックはクラスタ化する必要があるためです。

注: 現在のモデルおよび要件用のリソースの配置が必要でなくても、将来の要件に配慮して作成しておくことは重要です。そうすることにより、将来のシステム開発でのオーバーヘッドの発生を防止します。

Timestamp フィールドの選択

以前言及したように、Timestamp は相関エンジンがイベントを処理する方法にとって非常に重要です。メトリックは常に期間あたりのサービス レベルを計算し、この期間内に処理されるイベントは、そのタイムスタンプがその期間内にあるものです。

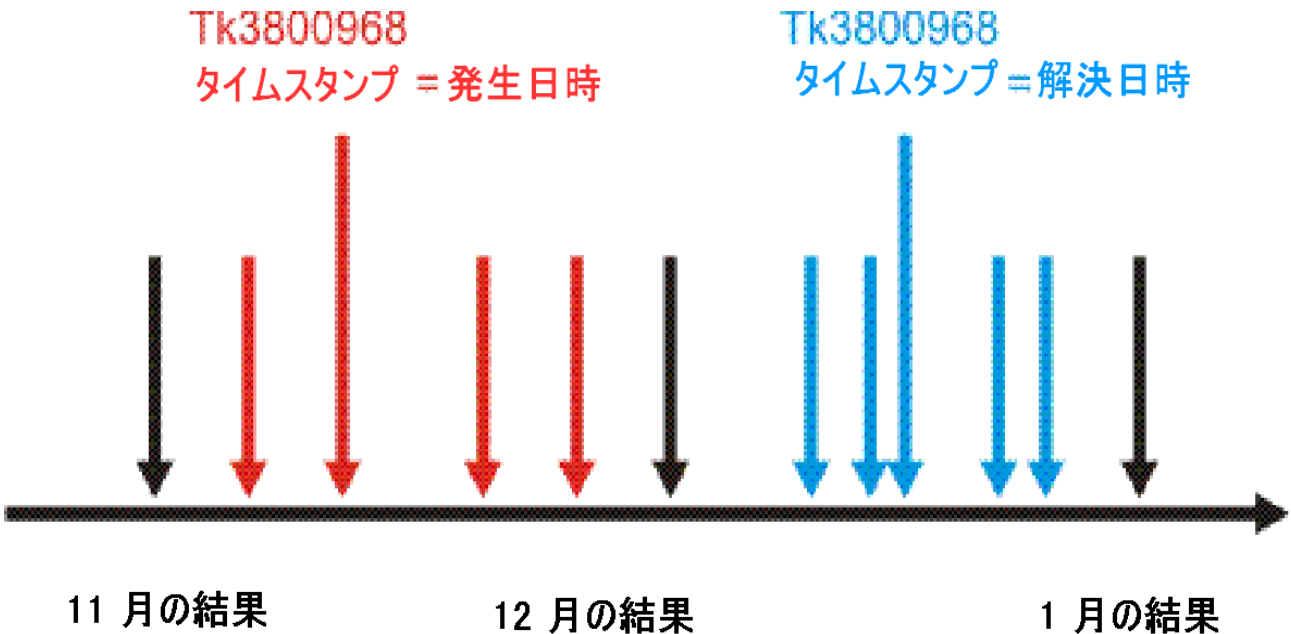
最初のケース スタディでは、データ ソースには時間のフィールドが 1 つしかありません。ただし、この場合、タイムスタンプとして設定可能な 3 つのフィールドがあります。最初の 2 つのレコードの検討

TK 番号	TK. 優先度	発生日時	終了日時	解決日時	Cust Ref	場所
3800968	1	19/12/2003 07:51	05/01/2004 11:31	22/12/2003 12:00	CM3	London
38000509	1	18/12/2003 09:21	05/01/2004 11:33	22/12/2003 12:00	CM1	Ipswich

解決時間を計算するには、発生日時および解決日時の両方が必要です。イベントとしては、1つのタイムスタンプのみを添付することができます。その後、他方は、値フィールド内の値として使用できます。

発生日時の値がタイムスタンプとして使用される場合、チケットは12月の結果に含まれています。「解決日時」の値がタイムスタンプとして使用される場合、チケットは1月の結果に含まれています。どちらの選択肢も有効です。選択内容は、チケットがレポートのどこに表示される必要があるかについて期待に応える必要があります。

それによってイベントがいつ計算に使用されるかが決まるので、実装にあたって非常に重要な点です。たとえば、チケットは11月に発生するが、12月まで終了しない場合、いつSLAの結果に貢献する必要がありますか。11月のデータに含めますか、それとも12月でしょうか。



この場合、チケットがデータソースにレポートされるのはクローズされてからのみなので、データがキャプチャされるのはチケットがクローズされてからのみです。通常、クローズされた日付は発生日時および解決日時のどちらよりも後の日付になります。発生日時がタイムスタンプとして選択された場合は、12月の結果の一部として処理される必要があります。終了日付は1月でしたから、このチケットが報告されたときにはすでに12月は過ぎていたことになります。従って、12月の結果はすでに発行済みです。その後、相関エンジンは、タイムスタンプが12月であることから過去のイベントであることに気づき、再計算をトリガします。そのため、12月の結果にさかのぼって変更されます。

どの時間フィールドがタイムスタンプとして選択される必要があるか定義するためには、これらの結果を完全に理解する必要があります。通常、クローズ日付は、遡及して変わらない最終レポートを持つために選択されます。

他方、タイムスタンプとしてクローズ日付を使用することは、チケットが計算に入るのを遅らせます。解決されたチケットは、相当な時間経過後にのみクローズされる場合があります。

ただし、このクローズ日付の利用は、組織内でチケットの終業時間を早めるプロセスのトリガとなる場合があることにも注意する必要があります。

したがって、最終提案は次のようになります。

イベント名	優先度 1 チケット（必要に応じて他の優先度に対しても定義可能）
動作	チケットがいつクローズされるかをレポートしました
タイムスタンプフィールド	終了時間
リソースフィールド	カスタマフィールド+サイトフィールド
イベントタイプフィールド	優先度
データフィールド	すべて
リソースタイプ属性	契約関係者サイト
契約関係者への割り当て	サイトはそれぞれ関連する契約関係者に割り当てられます。
サービスへの割り当て	上記と同じ
リソースグループへの割り当て	リソースグループは、クラスタリングを有効にするために契約関係者ごとに作成されます。
登録基準	クラスタ化メトリックについては、登録はリソース別で、メトリックは Customer CM3 サイトと呼ばれるリソースグループに結びつけられています。 クローズ時間メトリックについては、登録は契約関係者およびサービス別です。

カスタム属性の使用例

以下のケース スタディは、動的な複数ターゲットの例を示します。

ケース スタディ 9: 動的な複数ターゲット

顧客環境のハードウェア インフラストラクチャ デバイスすべてについて、可用性要件に対して個別のターゲットが設定されているシナリオ例を考えます。標準モデリング方法を使用すると、これは非常に達成困難なタスクで、リソース モデルを使用するデバイスおよび管理に対し、多数の論理グループ化を伴います。複雑さを追加するためには、これらのデバイスのターゲットは時間と共に変化できます。これらの目標値は、詳細が外部 **CMDB** に格納されているので、変換スクリプトによって **CA Business Service Insight** 内で更新されます (変換スクリプトの例については、「[変換スクリプトのベスト プラクティス \(P. 272\)](#)」を参照してください)。

このインスタンスでは、メトリックは以下のものが考えられます。

ハードウェア デバイス当たりの可用性の割合 (%)。

これを効果的にモデル化する 1 つの方法は、他の重要な機能の 1 つである '動的ターゲット' と共に 'カスタム属性' を使用することです。これらは両方ともクラスタ化メトリックと共に使用し、目標とする結果を実現できます。サービス レベル目標値をリソースに直接、追加することにより、ビジネス ロジックでリソースごとのサービス レベル (ハードウェア デバイス) を比較できます。クラスタ化メトリックは、単一のメトリックを使用して、ハードウェアの各部分に対する個別のサービス対応規格を提供します。

そのため、カスタム属性をこれらのデバイスのリソース タイプに追加することにより、最初にカスタム属性を作成する必要があります（ここで、すべてのデバイスは 'Infrastructure Device' タイプのリソースです）。作成されたカスタム属性は、'DeviceTarget' と呼ばれ、メニューの [サービス カタログ] - [カスタム属性] から追加できます。カスタム属性を作成するとき、その属性を必要としているリソース タイプにリンクする必要があります。

カスタム属性詳細

名前: * DeviceTarget

説明:

値タイプ: * 数

デフォルト値:

添付ファイル:

新規追加

エンティティ	タイプ	必須	デフォルト値
リソース	Infrastructure Devices	はい	

システムでリソースを表示するとき、新規カスタム属性がリンク先のリソース タイプで利用可能なのを見ることができます。

リソース

検索

13 件中 1 - 13 件を表示します。
(制限数 1000)

結果の適用先 2011/07/04 23:44

新規追加

カスタム属性の追加

名前	作成日時	リソース タイプ	リソース グループ	サービス	契約関係者	DeviceTarget	DialingPrefix
Equity Traders-Trans Exec	2011/07/04 14:20	TTickets			ACME Equibest Traders ACME IT		
Frankfurt	2011/07/04 14:26	Locations	Locations				+49
Hong Kong	2011/07/04 14:24	Locations	Locations				+852
InfraServers	2011/07/04 14:10	Infrastructure Devices					
Locations	2011/07/04 14:15			ECM Connectivity	ACME ESSIT		
Lodon	2011/07/04 14:22	Locations	Locations	Eagle App			+44
New York	2011/07/04 14:32	Locations	Locations	Eagle App			+1
Server111	2011/07/04 14:09	Infrastructure Devices					
Server222	2011/07/04 14:16	Infrastructure Devices					
Server333	2011/07/04 14:17	Infrastructure Devices					
Server444	2011/07/04 14:17	Infrastructure Devices					
Tokyo	2011/07/04 14:25	Locations					+81
USB Deutschland AG-UBS CMP	2011/07/04 14:30	TTickets		Trabsactuib Execution	ACME EMEA ACME ESSIT ACME USA Inc.		

また、個々のリソースには更新可能な新しいフィールドがあります。

リソース詳細

The screenshot displays the 'リソース詳細' (Resource Details) page. It features a sidebar with '全般' (General) and '詳細' (Details) tabs, and a '日付' (Date) field set to '2006/01/01 00:00'. The main content area includes several sections for resource configuration:

- 有効:** A dropdown menu set to 'はい' (Yes).
- リソース タイプ:** A selection box labeled '選択済み' (Selected) containing 'Infrastructure Devices'. To its right is a '利用可能' (Available) list containing 'DeviceTarget', 'Locations', and 'TTickets'. A button 'リソース タイプの追加' (Add Resource Type) is located below this section.
- リソース グループ:** A selection box labeled '選択済み' (Selected) containing 'InfraServers'. To its right is a '利用可能' (Available) list containing 'Locations'.
- サービス:** A selection box labeled '選択済み' (Selected) containing 'Eagle App'. To its right is a '利用可能' (Available) list containing 'ECM Connectivity', 'Email', 'Email2', and 'Trabsactuib Execution'.
- 契約関係者:** A selection box labeled '選択済み' (Selected) is empty. To its right is a '利用可能' (Available) list containing 'SMI', 'ACME EMEA ACME ESSIT ACME USA Ir', 'ACME Equitiest Traders ACME IT', and 'ACME ESSIT'.
- DeviceTarget:** A text field.
- DialingPrefix:** A text field containing '98'. This field is circled in red, and a red arrow points to it from the text 'カスタム属性の追加' (Add custom attribute).

この例では、このフィールドは通常、変換スクリプトによって挿入/更新されます。

それぞれのリソースには、それに対して指定されたターゲットがあるので、必要な計算を実行するためのロジックを開発できます。以下のサンプルコードは、リソース（太字）からカスタム属性を抽出する方法を示しています。

Option Explicit

```
Dim TotalTime
Dim OutageTime
Dim PeriodStart
```

```
Sub OnRegistration(dispatcher)
    dispatcher.RegisterByResource "OnDeviceOutageEvent", "DeviceOutageEvent", _
        Context.ClusterItem
End Sub
```

```
Sub OnLoad(TIME)
    TotalTime = 0
    OutageTime = 0
End Sub
```

```
Sub OnPeriodStart(TIME)
    TotalTime = 0
    OutageTime = 0
    PeriodStart = TIME
End Sub
```

```
Sub OnPeriodEnd(TIME, isComplete)
    TotalTime = Tools.NetTime(PeriodStart, TIME)
End Sub
```

```
Sub OnDeviceOutageEvent(eventDetails)
    OutageTime = OutageTime + Tools.NetTime (eventDetails ("OutageStartTime"), _
        eventDetails ("OutageEndTime"))
End Sub
```

```
Function Target
    Target = eventDetails.CustomAttribute ("DeviceTarget")
End Function
```

```
Function Result
    If TotalTime > 0 Then
        Result = (TotalTime - OutageTime) / TotalTime
    Else
        Result = Null
    End If
End Function
```

変換スクリプトの例

以下のケース スタディには、基本的な自動変換およびリソース モデル更新の例が含まれています。

ケース スタディ 10: 基本的な自動変換

ここで提供される変換スクリプト例は、[変換エントリ] 画面の [保留中エントリ] を処理する、かなり単純な場合です。

OnTranslationEvent ハンドラは、リソースの最初の文字に対する単純な確認を実行し、その値によってアクションを実行します。最初の文字が 'a' の場合、変換エントリは 'ignore' に設定されます。'b' の場合、削除されます。'c' の場合、変換されます。その他の場合は、変更されないまま手動で変換されます。コード全体にわたるカウンタが、スクリプト実行中に何のアクションが実行されるかを記録します。これは、実行されるごと、特にスクリプトが自動化されているときに、デバッグまたはスクリプトの実行の文書化に非常に役立ちます。関数の最後の **Tools.Commit** コマンドを忘れないようにするのは非常に重要です。このコマンドがないと、スクリプトによる変更はデータベースに保存されません。

TranslateResource () 関数が呼び出されると、(E2E プレフィックスと共に) 保留中の変換エントリにより引き渡されたのと同じ名前のリソースがシステムに存在するかどうかを確認します。存在しない場合は、スクリプトはこのリソースを追加してから変換を実行します。すでに存在している場合は、そのリソース文字列から既存の **CA Business Service Insight** リソースに変換エントリを作成します。

スクリプトの最後の **Result** 関数は、単にスクリプトによって実行されたタスクの説明を出力します。コードを以下に示します。

```
Option Explicit
```

```
dim translated
dim ignored
dim deleted
dim manually
dim ActionDate
```

```
Sub OnLoad()
    'tools.log "Translation Script: In OnLoad procedure", "I"
End Sub
```

```
Sub OnTranslationEvent(entryDetails)
    Dim dump
    dump = entryDetails.Dump
    tools.log dump

    Dim resource, entryId
    entryId = entryDetails.EntryId
    resource = entryDetails.FieldValue(1)
    ActionDate = entryDetails.LastActionDate

    If mid(resource,1,1) = "a" Then
        tools.IgnoreEntry entryId
        ignored = ignored + 1
        tools.log "ignored" & entryId & " " & resource
    Else If mid(resource,1,1) = "b" Then
        tools.DeleteEntry entryId
        deleted = deleted + 1
        tools.log "deleted" & entryId & " " & resource
    Else If mid(resource,1,1) = "c" Then
        TranslateResource resource, entryId
        tools.log "translated" & entryId & " " & resource
    Else
        tools.SetManualTranslationEntry entryId, 1
        manually = manually + 1
        tools.log "manually" & entryId & " " & resource
    End if

    Tools.commit
End Sub
```

```
Sub TranslateResource(resource, entryId)
    Dim newName
    Dim vector
    newName = "E2E-" & resource
```

```
if NOT tools.IsResourceExists(newName) Then
    Dim resourceDetails
    set resourceDetails = tools.GetResourceDetailsByDate(resource,ActionDate)
    resourceDetails("ResourceName") = newName
    resourceDetails("ResourceTypes") = "E2E Transactions"
    tools.AddResourceByMap resourceDetails
end if

tools.TranslateEntry entryId, newName
End Sub

Sub Main()
end Sub

Function Result
    Result = translated & "entries were translated, "& _
        ignored & "entries were ignored," & _
        deleted & "entries were deleted and "& _
        manually & "entries were set to manually update!"
End Function
```

ケース スタディ 11: リソース モデルの更新

変換スクリプトの別の利用法は、外部データ ソースで CA Business Service Insight リソース モデルを更新することです。厳密にはもはや変換とは関連しませんが、この例は、システムの自動更新として非常に価値のある機能です。

カスタム属性についての前のセクションでは、組織のハードウェア インフラストラクチャ デバイスが、各デバイスに対する可用性目標と共に外部 CMDB に格納されているシナリオを説明しました。インフラストラクチャ マッピング（およびそのデバイス目標）を常に最新にしておくために、この情報は CA Business Service Insight リソース モデルに複製しておく必要があります。

この場合、スクリプトは以下のタスクを実行する必要があります。

- システムに現在、存在しない新しいハードウェア デバイスを追加します。
- 既存のデバイスの予想される可用性目標を更新します（目標が変更された場合）。

スクリプトを以下に示します。

Option Explicit

```

'*****
'Global Variables and constants
'*****

Dim added
Dim updated
Dim ChangeSetName

added = 0
updated = 0

Const RESOURCE_TYPE = "Infrastructure Devices"
Const RESOURCE_GROUP = "InfraServers"
Const CHANGESSET_NAME = "Infrastructure Devices"
Const CHANGESSET_EFFECTIVE_DATE = "01/01/2007"

'*****
'Sub OnLoad :
'Preparing the foundation infrastructure enteties
'*****

Sub OnLoad()
    Tools.log "Translation Script : In OnLoad procedure", "D"

    'Search for existing preliminary resource version
    Dim ChangeSetMap

```

```
Set ChangeSetMap=Tools.SearchChangeSets(CHANGESET_EFFECTIVE_DATE,
CHANGESET_NAME)

'If no existing version create a new version
If ChangeSetMap.EMPTY Then
    Tools.AddChangeSet CHANGESET_NAME, CHANGESET_EFFECTIVE_DATE, CHANGESET_NAME
    Tools.Log "Changes set '" & CHANGESET_NAME & "' added."
End If

Set ChangeSetMap = Nothing
End Sub

Sub OnTranslationEvent(EntryDetails)
End Sub

Sub Main()
    Dim conn, rs, resource, deviceTarget, resource_id, resMap, custAttrib,
    custAttribValue
    Set conn = CreateObject("ADODB.Connection")
    Set rs = CreateObject("ADODB.RecordSet")
    conn.open "DSN=HardwareDevices"
    rs.Open "select * from tblServers", conn

    Do While Not rs.EOF
        resource = rs("serverName")
        deviceTarget= rs("DeviceTarget")
        'Add resources to latest version if it doesnt exist already
        If Not Tools.IsResourceExists(resource) Then
            resource_id = Tools.AddResource(resource, CHANGESET_NAME, "Infrastructure
Device", RESOURCE_TYPE, RESOURCE_GROUP, Null, Null)
            Tools.UpdateResourcesCustomAttribute resource, CHANGESET_NAME,
"DeviceTarget", deviceTarget
            Tools.Log "AddingResource: " & resource & " with target: " & deviceTarget
& " ; assigned ID= " & resource_id
            added = added + 1
        Else
            Set resMap = Tools.GetResourceDetails(resource,CHANGESET_NAME, False)
            Set custAttrib = resMap("CustomAttributes")
            custAttribValue =
CDBl(custAttrib("DeviceTarget")("CustomAttributeValue"))
            If CDBl(deviceTarget) <> custAttribValue Then
                Tools.UpdateResourcesCustomAttribute resource, CHANGESET_NAME,
"DeviceTarget", deviceTarget
                Tools.Log "Updating Resource target for : " & resource & " from: " &
deviceTarget & " to " & custAttribValue
                updated = updated + 1
            End If
        End If
    End While
End Sub
```

```
Tools.Commit
rs.MoveNext
ループ

rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
End Sub

Function Result
' Commit the transaction
Tools.CommitChangeSets CHANGESET_NAME
Result = added & " resources added and " & updated & " resources updated."
End Function

'*****
```

ビジネス ロジック スクリプティングの例

ビジネス ロジック スクリプティングについて、いくつかの一般的なガイドラインを以下に示します。

グローバル変数

- 宣言するグローバル値は確実に初期化します。PSL 状態メカニズムは、ヌル値の変数を保存できません。

コーディング一般

- 下にリスト表示されたオブジェクトのいずれかを使用する前に、（適切な `IsExist` メソッドを使用して）それが存在することを確認します。
 - すべてのタイプのパラメータ（たとえば `table`、`list` など）
 - カスタム属性
 - リソース
- ビジネス ロジック モジュールには必要なパラメータすべてが提供されていることを確認します。
- リソース名を変更する前に、それを使用しているメトリックを確認し、適宜更新します。

マップ

- クラスタ化メトリック内でのマップの使用。マップは、エンジンの計算能力をいっそう必要とします。メトリックについてクラスタ化していると、クラスタアイテム数をかけただけの能力が必要です。
- クラスタ化メトリックのビジネス ロジック内の大きなグローバル マップは、熟慮してからのみ使用する必要があります。クラスタ化メトリックを計算する一方で、エンジンはクラスタ内の各アイテムの状態から、別々にグローバル変数をロードするのにビジー状態になります。
- それらが完了した後、マップとベクターを必ずクリアします。
- 大きなマップを使用する必要がある場合、論理的な範囲に分割することによりマップを効率よく管理することを確認します。

ケース スタディ 12: カウンタ ロジックの障害数の計算への使用

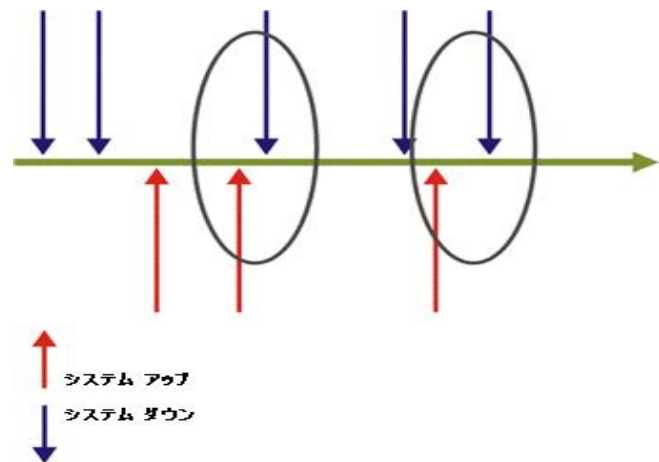
以下の例では、所定の計算期間内に発生した障害の数を計算します。この式とその実装に使用された方法は、何かを計算する際には常に必要になる公式の例と考えることができます。

計算には以下の仮定が使用されてます。

- 入力イベント
 - 可用性イベント、ステータス (0/1)
 - 可用性イベントは数分おきに到着します。イベントの頻度は予測できず (5 分ごとに発生する場合もあれば 1 時間に 1 度発生する場合もあります)、また不要なイベントが発生する場合があります (0、0、0、1、1、0 など)。
- 時間帯 - 就業時間帯。この時間帯外に発生した障害はカウントされません。
- 必要な結果は、期間中に発生した障害数です。

計算期間中に発生した障害を集計するには、周期的なカウンタ変数とシステム状態を格納する変数を保存する必要があります。不要なイベント情報を受信する場合もある (つまり、'Up' イベントのすぐ後に別の 'Up' イベントが続く) ので、システム状態が 'Up' から 'Down' に変化した場所の数をカウントし、毎回 'Down' イベントを受信されたときは、すでにカウント済みの障害を表している場合があるのでカウントしないことも必要です。

以下は、カウンタシステムの稼動時間と休止時間をカウントするシステムを図示したものです。



考慮する必要のある重要なポイント

- **定数** - コード中で定数値を使用するのではなく、定数定義を使用することが推奨されます。このようにして、値を変更する必要がある場合は、定数定義のみで変更するのが簡単で、コード中を探し回る必要がありません。さらに、必要に応じてパラメータ使用に変更するのは簡単です。上記の例では、イベントで受信されたシステム状態を示す値は、コードを理解しやすくするため、また、システム状態を表すのではなくカウント目的で数字のゼロが使用されるときと区別するために、定数として定義されています。
- **グローバル変数**
 - **カウンタ** - カウンタ変数の定義はグローバル定義です。式では、コードの先頭、かつサブルーチンまたは関数の外側で宣言されています。この場合、カウンタ変数をグローバル変数として定義することは重要です。これにより、コード内の複数のサブルーチン/関数中で使用できるようになり、また、計算期間中その値を保持し、期間の最後にその結果を提供することが可能になります。

この例では、カウンタ変数はコード中の 3 か所で使用される必要があります。

 - 期間の初めに初期化されること。
 - イベント ハンドラ内の失敗したイベントの場合には進められる必要があること。
 - 期間の最後に結果関数によって出力されること。
 - **システムの前の状態** - この変数にはシステムの状態が保持され、システム状態を備えた新しいイベントが受信されるたびに更新されます。この変数は、コード中の次のようなサブルーチン/関数でも使用されていることからグローバルである必要があります。
 - 計算の最初に初期化されること。
 - 新規イベントが受信されるときに更新されること。
- **時間帯に関する考慮事項** - 障害が発生したのが時間帯の期間内かどうかを確認するために `context.IsWithinTimeSlot` プロパティの値を使用できます。コンテキストは、コード内のどこからでも使用できるグローバル オブジェクトで、この場合、いつ障害が受信され、それが時間帯の期間内か期間外かを示すのに使用されます。イベントのタイムスタンプでそのフラグが **TRUE** を返す場合、イベントは時間帯の期間内に発生していますが、**FALSE** を返す場合、イベントは時間帯の外で発生したことを示します。必要なロジックによれば、時間帯の外で発生する障害は無視され、カウントされません。
- **変数の初期化**
 - **カウンタ変数** - 周期的な値を保持し、したがって **OnPeriodstart** ハンドラ内で初期化され、計算期間ごとに障害を別々に数えることを保証します。各期間はゼロで開始され、この期間内の障害数のみを出力します。

期間ごとの累積障害を計算する必要がある場合（つまり、各期間の結果はこの期間の最後までに発生した障害すべてで、それまでの全期間を含んでいる場合）、カウンタを *OnLoad* ハンドラ内でのみ初期化し、*OnPeriodStart* ハンドラから削除する必要があります。したがって、カウンタは数え続け、以下に示すように期間の間に累積を続けます。

```
Sub OnLoad(time)
    FingerprInted=0
End Sub
```

- **システム状態変数** - 計算の最初に一度初期化する必要があります、その点以降、ステータス イベントごとに更新される必要があります。この変数は計算の全体にわたってその値を保持し、一定の計算期間に制限されません。また、それは、計算期間の間のその値を保持する必要があります。最初のイベントを受信する前のシステム状態が不明なので、システムの状態についてはある仮定をする必要があります。一般的な仮定は、システムは "稼動中" であるということです。この仮定は合意される必要があります、以下につながる可能性があるのでチェックされる必要があります。

計算が開始し、システムの状態が実際は '休止' で、最初のイベントがこの '休止' 状態を示すために到着した場合、想定された状態は "稼動" なので、これは障害に数えられます。この障害は、その期間中に必ずしも発生しなかったとしても、最初の計算に対して数えられます。

```
Option Explicit
```

```
'Constants definitions
```

```
Const UP=1
```

```
Const DOWN=0
```

```
'Global variable for counting failures
```

```
Dim FingerprInted
```

```
Dim SystemStatus
```

```
Sub OnRegistration(dispatcher)
```

```
    ' The parameters of the method are: <procedure name>, <Event Type>
```

```
    dispatcher.RegisterByContractPartyAndService
```

```
    "OnAvailabilityEvent", "AvailabilityEvent"
```

```
End Sub
```

```
Sub OnLoad(time)
```

```
    SystemStatus = UP 'assume the first system status
```

```
End Sub
```

```
Sub OnPeriodStart(time)
```

```
    FingerprInted = 0
```

```
End Sub
```

```
Sub OnAvailabilityEvent(eventDetails)
```

```
    ' In case it's a failure and within the timeslot then it is counted
```

```
    If context.IsWithinTimeSlot and SystemStatus=UP and _
```

```
        eventDetails("Status")=DOWN Then
        FingerprInted = FingerprInted + 1
    End If
    ' update the system status for next comparison
    SystemStatus = eventDetails("Status")
End Sub

Function Result
    Result = FingerprInted
End Function
```

ケース スタディ 13: 動的コンポーネント グループの処理

グループのメンバが動的で計算期間中に変化する場合のあるリソースのグループの値を格納する必要がある場合がよくあります。以下の要件の例の計算では、リソースごとに中間計算を行い、最終集計結果を出す必要があります。

以下にそのような例をいくつか挙げます。

- **サイトでのインシデント** - 解決に X 時間を超える時間がかかるインシデントのあるサイトの割合を計算するか、または解決時間の平均が X を超えるインシデントのあるサイトを数えます。

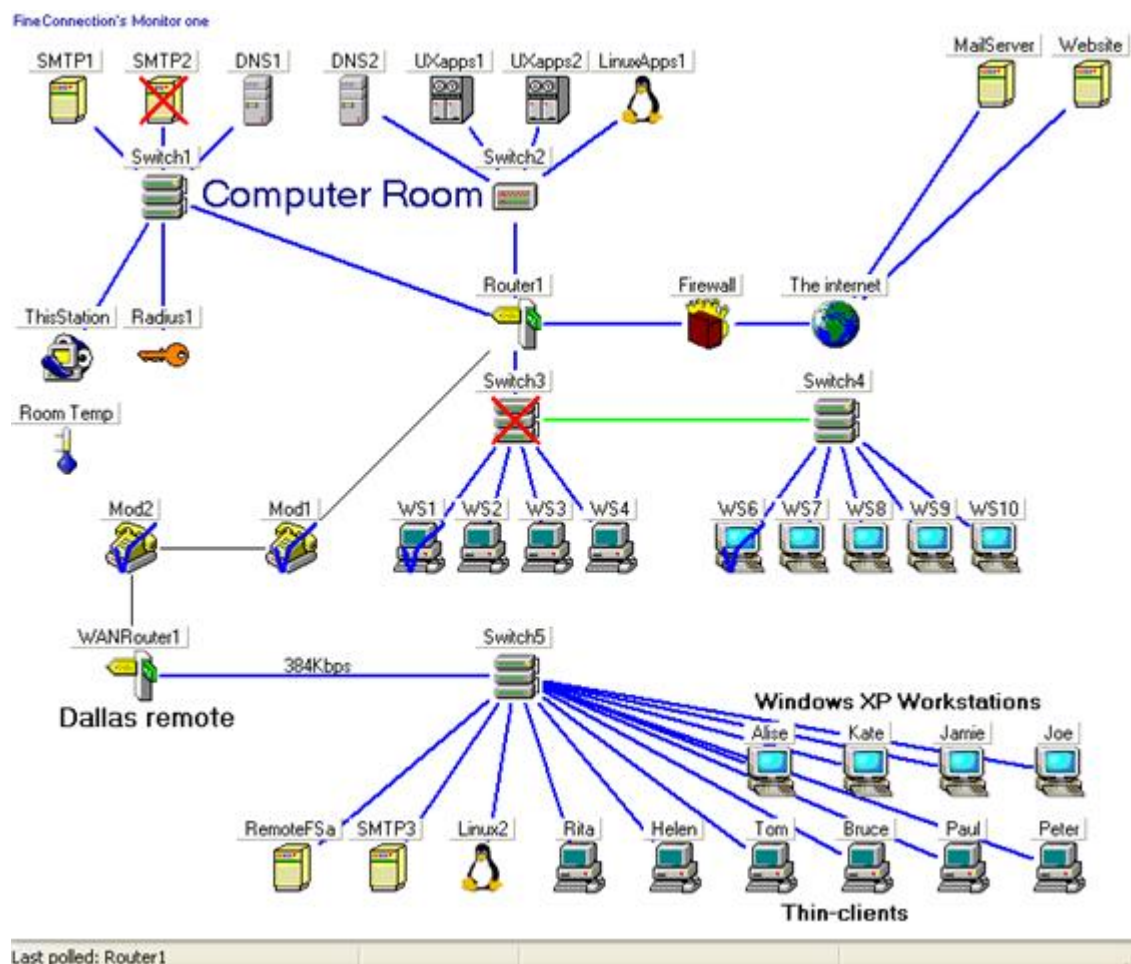
これらの例では、サイトは関連するインシデントのあるリソースです。

- **サーバの可用性** - 可用性の割合が $X\%$ を超えるサーバ数を数えます。

サーバは可用性の割合が評価される必要のあるリソースです。

- **トランザクションタイプ** - X 回を超えて失敗したトランザクションタイプの割合を計算します。

この場合、トランザクションタイプは、関連する障害イベントのあるリソースです。各トランザクションタイプについては、障害カウンタは中間結果として格納され、 X を超える障害のあるトランザクションタイプの数が数えられます。



例:

計算要件として、以下の内容があります。

サーバのクラスタで構成されるシステムの可用性の割合を計算します。システムが利用可能だと考えられるのは、基本的なサーバがすべて利用可能なときのみです。

ソリューション設計は以下のように実装されます。

システム可用性はその根本的なクラスタ リソースの可用性で評価されます。この場合、ポイントごとのシステム ステータスを遅れずに評価するために、クラスタ化されたリソースすべてのステータスを管理、格納する必要があります。

そうするためには、モニタされたリソースごとのエントリのあるマップ（下のサンプル コード中の **(ServerAvailabilityIndicators** マップ）を保持するための式が必要です。マップ オブジェクトはすべて、関連する値を参照するためにキー フィールドが必要なので、このマップはキー（リソース ID）としてリソース インジケータを持ちます。コンポーネントのステータスが変更される時はいつでも、このマップの関連するエントリが変更される必要があります。各可用性イベントについては、式は、マップ内の関連リソースのステータスを更新し、それに応じてシステム ステータスを再評価します。

モニタされるリソースが変わる場合がある（計算期間中に削除される場合もあれば、追加される場合もあるでしょう）ため、これはその式内部で管理される必要があります。ここで、その管理は、変化を識別する関数を追加し、新規コンポーネントに新しいエントリを追加したり、コンポーネントが削除されている場合は、新規コンポーネントに新規エントリを追加したり、エントリを削除したりすることにより、モニタされているコンポーネント マップを更新する関数を追加することにより管理されます。

OnRegistration は **Registration** イベントを管理するイベント ハンドラです。ここで、**Registration** イベントは、モニタされているリソース中に変化が生じたときに発生します。そのような変化は、リソースをコミット（または変更セット）した結果として生じる場合があります。これは、計算式の登録方法によっては計算に含まれるリソースに変更を生じる場合があります。

各登録中に、必要な変更についてのリソース状態を保持するマップを更新する必要があります。これは、リソース状態を保持するために使用されるマップを、登録が実行されるときにリソースを保持しているマップと（登録方法を基準として）比較することを意味します。そして、追加されたリソースをすべて含むか、または移動されたリソースを削除します。

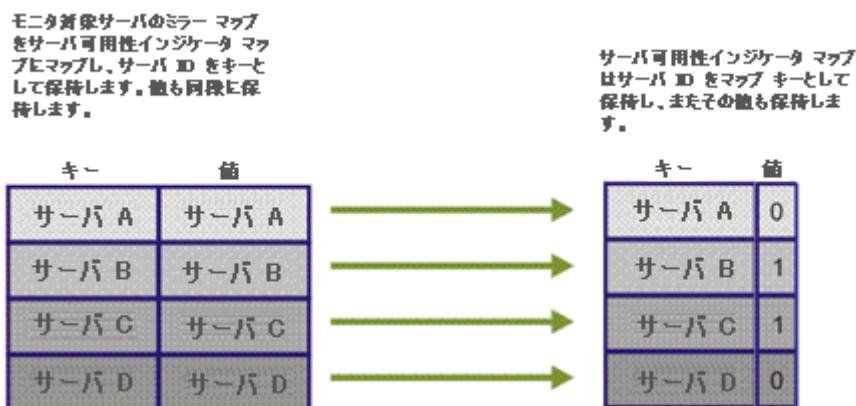
そのため、**OnRegistration** プロシージャは、モニタされたリソースをそれに応じて構造化するために、モニタされたリソースを新規に割り当てられたリソースと比較する関数を実行する必要があります。

コンテキストオブジェクトには、登録方法に匹敵する一連の方法があります。これらは、登録方法に応じたリソースの一部であるリソースすべてを備えたマップを返します。

次の例では、式の登録は `ByContractParty` で、したがって同じメソッドが `Context.ResourcesOfContractParty` によって使用されます。これは、登録時にこのセットの一部であるリソースすべてを備えたマップを返します。

2つのマップを比較するには、マップを並行に反復する必要があります。マップの反復は `"For Each"` ステートメントを使用して行われます。このステートメントにより、マップの値に沿って反復することが可能になります。したがって、ステータス値ではなくリソースに沿って反復するために、ステータスマップのミラーとして他のマップが必要です。このステートメントは、マップのキーではなく値を繰り返します。そのため、キーおよび値の両方として `ID` を保持する追加のマップが必要です。さらに、ミラーマップを維持して継続的にステータスマップをミラーし、常に同じリソースセットを保持するようにします。これは、ステータスマップが更新されたときは常にミラーマップも更新される必要があるということです。

以下の図は、マップのミラーリングの概念を示しています。



例:

```

Dim ServerAvailabilityIndicators
Dim MonitoredServers
Set ServerAvailabilityIndicators=CreateObject("SlalomMap.Map")
Set MonitoredServers=CreateObject("SlalomMap.Map")

Sub OnRegistration(dispatcher)
    dispatcher.RegisterByContractParty "OnAvailabilityEvent",_
        "Availability Event", "SAP Production Server"
    Dim AllocatedServers
    Set AllocatedServers = Context.ResourcesOfContractParty("SAP Production Server")
    UpdateMonitoredServers AllocatedServers
End Sub

Sub UpdateMonitoredServers(allocatedServers)
    Dim Server
    For Each Server In allocatedServers
        If Not MonitoredServers.Exist(Server) Then
            MonitoredServers(Server) = Server
            ServerAvailabilityIndicators(Server)=True
        End If
        次へ
    For Each Server In MonitoredServers
        If Not allocatedServers.Exist(Server) Then
            MonitoredServers.Erase Server
            ServerAvailabilityIndicators.Erase Server
        End If
    Next
End Sub

```

例:

以下の関数は、ミラー マップを使用してステータス マップを反復する方法を示しています。これが必要になるのは、モニタされているリソースごとのステータスを基準とするシステム全体のステータスを評価する必要がある場合です。

この例では、リソースすべてが利用可能な場合にシステムは利用可能だと見なされます。1つのコンポーネントが停止すれば、システムが停止したと見なすのに十分です。

```

Function SystemAvailability
    Dim Server
    For Each Server In MonitoredServers
        If ServerAvailabilityIndicators(Server) = DOWN then
            SystemAvailability=DOWN
            Exit Function
        End if
    Next
End Function

```

動的なリソース処理を備えたフル ビジネス ロジックの例を以下のデザイン パターン例で説明します。

ケース スタディ 14: 累積時間クロックの処理

このセクションで説明されているデザイン パターンは、以下の例のように、必要な結果がイベント間で消滅した期間の関数である場合は常に適切です。

- 利用可能な時間 - ある障害と次の障害間で経過した時間として計算されます。
- 解決時間 - インシデントがオープンされた時刻とクローズされた時刻の間の時間として計算されます。

累積時間については、時間が累積できる変数（秒）を割り当て、最後に更新されてからの累積時間と条件の両方をチェックする関数を実装することが必要です。そして、この関数は、式が受信したイベントすべてに対して実行されます。

以下の図は、クロック処理時間の累積を示しています。



LastUpdateTime 変数には、時間カウンタが更新されたかどうかに関わらず、最後に更新が実行された時刻が格納されます。その関数には、時間が更新されて累積される必要があるかどうかを決定する条件が保持されています。たとえば、時間帯の時間を過ぎる場合、システム状態が停止の場合、またはインシデントが保留状態の場合、時間を考慮してはいけません。

ここで詳しく説明されている状況では **Tools.NetTime()** 関数を使用して遅延を計算していることがよくありますが、標準の VB 関数である **DateDiff()** を使用の方が好ましい場合もある可能性があります。

Tools.NetTime 関数は、使用するたびに時間帯を確認するオーバーヘッドが生じます。これらのプロシージャは新規の着信イベントに対して呼び出され、したがって **NetTime** コールを呼び出すため、データ イベント プロシージャでは **NetTime** の使用を避けることが推奨されます。時間帯が 1 日 24 時間週 7 日ある場合、時間帯を確認するオーバーヘッドを避けるために **DateDiff** 関数を使用することを推奨します。

例 1:

以下の 'カウンタを更新する' ルーチンは、**PeriodNetTime** 変数内の合計期間を累積します。このルーチンは **AvailabilityTime** にシステムが「稼動」、つまりシステムが利用可能な状態である時間を累積します。

ツール オブジェクトには **NetTime** メソッドの **NetTime(beginTime, endTime)** が含まれます。このメソッドは、現在のメトリックの時間帯内にある **beginTime** と **endTime** 間の秒数を返します。これら 2 つの変数間の時間は、実行される時間帯の一部であり、したがって時間帯が使用される遅延計算用として非常に一般的に使用されます（たとえば、4 業務時間内に解決される優先度 1 チケットに対して、チケットは業務時間の最後に発生し、翌日朝まで解決されない可能性があります。が、除外されている時間帯の時間のせいでも SLA 内です）。

```
Sub UpdateCounters (time)
    PeriodNetTime = PeriodNetTime + tools.NetTime (LastUpdateTime, time)
    If SystemStatus = UP Then
        AvailabilityTime = AvailabilityTime + tools.NetTime (LastUpdateTime, time)
    End If
    LastUpdateTime = time
End Sub
```

例 2:

以下の例では、いくつかのきわめて重要なコンポーネント中の稼動および停止イベントおよびこれらのコンポーネントの保守イベントの処理により、アプリケーションの可用性を計算します。すべてのコンポーネントが保守中の場合、時間は予想される可用性時間とは見なされません。

UpdateCounters サブルーチンは必要に応じてタイム カウンタを早め、また式 (Raw データ イベント/エンジン イベント) に受診されたすべてのイベントで実行されます。またそれは、時間が時間帯期間内にあり、コンポーネントが予定のダウンタイム期間にない場合の予想使用可能時間を更新します。式が実際の可用性時間を更新するのは、システム状態が「稼動」のときのみです。

DateDiff は、2 つの日付間の時間を返す標準的な VB 関数ですが、時間帯情報を除外しません。

```
'Force variable declaration
Option Explicit

'Global variables
Dim ExpectedAvailabilityTime
Dim ActualAvailabilityTime
Dim LastUpdateTime
Dim AvailabilityIndicators
Dim MonitoredComponents
Dim DowntimeStatuses

'Map objects creation
Set AvailabilityIndicators=CreateObject("SlalomMap.Map")
Set MonitoredComponents=CreateObject("SlalomMap.Map")
Set DowntimeStatuses=CreateObject("SlalomMap.Map")

'After loading and whenever an infrastructure change occurs
Sub OnRegistration(dispatcher)
    dispatcher.RegisterByResourceGroup "OnComponentDownEvent", "Component
Down", "Application Components"
    dispatcher.RegisterByResourceGroup "OnComponentUpEvent", "Component
Up", "Application Components"
    dispatcher.RegisterByResourceGroup "OnMaintenanceStartEvent", "Maintenance
Start", "Application Components"
    dispatcher.RegisterByResourceGroup "OnMaintenanceEndEvent", "Maintenance
End", "Application Components"
    UpdateCounters Context.RegistrationTime

    Dim AllocatedComponents
    Set AllocatedComponents = Context.ResourcesOfResourceGroup("Application
Components")

    'make sure formula is monitoring only relevant and all the relevant resources
    UpdateMonitoredComponents AllocatedComponents
End Sub

Sub OnLoad(time)
    'When system goes up for the first time - assume availability OK
    LastUpdateTime = time
End Sub
```

```
Sub OnPeriodStart(time)
    'initializing counters to renew periodic calculation
    ExpectedAvailabilityTime = 0
    ActualAvailabilityTime = 0
End Sub

Sub OnTimeslotEnter(time)
    UpdateCounters time
End Sub

Sub OnTimeslotExit(time)
    UpdateCounters time
End Sub

Sub OnComponentDownEvent(eventDetails)
    UpdateCounters eventDetails.Time
    'write availability status of reported-on resource
    AvailabilityIndicators(eventDetails.ResourceId) = _
        AvailabilityIndicators(eventDetails.ResourceId)+1
End Sub

Sub OnComponentUpEvent(eventDetails)
    UpdateCounters eventDetails.Time
    'write availability status of reported-on resource
    AvailabilityIndicators(eventDetails.ResourceId)= _
        AvailabilityIndicators(eventDetails.ResourceId)-1
End Sub

Sub OnMaintenanceStartEvent(eventDetails)
    UpdateCounters eventDetails.Time
    'write availability status of reported-on resource
    DowntimeStatuses(eventDetails.ResourceId)= _
        DowntimeStatuses(eventDetails.ResourceId)+1
End Sub

Sub OnMaintenanceEndEvent(eventDetails)
    UpdateCounters eventDetails.Time
    'write availability status of reported-on resource
    DowntimeStatuses(eventDetails.ResourceId)= _
        DowntimeStatuses(eventDetails.ResourceId)-1
End Sub

Sub OnPeriodEnd(time,isComplete)
    UpdateCounters 時間
End Sub

Function Result
    If ExpectedAvailabilityTime <> 0 Then
        Result = 100 * (ActualAvailabilityTime / ExpectedAvailabilityTime)
```

```

Else
    Result = Null
End If
End Function

Sub UpdateCounters(time)
    If Context.IsWithinTimeslot And Not AllComponentsAreInPlannedDowntime Then
        'update counter of seconds in period (when availability is expected)
        ExpectedAvailabilityTime = ExpectedAvailabilityTime +
DateDiff("s",LastUpdateTime,time)
        If SystemIsAvailable Then
            'update seconds-of-availability counter
            ActualAvailabilityTime = ActualAvailabilityTime +
DateDiff("s",LastUpdateTime,time)
        End If
    End If
    LastUpdateTime=time
End Sub

Sub UpdateMonitoredComponents(allocatedComponents)
    Dim Component
    'add to monitored Components map all new Components to be monitored
    For Each Component In allocatedComponents
        If Not MonitoredComponents.Exist(Component) Then
            MonitoredComponents(Component) = Component
            AvailabilityIndicators(Component) = 0
            DowntimeStatuses(Component) = 0
        End If
    Next

    'remove from monitored Components map all no-longer-relevant Components
    For Each Component In MonitoredComponents
        If Not allocatedComponents.Exist(Component) Then
            MonitoredComponents.Erase Component
            AvailabilityIndicators.Erase Component
            DowntimeStatuses.Erase Component
        End If
    Next
End Sub

Function SystemIsAvailable
    Dim SystemAvailability
    SystemAvailability = True

    Dim Component
    Dim ComponentAvailability
    For Each Component In MonitoredComponents
        ComponentAvailability = AvailabilityIndicators(Component) = 0 _
            Or DowntimeStatuses(Component) > 0
    Next
End Function

```

```
'system availability is evaluated with availability
SystemAvailability = SystemAvailability And ComponentAvailability
Next
SystemIsAvailable = SystemAvailability
End Function

Function AllComponentsAreInPlannedDowntime
Dim ComponentsInPlannedDowntime
ComponentsInPlannedDowntime = 0
Dim Component
For Each Component In MonitoredComponents
    If DowntimeStatuses(Component) > 0 Then
        ComponentsInPlannedDowntime = ComponentsInPlannedDowntime + 1
    End If
Next

If ComponentsInPlannedDowntime = MonitoredComponents.Count Then
    AllComponentsAreInPlannedDowntime = True
Else
    AllComponentsAreInPlannedDowntime = False
End If
End Function
```

効果的なビジネス ロジック例の作成

効果的なビジネス ロジックの書き方についてのベスト プラクティスの推奨が以下の項目として挙げられています。

■ クラスタ化メトリック

- システムのボリュームを評価するとき、クラスタ化メトリックをメトリック内の項目の数として数え、すべてがかけ算されることを覚えておきます。
- 1つのクラスタ アイテムの再計算は、クラスタ全体の再計算を意味します。したがって、クラスタ化を検討するとき、アダプタが構成される方法およびリソース構造を変更するときにはこの点を忘れずに考慮します。
- 多くのクラスタ アイテムに移動する同じ **Raw** データ イベントには、高機能コスト（コンテキスト スwitchング）があります。

■ グローバル変数

- コード中の必要な場所それぞれでパラメータと属性値を取得します。それらをグローバル変数、特にクラスタ化メトリックに保持するのは避けます（“states” サイズが増加します）
- 大きなマップを処理するロジックは避けます。代わりに、各イベントを **OnXXEvent** メソッドで処理します
- マップからできるだけ早く項目を削除します。たとえば、チケットが、期間の最後でないときに閉じられる場合

■ デザイン パターン

事前定義コンテンツ パッケージには、共通シナリオ用にいくつかのデザイン パターンが用意されています。これらのデザイン パターンを使用するとパフォーマンスが改善する場合があります。

■ ビルトイン機能

ACE には以下のようなさまざまな目的のためのビルトイン機能とツールがあります。

- 時間帯機能
 - NetTime
 - IsWithinTimeslot
- イベントの時間
 - TimeOfLastEvent
 - TimeOfLastEventHandler
- コンテキスト オブジェクト

- 多くの環境に敏感なメソッドを含んでいます。
- これらのメソッドを使用し、"安全な ODBC" を回避します。

■ ビジネス ロジック出力

T_SLALOM_OUTPUTS 内に構造を保持します。これは、類似の構造を持つ T_SLALOM_OUTPUTS に論理テーブルをいくつか持っている場合、同じ物理フィールドに類似の論理フィールドを置くことは非常に役立つことを意味します。これにより、インデックス付けが容易になりレポートのパフォーマンスが向上します。

■ イベント再利用性

使用するとき

- いくつかのメトリックは第 1 フェーズの計算を実行中です。これ自体、契約に必要ですが、結果を計算する要約メトリックにイベントを送信します（たとえば会計計算、高レベル KPI）。
- Raw データに予備集約を実行し、他のいくつかのメトリックにイベントを送信する単一のメトリック。メトリックが、得たよりも相当に少ないイベントを送信するか、または何度も実行される重要な計算を実行する場合は合理的。

避けるとき

- メトリックの数を著しく増加させる場合
- 3 レベルを超える実装
- 実装の複雑さは増し、保守はさらに困難になります。

■ 再計算

- システムの通常動作の一部としての重い再計算を回避
- 再計算の理由は次のとおりです。
 - 過去のタイム スタンプを備えた Raw データ
 - 過去の Raw データを変更するイベント固有性
 - 修正
 - 例外
 - ビジネス ロジック モジュールでの変更
 - 契約の変更
 - 過去のタイム スタンプを備えたイベント再利用性イベント
 - リソース構造体の変更
- 計算されたデータのロック機能の使用検討

■ ビジネス ロジック モジュール

- ビジネス ロジック モジュールは、一度は完全に確認され、変更の必要がないという方法で書かれる必要があります。
- ガイドラインは、1つのモジュールは1つの一般的なロジックと等しい、というものです。
- 非常に具体的なビジネス ロジック モジュールは多くのメトリックで利用できず、コードおよびロジックの再利用を促進しません。
- 一般的すぎるビジネス ロジック モジュールは保守が困難です。さらに、ビジネス ロジック モジュールが多くの複雑なロジックを実装している場合、（メトリックの一部によって使用された）1つのフローで1箇所修正すると、すべてのメトリックを再計算することになります。

■ 登録

- 登録を使用して、イベントのフィルタリングをすべて実行します。コードにフィルタリングを残すとパフォーマンスに大きく影響します
- 単純化
- 登録自体でないコードについては、特にリソース変更が多数あるときは、`dispatcher.IsRunTimeMode` と `OnResourceStructureChange` のメソッドを使用します
- `RegisterByEventType` メソッド使用の回避
- ビジネス ロジック モジュールでは、（契約関係者、サービス、リソースタイプによる）一般的な形式を使用するか、パラメータを使用するか、またはユーザ インターフェース（イベント再利用性用の優先オプション）を使用して実行される登録を残します。

ケース スタディ 15: クラスタ化メトリック

通常、あるソフトウェアを説明するとき、説明は **WHAT** および **HOW** の 2 つの部分に分割できます。**WHAT** は、この一片のコードが何を実行するかの説明を意味します。**HOW** は、どのようにそれを実行するかです。**WHAT** の部分に専念し、**HOW** の部分を見捨てる傾向があります。その理由は単純で、多くの場合は正当化されます。そうすることによって、コンポーネント間の結合を縮小し、多くの場合無関係の情報に煩わされなくなります。しかし、多くの場合、**HOW** の部分を見捨てるのは割に合いません。

このケース スタディでは、エンジンがクラスタ化メトリックを計算する方法（**HOW** 部分への回答）を説明し、特定の実装について結果として伴うパフォーマンスコストを説明します。また、実装法を変更することにより、このコストを削減する方法をいくつか説明しています。

クラスタ化メトリックとは

クラスタ化メトリックは、それらの定義内にリソースのあるグループを埋め込んだメトリックです。このグループはメトリックのクラスタと呼ばれ、そのグループ内のリソースはそれぞれクラスタ アイテムと呼ばれます。クラスタ化メトリックを計算するとき、それらのクラスタ アイテムに対して個別の計算が実行されます。それらのクラスタ アイテムそれぞれの計算は次のものを除いて互いに似ています。

- **Context.ClusterItem** - 計算されているクラスタ アイテムに等しい **Context.ClusterItem** プロパティの値。
- クラスタ アイテムによる登録 - メトリックがクラスタ アイテムによりイベントに登録されるとき、各クラスタ アイテムは **Raw** データとこのクラスタ アイテムに送信される再利用性イベントを受信します。

クラスタ化メトリックの計算方法

クラスタ化メトリックの計算について理解する必要のある重要なことは、クラスタ アイテムはすべて並行して計算されるということです。並行ということは、別々のスレッドで計算されるという意味ではなく、さまざまなクラスタ アイテムにより処理される必要のあるさまざまなイベントを処理する間、イベントはシーケンシャルに処理され、それぞれのイベントに対して関連するクラスタ アイテムが呼び出されてこのイベントを処理するということです。たとえば、多くのクラスタ アイテムによって処理される必要のある多くのイベントがあります。これを行う方法は 2 とおりあります。

例: オプション 1

各クラスタ アイテム **C** に対し

C によって処理される必要のある各イベント **E** に対し

C に **E** を処理させる

例: オプション 2

各イベント E に対し

- E を処理する必要がある各クラスタ アイテム C に対し
- C に E を処理させる

エンジンはオプション 2 を使用して、クラスタ化メトリックを処理します。

理解する必要がある別の重要なポイントは、PslWriter の内部の VBScript は Script Control と呼ばれるコンポーネントによって実行されるということです。各メトリックについてこのコンポーネントのインスタンスはただ 1 つのみで、このインスタンスはさまざまなクラスタ アイテムの計算に再利用されます。前に述べたようにクラスタ アイテムは並行に計算され、Script Control コンポーネントは各瞬間に単一のクラスタ アイテムのみを含むことができるので、Script Control コンポーネントの内部のデータを頻繁にスイッチする必要があります。

これについて説明するために、その計算を実行するさらに詳しい疑似コードを下に示します。

- 1- 各メトリック M に対し
- 2- X を M によってまだハンドルされていない最初のイベントとする
- 3- T を X の前の最新の状態のタイム スタンプとする
- 4- L をタイム スタンプ T から現在の時刻までの、M に登録されたすべてのイベントのリスト（すべてのクラスタ アイテム）とする
- 5- L 内の各イベント E に対し
- 6- E を処理する必要がある各クラスタ アイテム C に対し
- 7- C' を、現在スクリプト コントロールにロードされているクラスタ アイテムとする
- 8- スクリプト コントロールからグローバル変数の値をとって C' 用に別に格納する
- 9- C 用の別に格納されたグローバル変数の値を取り、スクリプト コントロールにロードする
- 10- イベント E の処理

まだ考慮に入れられていなくて、次にこのポイントから先の計算を実行するイベントの時間を見つけるこの全工程は、再計算と呼ばれます。グローバル変数（上記のコードの手順 8 および 9）の値を置換する過程は、コンテキスト スイッチングと呼ばれます。

上記のコード中で容易にわかる、2 つの主要な問題は次のとおりです。

- 再計算は、すべてのクラスタ アイテムに対してまとめて行われます。時間 T の点（上記コードの手順 3）は一度発見され、次にクラスタ アイテムはすべてその点を基準に再計算を実行します。これは、単一のクラスタ アイテムに新規イベントがある場合は常にクラスタ アイテムすべてが再計算されることを意味します。
- コンテキスト スイッチングは頻繁に行われます。コンテキスト スイッチング（上記コードの手順 8 および 9）は内部ループの内側にあるので、簡単にわかります。

クラスタ化メトリックの再計算

- 問題の解説

すでに説明されていたように、クラスタ化メトリック内のクラスタ アイテムはすべてまとめて再計算されます。これは、**1000** クラスタ アイテムを超えてクラスタ化されるメトリックがあり、届いたある新イベントのためにそのうちの**1**つを再計算する必要がある場合、その**1000**のクラスタ アイテムすべてが昨年度に対して再計算されます。

■ 解決法

以下のソリューション提案は、この問題の負担を軽減できます。しかし、それらは必ずしも常に適用可能ではなく、各々不利な点もあります。重要なのは、問題とその見積もりコストを理解し、このコストを提案されたソリューションのコストと比較することです。

- クラスタ アイテムの数が少ないとき、それらの各々を個別のメトリックとして定義するオプションを検討できます。この方法の欠点はもちろん、複数のメトリックを維持するコストと、メトリック全体のレポートおよび次に特定のクラスタ アイテムの掘り下げができないことです。
- クラスタ アイテムの数は多いが、頻繁に再計算されるのはそれらのうちのほんの**1**つ（または少数）のとき、このクラスタ アイテムを別のメトリックに入れ、残りのクラスタ アイテムすべてを他のメトリックに入れることを検討できます。
- このメトリックが長時間の再計算を行わないように、関連する契約/メトリックに対する計算凍結を頻繁に使用します。
- 長い再計算がないように（つまり、タイムスタンプが**1**か月より古いイベントを送信しないように）、アダプタおよびそのデータ ソースである変更を実行します。

コンテキスト スイッチング

■ 問題の解説

すでに説明したように、コンテキスト スイッチングは最も内側のループ中で実行されます。言い換えれば、クラスタ アイテムごとに処理される必要のある各イベントに対して実行されます。メトリックが多くのイベントを受信し、各イベントが多くのクラスタ アイテムによって処理されるとき、この量は非常に高くなる可能性があります。コンテキスト スイッチング操作が（ビジネス ロジック内のイベント自体の取り扱いとの比較で）比較的高価で、問題であることを付け加えておきます。

コンテキスト スイッチング操作のコストは、スイッチングしたデータのサイズに比例します。私たちがコンテキスト スイッチ中に切り替えるデータは、ビジネス ロジック ("状態" と呼ぶ) 内のすべてのグローバル変数の値です。したがって、グローバル変数が多くなり、それらのグローバル変数のサイズが大きくなるにつれて、コンテキスト スイッチング操作はそれだけ高くつくようになります。

特に、ビジネス ロジック マップをクラスタ化メトリック中で使用することは、特にそれらのマップのサイズが大きくなる可能性がある場合、お勧めできません。

■ 解決法

■ コンテキスト スイッチの各々の時間を縮小

目的は状態（グローバル変数）のサイズを小さくすることです。この方法は、マップを含まないようにビジネス ロジックを書き直すことにより実現できます。もちろん、常に可能だとは限りませんが、可能な場合は推奨します。

■ コンテキスト スイッチの量を減らす

クラスタが小さいとき、クラスタ アイテムごとに個別のメトリックを作成することが可能です。

同一イベントに登録する多数のクラスタ化された項目を備えるクラスタ化メトリックは避けます。この目的を以下に示します。

イベントがそれぞれ単一のクラスタ アイテムによって処理される場合、コンテキスト スイッチングの量はイベントの数に比例します。

イベントがそれぞれすべてのクラスタ アイテムによって処理される場合、コンテキスト スイッチングの量は、イベント数にクラスタ アイテム数をかけたものに比例します。

元のクラスタ アイテム（それらは現在クラスタ アイテムではなく単純なリソースである）のすべてに対する結果を計算する非クラスタ化メトリックを作成します。このメトリックにイベントとしてクラスタ アイテムの各々の結果を送らせます。クラスタ化され、最初のメトリックからイベントを受信する別のメトリックを作成し、それらのイベントで受信した値を結果としてレポートします。ここでのアイデアは、大きな量の **Raw** データ イベントが非クラスタ化メトリックによって処理されるということです。また、クラスタ化メトリックは、クラスタ アイテムについて期間当たりの単一のイベントを処理します。

ケース スタディ 16: ビジネス ロジックの設計パターン

多くのビジネス ロジックで使用できるいくつかの "設計パターン" があります。それらの設計パターンはテストされ、適用可能なときにそれらを使用することで多くの時間を節約でき、ほとんどの場合、より効率的なビジネス ロジックを作成できます。このケース スタディはそのようなデザイン パターンの 1 つに焦点を当てます。

更新カウンタの設計パターン

この設計パターンはほとんどすべてのビジネス ロジックで役に立つもので、あるイベント間の時間を測定するように意図されます。そのようなビジネス ロジックの例は、可用性、ダウンタイム、平均故障間隔、平均回復時間、平均応答時間、平均解決時間、可用性が X 未満のコンポーネントの割合、時間どおりに解決されなかったケースの数などのビジネス ロジックです。

それらのビジネス ロジックすべてで共通の部分は、結果がそれらのビジネス ロジックが受信するさまざまなイベントのタイムスタンプに依存するということです。

ユーザのビジネス ロジックがこのデザイン パターンから利益を得ることができるかどうかを決めるための経験則は次のとおりです。ビジネス ロジックが、それが受信するさまざまなイベントのタイムスタンプに依存する場合、この設計パターンを使用する必要がある可能性が高くなります。

この設計パターンの骨格

このパターンを利用するビジネス ロジックのコードはフレームワークと実装の 2 つの部分に分割できます。フレームワークには、ほとんどの場合固定され、さまざまなビジネス ロジックに対して変わらないコードが含まれます。この部分は、可用性、および時間どおりに解決されないチケットの数の計算に対して同じです。その実装には、各ビジネス ロジックに固有のコードが含まれます。

コードのそれら 2 つの部分を別々のビジネス ロジック モジュールに入れ、フレームワークのモジュールを別々のメトリックで再利用することを推奨します。

フレームワークのコードを以下に示します。

```
Dim g_PrevEventTimestamp

Sub OnLoad(time)
    g_PrevEventTimestamp = time
    InitializeStatusVariables
End Sub

Sub OnRegistration(Dispatcher)
    ' If there is a separate copy of status variables
    ' for each registered resource depend on the registered
```

```
        ' resources you should set initial values to the
        ' status variables of the newly added resources here
End Sub

Sub OnPeriodStart(time)
    InitializeCounters
End Sub

Sub OnPeriodEnd(time, completePeriod)
    HandleEvent (time)
End Sub

Sub OnTimeslotEnter(time)
    HandleEvent (time)
End Sub

Sub OnTimeslotExit(time)
    HandleEvent (time)
End Sub

Function Result()
    Result = CalculateResult()
End Function

Sub HandleEvent(Time)
    Dim diff
    diff = TimeDiff( "s",Time,g_PrevEventTimestamp)
    UpdateCounters(diff)
    g_PrevEventTimestamp = Time
End Sub
```

実装モジュールのスケルトンを以下に示します。

```
' Define your status variables here. This can be one
' simple global variable or many complex global variables
' depending on the business logic
Dim g_StatusVar_1, g_StatusVar_2, ... ,g_StatusVar_n

' Define your counters here.
' This can be one simple global variable or many
' complex global variables depending on the business logic
Dim g_Counter_1, g_Counter_2, ... , g_Counter_n

Sub InitializeStatusVariables ()
    ' Set initial values to the various status variables
End Sub
```

```
Sub InitializeCounters ()
    ' Set initial values to the various counters
    g_Counter_1 = 0
    g_Counter_2 = 0
    ' ...
    g_Counter_n = 0
End Sub

Function CalculateResult ()
    ' Calculate the result. The result should depend on
    ' the values of the counters. It should not depend on
    ' the value of the status variables. It should not
    ' change the values of the counters or the status
    ' variables
End Function

Sub UpdateStatus(method, eventDetails)
    ' Update the value of the status variables based on
    ' the parameters (and possibly on the old value of the
    ' status variables)
End Sub

Sub UpdateCounters(diff)
    ' Update the values of the counters based on their
    ' previous value, on the value of the status variables
    ' and on the value of the diff parameter.
    ' In many cases this calculation is also based on the
    ' value of Context.IsWithinTimeslot
End Sub

Sub OnEvent_1(eventDetails)
    HandleEvent (eventDetails.Time)
    UpdateStatus("OnEvent_1",eventDetails)
End Sub

Sub OnEvent_2(eventDetails)
    HandleEvent (eventDetails.Time)
    UpdateStatus("OnEvent_2",eventDetails)
End Sub

' ...

Sub OnEvent_n(eventDetails)
    HandleEvent (eventDetails.Time)
    UpdateStatus("OnEvent_n",eventDetails)
End Sub
```

この設計パターンについてさらに説明するために、可用性の計算にこのパターンを実装した例を以下に示します。この例では、ビジネス ロジック内の個別の 2 つのイベントハンドラによって処理される UP および DOWN のイベントがあると仮定します。可用性は、タイムスロット内の総時間に対する、システムが稼動されていた時間のパーセントとして定義されます。システムのステータスは、最後に受信したイベント（UP または DOWN）のステータスであると仮定します。

実装するコードを以下に示します（フレームワークのコードは変更しません）。

```
' Status variable
Dim g_Status
' Counters.
Dim g_UpTime, g_TotalTime

Sub InitializeStatusVariables ()
    G_Status = "UP"
End Sub

Sub InitializeCounters ()
    g_UpTime = 0
    g_TotalTime = 0
End Sub

Function CalculateResult ()
    If g_TotalTime = 0 Then
        CalculateResult = Null
    Else
        CalculateResult = g_UpTime/g_TotalTime*100
    End If
End Function

Sub UpdateStatus(method, eventDetails)
    If method = "OnUP" Then
        G_Status = "UP"
    Else
        G_Status = "DOWN"
    End If
End Sub

Sub UpdateCounters(diff)
    If Context.IsWithinTimeslot Then
        G_TotalTime = g_TotalTime + diff
        If g_Status = "UP" Then
            G_UpTime = g_UpTime + diff
        End If
    End If
End Sub

Sub OnUp(eventDetails)
    HandleEvent (eventDetails.Time)
```



```
        UpdateStatus("OnUp",eventDetails)
    End Sub

    Sub OnDown(eventDetails)
        HandleEvent (eventDetails.Time)
        UpdateStatus("OnDown",eventDetails)
    End Sub
```

このパターンにはいくつかのバリエーションがあります。最も一般的なバリエーションの1つは、各種のエンティティに対して個別の時間カウンタを保持する必要がある場合です。たとえば、解決時間を測定する場合、各オープンチケットに対して個別のカウンタを保持する必要があります。この場合、1つのチケットにのみ関連するイベントを処理する場合は、そのチケットのカウンタのみを更新するほうが効率的です。共通のイベントが処理されるとき（OnPeriodEnd、OnTimeslotEnter など）は、すべてのチケットのカウンタが更新される必要があります。

注: このパターンのこのバリエーションでは、各チケットに対して `g_PrevEventTimestamp` グローバル変数の個別のコピーを保持する必要があります。

このパターンの良い使用例のいくつかは、事前定義済みのコンテンツで見られます。このパターンは事前定義済みコンテンツでは少し違った形で使用され、フレームワークと実装の間の区別はそれほど明白ではありません。

ケース スタディ 17: 組み込み機能

ACE には、さまざまな目的の組み込み機能およびツールがあります。この組み込み機能を使用するのは、VBS 内にその機能を記述するよりも望ましい方法です。VBS はインタープリタ型言語であるため、VBS 内で再現するとパフォーマンスに悪影響します。

組み込み関数のリストおよびそれらを使用する適切な方法を以下に示します。

IsWithinTimeslot

これは最も簡単な組み込み関数です。その用途は、システムが現在タイムスロット内にあるかどうかをビジネス ロジックが認識できるようにすることです。これにより、同じことを行うために、OnTimeslotEnter および OnTimeslotExit 関数の変数を管理する必要がなくなります。たとえば、以下のコードを実行する代わりに、

```
Dim amIWithinATimeslot
Sub OnTimeslotEnter(time)
    amIWithinATimeslot = 1
End sub
Sub OnTimeslotExit(time)
    amIWithinATimeslot = 0
End sub
Sub OnEvent(eventDetails)
    If amIWithinATimeslot = 1 Then
        count = count + 1
    End if
End sub
```

以下のより簡単なコードを実行できます。

```
Sub OnEvent(eventDetails)
    If context.IsWithinTimeslot Then
        count = count + 1
    End if
End sub
```

OnTimeslotEnter および OnTimeslotExit のタイムスタンプに関する情報を使用または維持する場合、この機能ではそのニーズを満たすことはできません。ただし、通常それは必要ではなく、このコードで十分です。

TimeOfLastEvent

この関数は、最後に処理された Raw データまたは中間データ イベントのタイムスタンプを返します。これは、イベント ハンドラにこの情報を保存する必要がなく、この関数を使用して直接利用できることを意味します。以下に例を挙げます。

```
Function result
    Dim LastEventTimestamp
    LastEventTimestamp = Context.TimeOfLastEvent
End function
```

TimeOfLastEventHandler

この関数は、ACE によって最後に呼び出されたイベント ハンドラのタイムスタンプを返します。これには、**Raw** データおよび中間データのイベント ハンドラだけでなく、呼び出されたすべてのシステム イベントも含まれます。これは、時間を受け取らないイベント ハンドラ（たとえば、**result** 関数）で特に役立ちます。以下に例を挙げます。

```
Function result
    Dim LastEventHandlerTimestamp
    LastEventHandlerTimestamp= Context.TimeOfLastEventHandler
End function
```

NetTime

この関数は、2 つのタイムスタンプの指定し、それらの 2 つのタイムスタンプの間で、システムが現在のルールタイムスロット内にあった実時間（秒単位）を受け取ることができます。これは特に煩雑な機能なので、VBS 内に実装しないでください。VBS 内にこれを実装すると、すべての **OnTimeslotEnter** および **OnTimeslotExit** のリストを維持するか、それらの間のタイムスパンを算出するために、タイムスロットに入ってから出るまでの時間を毎回直接計算する必要があります。極端な状況では、これには長い時間がかかる場合があり、計算のパフォーマンスによくありません。内部関数は、大幅に最適化された後に同じことを行うため、より効率的に実行されます。以下に例を挙げます。

```
Function result
    Dim MyNetTime
    MyNetTime = Tools.NetTime(MyBeginTimestamp, MyEndTimestamp)
End function
```

コンテキストオブジェクト

コンテキストオブジェクトには、以下の内容に関する情報を提供するさまざまなパラメータがあります。

- 現在のメトリック。
- それが含まれている契約。
- 現在の計算の状態。
- サービス ドメイン、ドメイン カテゴリ、およびそれらに関連する値（たとえば、しきい値）。
- クラスタ化情報 - クラスタ全般および処理される特定のクラスタ アイテム。
- ユーザの要求に基づいてリソースのリストを返す関数。
- リソース名をリソース ID に（または、その逆に）変換できる関数。

安全な ODBC を使用してデータベース内のこの情報に直接アクセスすることは非常に非能率的であり、この情報はコンテキスト オブジェクトから簡単に取得できるので意味がありません。情報の取得時は、可能であれば常に組み込み機能を使用します。

ケース スタディ 18: 登録

ビジネス ロジックは、計算中に使用されるメトリックのリソース構造体のマップを維持するように多くの場合記述されます。リソース構造体は時間の経過に従って変更されるため、そのようなビジネス ロジックは、リソース構造体に変更されたときに、マップ内の構造体を更新する必要があります。

リソース構造体に変更されると、**OnRegistration** メソッドが呼び出されます。このメソッドは、リソース構造体の変更に伴う登録およびクラスタの変更に対応するエンジンの動作の管理を担っているためです。リソース構造体に変更されるたびにこのメソッドが呼び出されることは、前述のマップの更新を行うためには良いタイミングです。ただし、マップを満たすことは登録処理とは無関係です。これは、マップを満たすために **OnRegistration** 関数のパフォーマンスが悪化することを意味します。通常、これは頻繁には発生しないため、実行中は重要ではありません。ただし、**OnRegistration** メソッドは、エンジンのインフラストラクチャ処理プロセス中にも呼び出されます。このプロセスでは、そのインスタンスが担当する特定の各メトリックの登録に、リソース構造体の変更が関連しているかどうかをシステムが判別します。このプロセスでは、構造体の変更が現在のメトリックに関連していなくても、リソース構造体のすべての変更に対して **OnRegistration** メソッドが呼び出されます。これは、1つのメトリックでこのメソッドが大量に呼び出される場合があることを意味します。

そのようなロジックが **OnRegistration** メソッドに実装されていると、実行中のパフォーマンスが少し低下することにより、インフラストラクチャ処理中のパフォーマンスが非常に大きく低下する場合があります。

この問題を解決するために、リソース構造体の変更が発生したときに行う必要があるが登録に関連しないマップの設定およびその他の初期化を、以下の 2 つの方法で行うことができます。

ディスパッチャ オブジェクトの **IsRunTimeMode** プロパティを使用する

このプロパティを使用すると、現在の実行が計算の実行であるかどうかを判別して、実行中にのみ実行される「if」ステートメントに、登録に関連しないロジックを埋め込むことができます。

以下の例で、青でマークされている部分は、登録に関連していて、常に実行する必要があるビジネス ロジックの部分です。緑でマークされている部分は、登録に関連しておらず、新しい「if」ステートメントに埋め込むことができます。

```
Sub OnRegistration (dispatcher)
    Dim MyResource
    MyResource = Context.ClusterItem
    Dispatcher.RegisterByResource "OnEvent", "My Event Type", MyResource
    Dim ThisResourceMap
    Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
    Dim resource
```

```
Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
For Each resource In ThisResourceMap
    GlobalResourceVector.Add resource
    次へ
End Sub
```

このコードは、以下のように変更することにより改善できます。

```
Sub OnRegistration (dispatcher)
    Dim MyResource
    MyResource = Context.ClusterItem
    Dispatcher.RegisterByResource "OnEvent", "My Event Type", MyResource
    If Dispatcher.IsRunTimeMode Then
        Dim ThisResourceMap
        Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
        Dim resource
        ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
        For Each resource In ThisResourceMap
            GlobalResourceVector.Add resource
        Next
    End If
End Sub
```

OnResourceStructureChanged メソッドを使用する

このメソッドは **OnRegistration** メソッドの直後に実行されるため、元の方法と同じ機能が提供されますが、実行中にのみ実行されます。このメソッドは、インフラストラクチャ処理では呼び出されないため、パフォーマンスは損なわれません。

以下の例で、青でマークされている部分は、登録に関連していて、**OnRegistration** メソッドに残す必要があるビジネス ロジックの部分です。緑でマークされている部分は、登録に関連しておらず、新しい関数に移動できます。

```
Sub OnRegistration (dispatcher)
    Dim MyResource
    MyResource = Context.ClusterItem
    Dispatcher.RegisterByResource "OnEvent", "My Event Type", MyResource
    Dim ThisResourceMap
    Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
    Dim resource
    Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
    For Each resource In ThisResourceMap
        GlobalResourceVector.Add resource
    Next
End Sub
```

このコードは、以下のように変更することにより改善できます。

```
Sub OnRegistration (dispatcher)
    Dim MyResource
    MyResource = Context.ClusterItem
    Dispatcher.RegisterByResource "OnEvent", "My Event Type", MyResource
```

```
End Sub

Sub OnResourceStructureChanged(time)
    Dim ThisResourceMap
    Set GlobalResourceVector= CreateObject("SlalomVector.Vector")
    Dim resource
    Set ThisResourceMap = Context.ResourcesOfResourceGroup(Context.ClusterItem)
    For Each resource In ThisResourceMap
        GlobalResourceVector.Add resource
    Next
End Sub
```

ケース スタディ 19: ファイル ベースのデータ ソースの場合のアダプタ ウィザード

このケース スタディでは、ファイルベースのデータソースと統合する場合にベストプラクティスとなる方法を検討します。このシナリオの例では、ソースシステムによって生成された CSV データ ファイルを処理します。CA では、ほとんどのファイルベースの統合において、統合のリスクが最小になるように、以下のいくつかの基本的なガイドラインに従うことをお勧めしています。詳細は以下のとおりです。

- オプションが利用できる場合、データを CA Business Service Insight アプリケーション サーバのファイルシステムに配信するようにリクエストしてください。これにより、アダプタがリモートストアからデータを取得しようとすることに配信メカニズムが依存しないようになります（ユーザ アカウントのアクセス権の問題、同期の問題などを最小化します）。
- アダプタは英数字の名前に従ってファイルを並べ替えるため、ファイルの命名規則は重要です。これを制御できる場合、CA は 2 つの点についてリクエストすることをお勧めします。
 - ソース ファイルの内容に基づいた妥当な命名規則（特にそのソースから複数のファイルが来る場合）
 - 最新のファイルが最後になるように並べ替えられる逆順のタイムスタンプ（つまり、<file_name>_yyyymmdd_hhmiss.csv）。使用するタイムスタンプの精度は、配信されるデータの頻度に依存します。

このシナリオでは、ソース ファイルは Topaz データ ソース（HP が所有し、現在は Mercury Global Monitor として知られています）からのものです。これは、必要な特定の KPI の必要なファイルが含まれるように、この製品の API を使用して作成されています。このファイルは、外部の自動プロセスによって CA Business Service Insight アプリケーション サーバに直接配信されます。このソース ファイルには、次のような名前が付けられます： *Topaz_yyyymmdd_hhmiss.csv*。

注: このファイルのタイムスタンプは、ファイルが作成された時間です。したがって、ファイル内のすべてのエントリはこの時間より前に発生しています。

ファイル内のデータの例を以下に示します。

	A	B	C	D	E	F	G	H	I	J	K
1	Profile	Status	TopazTransaction	Time	ResponseTime	WtdConnectionTime	WtdDNSTime	WtdSSLTime	WtdNetworkTime	WtdDownloadTime	
2	54	Pass	yh_L	30/03/2005 06:09	218	6	1	23	179	2	
3	54	Pass	yh_L	30/03/2005 06:24	270	7	1	25	226	4	
4	54	Pass	yh_L	30/03/2005 06:39	209	6	1	21	171	3	
5	54	Pass	yh_L	30/03/2005 06:54	243	5	0	22	204	3	
6	54	Pass	yh_L	30/03/2005 07:09	207	6	1	21	167	4	
7	54	Pass	yh_L	30/03/2005 07:24	245	5	1	24	203	4	
8	54	Pass	yh_L	30/03/2005 07:39	218	6	1	22	180	3	
9	54	Pass	yh_L	30/03/2005 07:54	207	7	1	24	163	6	
10	54	Pass	yh_L	30/03/2005 08:09	248	6	1	26	202	6	
11	54	Pass	yh_L	30/03/2005 08:24	177	6	1	22	136	3	
12	54	Pass	yh_L	30/03/2005 08:39	208	6	1	24	166	4	
13	54	Pass	yh_L	30/03/2005 08:54	252	5	0	22	210	7	
14	54	Pass	yh_L	30/03/2005 09:09	173	7	0	25	124	5	
15	54	Pass	yh_L	30/03/2005 09:24	211	7	1	20	171	6	

注: CA は、日付表示形式が予想通りであることを確実にするために、(Excel のみではなく) メモ帳で CSV ファイルを確認することを推奨します。Excel は、マシンの地域設定に従って日付をフォーマットする傾向があり、アダプタによって実際に参照される形式と一致していない場合があります。

Topaz_20050330_200000.csv - Notepad

File Edit Format View Help

```

Profile,Status,TopazTransaction,Time,ResponseTime,wtdConnectionTime,wtdDNS
54,Pass,yh_L,30/03/2005 06:09:00,218,6,1,23,179,2
54,Pass,yh_L,30/03/2005 06:24:00,270,7,1,25,226,4
54,Pass,yh_L,30/03/2005 06:39:00,209,6,1,21,171,3
54,Pass,yh_L,30/03/2005 06:54:00,243,5,0,22,204,3
54,Pass,yh_L,30/03/2005 07:09:00,207,6,1,21,167,4
54,Pass,yh_L,30/03/2005 07:24:00,245,5,1,24,203,4
54,Pass,yh_L,30/03/2005 07:39:00,218,6,1,22,180,3
54,Pass,yh_L,30/03/2005 07:54:00,207,7,1,24,163,6
54,Pass,yh_L,30/03/2005 08:09:00,248,6,1,26,202,6
54,Pass,yh_L,30/03/2005 08:24:00,177,6,1,22,136,3
54,Pass,yh_L,30/03/2005 08:39:00,208,6,1,24,166,4
54,Pass,yh_L,30/03/2005 08:54:00,252,5,0,22,210,7
54,Pass,yh_L,30/03/2005 09:09:00,173,7,0,25,124,5
54,Pass,yh_L,30/03/2005 09:24:00,211,7,1,20,171,6
54,Pass,yh_L,30/03/2005 09:39:00,233,10,1,28,183,5
54,Pass,yh_L,30/03/2005 09:54:00,166,7,0,23,124,6
54,Pass,yh_L,30/03/2005 10:09:00,177,7,1,26,127,7

```

アダプタ作成プロセスを開始する前に、データ ソースおよび関連する KPI について必要なすべての分析および調査を行い、以下の事項を確認することも重要です。

- ビジネス ロジックが必要とするフィールド
- ファイル内の日付形式
- ソース ファイル内の日付/時間値のタイム ゾーン (これらのタイム ゾーンは、アダプタ作成プロセスを開始する前に、システムに作成する必要があります)
- 空白/NULL エントリが存在する可能性がある日付フィールドの有無
- データ ソースの動作 (すべてが追加のレコードか、または前のイベントを更新するレコードか)
- KPI に必要なドリルダウン (これは、「リソース」の選択に影響する場合があります)
- ビジネス ロジック内のイベントを効果的にフィルタリングするために必要な「イベント タイプ」

これらの事項を確認したら、作成プロセスを開始できます。

アダプタ ウィザードを使用し、このシナリオに基づいてアダプタ作成プロセスを実行できるようになりました。

このシナリオでは、リソースとして「TopazTransaction」、イベント タイプとして「Profile」、およびタイムスタンプとしての「Time」フィールドを使用します。また、ビジネス ロジックで使用するために、イベント構造に「Status」、「ResponseTime」、および「WtdDownloadTime」フィールドを追加します。

アダプタの作成

まず、システムが新しいアダプタを作成して、サーバに適切に展開する準備ができていることを確認するために、以下のサービスがアプリケーション サーバで開始されていることを確認します。

- アダプタ展開サービス
- アダプタ リスナ サービス

次に、[アダプタ] ページに移動して、新しいアダプタを作成します。[アダプタ] ページで、[新規追加] - [テキスト ファイル アダプタ] を選択します。

全般手順

アダプタ ウィザードの [全般] 手順で、以下のフィールドに入力します。

- 名前: アダプタに適切な名前を指定します。

- **アダプタアドレス:** LOCALHOST がデフォルトのオプションです（アプリケーションサーバの展開の場合）が、必要に応じて、横にあるボタンを使用してほかのアドレスを入力できます。
- **時間形式:** これは、データソース内の日付/時間フィールドで使用されるデフォルトの時間形式です。これを適切に選択すると、この後のウィザードプロセスでフィールドが自動的に正しく検出されます。必要に応じて、このフィールドの横にあるボタンを使用して、新しい時間形式を入力できます。
- **タイムゾーン:** これは、データレコードが記録されるタイムゾーンです。これは、内部の時間シフトを適切に行われるように、アダプタが event_timestamp フィールド（およびその他の日付/時間フィールド）を UTC に正しくシフトするために必要となります。これは、前のセクションのデータソースチェックリストに従って事前に入力されます。

注:

また、このページには「詳細」ボタンがあり、アダプタの多くの設定パラメータにアクセスできます。変更が必要ではない場合、これらのほとんどはデフォルト値のままにしておくことができます。

アダプタの「ポート」は、CA Business Service Insight が処理を進めると、アダプタに自動的に割り当てられますが、必要に応じてここで上書きできます。変更可能なその他の主なパラメータには、次のものがあります：地域別設定、[モード]（[オンライン] / [オフライン]）、接続の詳細、[モニタリング]およびログ記録のオプション、一度実行/常時実行の設定、エラーの制限、ファイル名、およびコメント。

The screenshot shows a dialog box titled '全般 - 詳細設定 -- Webpage Dialog'. It has several tabs: '全般' (General), '接続性' (Connectivity), 'モニタリング' (Monitoring), '処理' (Processing), 'その他' (Other), and 'メモ' (Notes). The '全般' tab is active. In the 'モード' (Mode) section, there are two radio buttons: 'オンライン' (Online) which is selected, and 'オフライン' (Offline). Below this, the 'ポート' (Port) field is marked with an asterisk and contains the text '6218'; this field is circled in red. Further down, under the heading '地域別設定の詳細' (Regional Settings Details), there are two more fields: '小数点' (Decimal) marked with an asterisk and '数字桁区切り記号' (Numeric separator) marked with an asterisk. Both of these fields are currently empty.

このケーススタディでは、これらの設定をそれぞれ検討しませんが、「[アダプタ設定の指定](#) (P. 339)」で説明しています。

[次へ] をクリックして、ウィザードの次の手順を続行します。次の手順では、アダプタの「データソースインターフェース」にアクセスできます。

[データ ソース インターフェース]手順

アダプタ ウィザードの [データ ソース インターフェース] 手順で、以下のフィールドに入力します。

- **データ ソース名**: この特定のソース ファイルの名前 (1 つのアダプタに複数のソース ファイルを指定できます)。
- **ファイルパス**: アプリケーション サーバ (またはその他のサーバ) 上のソース データ ファイルが存在する場所へのパス。 アプリケーション サーバ以外のサーバの場合は、UNC 記法を使用します (たとえば、//server01/sourcefolder)。
- **名前パターン**: ワイルドカード文字と共に使用して、アダプタによってロードされた、[ファイルパス] にあるファイルをフィルタします。

[解析定義] タブも、インポートされているファイルの構造体を定義するために利用できます。以下のようにフィールドを使用します。

- **タイトル**: データ ファイルに「タイトル」行がある (つまり、CSV の最初の行にタイトル名があり、以降の行に値が続く) かどうかを指定するブール値のチェック ボックス。
- **区切り文字**: 個別のフィールドを区別するファイル内の区切り文字を指定します。

注: また、ここにはほかに [詳細] ボタンが 2 つあり、追加の設定オプションを指定できます。1 つのボタンは [詳細] タブにあり、もう 1 つは [解析定義] タブにあります。[詳細] タブの [詳細] ボタンからは、以下のパラメータにアクセスできます。

アクティブ データ ソース: アダプタのこの特定のソース セクションをオン/オフにできるブール型のチェック ボックス

処理後: 処理の後にファイルを削除するか、または維持するかを指定できます。

初期ファイル名: (特定のディレクトリのすべてのファイルをロードしない場合に) 処理を開始する最初のファイルの名前を設定します

データを *確認* する *間隔*: アダプタが継続的に実行されるように設定されている場合に、新しいファイルを確認する間隔を定義します

[解析定義] タブの [詳細] ボタンからは、複数行のレコードなどのレコード終了オプションを定義するための設定にアクセスできます。

詳細および解析定義を設定すると、ソース ファイルのサンプルをウィザードにロードして、設定オプションをテストし、ファイルの内容をプレビューできます。

[参照] ボタンをクリックすると、サンプル ファイルを下のプレビュー ウィンドウにロードできます。サンプル ファイルに移動し、[テスト ファイル] ボタンを押します。これはオプションの手順です。

注: 参照機能は、作業しているローカル マシンで開きます。作業しているのがアプリケーション サーバではない場合、参照機能が動作するためには、サーバ上にあるソース ファイルのコピーを持っている必要があります。この機能を使用して直接サーバを参照することはできません。

ファイルがロードされると、その内容がプレビュー ウィンドウに下のように表示されます。

注: 「フィールド」の名前が、検出されたフィールド タイプ (整数、文字列、時間など) と共にヘッダに表示されます。次に進む前に、これらが要件に従って正しく検出されたことを確認してください。特に、以下の事項を確認します。

- タイムスタンプが「時間」として検出されている - これは、ウィザードの最初の手順で指定した [時間形式] に従って処理されます
- リソースが「文字列」として検出されている

ファイルのプレビューを確認したら、[次へ] をクリックします。[マッピング手順] が表示されます。

マッピング手順

アダプタ ウィザードの次の（そして最後の）手順では、変換タスクを実行し、データ ソースの入力フィールドを **CA Business Service Insight** の「イベント」を形成する出力フィールドにマップできます。システムに「イベント タイプ」がすでに作成されているかどうかによって、続行するための 2 つのオプションがあります。また、設定のために選択可能で、命名規則に従うようにすることができるその他の多数のオプションがあります。ただし、これらのほとんどは省略可能であり、処理を簡略化し、必要な手順を減らすためのものです。必須の手順を以下に示します。

マッピングの手順を設定する手順は、以下のとおりです（オプションの手順を含む）。

1. 入力フォーマットの名前を指定します（複数の入力を持つアダプタの場合に役立ちます）
2. 必要な追加のフィールド（定数値、データ ソース、タイトル、ファイル名、または複合フィールド タイプなど）を加えます
3. 必要な入力検証基準を作成します
4. 既存のイベント タイプを選択するか、新しいイベント タイプを作成します（必須）
5. 入力から出力にフィールドをマッピングします（必須）
6. 出力フォーマットの名前を指定します
7. **ResourceId**、タイムスタンプ、およびイベント タイプをマッピングします
8. 必要な出力検証基準を作成します
9. イベントの「**OnDuplication**」設定を指定します

新しいイベント タイプの作成を選択すると、新しいウィンドウが表示され、メイン画面の入力フォーマットに基づいて自動入力されます。ただし、イベント タイプの名前を入力し、イベント タイプにリソース タイプを割り当てる必要があります。

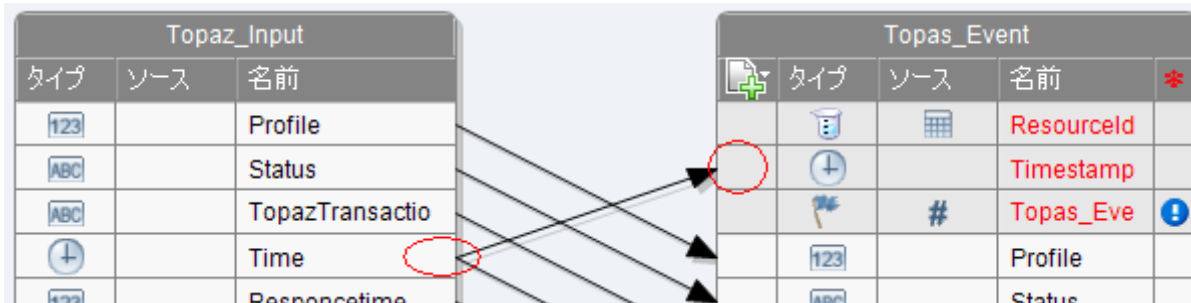
完了すると、[保存して閉じる]ことができ、マッピングが自動的に完了します。

[イベントタイプの選択] オプションを選択すると、システムで選択できる既存のイベントタイプのリストが表示されます。ただし、次に進むときに、イベントタイプの名前およびデータタイプと一致する入力フィールドのみが自動でリンクされます。残りのフィールドは手動でマップする必要があります。

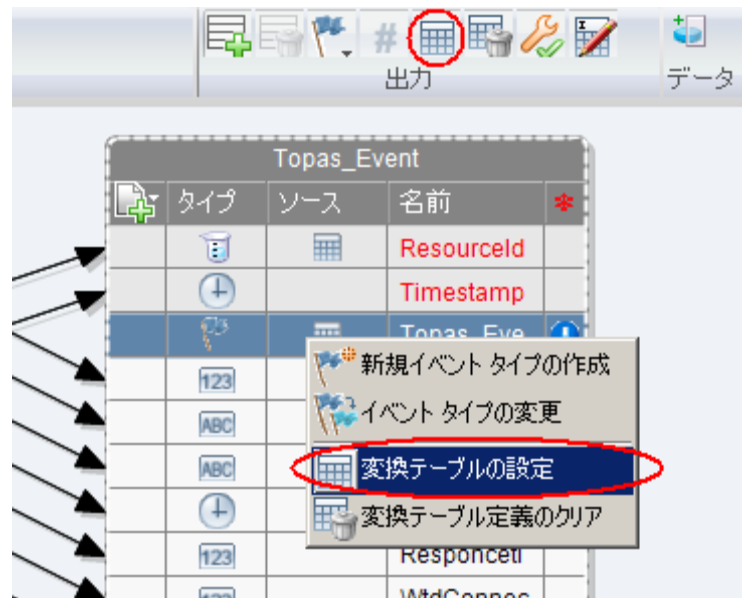


次の手順では、ResourceId、Timestamp、およびイベント タイプを設定します。フィールドがすでに存在する（前の手順で作成されます）場合、必要に応じてそれらのフィールドにリンクできます。

このインターフェースでは、ドラッグアンドドロップスタイルの関連付けがサポートされます。関連付けが設定しても保持されない場合は、それぞれの型が一致していることを確認します（つまり、時間/文字列/整数など）。



- ResourceId は、（分析で識別された）要件に従って、入力内の文字列値からマップする必要があります。
- Timestamp には、イベントが発生した時間を定義する時間変数が設定される必要があります、計算のために使用されます。
- イベント タイプには、前の画面の [データ ソース名] フィールドに従って、定数値がデフォルト設定されます。これは必要に応じて上書きできます。これは、特定のフィールドの値に従って、複数のイベントをこのアダプタから送信する場合（つまり、入力フィールドの値に応じて、異なるイベントを送信する場合）に必要となります。これを行うには、イベント タイプ行を右クリックし（または、下の画像に示されているように、上にあるボタンをクリックして）、[変換テーブルの設定] を選択します。



これにより、ポップアップウィンドウが表示されます。

変換テーブル詳細 -- Web ページ ダイアログ

変換テーブル詳細

名前: イベント タイプ ID

ソース タイプ: テーブル

タイプ: 整数

テーブル名: *

デスティネーション タイプ:

変換前: *

フィールド名 フィールド値

新規変換テーブルの追加

この値フィールドの変換テーブルがすでに存在する場合は、ここで選択できます。または、この値に新しい変換テーブルをセットアップできます。上の画面のボタンはこのために使用できます。

変換テーブル詳細 -- Web ページ ダイアログ

変換テーブル詳細

名前: * Topaz_Events_Table

説明: A translation table strong the different Topaz Events that are recieved from the souce system

ソース フィールド: *

ソース フィールドの追加

フィールド名

event_type

デスティネーション タイプ: イベント タイプ

テーブルを作成したら、上記で指定した [ソース フィールド] にマップする入力フィールドを指定する必要があります。

アダプタのリソースに別の変換テーブルを指定する場合は、それもこの時点で行う必要があります。これは、上記で説明したイベント タイプと同様のプロセスを使用して設定します。

注: デフォルトでは、アダプタ ウィザードは、特に指定されていない場合、「Default_Translation_Table」という既存の変換テーブルにすべてのリソースを割り当てます。これは、単純な実装の場合は適切ですが、より複雑な実装の場合（およびデータを区別する目的の場合）、CA は別のテーブルを使用することを推奨します。アダプタ マッピング セクションの [ソース フィールド] が異なるか、複数の値が含まれる場合も必須となります。

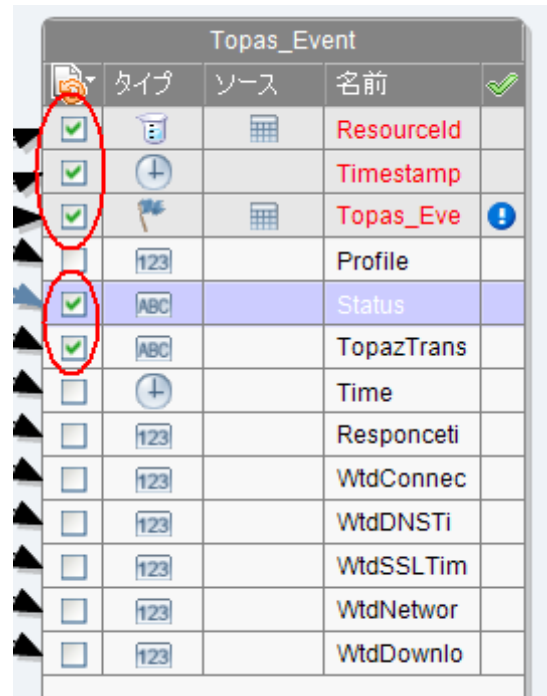
マッピング手順の最後の手順では、アダプタの「OnDuplication」を設定します。この設定では、すべての「キー」フィールドの値が一致する 2 つ目のイベントを受信したときに取るアクションを記述します。この一意の「キー」は、アダプタの各出力に定義できます（この詳細については、以降の説明を参照）。デフォルトでは、この「OnDuplication」値には [追加] が設定されるため、変更する必要があります。別のアクションが必要な場合のみです。使用できる値は以下のとおりです。

- **追加:** 新しいイベントは、個別の新しいイベントとして追加されます
- **無視:** 新しいイベントは、アダプタによって無視（破棄）されます
- **更新:** イベントの「値」フィールドが異なる場合にのみ、以前にロードされたイベントが新しいイベントに置き換わります
- **常に更新:** 以前にロードされたイベントが新しいイベントに置き換わります

[追加] 以外のオプションを使用すると、アダプタのパフォーマンスに影響する場合があります、特にデータ量が非常に多い場合は、実装する前に注意深く検討する必要があります。



値に [追加] 以外を設定した場合、一意の「キー」を定義するために使用される一連のチェックボックスが出力構造に表示されます。「キー」は、チェックボックスがオンにされている各項目から構成されます。キーの選択は、注意深く分析してから、要件に基づいて決定する必要があります。



マッピングセクションの設定が完了したら、画面の右下にある [完了] ボタンをクリックします。システム内のアダプタのリストに戻ります。作成したアダプタが表示され、そのステータスには「非アクティブ」と示されます。

アダプタ

検索

新規追加

5 件中 1 - 5 件を表示します。

   	名前	モード	タイプ	開始者	アダプタ アドレス	ポート	ステータス
   	AccessDB2	オンライン	SQL アダプタ	サーバ	LOCALHOST	6203	非アクティブ
   	AvailabilityDown	オンライン	SQL アダプタ	サーバ	LOCALHOST	6213	停止中
   	AvailabilityMonitoring	オンライン	SQL アダプタ	サーバ	LOCALHOST	6204	非アクティブ
   	AvailabilityUp	オンライン	SQL アダプタ	サーバ	LOCALHOST	6212	停止中
   	Topaz Adapter	オンライン	テキスト ファイル アダプタ	サーバ	LOCALHOST	6219	非アクティブ

アダプタ展開

アダプタの設定が完了しました。このアダプタをアプリケーション サーバに展開する必要があります。[アダプタの開始] ボタンをクリックすると、アダプタ展開サービスがファイル システムにアダプタを展開します。これには、以下の処理が含まれています。

- アダプタの実行に必要なフォルダの作成
- そのフォルダへの XML 設定のコピー
- アダプタを実行するためのショートカットの作成

これが完了すると、変更がサーバに反映されます。

これで、GUI からアダプタを実行できるようになりました。アクティブになった [今すぐ実行] ボタンをクリックします。これにより、アダプタ展開サービスがサーバ上でアダプタを実行し、データ収集が開始されます。



アダプタが実行されると、システムの保留中の変換のセクションに保留中のリソースおよびイベントが表示されます。

変換エントリ

検索

▼

エントリの追加

▼

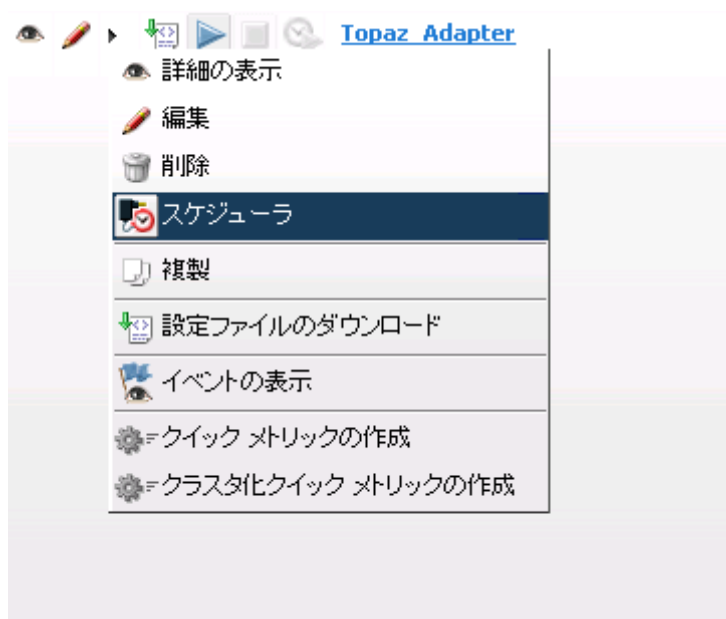
22件中 1 - 22 件を表示します。

☐				テーブル名	デスティネーション タイプ	ステータス	変換前
☐				Topaz Events Table	イベント タイプ	保留中	hp_A
☐				Topaz Events Table	イベント タイプ	保留中	hp_L
☐				Topaz Events Table	イベント タイプ	保留中	hpmta_A
☐				Topaz Events Table	イベント タイプ	保留中	hpmta_L
☐				Topaz Events Table	イベント タイプ	保留中	motormta_A
☐				Topaz Events Table	イベント タイプ	保留中	motormta_L
☐				Topaz Events Table	イベント タイプ	保留中	motorNBA
☐				Topaz Events Table	イベント タイプ	保留中	yh_A

[保留中] のエントリは、通常のシステム設定に従って変換できます。変換が完了したら、システムに Raw データをロードするために、アダプタを再度実行する必要があります。

アダプタのスケジュール

アダプタの実行に加えて、アダプタのスケジュールを GUI から設定することもできます。ただし、これを行うためには、アダプタが「停止」ステータスである必要があります。停止させたら、アダプタのコンテキスト メニューを表示して、[スケジュール] を選択することにより、スケジュールを設定できます。



スケジュールのオプションは、Windows のタスク スケジューラによって提供されるものと同じです。アダプタ展開サービスは、この操作の背後で、実際にはタスク スケジューラにこのスケジュールをアイテムとして作成します。

The screenshot shows a dialog box titled "スケジューラ" (Scheduler) with a subtitle "Web ページ ダイアログ". The dialog is divided into several sections for configuring a task schedule.

開始時刻: (Start Time) Set to 00:00, with a dropdown for "適用: GMT" (Apply: GMT) and a link to "タイムゾーン" (Time Zone).

繰り返しのパターン: (Repeat Pattern) Includes radio buttons for "日単位" (Daily), "週単位" (Weekly), and "月単位" (Monthly). The "日単位" option is selected, with a sub-section for "間隔" (Interval) set to 1 "日おき" (days).

繰り返しの範囲: (Repeat Range) Includes a "開始日:" (Start Date) field set to 2011/07/01. It has two options: "終了日なし" (No end date) which is selected, and "終了日:" (End Date) which is empty.

タスクを繰り返す (Repeat task) is checked. Below it, the "間隔" (Interval) is set to "分" (minutes) with a value of 05. The "期間:" (Duration) is set to "時間" (hours) with a value of 23 and "分" (minutes) with a value of 59.

At the bottom are "OK" and "キャンセル" (Cancel) buttons.

スケジュールを追加して、アダプタが次に展開されると、タスクが実行されるサーバのアカウントのユーザ認証情報を入力するように求められます。これには、CA Business Service Insight システムを実行するために作成されたサービス アカウント（デフォルトは OblicoreSrv）、または必要に応じて別の管理アカウントを入力する必要があります。

スケジュール用認証情報 -- Webpage Dialog

スケジュール用認証情報の設定

ユーザ名 *

パスワード *

ローカル ユーザ ☐

ドメイン *

保存 キャンセル

この統合されたスケジューラは、非常に便利な機能であり、GUI のユーザがアプリケーションサーバのデスクトップに直接アクセスすることなく、アダプタのスケジュールを制御できます（もちろん、該当するユーザ権限は保留されます）。

ケース スタディ 21: LDAP との統合の例

組織の要件

組織の LDAP サーバにすでに定義されている既存のユーザを使用します。また、組織のポータルは、CA Business Service Insight ログイン、およびシングル サインオン (SSO) ポータルの CA Business Service Insight サイレント ログイン機能を使用したアクセスに使用されます。

CA Business Service Insight システム (LDAP 同期) に自動的にユーザを作成する Visual Basic (VB) 変換スクリプトを定義します。この変換スクリプトは、組織の LDAP サーバに接続し、そこからユーザのリストを抽出するために使用されます。ユーザ、グループ、および役割の作成には、CA Business Service Insight ツールパッケージのメソッドが使用されます。

LDAP に接続する VB コードの例

```
Option Explicit On
Imports System.DirectoryServices

Public Function GetLDAPUsers(ByVal ldapServerName As String, ByVal pFindWhat As String) As ArrayList
    Dim oSearcher As New DirectorySearcher
    Dim oResults As SearchResultCollection
    Dim oResult As SearchResult
    Dim RetArray As New ArrayList
    Dim mCount As Integer
    Dim mIdx As Integer
    Dim mLDAPRecord As String
    Dim ResultFields() As String = {"securityEquals", "cn"}
    Try
        With oSearcher
            .SearchRoot = New DirectoryEntry("LDAP://" & ldapServerName & _
"/dc=lippogeneral,dc=com")
            .PropertiesToLoad.AddRange(ResultFields)
            .Filter = "cn=" & pFindWhat & "*"
            oResults = .FindAll()
        End With
        mCount = oResults.Count
        If mCount > 0 Then
            For Each oResult In oResults
                mLDAPRecord =
oResult.GetDirectoryEntry().Properties("cn").Value & " " &
oResult.GetDirectoryEntry().Properties("mail").Value
                RetArray.Add(mLDAPRecord)
            Next
        End If
    Catch e As Exception
        MsgBox("Error is " & e.Message)
    Return RetArray
End Function
```



```

        End Try

        Return RetArray
    End Function

```

```

Sub CheckAddUser
Dim map
Set map = Tools.GetUserDetails("acme@test")
'Check user already exists
'Tools.AddUserByMap map
'Check with duplicate
map("UserName") = "acme2"
map("UserPassword") = "acme2"
map("UserPasswordExpirationInterval") = "50"
map("UserDescription") = "New description"
map("UserStatus") = "INACTIVE"
Tools.AddUserByMap map
Tools.Commit
End Sub

```

CA Business Service Insight VB 変換スクリプトのメソッド

- 組織のメソッド
 - AddOrganization / IsOrganizationExists
- 役割のメソッド
 - IsRoleExists / SearchRoles
- ユーザのメソッド
 - AddUserByMap / GetUserName
 - GetOrganizationName / IsUserExists
 - GetUserDetails / SearchUsers
 - GetUserFullName / UpdateUserByMap
- ユーザ グループのメソッド
 - AddUserGroupByMap / IsUserGroupExists
 - DeleteUserGroup / SearchUserGroups
 - GetUserGroupDetails / UpdateUserGroupByMap

「サイレント ログイン」を行うコードを作成して、CA Business Service Insight ログインに使用される組織のポータルに統合します。

CA Business Service Insight ゲートウェイの C# コードの例(組織のポータルに統合される)

```

using System;
using System.Data;

```

```

using System.Configuration;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Web;
using System.Web.Security;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Security.Cryptography.X509Certificates;
using OblicoreAuthenticationWebService;

namespace Oblicore.SSO
{
    /// <summary>
    /// This sample page is a sample gateway to Oblicore Guarantee(tm) application
    interface
    /// The page should be called prior navigating to any page at Oblicore Guarantee
    website
    /// or any page using Web Services supplied by Oblicore
    /// The OblicoreGateway page should perform the following actions:
    ///      1) Call Oblicore Authentication Web service in order to authenticate
    current user
    ///      2) Call SilentLogin.asp page at Oblicore website to login silently
    at Oblicore website
    ///      and create user session context
    ///      3) Redirect to desired page
    /// </summary>
    public partial class _Default : System.Web.UI.Page
    {

        /// <summary>
        /// Oblicore user credentials
        /// </summary>
        struct UserCredentials
        {
            public string UserName;
            public string Organization;
        }

        private void Page_Load(object sender, System.EventArgs e)
        {

            if (Request["OGSESSIONID"]!=null)
            {
                //We have been redirected back to this page from
                SilentLogin.asp after authentication.
            }
        }
    }
}

```

```

        //Save OGSESSIONID in cookie for further use
        HttpCookie SessionCookie = new
HttpCookie("OGSESSIONID",Request["OGSESSIONID"]);
        Response.Cookies.Add(SessionCookie);
        //Redirect to desired page
        Response.Redirect("/");
    }
    else
    {
        //First time we enter the page.
        //Let's perform authentication.

        string sAuthToken = string.Empty;

        // Obtain OG user name and organizations from portal user
directory
        UserCredentials ucOblicoreUser =
GetOblicoreUserCredentials();

        //Initialize Oblicore Authentication WebService
        //Project should include Web Reference to the service
        //Service is located on Oblicore Guarantee website at
/WebServices/OblicoreAuth.asmx
        OblicoreAuth oAuthService = new OblicoreAuth();
        // oAuthService.ClientCertificates.Add(x509);
        oAuthService.Url = "https://" + "localhost" +
"/WebServices/OblicoreAuth.asmx";
        try
        {
            //Invoke authentication Web Service.
            //The AuthenticateUser method returns encrypted token,
which should be passed to
            //SilentLogin.asp page, located in root folder of
Oblicore Guarantee website
            sAuthToken =
oAuthService.AuthenticateUser(ucOblicoreUser.UserName,ucOblicoreUser.Organization
);
        }
        catch (Exception ex)
        {
            //Proceed authentication error if any
            Response.Write("The error has occurs during Oblicore
authentication: " + ex.Message);
            Response.End() ;
        }

        //Call SilentLogin.asp page along with passing it
authentication folder
    }
}

```

```

//SilentLogin.asp page is located Oblicore Guarantee website
root folder

//After logging in, SilentLogin.asp page will redirect us
back to the current page along with passing OGSESSIONID parameter

//Response.Redirect(ConfigurationSettings.AppSettings["OGURL"].ToString() +
"/SilentLogin.asp?AuthToken="+Server.UrlEncode(sAuthToken)+"&DesiredPage="+GetCur
rentPageURL());

Response.Redirect("https://vit-05/SilentLogin.asp?AuthToken=" +
Server.UrlEncode(sAuthToken) + "&DesiredPage=/Oblicore.asp"); // +
GetCurrentPageURL());
    }
}

/// <summary>
/// Obtain Oblicore Guarantee user name and organization from portal user
directory
/// The method is supposed to call ActiveDirectory or another repository
using portal API
/// to obtain current user name and organization in terms of Oblicore
Guarantee
/// </summary>
/// <returns>Oblicore Guarantee user credentials struct</returns>
private UserCredentials GetOblicoreUserCredentials()
{
    UserCredentials ucOblicoreUser = new UserCredentials();
    //currently alwasy assume user is sadmin and organization is
Oblicore (default)
    ucOblicoreUser.UserName = "sadmin";
    ucOblicoreUser.Organization = "Oblicore";
    return ucOblicoreUser;
}

/// <summary>
/// Retrieves current page URL
/// </summary>
/// <returns>Full URL of current page</returns>
private string GetCurrentPageURL()
{
    string s =
(Request.ServerVariables["HTTPS"]==null||Request.ServerVariables["HTTPS"].ToLower
()=="off")?"http://":"https://";
    s += Request.ServerVariables["SERVER_NAME"] +
Request.ServerVariables["URL"];
    if (Request.QueryString.ToString() != string.Empty)

```

```

        {
            s += "?" + Request.QueryString.ToString();
        }
        return s;
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        //
        // CODEGEN: This call is required by the ASP.NET Web Form Designer.
        //
        InitializeComponent();
        base.OnInit(e);
    }

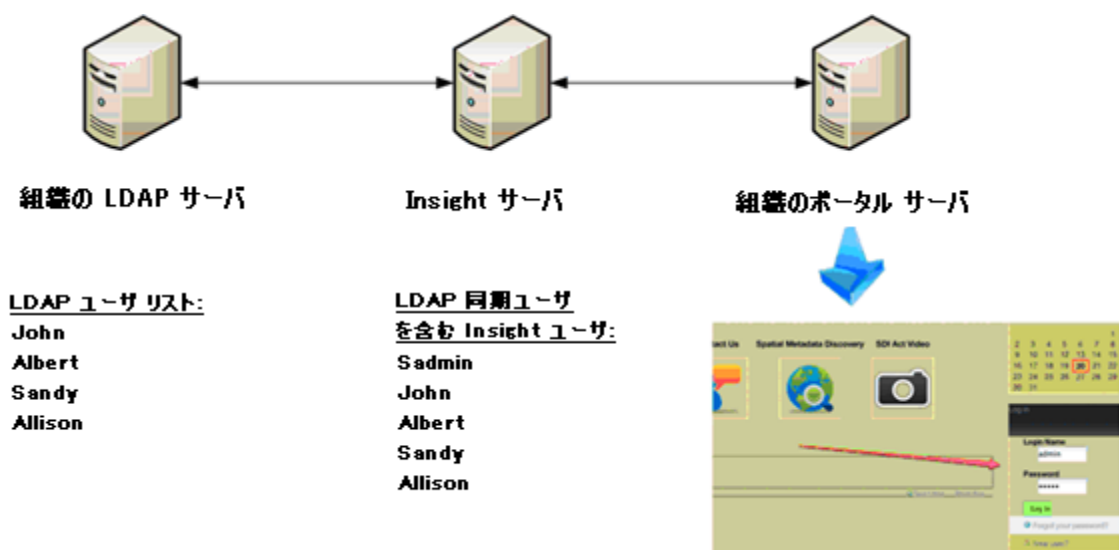
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
}

```

■ フロー ダイアグラム

以下のダイアグラムは、ユーザ同期プロセスおよびポータル アクセスのフローを示しています。変換スクリプトは定期的に実行されるように設定されています。このスクリプトは、LDAP のユーザ リストを最新の状態に維持し、必要に応じてユーザを追加/削除します。

ユーザが組織のポータルにログインします。ポータルは、CA Business Service Insight サーバにそれらをリダイレクトするか、別の利用可能なアプリケーションのリストを表示するように設定できます。CA Business Service Insight サーバは、最初のポータルへのログインで指定された認証情報を使用します。



ケース スタディ 22: PERL を使用してレポートを生成する

以下の例では、PERL スクリプトを使用して、CA Business Service Insight レポート Web サービスに接続し、HTTP ストリームを使用して条件 XML パラメータを SOAP エンベロープで転送する方法を示します。

注: SOAP エンベロープで転送される XML コードには、「<」または「>」記号を含めることはできないため、これらの記号には HTML コード（つまり、<=>、>=<）を使用します

```
#!/usr/bin/perl

#use strict;

use LWP::UserAgent;
use HTTP::Request::Common;
use XML::Simple;
use Data::Dumper;

my $userAgent = LWP::UserAgent > new(agent => 'Mozilla/5.0');

### Creating a Oblicore Session Context - Oblicore Session ID is stored in $scid ###

my $message = "<?xml version='1.0' encoding='utf-8'?">
<soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <CreateSessionContext xmlns='http://www.oblicore.com'>
      <userName>sadmin</userName>
      <organizationName>Oblicore</organizationName>
    </CreateSessionContext>
  </soap:Body>
</soap:Envelope>";

my $response = $userAgent > request(POST
'http://obonob/WebServices/OblicoreAuth.asmx',Content_type => 'text/xml', Content
=> $message);

print $response > error_as_HTML unless $response > is_success;

my $result = $response > as_string;

my $xmlstart      = index $result, "<?xml";
my $xmlend        = length $result;

my $xmltext       = substr $result, $xmlstart, ($xmlend - $xmlstart);

my $xml           = new XML::Simple;
```

```

my $data          = $xml > XMLin($xmltext);

my $scid = ($data > {'soap:Body'} > {'CreateSessionContextResponse'} >
{'CreateSessionContextResult'});

print "Session ID : ".$scid."\n";

### Try to get contract list from Oblicore ###

my $criteria_xml =
"<&lt;Criteria&&gt;&&lt;Name&&gt;a*&&lt;/Name&&gt;&&lt;Status&&gt;EFFECTIVE&&lt;/Status&&gt;
&&lt;/Criteria&&gt;";

print "<?xml version='1.0' encoding='utf-8'?">
<soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <GetContractsAdvanced xmlns='http://www.oblicore.com'>
      <sessionID>".$scid."</sessionID>
      <criteriaXML>".$criteria_xml."</criteriaXML>
    </GetContractsAdvanced>
  </soap:Body>
</soap:Envelope>";

my $message = "<?xml version='1.0' encoding='utf-8'?">
<soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <GetContractsAdvanced xmlns='http://www.oblicore.com'>
      <sessionID>".$scid."</sessionID>
      <criteriaXML>".$criteria_xml."</criteriaXML>
    </GetContractsAdvanced>
  </soap:Body>
</soap:Envelope>";

#my $message = "<?xml version='1.0' encoding='utf-8'?">
#<soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
#xmlns:xsd='http://www.w3.org/2001/XMLSchema'
#xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
#  <soap:Body>
#    <GetContracts xmlns='http://www.oblicore.com'>
#      <sessionID>".$scid."</sessionID>
#    </GetContracts>
#  </soap:Body>
#</soap:Envelope>";

### is_well_formed($message)

```



```

print $message;

my $response = $userAgent > request(POST 'http://obonob/WebServices/Contracts.asmx',
Content_Type => 'text/xml', Content => $message);

print $response > error_as_HTML unless $response > is_success;

my $result = $response > as_string;

print Dumper($result); # Output of contract list

### Close the Oblicore Session ###

my $message = "<?xml version='1.0' encoding='utf-8'?">
<soap:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <ClearSessionContext xmlns='http://www.oblicore.com'>
      <sessionContextID>".$scid."</sessionContextID>
    </ClearSessionContext>
  </soap:Body>
</soap:Envelope>";

my $response = $userAgent > request(POST
'http://obonob/WebServices/OblicoreAuth.asmx', Content_Type => 'text/xml', Content
=> $message);

print $response > error_as_HTML unless $response > is_success;

```


付録 C: アダプタ設定の指定

このセクションには、以下のトピックが含まれています。

[全体の構造](#) (P. 339)

[General セクション](#) (P. 340)

[CA Business Service Insight Interface セクション](#) (P. 346)

[DataSourceInterface セクション](#) (P. 349)

[SQL インターフェース セクション](#) (P. 352)

[InputFormatCollection セクション](#) (P. 356)

[TranslationTableCollection セクション](#) (P. 361)

[TranslatorCollection セクション](#) (P. 363)

全体の構造

設定 XML ファイルの一般的な構造を以下に示します。

```
< AdapterConfiguration>
  <General...>
  <OblicoreInterface...>
  <DataSourceInterface...>
  <InputFormatCollection...>
  <TranslatorCollection...>
  <TranslationTableCollection...>
</ AdapterConfiguration>
```

各ノードについては、以下のセクションで説明します。

General セクション

General セクションは、以下の属性とサブノードで構成されます。

XML 構造 :

```
<General
MajorVersion="2345"
MinorVersion="01245"
WorkingDirectoryName=" output"
DataSourceControlFileName="DatasourceControl.xml"
SendFileName="Send.txt"
SendControlFileName="SendControl.xml"
LogDebugMode="no"
ConsoleDebugMode="no"
RunOnce="yes"
RepeatUntilDry="yes"
RuntimeErrorLimit="1"
RetryRejectedEvents="yes"
RejectedEventsFileName="rejectedEvents.txt"
RejectedEventsUpperLimit="1000"
RegexUnlimitedSearch (V3.1 Patch)="no"
HoldRejectedEventsSpan="24"
GenerateStatistics="yes"
StatisticsFileName="MyStatistics.txt" >
KeepHistoricState="yes" >
DefaultTimeFormat="%Y/%m/%d-%H:%M:%S"
DefaultDecimalSymbol="."
DataSourceIdMaxLength="60"
DefaultDigitGroupingSymbol=","
SaveIllegalEvents ="no"
WriteEventErrorToLog ="yes"
IllegalEventsDirectoryName="xxxpath"
<DataSourceDifferenceFromUTC
    DefaultOffset="2"
    TimeFormat="%Y/%m/%d-%H:%M:%S">
    <DayLightSaving
        From="2001/04/09-12:00:00"
        To="2001/09/01-12:00:00"
        Shift="1"/>
    </ DataSourceDifferenceFromUTC >
</General>
```

- MajorVersion - メジャー バージョン番号を指定します。
- MinorVersion - マイナー バージョン番号を指定します。
- WorkingDirectoryName - (オプション)

アダプタ出力ファイル (データ ソース管理、変換、送信) 用のデフォルト パスを指定します。

デフォルト値 - 設定ファイルが格納されているディレクトリ内の出力ディレクトリ（フォルダ）

■ DataSourceControlFileName - （オプション）

データ ソースから最後に取得されたデータ レコードを追跡するためにアダプタが使用するファイルの名前

デフォルト値 - DataSourcecontrol.xml（他の値を指定しないと、この値がデフォルト値として使用されます。）

■ SendFileName - （オプション）

CA Business Service Insight に送信されるまでイベントを保持しておくファイルの名前

デフォルト値 - Send.txt（他の値を指定しないと、この値がデフォルト値として使用されます。）

■ SendControlFileName - （オプション）

送信プロセスを追跡するためにアダプタが使用するファイルの名前

デフォルト値 - SendControl.xml（他の値を指定しないと、この値がデフォルト値として使用されます。）

■ DataSourceCldMaxLength -

t_raw_data テーブル内の DataSourceCld フィールドの最大長。これより長い文字列を挿入すると、アダプタにエラーが表示されます。

この値は、テーブルの実際のサイズ以下にする必要があります。

デフォルト値 - 60

■ SaveIllegalEvents - この属性を選択した場合は、不正なイベントをファイルに書き込む必要があることを意味します。

デフォルト値 - no

■ WriteEventErrorToLog - データエラーを T_Log テーブルに書き込むかどうかを指示します。

有効な値 - yes/no

デフォルト値 - yes

■ IllegalEventsDirectoryName - （デフォルト値なし）

■ SendFileName - （オプション）

CA Business Service Insight に送信されるまでレコードを保持しておくファイルの名前

デフォルト値 - Send.txt（他の値を指定しないと、この値がデフォルト値として使用されます。）

■ \$SendControlFileName - （オプション）

送信プロセスを追跡するためにアダプタが使用するファイルの名前

デフォルト値 - SendControl.xml (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ LogDebugMode - (オプション)

有効な値 - yes/no

「yes」に設定すると、データソースの最初の行、解析結果、CA Business Service Insight 統合イベントがログに送信されます。

■ ConsoleDebugMode - (オプション)

有効な値 - yes/no

「yes」に設定すると、コンソールにデバッグメッセージが表示されます。

- レコードの読み取りと変換を示すインジケータ :

- .: 読み取られてから 1 つの出力イベントに変換されたレコードを表します。
- i: 正規表現の解析で無視されたレコードを表します。
- l: 読み取られてから変換テーブルで無視されたレコードを表します。
- R: 読み取られてから変換テーブルで拒否された (変換テーブルで変換できない) レコードを表します。
- X: 解析中に問題が発生したレコードを表します。このレコードは、無視されて失われるか、または不正なイベントディレクトリに保存されます。

注: 読み取られてから複数のトランスレータを通過したレコードは、かっこで囲まれて表示されます。たとえば、[...] や [RRI] のように表示されます。

- アダプタのステータス インジケータ :

- O: アダプタが稼働中であり、最後の 1 秒のレコードが読み取られていません。
- E: エラー ステータス
- P: 一時停止ステータス
- S: CA Business Service Insight から停止コマンドを受信しました。
- B: ブロック ステータス。拒否されたイベントのテーブルがいっぱいです。
- 変換テーブルのインジケータ :
 - L: 変換テーブルの待機中です。
 - T: 変換テーブルを CA Business Service Insight に送信しました/CA Business Service Insight から受信しました。

- **t** : CA Business Service Insight に送信した/CA Business Service Insight から受信した変換テーブルに変更が加えられています。
- **接続のインジケータ** :
 - **<Connecting CA Business Service Insight** : CA Business Service Insight の接続中です。
 - **<Connecting CA Business Service Insight>** : 接続が確立されました。
 - **<Connecting DataSource** : データ ソースの接続中です。
 - **<Connecting DataSource>** : 接続が確立されました。

- **RunOnce** - (オプション)

有効な値 - yes/no

「no」に設定すると、アダプタが継続して実行されます。

「yes」に設定すると、テキストファイルアダプタが実行され、レコードを読み取ってから自動的に停止します。ファイルアダプタは、すべてのファイルを読み取り、数秒間待機した後で新しいレコードの読み取りを試行します (SleepTime の設定によって異なる)。新しいレコードがない場合は、自動的に停止します。

SQL アダプタは、各クエリを 1 回だけ実行します。RepeatUntilDry を「no」に設定すると、すぐに自動停止します。RepeatUntilDry を「yes」に設定すると、待機した後で (SleepTime の設定によって異なる)、クエリを再実行し (クエリの SleepTime の設定によって異なる)、新しいレコードがない場合は、自動的に停止します。

- **RepeatUntilDry** - (オプション)

有効な値 - yes/no、デフォルト値 - yes

上記の RunOnce 属性を参照してください。

- **RuntimeErrorsLimit** - (オプション)

アダプタがブロックされるまでのエラー数の上限 (入力イベントでのエラー数の上限など) を設定します。0 に設定すると、アダプタがブロックされません。

デフォルト値 -1 (最初のエラーの後でアダプタがブロックされることを意味します。)

- **RetryRejectedEvents** - (オプション)

有効な値 - yes/no、デフォルト値 - yes

「yes」に設定すると、変換する必要のあるレコードが拒否されたイベントファイルに保持され、変換待ちの状態になります。

この属性を「no」に設定しないことをお勧めします。「no」に設定すると、重要なデータが失われる可能性があります。

■ **RejectedEventsFileName** - (オプション)

データ ソースから取得されたデータ レコードのうち、変換待ち状態にあるデータ レコードを追跡するためにアダプタが使用するファイルの名前

デフォルト値 - **rejected.txt** (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ **RejectedEventsUpperLimit** - (オプション)

拒否されたレコードが何件まで **rejected.txt** ファイルに保持されるかを設定します。アダプタは、この上限に到達すると、自動的に停止し、ブロック ステータスになります。このステータスは、これらのレコードが変換されるまで続きます。

デフォルト値 - **1000**

■ **RegExUnlimitedSearch** - 正規表現に基づいて完全検索を実行するようにアダプタに指示します。

有効な値 - **yes/no**

デフォルト値 - **no**

■ **HoldRejectedEventsSpan** - (オプション)

拒否されたイベント ファイルを何時間保存しておくかを設定します。この属性を指定しないと、アダプタは、それ自体が再起動する前に処理しておく必要のあるイベントを消去しません。

この属性を使用しないことをお勧めします。この属性を使用すると、重要なデータが失われる可能性があります。

■ **GenerateStatistics** - (オプション)

有効な値 - **yes/no**、デフォルト値 - **yes**

「**yes**」に設定すると、アダプタは統計情報を含む統計ファイルを 1 分おきに作成します。

■ **StatisticsFileName** - (オプション)

統計ファイルの名前

デフォルト値 - **statistics.txt** (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ **KeepHistoricState** - (オプション)

有効な値 - **yes/no**、デフォルト値 - **no**

「**yes**」に設定すると、アダプタは、ログ ファイルを除くすべてのファイルを「**Historic state yyyyymmdd-hhmmss**」という名前の新規ディレクトリに保存します。「**yyyyymmdd**」と「**hhmmss**」は、作成された日付と時刻の形式です。

■ **DefaultTimeFormat** - (オプション)

デフォルトの時間形式。この属性を指定した場合は、**TimeFormat** 属性が省略されている場所の時間形式として使用されます。この属性を指定しない場合は、他のエレメントで **TimeFormat** 属性を指定する必要があります。

■ **DefaultDecimalSymbol** - (オプション)

実数フィールドに使用されるデフォルトの小数点記号

デフォルト値 - (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ **DefaultDigitGroupingSymbol** - (オプション)

整数フィールドと実数フィールドに使用されるデフォルトの桁区切り記号

デフォルト値 - (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ **DataSourceDifferenceFromUTC** - UTC とマシンが配置されているデータ ソースのタイムゾーンとの時差を示します。

- **DefaultOffset** - 夏時間に入っていない場合の UTC からのオフセット
- **TimeFormat** - 夏時間詳細 (次の項目の説明を参照) を解析する際に使用する形式を指定します。
- **DayLightSaving** - データ ソース タイムゾーンの夏時間の期間を 1 つ指定します。このエレメントはオプションであり (指定しない場合は、夏時間が存在しないことを意味する)、複数回指定できます。複数のエレメントが存在する場合は、これらの期間が重ならないように時間別に順序を付ける必要があります。
 - **From** - 期間の開始日付
 - **To** - 期間の終了日付
 - **Shift** - 夏時間の期間内で **DefaultOffset** に加算されるシフト

CA Business Service Insight Interface セクション

CA Business Service Insight Interface セクションの各属性では、CA Business Service Insight への接続を指定します。オンラインとオフラインの 2 とおりのモードがあります。オンラインモードでは、アダプタは CA Business Service Insight に接続した後、CA Business Service Insight から変換テーブルと制御コマンドを取得し、イベント、ステータス、および変換リクエストを CA Business Service Insight に送信します。オフラインモードでは、アダプタはローカル変換テーブルファイルを使用し、出力ファイルにイベントを書き込みます。

オンラインモードの XML 構造：

```
<OblicoreInterface
  Mode="online">
  <OnlineInterface
    Port="3006"
    ConnectionInitiator="server"
    Address="localhost"
    SecurityLevel="high"
    SecurityKey="12345678901234567890123456789012"
    UseAcknowledgeProtocol="yes"
    PacketSize="50"
    PacketDeadline="60"
    PacketResendTimeout="60"
    WindowSize="10"
  />
</OblicoreInterface>
```

- Port - CA Business Service Insight サーバと通信するためにアダプタが使用する TCP/IP ポート番号
- ConnectionInitiator - (オプション)
有効な値 - server/adapter デフォルト値 - server

CA Business Service Insight 側で接続イニシエータ、アダプタ側、またはアダプタ リスナを定義します。

イニシエータがスレーブ (クライアント) の役割になり、もう一方がマスタ (サーバ) の役割になります。
- Address - サーバのネットワーク アドレス。イニシエータがアダプタの場合に必要となります。
- SecurityLevel - アダプタと CA Business Service Insight サーバ間の認証のレベルを定義します (ハンドシェイク)。レベルのオプション：
 - none - 認証が行われません。
 - high - 認証が行われます。SecurityKey を指定する必要があります。
- SecurityKey - 32 桁の文字列。CA Business Service Insight データベース内でアダプタに対して定義されている文字列と同じにする必要があります。

「ハンドシェイク」プロセスの流れを以下に示します。

- CA Business Service Insight サーバがアダプタに接続します。
- アダプタから CA Business Service Insight サーバにランダムな文字列が送信されます。
- CA Business Service Insight サーバは、事前に設定されたキーを含む文字列を暗号化してアダプタに返送します。
- アダプタは SecurityKey 文字列を使用して、CA Business Service Insight サーバに送信された同じランダムな文字列を暗号化し、受信した文字列と比較します。
- CA Business Service Insight サーバは、文字列をランダムに抽出してアダプタに送信します。
- アダプタは、その文字列を暗号化して CA Business Service Insight サーバに返送します。
- CA Business Service Insight サーバは同じ文字列を暗号化し、アダプタから受信した文字列と比較します。
- 接続が確立されます。
- 上記のいずれかの段階で誤りが検出された場合は、接続が確立されません。

■ UseAcknowledgeProtocol - (オプション)

有効な値 - yes/no、デフォルト値 - yes

「yes」に設定すると、アダプタは確認プロトコルを使用します。「no」に設定すると、アダプタは CA Business Service Insight からの確認応答を待たずにメッセージ/パケットを送信します。

この属性を「no」に設定しないことをお勧めします。「no」に設定すると、イベントが失われる可能性があります。

■ PacketSize - (オプション)

有効な値 - 1 ~ 1000

デフォルト値 - 50

1つのパケットに含まれるイベントの最大数

■ PacketDeadline - (オプション)

有効な値 - 1 ~ 3600

デフォルト値 - 60

不完全なパケットが送信されるまでの秒数

■ PacketResendTimeout - (オプション)

有効な値 - 1 ~ 3600

デフォルト値 - 60

パケットが再送信されるまで確認応答メッセージを待機する時間(秒単位)。
この属性は、UseAcknowledgeProtocol 属性が「yes」に設定されている場合にのみ適用されます。

■ WindowSize - (オプション)

有効な値 - 1 ～ 100

デフォルト値 - 10

ウィンドウ内のパケットの数。この属性は、UseAcknowledgeProtocol 属性が「yes」に設定されている場合にのみ適用されます。

オフラインモードのXML構造：

```
<OblicoreInterface
  Mode="offline">
  <OfflineInterface
    OutputFileName="outputEvents.txt"
    OutputFileType="tsv"
    OutputTimeFormat="%Y/%m/%d %H:%M:%S"
    OutputDecimalSymbol="."
  />
</OblicoreInterface>
```

■ OutputFileName - (オプション)

アダプタが出力イベントを書き込むファイルの名前。

デフォルト値 - AdapterOutput.txt (他の値を指定しないと、この値がデフォルト値として使用されます。)

■ OutputFileType - (オプション)

有効な値 - csv/tsv/xml

イベントの形式を定義します。

■ OutputTimeFormat - 日付フィールドの形式を定義します。General セクションで DefaultTimeFormat 属性が定義されている場合は、この属性を省略してもかまいません。

■ OutputDecimalSymbol - (オプション)

実数フィールドに使用される小数点記号を定義します。

デフォルト値 - . (小数点)

DataSourceInterface セクション

DataSourceInterface セクションは、アダプタおよびデータ ソース（測定ツール、CRM、システム ログなど）の間の接続および接続タイプを指定する属性から構成され、2 つの主要なタイプに分類されます：ファイル インターフェースおよび SQL インターフェース。

ファイル インターフェース

ファイル アダプタは、ログ ファイル、スケジュール済みレポート、またはその他のテキスト ベースのファイルからデータを取得するために使用できます。

DataSourceInterface は、ファイル データ ソースからの情報を解析し、それをフィールドに抽出するルールを定義します。

DataSourceInterface セクションでは、アダプタがソース ファイルを扱う方法（元のファイルがアダプタ用にのみ作成されている場合にファイルを削除するかどうか、またほかの目的で使用する場合にデータを維持するかどうかなど）も定義します。

XML 構造：

```
<DataSourceInterface WorkFileName="MyWorkingFile.txt" >
  <Files>
    <File
      IsActive="yes"
      InputFormat="events"
      Path="D:\adapters\sample_data%"
      NamePattern="adapterXXX*.log"
      DeleteFileAfterProcessing="no"
      Delimiters=","
      IgnoreRedundantDelimiters ="no"
      TitleExists="no"
      SleepTime="10">
    </File>
  </Files>
</DataSourceInterface>
```

- **WorkFileName** - （オプション）。 **DeleteFileAfterProcessing** が「no」に設定されている場合、ファイルはこの名前のファイルにコピーされ、「yes」に設定されている場合は、ファイルの名前がこの名前に変更されます。 指定されていない場合は、デフォルト値が使用されます（「WorkFile.txt」）。
- **Files** - ファイル エレメントのコレクション（1 つのアダプタに複数のファイルを定義できます）。
- **File** - ファイルの属性を指定します。
 - **IsActive** - （オプション） [yes/no]。 このファイルがアクティブであるかどうかを定義します（「no」に設定すると、このファイルは読み取られません）。

- **InputFormat** - このファイルに関連付けられている入力フォーマット。アダプタは **InputFormat** を使用してデータ ソースからデータを抽出します。
- **Path** - ソース データ ファイルの場所へのパス。
- **NamePattern** - データ ソースのファイル名を指定します。複数のファイルで同じ入力フォーマットが使用される場合は、ワイルドカードを使用できます。
- **DeleteFileAfterProcessing** [yes | no] - アダプタがソース ファイルを扱う方法。ファイルがアダプタ用にのみ作成され、処理の後に削除できる場合は、「yes」に設定します。ファイルは、名前が変更され、処理が行われた後に削除されます。「no」に設定すると、ファイルはコピーされ、コピーされたファイルに対して処理が行われます。このファイルに新規レコードが追加された場合、アダプタは次のサイクルでこれらの新規レコードをワーク ファイルにコピーします。このファイルに新規レコードが追加されていない場合、アダプタは、現在のファイルと同じパターンを持ち、(辞書式順序で) 名前が大きい最初のファイルを検索します。アダプタがそのようなファイルを見つけると、そのファイルの処理を開始します。このファイルに新規レコードが追加されても、アダプタは前のファイルに戻りません。ソース ファイルの整合性を維持する必要がある場合は、「no」を使用します。
- **InitialFileName** - アダプタが指定されたパターンでファイルの検索を開始する最初のファイルの名前。**NamePattern** にワイルドカードが含まれていて、アダプタが古いファイルを読み取らないようにする場合は、この属性を使用します。
- **Delimiters** - (オプション)。区切り文字として使用する 1 文字以上の文字。これに従ってデータ行がフィールドに解析されます。指定しない場合、デフォルト値は「¥t」です。
- **IgnoreRedundantDelimiters** - (オプション) [yes/no]。「yes」に設定すると、連続する区切り文字は 1 文字として扱われます。「no」に設定すると、区切り文字の間に空白のフィールドが作成されます。
- **RegexForParser** - (オプション)。前に指定した区切り文字を置換して、データ ソースからフィールドを抽出するために使用する正規表現。以下に例を挙げます。

```
<File
...
  RegexForParser="^(.{8}) (.{6}) (?:[ ])*([0-9]+) "
/>
```

詳細については、正規表現のドキュメントを参照してください。

- **TitleExists** - (オプション) [yes/no]。データ ソース ファイル内の空白ではない最初の行がタイトル行であるかどうかを指定します。タイトルは、データの解析時にアダプタが使用することができます。

- **SleepTime** - データ取得の間隔（秒）（つまり、アダプタがソース データ ファイルからデータを取得する間隔の秒数）を指定します。

- **LogicLineDefinition** - （オプション）

```
<File
```

```
...
```

```
<LogicLineDefinition
```

```
    FirstLine="Job server:.*"
```

```
    NumberOfLines="5"
```

```
</>
```

```
</File
```

データセットが 1 行ではなく複数の行から構成されている場合、以下の属性によって、抽出の開始位置、終了位置、および一組にするデータの行数を定義します。

- **AllFile** - （オプション） [yes/no] 。 「yes」に設定すると、ファイル内のすべてが 1 つのレコード（1 つの論理行）と見なされます。
- **FirstLine** - （オプション） 論理行の最初の行を定義する正規表現。 **LastLine** または **NumberOfLines**（またはその両方）と一緒に指定できます（単独でも指定できます）。
- **LastLine** - （オプション） 論理行の最後の行を定義する正規表現。 **FirstLine** または **NumberOfLines**（またはその両方）と一緒に指定できます（単独でも指定できます）。
- **NumberOfLines** - （オプション） 論理行の行数。 **FirstLine** または **LastLine**（またはその両方）と一緒に指定できます（単独でも指定できます）。
- **MatchCase** - （オプション） [yes/no] 。 正規表現の比較で大文字と小文字を区別するかどうかを定義します。

SQL インターフェース セクション

SQL アダプタは、SQL ステートメントを使用してデータベースからデータを取得するために使用できます。

SQL インターフェースは、以下に説明するように、データベースへの接続、およびデータの取得に使用されるクエリを定義します。

XML 構造 :

```
< DataSourceInterface >
  <ConnectionString ConnectionTimeout="60" QueryTimeout="30">
    <![CDATA[ Driver={Microsoft Access Driver
(*.mdb)};DataBaseeq=d:\Oblicore\database1.mdb; ]]>
  </ConnectionString>
  <QueryCollection>
    <Query QueryName="cases" InputFormat="cases" SleepTime="3600">
      <SelectStatement AutoCompleteQuery="yes">
        select
dateclosed,callid,dateopened,companyname,priority,closedmn,responsemn
        from calls where dateclosed is not NULL
      </SelectStatement>
      <QueryKeyFields>
        <KeyField Name="dateclosed" Sort="asc"/>
        <KeyField Name="callid" Sort="desc"/>
        <SelectInitialValues>
          Select min(dateclosed) , 'min date' from calls
        </SelectInitialValues>
      </QueryKeyFields>
    </Query>
    <Query QueryName="contracts" InputFormat="contracts" SleepTime="3600">
      <ConnectionString>
        <Segment Type="text"
Text=" Driver={Microsoft Excel Driver (*.xls)}; DriverId=790;
DataBaseeq="/>
        <Segment Type="File">
          <File Path="d:\Oblicore " NamePattern="Availabilty_*.XLS"
        </Segment>
        <Segment Type="text" Text=";" />
      </ConnectionString>
      <SelectStatement AutoCompleteQuery="yes">...</SelectStatement>
      <QueryKeyFields>...</QueryKeyFields>
    </Query>
  </QueryCollection>
</DataSourceInterface>
```

- ConnectionString - ソース データベースにアクセスするための接続文字列。

ConnectionString は **DataSourceInterface** エlementおよび（または） **Query** エlementに定義できます。 **DataSourceInterface** Elementの **ConnectionString** 定義はデフォルトであり、**ConnectionString** 定義がないクエリにのみ使用されます。

接続文字列は、1つの文字列で定義することも、セグメントによって定義することもできます。 **ConnectionString** Elementに **Segment** Elementが含まれていない場合は、**ConnectionString** Elementのテキストから接続文字列が取得されます。 1つ以上の **Segment** 要素が含まれる場合は、その **Segment** 要素から接続文字列が連結されます。

セグメントには2つのタイプがあります。 1つ目のタイプはテキストで、接続文字列にそのまま連結されるテキストを含んでいます。 2つ目のタイプはファイルで、ワイルドカードがある（またはない）ファイル名を含んでいます。 ファイルセグメントは、1回のみ指定することができ、読み取りファイルの扱い方を定義するその他の属性が含まれています。

- **ConnectionTimeout** - （オプション）。 接続がオープンされるまでの待機時間（秒単位）を示す正の整数値。 値 0 は、接続がオープンされるまで無制限に待機することを示します。 デフォルト値は 15 秒です。

注: 一部のプロバイダはこの機能をサポートしていません。

- **QueryTimeout** - （オプション）。 クエリの実行が完了するまでの待機時間（秒単位）を示す正の整数値。 値 0 は、実行が完了するまで無制限に待機することを示します。 デフォルト値は 30 秒です。

注: 一部のプロバイダはこの機能をサポートしていません。

- **Segment** - セグメントの属性を指定します。
- **Type** - （オプション） [text、file] 。 セグメントのタイプ。
- **Text** - テキストセグメントのテキスト。
- **File** - ファイルの属性を指定します。
- **Path** - ソース データ ファイルの場所へのパス。
- **NamePattern** - ソース データのファイル名を指定します。 ワイルドカードを使用できます。
- **InitialFileName** - （オプション）。 どのファイルから検索を開始するか、および検索するパターンを示します。
- **UsePath** - （オプション） [yes、no] 。 「yes」に設定すると、ファイルパスが接続文字列に連結されます。
- **UseFileName** - （オプション） [yes/no] 。 「yes」に設定すると、ファイル名が接続文字列に連結されます（Excel ファイルの場合に必要）。
- **UpdateSchemaFile** - （オプション） [yes/no] 。 「yes」に設定すると、アダプタは schema.ini ファイルを現在のファイル名で更新します。

注: アダプタによる `schema.ini` ファイル (テキスト ファイル用のデータベース) の変更を許可する場合のみこの属性を使用します。

- **QueryCollection** - クエリのコレクション (1 つのアダプタに複数のクエリを定義できます)。
 - **PreliminaryCheck** - (オプション) `[yes/no]`。データベースへの接続時に、アダプタがクエリをチェックします。この属性を「no」に設定すると、アダプタは変更しないで実行することによりクエリをチェックします。その後、アダプタは返されたレコードに対して処理を行い、それらを再度読み取りません。「yes」に設定すると、アダプタは `SELECT` ステートメントの各「where」文字列を「where 1=2 and」に置換した後にクエリを実行します。このオプションは、アダプタがレコードに対する実行を開始する前にすべてのクエリを確認し、この最適化によってクエリ時間が大幅に短縮される場合に使用します。注 - 一部のプロバイダはクエリ処理を最適化しません。クエリが正しい場合、実行時間はこの追加を行わない場合にかかる時間と同じです。
 - **ExternalCommand** - (オプション)。アダプタが新しい読み取りサイクルを開始する前に実行される外部コマンド。
 - **Command** - 実行されるバッチ ファイルの名前。
 - **SleepTime** - コマンドを再度実行するまでの待機時間 (秒単位) を示す正の整数値。
- **Query** - クエリの属性を指定します。
 - **QueryName** - クエリの名前。
 - **IsActive** - (オプション) `[yes/no]`。「no」に設定すると、アダプタはこのファイルからレコードを読み取りません。
 - **InputFormat** - このクエリに関連付けられている入力フォーマット。アダプタは `InputFormat` を使用してソース レコードからデータを抽出します。
 - **SleepTime** - アダプタが新しいデータの到着を待機するためにダウンする時間 (秒)。
 - **Critical** - (オプション) `[yes/no]`。「yes」に設定すると、クエリにエラーが見つかった場合、アダプタはただちに停止します。オプション「no」を使用するのは、設定ファイルを変更せずに解決される既知のエラーの場合です。
 - **TransactionMode** - (オプション) `[implicit/explicit]`。「explicit」に設定すると、アダプタはクエリを実行する前にデータベース トランザクションを開始します。このオプションを使用すると、大きくて複雑なクエリでのいくつかの問題が解決されます。
 - **RollbackSegmentName** - (オプション)。アダプタが使用するロールバック セグメントを定義します。定義しない場合は、データベースがロールバック セグメントを選択します。

- **SelectStatement** - ソース データベース上で実行される、選択されたステートメントを格納します。
 - **AutoCompleteQuery** - (オプション) [yes/no]。「yes」に設定すると、アダプタが以下のように自動で **WHERE** ステートメントを指定したクエリに連結します。
 - キー フィールドを基準にして新しい値のみを取得する **WHERE** ステートメントを作成する。
 - キー フィールドを基準にしてステートメントの結果を並べる。
- **QueryKeyFields** - 次のデータ取得を開始するフィールドを定義します。
 - **KeyField** :
 - **Name** - クエリのフィールドから取得されるフィールドの名前。
 - **Sort** - (オプション) [asc/desc]。データの並べ替え方法 (昇順/降順)。
 - **Function** - (オプション)。フィールド値を関数と組み合わせて使用 (:fieldname) するために、特殊な関数でこのフィールドを操作する必要がある場合に、この属性を使用します。たとえば、Oracle の日付関数をフィールド名 Ts と使用する場合は次のように指定します:
Function="to_date(':ts','dd/mm/yyyy')"
 - **ValueLeftWrapper** - (オプション)。フィールドの値の前に文字を連結するには、この属性を使用します。文字列フィールドおよび日付フィールドのデフォルトは「|」(アポストロフィ) です。
 - **ValueRightWrapper** - (オプション)。フィールドの値の後に文字を連結するには、この属性を使用します。文字列フィールドおよび日付フィールドのデフォルトは「|」(アポストロフィ) です。
 - **NameLeftWrapper** - (オプション)。フィールド名の前に文字を連結するには、この属性を使用します。デフォルトは **NULL** 文字列です。
 - **NameRightWrapper** - (オプション)。フィールド名の後に文字を連結するには、この属性を使用します。デフォルトは **NULL** 文字列です。
 - **IsPrimaryKey** - (オプション) [yes/no]。「no」に設定すると、アダプタはクエリの自動 **where** ステートメントでこのキーを使用しません。
 - **DefaultInitialValue** - (オプション)。**SelectInitialValues** クエリがレコードを返さなかった場合、アダプタはここからデフォルト値を取得します。キーフィールドが複数ある場合は、すべてのキーフィールドにこの属性が必要となります。
 - **SelectInitialValues** - 最初の **WHERE** ステートメントの初期値を提供する **SELECT** ステートメント (制御ファイルが空の場合)。

注: 値の順序は、この **QueryKeyFields** のフィールドエレメントと同じにして、各フィールドに結果を返す必要があります。

InputFormatCollection セクション

このセクションでは、データ行をフィールドに分割する方法およびフィールドタイプや形式など、データソースから取得したデータの構造を指定します。初期データフィルタリングおよびデータ操作は、**InputFormatSwitch** フィールドおよび**Compound** フィールドをそれぞれ使用して、このセクションで実行される場合があります。

このセクションの一般的なワークフローを以下に示します。

- データ行が 1 つ以上の **InputFormat** に対して一致します。
- データは **InputFormat** の仕様と一致するフィールドに分割されます。
- データフィールドを結合または分割することにより、複合フィールドに値が割り当てられます。
- 処理されたデータが、**TranslatorSwitch** 条件に対して検証されます。
- 処理されたデータが、一致したトランスレータに送信されるか、または無視されます。

InputFormatCollection ノードには 1 つ以上の **InputFormat** ノードが含まれていることがあります。

XML 構造 :

```
<InputFormatCollection>
  <InputFormat InputFormatName="MyInputFormat">
    <InputFormatFields>
      <InputFormatField Name="sid_id" Type="string"/>
      <InputFormatField Name="content" Type="string"/>
      <InputFormatField Name="date" Type="time"
        TimeFormat="%d/%m/%Y %H:%M:%S"/>
      <InputFormatField Name="server" Type="string"
        Source="compound">
        <Compound>
          <Segment SourceField="content"
            RegularExpression=".*Job server: ([^\n]+).*" />
        </Compound>
      </InputFormatField>
    </InputFormatFields>
  </InputFormat>
  <TranslatorSwitch DefaultTranslator="GeoTranslator">
    <TranslatorCase TranslatorName="NonGeoTranslator" Break="yes">
      <Condition SourceField="routing_info" Operator="EQ"
        Value="cnano"/>
    </TranslatorCase>
  </TranslatorSwitch>
</InputFormatCollection>
```

- **InputFormat :**
 - **InputFormatName** - この形式の名前は **DataSourceInterface** セクションによって参照されます。
- **RequiredFields** - (オプション) データ行で見つかることが予期されるフィールドの最少数。含まれているフィールドがこれより少ない行は無視され、エラーがログ記録されます。
- **InputFormatFields** - **InputFormatFields** には、データ ソース内の入力フィールドの数により 1 つ以上のフィールド ノードが含まれる場合があります。
 - **InputFormatField** - 元のデータ行のデータ フィールドまたは複合フィールドを 1 つ指定します。

```
<InputFormatField Name="timestamp" Type="time"
TimeFormat="%d/%m/%Y %H:%M:%S"/>
```
 - **Name** - このフィールドの名前。他のエレメントからこの名前でも参照されます。
 - **Type** - フィールドのデータ タイプ - **string/integer/real/time**。
 - **Source** - (オプション) デフォルト値は「**event**」です。有効な値は以下のとおりです。
 - **event** - フィールド値はデータ ソースから送られてくるイベントから取得されます。フィールドの値はデータ ソースから送られてくる順序で取得されます。
 - **compound** - フィールドは複合フィールドです。別のフィールド値または定数になんらかの操作を行った後の値です。
 - **title** - フィールド値はタイトル フィールドの名前から取得されます。参照されたフィールドは、定義済みである必要があります。
 - **filename** - フィールド値はデータ ソースのファイル名から取得されます。テキスト ファイル アダプタ専用です。
 - **constant** - フィールド値は定数であり、隣に表示される **ConstantValue** プロパティから取得します。
 - **FieldTitleName** - ソースがタイトルである場合、使用するフィールドのタイトルを指定します。ソース フィールドは事前に定義されている必要があります。
 - **ConstantValue** - ソースが定数である場合に、照合する値を指定します。フィールドのタイプが「**time**」の場合、定数値は **TimeFormat** に基づく形式の文字列（「**Now**」または「**NowUtc**」）になります。「**Now**」はデータ ソース ロケールでの現在の時刻、「**NowUtc**」は UTC での現在の時刻です。
 - **TimeFormat** - フィールドの解析に適用される書式。以下の文字コードを日付および時間形式に使用できます。

記号	説明
%d	1～31 日
%a	週の省略名(日、月..)
%A	週の完全名(日曜日、月曜日..)
%m	1～12 月
%b	月の省略名(1月、2月..)
%B	月の完全名(1月、2月..)
%y	年(2 桁)
%Y	年(4 桁)
%H	24 時間形式の時間(00～23)
%I	12 時間形式の時間(01～12)
%p	12 時間表示の A.M./P.M. インジケータ
%M	0～59 分
%S	0～59 秒
%D	区切り記号なしの完全な日付(yyyymmdd)
%T	秒と区切り記号なしの時間(hhmm)
%n	1970 年からの秒数
%E	16 進数形式の日付 & 時間
%%	パーセント記号

- DecimalSymbol - (オプション) フィールドの小数点記号。指定しない場合は、General セクションの DefaultDecimalSymbol が使用されます。
- DigitGroupingSymbol - (オプション) 整数フィールドと実数フィールドに使用されるデフォルトの桁区切り記号。指定しない場合は、General セクションの DefaultDigitGroupingSymbol が使用されます。
- Compound - source = compound の場合に必要となります。複合フィールドに収集されるフィールドの操作を指定します。

- **Segment** - 作成された複合フィールドに追加するフィールドの操作を指定します。 **SourceField** 属性のみが必須です。
- **SourceField** - 基となるフィールド。参照されたフィールドは、定義済みである必要があります。
- **RegularExpression** - 正規表現を操作します。
- **MatchCase** - (オプション) [yes/no]。正規表現によるマッチングで大文字と小文字を区別するかどうかを定義します。
- **SelectionStart** - テキストの抽出を開始する位置 (0 から始まる)。
- **SelectionLength** - テキストを抽出するサイズ。
- **Prefix** - 操作結果の先頭に付加する文字列。
- **Suffix** - 操作結果の末尾に付加する文字列。
- **XpathExpression** - XPath 式を操作します。

- **InputFormatSwitch** - データ行が複数のフォーマットで送られてくる場合に、フォーマットの基準を指定するために使用します。

注: **InputFormatSwitch** を使用する場合、**InputFormat** ノードの順序が重要です。参照される **InputFormat** はすでに定義されている必要があります。

DefaultInputFormat - 一致する条件がない場合に適用される **InputFormat** の名前を指定します。

- **InputFormatCase** - 適用する **InputFormat** を判断するために、データ行をテストする条件を指定します。
- **InputFormatName** - 基準が一致した場合に使用される **InputFormat**。
- **LogicOperator** - (オプション) [and/or]。
- **and** - すべての基準が一致する必要があります (デフォルト)。
- **or** - 少なくとも 1 つの基準が一致する必要があります。

- **Condition** - フォーマットを判断するためにデータ行でテストされる基準。

SourceField - テストするフィールド。

Operator - テストのタイプ。以下のオプションがあります。

- **EQ** - 等しい
- **NE** - 等しくない
- **GT** - より大きい
- **LT** - より小さい
- **GE** - 以上
- **LE** - 以下
- **MATCH** - 正規表現が一致する必要があります

- **UNMATCH** - 正規表現が一致しない必要があります

ValueType - (オプション) [constant/field/previousValue]

- **constant** - 値属性の内容はソース データに関係なく一定です。
- **field** - 値属性の内容は同じレコードのフィールドの名前です。
- **previousValue** - 値属性の内容は、同じ入力フォーマットの同じクエリの前のレコードのフィールド名です。

Value - マッチングする値、または正規表現。

MatchCase - (オプション) [yes/no]。テストで大文字と小文字を区別するかどうかを定義します。「yes」に設定すると、マッチング対象の2つの値はテストの前に小文字に変換されます。

- **TranslatorSwitch** - データ行を **CA Business Service Insight** 統合イベントに変換するために使用するトランスレータを決定します。
 - **DefaultTranslator** - 一致する条件がない場合に使用されるトランスレータ。値が「Ignore」の場合、トランスレータは使用されず、その行は無視されます。
 - **TranslatorCase** - 適用するトランスレータを判断するために、処理対象のデータでテストされる条件指定します。

Break [yes|no]

yes - (デフォルト) 条件が一致する場合は、次の条件をチェックしません。

no - 常に条件を評価し、一致した場合はトランスレータを適用した後に、次の条件に進みます。

LogicOperator - (オプション) [and/or]。

- **and** - すべての条件が一致する必要があります (デフォルト)。
- **or** - 少なくとも1つの条件が一致する必要があります。

TranslatorName - 条件を満たした場合にリダイレクトされるトランスレータ。

Condition - 使用する適切なトランスレータを判断するために、処理対象のデータでテストされる条件。 **InputFormatSwitch** の条件と同じです。

TranslationTableCollection セクション

このセクションには、データ ソース値から CA Business Service Insight のイベントフィールドへのマッピングテーブルがあります。

XML 構造 :

```
<TranslationTableCollection
  LoadingMode="remote"
  TranslationTablesFileName="Translations.xml">
  <TranslationTable
    Name="ResourcesTranslateTable"
    DestinationType="resource">
    <TranslationField>nodeName</TranslationField>
  </TranslationTable>
  <TranslationTable
    Name="EventTypesTranslateTable"
    DestinationType="event_type">
    <TranslationField>eventType</TranslationField>
  </TranslationTable>
  <TranslationTable
    Name="valueUpDownTranslateTable"
    DestinationType="value"
    ValueType="string">
    <TranslationField>eventType</TranslationField>
  </TranslationTable>
</TranslationTableCollection>
```

- TranslationTablesFileName : (オプション) テーブルをローカルに格納するためのファイル名を指定します。指定しない場合は、デフォルト値が使用されます (「Translation.XML」)。
- LoadingMode : (オプション) [standalone、remote]。

注: オンラインインターフェースのデフォルトは「remote」、オフラインインターフェースは「standalone」です。変換テーブルをロードする方法は以下のとおりです。
- standalone : アダプタは変換テーブルをローカルにロードします。変換する際に CA Business Service Insight サーバに接続しません。変換テーブルへの変更は、ローカルファイル内にのみ格納されます。
- remote : アダプタは CA Business Service Insight サーバからすべてのテーブルをロードするリクエストを送信します。変換テーブルに加えられた変更は、ローカルにも格納されます。
- LoadTimeout : (オプション) ロードのモードが「remote」の場合、タイムアウト (秒単位) をここで指定できます。
 - TranslationTable : イベント値をマッピングテーブルにリンクします。

- **Name** : トランスレータによって使用および参照される名前。適切な名前は文字またはアンダースコアで始まり、文字、数字、およびアンダースコアが含まれています。
- **DestinationType** : [resource、event_type、contract_party、service、time_zone、value]。
- **ValueType** : (オプション) [integer、real、string]。テーブルから返される値のタイプ。文字列値および実数値は、DestinationType="value" のテーブルにのみ指定できます。
- **TranslationField** : 変換元のフィールド名。フィールド名は入力フォーマットのフィールドから取得されます。フィールドは5つまで保持できます。

TranslatorCollection セクション

TranslatorCollection セクションでは、前のセクションで CA Business Service Insight イベントに抽出された解析および操作済みのデータ ソース レコードを変換する方法を記述します。

インターフェース モードが「オンライン」の場合、CA Business Service Insight イベントは、以下のフィールドが含まれる統合型の構造体になります。

- **Timestamp** : イベントが発生した時刻
- **ResourceId** : イベントに関連付けられた CA Business Service Insight のリソース ID (測定されたリソースなど)
- **EventTypeId** : イベントに関連付けられた CA Business Service Insight のイベントタイプ。イベントのタイプを記述します (リソースに関する測定のタイプ、チケットアクションのタイプなど)。
- **DataSourceId** : (オプション) 入力データ ソースの名前 (ファイル名、テーブル名など)
- **Value** - イベントの値 (測定結果、チケット番号など) このフィールドは、複数回表示されることがあります。

インターフェース モードが「オフライン」の場合、フィールド数やフィールド名に関する制限はありません。

XML 構造 :

```
<TranslatorCollection>
  <Translator TranslatorName="events" OnDuplication = "ignore">
    <TranslatorFields>
      <TranslatorField Name="ResourceId" SourceType="table"
        SourceName="ResourcesTranslateTable" Iskey="yes"/>
      <TranslatorField Name="EventTypeId" SourceType="constant"
        ConstantValue="1002" Iskey="yes"/>
      <TranslatorField Name="Timestamp" SourceType="field"
        SourceName="timestamp" Iskey="yes"/>
      <TranslatorField Name="Value" SourceType="table"
        SourceName="valueUpDownTranslateTable" Iskey="yes"/>
      <TranslatorField Name="Value" SourceType="field"
        SourceName="nodeName" Iskey="yes"/>
      <TranslatorField Name="Value" SourceType="constant"
        Type="integer" ConstantValue="1000" Iskey="yes"/>
      <TranslatorField Name="Value" SourceType="field"
        SourceName="timestamp" TimeShift="-3600"
        TimeShiftFieldName="createDate" Iskey="yes"/>
      <TranslatorField Name="Value" SourceType="lookup"
        SourceName="ServiceTable" LookupValue="word"
        Iskey="yes"/>
    </TranslatorFields>
  </Translator>
</TranslatorCollection>
```

```
</Translator>
</TranslatorCollection>
```

- **Translator** : 受信した一連のフィールドを出力イベントに変換する方法を記述します。
 - **TranslatorName** : 一連のフィールドをこのトランスレータに送信する場合に **InputFormat** で使用される名前。
 - **OnDuplication** : 重複イベントをどのように処理するかを決める値（「Ignore」、「Add」、「Update」、または「updateAlways」）が保持されるメンバ（「イベント固有性」を参照）。
 - **TranslatorFields** : **TranslatorField** エレメントのリストが含まれます。各エレメントは、以下の属性で構成されます。
 - **Name** : フィールド名。オンラインインターフェースでは、必ず「Timestamp」、「ResourceId」、「EventTypeId」、「Value」、または「DataSourceId」のいずれかの値になります。
 - **SourceType** :
 - field** : このフィールドの値が入力フォーマットのフィールドから取得されます。 **SourceName** 属性にフィールド名が含まれます。
 - table** : このフィールドの値が変換テーブルから取得されます。 **SourceName** 属性にテーブル名が含まれます。
 - lookup** : このフィールドの値が変換テーブルから取得されます。 **SourceName** 属性にテーブル名が含まれます。変換される値は、入力フォーマットではなく **LookupValue** 属性から取得されます。
 - constant** : このフィールドの値が定数になり、**ConstantValue** 属性に含まれます。
 - **SourceName** : フィールド名または変換テーブル名が含まれます。
 - **Type** : [integer/real/string/time] フィールドのタイプが（フィールド名または **SourceType** で）事前定義されていない場合にのみ必須です。オンラインインターフェースでは、**SourceType=constant** の **Value** フィールドでのみ必要となります。オフラインインターフェースでは、**SourceType=constant** の各フィールドで必要となります。
 - **IsKey** : イベントに一意のキーを表します。このキーは、**TranslatorFields?IsKey = "yes"** と表示されたフィールドから構築されます。（「イベント固有性」を参照）
 - **LookupValue** : **SourceType="lookup"** を設定している場合に検索値を格納します。

- **ConstantValue** : **SourceType=constant** を設定している場合に定数値を格納します。フィールドのタイプが「time」の場合、定数値は **TimeFormat** に基づく形式の文字列（「Now」または「NowUtc」）になります。「Now」はデータ ソース ロケールでの現在の時刻、「NowUtc」は UTC での現在の時刻です。
- **TimeFormat** : 時間形式が含まれます。**SourceType=constant** および **Type=time** のフィールドでのみ必要となります。
- **TimeShift** : 時間シフト（秒単位）を定義します（時間フィールドにのみ使用される）。
- **TimeShiftFieldName** : （オプション）時間シフト（秒単位）を含むフィールド名が入力フォーマットから取得されます。**TimeShift** と **TimeShiftFieldName** は併用可能です。

付録 D: ビジネス ロジック計算式の定義(ビジネス ロジック エキスパート)

このセクションには、以下のトピックが含まれています。

[ビジネス ロジック計算式の作成時に避けるべき事項 \(P. 367\)](#)
[クラスタ化メトリックおよびリソースの有効性 \(P. 367\)](#)

ビジネス ロジック計算式の作成時に避けるべき事項

ビジネス ロジック計算式を作成する際に、必ず以下の事項を念頭に置いてください。

- 絶対に **NULL** 値をグローバル変数に割り当てないでください。 **NULL** 値を割り当てると、メトリックの計算時に **PSLWriter** が失敗する場合があります。 初期化されていない値が必要な場合は、**Empty** を代わりに使用します。
- クラスタ化メトリック内でマップ オブジェクトおよびベクトル オブジェクトを使用しないでください。 これらのオブジェクトを使用する必要がある場合、できる限り小さくしておきます。 大きなマップおよびベクトルを含むクラスタ化メトリックは、計算に長い時間がかかります。

クラスタ化メトリックおよびリソースの有効性

クラスタ化メトリックについて

クラスタ化メトリックでは、リソース グループの個々のメンバに対して 1 つのメトリックの定義を実行して、一連のアイテムに同じ定義およびロジックを適用することができます。クラスタ化は、事前に定義されたリソース セットに静的に設定することもできますが、グループを長期にわたって変更し、グループに対してメンバを対象、または対象外にしたりして、リソース グループに動的に設定することもできます。

リソースまたはリソース グループは、時間の経過に従ってグループから除外したり、参加させることができます。同じ計算期間（日、月、年など）内であっても、グループから除外して、またそのグループに戻すことを何度も繰り返すことができます。

クラスタ アイテムをクラスタ ベース グループから削除すると、ビジネス ロジックにどのように影響しますか?

OnPeriodEnd メソッドおよび Result 関数が、クラスタ アイテムに対してトリガされます。これが計算期間中に発生した場合、結果は、元の計算期間が終了したときにのみ（たとえば、月末、年末）データベースに書き込まれます。

クラスタ アイテムをクラスタ ベース グループに参加させると、ビジネス ロジックにどのように影響しますか?

グローバル変数が作成され、OnLoad、OnRegistration、および OnPeriodStart メソッドがクラスタ アイテムに対してトリガされます。

同じ計算期間内に、クラスタ アイテムをクラスタ ベース グループから削除し、その後、クラスタ ベース グループに参加させた場合、ビジネス ロジックにどのように影響しますか?

クラスタ アイテムがグループの一部であった期間に設定された結果が、新しい結果に上書きされます。言い換えると、計算期間が終わった後の結果には、クラスタ アイテムがグループの一部であったときに計算された期間のうちの最後の期間のみが反映されます。

ビジネス ロジックのリソースを有効にする属性にはどのような効果がありますか?

リソースが有効ではない期間には、有効ではないリソースの Raw データは収集されません。

クラスタ化のリソースを有効にする属性にはどのような効果がありますか?

リソースが無効になるように変更すると、グループからリソースを除外したときと同じような影響がクラスタ化に及びます。クラスタ化は、リソースの有効性およびグループ メンバシップに対して同様に動作します。

リソースに例外を実装するにはどうすればよいですか? リソースの有効性を使用するのは正しい方法ですか?

例外期間を個別のリソースに設定する必要があるビジネス ケースがあります。たとえば、サーバがメンテナンスに入ったために、そのメンテナンス期間を計算から除外する必要がある場合です。

ビジネス ロジックは有効ではないリソースの Raw データ イベントを無視するため、有効性のメカニズムを使用してリソースに例外を実装することを選択できます。これはいくつかのユーザ ケースで適用できる場合があります。ただし、リソースがクラスタ化メトリックの一部で、リソースが有効になる予定で、また同じ計算時期内に無効になる予定の場合は、前述のようにリソースが有効だった最後の期間のみが結果内で考慮されます。この場合は、カスタム属性機能を使用することをお勧めします。リソースのステータスを保持する追加の属性を管理することができます。その後、ビジネス ロジック計算式で、スクリプト内の適切な場所から、リソースのステータスを問い合わせます。

用語集

KPI

キー パフォーマンス インジケータ。ビジネスまたはサービスによって定量化可能な目標がどの程度達成されているかを監視するために、単独または他のキー パフォーマンス インジケータと組み合わせて使用する重要な指標。

KQI

キー クオリティ インジケータ。ビジネスまたはサービスによって品質目標がどの程度達成されているかを監視するために、単独または他のキー パフォーマンス インジケータと組み合わせて使用する重要な指標。

OLA

オペレーショナル レベル アグリーメント。OLA は、サプライヤが内部組織で、顧客が内部ビジネス ユニットである SLA です。

Raw データ イベント

CA Business Service Insight で Raw データによって生成されたイベント。

SLO

サービス レベル目標。契約で合意されている項目の測定に使用します。

アーカイブ済み契約

データがすでにアーカイブに移動されている契約。アーカイブ済み契約は計算の対象外であり、レポートには表示できません。

アダプタ

CA Business Service Insight とサードパーティ データ ソースの間のインターフェース。アダプタによって、サードパーティ データ ソースから取得したデータが、契約関係者に提示するサービス レベル計算で使用できる形式に変換されます。

アラート

システムで発生するイベントに関するユーザへの通知。アラートによって、ユーザは、契約違反やペナルティなどを避けるための是正処置を実行できます。

アラート デバイス

電子メール、ポップアップまたは SMS、またはサードパーティ プログラムを介してアラート メッセージを受信するネットワーク内デバイス。

アラート プロファイル

アラート メッセージのトリガ条件、通知先ユーザ、および送信方法が定義されるパラメータ セット。

イベント タイプ

イベントは、サードパーティ測定ツールによってリソースに対して実行され、アダプタによって **CA Business Service Insight** で使用可能な形式に変換される測定値です。 イベントタイプによって、**CA Business Service Insight** に対するイベントの定義方法および形式設定方法が決まります。

イベント注釈

特定のイベントに関する追加情報。 イベント注釈は、自動または手動で（レポート内で）設定されます。

請負契約

請負契約。 組織のサービス デリバ리를サポートする外部サプライヤのサービス デリバ리に関する契約。

エージェント

指定された時間単位のメトリックを表すオブジェクト。

価格アイテム

価格を計算するメトリック。 価格は、消費に基づいて決めるか、または定額として定義できます。

簡易レポート

CA Business Service Insight によって計算された結果を豊富な基準に基づいて分析する手段。

関係

2つのメトリックの間で確立されるある種のつながりを示すエンティティで、いくつかのプロパティが存在します。

監査証拠

CA Business Service Insight のユーザおよびシステムによるアクティビティの時系列の記録。 システムに保存されます。 監査証拠を後で確認することによって、システム上で発生したシステムまたはプロセスに対するユーザ アクションを確認できます。

クラスタ化メトリック

複数のリソースまたはリソース グループに適用されるタイプのメトリック。

グループ レポート

複数のレポートが 1 ページに並べられているレポート。

契約関係者

契約を締結する相手である顧客またはプロバイダ。 大規模組織内の内部エンティティが契約関係者である場合もあります。

契約関係者グループ

論理的に定義された契約関係者（顧客）のグループ。個別の契約関係者の代わりにグループを参照することで、効率的に契約を作成できます。契約関係者は複数の契約関係者グループに属することができます。

契約監査証跡

契約のすべてのアクティビティのログ。

契約作成者

特定の契約の作成を担当するユーザ。

契約テンプレート

新規契約の作成に使用する事前定義済み契約。

契約レベルパラメータ

ビジネス ロジック内のメトリックによって使用されている、標準契約、契約テンプレート、およびサービス レベルテンプレートのパラメータ。

現行ステータス エンジン

現在のステータス メトリックを計算するスタンドアロンプロセス。1つ以上のマシン上で実行できるインスタンスの数に制限はありません。

サービス カタログ

CA Business Service Insight のサービス、ドメインと単位、テンプレート ライブラリ、およびサービス レベルテンプレートに関するすべての情報の集合。

サービス レベルテンプレート

サービス定義は、サービスとメトリックの事前定義済みセットです。共通のビジネス プロセスに基づく契約の作成および変更を容易にするために使用されます。

自由形式レポート

CA Business Service Insight データベースまたは他の外部データベースに基づくユーザ定義 SQL ステートメントによって作成されるレポート。

受信したイベント

他のメトリックから受信したイベントのリスト。[受信したイベント] タブでクリックすると、[ビジネス ロジック スコープ] ウィンドウに表示されます。

手動根本原因コメント

特定のメトリックのサービス レベルの結果について説明または情報を追加するコメント。レポート ビューアで手動で設定します。手動根本原因コメントは、同じメトリックのすべてのレポート ビューアによって共有できます。

消費メトリック

レポートで消費と価格を別々に表示できるメトリック。

情報メトリック

レポート機能専用の情報を計算するメトリック。

測定単位

ドメイン カテゴリに定義されているすべてのメトリックの測定単位。

ダッシュボード

リアルタイム情報とレポートをまとめて読みやすい形式で表示する上位レベルの管理者向けのユーザ インターフェース。

ダッシュボードマップ

ダッシュボードのレイアウト。ダッシュボード内のウィジェットの配置。

単位

測定可能な単位のカatalog。たとえば、パーセンテージや通貨があります。

地域オプション

契約関係者の地域の詳細。パラメータには、言語、通貨、GMT との時差、夏時間、日付表示形式が含まれます。

中間メトリック

イベントを計算するためのイベントを生成できるメトリック。中間メトリックを計算することはできません。

粒度

粒度は、メトリックの作成者が、メトリックのトラッキング期間以外に、結果取得を希望する時間単位を決定します。

データ イベント

CA Business Service Insight アダプタによって生成されるイベント。

ドメイン カテゴリ

契約で合意されているサービス レベルの特定の側面。たとえば、ヘルプ デスクと呼ばれるサービス ドメインに対して、ヘルプ デスク サービスの特定の側面を表すドメイン カテゴリとしてヘルプ デスクの応答時間などが存在します。通常は、サービス プロバイダのシステム管理者がドメイン カテゴリを定義します。CA Business Service Insight も事前定義済みドメイン カテゴリを提供します。

トラッキング期間

契約で定義されているサービス レベル目標に対して、提供されるサービス レベルを CA Business Service Insight で測定する期間。この期間中に測定された値によって、ビジネス ロジックで定義されている目標が達成されているかどうか、およびペナルティまたはボーナスが発生しているかどうかが判断されます。また、トラッキング期間によって、ビジネス レポートの生成方法が決まります。

パッケージ

インストールおよび別の CA Business Service Insight インスタンスでの解凍のためにまとめて圧縮されているサービス レベル テンプレートとテンプレートの集合。

パラメータ

ビジネス ロジックの外部で設定し、ビジネス ロジック 計算式に作用させることができる値。ビジネス ロジックは、パラメータを使用して、サービス レベル結果を決定します。パラメータのタイプとして、テキスト、リスト、数、日付、またはテーブルを使用できます。パラメータの例として、しきい値やリソース名があります。

ビジネス ロジック テンプレート

新しいビジネス ロジック 計算式を定義するために使用できる事前定義済みビジネス ロジック 計算式。

ビジネス ロジック モジュール

ビジネス ロジックで使用できるスクリプト関数のライブラリ。

比率

単位のタイプ。

複合レポート

1 つのグラフ上に複数のシリーズとして表示される、複数の標準レポート

ブックレット

事前定義済みの設定可能なデザイン テンプレートを使用した便利な形式の契約データおよび関連レポートを含む、RTF ベースのファイル

ペナルティ

サービス レベル目標値が定義されたトラッキング期間内に満たされない場合に契約関係者に支払う義務が発生する補償。CA Business Service Insight 内でペナルティを有効または無効にできます。

変換

アダプタによってデータ ソースから収集されたデータを、CA Business Service Insight で定義されているエンティティに変換します。

変換スクリプト

変換を簡単にメンテナンスしたり、エラーを防いだりするために使用できるスクリプト。

変換テーブル

Raw データから取得したデータの値を、システムで定義されているデータに変換するためのメカニズム。

変更セット

サービス プロバイダのリソース マップに対する一連の変更。

メトリック

特定の時点の特定のサービスのターゲット サービス レベルを定義する複数のパラメータの組み合わせ。メトリックはそれぞれ 1 つのサービス ドメインに関連付けられます。メトリックのフィールドと属性には、ドメイン カテゴリ、サービス レベル計算式、サービス、タイムスロット、サービス レベル目標値、およびトラッキング期間が含まれます。

メトリック イベント

CA Business Service Insight でメトリックによって生成されたイベント。

メトリック ウィザード

サービス レベル テンプレートのメトリックを最初に契約に追加するときに、ユーザが容易にメトリックを変更できるように開発されたユーザ フレンドリなインターフェース。

メトリック タイプ

特定のメトリックの計算根拠を説明し、フィールドのリストを定義します。このリストでは、必須フィールドおよび特定のタイプのメトリックに必要なデフォルトに関するフィールドの動作が示されます。

メトリック レベル パラメータ

メトリックのパラメータ。

目標ステートメント

メトリックを定義するパラメータが含まれるメトリックの論理的な説明。目標ステートメントは、メトリックの本質を効果的に把握するために、CA Business Service Insight GUI に表示されます。

役割

責任、アクティビティ、および許可のセット。ユーザの役割によって、CA Business Service Insight のビュー モードが定義されます。すべての CA Business Service Insight ユーザは 1 つ以上の役割が割り当てられ、割り当てられた役割によってユーザが CA Business Service Insight 内で実行できるアクションが決まります。ユーザがアプリケーションにアクセスするとき、役割で実行が許可されていないアクションは、CA Business Service Insight ユーザ インターフェースから削除されます。

リソース

サービス プロバイダの総合的インフラストラクチャの 1 つの要素。リソースの例として、個別のサーバ、スイッチ、ハブ、ルータ、ヘルプ デスク、または他の測定可能な要素があります。複数のリソースをリソース グループの一部として定義できます。リソース グループは、それ自身が 1 つのリソースとして扱われます。

リソース グループ

論理単位としてグループ化されたリソースの集合。1つ以上の個々のリソースまたはリソース グループを、1つのリソース グループとしてグループ化できます。

リソース タイプ

あらかじめ用意されているリソースのカテゴリ。リソースは少なくとも1つのリソース タイプに割り当てる必要があります。

リソース入力日

リソースが CA Business Service Insight に入力された日付。リソース発効日と混同しないでください。

リソース配置

リソースをサービスに割り当てると、リソースのイベント ストリームがそのサービスに転送されます。リソースは、サービス、契約関係者、タイプ、またはグループに割り当てることができます。

リソース発効日

CA Business Service Insight がリソースを使用できる開始日。リソース発効日は、リソースが含まれるリソース バージョンによって定義されます。リソースは、それが含まれるリソース バージョンの発効日より古い発効日のリソース バージョンによって認識されることはありません。

例外

サービス レベルの計算に算入されない期間。たとえば、ダウンタイムです。

レポート ウィザード

レポート パラメータの定義に使用するグラフィカル ユーザ インターフェース (GUI)。

レポート グループ レポート

複数の標準レポートをカプセル化して並べたレポート

レポート フォルダ

レポート フォルダは、何らかの関連があるレポートのグループ化に使用されるコンテナです。

関連メトリック

関係でターゲットとして参照されるメトリック。

根本原因コメント

サービス レベルの結果について説明するためにビジネス ロジック内で設定されたコメント。根本原因コメントは、メトリックに関連付けられます。

消費

消費を計算するメトリック タイプ。主に会計モジュールで使用します。このメトリック タイプを使用すると、レポートで消費と価格を別々に表示できます。消費は、価格アイテム メトリックでも同様に計算できます。

派生メトリック

契約テンプレートまたはサービス レベルテンプレートに基づいて作成されたメトリック。

変換エントリ

変換テーブルによって使用されるソース値とデスティネーション値の定義を表します。