DevTest Solutions

CA Continuous Application Insight の使用

バージョン 8.0

ca technologies

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: CA Continuous Application Insight の概要

チュートリアルを表示するには、「*はじめに*」の「CA Continuous Application Insight チュートリアル」を参照してください。

このセクションには、以下のトピックが含まれています。

CA Continuous Application Insight の概要

CA Continuous Application Insight （CAI）は、市場にアプリケーションを投入するプロセスを強化するように設計されています。

以下の役割を持つユーザが、CAI の対象ユーザです。

- アーキテクト
- 開発者
- 品質保証エンジニア
- データ アーキテクト
- リリース エンジニア

CAI は、アプリケーションを通過するデータをキャプチャします。CAI は、このデータを使用して、Web ベース DevTest ポータルに表示できるパスを生成します。 DevTest ポータルから、以下のタスクを実行できます。

- ビジネス トランザクションのドリルダウンと異常な動作の分析
- 機能横断チーム連携のためのアーキテクチャ図の生成
- レグレッション テストのためのベースライン テスト ケースの生成
- 仮想サービスの生成
- 不良コンポーネントの分離と不具合を再現するためのテスト ケースおよび仮想サービスの生成

アプリケーションのユーザ インターフェースから不具合チケットを作成することもできます。 このチケットは DevTest ポータルで管理できます。

CA Continuous Application Insight の前提条件

CAI 固有のシステム要件はありません。 DevTest サーバ の要件を確認してください。

システム要件の詳細については、「*インストール*」を参照してください。

以下の方法を使用して、トランザクションを CA Continuous Application Insight データベースに追加することができます。

- Java ベース アプリケーション用の DevTest Java エージェントをインストールして設定します。
- CAI Agent Light を実行します。
- ネイティブ MQ CAI を実行します。

DevTest Java エージェントの詳細（ブローカの役割を含む）については、「*エージェント*」を参照してください。

LISA Bank アプリケーションでは、DevTest Java エージェントはデフォルトでインストールされます。

トランザクションをデータベースに追加すると、DevTest ポータルでそれらを表示および分析できます。

CA Continuous Application Insight でサポートされているプラットフォーム

CA Continuous Application Insight （CAI）では、以下のプラットフォームがサポートされています。

- IBM WebSphere Application Server 7.0
- IBM WebSphere Application Server 8.5
- JBoss 4.2
- JBoss 7
- Jetty 8
- Jetty 9.*x*
- Oracle WebLogic Server 10.3
- Oracle WebLogic Server 12.1.1
- TIBCO BusinessWorks 5.*x*
- TIBCO Enterprise Message Service 6.3
- webMethods Integration Server 9.0
- webMethods Integration Server 9.5

JDBC 仮想化 (P. 184)では、以下の JDBC ドライバおよびデータベースがサポートされています。

- Apache Derby
  - Apache Derby データベースに対応する Apache Derby 4.2.3 JDBC ドライバ
- Oracle
  - Oracle 11*g* データベースに対応する Oracle 11*g* JDBC ドライバ
  - Oracle 12*c* データベースに対応する Oracle 11*g* JDBC ドライバ
- IBM DB2
  - IBM DB2 9.5 データベースに対応する IBM DB2 9.5 JDBC ドライバ
  - IBM DB2 9.8 データベースに対応する IBM DB2 9.5 JDBC ドライバ
  - IBM DB2 10.5 データベースに対応する IBM DB2 9.5 JDBC ドライバ
- Microsoft SQL Server

- Microsoft SQL Server 2008 R2 データベースに対応する Microsoft JDBC ドライバ 4.0
- Microsoft SQL Server 2012 データベースに対応する Microsoft JDBC ドライバ 4.0
- Microsoft SQL Server 2008 R2 データベースに対応する jTDS 1.3.1 JDBC ドライバ
- Microsoft SQL Server 2012 データベースに対応する jTDS 1.3.1 JDBC ドライバ

DevTest ポータルを開く

Web ブラウザから DevTest ポータルを開きます。

注: 実行している必要のあるサーバ コンポーネントについては、「*インストール*」の「DevTest プロセスまたはサービスの起動」を参照してください。

次の手順に従ってください:

1. 以下のアクションのいずれかを完了します。
   - Web ブラウザに「**http://localhost:1507/devtest**」と入力します。レジストリがリモート コンピュータ上にある場合は、**localhost** をそのコンピュータの名前または IP アドレスに置き換えます。
   - DevTest ワークステーション から［表示］-［DevTest ポータル］を選択します。
2. ユーザ名とパスワードを入力します。
3. ［ログイン］をクリックします。

ビジネス トランザクションおよびトランザクション フレーム

ビジネス トランザクションおよびトランザクション フレームは CA Continuous Application Insight における重要な概念です。 これらの用語は、ドキュメント内で、トランザクション、パス、トランザクション フレーム、およびフレームと呼ばれる場合もあります。

*ビジネス トランザクション*は、ユーザの要求がアプリケーションによって処理される方法の表現です。 各トランザクションには、クライアント要求の結果として 1 つ以上のサーバが実行されるコード パスが含まれています。

*トランザクション フレーム*は、DevTest Java エージェントまたは CAI Agent Light がインターセプトしたメソッド コールに関するデータをカプセル化します。 このデータには、以下のような情報が含まれます。

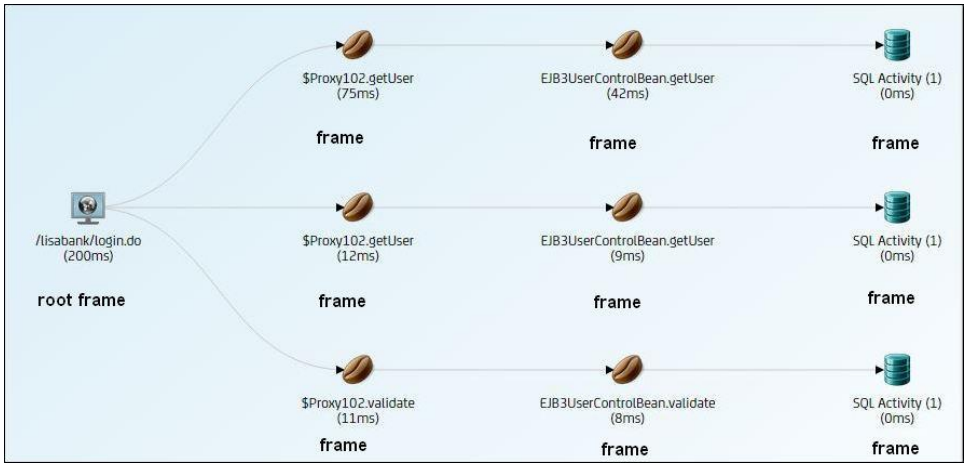
- メソッドの名前
- メソッドが属するクラスの名前
- メソッドに渡された引数
- メソッドが返した値
- 書き込まれている Log4j メッセージ
- Java クラスによってスローされる例外

トランザクション フレームにはそれぞれ一意の識別子およびカテゴリがあります。 通常、カテゴリはプロトコルまたはオペレーティング環境を表します。 例として、EJB、JDBC、JMS、REST、SOAP、WebSphere MQ などがあります。 トランザクション フレームは、トランザクション パス内のノードです。 各ノードは、ビジネス トランザクション全体および実行されるコードまたはビジネス ロジックの 1 つのステップを表します。 一連のトランザクション フレームは、ビジネス トランザクションを処理するために実行されるコードの順序を示します。

トランザクションにはそれぞれ、トランザクション フレームの階層セットが含まれます。 階層の最上位のフレームは*ルート トランザクション フレーム*と呼ばれます。 トランザクションにはそれぞれ一意の識別子があります。

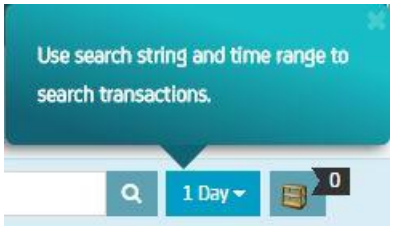
トランザクションはパスとも呼ばれます。*トランザクション パス*は、ユーザの要求が処理される方法のフローを示す視覚的な表現です。 たとえば、パスにより、トランザクションがアクセスするすべてのフロントエンドおよびバックエンド サービスや、それらのサービス間の関係が明らかになります。

以下の図は、DevTest ポータルのトランザクション パスを示しています。



ヘルプ ヒントの有効化および無効化

CAI では、トランザクションの検索やアーティファクトの作成などの主要タスクを実行する方法のヒントが提供されます。 以下の図は、検索条件の入力のヒントを示しています。



ヘルプ ヒントは、有効または無効にすることができます。

次の手順に従ってください:

1. DevTest ポータルを開きます。
2. ［管理者］-［環境設定］を選択します。

   ［設定］ダイアログ ボックスが表示されます。
3. ヘルプを有効にするには、［Enable help for first time user］をオンにします。
4. ヘルプを無効にするには、［Enable help for first time user］をオフにします。
5. ［OK］をクリックします。

CA Continuous Application Insight の無効化

CA Continuous Application Insight を無効にするには、**local.properties** ファイルに以下の行を追加します。

```
lisa.pathfinder.on=false
```

次の手順に従ってください:

1. **local.properties** ファイルを開きます。
2. **lisa.pathfinder.on** プロパティを追加し、値を false に設定します。
3. **local.properties** ファイルを保存します。

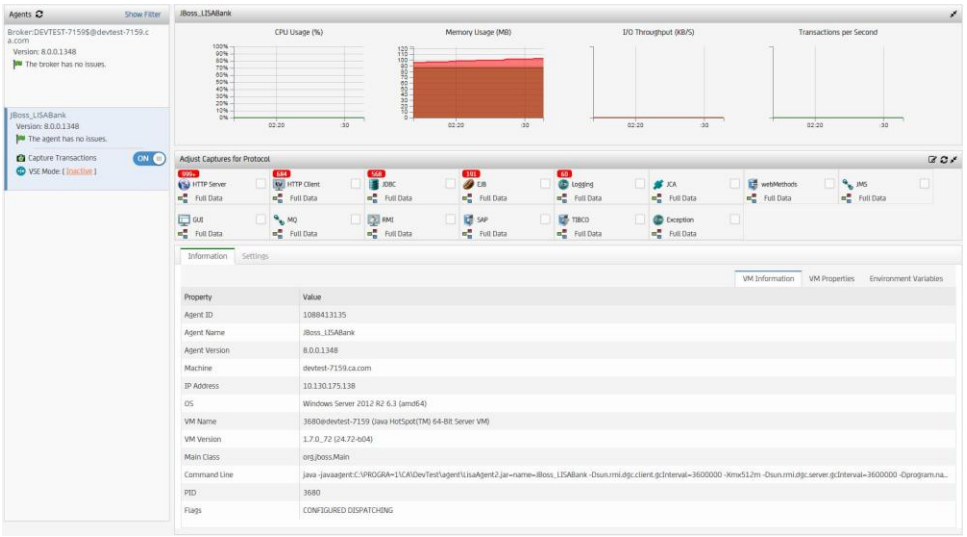
第 2 章: エージェントの設定

［エージェント］ウィンドウを使用すると、以下のタスクを実行できます。

- エージェント ステータスの表示 (P. 22)
- VSE ステータスの表示 (P. 23)
- エージェントのフィルタ (P. 24)
- パフォーマンス データの表示 (P. 26)
- エージェントのキャプチャ レベルの設定 (P. 27)
- エージェントおよびブローカ情報の表示 (P. 29)
- エージェント キャプチャ データの停止および開始 (P. 30)
- 設定プロパティの表示および更新 (P. 30)
- エージェントのアップグレード (P. 31)
- オフライン エージェントの削除 (P. 32)

DevTest ポータルの左側のナビゲーション メニューから［Settings］-［Agenats］を選択することにより、［Agenats］ウィンドウを表示します。

以下の図は、［Agenats］ウィンドウを示しています。



［Agenats］ウィンドウには、以下のコンポーネントが含まれています。

- ［Agenats］ペイン

このペインには、ブローカおよび 1 つ以上のエージェントが含まれています。 ブローカが最上部に表示されます。 エージェントまたはブローカを選択すると、［Agenats］ウィンドウの右側のペインに詳細情報が表示されます。このペインから、エージェントを削除したり、エージェントを最新バージョンにアップグレードすることができます。

- ［Performance］ペイン

  パフォーマンスの詳細が、上部の右側のペインに表示されます。［Agenats］ペインで選択されているブローカまたはエージェントにより、ペインのタイトル バーに表示される名前が決まります。

- ［Adjust Captures for Protocols］ペイン

  このペインには、データをキャプチャしているエージェントのプロトコルが含まれています。［Agenats］ペインでブローカが選択されている場合、このペインは使用できません。

- ［Information］タブ

  このタブには、ブローカまたはエージェントに関する詳細情報が含まれています。

- ［Settings］タブ

  このタブには、ブローカまたはエージェントの設定プロパティが含まれています。

このセクションには、以下のトピックが含まれています。

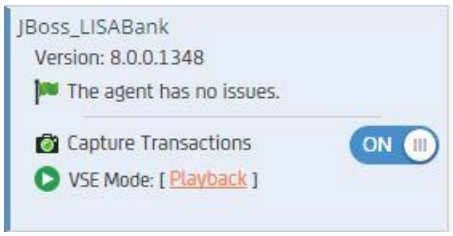
エージェント ステータス インジケータ

［エージェント］ウィンドウの左側のペインでは、各エージェントに、ステータス情報を示すフラグが付いています。

ステータスには、以下の種類があります。

- **緑色：** エージェントに問題はありません。
- **黄色：** スラッシングまたはフラッディングの例外。
- **オレンジ色：** CPU またはガベージ コレクションのスラッシング。
- **オレンジと赤で点滅：** ヒープまたは perm gen による消費。
- **赤色：** JVM アラーム状態またはデッドロック。
- **灰色：** エージェントはオフラインです。

以下の図は、問題のないステータスのエージェントを示しています。



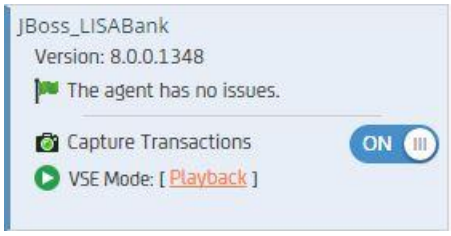
VSE モード インジケータ

［エージェント］ウィンドウの左側のペインでは、各エージェントに、VSE 機能のステータスを示すインジケータが付いています。

ステータスには、以下の種類があります。

- **白いドットがある赤色：** エージェントは記録モードです。
- **白い矢印がある緑色：** エージェントは再生モードです。
- **青色：** エージェントは検証モードです。
- **灰色：** エージェントはパススルー モードです。

以下の図は、再生モードのエージェントを示しています。



状態を変更するには、［VSE Mode］ の横にあるの赤色のテキストの状態をクリックし、メニューからオプションを選択します。

エージェントのフィルタ

［Agenats］ウィンドウでは、左側のペインに表示されるエージェントのリストを絞り込むことにより、エージェントを検索できます。

名前によってエージェントをソートおよびフィルタしたり、検索テキスト条件を入力することができます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Settings］-［Agenats］を選択します。

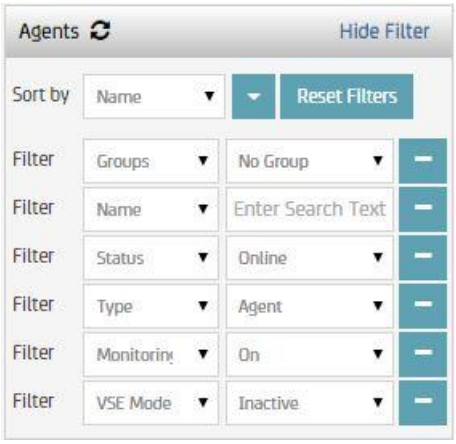
   ［Agenats］ウィンドウが開き、ウィンドウの左側に［Agenats］ペインが表示されます。 デフォルトでは、フィルタは非表示です。

2. ［Show Filter］をクリックします。
3. 名前でエージェントをソートするには、［Sort by］-［Name］リストからオプションを選択します。

   デフォルトのタイプは「Name」です。

4. フィルタ タイプを変更するには、矢印をクリックして、オプションのリストから選択します。
5. 名前でフィルタするには、［Filter］-［Name］リストからオプションを選択します。
6. 左側のペインで、上方向および下方向の矢印を使用して、エージェントを昇順または降順でリスト表示します。
7. 名前でエージェントを検索するには、［Enter Search Text］フィールドに検索条件を入力します。
8. （オプション）フィルタを追加するには、［+］をクリックします。

   追加のフィルタが表示されます。



9. フィルタを削除するには、［-］をクリックします。
10. フィルタを非表示にするには、［Hide Filter］をクリックします。
11. フィルタをリセットするには、［Reset Filters］をクリックします。

パフォーマンス データの表示

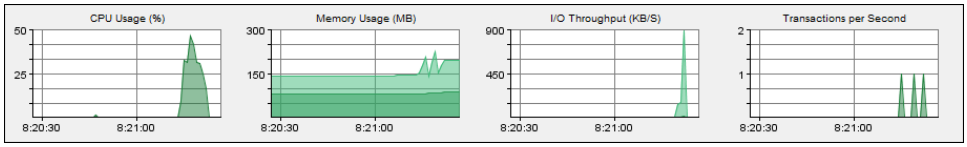
エージェントまたはブローカのパフォーマンス グラフのセットを表示できます。

グラフには、以下の情報が表示されます。

- CPU 使用状況
- メモリ使用率
- I/O スループット
- 1 秒あたりのトランザクション数

I/O スループット グラフは、ブローカには適用できません。

以下の図は、エージェントのパフォーマンス グラフを示しています。



次の手順に従ってください:

1. 左側のナビゲーション メニューで［設定］-［エージェント］を選択します。

   ［エージェント］ウィンドウが表示されます。

2. ［エージェント］ペインで、エージェントまたはブローカを選択します。

   右側のペインにパフォーマンス データが表示されます。

3. （オプション）パフォーマンス グラフをクリックして凡例を表示します。

キャプチャ レベルの設定

DevTest Java エージェントがキャプチャできるプロトコルにはそれぞれ *キャプチャ レベル*があります。

以下のキャプチャ レベルが使用可能です。

カウント

エージェントはカウント情報のみを収集します。 このレベルはテスト中のシステムへの影響が最小です。

カウント数およびパス

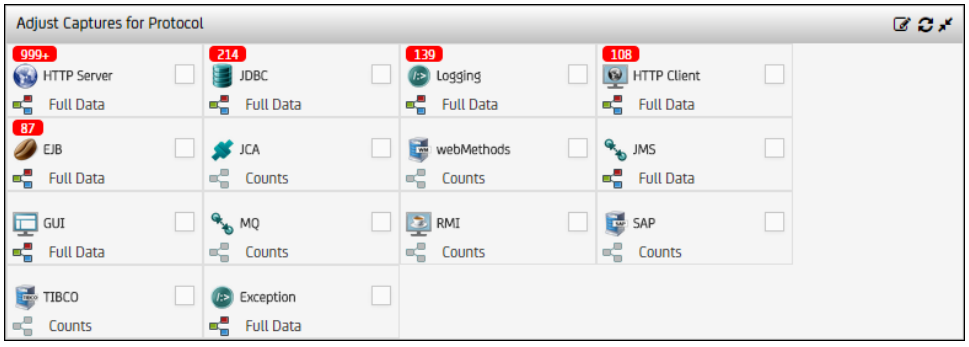
エージェントはトランザクション フレームを収集しますが、要求または応答をすべて収集しません。

全データ

エージェントはデータをすべて収集します。 このレベルはテスト中のシステムへの影響が最大です。

デフォルトのキャプチャ レベルは、［カウント］です。

以下の図は、プロトコルとそのキャプチャ レベルがリスト表示される［Adjust Captures for Protocol］を示しています。



以下のワークフローをお勧めします。

1. カウント モードですべてのプロトコルから開始します。
2. アプリケーションを実行し、カウント数を表示します。
3. 目的のプロトコルを決定し、キャプチャ レベルをそれに応じて上げます。

チケット (P. 69)を作成する場合は、HTTP サーバ プロトコルのキャプチャ レベルが［全データ］に設定されていることを確認します。

ベースライン (P. 93)または仮想サービス (P. 149)を作成する場合は、適切なプロトコルのキャプチャ レベルが［全データ］に設定されていることを確認します。

情報は 1 分間に 1 回更新されます。

赤いバッジには、エージェントが起動されてから、エージェントがプロトコルを確認した回数が表示されます。

カウントが 0 である場合、赤いバッジは表示されません。

カウントが 1000 を超えている場合、赤いバッジにはテキスト **999+** が表示されます。 アイコンのツールヒントに正確な数が表示されます。

注: WebSphere MQ、JMS など、キューベースのクライアント/サーバ通信では、異なるキャプチャ レベルの設定はサポートされていません。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［設定］-［エージェント］を選択します。

   ［エージェント］ウィンドウが表示されます。
2. ［エージェント］ペインから、エージェントを選択します。
3. 1 つ以上のプロトコルのチェック ボックスをオンにします。 複数のプロトコルを選択すると、それらのプロトコルを一度に同じキャプチャ レベルに変更できます。
4. すべてのプロトコルを選択するには、 をクリックし、［すべて選択］を選択します。
5. 現在のキャプチャ レベルをクリックし、ドロップダウン リストからオプションを選択します。
6. すべてのプロトコルの表示とカウントが 0 を超えているプロトコルのみの表示を切り替えるには、 をクリックし、［Show Protocols with Counts］を選択します。
7. カウントをすべて 0 にリセットするには、 をクリックし、［Reset All Counts］を選択します。

8. カウントおよびキャプチャ レベルをリフレッシュするには、  をクリックし、［Enable Auto fresh］を選択します。

エージェントおよびブローカ情報の表示

エージェントまたはブローカに関する以下のカテゴリの情報を表示できます。

- VM 情報

  ［VM 情報］タブには、エージェントおよびエージェントが実行されている仮想マシンに関する基本情報が表示されます。 値は読み取り専用です。

  ブローカが選択されている場合、エージェント名はユーザ名の後にアンパサンドとコンピュータ名が続く形式です。

  メイン クラスは、エージェントに対して呼び出された main メソッドを含む Java クラスです。

  PID は、エージェントが実行されているプロセス ID です。

- VM プロパティ

  ［VM プロパティ］タブには、システム プロパティおよび環境変数が表示されます。 値は読み取り専用です。

  注: ［VM プロパティ］タブには、オフラインのエージェントの情報を表示できません。

- 環境変数

  ［環境変数］タブには、エージェントが実行されているコンピュータのシステム環境変数が表示されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［設定］-［エージェント］を選択します。

   ［エージェント］ウィンドウが表示されます。

2. ［エージェント］ペインで、エージェントまたはブローカを選択します。
3. 右側のペインで［情報］タブを選択します。

エージェントのトランザクションのキャプチャの停止および開始

［エージェント］ウィンドウでは、エージェントがトランザクションをキャプチャするかどうかを制御できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［設定］-［エージェント］を選択します。

   ［エージェント］ウィンドウが表示されます。

2. エージェントのトランザクションのキャプチャを停止するには、データのキャプチャを停止するエージェントについて、スライダを［オフ］に移動させます。
3. エージェントのトランザクションのキャプチャを開始するには、データのキャプチャを開始するエージェントについて、スライダを［オン］に移動させます。

プロパティの表示および更新

エージェントまたはブローカの設定プロパティを表示し更新できます。

プロパティは、たとえば、「全般」と「キャプチャ」というカテゴリに分類されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［設定］-［エージェント］を選択します。

   ［エージェント］ウィンドウが表示されます。

2. 左側の部分で、エージェントまたはブローカを選択します。
3. ［設定］タブをクリックします。
4. 1 つ以上のプロパティの値を変更します。
5. 保存していないすべての変更を破棄するには、［元に戻す］をクリックします。
6. 変更を保存するには、［保存］をクリックします。

DevTest ポータルでのエージェントのアップグレード

DevTest ポータルの［Settings］-［Agenats］ウィンドウから、エージェントを最新バージョンにアップグレードできます。 新しいバージョンが使用可能になると、エージェントの横に［Download］ボタンが表示されます。マウス カーソルを［Download］および［Pending］ボタンに重ねると、バージョン情報が表示されます。

注: 一部のエージェントは、DevTest ポータルでアップグレードできません。 ネイティブ エージェントや JRE 1.5 および **LisaAgent2.jar** を使用するエージェントは、手動でアップグレード (P. 31)する必要があります。

アップグレードできるのは、以下の要件を持つ純正エージェントのみです。

- JRE 1.6 以降
- エージェントは **InsightAgent.jar** ファイルから起動する必要があります。

次の手順に従ってください:

1. 最新の **LisaAgent.jar** ファイルを **LISA_HOME¥lib¥core¥agentupgrade** フォルダに保存します。
2. 左側のナビゲーション メニューから［Settings］-［Agenats］を選択します。

   ［Agenats］ウィンドウが表示されます。
3. ［Agenats］ペインから、［Download］ボタンをクリックします。

   ［Download］ボタンが「Pending」というテキストに変わります。
4. エージェントを再起動してアップグレード プロセスを完了します。

手動でのエージェントのアップグレード

エージェントが DevTest ポータルでアップグレードされる (P. 31)のが適切でない場合、この手順を使用してアップグレードすることができます。

次の手順に従ってください:

1. エージェントを停止します。
2. 既存の **LisaAgent.jar** ファイルを新しいバージョンに置換します。
3. エージェントを起動します。

オフライン エージェントの削除

［Agenats］ウィンドウの［Agenats］ペインから、オフライン エージェントを削除できます。 エージェントを削除すると、エージェントのトランザクションおよびフレームがすべて削除されます。

注: エージェントがオフラインのときに、［削除］ボタンが表示されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Settings］-［Agenats］を選択します。

   ［Agenats］ウィンドウが表示されます。

2. ［Agenats］ペインから、削除するエージェントを見つけます。
3. （オプション）エージェントを検索するためにフィルタ条件を使用するには、［Show Filter］をクリックします。
4. ［削除］をクリックします。

第 3 章: ビジネストランザクションの分析

CAI では、キャプチャされたビジネス トランザクション (P. 16)を表示できます。 これらのトランザクションはパスとも呼ばれます。

CAI は、トランザクションおよびそれらのフレームに関する情報を表示します。 この情報には、実行時間、スレッド使用率、ログ メッセージ、SQL ステートメント、要求および応答データ、およびスタック トレースが含まれます。

［Analyze Transactions］ウィンドウからトランザクションを表示します。トランザクションはグラフィカル ビューまたはリスト ビューに表示されます。 デフォルトでは、トランザクションはリスト ビューに表示されます。

注: 各トランザクションのフレームは、特定の実行順序に従います。これらは、特定のエージェントによって配置されません。

グラフィカル ビューは、トランザクションの視覚的な表現です。 このビューには、最大 100 のトランザクションが表示されます。このビューでは以下の操作を実行できます。

- フルテキストまたは期間によるパスの検索
- フレームをクリックすることによるトランザクションの詳細の表示。トランザクションの詳細では、トランザクションの視覚的なパスが表示され、フレームの詳細を含むタブが表示されます。たとえば、フレーム情報、タグ、要求、応答、スレッド名、ログ メッセージ、XML、例外（可能な場合）が表示されます。
- パスが繰り返されたトランザクションのパスの表示
- カテゴリによるパスのフィルタ
- パスのシェルフへの追加およびシェルフからの削除
- パスの図の PDF ファイルへのエクスポート
- パスの概要の表示および非表示
- キャプチャの無効化

リスト ビューでは、トランザクションは［Results］ペインにリスト表示されます。 このビューでは以下の操作を実行できます。

- フルテキストまたは期間によるトランザクションの検索
- パスの情報の表示。たとえば、パスおよびエージェント名、開始時刻、実時間、CPU 時間が表示されます。
- トランザクションのインポートおよびエクスポート機能

- ［Actions］列の  をクリックすることによるトランザクションの詳細の表示。 トランザクションの詳細では、パス グラフにトランザクションの視覚的なパスが表示され、フレームの詳細を含むタブが表示されます。たとえば、フレーム情報、タグ、要求、応答、スレッド名、ログ メッセージ、XML、例外（可能な場合）が表示されます。
- トランザクションのインポートおよびエクスポート機能
- パスが繰り返されたトランザクションのパスの表示

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウが表示されます。デフォルトでは、トランザクションはリスト ビューに表示されます。

2. グラフィカル ビューを開くには、  をクリックします。

   トランザクションがグラフィカル ビューに表示されます。

3. （オプション）［Show Outline］をクリックして、大きなトランザクションの一部分を一度に表示します。

このセクションには、以下のトピックが含まれています。

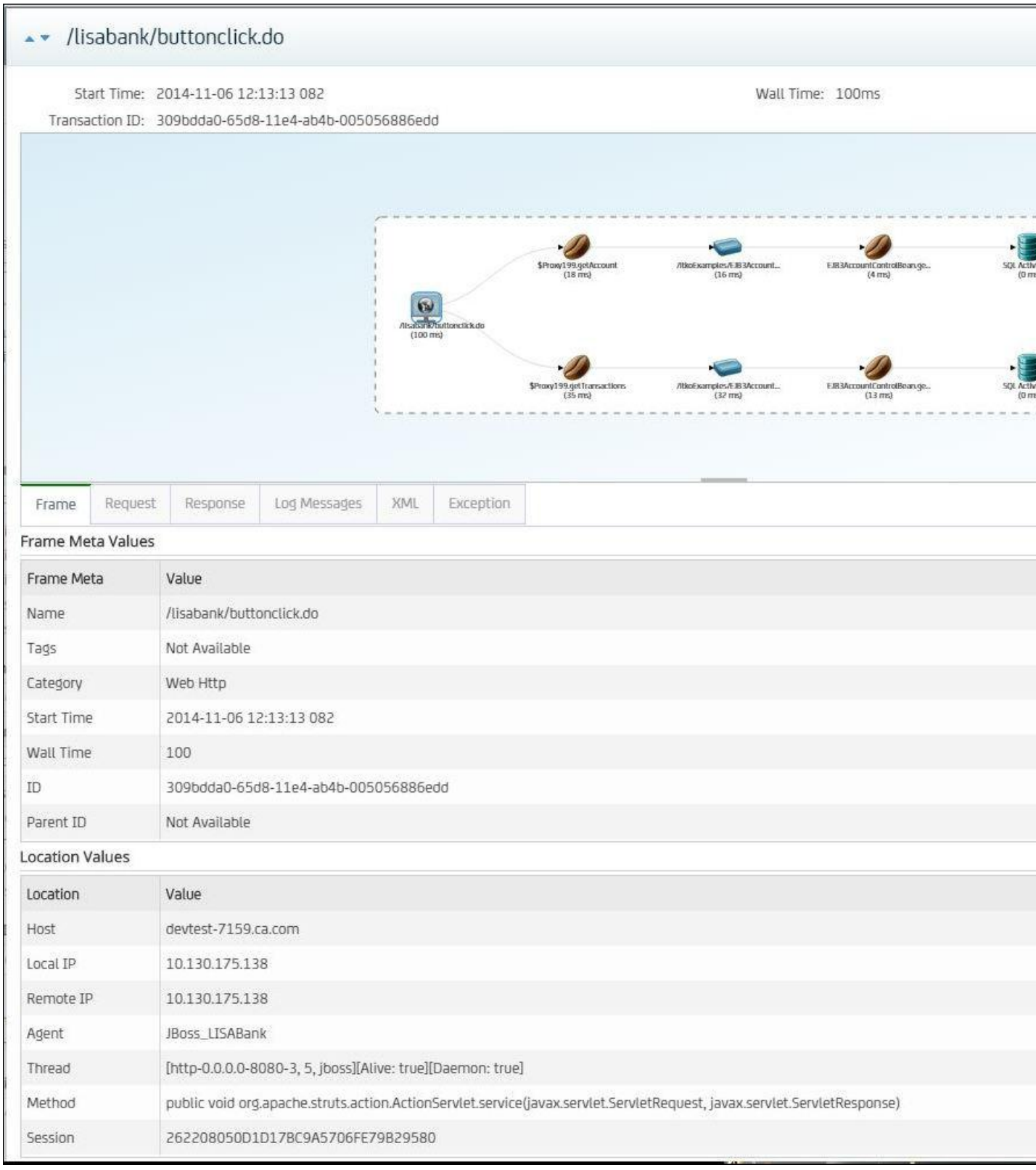
トランザクションの詳細の表示 (P. 36)
パスの図の PDF へのエクスポート (P. 47)
繰り返されるパスのマージ (P. 48)
エージェント キャプチャからのデータの除外 (P. 50)
トランザクションの検索およびフィルタ (P. 51)
パスのエクスポートおよびインポート (P. 54)

トランザクションの詳細の表示

トランザクション詳細ダイアログ ボックスには、パス内の各トランザクションに関する情報が含まれています。 このダイアログ ボックスは、以下の領域で構成されています。

- 上部の領域にはパス グラフ (P. 41) が含まれています。
- それより下の領域にはフレーム情報 (P. 43) が含まれています。

以下の図は、トランザクション詳細ダイアログ ボックスを示しています。


/lisabank/buttonclick.do
Start Time: 2014-11-06 12:13:13 082
Wall Time: 100ms
Transaction ID: 309bdda0-65d8-11e4-ab4b-005056886edd
$Proxy199.getAccount
(18 ms)
(16 ms)
(4 ms)
(100 ms)
$Proxy199.getTransactions
(35 ms)
(32 ms)
(13 ms)
Frame
Request
Response
Log Messages
XML
Exception
Frame Meta Values
Frame Meta
Value
Name
/lisabank/buttonclick.do
Tags
Not Available
Category
Web Http
Start Time
2014-11-06 12:13:13 082
Wall Time
100
ID
309bdda0-65d8-11e4-ab4b-005056886edd
Parent ID
Not Available
Location Values
Location
Value
Host
devtest-7159.ca.com
Local IP
10.130.175.138
Remote IP
10.130.175.138
Agent
JBoss_LISABank
Thread
[http-0.0.0.0-8080-3, 5, jboss][Alive: true][Daemon: true]
Method
public void org.apache.struts.action.ActionServlet.service(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
Session
262208050D1D17BC9A5706FE79B29580

パス グラフ情報

パス グラフには、パスとそのフレームがグラフィカルに表示されます。

パス グラフでは、以下の操作を実行できます。

- フレームを右クリックし、［Shelve］または［Shelve All Occurred Transactions］オプションを選択することにより、トランザクションをシェルフに追加したりシェルフから削除したりします。
- フレームを右クリックし、［Disable Capture］または［Disable URL Capture］オプションを選択することにより、キャプチャを無効にします (P. 50)。

Agent name

パスを生成したエージェントの名前。

Start time

フレームが実行を開始した日時。

Wall Time

フレームの実行時間。

Transaction ID

パスの一意の識別子。

Frame Information

フレーム情報には、パス グラフ内の選択されたフレームの詳細が表示されます。

Frame

メタデータの値と場所の値が含まれているフレーム情報。

Tags

フレームにあるフレーム タグ。

Request

選択したフレームが送信した実際の要求。この情報は、JDBC フレームに対しては表示されません。

Response

選択したフレームが受信した実際の応答。この情報は、JDBC フレームに対しては表示されません。

SQL

SQL コールに関する情報。この情報には SQL ステートメント、結果、および出力が含まれます。この情報は、JDBC フレームに対してのみ表示されます。

Thread Name

フレームが実行されたスレッドの名前。

CPU time

プロセッサがフレームの実行に費やした時間。

Utilization

スレッド使用率のパーセンテージ。

Log Messages

任意のログ メッセージ。デフォルトでは、ログ メッセージの最低レベルは**警告**です。最低レベルを変更するには、Java ログ レベル プロパティを設定します。

このプロパティは、DevTest ポータルの［エージェント］ウィンドウ (P. 19)から設定できます。プロパティは［設定］タブに表示されます。

XML

フレームでのインスツルメンテーション応答の RAW XML。この情報は、JDBC フレームに対しては表示されません。

Exception

トランザクションに例外が含まれている場合に表示詳細が情報されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウが表示されます。

2. （デフォルト）リスト ビューで、［Actions］列の  をクリックします。

3. グラフィカル ビューで、親フレームをダブルクリックします。

トランザクション詳細ダイアログ ボックスが表示されます。タイトル バーには、フレームのフル ネームが含まれています。

4. パス グラフ内の情報およびフレーム情報を確認します。
5. タイトル バーの左にある上方向および下方向の矢印をクリックして、前のトランザクションおよび次のトランザクションに移動します。
6. ［X］をクリックしてダイアログ ボックスを閉じます。

パス グラフ

パス グラフには、パスおよびそのフレームのグラフ表示が含まれています。 パス グラフは、トランザクション詳細ダイアログ ボックスの上部の領域に表示されます。

［Analyze Transactions］ウィンドウからグラフィカル ビューまたはリスト ビューのルート トランザクション フレーム (P. 16)を選択すると、トランザクション詳細ダイアログ ボックスが表示され、その上部にパス グラフが表示されます。

以下の方法で、［Analyze Transactions］ウィンドウからパス グラフを表示します。

- グラフィカル ビューのパスをクリックする。
- リスト ビューの［Actions］列から［Open transaction details］をクリックする。

アイコンは、フレームのタイプを示します。フレーム タイプの例として、EJB、HTTP、JMS、REST、Web サービス、DevTest、および不明があります。DevTest フレーム タイプは、テスト ケースまたは仮想サービス モデルの内部で実行するステップを表します。

アイコンの下のテキストは以下の情報を提供します。

- フレーム名
- 実行時間

［ズーム イン］および［ズーム アウト］を使用してパス表示のサイズを変更します。

フレーム名

フレーム名の形式は、フレームのタイプによって異なります。

HTTP のフレーム名は URL のパスの部分です。 名前には、メソッドも含まれています。 CAI コンパニオンへの VS トラフィックを使用して作成されたパスは、コンポーネント名の「virtualized」ラベルによって示されます。

たとえば、「**POST/itkoExamples/EJB3AccountControlBean(virtualized)** 」のようになります。

このコンパニオンの使用の詳細については、「CAI コンパニオンへの VS トラフィック (P. 225)」を参照してください。

JDBC のフレーム名は「SQL アクティビティ（*N*）」です。*N* は、フレームを生成するために使用された SQL ステートメントの数です。

JMS のフレーム名は以下の部分から構成されます。

- 操作のタイプを識別する文字列。 操作にメッセージの作成が含まれている場合、文字列は **send:** です。 操作にメッセージの消費が含まれている場合、文字列は **recv:** です。
- キューの名前
- （オプション）接続ファクトリの名前

たとえば、**recv:queue/ORDERS.REQUEST@ConnectionFactory** です。

JMS のフレーム名は **DevTest** にすることもできます。テスト ケースが JMS メッセージを受信すると、この名前が使用されます。

RMI のフレーム名は Java クラスおよび実行されたメソッドです。

webMethods のフレーム名はフロー サービス名です。

WebSphere MQ のフレーム名は以下の部分から構成されます。

- 文字列 **put** または **get**
- WebSphere MQ ホスト名
- キュー マネージャ名
- キュー名

たとえば、**put-172.24.255.255-QueueManager-ORDERS.REQUEST** などです。

Web サービスのフレーム名は URL のパスの部分です。

実行時間

特定の状況では、左端のコンポーネントの下の実行時間がトランザクション全体の時間になります。

実行時間が非常に短く、ゼロに切り捨てられる場合、値［**0 ミリ秒**］が表示されます。

フレーム情報

トランザクション詳細ダイアログ ボックスには、パス グラフで選択されているフレームに関する情報が表示されます。 フレーム情報は、トランザクション詳細ダイアログ ボックスの下の部分に表示されます。

フレーム情報には、以下のタブが含まれています。

- Frame
- 要求
- 応答
- ログ メッセージ
- XML
- 例外
- ペイロード
- SQL サマリ

［Frame］タブおよび［ログ メッセージ］タブは、すべてのフレーム タイプに対して表示されます。

Frame

［Frame］タブには、選択されたコンポーネントのメタデータの値と場所の値が表示されます。 以下の詳細が表示されます。

フレーム メタデータ

- 名前： フレームの名前。
- タグ： フレームと関連付けられたフレーム タグ。
- カテゴリ： フレームのカテゴリ。
- 開始時間： フレームが実行を開始した時間。
- 実時間： フレームの実行の開始から終了までにかかった時間。「ルート フレーム応答時間」とも呼ばれます。
- ID： 一意のフレーム ID。
- 親 ID： 現在のフレームの親フレーム ID。

場所の値

- ホスト： フレームが実行されたコンピュータの名前。

- エージェント： フレームをキャプチャしたエージェントの名前。
- スレッド名： フレームが実行中のスレッド名。
- メソッド： フレームでキャプチャされた Java メソッド。
- セッション： このフレームが関連付けられるセッションのセッション ID。

フレームの実行時間が非常に短く、ゼロに切り捨てられる場合、値［**0 ミリ秒**］が表示されます。

［Request］タブ

［要求］タブには、選択したフレームが送信した実際の要求が表示されます。 このタブは、非 JMS および非 JDBC フレームに対してのみ表示されます。

WebSphere MQ トランザクションにバイナリ ペイロードが含まれる場合、［要求］タブにはペイロードの人間が理解できるバージョンが表示されます。 非テキスト文字は削除されます。 ［XML］タブで元のバイナリ ペイロードを表示できます。

［Response］タブ

［応答］タブには、選択したフレームが受信した実際の応答が表示されます。 このタブは、非 JMS および非 JDBC フレームに対してのみ表示されます。

WebSphere MQ トランザクションにバイナリ ペイロードが含まれる場合、［応答］タブにはペイロードの人間が理解できるバージョンが表示されます。 非テキスト文字は削除されます。 ［XML］タブで元のバイナリ ペイロードを表示できます。

［ログ メッセージ］タブ

［ログ メッセージ］タブには、すべてのログ メッセージが含まれます。以下の詳細が表示されます。

- 情報のレベル
- ロガー名（ある場合）
- ログ メッセージのテキスト

デフォルトでは、ログ メッセージの最低レベルは**警告**です。 最低レベルを変更するには、**Java ログ レベル** プロパティを設定します。

このプロパティは、DevTest ポータルの［エージェント］ウィンドウ (P. 19) から設定できます。 プロパティは［設定］タブに表示されます。

列を並べ替えるには、列見出しのドロップダウン矢印をクリックし［昇順にソート］または［降順にソート］を選択します。 列を表示するか非表示にするには、列見出しのドロップダウン矢印をクリックし、列をポイントして、チェック ボックスをオンまたはオフにします。

［XML］タブ

［XML］タブは、フレームでのインスツルメンテーション応答の RAW XML を表示します。 このタブは、JDBC フレーム以外のすべてのフレームに対して表示されます。

［XML］タブは付加的です。 右側のフレームに含まれる情報は左側のフレームより少なくなります。 最初のフレームには、完全な応答が含まれます。

［例外］タブ

フレームに例外が含まれる場合、［例外］タブにスタック トレースが表示されます。

この機能には以下の前提条件があります。

- 例外プロトコルのキャプチャ レベル (P. 27) が、エージェントに関して［全データ］に設定されている必要があります。
- Capture all the supported exception frames プロパティが、エージェントに関して有効になっている必要があります。 ［設定］タブのこのプロパティにアクセスするには、［トランザクション］カテゴリを選択します。

［ペイロード］タブ

［ペイロード］タブには、JMS ペイロードが表示されます。 このタブは、JMS フレームに対してのみ表示されます。

［SQL サマリ］タブ

［SQL サマリ］タブは、SQL 情報のサマリを提供します。このタブは、JDBC フレームに対してのみ表示されます。

以下の詳細が表示されます。

- 各 SQL ステートメントに割り当てられる連続する識別子
- SQL ステートメント
- 返された行の数
- 呼び出しの数
- 平均時間（ミリ秒）
- 合計時間（ミリ秒）

SQL ステートメントをクリックすると、結果が表示されます。

SQL ステートメントをダブルクリックすると、ステートメント全体を表示するダイアログ ボックスが表示されます。

列を並べ替えるには、列見出しのドロップダウン矢印をクリックし［昇順にソート］または［降順にソート］を選択します。 列を表示するか非表示にするには、列見出しのドロップダウン矢印をクリックし、列をポイントして、チェック ボックスをオンまたはオフにします。

パスの図の PDF へのエクスポート

パスの図を PDF ファイルにエクスポートして、パス情報を表示、印刷、および保存できます。 リスト ビューまたはグラフィカル ビューからパスを PDF ファイルにエクスポートします。

次の手順に従ってください:

リスト ビューの場合

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。
2. ［Actions］列の  をクリックします。

   トランザクション詳細ダイアログ ボックスが表示されます。
3.  をクリックします。

   パスの図がプレビュー ブラウザに表示されます。
4. マウス カーソルをブラウザの下部の右側に移動させて、タスク バーを表示させます。

   タスク バーには、ズーム、保存、および印刷のためのオプションがあります。
5. PDF ファイルを保存するには、［Save］をクリックします。
6. ファイル名を入力して、［Save］をクリックします。
7. PDF を印刷するには、印刷ボタンをクリックします。

   印刷セットアップが表示されます。
8. ［Print］をクリックします。

グラフィカル ビューの場合

1. グラフィカル ビューで、  をクリックします。

   パスの図がプレビュー ブラウザに表示されます。
2. マウス カーソルをブラウザの下部の右側に移動させて、タスク バーを表示させます。
3. PDF ファイルを保存するには、［Save］をクリックします。
4. ファイル名を入力して、［Save］をクリックします。

5. PDF を印刷するには、印刷ボタンをクリックします。

   印刷セットアップがブラウザに表示されます。

6. ［Print］をクリックします。

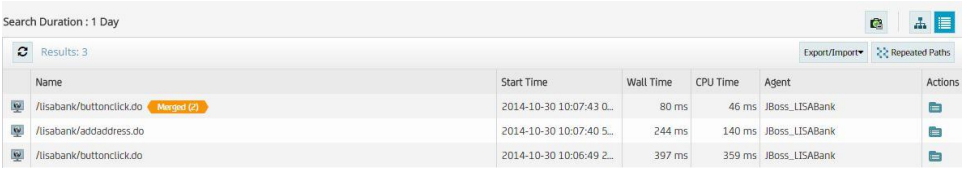
繰り返されるパスのマージ

繰り返されるパスには、パスに発生する重複したトランザクションが含まれています。 これらのパスをマージして、同じトランザクションを識別し、［Results］ペインにリスト表示されるトランザクションの数を減らすことができます。 繰り返されるパスのマージは、不具合の原因を識別に役立ちます。 たとえば、あるアプリケーションに関して発生し、繰り返されるすべてのログイン トランザクションが失敗する場所を確認することができます。

リスト ビューまたはグラフィカル ビューで繰り返されるパスをマージできます。

グラフィカル ビューおよびリスト ビューでは、繰り返されるパスがオレンジ色のバッジで示されます。 バッジの数字は、そのトランザクションに関して繰り返されたパスの総数です。

以下の図には、2 つのマージされた繰り返されるパスを持つトランザクションのリスト ビューが示されています。

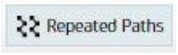


以下の図には、2 つのマージされた繰り返されるパスを持つトランザクションのグラフィカル ビューが示されています。

次の手順に従ってください:

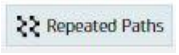
1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウが表示されます。

2. リスト ビューまたはグラフィカル ビューで、 

をクリックします。

   繰り返されるパスが 1 つのトランザクションにマージされます。 オレンジ色のバッジには、繰り返されるパスの数が表示されます。

3. トランザクションのマージを解除するには、 

クリックします。

エージェント キャプチャからのデータの除外

DevTest Java エージェントは、関連しないデータや貴重でないデータを含む大量のデータをキャプチャする場合があります。たとえば、アプリケーションへのログインに関するデータは重要ではない場合があります。

エージェントが特定のタイプのトランザクション フレームをキャプチャしないように指定することができます。

一部のフレームは、除外されることをサポートしません。 サポートされていないフレームを除外しようとすると、以下のメッセージが表示されます。

```
Failed to create disable capture rule for {{名前}}. {{理由}}
```

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウが表示されます。
2. 以下のいずれかの操作を実行します。
   - グラフィカル ビューを表示します。
   - リスト ビューを表示し、［Actions］列から［Open transaction details］をクリックします。
3. 除外するフレームを右クリックし、［Disable Capture］または［Disable URL Capture］を選択します。
4. 現在除外されているフレームを表示するには、［Manage exclude frame capture rules］ボタン  をクリックします。
5. 除外されるフレームのリストからフレームを削除するには、以下のアクションを実行します。

   a. ［Manage exclude frame capture rules］ボタン  をクリックします。

   b. フレームを選択します。

   c. ［Include］をクリックします。

トランザクションの検索およびフィルタ

ペイロードの全文検索の入力 (P. 51)、期間の制限 (P. 52)、およびフィルタオプションを使用した検索の絞り込み (P. 53)により、トランザクションを検索およびフィルタできます。

トランザクションの検索

CAI では、インテリジェント検索機能を使用してトランザクションを検索できます。 ［Enter Search Text］フィールドからペイロード テキスト検索を入力します。 たとえば、**応答に lisa_simpson があり、カテゴリに EJB がある**ペイロード検索文字列を入力できます。

また、たとえば、トランザクション ID、カテゴリ（EJB など）、ローカルまたはリモート IP アドレスといったキーワードを使用することもできます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。
2. 検索テキスト フィールドにテキスト検索条件を入力して、  をクリックします。

   ［Results］リストに、検索条件に基づくパスのリストが表示されます。
3. （オプション） ［Refine by］ペインのオプションを選択することにより、検索をさらに絞り込みます。

時間によるトランザクションの検索

［Analyze Transactions］ウィンドウには、数百のトランザクション パスが含まれる場合があります。 時間および日付によってトランザクションをフィルタすることにより、分析するパスを絞り込むことができます。 検索の結果は、［Results］ペインにリスト表示されます。 リストでは一度に最大 25 のトランザクションを表示できます。

以下の周期時間範囲を指定します。

- 分（1 ～ 30）
- 時間（1 ～ 12）
- 日（1 ～ 15）
- すべての時間

［詳細］時間オプションで以下の日付範囲条件を指定します。

- 開始日時
- 終了日時

デフォルトの日付範囲は、1 日です。

トランザクションのフィルタ

［Refine By］ペインのフィルタ オプションを使用して、［Results］リストに表示されるトランザクションの数を絞り込みます。

検索テキスト フィールドからフィルタ オプションを入力するか、チェック ボックスをオンにします。 たとえば、フィルタするクラス名を知っている場合は、名前を入力し、Enter キーを押します。 フィルタにより、そのクラス名に対して特定のチェック ボックスが自動的にオンになります。［Results］ペインに、そのクラス名が含まれているトランザクションのリストが表示されます。

以下のフィルタ オプションを使用できます。

- Agent
- Category
- Class Name
- Execution Time
- Local IP
- Operation
- Point of Interest
- Remote IP
- Tags
- Transaction ID

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   デフォルトでは、［Analyze Transactions］ウィンドウはリスト ビューで表示されます。

2. ［Refine By］ペインで、適切なチェック ボックスをオンにするか、テキストを入力します。

   ［Search Duration］ペインに、指定されたフィルタ オプションに基づくトランザクションのリストが表示されます。

カテゴリによるグラフのフィルタ

グラフィカル ビューから、特定のカテゴリのフレームを持つトランザクションをフィルタできます。カテゴリ リストに表示されるオプションは、そのトランザクションに対してサポートされているカテゴリによって異なります。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   デフォルトでは、［Analyze Transactions］ウィンドウはリスト ビューで表示されます。

2. ［Search Duration］ペインの上部にある  をクリックします。

3. 

をクリックし、適切なカテゴリ チェック ボックスをオンまたはオフにします。

   ［Results］ペインに、指定されたカテゴリに基づくトランザクションのリストが表示されます。

パスのエクスポートおよびインポート

［Analyze Transactions］ウィンドウを使用して、CAI データベースからの 1 つ以上のパスをエクスポートできます。 このパスは、他の CAI インストールへインポートすることができます。また、Splunk からインポートすることもできます。 CA サポートによるデバッグのために対象のパスをエクスポートすることができます。

パスのエクスポート

キャプチャされたトランザクション パスを ZIP ファイルにエクスポートできます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。
2. ［Results］ペインから、エクスポートする 1 つ以上のパスを選択します。
3. ［Export/Import］をクリックし、いずれかのオプションを選択します。
   - Export Selected -［Results］リスト内の選択されたパスをエクスポートします。
   - Export Page -［Results］リスト内のすべてのトランザクションをエクスポートします。

   ［Export Path］ウィザードが表示されます。
4. エクスポート先の ZIP ファイルの名前を入力します。
5. ［Export and download path］をクリックします。

   ZIP ファイルが作成され、ダウンロードされます。

パスのインポート

トランザクション パスは、［Analyze Transctions］ウィンドウを使用してインポートできます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。
2. ［Export/Import］をクリックし、［Import］を選択します。

   ［Import Path］ウィザードが表示されます。
3. ［Choose File］をクリックし、ZIP ファイルを選択して、［Open］をクリックします。
4. ［Open］をクリックします。

   ［Imported］列に以下のように表示されます。
   - ［Yes］（パスがデータベースにインポートされる場合）。
   - ［Duplicate］（パスすでにデータベースに存在する場合）。
   - ［Failed］（パスがインポートされなかった場合）。
5. ［OK］をクリックします。

   確認ダイアログ ボックスが表示されます。
6. ［OK］をクリックし、［Results］ペインの［Refresh］をクリックします。

   インポートされたパスは、［Results］リストに表示されます。

Splunk からのパスのインポート

［Analyze Transactions］ウィンドウで、Splunk からキャプチャされたパスをインポートできます。 Splunk および CAI Agent Light の使用の詳細については、「Agent Light (P. 229)」を参照してください。 ［Import from Splunk］ダイアログ ボックスの［Search］フィールドは、エージェント ライト ビルトイン Splunk 検索文字列です。 Splunk HTTP トランザクションを生成する別の検索文字列を入力することができます。 HTTP 検索クエリ言語構文の詳細については、「HTTP 用の Agent Light (P. 234)」を参照してください。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウが表示されます。

2. ［Export/Import］をクリックし、［Import from Splunk］を選択します。

   ［Import from Splunk］ダイアログ ボックスが表示されます。

3. フィールドに入力し、［Import］をクリックします。

   確認ダイアログ ボックスは、正常にインポートされたことを示します。

4. ［OK］をクリックしてダイアログ ボックスを閉じます。

5. ［Results］ペインから［Refresh］  をクリックして、Splunk パスを表示します。

   Splunk からのインポートされたパスは、［Results］ペインのリストの一番上に表示されます。

第 4 章: シェルフの使用

CAI では、一連のトランザクション フレームをシェルフに追加して、アーティファクトを生成することができます。 トランザクション フレームをシェルフに追加することは、「*シェルビング*」と呼ばれます。 フレームに関する作業を一度に 1 つずつ行う代わりに、シェルフから、複数のトランザクション フレームに関する仮想サービス、ベースライン、およびドキュメントを作成することにより、アーティファクトを生成します。

注: アーティファクトを生成する場合は、必ず、含めないトランザクション フレームを削除してください。 トランザクションの削除の詳細については、「シェルフのフレームの削除 (P. 66)」を参照してください。

トランザクションは、以下のものからシェルフに追加できます。

- グラフィカル ビューの［Analyze Transactions］ウィンドウ
- トランザクション詳細ダイアログ ボックス (P. 36)

トランザクションをシェルフに追加する方法の詳細については、「トランザクション フレームのシェルフへの追加 (P. 63)」を参照してください。

［Analyze Transactions］ウィンドウの［Search Duration］バーの上部にシェルフ  アイコンがあります。アイコンの右隅の数字は、シェルフに追加されたフレームの総数を示しています。

以下のアーティファクトを生成できます。

- VS の作成
- ベースラインの作成
- ドキュメントの作成

各トランザクションに対して生成できるアーティファクトのタイプは、色の付いたラベルで示されます。 以下の図は、トランザクションに対して生成できるアーティファクトのタイプの例を示しています。 このトランザクションから、ステートレス仮想サービスや、ステートフル ベースライン、統合されたベースライン、または展開されたベースラインを作成できます。

シェルフには、アーティファクトを生成する 2 つの方法があります。

- ワンクリック方式

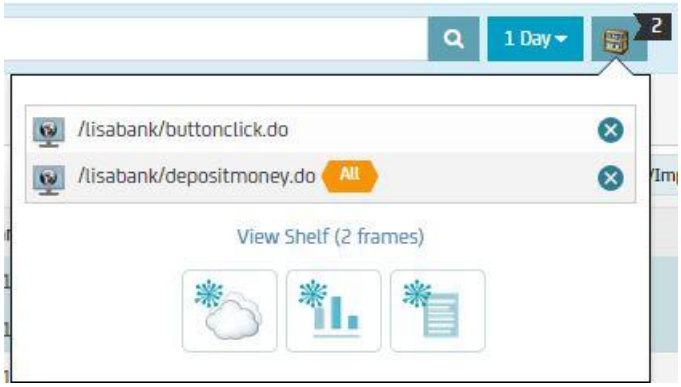
  この方法では、シェルフダイアログ ボックスを開かずに、シェルフに追加されているトランザクションについてのベースライン、仮想サービス、およびドキュメントを迅速に作成できます。

- シェルフ ダイアログ ボックス

  この方法では、シェルフ ダイアログ ボックスにアクセスして、シェルフに追加されたトランザクションを検索し、ベースライン、仮想サービス、およびドキュメントを作成することができます。

ワンクリック方式

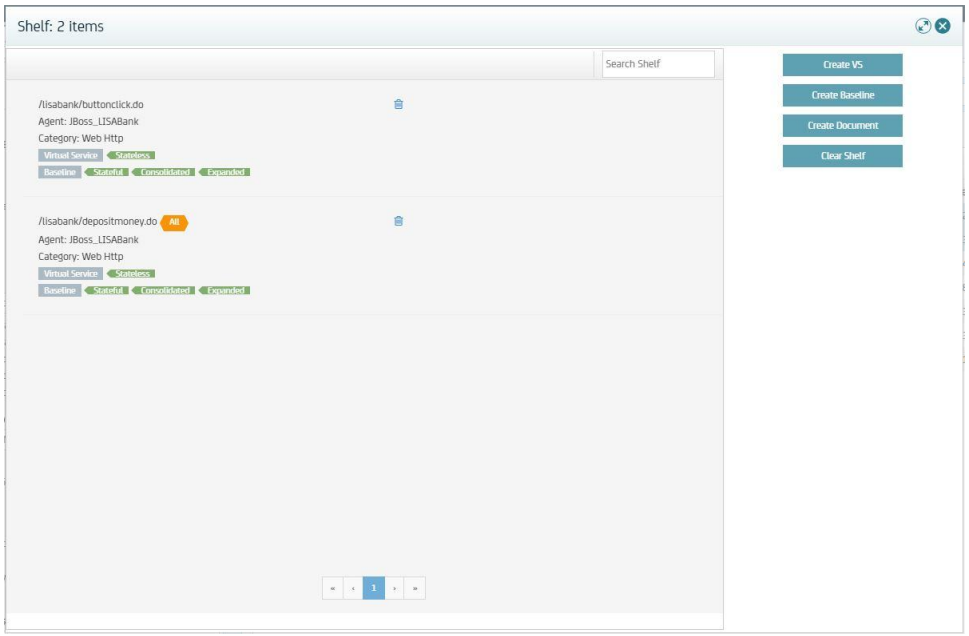
シェルフは、ワンクリックでアーティファクトを生成する方法を提供します。シェルフ アイコン  にマウス カーソルを重ねることにより、シェルフに追加されたトランザクションを表示し、シェルフダイアログ ボックスを開かずにアーティファクトを生成することができます。 以下の図は、ワンクリック ポップアップを示しています。



最大 5 つのトランザクションが一度に表示されます。トランザクションを削除すると、シェルフの次のトランザクションが表示されます。

シェルフ ダイアログ ボックス

シェルフ ダイアログ ボックスを開くには、シェルフ アイコン  をクリックするか、ワンクリック方式から［View Shelf (X frames)］をクリックします。 以下の図は、シェルフ ダイアログ ボックスを示しています。



シェルフ ダイアログ ボックスから、リスト表示されるトランザクションに関して以下の操作を実行できます。

- トランザクションの検索 (P. 66)
- トランザクション フレームの削除 (P. 66)
- アーティファクト（仮想サービス、ベースライン、およびドキュメント）の作成 (P. 67)
- シェルフのすべての内容のクリア (P. 68)

注: 仮想サービスおよびベースラインの作成の詳細については、「仮想サービスの作成 (P. 149)」および「ベースラインの作成 (P. 93)」を参照してください。

このセクションには、以下のトピックが含まれています。

シェルフのトランザクションの作業方法 (P. 63)

シェルフのトランザクションの作業方法

このシナリオでは、一連のトランザクション フレームをシェルフに追加して、アーティファクトを生成する方法を説明します。

1. トランザクション フレームをシェルフに追加します (P. 63)。
2. 発生したすべてのトランザクションをシェルフに追加します (P. 64)。
3. マージされたトランザクションをシェルフに追加します (P. 65)。
4. シェルフのフレームを削除します (P. 66)。
5. シェルフのパスを検索します (P. 66)。
6. アーティファクトを生成します (P. 67)。
7. シェルフからすべてのトランザクションをクリアします (P. 68)。

トランザクション フレームのシェルフへの追加

トランザクションおよびフレームをシェルフに追加して、仮想サービス、ベースライン、およびドキュメントを作成します。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   デフォルトでは、［Analyze Transactions］ウィンドウに、キャプチャされたトランザクションがリスト ビューで表示されます。

2. リスト ビューで、［Actions］列の  をクリックします。

3. グラフィカル ビューで、  をクリックします。

   そのトランザクションがパス グラフに表示されます。

4. ルート トランザクションまたはフレームを右クリックして、［Shelve］を選択します。

   グラフィカル ビューおよびリスト ビューで、青色のピンのアイコンが、シェルフに追加されたトランザクション フレームの上に表示されます。フレームがシェルフに追加されます。シェルフに追加されているトランザクションの総数が 1 つ増加します。

発生したすべてのトランザクションのシェルフへの追加

キャプチャされたトランザクション中に発生したトランザクションおよびフレームを、別々にシェルフに格納せず、すべてシェルフに格納することができます。シェルフには、発生したトランザクションすべてが 1 つのトランザクションとして表示されます。 シェルフに格納される単一の項目として、仮想サービスを作成し、ベースラインを作成し、ドキュメントを作成します。 たとえば、ルート トランザクション レベルの同じパスで発生するトランザクションすべてをシェルフに格納すると、すべてのフレームが 1 つのトランザクションとしてシェルフに追加されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。

   ［Analyze Transactions］ウィンドウには、デフォルトではリスト ビューで表示されます。

2. ［Actions］列の  をクリックします。
3. トランザクション詳細が表示され、上部にパス グラフも表示されます。
4. グラフィカル ビューで、  をクリックします。

   そのトランザクションがパス グラフに表示されます。

5. トランザクションを右クリックして、［Shelve All Occurred Transactions］を選択します。

   グラフィカル ビューおよびパス グラフでは、青色のピンおよび単語［**すべて**］が、シェルフに格納されたトランザクション フレームの上部に表示されます。フレームがシェルフに追加されます。シェルフに追加されているトランザクションの総数が 1 つ増加します。

   以下の図は、パス内で発生したトランザクションすべてがシェルフに格納されたトランザクションを示しています。

マージされたトランザクションのシェルフへの追加

パスが反復しているマージされたトランザクションをシェルフに追加できます。 パスが反復しているグループをシェルフに追加することにより、一度に 1 つのトランザクションではなく、一度に複数のトランザクションのアーティファクトを作成できます。 マージされたトランザクションがシェルフに追加されると、マージされた反復するパスの数が青色のバッジに表示されます。

注: ステートフル ベースラインは、マージされたトランザクションについてはサポートされません。 統合されたベースラインと展開されたベースラインのみを作成できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Analyze Transactions］を選択します。
2. グラフィカル ビューまたはリスト ビューで、［Repeated Paths］をクリックします。
3. **両方**のビューで、パスにオレンジ色のマージ バッジが表示されます。
4. グラフィカル ビューから、マージされたパスを右クリックし、［Shelve (x number) Merged Transactions］を選択します。

   マージされたトランザクションがシェルフに配置されます。

シェルフのフレームの削除

アーティファクトを生成しないトランザクション フレームを削除できます。 トランザクション フレームは、シェルフ ダイアログ ボックスを使用するか、ワンクリック方式を使用して削除できます。 ワンクリック方式の詳細については、「シェルフの使用 (P. 59)」を参照してください。

次の手順に従ってください:

シェルフ ダイアログ ボックスの場合

1. ［Analyze Transactions］ウィンドウで  をクリックします。

   シェルフ ダイアログ ボックスに、シェルフに追加されたフレームのリストが表示されます。

2. 削除するフレームを見つけて、  をクリックします。

   シェルフからフレームが削除されます。

ワンクリック方式ポップアップの場合

1. ［Analyze Transactions］ウィンドウで、マウス カーソルをシェルフ アイコンに重ねます。
2. 削除するフレームを見つけて、［X］をクリックします。

シェルフのパスの検索

検索機能により、全文検索条件を入力して、シェルフのパスを検索できます。

次の手順に従ってください:

1. ［Analyze Transactions］ウィンドウで  をクリックします。

   シェルフに追加されたすべてのコンポーネントのリストがシェルフに表示されます。

2. ［Search Shelf］フィールドにテキスト検索条件を入力します。

   シェルフは、検索条件に基づくコンポーネントのリストを返します。

アーティファクトの生成

シェルフにあるトランザクション フレームについて、以下のビジネス アーティファクトを生成します。

- 仮想サービス
- ベースライン
- ドキュメント

生成できるアーティファクトのタイプは、トランザクションのリストの色の付いたラベルで示されます。

特定のタイプのトランザクションに関するアーティファクトの作成の詳細については、「ベースラインの作成 (P. 93)」および「仮想サービスの作成 (P. 149)」を参照してください。

以下の手順は、シェルフへのトランザクション フレームの追加が完了していることが前提となっています。 トランザクション フレームをシェルフに追加する方法の詳細については、「トランザクション フレームのシェルフへの追加 (P. 63)」を参照してください。

次の手順に従ってください:

ワンクリック方式の場合

1. ［Analyze Transactions］ウィンドウを開きます。
2. マウス カーソルをシェルフ アイコン  に重ねます。

   シェルフに追加されたトランザクション フレームのリストがシェルフに表示されます。
3. （オプション）［X］をクリックして、アーティファクトを作成しないフレームを削除します。
4. アーティファクト オプションをクリックします。
5. プロンプトに従います。

シェルフ ダイアログ ボックスの場合

1. ［Analyze Transactions］ウィンドウを開きます。
2.  をクリックします。

シェルフに追加されたトランザクション フレームのリストがシェルフに表示されます。

3. （オプション）［X］をクリックして、アーティファクトを作成しないフレームを削除します。
4. アーティファクト オプションをクリックします。
5. （オプション）［Locate agent in the topology map］をクリックして、エージェント マップを表示します。
6. （オプション）［Show All Agents］を選択して、トランザクションに関するすべてのエージェントを表示します。
7. ［Create］をクリックします。

シェルフからのすべてのトランザクションのクリア

シェルフ ダイアログ ボックスからの［Clear Shelf］をクリックすることにより、現在シェルフに追加されているすべてのトランザクションを削除します。

第 5 章: チケットの作成と管理

CAI では、障害を含む、アプリケーションの動作に関する詳細情報をキャプチャできます。 この機能は、テスターと開発者の連携を強化することにより、障害の解決の迅速化に役立ちます。

アプリケーションは、エージェントが有効なコンピュータで実行されている必要があります。 チケットはアプリケーション内で作成されます。 チケットは DevTest ポータルに表示されます。

チケットには、タイトル、重大度レベル、および説明などの情報が含まれます。 また、チケットには、対応するパス (P. 33)も含まれます。パスは、基盤となるコンポーネントのアクションを示します。

このセクションには、以下のトピックが含まれています。

アプリケーションでのチケットの作成

この手順は、エージェントが有効なコンピュータでアプリケーションが実行されていると仮定します。

電子メール機能が動作するには、電子メール設定 (P. 72)を設定する必要があります。

HP ALM - Quality Center 機能を有効にするには、「HP ALM - Quality Center へのチケットの送信 (P. 73)」を参照してください。

作業を開始する前に、［設定］-［エージェント］ウィンドウを開き、HTTP サーバ プロトコルのキャプチャ レベル (P. 27)が［全データ］に設定されていることを確認します。

次の手順に従ってください:

1. Alt キーを押したまま、アプリケーション ウィンドウ内の任意の場所をクリックします。

   チケット送信ダイアログ ボックスが表示されます。

2. ［Create a Ticket］をクリックします。
3. 以下の情報を入力します。
   - タイトル： チケットのタイトル。
   - 電子メール： 新しいチケットに関するメッセージはこの電子メール アドレスに送信されます。 電子メール設定が設定されている場合にのみ、このフィールドが表示されます。
   - 外部の障害のトラッキング番号
   - 重大度： オプションは、［低］、［中］、［高］、および［クリティカル］です。
   - 説明： このフィールドはオプションです。
4. キャプチャ時にアプリケーションのスクリーン ショットを追加する場合は、［スクリーンショットの取得］チェック ボックスがオンになっていることを確認します。
5. HP ALM - Quality Center にチケットを送信する機能が有効になっている場合は、［Raise this as a defect in Quality Center］チェック ボックスをオンにします。
6. ［送信］をクリックします。

7. HP ALM - Quality Center にチケットを送信する機能が有効になっている場合は、以下のアクションを実行します。

    a. HP ALM - Quality Center のユーザ名、パスワード、ドメイン、およびプロジェクトを入力し、［保存］をクリックします。

    b. 追加の一連のフィールドが表示される場合は、必要な情報を入力し、［保存］をクリックします。

    チケットが送信されたことを示すメッセージが表示されます。

8. ［Manage Tickets］ウィンドウにチケットを表示するには、［View Ticket］をクリックします。

9. ［閉じる］をクリックして、チケット送信ダイアログ ボックスを閉じます。

10. ［Home］ページの［New Ticket Alerts］ポートレットにチケットを表示するには、［Home］ページの［Refresh］をクリックします。

新しいチケット用の電子メール設定

チケットを作成すると、CAI は指定された電子メール アドレスに電子メール通知を送信します。 電子メールには、［Actions］列の［Link］をクリックすることににより［Manage Tickets］ウィンドウからアクセスできるチケットへのリンクが含まれています。

チケットを作成する前に、ブローカの以下のプロパティ (P. 30)を設定します。 ［設定］タブのこれらのプロパティにアクセスするには、［Tickets］カテゴリを展開し、［電子メール］選択します。

以下のプロパティが表示されます。

SMTP サーバ

電子メールの送信に使用する SMTP サーバの名前が含まれます。 認証なしの接続によって電子メールを送信できるサーバを使用する必要があります。

送信元

電子メールに表示される送信者を指定します。

サブジェクト

電子メールの件名を指定します。デフォルト値では、「**%1**」という文字がチケットのタイトルに置き換えられます。

ボディ

電子メールの本文を指定します。デフォルト値では、「**%1**」という文字がチケットへのリンクに置き換えられます。

HP ALM - Quality Center へのチケットの送信

チケットを作成する (P. 70)ときに、チケットを HP ALM - Quality Center に送信するように指定できます。

HP ALM - Quality Center 11 および 12 がサポートされています。

この機能を有効にするには、以下のタスクを実行します。

- 障害サブミッタのインストール
- HP ALM - Quality Center の設定プロパティの設定

障害サブミッタのインストール

インストール プロセスは、**LISA_HOME¥bin** ディレクトリに以下のログ ファイルを作成します。

- **InstallUtil.InstallLog**
- **LisaQCServices.InstallLog**

また、インストール プロセスは、**LisaQCServices.exe.config** という名前の設定ファイルも作成します。デフォルトのポート番号 10017 を変更するためにこのファイルを使用できます。

注: **lisa-qc-defect-submitter-install.bat** ファイルを実行し、指定されたサービスがすでに存在することを示すエラーが表示される場合は、障害サブミッタの以前のインストールが削除されていない可能性があります。**lisa-qc-defect-submitter-uninstall.bat** ファイルを実行し、再度 **lisa-qc-defect-submitter-install.bat** ファイルを実行します。

次の手順に従ってください:

1. HP Quality Center クライアント コンポーネントをインストールします。
2. DevTest Solutions をインストールします。
3. **LISA_HOME¥addons¥qc-defect-submit** ディレクトリに移動し、**qc_defect_submitter.exe** ファイルを実行します。

   セットアップ ウィザードが表示されます。
4. ［次へ］をクリックします。

   ［Select Destination Directory］手順が表示されます。

5. インストール先ディレクトリを **LISA_HOME** ディレクトリに設定し、［Next］をクリックします。

   ディレクトリがすでに存在することを示すメッセージが表示されます。

6. ［Yes］をクリックします。

   ファイルが抽出されます。 ファイルには、インストールおよびアンインストール用のバッチ ファイルが含まれます。

7. ［終了］をクリックします。
8. 管理者としてコマンド プロンプトを開きます。
9. **LISA_HOME¥bin** ディレクトリに移動し、**lisa-qc-defect-submitter-install.bat** ファイルを実行します。

   インストールが完了したことを示すメッセージが表示されます。

10. コントロール パネルを開き、サービス管理ツールを表示します。
11. **LisaQCService.QCDefectSubmitter** サービスを見つけます。
12. 自動的に実行されるようにサービスを設定し、サービスを開始します。

HP ALM - Quality Center の設定プロパティの設定

CAI は、必要なフィールドについて HP ALM - Quality Center インスタンスにクエリを実行します。 これらのフィールドは、チケット送信ダイアログ ボックスに追加されます。

次の手順に従ってください:

1. ［エージェント］ウィンドウを開きます。
2. 左側の部分で、ブローカを選択します。
3. ［設定］タブをクリックします。
4. ［Application Insight］カテゴリを展開し、［Quality Center］を選択します。
5. 以下のプロパティを設定します。

   QC host

   HP ALM - Quality Center REST API の IP アドレスまたはホスト名。 以下に例を示します。

   ```
   172.24.255.255
   ```

QC port

HP ALM - Quality Center REST API がリスンしているポート。 以下に例を示します。

8080

6. プロパティの変更を保存します。
7. DevTest Java エージェントを再起動します。

チケットの管理

［Manage Tickets］ウィンドウでは、キャプチャされ、CAI データベースに格納されたすべてのチケットを表示できます。

以下の操作を実行できます。

- チケットの検索 (P. 76)
- チケットのトランザクションの表示 (P. 79)
- チケット情報の編集 (P. 80)
- チケットの直接 URL の取得 (P. 78)
- チケット イメージの表示

特定のレポート元、ステータス、および期間によってチケットを検索します。 チケットは、日付によって降順に表示されます。 新しいチケットおよび更新されたチケットは、［Home］ページの［New Tickets Alert］ポートレットにも表示されます。

チケットは、以下のいずれかのステータスになります。

- 新規
- 識別済み
- クローズ

チケットを管理する方法

このシナリオでは、CAI でチケットを管理する手順を説明します。

1. チケットの検索 (P. 76)
2. トランザクションの不具合の識別 (P. 77)
3. チケットの直接 URL の取得 (P. 78)
4. チケットのトランザクションの表示 (P. 79)
5. チケット情報の編集 (P. 80)

チケットの検索

［Manage Tickets］ウィンドウから、チケットを検索して、テスト中に見つかった不具合を表示します。［Manage Tickets］ウィンドウには、CAI データベースのチケットを検索するための検索条件が含まれています。

以下の検索条件を使用してチケットを検索できます。

- レポータ - チケットを作成したユーザ
- ステータス - 新規、識別済み、またはクローズ
- 期間 - 分ごと、時間ごと、日ごと、およびすべての時間
- 詳細 - 開始日/終了日および開始時間/終了時間
- テキスト入力による検索

チケットの初期ステータスは「新規」です。 チケット情報の編集時に、ステータスを［識別済み］または［クローズ］に変更できます。 更新されたチケットは、［Home］ページの［New Ticket Alerts］ポートレットに表示されます。

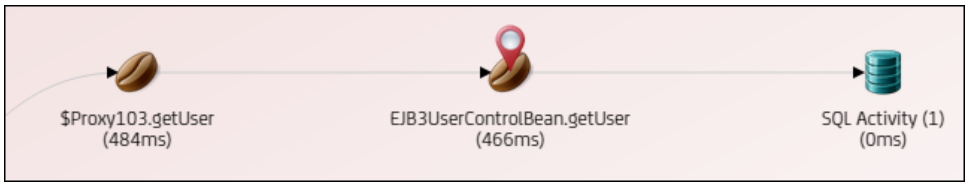
トランザクションの不具合の識別

チケットのトランザクションのフレームをマークして、不具合の原因を識別することができます。

「新規」のステータスのチケットのみ、識別できます。

チケットを識別する場合、フレームに赤いアイコンが付いており、パス グラフの背景が赤くなります。チケットは、［Home］ページの［New Ticket Alerts］ポートレットから削除されます。

以下の図は、パス グラフで識別されているフレームを示しています。



次の手順に従ってください:

1. 左側のナビゲーション メニューで［Application Insight］-［Manage Tickets］を選択します。

   ［Manage Tickets］ウィンドウが表示されます。

2. ［Actions］列のから「Graphical View」をクリックします。

   ［Transactions］ダイアログ ボックスが開き、チケットのトランザクション パスが視覚的な表現で表示されます。

3. フレームをマークするには、フレームを右クリックし、［Identify Problem (Update Ticket Status)］を選択します。

   ［Actions］列でステータスが「新規」から「識別済み」に変更されます。

4. フレームのマークを解除するには、フレームを右クリックし、［Unidentify Problem］を選択します。

   ［Actions］列でステータスが「識別済み」から「新規」に変更されます。

5. （オプション）［Home］ページに移動し、［リフレッシュ］ボタンをクリックして、［New Ticket Alerts］ポートレットを更新します。

チケットの直接 URL の取得

CAI では、既存のケースを直接示している URL を取得できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［Application Insight］-［Manage Tickets］を選択します。

   ［Manage Tickets］ウィンドウが表示されます。

2. チケットの  をクリックします。

   ケース URL が表示されます。

3. フィールドに表示される URL をコピーします。
4. この時点で、以下の方法により、チケットに直接移動できます。
   - Web ブラウザに URL を入力するか、 URL を強調表示し、URL を右クリックする。
   - リンクを強調表示し、右クリックして、［Go to http://<URL>］を選択する。

チケットのトランザクションの表示

パス グラフでは、各チケットの対応するトランザクションを表示できます。 パスをペイロード データと一緒に表示して、不具合の原因を特定することができます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［Application Insight］-［Manage Tickets］を選択します。

   ［Manage Tickets］ウィンドウが表示されます。

2. ［Actions］列の  をクリックします。

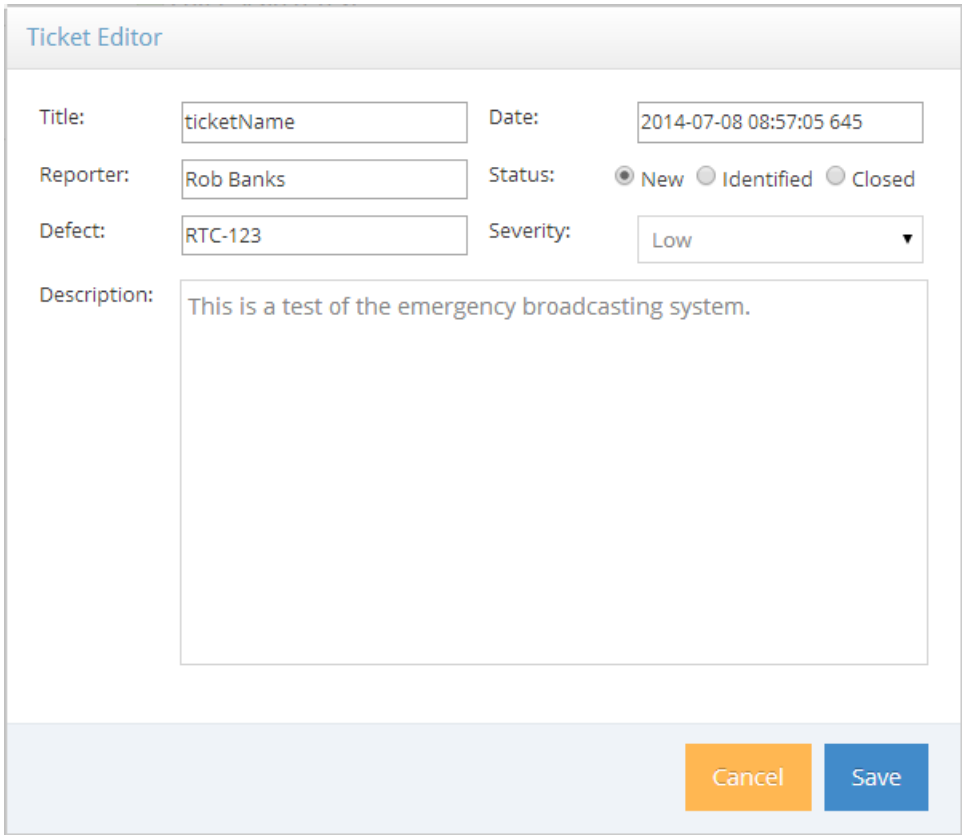
   トランザクションのパス グラフが表示されます。

3. トランザクションを確認します。
4. （オプション）フレームを右クリックし、以下のオプションから選択します。
   - Identify Problem (Update Ticket Status)
   - Generate All Artifacts
   - Generate Document
   - Unidentify Problem

チケット情報の編集

［Ticket Editor］から既存のチケットの情報を変更できます。

以下の図は、［Ticket Editor］ダイアログ ボックスを示しています。



次の手順に従ってください:

1. 左側のナビゲーション メニューで［Application Insight］-［Manage Tickets］を選択します。

   ［Manage Tickets］ペインにチケットのリストが表示されます。

2. 編集するチケットの［Actions］列で［Edit Ticket］をクリックします。

   ［Ticket Editor］ダイアログ ボックスが表示されます。

3. 適切なフィールドを編集して、［Save］をクリックします。

   ［Manage Tickets］ウィンドウの上部に、チケットが正常に更新されたことを示すメッセージが表示されます。

   注: ステータスを「New」から「Identified」または「Closed」に変更すると、チケットが［Home］ページの［New Tickets Alert］ポートレットから削除されます。

スクリーンショットの表示

CAI では、キャプチャ時に得られた、アプリケーションのスクリーンショットを表示できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで［Application Insight］-［Manage Tickets］を選択します。

   ［Manage Tickets］ウィンドウが表示されます。

2. ［Actions］列の［View Screen shot］をクリックします。

   スクリーンショットが表示されます。

3. スクリーンショットを閉じるには、［X］をクリックします。

第 6 章: 不具合の操作

CAI では、トランザクションの不具合を識別して分析し、パフォーマンスボトルネックを検出し、不具合レポートを作成することができます。

［Explore Defects］ウィンドウから、以下のことができます。

- 不具合のあるトランザクションの検索（P. 84）
- トランザクションへの例外、ログ メッセージ、応答時間パーセンテージによる注釈を付けと、注釈が付けられているトランザクションの表示（P. 85）
- 注釈が付けられたトランザクションのピン留め（P. 86）
- 繰り返されるパスを持つトランザクションのマージ（P. 87）
- アーティファクトの生成（P. 89）
- 不具合レポートの生成（P. 91）

このセクションには、以下のトピックが含まれています。

不具合のあるトランザクションの検索

［Explore Defects］ウィンドウで、検索およびフィルタ絞り込みにより、不具合を持つトランザクションを検索できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Explore Defects］を選択します。

   ［Explore Defects］ウィンドウが表示されます。

2. テキスト文字列によって検索するには、［Enter search text］フィールドに検索条件を入力し、［Search］をクリックします。
3. 期間によって検索するには、［1 Day］をクリックし、オプションを選択します。
4. 詳細な検索を実行するには、［1 Day］をクリックし、［Advanced］を選択します。
5. 検索条件を入力するか選択して、［Apply］をクリックします。

   ［Results］ペインに、検索条件に基づく不具合のリストが表示されます。

6. 検索をフィルタするには、［Refined By］ペインを使用します。

   以下のフィルタ オプションを使用できます。

   - Exec Time (ms)
   - Transaction ID
   - Point of Interest

不具合を表示するためのトランザクションの注釈付け

CAI では、違反のあるトランザクションを見つけ出し、それらをパスに表示することができます。

［Explore Defects］ウィンドウから、以下のものについてトランザクションに注釈を付けることができます。

- 例外

  例外違反を持つトランザクションを識別して表示することができます。

- ログ メッセージ

  エラーおよび警告のログ メッセージを識別して表示することができます。

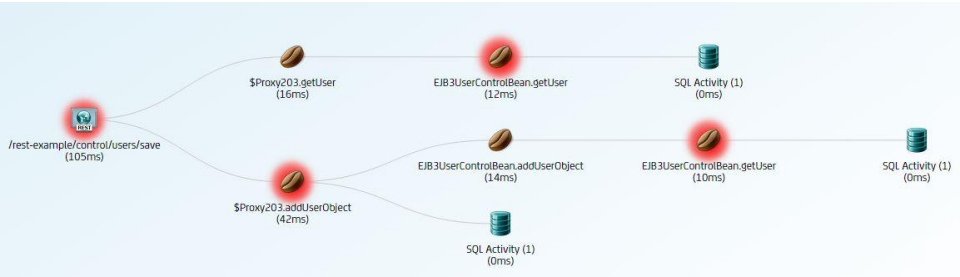
- 応答時間パーセンタイル

  最低パフォーマンスのパーセンタイルを設定し、それらをトランザクション パス内に表示することができます。

- 注釈を付けられたトランザクションのみの表示

  注釈を付けられたトランザクションのみを表示することができます。

以下の図は、注釈を付けられた例外を持つトランザクションの例を示しています。



警告を含むログ メッセージを持つトランザクションは黄色で表示され、例外、エラー ログ メッセージ、および応答時間違反の場合は赤色で表示されます。

［Explore Defects］ウィンドウで注釈を付けられ、ピン留めされたトランザクションは、［Home］ページの［Point of Interests］リストに表示されます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Explore Defects］を選択します。

   ［Explore Defects］ウィンドウに、トランザクションがグラフィカル ビューで表示されます。

2. ペインの上部で検索絞り込みを使用してトランザクションを検索します。
3. ［Annotation］をクリックし、1 つ以上のオプションを選択します。

   不具合を持つトランザクションが表示されます。

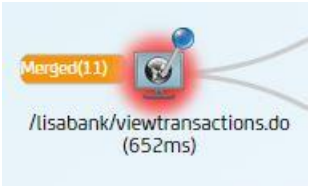
4. （オプション）トランザクションを選択して、不具合に関する詳細情報を表示します。

   トランザクション詳細ダイアログ ボックスが表示されます。

関心のある点に関する注釈付きトランザクションのピン留め

［Explore Defects］ウィンドウで不具合のあるトランザクションに注釈を付けた後、それらをピン留めして、それらを可視化するとともに、［Home］ページの［Points of Interest］リストに表示されるようにします。

以下の図は、ピン留めされた不具合のあるトランザクションを示しています。



次の手順に従ってください:

1. ［Explore Defects］ウィンドウで、違反のあるノードを右クリックして、［Pin Transaction］を選択します。
2. 説明を入力し、［OK］をクリックします。

   トランザクションに青色のピンが表示されます。 ツールヒントには、入力されたピンの説明が含まれています。
3. ［Home］ページに移動し、［Points of Interest］ポートレットの［Refresh］をクリックします。

   ピン留めされたトランザクションが［Points of Interest］リストに表示されます。

トランザクションのピン留めを解除する方法

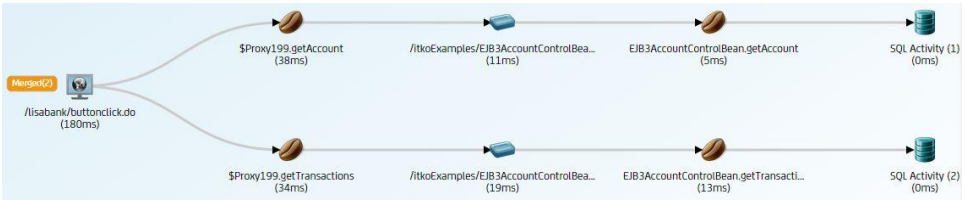
1. ［Explore Defects］ペインで、青色のピン留めされたノードを右クリックして、［Remove Pin］を選択します。

   ピンが削除されます。
2. ［Home］ページに移動し、［Points of Interest］ポートレットの［Refresh］をクリックします。

   ピン留めされたトランザクションが［Points of Interest］リストから削除されます。

繰り返されるパスを持つトランザクションの表示

繰り返されるパスを持つトランザクションをグラフィカル ビューで表示できます。



次の手順に従ってください:

1. ナビゲーション メニューから［Application Insight］-［Explore Defects］を選択します。

   ［Explore Defects］ウィンドウが表示されます。

2. 特定のトランザクションを検索するには、検索および絞り込みフィルタを使用します。

3. ［Repeated Paths］をクリックします。

   繰り返されるトランザクションを持つパスには、繰り返されるトランザクションの数が示されたオレンジ色のマージ バッジが表示されます。

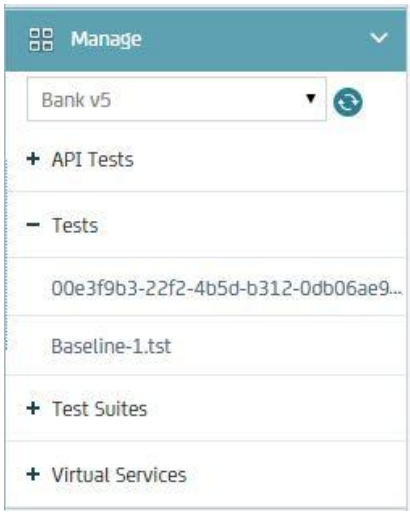
［Explore Defects］でのアーティファクトの生成

［Explore Defects］ウィンドウから、以下のアーティファクトを生成できます（該当する場合）。

- ベースライン（展開されたベースラインのみサポートされている）
- 仮想サービス

CAI でのベースラインおよび仮想サービスの生成の詳細については、「ベースラインの生成 (P. 93)」および「仮想サービスの作成 (P. 149)」を参照してください。

生成されたアーティファクトは、［Manage］メニュー オプションで選択されたプロジェクトに対して表示されます。 以下の図は、［Explore Defects］ウィンドウから Bank v5 プロジェクトで作成されたベースラインを示しています。



次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Explore Defects］を選択します。
2. ［Results］ペインで、フレームを右クリックし、［Generate All Artifacts］を選択します。

   ［Generate All Artifacts］ダイアログ ボックスが表示されます。
3. 変更するアーティファクトのタイプを選択し、［Generate］をクリックします。

   オプションを設定する場合は、［Generate All Assets］ダイアログ ボックスから［Advanced］をクリックします。
4. ［Select Project］ダイアログ ボックスで、ドロップダウン リストからプロジェクトを選択し、［Create］をクリックします。

   確認ダイアログが表示されます。
5. ［OK］をクリックします。

不具合レポートの生成

［Explore Defects］ウィンドウからの不具合レポートを生成できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューから［Application Insight］-［Explore Defects］を選択します。

   ［Explore Defects］ウィンドウが表示されます。

2. ［Search Text］フィールドを使用するか、［Refined by］フィルタ条件を使用するか、［Results］リストをスクロールすることによって、パスを検索します。

3. （オプション）［Annotate］をクリックし、メニューから注釈を付けるための 1 つ以上のオプションを選択します。

   違反を持つフレームは赤色で表示されます。

4. レポートの必要なトランザクションを右クリックし、［Generate Document］を選択します。

5. タイトルを入力して、［OK］をクリックします。

   そのレポートは、PDF ファイルとしてブラウザに表示されます。

6. ブラウザで［Save］をクリックして、レポートをダウンロードし、保存します。

7. ファイル名を入力し、［Save］をクリックします。

8. ［Print］をクリックして、レポートを印刷します。

第 7 章: ベースラインの作成

CAI では、CAI データベースのトランザクションからベースライン テスト ケースおよびスイートを作成できます。

トランザクションを生成する場合、適切なプロトコルのキャプチャ レベル (P. 27)が［全データ］に設定されていることを確認します。

カテゴリが「GUI」のトランザクション フレームのベースラインを作成することはできません。

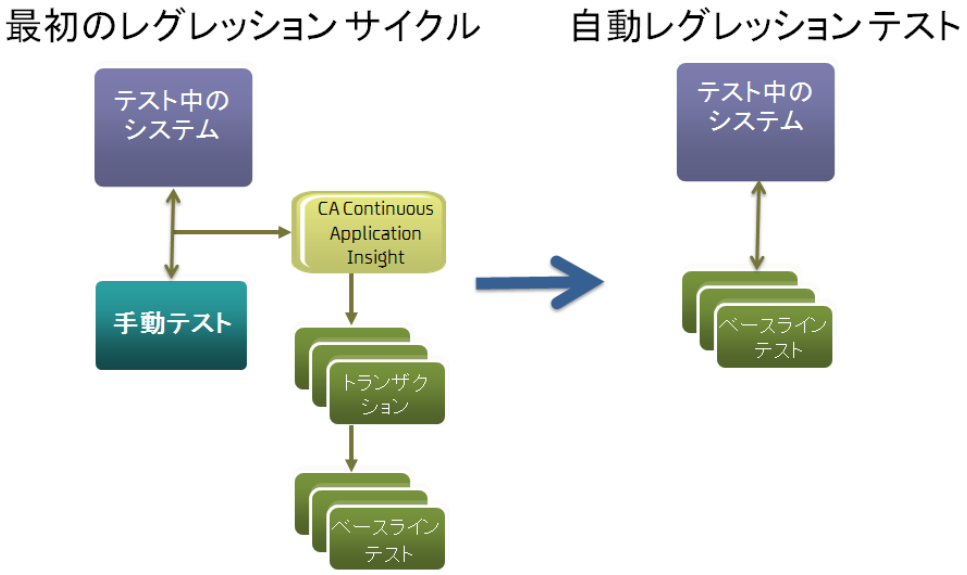
注: この機能には、個別のライセンスが必要です。

このセクションには、以下のトピックが含まれています。

ベースラインの概要

ベースラインは、エンタープライズ アプリケーションのテストの自動化を支援するように設計されています。

以下の図は、ほとんどのベースラインがどのように動作するかについての概要を示しています。



左の部分は、最初のレグレッション サイクルを示しています。 手動のテストがシステムで実行されます。 CA Continuous Application Insight はアクティビティをキャプチャし、ベースライン テスト ケースを生成するために使用されるトランザクションを作成します。

右の部分は、その後の自動レグレッション テストを示しています。 ベースライン テスト ケースは、必要に応じてテスト中のシステムに対して実行されます。

各ベースライン テスト ケースは、システムからの予期される応答に関する情報を含みます。 自動レグレッション サイクル中に、ベースライン テスト ケースは、予期される応答を実際の応答と比較します。 応答が一致しない場合、違いが強調表示されます。

ステートフル ベースライン

*ステートフル ベースライン*は、会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用されるベースライン テスト ケースです。 会話は、セッション内のトランザクションのセットです。

ほとんどの統合ベースラインには単一のテスト ステップが含まれますが、ステートフル ベースラインには通常複数のテスト ステップが含まれます。

ステートフル ベースラインは、DevTest ポータルまたは CAI コマンドライン ツールから作成できます。

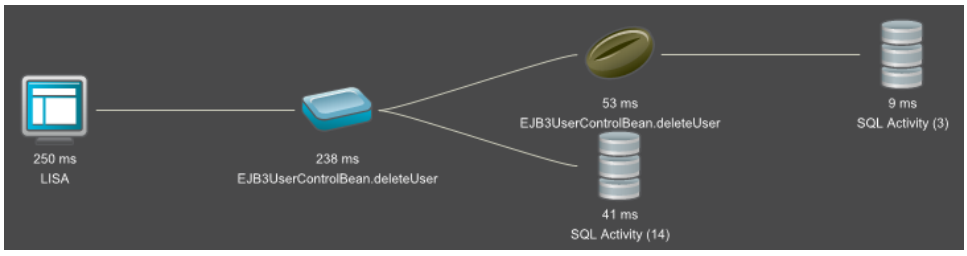
ステートフル ベースラインは以下のカテゴリのトランザクション フレームに対してサポートされています。

- EJB
- REST
- Web HTTP
- Web サービス

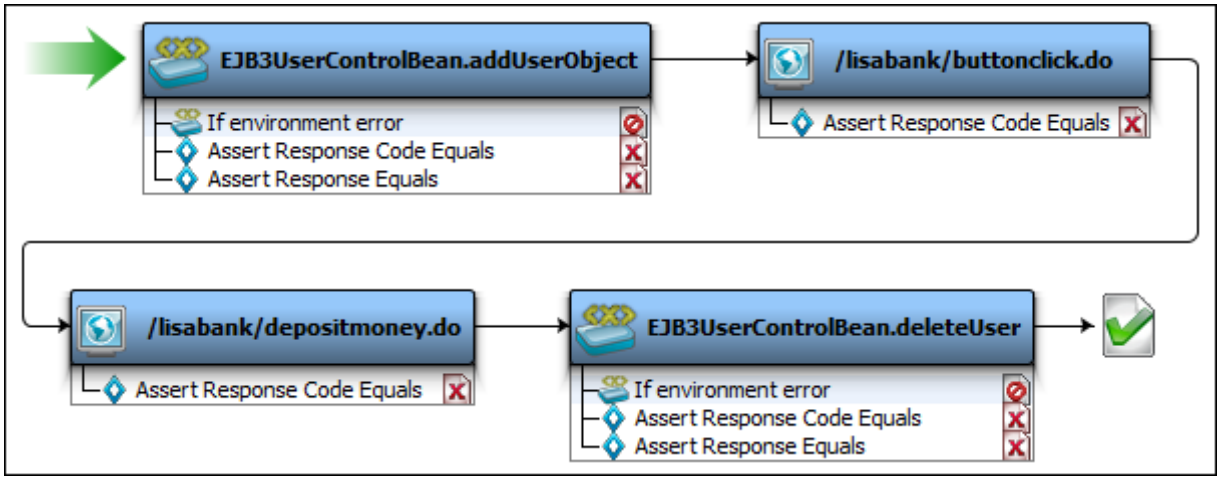
単一のトランザクション フレームをシェルフに追加すると、ステートフル ベースラインが自動的に作成されます。

マージされたトランザクション フレームをシェルフに追加する場合は、ステートフル ベースラインを作成できません。統合され、展開されたベースラインを作成することができます。

examples プロジェクトで **multi-tier-combo** テスト ケースを実行すると仮定します。複数のパスが生成されます。以下の図は、1 つのパスの主要な部分を示しています。 「発生したトランザクションをすべてシェルフに追加する」アクションが、Web サービス フレームに適用されています。



以下の図は、シェルフに追加されたフレームから生成されたステートフル ベースラインを示しています。



EJB ベースライン

このセクションには、以下のトピックが含まれます。

EJB ベースラインの生成

CAI データベースのトランザクションのセットから EJB ベースラインを生成するには、以下の手順に従います。

次の手順に従ってください:

1. 1 つ以上の EJB トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   ステートフル

   会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用される 1 つ以上のテスト ケースが含まれるベースラインを作成します。 このオプションは、マージされたトランザクションには使用できません。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. テスト ケースに含まれるステップのタイプを指定します。

   アプリケーション テスト ステップの使用

   テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

   トランザクション フレーム ステップの使用

   テスト ケースには、トランザクション フレームの実行ステップが含まれます。

8. （オプション）マジック デートを使用するようにベースラインを設定します。

   テスト ケースにマジック デートを適用

   ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
10. 以下のフィールドが編集可能な場合は、これらのフィールドを設定します。

    JNDI ファクトリ

    EJB ベースライン テストを生成するときに使用する JNDI ファクトリ クラス。

    JNDI URL

    EJB ベースライン テストを生成するときに使用する JNDI URL。

    セキュリティ プリンシパル

    EJB ベースライン テストを生成するときに使用する JNDI ユーザ。

    セキュリティ認証情報

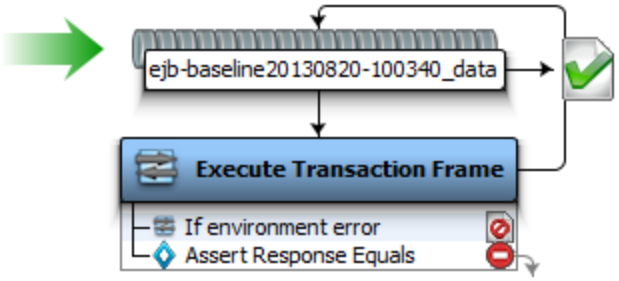
    EJB ベースライン テストを生成するときに使用する JNDI パスワード。

11. ［Create］をクリックします。
12. ベースラインが作成されるプロジェクトを選択します。
13. ［Create］をクリックします。

EJB ベースライン テスト ケース

統合 EJB ベースライン テスト ケースには、大容量データのデータ セットから読み取るトランザクション フレームの実行ステップが含まれます。

以下の図は、統合 EJB ベースライン テスト ケースの例を示しています。



データ セットには以下の情報が含まれます。

- エージェント名
- トランザクション フレーム
- 予期される応答

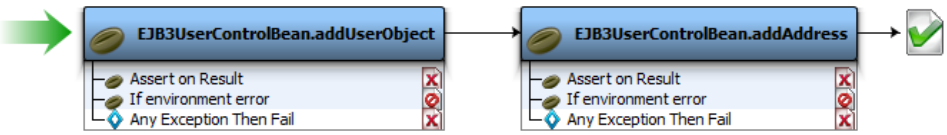
デフォルトでは、データ セットはグローバルではなく、ローカルです。

トランザクション フレームの実行ステップには、「**応答の一致を確認**」という名前のグラフィカル XML 比較アサーションが含まれます。 このアサーションは、実際の応答が予期される応答に一致することを確認します。

ステートフル EJB ベースライン テスト ケースには、1 つ以上の Enterprise JavaBean 実行ステップが含まれます。 各ステップには以下のアサーションが含まれます。

- 「**結果に対するアサート**」という名前の埋め込みのアサーション
- 「**環境エラーの場合**」という名前の埋め込みのアサーション
- 「**例外の場合は失敗**」という名前の呼び出し例外でのアサート アサーション

以下の図は、ステートフル EJB ベースライン テスト ケースの例を示しています。



テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

EJB ベースライン スイート

EJB ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

EJB ベースライン スイートのテスト ケースには、Enterprise JavaBean 実行ステップまたはトランザクション フレームの実行ステップが含まれます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

JMS ベースライン

JMS ベースラインは、JMS 接続ファクトリを取得するために JNDI が使用される設定に対してサポートされています。

また、JMS ベースラインは、JMS 接続ファクトリを取得するために IBM WebSphere MQ、Java CAPS、SonicMQ、または TIBCO から直接 API が使用される設定に対してもサポートされています。

注: サービスまたはクライアントが JMS メッセージ オブジェクトを変更せずに再利用または転送する場合、CA Continuous Application Insight の動作は定義されません。

このセクションには、以下のトピックが含まれます。

JMS ベースラインの生成

CAI データベースのトランザクションのセットから JMS ベースラインを生成するには、以下の手順に従います。

次の手順に従ってください:

1. JMS トランザクション フレーム (P. 104) をシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. テスト ケースに含まれるステップのタイプを指定します。

   アプリケーション テスト ステップの使用

   テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

   トランザクション フレーム ステップの使用

   テスト ケースには、トランザクション フレームの実行ステップが含まれます。
8. （オプション）マジック デートを使用するようにベースラインを設定します。

   テスト ケースにマジック デートを適用

ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテストケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

JMS ベースライン トランザクション フレーム

JMS ベースラインを生成する手順には、JMS トランザクション フレームをシェルフに追加する手順が含まれます。

1 つの要求および 1 つの応答を持ったシナリオの場合、シェルフに追加するトランザクション フレームを選択するパス グラフには、4 つのトランザクション フレームが含まれます。 これらのフレームは以下のアクティビティを表します。

- クライアントは要求メッセージを作成します。
- サービスは要求メッセージを処理します。
- サービスは応答メッセージを作成します。
- クライアントは応答メッセージを処理します。

要求メッセージを作成するクライアントを表すフレームを選択します。このルールの例外については、重複したノードについての以下のセクションを参照してください。

CAI は、選択したフレームと同じ名前が含まれるトランザクションを検索します。 ベースラインには、同じキューに対してメッセージを送受信するすべてのトランザクションが含まれます。

重複したノード

状況によっては、パス グラフに同じ生成/処理操作を表す 2 つの隣接したフレームが含まれる場合があります。 隣接したフレームの名前は両方とも、**send:** または **recv**: のいずれかで始まります。

パス グラフ内の最初の 2 つのフレームが **send:** で始まる場合、1 番目のフレームを選択します。

パス グラフ内の最初の 2 つのフレームが **recv:** で始まる場合、2 番目のフレームを選択します。

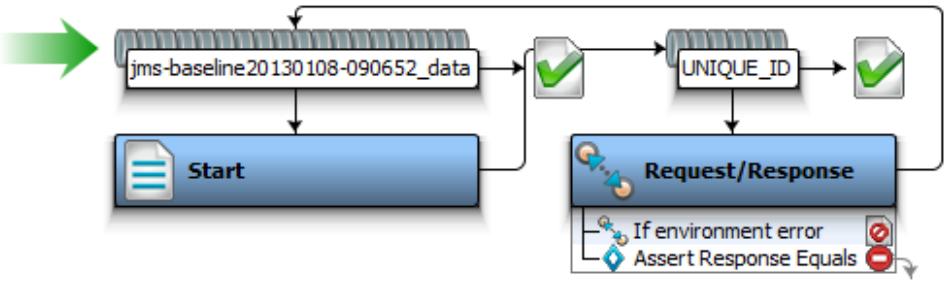
JMS ベースライン テスト ケース

ほとんどの単一要求および単一応答のシナリオ用のテスト ケースには、2 つのステップが含まれます。

- 大容量データのデータ セットから読み取る何も実行しないステップ
- 要求と応答を組み合わせたメッセージング ステップ。メッセージング ステップは、JMS メッセージング（JNDI）、IBM WebSphere MQ、JCAPS メッセージング（ネイティブ）、SonicMQ メッセージング（ネイティブ）、または TIBCO ダイレクト JMS のいずれかにすることができます。WebSphere MQ ステップでは、クライアント モードは JMS に設定されます。

また、テスト ケースには、相関 ID を生成するための一意コード ジェネレータ データ セットを含めることができます。

以下の図は、テスト ケースの例を示しています。 最初のステップは、何も実行しないステップです。 2 番目のステップは、**Request/Response**（要求/応答）という名前のメッセージング ステップです。

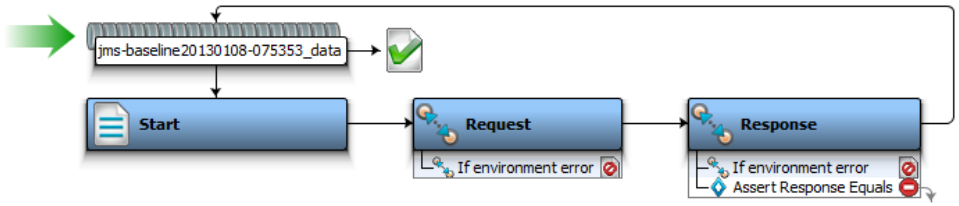


データ セットには、ベースライン内のトランザクションごとに異なる情報が含まれます。 この情報には、要求ペイロード、応答ペイロード、および JMS メッセージ プロパティが含まれます。 デフォルトでは、データ セットはグローバルではなく、ローカルです。

メッセージング ステップには、「**応答の一致を確認**」という名前のグラフィカル XML 比較アサーションが含まれます。ベースライン テスト ケースを実行すると、このアサーションは実際の応答が予期される応答に一致することを確認します。 応答が一致しない場合、テストはエラーを生成します。

その他のシナリオでは、要求と応答を個別のステップに分割します。

以下の図は、テスト ケースの例を示しています。 最初のステップは、何も実行しないステップです。 2 番目のステップは、**要求**という名前のメッセージング ステップです。 3 番目のステップは、**応答**という名前のメッセージング ステップです。



設定プロパティは、送信先情報、接続情報および XML 差分オプションに対して生成されます。 クライアントとサービスが別の JNDI 接続設定を使用する場合、接続情報に対してプロパティのセットが 2 つ生成されます。

テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

JMS ベースライン スイート

JMS ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

JMS ベースライン スイートのテスト ケースは、JMS ベースライン テスト ケースに似ています。 ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはステップに格納されます。

JMS ベースライン テスト ケースのように、設定プロパティが生成されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

REST ベースライン

このセクションには、以下のトピックが含まれます。

REST ベースラインの生成

CAI データベースのトランザクションのセットから REST ベースラインを生成するには、以下の手順に従います。

次の手順に従ってください:

1. 1 つ以上の REST トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   ステートフル

   会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用される 1 つ以上のテスト ケースが含まれるベースラインを作成します。 このオプションは、マージされたトランザクションには使用できません。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. テスト ケースに含まれるステップのタイプを指定します。

   アプリケーション テスト ステップの使用

   テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

   トランザクション フレーム ステップの使用

   テスト ケースには、トランザクション フレームの実行ステップが含まれます。

8. （オプション）マジック デートを使用するようにベースラインを設定します。

   テスト ケースにマジック デートを適用

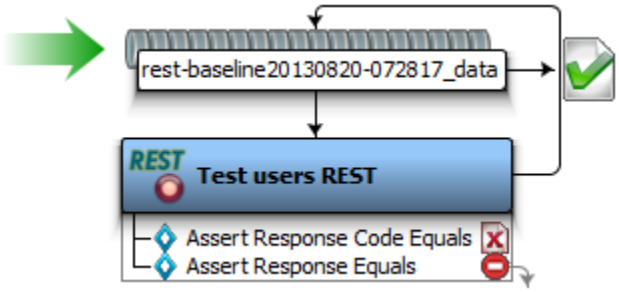
   ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

REST ベースライン テスト ケース

統合 REST ベースライン テスト ケースには、大容量データのデータ セットから読み取るテスト ステップがあります。 テスト ステップは、REST ステップまたはトランザクション フレームの実行ステップのいずれかです。

以下の図は、統合 REST ベースライン テスト ケースの例を示しています。テスト ケースには REST ステップがあります。



テスト ケースに REST ステップがある場合、データ セットには URL、要求のボディ、予期される応答、応答コードなどの情報が含まれます。

テスト ケースにトランザクション フレームの実行ステップがある場合、データ セットにはエージェント名、トランザクション フレーム、および予期される応答が含まれます。

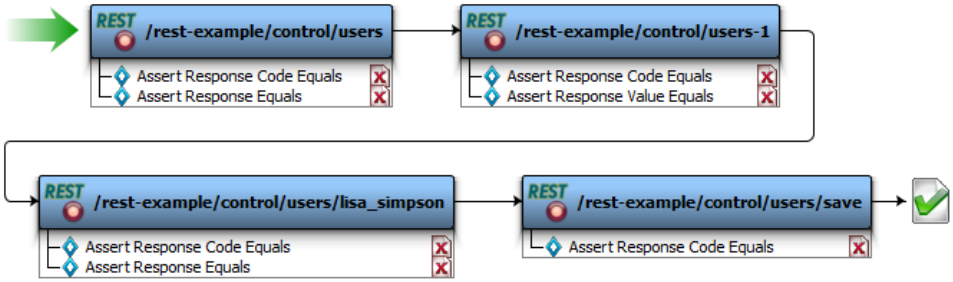
デフォルトでは、データ セットはグローバルではなく、ローカルです。

テスト ステップには、以下の 1 つ以上のアサーションが含まれます。

- **応答コードの一致を確認。** このアサーションは、実際の応答コードが予期される応答コードに一致することを確認します。
- **応答の一致を確認。** このアサーションは、実際の応答が予期される応答に一致することを確認します。 アサーションのタイプは、応答の形式が XML、JSON、または RAW テキストであるかどうかによって異なります。

ステートフル REST ベースライン テスト ケースには、1 つ以上の REST ステップがあります。各ステップには、**応答コードの一致を確認**アサーションが含まれます。 各ステップには、**応答の一致を確認**アサーションまたは**応答の値の一致を確認**アサーションが含まれる場合があります。

以下の図は、ステートフル REST ベースライン テスト ケースの例を示しています。



テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

REST ベースライン スイート

REST ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

REST ベースライン スイートのテスト ケースは、REST ベースライン テスト ケースに似ています。 ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはテスト ステップに格納されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

HTTP クライアント コール

アプリケーションが POST、GET、PUT、および DELETE に対して Apache HttpClient API を使用して HTTP クライアント コールを行う場合、CAI はトランザクション フレームに REST カテゴリを割り当てます。

代わりに、POST および GET コールを強制的に HTTP として分類するには、**lisa.agent.rest.enabled** プロパティを **rules.xml** ファイルの **agent** エレメントに追加します。 値を false に設定します。

```
<property key="lisa.agent.rest.enabled" value="false"/>
```

rules.xml ファイルの更新時にレジストリが実行されている場合は、レジストリを再起動します。

PUT および DELETE コールを強制的に HTTP として分類することはサポートされていません。

TIBCO BusinessWorks ベースライン

このセクションには、以下のトピックが含まれます。

TIBCO BusinessWorks ベースラインの生成

CAI データベースのトランザクションのセットから TIBCO ActiveMatrix BusinessWorks ベースラインを生成するには、以下の手順に従います。

次の手順に従ってください:

1. TIBCO ActiveMatrix BusinessWorks トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. （オプション）マジック デートを使用するようにベースラインを設定します。

   テスト ケースにマジック デートを適用

   ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。
8. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
9. ［Create］をクリックします。

10. ベースラインが作成されるプロジェクトを選択します。
11. ［Create］をクリックします。

TIBCO BusinessWorks ベースライン テスト ケース

TIBCO ActiveMatrix BusinessWorks ベースライン テスト ケースには、大容量データのデータ セットから読み取るトランザクション フレームの実行ステップが含まれます。

データ セットには以下の情報が含まれます。

- エージェント名
- トランザクション フレーム
- 予期される応答

デフォルトでは、データ セットはグローバルではなく、ローカルです。

トランザクション フレームの実行ステップには、「**応答の値の一致を確認**」という名前のグラフィカル XML 比較アサーションが含まれます。ベースライン テスト ケースを実行すると、このアサーションは実際の応答が予期される応答に一致することを確認します。 応答が一致しない場合、テストはエラーを生成します。

テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

TIBCO BusinessWorks ベースライン スイート

TIBCO ActiveMatrix BusinessWorks ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

TIBCO ActiveMatrix BusinessWorks ベースライン スイートのテスト ケースには、トランザクション フレームの実行ステップがあります。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

TIBCO Enterprise Message Service ベースライン

ベースライン機能は、TIBCO Enterprise Message Service （EMS）に対してサポートされています。

手順は、JMS ベースラインの手順と同じです。 ただし、開始する前に、**LISA_HOME¥lib** ディレクトリに **tibjms.jar** ファイルをコピーします。

Web HTTP ベースライン

このセクションには、以下のトピックが含まれます。

Web HTTP ベースラインの生成

CAI データベースのトランザクションのセットから Web HTTP ベースラインを生成するには、以下の手順に従います。

署名された Web HTTP トランザクションがサポートされています。ただし、CAI は、SSL 情報をベースラインに保存しません。SSL 情報が必要な場合は、生成されたベースラインの以下のフィールドを設定します。

- SSL キーストア ファイル
- SSL キーストア パスワード
- SSL キー エイリアス
- SSL キー パスワード

これらのフィールドは、HTTP/HTML 要求ステップにあります。

次の手順に従ってください:

1. 1 つ以上の Web HTTP トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   ステートフル

   会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用される 1 つ以上のテスト ケースが含まれるベースラインを作成します。 このオプションは、マージされたトランザクションには使用できません。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。

7. テスト ケースに含まれるステップのタイプを指定します。

アプリケーション テスト ステップの使用

テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

トランザクション フレーム ステップの使用

テスト ケースには、トランザクション フレームの実行ステップが含まれます。

8. （オプション）マジック デートを使用するようにベースラインを設定します。

テスト ケースにマジック デートを適用

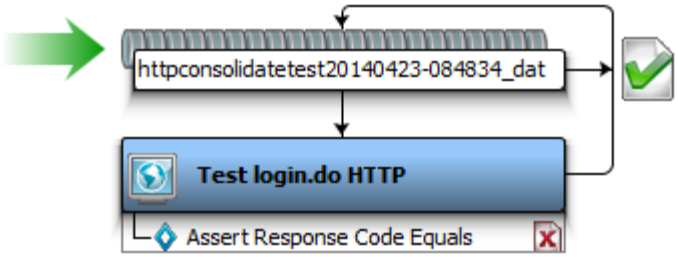
ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

Web HTTP ベースライン テスト ケース

統合 Web HTTP ベースライン テスト ケースには、大容量データのデータセットから読み取るテスト ステップがあります。 テスト ステップは HTTP/HTML 要求ステップです。

以下の図は、統合 Web HTTP ベースライン テスト ケースの例を示しています。 このテスト ケースには、HTTP/HTML 要求ステップがあります。



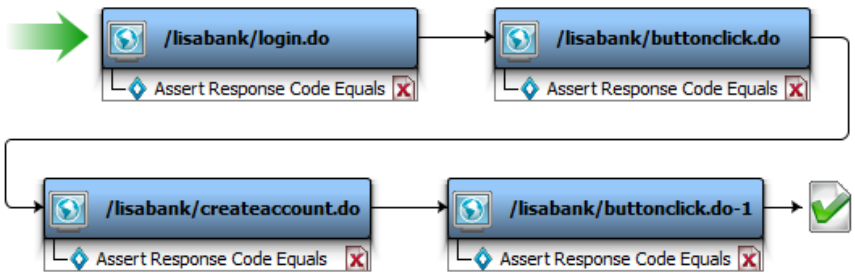
データ セットには、URL、要求のボディ、予期される応答、応答コードなどの情報が含まれます。

デフォルトでは、データ セットはグローバルではなく、ローカルです。

テスト ステップには、**応答コードの一致を確認**アサーションが含まれます。 このアサーションは、実際の応答コードが予期される応答コードに一致することを確認します。

ステートフル Web HTTP ベースライン テスト ケースには、1 つ以上の HTTP/HTML 要求ステップが含まれます。 各ステップには、**応答コードの一致を確認**アサーションが含まれます。

以下の図は、ステートフル Web HTTP ベースライン テスト ケースの例を示しています。



テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

Web HTTP ベースライン スイート

Web HTTP ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

Web HTTP ベースライン スイートのテスト ケースは、Web HTTP ベースライン テスト ケースに類似しています。 ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはテスト ステップに格納されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

HTTP クライアント コール

アプリケーションが POST、GET、PUT、および DELETE に対して Apache HttpClient API を使用して HTTP クライアント コールを行う場合、CAI はトランザクション フレームに REST カテゴリを割り当てます。

代わりに、POST および GET コールを強制的に HTTP として分類するには、**lisa.agent.rest.enabled** プロパティを **rules.xml** ファイルの **agent** エレメントに追加します。 値を false に設定します。

```
<property key="lisa.agent.rest.enabled" value="false"/>
```

rules.xml ファイルの更新時にレジストリが実行されている場合は、レジストリを再起動します。

PUT および DELETE コールを強制的に HTTP として分類することはサポートされていません。

Web サービス ベースライン

このセクションには、以下のトピックが含まれます。

Web サービス ベースラインの生成

CAI データベースのトランザクションのセットから Web サービス ベースラインを生成するには、以下の手順に従います。

添付ファイル (P. 124)を持つ Web サービスはサポートされています。

署名付き SOAP メッセージは、IBM WebSphere Application Server 8.5 以外のすべてのプラットフォーム上でサポートされています。 ただし、CAI は、SSL 情報をベースラインに保存しません。 SSL 情報が必要な場合は、生成されたベースラインの以下のフィールドを設定します。

- SSL キーストア ファイル
- SSL キーストア パスワード
- SSL キー エイリアス
- SSL キー パスワード

これらのフィールドは、Web サービス実行（XML）ステップにあります。

この機能は、トランスポート レベルのセキュリティ（ポイント ツー ポイント セキュリティ モデル）をサポートしています。 この機能は、署名付きメッセージ レベルのセキュリティ (エンド ツー エンド セキュリティ モデル）をサポートしていません。

次の手順に従ってください:

1. 1 つ以上の Web サービス トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   ステートフル

   会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用される 1 つ以上のテスト ケースが含まれるベースラインを作成します。 このオプションは、マージされたトランザクションには使用できません。

統合

単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

展開

テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。

7. テスト ケースに含まれるステップのタイプを指定します。

アプリケーション テスト ステップの使用

テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

トランザクション フレーム ステップの使用

テスト ケースには、トランザクション フレームの実行ステップが含まれます。

8. （オプション）マジック デートを使用するようにベースラインを設定します。

テスト ケースにマジック デートを適用

ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

添付ファイルを持つ Web サービス

以下の添付ファイル タイプを持つ Web サービスのベースラインを生成できます。

- MIME
- DIME
- XOP
- MTOM

CAI は、Web サービスに複数の添付ファイルがある場合に添付ファイルが同じタイプであることを前提としています。

［Use Transaction Frame Step］オプションは、添付ファイルを持つ Web サービスに対してサポートされていません。

添付ファイルを持つ送信 Web サービス コールは、サポートされていません。

ベースラインが作成される場合、添付ファイル コンテンツはフレームで保存されます。 タイプは［**Base64 Encoded**］です。

フレームが MTOM 添付ファイル トランザクションから生成される場合、ベースラインの添付ファイル タイプは XOP です。

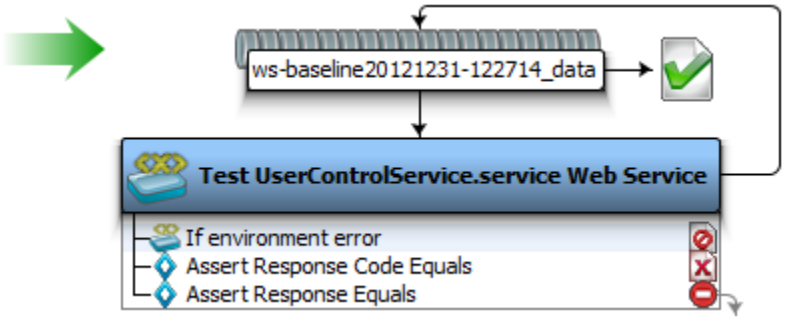
デフォルトでは、添付ファイルの最大サイズは 10 MB です。 添付ファイルがこれより大きい場合は、トランザクション フレームは切り捨てられたものとしてマークされ、添付ファイル コンテンツはフレームに保存されません。

デフォルトの最大サイズを変更するには、［Agenats］ウィンドウ (P. 19)に移動します。エージェントまたはオプションを選択します。［Settings］タブをクリックし、［Transactions］カテゴリを選択します。 ［Max Attachment Size］プロパティを設定します。 この値は、メガバイト単位ではなく、キロバイト単位の数値で設定します。 その後、［Save］をクリックします。

Web サービス ベースライン テスト ケース

統合 Web サービス ベースライン テスト ケースには、大容量データのデータ セットから読み取るテスト ステップがあります。 テスト ステップは、Web サービス実行（XML）ステップまたはトランザクション フレームの実行ステップのいずれかです。

以下の図は、統合 Web サービス ベースライン テスト ケースの例を示しています。 テスト ケースには Web サービス実行（XML）ステップがあります。



テスト ケースに Web サービス実行（XML）ステップがある場合、データ セットには URL、要求のボディ、予期される応答、応答コードなどの情報が含まれます。データ セットには、Web サービス添付ファイルからのデータを含めることもできます。

テスト ケースにトランザクション フレームの実行ステップがある場合、データ セットにはエージェント名、トランザクション フレーム、および予期される応答が含まれます。

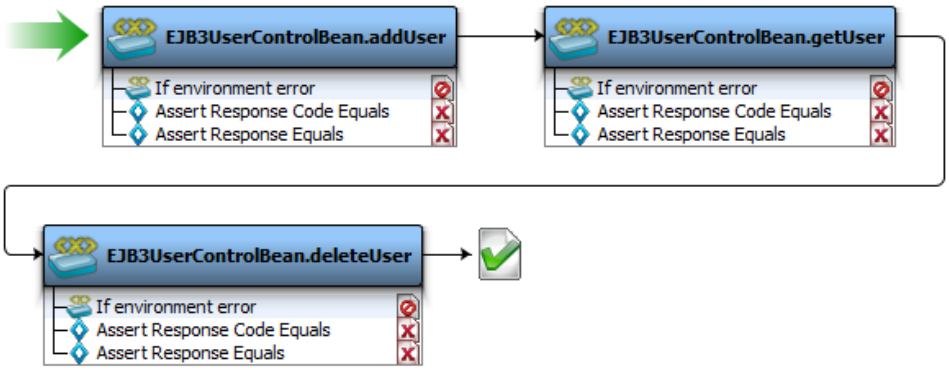
デフォルトでは、データ セットはグローバルではなく、ローカルです。

テスト ステップには、以下の 1 つ以上のアサーションが含まれます。

- **応答コードの一致を確認。** このアサーションは、実際の応答コードが予期される応答コードに一致することを確認します。
- **応答の一致を確認。** このアサーションは、実際の応答が予期される応答に一致することを確認します。 アサーションのタイプは、応答の形式が XML、JSON、または RAW テキストであるかどうかによって異なります。

ステートフル Web サービス ベースライン テスト ケースには、1 つ以上の Web サービス実行 (XML) ステップがあります。各ステップには、**応答コードの一致を確認**アサーションおよび**応答の一致を確認**アサーションが含まれます。

以下の図は、ステートフル Web サービス ベースライン テスト ケースの例を示しています。



テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

Web サービス ベースライン スイート

Web サービス ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

Web サービス ベースライン スイートのテスト ケースは、Web サービス ベースライン テスト ケースに似ています。ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはテスト ステップに格納されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

webMethods ベースライン

このセクションには、以下のトピックが含まれます。

webMethods ベースラインの生成

CAI データベースのトランザクションのセットから webMethods Integration Server ベースラインを生成するには、以下の手順に従います。

webMethods Integration Server には、フロー サービスおよびパイプラインの概念が含まれます。 フロー サービスでは、サービスのグループをカプセル化し、それらの間のデータのフローを管理できます。 パイプラインは、フロー サービスに対する入力値および出力値が含まれるデータ構造です。 フロー サービスにステップを追加して、サービスの呼び出し、パイプラインのデータの変更などのアクションを実行できます。

フロー サービスはベースライン化される単位です。

次の手順に従ってください:

1. webMethods Integration Server トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   ステートフル

   会話またはセッションのトランザクションの 1 つではなく、会話またはセッション全体に適用される 1 つ以上のテスト ケースが含まれるベースラインを作成します。 このオプションは、マージされたトランザクションには使用できません。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. テスト ケースに含まれるステップのタイプを指定します。

アプリケーション テスト ステップの使用

テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

トランザクション フレーム ステップの使用

テスト ケースには、トランザクション フレームの実行ステップが含まれます。

8. （オプション）マジック デートを使用するようにベースラインを設定します。

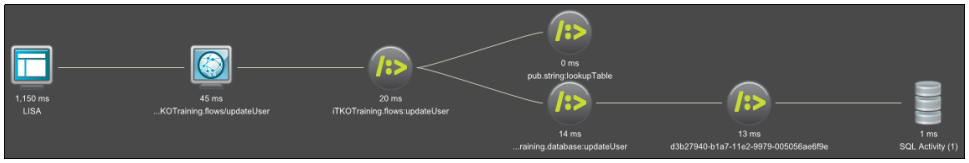
テスト ケースにマジック デートを適用

ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

例： ユーザの更新

以下の図は、webMethods Integration Server フレームが含まれるパス グラフを示しています。 この例のフロー サービスには、**updateUser** という名前が付けられています。 目的はデータベースにユーザの更新情報を送信することです。



注: webMethods Integration Server フレームのいずれかを選択すると、トランザクション詳細ダイアログ ボックスでパイプライン データを表示できます。

パス グラフに複数の webMethods Integration Server フレームが含まれる場合、ベースライン化するためにはフレームを 1 つ選択する必要があります。通常、エントリ ポイントを選択します。 この例では、左端の webMethods Integration Server フレームを選択します。

webMethods ベースライン テスト ケース

webMethods ベースライン テスト ケースには、大容量データのデータ セットから読み取るテスト ステップがあります。 テスト ステップは、webMethods Integration Server サービス ステップまたはトランザクション フレームの実行ステップのいずれかです。

テスト ケースに webMethods Integration Server サービス ステップがある場合、データ セットには URL、要求のボディ、予期される応答、応答コードなどの情報が含まれます。

テスト ケースにトランザクション フレームの実行ステップがある場合、データ セットにはエージェント名、トランザクション フレーム、および予期される応答が含まれます。

デフォルトでは、データ セットはグローバルではなく、ローカルです。

テスト ステップには、以下の 1 つまたは両方のアサーションが含まれます。

- **応答コードの一致を確認**。 ベースライン テスト ケースを実行すると、このアサーションは実際の応答コードが予期される応答コードに一致することを確認します。 コードが一致しない場合、テストはエラーを生成します。
- **応答の一致を確認**。 ベースライン テスト ケースを実行すると、このアサーションは実際の応答の値が予期される応答の値に一致することを確認します。 値が一致しない場合、テストはエラーを生成します。

テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

webMethods ベースライン スイート

webMethods ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

webMethods ベースライン スイートのテスト ケースは、webMethods ベースライン テスト ケースに似ています。 ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはテスト ステップに格納されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

WebSphere MQ ベースライン

このセクションには、以下のトピックが含まれます。

WebSphere MQ ベースラインの生成

CAI データベースのトランザクションのセットから WebSphere MQ ベースラインを生成するには、以下の手順に従います。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

次の手順に従ってください:

1. 使用する WebSphere MQ シナリオ (P. 135) を決定し、必要に応じて DevTest Java エージェントを有効にします。
2. （オプション） **rules.xml** ファイル内の WebSphere MQ プロパティ (P. 136) を設定します。
3. WebSphere MQ トランザクション フレームをシェルフに追加します。シェルフに追加するトランザクション フレームを選択するパス グラフには、複数の WebSphere MQ トランザクション フレームが含まれます。必ず、最初の *WebSphere MQ トランザクション フレーム*を選択してください。
4. シェルフを開きます。
5. ［ベースラインの作成］をクリックします。
6. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
7. 下向きの矢印をクリックします。
8. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
9. テスト ケースに含まれるステップのタイプを指定します。

   アプリケーション テスト ステップの使用

テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

トランザクション フレーム ステップの使用

テスト ケースには、トランザクション フレームの実行ステップが含まれます。

10. （オプション）マジック デートを使用するようにベースラインを設定します。

テスト ケースにマジック デートを適用

ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

11. （オプション）ベースラインのテンプレートとして使用されるテスト ケースを指定します。
12. ［作成］をクリックします。
13. ベースラインが作成されるプロジェクトを選択します。
14. ［作成］をクリックします。

WebSphere MQ ベースライン シナリオ

WebSphere MQ を使用する場合、クライアント、サーバ、またはクライアントとサーバの両方で DevTest Java エージェントを有効にできます。パス グラフの内容はシナリオによって異なります。

注: DevTest は、エージェントが有効なクライアントのように動作する場合があります。 この動作は、テスト ケースが IBM WebSphere MQ ステップを使用してサービスを呼び出すときに発生します。

シナリオ 1： クライアントで有効、サービスでは有効でない

クライアントではエージェントが有効で、サービスではエージェントが有効ではありません。 このシナリオは、エージェントを有効にできない Windows またはメインフレーム ベースのサービスに対して実行される Java ベースのクライアントをシミュレートします。

DevTest ポータルでトランザクションの詳細を表示すると、パス グラフには 2 つのノードが含まれます。

最初のノードは要求を表します。メッセージ ボディを表示するには、ノードを選択し、［要求］タブをクリックします。

2 番目のノードは応答を表します。 メッセージ ボディを表示するには、ノードを選択し、［応答］タブをクリックします。

シナリオ 2： クライアントでは有効でない、サービスで有効

クライアントではエージェントが有効ではなく、サービスではエージェントが有効です。このシナリオは、Java ベースのサービスに対して実行される、エージェントを有効にできない Windows またはメインフレーム ベースのクライアントをシミュレートします。

DevTest ポータル内の最初のトランザクションは準備トランザクションです。 サービス側のエージェントは、それがサービス側で実行されていることを検出し、それ自体を初期化します。

後のトランザクションはシナリオ 1 と同じです。 パス グラフには要求ノードおよび応答ノードが含まれます。

シナリオ 3： クライアントで有効、サービスで有効

クライアントとサービスの両方でエージェントが有効です。 このシナリオは、Java ベースのサービスに対して実行される Java ベースのクライアントをシミュレートします。

DevTest ポータル内の最初のトランザクションは準備トランザクションです。 サービス側のエージェントは、それがサービス側で実行されていることを検出し、それ自体を初期化します。

後のトランザクションでは、パス グラフに 4 つのノードが含まれます。

- 最初のノードは、要求のクライアント PUT を表します。
- 2 番目のノードは、要求のサービス GET を表します。
- 3 番目のノードは、応答のサービス PUT を表します。
- 4 番目のノードは、応答のクライアント GET を表します。

WebSphere MQ ベースライン プロパティ

DevTest Java エージェントの設定ファイルの名前は、**rules.xml** です。

注: このファイルの詳細については、「*エージェント*」を参照してください。

WebSphere MQ ベースラインを作成する前に、**rules.xml** ファイルに **QUEUE_REQUEST_MATCHES** および **QUEUE_RESPONSE_MATCHES** プロパティを追加できます。 これらのプロパティは、どのキューが要求用か、どのキューが応答用かを示します。

QUEUE_REQUEST_MATCHES および **QUEUE_RESPONSE_MATCHES** プロパティはオプションです。 これらは、メッセージが要求なのか応答なのかをエージェントが独力で判断できない場合にのみ必要です。

これらのプロパティを含める場合、プロパティ名の後にコロンおよび実際のキュー名を追加する必要があります。 各プロパティの値は、ピリオドの後にアスタリスクが続きます。 以下に例を示します。

```
property key=QUEUE_REQUEST_MATCHES:ORDERS.REQUEST value=.*
property key=QUEUE_RESPONSE_MATCHES:ORDERS.RESPONSE value=.*
```

WebSphere MQ ベースライン テスト ケース

WebSphere MQ ベースライン テスト ケースには、以下のステップがあります。

- 大容量データのデータ セットから読み取る何も実行しないステップ
- 要求を送信する WebSphere MQ ステップ
- 応答を受信する WebSphere MQ ステップ

データ セットには、ベースライン内のトランザクションごとに異なる情報が含まれます。 この情報には、要求ペイロードおよび応答ペイロードが含まれます。

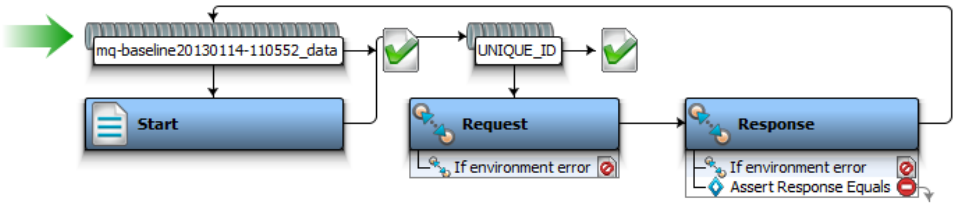
WebSphere MQ ステップでは、クライアント モードはネイティブ クライアントに設定されます。

最初の WebSphere MQ ステップは、データ セットからのデータを使用して要求を送信します。

2 番目の WebSphere MQ ステップは、応答を受信します。 このステップには、「**応答の一致を確認**」という名前のグラフィカル XML 比較アサーションが含まれます。 ベースライン テスト ケースを実行すると、このアサーションは実際の応答が予期される応答に一致することを確認します。 応答が一致しない場合、テストはエラーを生成します。

また、テスト ケースには、メッセージ/相関 ID ジェネレータ データ セットを含めることができます。これは、相関 ID として使用される 24 バイトのコードを生成します。

以下の図は、テスト ケースの例を示しています。



バイナリ ペイロードは、WebSphere MQ ベースライン テスト ケースの構造に影響します。 このシナリオでは、テスト ケースに以下のステップがあります。

- 大容量データのデータ セットから読み取る何も実行しないステップ

- Base64 文字列からバイトの配列に要求ペイロードをデコードする JavaScript ステップ
- 要求を送信する WebSphere MQ ステップ
- 応答を受信する WebSphere MQ ステップ
- バイトの配列から Base64 文字列に応答ペイロードをエンコードする JavaScript ステップ

大容量データのデータ セットには、元の要求ペイロードの 2 つのバージョン（人間が理解できるバージョンおよび Base64 文字列）が含まれます。

大容量データのデータ セットには、元の応答ペイロードの 2 つのバージョン（人間が理解できるバージョンおよび Base64 文字列）が含まれます。

2 番目の JavaScript ステップには、**Check Base64 Response** （Base64 応答のチェック）という名前のアサーションが含まれます。 ベースライン テスト ケースを実行すると、このアサーションは実際の応答の Base64 文字列が予期される応答の Base64 文字列に一致することを確認します。 応答が一致しない場合、テストはエラーを生成します。

テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

WebSphere MQ ベースライン スイート

WebSphere MQ ベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

WebSphere MQ ベースライン スイートのテスト ケースは、WebSphere MQ ベースライン テスト ケースに似ています。ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはステップに格納されます。

バイナリ ペイロードでは、テスト ケースに以下のステップがあります。

- 元の要求ペイロードの Base64 文字列が含まれる応答としてのテキストの解析ステップ
- ログ ファイルに元の要求ペイロードの人間が理解できるバージョンを書き込むログ メッセージの出力ステップ
- Base64 文字列からバイトの配列に実際の要求 ペイロードをデコードする JavaScript ステップ
- 要求を送信する WebSphere MQ ステップ
- 応答を受信する WebSphere MQ ステップ
- 元の応答ペイロードの Base 64 文字列が含まれる応答としてのテキストの解析ステップ
- ログ ファイルに元の応答ペイロードの人間が理解できるバージョンを書き込むログ メッセージの出力ステップ
- バイトの配列から Base64 文字列に実際の応答ペイロードをエンコードする JavaScript ステップ

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

一般的なベースライン

「*一般的なベースライン*」という用語は、以下のプロトコル以外のプロトコルに対して生成されるベースラインを指します。

- EJB
- JMS
- REST
- TIBCO ActiveMatrix BusinessWorks
- TIBCO Enterprise Message Service（EMS）
- Web HTTP
- Web サービス
- webMethods Integration Server
- WebSphere MQ

このセクションには、以下のトピックが含まれます。

一般的なベースラインの生成

一般的なベースラインを生成するには、以下の手順に従います。

次の手順に従ってください:

1. 1 つ以上のトランザクション フレームを、「一般的なベースライン (P. 140)」に示されているプロトコル以外のプロトコルのシェルフに追加します。
2. シェルフを開きます。
3. ［ベースラインの作成］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 下向きの矢印をクリックします。
6. 作成モード フィールドが使用可能な場合は、オプションを選択します。

   統合

   単一のテスト ケースおよびデータ セットが含まれるベースラインを作成します。

   展開

   テストのスイート（各トランザクションに 1 つ）およびテストを実行するためのスイート ドキュメントが含まれるベースラインを作成します。
7. テスト ケースに含まれるステップのタイプを指定します。

   アプリケーション テスト ステップの使用

   テスト ケースには、シェルフに追加したトランザクション フレームに対応するステップが含まれます。 たとえば、SOAP フレームを選択すると、Web サービス実行（XML）ステップが含まれるテスト ケースになります。

   トランザクション フレーム ステップの使用

   テスト ケースには、トランザクション フレームの実行ステップが含まれます。
8. （オプション）マジック デートを使用するようにベースラインを設定します。

   テスト ケースにマジック デートを適用

ベースライン テスト ケースまたはスイート内の日付文字列を変数定義文字列に変換します。 たとえば、特定の日時が含まれる文字列の代わりに、文字列に現在の日時から 7 日を指定する関数を含めることができます。 この動作は、CA Service Virtualization のマジック デートの **doDateDeltaFromCurrent** フォームと同等です。 このオプションがサポートされていない場合、オプションは表示されません。

9. （オプション）ベースラインのテンプレートとして使用されるテストケースを指定します。
10. ［Create］をクリックします。
11. ベースラインが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

一般的なベースライン テスト ケース

一般的なベースライン テスト ケースには、大容量データのデータ セットから読み取るテスト ステップがあります。 テスト ステップは以下のいずれかです。

- パス グラフで選択したノードに対応するステップ
- トランザクション フレームの実行ステップ

テスト ケースにパス グラフで選択したノードに対応するステップがある場合、データ セットには URL、要求のボディ、予期される応答、応答コードなどの情報が含まれます。

テスト ケースにトランザクション フレームの実行ステップがある場合、データ セットにはエージェント名、トランザクション フレーム、および予期される応答が含まれます。

デフォルトでは、データ セットはグローバルではなく、ローカルです。

テスト ステップには、「**応答の値の一致を確認**」という名前のアサーションが含まれます。 ベースライン テスト ケースを実行すると、このアサーションは実際の応答の値が予期される応答の値に一致することを確認します。 値が一致しない場合、テストはエラーを生成します。

テスト ケースを実行するには、以下のいずれかのアクションを実行します。

- クイック テストのステージング
- テスト ケースのステージング

［統合ベースライン］タブ (P. 145)で結果をモニタできます。

注: クイック テストまたはテスト ケースのステージング方法の詳細については、「*CA Application Test の使用*」の「テスト ケースおよびスイートの実行」を参照してください。

一般的なベースライン スイート

一般的なベースライン スイートのテスト ケースは、Tests フォルダではなく、Baselines フォルダにあります。

スイート ドキュメントは Suites フォルダに配置されます。

一般的なベースライン スイートのテスト ケースは、一般的なベースライン テスト ケースに似ています。 ただし、それらには大容量データのデータ セットが含まれません。 代わりに、データはテスト ステップに格納されます。

スイートを実行すると、各テストの結果は［ベースライン結果］パネル (P. 147)に表示されます。

注: スイートを実行する方法の詳細については、「*CA Application Test の使用*」の「テスト スイートの実行」を参照してください。

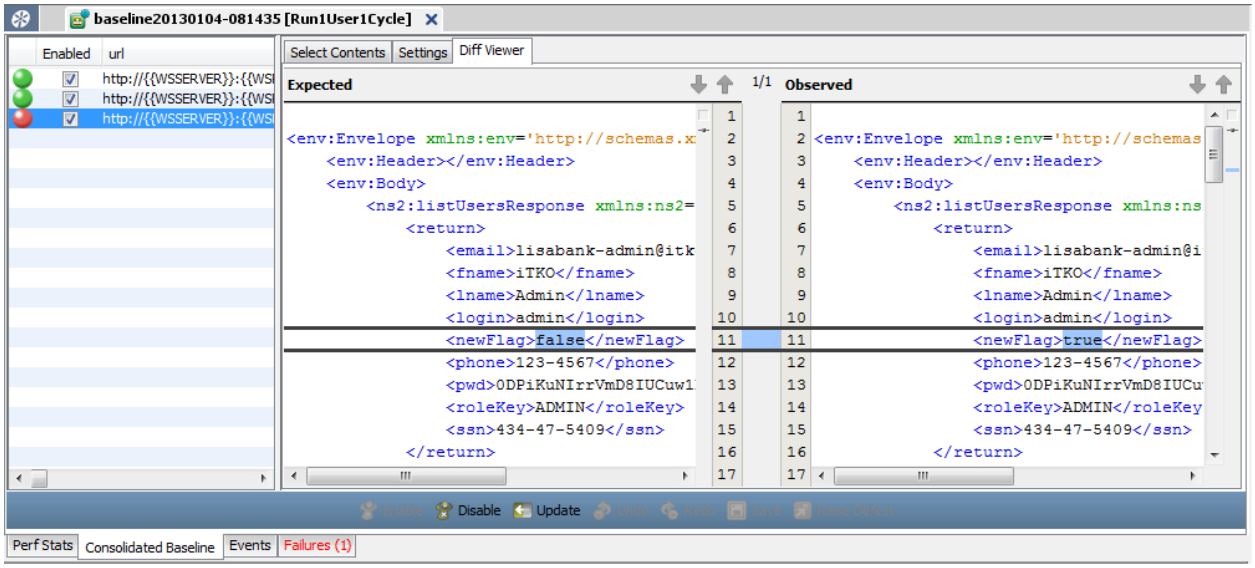
[統合ベースライン]タブ

DevTest ワークステーション の［統合ベースライン］タブでは、実行時にベースライン テスト ケースをモニタおよび更新できます。

以下のタスクのいずれかを実行すると、［統合ベースライン］タブがテスト モニタ ウィンドウに表示されます。

- ベースライン テスト ケースに対するクイック テストの実行
- ベースライン テスト ケースのステージングおよび実行

以下の図は、3 つのシナリオが含まれるベースライン テスト ケースの［統合ベースライン］タブを示しています。 テスト ランで、最後のシナリオは失敗しています。 差分ビューアには、最後のシナリオで問題が発生した箇所が表示されます。



タブの左側の部分には、ベースライン テスト ケースの大容量データのデータ セットの行が含まれます。 各行にはステータス アイコンが含まれます。

- 灰色は、シナリオがまだ実行されていないことを示します。
- 緑と白い矢印は、シナリオが実行中であることを示します。
- 緑は、シナリオが成功したことを示します。
- 赤は、シナリオが失敗したことを示します。
- オレンジは、シナリオが停止されたことを示します。

- 黄色は、予期される応答が実際の応答で更新されたことを示します。

ベースライン テスト ケースが実行された後、結果を表示するために行を選択できます。 シナリオが成功または失敗した場合、差分ビューアには予期される応答および実際の応答が表示されます。 すべての違いが強調表示されます。 シナリオが停止された場合、サイクル履歴が代わりに表示されます。

［設定］タブを使用すると、比較オプションを変更できます。

注: 比較オプションの詳細については、「*CA Application Test の使用*」のグラフィカル XML 比較アサーションの説明を参照してください。

下部のツールバーには以下のボタンが含まれます。

有効

後続のテスト ランにシナリオを含めます。 設定した後、必ず［保存］をクリックしてください。

無効

シナリオが後続のテスト ランで実行されるのを防ぎます。設定した後、必ず［保存］をクリックしてください。

更新

予期される応答を実際の応答で更新します。 このアクションを実行できるのは、機能上のエラーが原因ではなく、テスト中のシステムが変更されたことが原因でシナリオが失敗した場合です。 設定した後、必ず［保存］をクリックしてください。

元に戻す

最新の変更を元に戻します。

やり直し

最新の変更を再度実行します。

保存

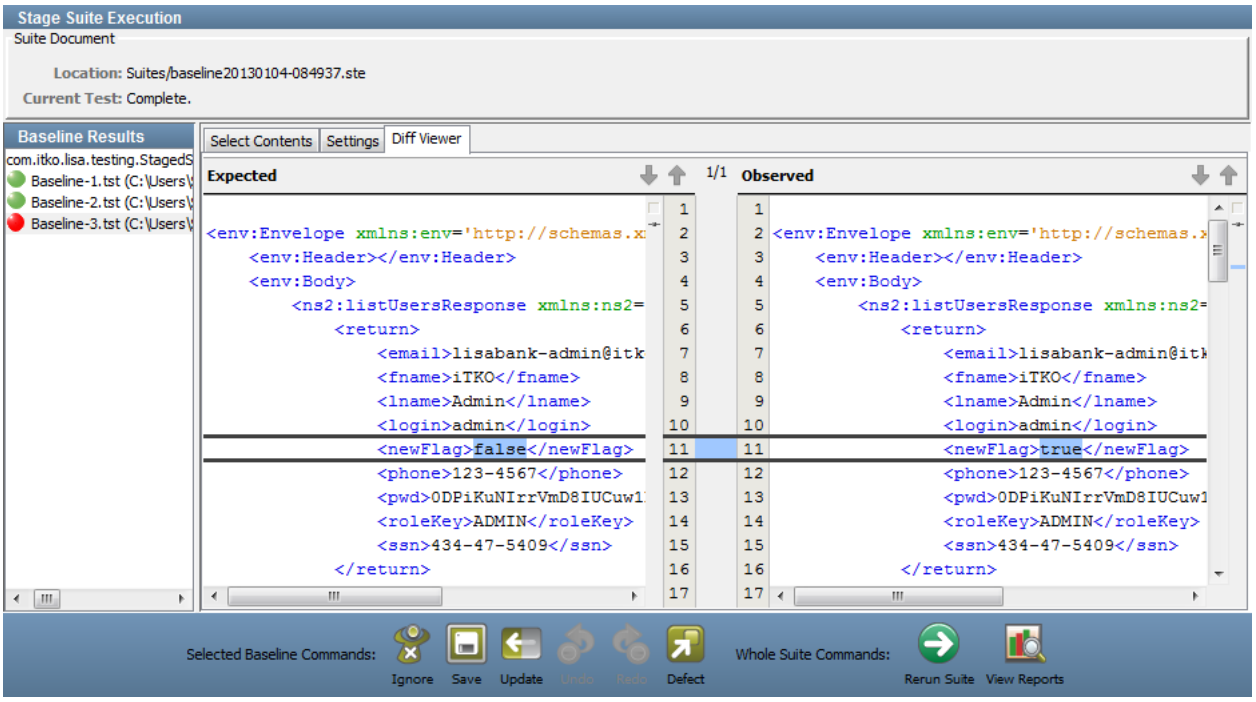
行なわれたすべての変更を保存します。

［ベースライン結果］パネル

DevTest ワークステーション の［ベースライン結果］パネルでは、実行時にベースライン スイートをモニタおよび更新できます。

ベースライン スイートを実行すると、［ベースライン結果］パネルが［スイートのステージングの実行］タブに表示されます。

以下の図は、3 つのテストが含まれるベースライン スイートの［ベースライン結果］パネルを示しています。 スイート実行で、最後のテストは失敗しています。 差分ビューアには、最後のテストで問題が発生した箇所が表示されます。



各テストにはステータス アイコンが含まれます。

- 緑は、テストが成功したことを示します。
- 赤は、テストが失敗（テストを失敗にするアサーションが起動）したことを示します。
- オレンジは、テストが停止されたことを示します。 たとえば、テストステップが予期しない例外をスローしています。

ベースライン スイートが実行された後、結果を表示するためにテストを選択できます。 テストが成功または失敗した場合、差分ビューアには予期される応答および実際の応答が表示されます。 すべての違いが強調表示されます。 テストが停止された場合、サイクル履歴が代わりに表示されます。

テストが失敗した場合、以下のいずれかのアクションを実行できます。

- テストを選択し、［差分ビューア］タブ内の行を右クリックし、［無視］をクリックします。 スイートの後続の実行で、この行はスイートのすべてのテストに対してスキップされます。
- テストを選択し、下部の［無視］アイコンをクリックします。 それ以降のスイートの実行で、このテストは実行されません。
- テストを選択し、下部の［更新］アイコンをクリックします。 予期される応答が実際の応答で更新されます。

下部のツールバーには、以下のスイート全体のコマンドが含まれます。

スイートの再実行

スイートを再実行します。

レポートの表示

レポート コンソールに実行の統計を表示します。

第 8 章: 仮想サービスの作成

CAI では、CAI データベースのトランザクションから仮想サービスを作成できます。

トランザクションを生成する場合、適切なプロトコルのキャプチャ レベル (P. 27)が［全データ］に設定されていることを確認します。

カテゴリが「GUI」のトランザクション フレームの仮想サービスを作成することはできません。

注:

- この機能には、個別のライセンスが必要です。
- サービス仮想化の詳細については、「CA Service Virtualization の使用」を参照してください。

このセクションには、以下のトピックが含まれています。

仮想サービスを作成する場合のトランザクションの統合

同じカテゴリのトランザクションの仮想サービスを作成する場合、CAI は 1 つのカテゴリ当たり 1 つの仮想サービスにトランザクションを統合します。 たとえば、3 つの EJB トランザクションがある場合、1 つの仮想サービスのみが作成されます。 シェルフの仮想サービスに統合されたトランザクションすべてを表示できます。 単一およびマージされたトランザクションのみを統合できます。 すべての発生したトランザクション フレームは、統合できません。

すべてのトランザクションが 1 つの仮想サービスに統合される場合、統合された仮想サービス名はカテゴリ名です。

1 つのエージェントのすべてのカテゴリ フレームが統合される場合、統合された仮想サービス名は、カテゴリ名およびエージェント名です。

単一の仮想サービスに統合されたトランザクションを表示するには、シェルフで［Create VS］をクリックして、  をクリックします。

以下の図は、シェルフの 3 つの EJB トランザクションから作成された、統合された EJB 仮想サービスを示しています。

トランザクション **$Proxy102.getUser**、**EJB3UserControlBean.getUser**、および **$Proxy102.validate** が、**JBoss_LISABank_EJB** という名前の 1 つの仮想サービスに統合されたことに注目します。

注: 仮想サービスを作成する前に、統合元の任意の EJB トランザクションを削除するオプションがあります。

仮想サービス作成時の統合では、以下のプロトコルがサポートされています。

- REST： 1 つの仮想サービス
- すべての SOAP： 1 つの仮想サービス
- すべての HTTP： 1 つの仮想サービス
- EJB： 1 つのエージェント当たり 1 つの仮想サービス
- JDBC： 1 つのエージェント当たり 1 つの仮想サービス
- JCA： 1 つのエージェント当たり 1 つの仮想サービス
- TIBCO： 1 つのエージェント当たり 1 つの仮想サービス

注: JMS および SAP はサポートされていません。

発生したトランザクションすべての詳細については、「発生したすべてのトランザクションのシェルフへの追加 (P. 64)」を参照してください。マージされたトランザクションの詳細については、「繰り返しパスのマージ (P. 48)」を参照してください。

EJB トランザクションからの仮想サービスの作成

CAI データベースの EJB トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

同じ EJB のメソッドがすべて仮想化されます。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

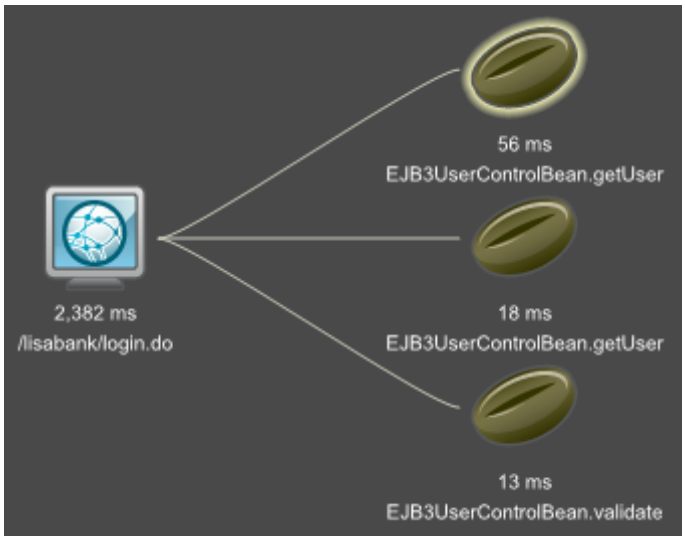
次の手順に従ってください:

1. EJB トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. 統合されたトランザクションを表示するには、  をクリックします。
8. 統合されたトランザクションを削除するには、  をクリックします。
9. ［Create］をクリックします。
10. 仮想サービスが作成されるプロジェクトを選択します。
11. ［Create］をクリックします。

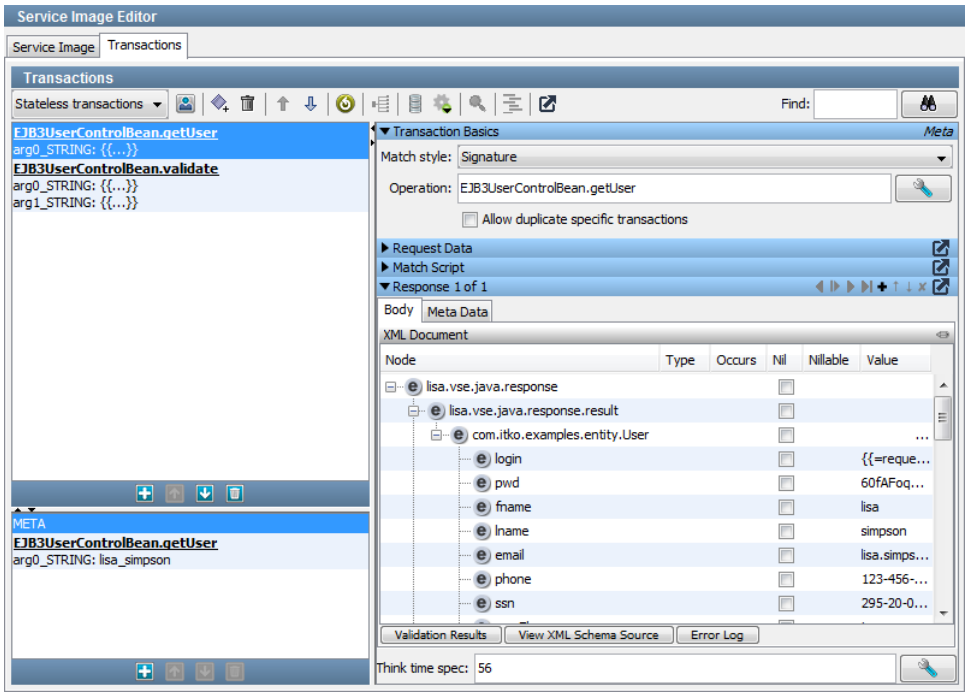
注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

例: EJB ユーザ制御ビーン

以下の図は、EJB コンポーネントが含まれるパス グラフを示しています。「発生したトランザクションをすべてシェルフに追加する」アクションが、いずれかの **getUser()** コンポーネントに適用されています。



以下の図は、生成されたサービス イメージを示しています。 このサービス イメージには、ステートレス トランザクションが含まれます。 複数のメソッドが仮想化されることに注意してください。



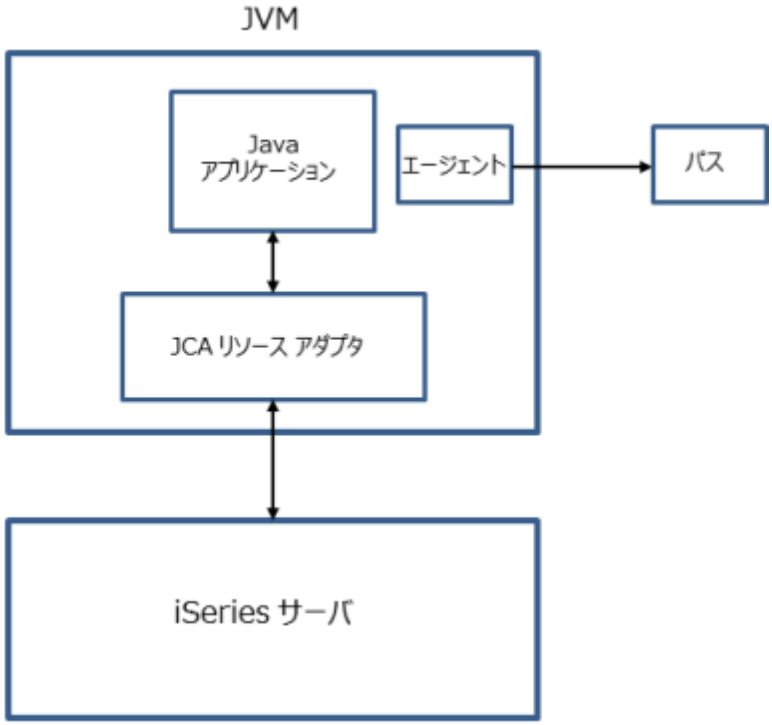
仮想サービス モデルでは、仮想 Java リスナ ステップにはエージェント名および EJB 名が含まれます。

JCA iSeries トランザクションからの仮想サービスの作成

この機能は以下の IBM 製品で動作します。

- IDE： IBM Integration Designer 8.0
- アダプタ： IBM WebSphere Adapter for IBM i バージョン 7.5.0.2

以下の図は、レコーディング段階のアーキテクチャを示しています。 DevTest Java エージェントは Java アプリケーションに対して設定されます。 Java アプリケーションは、JCA リソース アダプタを使用して iSeries サーバに対してプログラム コールを発行します。 エージェントはコールを監視し、DevTest ポータルで表示できる対応するパスを生成します。



レコーディング段階の後、仮想化アーティファクトを生成し、仮想サービスを展開します。

注: エージェントのインストールおよび設定については、「エージェント」を参照してください。 仮想サービスの展開については、「*CA Service Virtualization の使用*」を参照してください。

JCA iSeries 仮想化アーティファクトの生成

JCA iSeries 仮想化アーティファクトを生成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

1. JCA トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. 統合されたトランザクションを表示するには、  をクリックします。
8. 統合されたトランザクションを削除するには、  をクリックします。
9. ［Create］をクリックします。
10. 仮想サービスが作成されるプロジェクトを選択します。
11. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

JMS トランザクションからの仮想サービスの作成

最初の手順の結果は、仮想サービス イメージ レコーダにインポートできる RAW トラフィック ファイルです。この方法の利点は、プロキシ レコーディングを実行する必要がなくなることです。

要求および応答のボディに加えて、RAW トラフィック ファイルにはメタデータおよび属性にすべての接続およびキュー情報が含まれます。

2 番目の手順では、サービス イメージおよび仮想サービス モデルを作成するために RAW トラフィック ファイルを使用します。

注: この機能は、JMS 接続ファクトリを取得するために JNDI が使用される設定に対してサポートされています。この機能は、JMS 接続ファクトリを取得するために IBM WebSphere MQ から直接 API が使用される設定に対してもサポートされています。

注: 仮想サービス イメージ レコーダ、データ プロトコル、マジック ストリング、および仮想サービス モデルの展開の詳細については、「*CA Service Virtualization の使用*」を参照してください。

JMS トランザクションから RAW トラフィック ファイルを作成する方法

1. JMS トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. ［Create］をクリックします。
6. RAW トラフィック ファイルが追加されるプロジェクトを選択します。
7. ［Create］をクリックします。

サービス イメージおよび仮想サービス モデルを作成する方法

1. DevTest ワークステーション のメイン メニューから［ファイル］-［新規］-［VS イメージ］-［レコーディングから作成］を選択します。

   仮想サービス イメージ レコーダが表示されます。
2. 以下の手順を実行します。

a. ［イメージの書き込み先］フィールドで、作成するサービス イメージの完全修飾名を入力します。

b. ［インポート トラフィック］フィールドで、Data フォルダ内の RAW トラフィック ファイルを参照して選択します。

c. ［トランスポート プロトコル］フィールドで、［標準 JMS］を選択します。

d. ［モデル ファイル］フィールドで、作成する仮想サービス モデルの完全修飾名を入力します。

e. ［次へ］をクリックします。次の手順では、メッセージ レコーディング スタイルを選択するように促されます。

3. 要求および応答キュー情報を確認する場合は、以下の手順に従います。

a. ［送信先およびトランザクション トラッキング モードを確認］チェック ボックスをオンにします。

b. ［相関］ドロップダウン リスト内の相関スキームが正しくない場合は、値を変更します。

c. ［次へ］をクリックします。 次の手順には、［送信先情報］タブおよび［接続のセットアップ］タブが含まれます。

d. ［送信先情報］および［接続のセットアップ］タブ内の値は自動的に入力されます。 プロキシ レコーディングを実行していないので、［プロキシ送信先］フィールドは［**該当なし**］に設定されます。 どちらのタブも更新する必要はありません。ただし、CAI が正しい値を設定しない場合は更新できます。 ［次へ］をクリックします。 次の手順には応答情報が含まれます。

e. この手順の値は自動的に入力されます。 ［応答先］領域には 1 つ以上の応答キューが含まれます。 変更する必要はありません。ただし、CAI が正しい値を設定しない場合は変更できます。 ［次へ］をクリックします。 データ プロトコル手順が表示されます。

4. 以下の手順を実行します。

a. ［要求側データ プロトコル］領域で、プラス記号をクリックします。

b. 新しく追加された行の左の列をクリックし、適切なデータ プロトコルを選択します。 XML ベースのアプリケーションに対しては、［ジェネリック XML ペイロード パーサ］は一般的な良い選択です。

c. アプリケーション応答が XML またはテキスト形式でない場合、VSE が応答でマジック ストリングの置換を実行するために応答側のデータ プロトコルが必要な場合があります。

d. ［次へ］をクリックします。

5. 表示される次の手順（ある場合）は、選択したデータ プロトコルによって異なります。 たとえば、［ジェネリック XML ペイロード パーサ］データ プロトコルを選択した場合、次の手順は VSE 要求を形成する XPath を作成するようにユーザに促します。必要に応じて、「*CA Service Virtualization の使用*」を参照して手順を完了します。 最後の手順は、レコーダが記録されたもので後処理を実行していることを示します。

6. ［終了］をクリックします。

   トランザクションがサービス イメージに保存されます。また、仮想サービス モデルが作成されます。

仮想サービスを実行する方法

1. 元のサービスをシャットダウンします。
2. DevTest ワークステーション に移動し、作成した仮想サービス モデルを展開します。
3. サービスとして仮想サービスを持ったクライアント アプリケーションを実行します。

REST トランザクションからの仮想サービスの作成

CAI データベースの REST トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

1. REST トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 不明な要求用の応答を設定します。
6. 統合されたトランザクションを表示するには、  をクリックします。
7. 統合されたトランザクションを削除するには、  をクリックします。
8. ［Create］をクリックします。
9. 仮想サービスが作成されるプロジェクトを選択します。
10. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

SAP ERPConnect トランザクションからの仮想サービスの作成

ERPConnect は、.NET アプリケーションから SAP と対話できるライブラリです。

CAI がキャプチャした ERPConnect トランザクションのセットから仮想サービスを作成できます。

一般的な手順は以下のとおりです。

1. エージェント ファイルをインストールして設定します。
2. .NET アプリケーションを実行します。
3. 仮想化アーティファクトを生成します。
4. 仮想サービスを展開します。

注:

- ERPConnect は DevTest Solutions に含まれていません。
- 仮想サービスの展開については、「*CA Service Virtualization の使用*」を参照してください。

SAP ERPConnect のエージェント ファイルのインストールおよび設定

.NET Framework の以下のバージョンがサポートされています。

- .NET 2.0 Framework, v2.0.50727
- .NET 3.0 Framework, v3.0.4506
- .NET 3.5 Framework, v3.5.21022

開始する前に、Visual Studio 2012 Update 4 の Visual C++ 再頒布可能パッケージをインストールします。 現時点では、Microsoft の Web サイトからこのコンポーネントを入手できます。

手順は、以下のどの .NET アプリケーションを使用しているかによって異なります。

- インターネット インフォメーション サービス（IIS）サーバで実行されている .NET アプリケーション
- スタンドアロン .NET アプリケーション

IIS サーバで実行されている .NET アプリケーションを使用している場合に SAP ERPConnect のエージェント ファイルをインストールおよび設定する方法

1. 以下のファイルを **LISA_HOME¥agent** ディレクトリから IIS サーバが配置されているコンピュータ上のディレクトリにコピーします。
   - **LisaAgent.dll**
   - **NativeAgent32.dll**
   - **NativeAgent64.dll**
   - **LisaAgentLauncher.exe**
2. ファイルをコピーしたディレクトリからコマンド プロンプトを開きます。
3. 以下のコマンドを実行します。

   ```
   LisaAgentLauncher.exe /i
   ```

   エージェント ファイルがそのコンピュータで登録されます。
4. 以下のコマンドを実行します。

   ```
   LisaAgentLauncher.exe /iis /name <agent-name> /url
   tcp://<broker-host>:<broker-port>
   ```

   ドメインを指定するには、**/domain** オプションを追加します。
5. IIS のワーカ プロセスが開始されると、エージェントが有効になります。

スタンドアロン .NET アプリケーションを使用している場合に SAP ERPConnect のエージェント ファイルをインストールおよび設定する方法

1. 以下のファイルを **LISA_HOME¥agent** ディレクトリから .NET アプリケーションの実行可能ファイルが含まれるディレクトリにコピーします。
   - **LisaAgent.dll**
   - **NativeAgent32.dll**
   - **NativeAgent64.dll**
   - **LisaAgentLauncher.exe**
2. ファイルをコピーしたディレクトリからコマンド プロンプトを開きます。
3. 以下のコマンドを実行します。

   ```
   LisaAgentLauncher.exe /i
   ```

   エージェント ファイルがそのコンピュータで登録されます。
4. 以下のように環境変数を設定します。

   ```
   set COR_PROFILER={BEB45448-91FA-4A7C-BF9A-68AA889DC873}
   set COR_ENABLE_PROFILING=1
   set COR_LISA_AGENT=name=<agent-name>
   ```
5. .NET アプリケーションを再起動します。

SAP ERPConnect 仮想化アーティファクトの生成

CAI データベースの ERPConnect トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

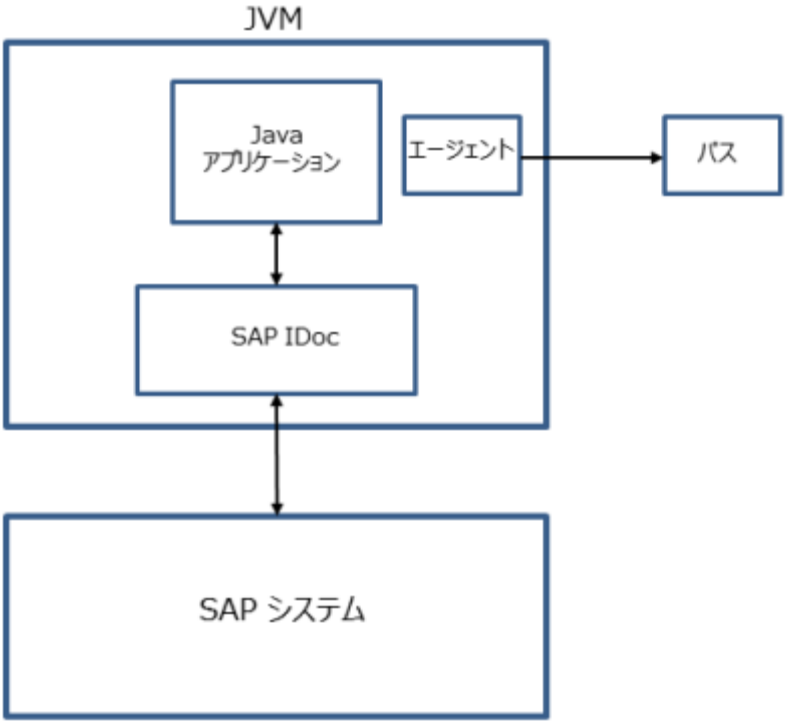
クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

1. SAP トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. ［Create］をクリックします。
8. 仮想サービスが作成されるプロジェクトを選択します。
9. ［Create］をクリックします。

SAP IDoc トランザクションからの仮想サービスの作成

以下の図は、レコーディング段階のアーキテクチャを示しています。DevTest Java エージェントは Java アプリケーションに対して設定されます。Java アプリケーションは SAP システムに IDoc メッセージを送信します。エージェントはメソッド コールを監視し、DevTest ポータルで表示できる対応するパスを生成します。



ステートフルな会話はサポートされていません。

エージェントを SAP システムにインストールおよび設定する必要はありません。

レコーディング段階の後、仮想化アーティファクトを生成し、仮想サービスを展開します。

注: エージェントのインストールおよび設定については、「エージェント」を参照してください。 仮想サービスの展開については、「*CA Service Virtualization の使用*」を参照してください。

SAP IDoc 仮想化アーティファクトの生成

SAP IDoc 仮想化アーティファクトを生成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

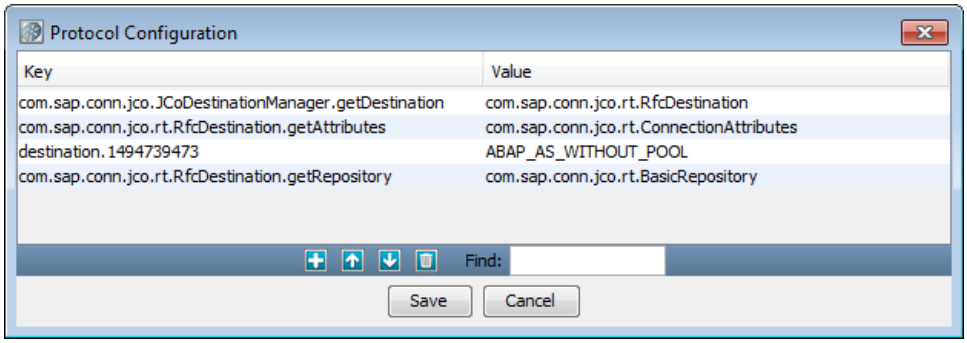
1. SAP トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. ［Create］をクリックします。
8. 仮想サービスが作成されるプロジェクトを選択します。
9. ［Create］をクリックします。

SAP IDoc 送信先のフィルタリング

仮想化される SAP IDoc 送信先を変更できます。

生成された仮想サービス モデルで、仮想 Java リスナ ステップを開き、右下のリスト内の［**SAP**］プロトコルをダブルクリックします。［プロトコル設定］ダイアログ ボックスが表示されます。 このダイアログ ボックスには、1 セットのキー/値ペアが含まれます。

以下の図は、［プロトコル設定］ダイアログ ボックスを示しています。



destination. で始まり、 数字で終わるキーは、仮想化する送信先を指定します。

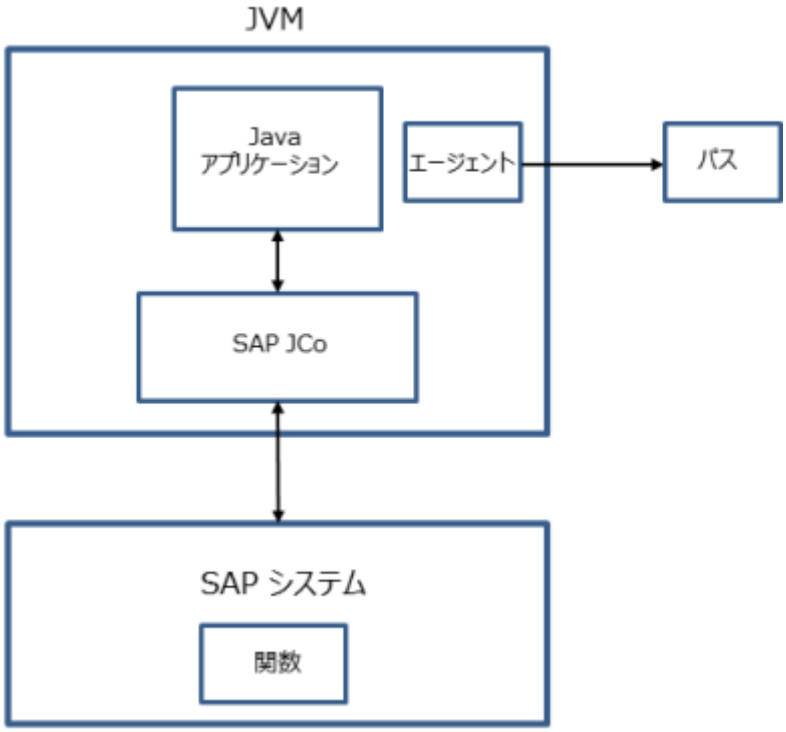
複数の送信先を仮想化するには、各送信先に送信先キーを追加します。［プロトコル設定］ダイアログ ボックス内でその数が一意である限り、任意の数を使用できます。

送信先をすべて仮想化するには、送信先キーを削除します。

Java クラス名のようなキーは無視できます。

SAP JCo トランザクションからの仮想サービスの作成

以下の図は、レコーディング段階のアーキテクチャを示しています。 DevTest Java エージェントは Java アプリケーションに対して設定されます。 Java アプリケーションは、SAP JCo を使用して SAP システム内の関数への RFC コールを実行します。 エージェントはコールを監視し、DevTest ポータルで表示できる対応するパスを生成します。



この機能は、SAP JCo 3.0 のみをサポートします。

この機能は、入出力パラメータおよびテーブル データを使用する RFC コール、およびステートフル コールをサポートします。

エージェントを SAP システムにインストールおよび設定する必要はありません。

レコーディング段階の後、仮想化アーティファクトを生成し、仮想サービスを展開します。

注: エージェントのインストールおよび設定については、「エージェント」を参照してください。 仮想サービスの展開については、「*CA Service Virtualization の使用*」を参照してください。

SAP JCo 仮想化アーティファクトの生成

SAP JCo 仮想化アーティファクトを生成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

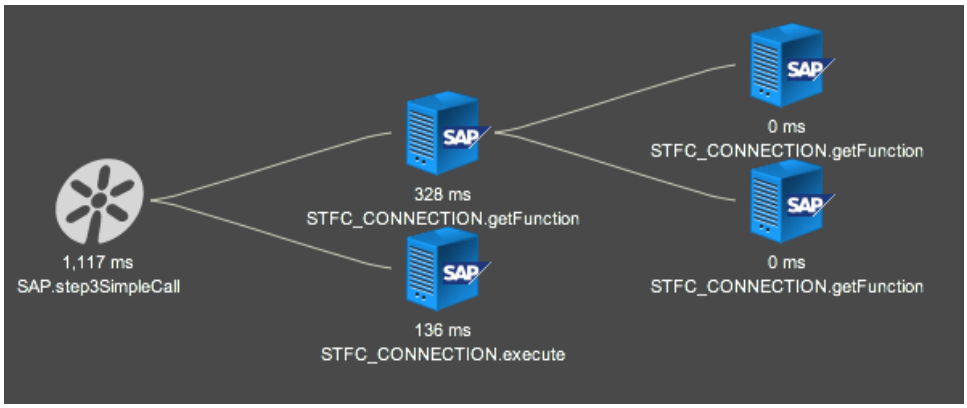
"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

以下の図は、SAP フレームが含まれるパス グラフを示しています。 このパス グラフは、関数を取得し、その関数を実行する標準的な SAP JCo パターンを示しています。



仮想化アーティファクトが生成されると、CAI は選択された SAP コンポーネントの送信先名を確認します。 その後、CAI はこの送信先がある SAP コンポーネントの表示可能なパスをすべて検索します。 これらの SAP コンポーネントは、仮想サービスを作成するために使用されます。

選択した SAP コンポーネントの送信先名を表示するには、トランザクション詳細ダイアログ ボックスの［XML］タブをクリックします。

次の手順に従ってください:

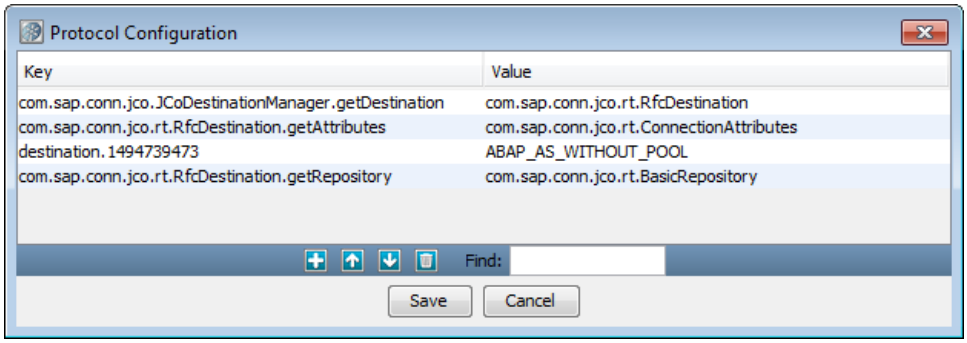
1. SAP トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. ［Create］をクリックします。
8. 仮想サービスが作成されるプロジェクトを選択します。
9. ［Create］をクリックします。

SAP JCo 送信先のフィルタリング

仮想化される SAP JCo 送信先を変更できます。

生成された仮想サービス モデルで、仮想 Java リスナ ステップを開き、右下のリスト内の［**SAP**］プロトコルをダブルクリックします。［プロトコル設定］ダイアログ ボックスが表示されます。 このダイアログ ボックスには、1 セットのキー/値ペアが含まれます。

以下の図は、［プロトコル設定］ダイアログ ボックスを示しています。



destination. で始まり、 数字で終わるキーは、仮想化する送信先を指定します。

複数の送信先を仮想化するには、各送信先に送信先キーを追加します。［プロトコル設定］ダイアログ ボックス内でその数が一意である限り、任意の数を使用できます。

送信先をすべて仮想化するには、送信先キーを削除します。

Java クラス名のようなキーは無視できます。

TIBCO ActiveMatrix BusinessWorks トランザクションからの仮想サービスの作成

TIBCO ActiveMatrix BusinessWorks 内のプロセスを仮想化するために CA Continuous Application Insight を使用できます。仮想サービスが展開されると、仮想サービスはサービス イメージで収集された応答でプロセス アクティビティの動作を置き換えます。

以下の方法がサポートされています。

- DevTest ワークステーション から仮想化します。仮想サービス イメージ レコーダを使用して、トラフィックをキャプチャし、仮想サービスを生成します。 CAI を有効にする必要はありません。
- DevTest ポータルから仮想化します。 DevTest ポータルに表示されるトランザクションを生成します。 後からいつでも、仮想サービスを作成するためにトランザクションを使用できます。 CAI が有効になっている必要があります。 この方法は最初の方法より高い柔軟性を提供しますが、オーバーヘッドも大きくなります。

ステートフルまたはステートレス仮想サービスを生成できます。 ステートレスを推奨します。 この設定は、仮想サービス イメージ レコーダおよび DevTest ポータルから制御できます。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

これらの手順は、TIBCO ActiveMatrix BusinessWorks に DevTest Java エージェントがインストールおよび設定されていると仮定しています。

注: DevTest Java エージェントのインストールおよび設定の詳細については、「*エージェント*」を参照してください。仮想サービス イメージ レコーダの使用および仮想サービス モデルの展開の詳細については、「*CA Service Virtualization の使用*」を参照してください。

DevTest ワークステーション から TIBCO ActiveMatrix BusinessWorks を仮想化する方法

1. DevTest ワークステーション に移動し、仮想サービス イメージ レコーダを起動します。

   基本情報を提供するように促されます。

2. サービス イメージおよび仮想サービス モデルの名前を指定します。
3. トランスポート プロトコルを［Java］に設定します。
4. ［次へ］をクリックします。

   仮想化する Java クラスを選択するように促されます。

5. エージェントを選択して［接続されたエージェント］リストに移動させます。
6. ［プロトコル］矢印を展開し、右ペインに TIBCO BW プロトコルを移動させます。
7. ［次へ］をクリックします。 必要に応じて、記録する TIBCO プロセスの実行をトリガします。 プロセスが実行されると、エージェントはアクティビティを記録します。
8. 記録が完了したら、［次へ］をクリックし、仮想サービス イメージ レコーダの残りのステップを実行します。データ プロトコルを選択する必要はありません。

   サービス イメージおよび仮想サービス モデルが作成されます。

9. 仮想サービス モデルを展開して開始します。

DevTest ポータルから TIBCO ActiveMatrix BusinessWorks を仮想化する方法

1. TIBCO プロセスの実行をトリガします。
2. TIBCO トランザクション フレームをシェルフに追加します。 左端のフレームを選択することにより、会話全体を仮想化できます。 アクティビティ名が含まれるフレームを選択することにより、特定のアクティビティを仮想化できます。
3. シェルフを開きます。
4. ［Create VS］をクリックします。
5. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
6. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。

7. 不明な要求用の応答を設定します。
8. 統合されたトランザクションを表示するには、  をクリックします。
9. 統合されたトランザクションを削除するには、  をクリックします。
10. ［Create］をクリックします。
11. 仮想サービスが作成されるプロジェクトを選択します。
12. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

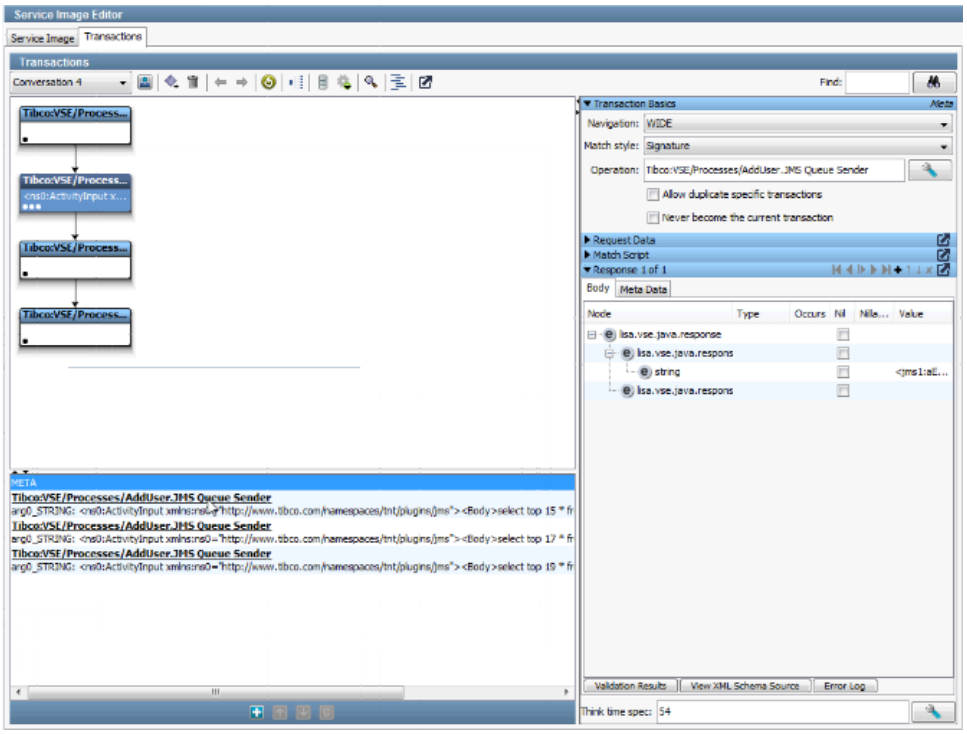
例: ユーザの追加プロセスの仮想化

以下の図は、TIBCO Designer でのプロセス定義を示しています。 プロセスには、File Poller、JMS Queue Sender、Confirm、および End アクティビティが含まれています。

File Poller アクティビティはテキスト ファイルをモニタします。 ファイルが変更されると、アクティビティはプロセスを起動します。

JMS Queue Sender アクティビティは指定されたキューにメッセージを送信します。

以下の図は、仮想サービス イメージ レコーダが生成したサービス イメージを示しています。 4 つのノードとの会話が［トランザクション］タブに表示されます。 4 つのノードはプロセス定義内の 4 つのアクティビティに対応します。



会話の各ノードには［操作］フィールドが含まれます。 このフィールドの値は完全修飾プロセス名およびアクティビティ名から構成されます。 上記の画像では、選択したノードの値は **Tibco:VSE/Processes/AddUser.JMS Queue Sender** です。

仮想サービス モデルは Java トランスポート プロトコル用のデフォルトです。

Web HTTP トランザクションからの仮想サービスの作成

CAI データベースの Web HTTP トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

1. Web HTTP トランザクションをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. 不明な要求用の応答を設定します。
6. 統合されたトランザクションを表示するには、 をクリックします。
7. 統合されたトランザクションを削除するには、 をクリックします。
8. ［Create］をクリックします。
9. 仮想サービスが作成されるプロジェクトを選択します。
10. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

Web サービストランザクションからの仮想サービスの作成

CAI データベースの Web サービス トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

添付ファイルを持つ Web サービスはサポートされていません。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

1. SOAP トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 不明な要求用の応答を設定します。
6. 統合されたトランザクションを表示するには、  をクリックします。
7. 統合されたトランザクションを削除するには、  をクリックします。
8. ［Create］をクリックします。
9. 仮想サービスが作成されるプロジェクトを選択します。
10. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

webMethods トランザクションからの仮想サービスの作成

CAI データベースの webMethods Integration Server トランザクションのセットから 仮想サービスを作成するには、以下の手順に従います。

webMethods Integration Server には、フロー サービスおよびパイプラインの概念が含まれます。 フロー サービスでは、サービスのグループをカプセル化し、それらの間のデータのフローを管理できます。 パイプラインは、フロー サービスに対する入力値および出力値が含まれるデータ構造です。 フロー サービスにステップを追加して、サービスの呼び出し、パイプラインのデータの変更などのアクションを実行できます。

フロー サービスは仮想化される単位です。

仮想サービスには、不明な会話型要求および不明なステートレス 要求に対して送信される応答が含まれます。 仮想サービスを作成する場合、これらの応答のボディを設定できます。 以下のリストではオプションについて説明します。

"一致なし" をレポート

仮想化されたアプリケーションで例外を発生させます。

仮想サービスをバイパス

クラスおよびメソッドがまったく仮想化されなかったように、元の要求がそのまま通過できます。

次の手順に従ってください:

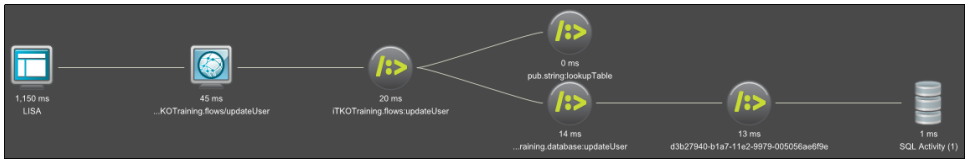
1. webMethods Integration Server トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから をクリックして保存します。
5. すべてのトランザクションをステートレスとして扱う場合は、チェック ボックスがオンになっていることを確認します。
6. 不明な要求用の応答を設定します。
7. 統合されたトランザクションを表示するには、 をクリックします。
8. 統合されたトランザクションを削除するには、 をクリックします。

9. ［Create］をクリックします。
10. 仮想サービスが作成されるプロジェクトを選択します。
11. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

例： ユーザの更新

以下の図は、webMethods Integration Server フレームが含まれるパス グラフを示しています。 この例のフロー サービスには、**updateUser** という名前が付けられています。 目的はデータベースにユーザの更新情報を送信することです。



注: webMethods Integration Server フレームのいずれかを選択すると、トランザクション詳細ダイアログ ボックスでパイプライン データを表示できます。

パス グラフに複数の webMethods Integration Server フレームが含まれる場合、仮想化するためにはフレームを 1 つ選択する必要があります。選択するフレームは、テスト中のシステムによって決まります。 この例では、以下のように決定されます。

- ソフトウェアと webMethods プロセスのインターフェースを確保するには、左端の webMethods Integration Server フレームを選択します。
- データベースを削除するには、直接データベースをコールする webMethods Integration Server フレームを選択します。

WebSphere MQ トランザクションからの仮想サービスの作成

最初の手順の結果は、仮想サービス イメージ レコーダにインポートできる RAW トラフィック ファイルです。この方法の利点は、プロキシ レコーディングを実行する必要がなくなることです。

要求および応答のボディに加えて、RAW トラフィック ファイルにはメタデータおよび属性にすべての接続およびキュー情報が含まれます。

バイナリ ペイロードを処理する場合は、拡張データ プロトコル ハンドラを使用して、バイナリ 要求を VSE が処理できる XML 形式に変換します。このデータ プロトコル ハンドラは仮想サービス イメージ レコーダで選択します。

また、応答がバイナリの場合は、2 つのオプションがあります。

- 拡張データ プロトコル ハンドラを使用して、サービス イメージ用のテキスト形式にバイナリ応答を解析します。 実行時に、元のバイナリ形式が各応答に対して再構築されます。
- 拡張データ プロトコル ハンドラを応答に使用しません。 実行時に、VSE エンジンは元のバイナリ応答を返し、マジック ストリングの置換を実行できません。

2 番目の手順では、サービス イメージおよび仮想サービス モデルを作成するために RAW トラフィック ファイルを使用します。

注: 仮想サービス イメージ レコーダ、データ プロトコル、マジック ストリング、および仮想サービス モデルの展開の詳細については、「*CA Service Virtualization の使用*」を参照してください。

WebSphere MQ トランザクションから RAW トラフィック ファイルを作成する方法

1. WebSphere MQ トランザクション フレームをシェルフに追加します。
2. シェルフを開きます。
3. ［Create VS］をクリックします。
4. デフォルトの名前を変更するには、名前を選択し、編集してから  をクリックして保存します。
5. 統合されたトランザクションを表示するには、  をクリックします。
6. 統合されたトランザクションを削除するには、  をクリックします。

7. ［Create］をクリックします。
8. RAW トラフィック ファイルが追加されるプロジェクトを選択します。
9. ［Create］をクリックします。

注: 統合されたトランザクションの詳細については、「仮想サービスを作成する場合のトランザクションの統合 (P. 150)」を参照してください。

サービス イメージおよび仮想サービス モデルを作成する方法

1. DevTest ワークステーション のメイン メニューから［ファイル］-［新規］-［VS イメージ］-［レコーディングから作成］を選択します。

   仮想サービス イメージ レコーダが表示されます。
2. 以下の手順を実行します。
   a. ［イメージの書き込み先］フィールドで、作成するサービス イメージの完全修飾名を入力します。
   b. ［インポート トラフィック］フィールドで、Data フォルダ内の RAW トラフィック ファイルを参照して選択します。
   c. ［トランスポート プロトコル］フィールドで、［IBM MQ シリーズ］を選択します。
   d. ［モデル ファイル］フィールドで、作成する仮想サービス モデルの完全修飾名を入力します。
   e. ［次へ］をクリックします。次の手順では、メッセージ レコーディング スタイルを選択するように促されます。
3. 要求および応答キュー情報を確認する場合は、以下の手順に従います。
   a. ［キューおよびトランザクション トラッキング モードを確認］チェック ボックスをオンにします。
   b. ［相関］ドロップダウン リスト内の相関スキームが正しくない場合は、値を変更します。
   c. ［次へ］をクリックします。 次の手順には、［送信先情報］タブおよび［接続のセットアップ］タブが含まれます。
   d. ［送信先情報］および［接続のセットアップ］タブ内の値は自動的に入力されます。 プロキシ レコーディングを実行していないので、［プロキシ キュー］フィールドは［該当なし］に設定されます。 どちらのタブも更新する必要はありません。ただし、CAI が正しい値を設定しない場合は更新できます。 ［次へ］をクリックします。 次の手順には応答情報が含まれます。

e. この手順の値は自動的に入力されます。［応答先］領域には 1 つ以上の応答キューが含まれます。 変更する必要はありません。ただし、CAI が正しい値を設定しない場合は変更できます。 ［次へ］をクリックします。 データ プロトコル手順が表示されます。

4. 以下の手順を実行します。

   a. ［要求側データ プロトコル］領域で、プラス記号をクリックします。

   b. 新しく追加された行の左の列をクリックし、適切なデータ プロトコルを選択します。 XML ベースのアプリケーションに対しては、［ジェネリック XML ペイロード パーサ］は一般的な良い選択です。

   c. アプリケーション応答が XML またはテキスト形式でない場合、VSE エンジンが応答でマジック ストリングの置換を実行するために応答側のデータ プロトコルが必要な場合があります。

   d. ［次へ］をクリックします。

5. 表示される次の手順（ある場合）は、選択したデータ プロトコルによって異なります。 たとえば、［ジェネリック XML ペイロード パーサ］データ プロトコルを選択した場合、次の手順は VSE 要求を形成する XPath を作成するようにユーザに促します。必要に応じて、「*CA Service Virtualization の使用*」を参照して手順を完了します。 最後の手順は、レコーダが記録されたもので後処理を実行していることを示します。

6. ［終了］をクリックします。

   トランザクションがサービス イメージに保存されます。また、仮想サービス モデルが作成されます。

仮想サービスを実行する方法

1. 元のサービスをシャットダウンします。
2. DevTest ワークステーション に移動し、作成した仮想サービス モデルを展開します。
3. サービスとして仮想サービスを持ったクライアント アプリケーションを実行します。

Web ベース レコーダの使用によるデータベースの仮想化

JDBC ベースのデータベース トラフィックを仮想化するために、Web ベース レコーダが含まれる DevTest Java Agent を使用できます。 この機能は、*エージェントベースの JDBC 仮想化*とも呼ばれます。

注: JDBC は通信が多いプロトコルです。たとえば、データベースからデータを取得するためにユーザ インターフェースで数回クリックしただけで、大量の JDBC API コールが発生する場合があります。 この動作により、適切にモデル化することが難しくなる可能性があります。 JDBC データベース トラフィックをキャプチャして仮想化する場合は、より高いレイヤで行うことを検討してください。

以下の JDBC ドライバおよびデータベースがサポートされています。

- Apache Derby
  - Apache Derby データベースに対応する Apache Derby 4.2.3 JDBC ドライバ
- Oracle
  - Oracle 11*g* データベースに対応する Oracle 11*g* JDBC ドライバ
  - Oracle 12*c* データベースに対応する Oracle 11*g* JDBC ドライバ
- IBM DB2
  - IBM DB2 9.5 データベースに対応する IBM DB2 9.5 JDBC ドライバ
  - IBM DB2 9.8 データベースに対応する IBM DB2 9.5 JDBC ドライバ
  - IBM DB2 10.5 データベースに対応する IBM DB2 9.5 JDBC ドライバ
- Microsoft SQL Server
  - Microsoft SQL Server 2008 R2 データベースに対応する Microsoft JDBC ドライバ 4.0
  - Microsoft SQL Server 2012 データベースに対応する Microsoft JDBC ドライバ 4.0
  - Microsoft SQL Server 2008 R2 データベースに対応する jTDS 1.3.1 JDBC ドライバ
  - Microsoft SQL Server 2012 データベースに対応する jTDS 1.3.1 JDBC ドライバ

JDBC トラフィックをキャプチャする前に、アプリケーションがデータベースへの JDBC コールを行うコンピュータに、エージェントをインストールおよび設定します。 データベース自体は個別のコンピュータに存在できます。 エージェントを設定する場合、必ず以下のアクションを実行します。

- JDBC プロトコルのキャプチャ レベル (P. 27) を［全データ］に設定します。
- JDBC プロトコルの上の任意のプロトコルのキャプチャ レベルを［カウント数およびパス］または［全データ］に設定します。

重要: 再生時には、テスト中のシステムを起動する *前*に、仮想サービスを開始します。

注: エージェントのインストールの詳細については、「エージェント」を参照してください。 仮想サービスの展開については、「*CA Service Virtualization の使用*」を参照してください。

JDBC トラフィックのキャプチャ

DevTest ポータルの［Virtualize JDBC］ウィンドウには、以下のビューが含まれます。

- Entity View
- Conversation View

この手順では、Entity View が最初に表示されると仮定します。

注: レコーディングを停止した後に新しいレコーディングを開始すると、以前のレコーディングのクエリはすべて破棄されます。

次の手順に従ってください:

1. 左のナビゲーション メニューで、［Create］-［Virtualize JDBC］を選択します。

   ［Virtualize JDBC］ウィンドウが表示されます。
2. ［Start recording］ボタンをクリックします。
3. 仮想化するクエリが作成されるアプリケーションの一部を実行します。

   一意のクエリがキャプチャされるたびに、そのクエリが［Results］ペインに表示されます。
4. （オプション）キャプチャされたクエリを検査します (P. 187)。
5. （オプション）会話をプレビューします (P. 188)。
6. 必要なクエリおよび会話がすべて揃ったら、［Stop recording］ボタンをクリックします。
7. ［Create VS Artifact］をクリックします。

   仮想サービスが作成されます。

キャプチャされたクエリの検査

JDBC トラフィックのキャプチャ中に、キャプチャされたクエリに関する詳細を検査できます。

また、フィルタ機能を使用して、表示されるクエリの範囲を絞り込むこともできます。

注: フィルタリングは、仮想化されるクエリの対象範囲には影響しません。

次の手順に従ってください:

1. クエリのインスタンスを表示するには、［Results］ペイン内でクエリを選択します。

   ［SQL クエリ］ペインおよび［Transactions］ペインにデータが入力されます。

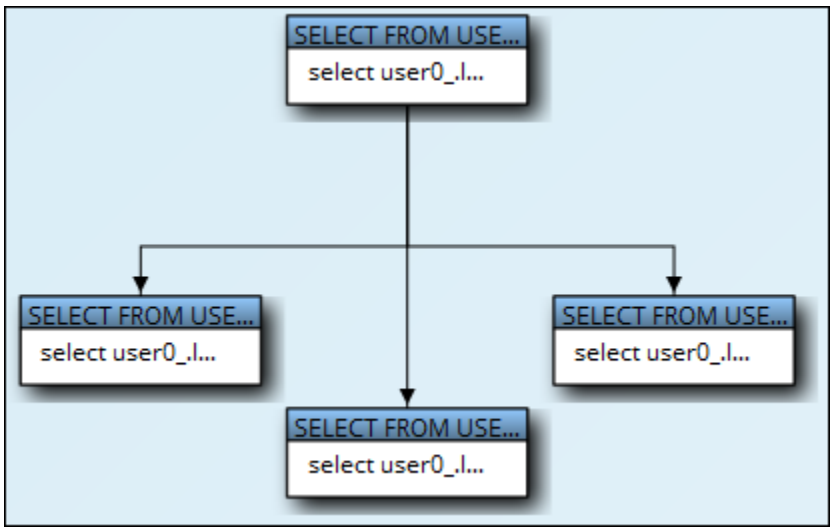
2. クエリをフィルタするには、［Databases］セクションに移動し、対象に含めるデータベースのチェック ボックスをオンにします。

会話のプレビュー

仮想サービスには、会話とステートレス トランザクションの両方を含めることができます。

JDBC トラフィックのキャプチャ中に、生成される仮想サービスの会話をプレビューできます。

以下の図は、会話の例を示しています。



全体的な構造についての小型のビューを右上隅に表示できます。 この機能は、大量の会話の場合に特に有用です。

次の手順に従ってください:

1. ドロップダウン リストで、［Conversation View］を選択します。
2. 全体的な構造についての小型のビューを表示するには、［Show Outline］をクリックします。
3. 会話でのクエリに関する詳細情報を表示するには、ノードをクリックします。

JDBC サービス イメージ

エージェントベースの JDBC 仮想化では、生成する仮想サービスにサービス イメージが含まれます。

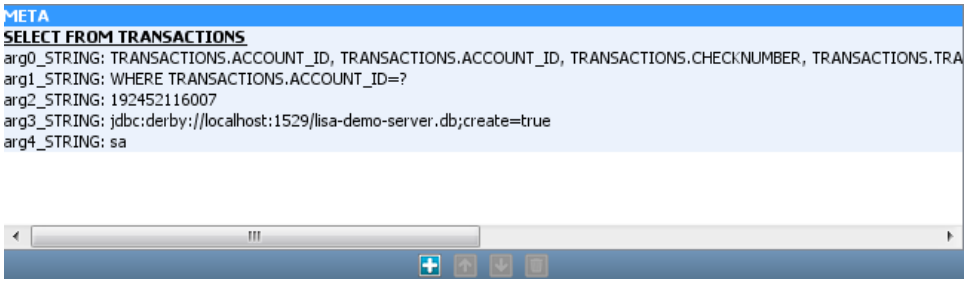
サービス イメージ内の各トランザクションには、1 つの操作と一連の引数があります。

操作には、SQL ステートメントの主な目的が含められます。以下に例を示します。

```
SELECT FROM TRANSACTIONS
```

引数には、列のリスト、SQL ステートメントの主要な句、および SQL ステートメントにバインドされるパラメータ値を含めることができます。 最後の 2 つの引数は、常に接続 URL とデータベース ユーザです。

以下の図は、トランザクションを示しています。この操作は、SELECT FROM TRANSACTIONS です。最初の引数には、列のリストが含まれます。この SQL ステートメントには、1 つのパラメータ値（アカウント ID）があります。



また、サービス イメージ内の各トランザクションには、SQL ステートメントを実行した結果が含まれる XML ドキュメントがあります。 この XML ドキュメントは、サービス イメージ エディタの [応答] パネルにあります。

結果は、SQL ステートメントの種類によって異なります。

- SQL ステートメントがデータを取得した場合、結果には 1 つ以上のデータ セットが含まれます。
- SQL ステートメントがデータを更新した場合、結果には更新数が含まれます。

複数のデータ セット エディタ

トランザクションがサービス イメージで選択されている場合、［応答］パネルで SQL ステートメントを実行した結果を表示できます。

また、表示されるデータを編集できます。 たとえば、コール元のコードがデータベースから返されたデータをどのように処理するのかをテストするために、結果セット内の値を変更できます。

［応答］パネル内のデフォルトのエディタの名前は、MultiDataSet ドキュメントです。 このエディタは、以下のコンポーネントで構成されています。

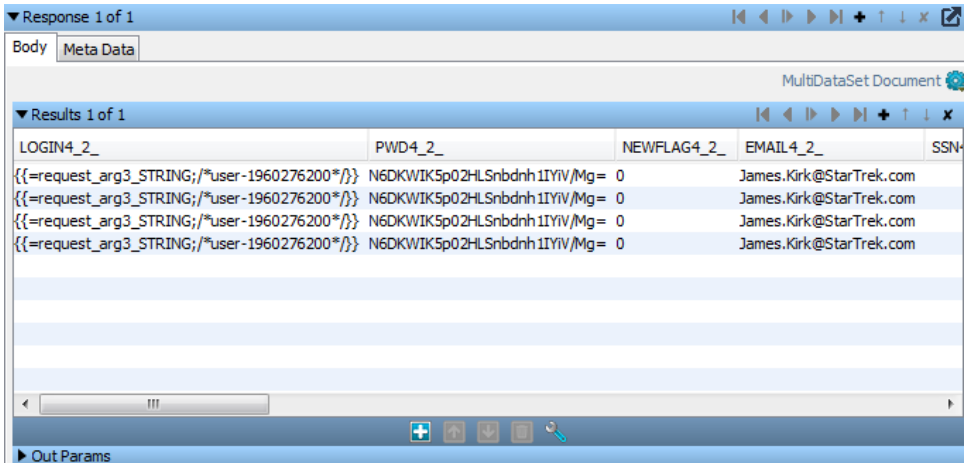
- ［結果］パネル
- ［出力パラメータ］パネル

［結果］パネル

［結果］パネルには、任意の数の結果セットおよび任意の数の更新数を含めることができます。 たとえば、以下の内容はすべて有効です。

- 1 つの結果セット
- 1 つの更新数
- 1 つの結果セットおよび 1 つの更新数
- 3 つの結果セット

以下の図は、結果セットの例を示しています。

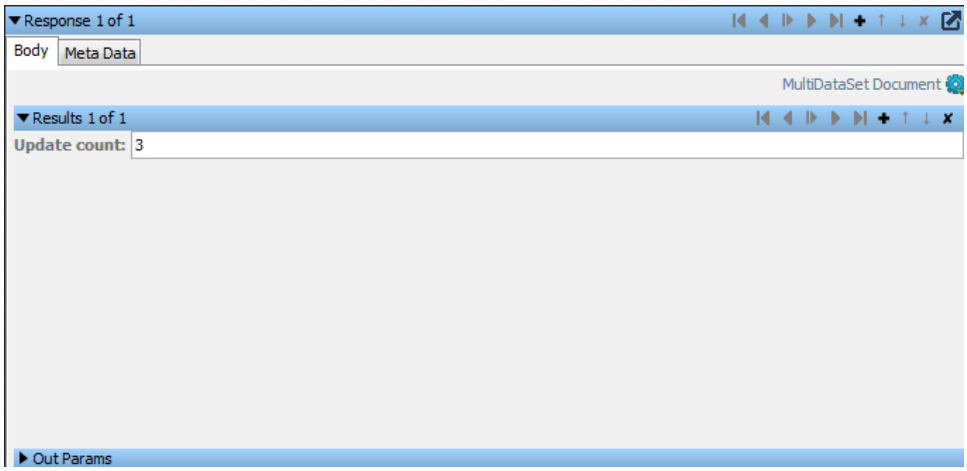


結果セットを追加および削除できます。

結果セットでは、以下のアクションを実行できます。

- 値の変更
- 行の追加
- 行の上下への移動
- 行の削除
- ラベルおよびデータ型などの列メタデータの表示および編集

以下の図は、更新数の例を示しています。



更新数を追加および削除できます。

更新数では、値を変更できます。 値は、整数またはプロパティ式である必要があります。

[出力パラメータ]パネル

ストアド プロシージャ コールに出力パラメータがある場合、［出力パラメータ］パネルが使用されます。

2 種類の出力パラメータを追加できます。

- シンプル
- 結果セット

シンプル出力パラメータには、キーおよび値があります。

結果セット出力パラメータには、キーおよび結果セットがあります。

テスト中のシステムは、インデックス番号または文字列名によってパラメータにアクセスします。 インデックス番号または文字列名は、キーで指定します。

JDBC 仮想サービス モデル

エージェントベースの JDBC 仮想化では、生成する仮想サービスに仮想サービス モデルが含まれます。

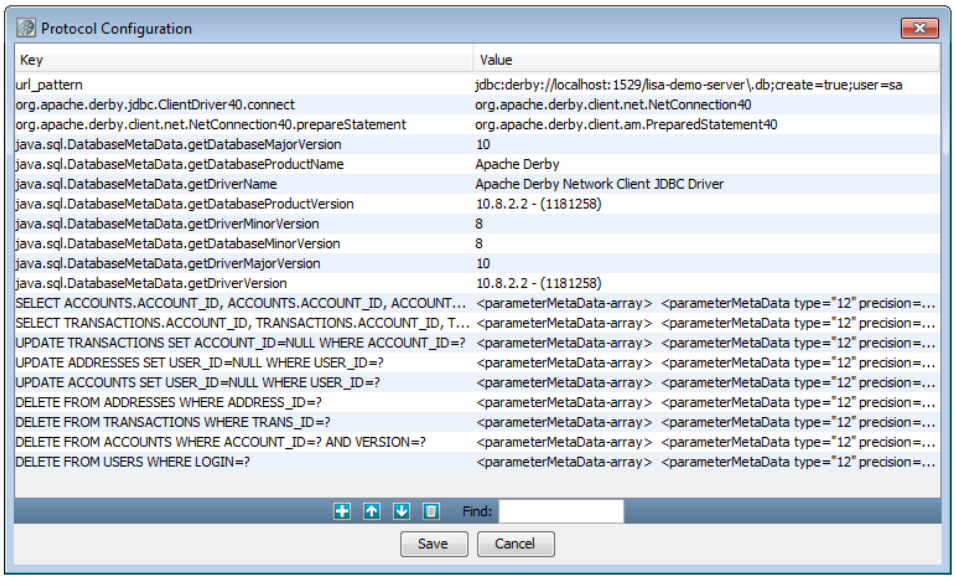
仮想サービス モデルには、Java VSE と同じリスナおよびレスポンダ ステップがあります。

- 仮想 Java リスナ
- 仮想 Java レスポンダ

プロトコル設定情報を表示するには、リスナ ステップを開き、右下のリスト内の JDBC プロトコルをダブルクリックします。 [プロトコル設定] ダイアログ ボックスが表示されます。 このダイアログ ボックスには、1 セットのキー/値ペアが含まれます。

重要: どの行も削除しないでください。行を削除した場合、テスト中のシステムは正しく動作しません。

以下の図は、[プロトコル設定] ダイアログ ボックスを示しています。この例は、Apache Derby データベースに基づいています。



url_pattern キーは、再生時に仮想化をトリガする接続 URL を指定します。値が、セミコロンの後にテキスト **user=** およびデータベース ユーザが続く形で終わる場合、仮想化はそのデータベース ユーザに制限されます。 追加のデータベース ユーザ用のキー/値ペアを追加できます。 各ユーザのキーは、**url_pattern_** から始まる必要があります。

2 番目のキーは、ドライバ クラス名と **connect** メソッドで構成されます。値は接続クラスの名前です。

3 番目のキーは、接続クラス名および **prepareStatement** メソッドで構成されます。 値は準備したステートメント クラスの名前です。

2 番目および 3 番目のキーがステートメントへの接続へのドライバの「チェーン」をどのように作成するかに注目してください。

java.sql.DatabaseMetaData で始まるキーには、以下のものの名前およびバージョン情報が含まれます。

- データベース ドライバ
- SQL トラフィックがキャプチャされたときにそれが通信していたデータベース

下部のキーには、特定の SQL ステートメントとステートメントが表すパラメータ メタデータのマッピングが含まれます。

一致例外

［**一致しない場合に例外をスローする**］プロパティでは、仮想サービス モデルに SQL ステートメント用の応答がない場合の動作を制御できます。デフォルトでは、このプロパティは有効です。

- このプロパティが有効な場合、エージェントは仮想化されることに失敗した SQL ステートメントを示す SQL 例外をスローします。 また、例外は、ユーザがより多くの JDBC フレームをキャプチャし、サービス イメージにそれらを追加することを提案します。 このアクションは、新しい仮想化アーティファクトを生成し、DevTest ワークステーションまたは **ServiceImageManager** コマンドライン ツールのいずれかでサービス イメージの組み合わせ機能を使用することを必要とします。
- このプロパティが無効な場合、エージェントは適切な空の結果と列定義に関する最適な推測を提供します。

このプロパティは、DevTest ポータルの［エージェント］ウィンドウ (P. 19) から設定できます。 プロパティは［設定］タブに表示されます。

データベース仮想化のトラブルシューティング

問題の状況:

Oracle データベースを仮想化しようとしています。 再生モードで JDBC トランザクションを実行すると、VSE コンソールでトランザクション数が増加しません。

解決方法:

エージェントのログ ファイルを開きます。 以下の例外を検索します。

```
java.lang.RuntimeException: com.itko.javassist.NotFoundException:
oracle.xdb.XMLType
    場所：
itko.javassist.CtClassType.getClassFile2(CtClassType.java:202)
    場所：
com.itko.javassist.CtClassType.getSuperclass(CtClassType.java:746)
```

例外を見つけたら、以下の手順に従います。

1. アプリケーションおよび仮想サービスを停止します。
2. 仮想サービスを開始します サービスが［実行中］状態であることを確認します。
3. テスト アプリケーションを起動します。
4. 再度 JDBC トランザクションを実行します。

   トランザクション数は増加します。

または、**xdb.jar** ファイルをダウンロードして、アプリケーションのクラスパスに配置することもできます。

第 9 章: テストのトランザクションの文書化

ビジネス トランザクションの文書化によって、システム アーキテクチャおよびデータのフローが可視化されます。 この可視化によって、バックエンド コードおよびデータベースで発生していることが明らかになります。

テスト トランザクションを記録し、生成されたバックエンド コールを分析して、テストを文書化します。1 つ以上のテストを比較できます。 トランザクションは青色のピンでマークされてグラフィカル ビューに表示され、レコーディング中に発生した例外が特定されます。 記録されたテスト ケースの名前が、ホーム ページの［Points of Interest］リスト、および［Explore Defects］ウィンドウに表示されます。

［Document Transactions］ウィンドウから、テストのトランザクションを文書化します。 以下のタスクを実行できます。

- テスト トランザクションの記録 (P. 199)
- CAI データベースでの保存されたトランザクション レコーディングの検索 (P. 200)
- リスト ビューまたはグラフィカル ビューでのレコーディングの表示および分析 (P. 201)
- 記録されたトランザクションからのドキュメントの作成 (P. 202)

このセクションには、以下のトピックが含まれています。

手動テストのトランザクションの文書化方法 (P. 198)

手動テストのトランザクションの文書化方法

このシナリオでは、手動テストのトランザクションを文書化する手順を実行します。

1. テスト トランザクションの記録 (P. 199)
2. 記録されたテスト トランザクションの検索 (P. 200)
3. 記録されたテスト トランザクションの表示および分析 (P. 201)
4. 記録されたトランザクションからのドキュメントの作成 (P. 202)

テストトランザクションの記録

レコーディング機能を使用して、特定の IP アドレスのビジネス トランザクションで発生する例外を文書化します。

次の手順に従ってください:

1. 左側のナビゲーション メニューで、［Application Insight］-［Document Transactions］を選択します。

   ［Document Transactions］ウィンドウが表示されます。

2. ［Create a Recording］をクリックすると、新しいケースのレコーディングが開始します。

   ［Record Case］ダイアログ ボックスが表示されます。デフォルトでは、ブラウザを使用して DevTest サーバ に接続しているシステムの IP アドレスが表示されます。

   ブラウザを実行し、リモート システムからエージェント アプリケーションに接続している場合は、IP アドレスを変更できます。

3. （オプション）キャプチャされたトランザクションを自動的に表示するには、［Auto Refresh］を選択します。

4. テスト ケースの名前を入力します。

   ALM のテスト識別子を名前に含めることをお勧めします。この名前が、ホーム ページの［Points of Interest］ポートレットのリストに表示されます。

5. ［Start］をクリックします。

   ［Start］ボタンがアニメーション表示の［Recording］アイコンに変わり、レコーディングが進行中であることが示されます。

6. 同じシステム上またはリモート システム上のいずれかにあるエージェント アプリケーションを、別のブラウザ ウィンドウで実行します。

7. 記録されているエージェント システムに関して、必要な作業を実行します。

   記録されたフレームが、［Captured Frames］ペインに表示されます。

   記録されている IP アドレスのフレームに、紫色のピンが追加されます。［Auto Refresh］が選択されている場合、キャプチャされたフレームは、［Captured Frames］ペインに自動的に表示されます。 そうでない場合は、［Refresh］ボタンをクリックします。

8. （オプション）記録されたトランザクションをグラフィカル ビューで表示するには、  をクリックします。
9. （オプション）記録されたトランザクションをリスト ビューで表示するには、  をクリックします。
10. （オプション）大量のトランザクションの一部を一度に表示するには、［Show Outline］をクリックしてマップを移動します。
11. ［Stop］をクリックします。
12. レコーディングを保存するには、［Save］をクリックします。

    保存されたレコーディングは、［Document Transactions］ウィンドウの［Recording］ペインにリスト表示されます。
13. 記録されたトランザクションのドキュメントを作成するには、［Create Document］をクリックします。

記録されたテストトランザクションの検索

［Document Transactions］ウィンドウで検索機能を使用して、分析するレコーディングのリストを絞り込みます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで、［Application Insight］-［Document Transactions］を選択します。
2. ［Enter Search Text］フィールドにテキスト検索基準を入力し、［検索］をクリックします。

記録されたテストトランザクションの表示および分析

［Document Transactions］ウィンドウの［レコーディング］ペインには、記録されて保存されたすべてのテスト トランザクションのリストが表示されます。記録されたテストをグラフィカル ビュー、またはリスト ビューで表示して分析できます。

次の手順に従ってください:

1. 左側のナビゲーション メニューで、［Application Insight］-［Document Transactions］を選択します。
2. ［Recording］ペイン内のリストから、保存されたテスト レコーディングを選択します。

   デフォルトでは、テスト トランザクションが下部ペインにグラフィカル ビューで表示されます。
3. トランザクション詳細を表示するには、ツールヒントを表示します。
4. トランザクション詳細をリスト ビューで表示するには、  をクリックします。
5. レコーディングを削除するには、  をクリックします。

記録されたトランザクションからのドキュメントの作成

［Document Transactions］ウィンドウから、テスト中に記録されたトランザクションのレポートを作成できます。 レポートには、記録された各フレームに関する情報が含まれます。 レポートを保存して印刷できます。

次の手順に従ってください:

1. ［Document Transaction］ウィンドウから、テストを記録して実行します。
2. ［Create Document］をクリックします。
3. レポートのタイトルを入力して、［OK］をクリックします。

   レポートが生成されて、ブラウザで表示されます。

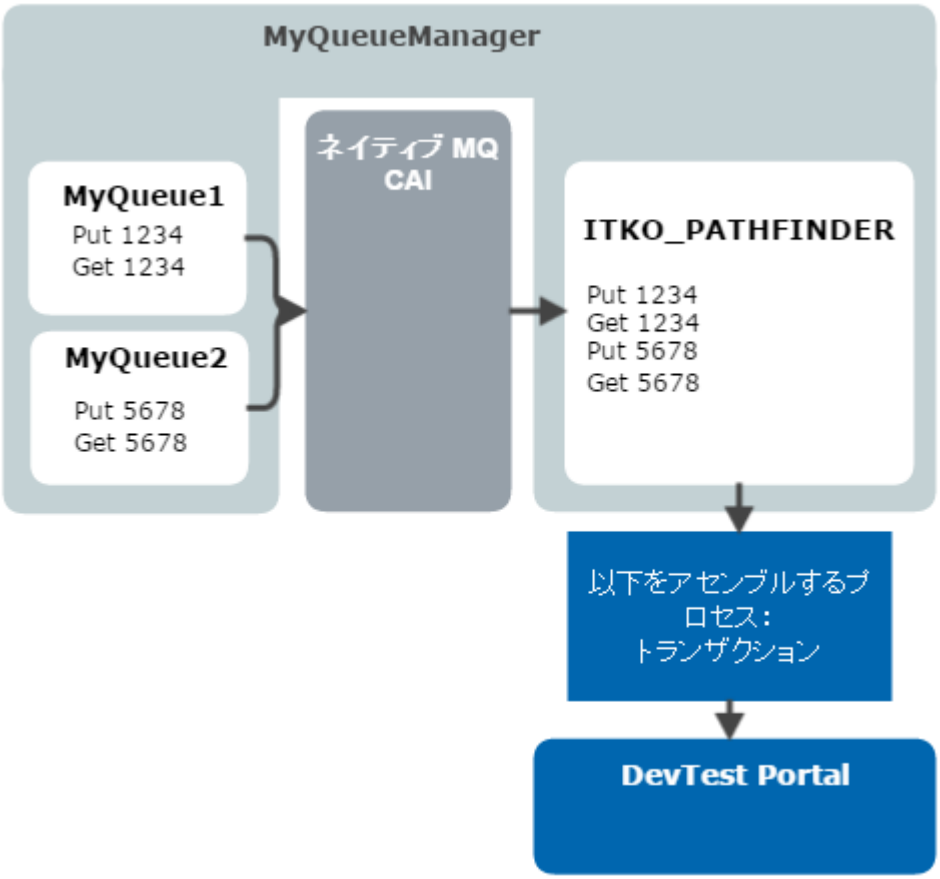
   注: トランザクションの数によっては、レポートの生成に数分かかる場合があります。
4. レポートを保存するには、マウスをブラウザの下部右側に移動してタスク バーを表示し、［保存］ボタンをクリックして、フィールドに入力します。
5. 印刷するには、［印刷］ボタンをクリックしてフィールドに入力します。

第 10 章: ネイティブ MQ CAI

ネイティブ MQ CAI は、キュー マネージャのキューに対する get コールと put コールをモニタする WebSphere MQ API 出口です。 ネイティブ MQ CAI は、ITKO_PATHFINDER という名前の特別なキューに get コールと put コールをコピーします。 DevTest ポータルで表示できるトランザクションをアセンブルするために、個別の DevTest コンポーネントは ITKO_PATHFINDER キューのデータを使用します。 その後、ベースライン、仮想サービスなどを作成するためにトランザクションを使用できます。

この機能は、WebSphere MQ 6.0 および 7.0 に対してサポートされています。

以下の図は、ネイティブ MQ CAI がどのように動作するかの例を示しています。 キュー マネージャには 3 つのキュー（MyQueue1、MyQueue2、および ITKO_PATHFINDER）があります。 ITKO_PATHFINDER キューには、MyQueue1 および MyQueue2 の get コールと put コールのコピーがあります。 データはトランザクションに変換され、DevTest ポータルに送信されます。



ネイティブ MQ CAI は、関連しない特定のタイプのキューを無視します。たとえば、**SYSTEM.** で始まる名前のキューは無視されます。

MQPUT および MQPUT1 のコールが両方ともサポートされています。

このセクションには、以下のトピックが含まれています。

ITKO_PATHFINDER キューの作成

ネイティブ MQ CAI は、get コールと put コールのコピーを ITKO_PATHFINDER という名前のキューに配置します。

キューがいっぱいになった場合は、最も古いメッセージが削除されます。

次の手順に従ってください:

1. WebSphere MQ Explorer に移動します。
2. ITKO_PATHFINDER という名前のキューを作成します。 このキューは、モニタするキューと同じキュー マネージャによって所有する必要があります。 永続設定、キューの最大深度などは、任意に指定します。

ネイティブ MQ CAI API 出口のインストール

LISA_HOME¥agent¥native_mq ディレクトリには、さまざまなオペレーティング システム用の API 出口ファイルが含まれます。

次の手順に従ってください:

1. WebSphere MQ をシャットダウンします。
2. LISA_HOME¥agent¥native_mq ディレクトリに移動します。
3. オペレーティング システムに応じて、適切な場所に出口をコピーします。

   Windows

   MQ_HOME¥exits ディレクトリに **iTKO_MQ_Pathfinder.dll** ファイルをコピーします。

   Linux

   /var/mqm/exits ディレクトリに **iTKO_MQ_Pathfinder_linux** および **iTKO_MQ_Pathfinder_linux_r** ファイルをコピーします。 これらのファイルは、mqm ユーザが所有している必要があります。 これらのファイルには、読み取りと実行のビット セット（chmod a+rx）が必要です。

   Linux 64 ビット

   /var/mqm/exits64 ディレクトリに **iTKO_MQ_Pathfinder_linux_x64** および **iTKO_MQ_Pathfinder_linux_x64_r** ファイルをコピーします。 これらのファイルは、mqm ユーザが所有している必要があります。 これらのファイルには、読み取りと実行のビット セット（chmod a+rx）が必要です。

   SUSE Linux 64 ビット

   /var/mqm/exits64 ディレクトリに **iTKO_MQ_Pathfinder_sles10.3_x64** および **iTKO_MQ_Pathfinder_sles10.3_x64_r** ファイルをコピーします。 これらのファイルは、mqm ユーザが所有している必要があります。 これらのファイルには、読み取りと実行のビット セット（chmod a+rx）が必要です。

   Solaris

/var/mqm/exits ディレクトリに **iTKO_MQ_Pathfinder_sol_sparc** ファイルをコピーします。 このファイルは、mqm ユーザが所有している必要があります。 このファイルには、読み取りと実行のビットセット（chmod a+rx）が必要です。

Solaris 64 ビット

/var/mqm/exits64 ディレクトリに **iTKO_MQ_Pathfinder_sol_sparc.64** ファイルをコピーし、.64 拡張子を削除します。 このファイルは、mqm ユーザが所有している必要があります。 このファイルには、読み取りと実行のビット セット（chmod a+rx）が必要です。

AIX

/var/mqm/exits ディレクトリに **iTKO_MQ_Pathfinder_aix** および **iTKO_MQ_Pathfinder_aix_r** ファイルをコピーします。これらのファイルは、mqm ユーザが所有している必要があります。 これらのファイルには、読み取りと実行のビット セット（chmod a+rx）が必要です。

AIX 64 ビット

/var/mqm/exits64 ディレクトリに **iTKO_MQ_Pathfinder_aix.64** および **iTKO_MQ_Pathfinder_aix_r.64** ファイルをコピーし、.64 拡張子を削除します。 これらのファイルは、mqm ユーザが所有している必要があります。 これらのファイルには、読み取りと実行のビットセット（chmod a+rx）が必要です。

4. WebSphere MQ を再起動します。

ネイティブ MQ CAI API 出口の設定

API 出口をインストールした後、モジュールおよびエントリ ポイントなどの情報を指定します。

次の手順に従ってください:

1. WebSphere MQ Explorer に移動するか、または **qm.ini** 設定ファイルを開きます。
2. API 出口を実行するためにキュー マネージャを設定します。
   - Name 属性を任意のわかりやすい名前に設定します。
   - Function 属性を **EntryPoint** に設定します。この値は、大文字と小文字が区別されます。
   - Module 属性をインストールした API 出口ファイル（例：**/var/mqm/exits/iTKO_MQ_Pathfinder_linux**）に設定します。
   - ネイティブ MQ CAI でログ ファイルを生成する場合は、Data 属性をログ ファイルが書き込まれるディレクトリに設定します。
3. キュー マネージャを再起動します。
4. キューにテスト メッセージを送信します。
5. ITKO_PATHFINDER キューを参照し、テスト メッセージが反映されたことを確認します。

例: AIX の qm.ini API 出口の設定

以下の行を **/var/mqm/qmgrs/[QueueManager]/qm.ini** に追加します。**[QueueManager]** は、使用している環境でのキュー マネージャの名前です。

```
ApiExitLocal:
   Module=/var/mqm/exits64/iTKO_MQ_Pathfinder_aix
   Name=Pathfinder
   Sequence=100
   Function=EntryPoint
```

この例は、64 ビットのカーネル パッケージがインストールされている AIX システムに適用されます。 システムが純粋な 32 ビットである場合は、Module 属性を **/var/mqm/exits/iTKO_MQ_Pathfinder_aix** に変更します。

/var/mqm/exits/（32 ビット ライブラリ）および **/var/mqm/exits64/**（64 ビット ライブラリ）の両方に、**iTKO_MQ_Pathfinder_aix** および **iTKO_MQ_Pathfinder_aix_r** をコピーすることもできます。**Module=iTKO_MQ_Pathfinder_aix** のように、相対パスを使用して Module 属性を設定します。このアクションにより、グローバル WebSphere MQ 設定ファイル（**/var/mqm/mqs.ini**）で定義されているデフォルト ClientExitPath 属性パスに基づいて、WebSphere MQ がライブラリの正しいバージョンを取得できます。 **/var/mqm/mqs.ini** ファイルは、以下の例のようなものです。

```
   DefaultPrefix=/var/mqm
ClientExitPath:
   ExitsDefaultPath=/var/mqm/exits
   ExitsDefaultPath64=/var/mqm/exits64
```

ネイティブ MQ CAI 用のエージェント プロパティの設定

DevTest Java エージェントには、設定する必要があるネイティブ MQ CAI プロパティのセットが含まれます。

デフォルトでは、ネイティブ MQ CAI は、キュー マネージャのすべてのキューをモニタします。 この動作を上書きするために、［キューを含める］および［キューを除外］プロパティを使用できます。このプロパティ値は正規表現です。 ［キューを除外］プロパティは、［キューを含める］プロパティより優先されます。 ドット文字が含まれるキュー名を指定する場合は、ドット文字の直前に円記号を配置します。 ドット文字を正規表現の要素として使用する場合は、円記号を含める必要はありません。

テスト中のシステムが特殊なパターンを使用する場合、ネイティブ MQ CAI は、どのキュー/メッセージが要求でどのキュー/メッセージが応答かを判断するために手助けを必要とする場合があります。 このシナリオでは、**rules.xml** ファイルにオプションの **QUEUE_REQUEST_MATCHES** および **QUEUE_RESPONSE_MATCHES** プロパティを追加してみます。値を正規表現に設定します。正規表現はメッセージ ペイロードに対して評価されます。以下に例を示します。

```
<property key="QUEUE_REQUEST_MATCHES:SOME_REQUEST_QUEUE" value=".*"/>
<property key="QUEUE_RESPONSE_MATCHES:SOME_RESPONSE_QUEUE"
value=".*<response>.*"/>
```

次の手順に従ってください:

1. ［エージェント］ウィンドウを開きます。
2. 左側の部分で、エージェントを選択します。
3. ［設定］タブをクリックします。
4. MQMirror カテゴリを選択します。
5. ［キュー名］プロパティを **ITKO_PATHFINDER** に設定します。 この値は唯一の有効な値です。
6. ［キュー マネージャ名］プロパティをモニタするキューを所有するキュー マネージャに設定します。
7. ［ホスト］プロパティを WebSphere MQ が実行されているサーバの IP アドレスまたはホスト名に設定します。
8. ［ポート］プロパティを WebSphere MQ が接続要求をリスンしているポート番号に設定します。

9. ［チャネル名］プロパティをキュー マネージャに接続するための WebSphere MQ チャネルに設定します。
10. （オプション）［接続間隔］プロパティを設定します。
11. WebSphere MQ にアクセスするためにユーザおよびパスワードを必要とする場合は、［ユーザ］および［パスワード］プロパティを設定します。
12. 含めるキューを指定する場合は、［キューを含める］プロパティを設定します。
13. 除外するキューを指定する場合、［キューを除外］プロパティを設定します。
14. ［保存］をクリックします。

ITKO_PATHFINDER キューからのトランザクションの生成

LisaAgent.jar ファイルには、ITKO_PATHFINDER キューに反映されたメッセージに基づくトランザクションを作成するためのオプションが含まれます。

ブローカが別のコンピュータで実行されている場合、ブローカの URL を使用して **-u** オプションを指定する必要があります。

次の手順に従ってください:

1. WebSphere MQ JAR ファイルを、**LisaAgent.jar** ファイルがあるディレクトリにコピーします。 JAR ファイルのリストについては、「*管理*」の「サードパーティ ファイル要件」の WebSphere MQ に関するセクションを参照してください。
2. 以下のコマンドを実行します。
   ```
   java -jar LisaAgent.jar -m
   ```
   メッセージは ITKO_PATHFINDER キューから取得され、トランザクションにアセンブルされます。 トランザクションは、DevTest ポータルで表示できます。

第 11 章: CAI コマンドライン ツール

PFCmdLineTool コマンドでは、コマンドラインからさまざまな CA Continuous Application Insight タスクを実行できます。

主なオプションは、**--count**、**--roots**、**--paths**、**--export**、**--import**、**--baseline**、および **--virtualize** です。

コマンドの形式は以下のとおりです。

```
PFCmdLineTool
[--count|--roots|--paths|--export|--import|--baseline|--virtualize|--help|--version] [タスク固有のオプション] [検索条件]
```

検索条件

--import 以外の主要なすべてのオプションでは、操作されるトランザクション フレームのセットは指定した検索条件に一致するオプションによって定義されます。 検索条件を指定するには、**--from**、**--to**、**--localIP**、**--remoteIP**、**--category**、**--class**、**--method**、**--session**、**--transaction**、**–agent**、**--min-time**、**--tag**、**--max-frames** の中から任意のオプションを使用します。これらのオプションは、操作されるルート トランザクション フレームのセットを絞り込むために使用されます。

また、**--frame** オプションを使用して検索結果を単一のトランザクション フレームに制限することもできます。

--from=**開始時刻**

必要なトランザクション フレームのウィンドウの開始時刻を指定します。 **yyyy-mm-dd** または **yyyy-mm-ddThh:mm:ss** のいずれかの形式を使用します。 このオプションおよび **--frame** オプションが指定されない場合、現在の日の開始時刻が使用されます。

--to=**終了時刻**

必要なトランザクション フレームのウィンドウの終了時刻を指定します。 **yyyy-mm-dd** または **yyyy-mm-ddThh:mm:ss** のいずれかの形式を使用します。 このオプションおよび **--frame** オプションが指定されない場合、現在の日の終了時刻が使用されます。

--localIP=IP **アドレス**

CAI が記録するクライアント側 IP アドレスを指定します。

--remoteIP=IP **アドレス**

CAI が記録するサーバ側 IP アドレスを指定します。

--category=**カテゴリ**

トランザクション フレームのタイプを指定します。このオプションを繰り返して指定することにより、複数のカテゴリを一度に検索できます。

値： amx、client、dn_default、dn_remoting、dn_sql、ejb、gui、jca、jdbc、jms、logging、mq、rest_http、rmi、rmi_http、rmi_ssl、thread、throwable、tibco、web_http、web_https、wm、wps、ws_http、ws_https

--class=**クラス名**

クラス名を指定します。 この値は、完全なクラス名か、または「%」で終了する文字列（「前方一致」検索を実行）のいずれかになります。

--method=**メソッド名**

メソッド名を指定します。この値は、完全なメソッド名か、または「%」で終了する文字列（「前方一致」検索を実行）のいずれかになります。

--session=**セッション** ID

セッション識別子を指定します。

--transaction=**トランザクション** ID

トランザクション識別子を指定します。

--agent=**エージェント名**

必要なトランザクション フレームに対するソースであるエージェントの名前を指定します。

--min-time=min-time

参照するトランザクション フレームの最小実行時間（ミリ秒）を指定します。 この値より速く実行されるフレームはフィルタで除外されます。

--tag=**名前**、--tag=**名前**=**値**

トランザクション フレームと関連付けられたフレーム タグを指定します。名前のみ、または名前と値の両方を指定できます。このオプションを繰り返すことにより、複数のタグを一度に検索できます。

--max-frames=**最大フレーム数**

取得されるルート トランザクション フレームの最大数を指定します。このオプションが指定されない場合、デフォルトで 10000 になります。この最大数より多くのフレームが処理される場合があります。

--frame=frame-id

必要な特定のトランザクション フレームの識別子を指定します。

クエリの実行および表示

以下のオプションにより、トランザクションに対してクエリを実行し、トランザクションを表示できます。

-c、--count

指定された検索条件に一致するルート トランザクション フレームの数を表示します。

-r、--roots

指定された検索条件に一致するルート トランザクション フレームを表示します。

-p、--paths

ルート トランザクション フレームのセットのトランザクション フレーム階層を表示します。

エクスポートおよびインポート

以下のオプションにより、パスをエクスポートおよびインポートできます。

-e 出力 ZIP ファイル名、--export=出力 ZIP ファイル名

選択されたトランザクション フレームをデータベースからエクスポートします。

-i 入力 ZIP ファイル名、--import=入力 ZIP ファイル名

トランザクションをデータベースにインポートします。指定した検索条件は無視されます。

--no-persist

インポート操作でデータを保持しないように指定します。インポート ファイルの検証のみが実行されます。

注: DevTest がデフォルトの Derby 以外のデータベースに接続される場合、**rules.xml** ファイルの broker エレメントにデータベースを指定する必要があります。

<database driver="yourdatabasedriver" url="yourdatabaseurl" user="yourdatabaseuser" password="yourdatabasepassword"/>

この設定は、エージェントのライフスパンにおいて使用されます。

ベースライン

以下のオプションにより、ベースライン テスト ケースおよびスイートを作成できます。 **--to-dir** オプションまたは **--to-project** オプションは必須です。

-b **名前**、--baseline=**名前**

選択されたトランザクション フレームに対するベースライン テストを生成します。

--refer=**フレーム** ID

キー トランザクション フレームとして指定される特定のトランザクション フレームの識別子を指定します。このオプションが指定されない場合、クエリ結果内の最初のトランザクション フレームがキー トランザクション フレームとして指定されます。

--consolidated

ベースライン テスト ケースを生成するように指定します。 このオプションが指定されない場合、ベースライン スイートが生成されます。

--stateful

ステートフル ベースラインが生成されることを指定します。

--force-tf-steps

ベースライン テストをトランザクション フレームの実行ステップのみを使用して生成するように指定します。 このオプションが指定されない場合、使用可能なすべての CA Application Test テスト ステップが可能な限り使用されます。

注: トランザクション フレームの実行ステップの詳細については、「*CA Application Test の使用*」を参照してください。

--magic-dates

ベースライン テストでマジック デート処理が適用されることを指定します。

--jndi-factory=**ファクトリ クラス名**

EJB ベースライン テストを生成するときに使用する JNDI ファクトリ クラスを指定します。

--jndi-url=URL

EJB ベースライン テストを生成するときに使用する JNDI URL を指定します。

--jndi-user=**ユーザ ID**

EJB ベースライン テストを生成するときに使用する JNDI ユーザを指定します。

--jndi-user-cred=**ユーザ認証情報**

EJB ベースライン テストを生成するときに使用する JNDI ユーザ認証情報を指定します。

--to-dir=**出力 PFI ファイル名**

生成された PFI ファイルが書き込まれるディレクトリの名前を指定します。

--to-project=LISA **プロジェクト ディレクトリ**

中間のインポート ファイルを必要とせずに、生成されたアーティファクトが書き込まれるプロジェクトのルート ディレクトリを指定します。**--to-dir** オプションの代わりに、または追加してこのオプションを指定できます。

仮想化アーティファクト

以下のオプションにより、仮想化アーティファクトを生成できます。**--to-dir** オプションまたは **--to-project** オプションは必須です。

-v **名前**、--virtualize=**名前**

選択されたトランザクション フレームに対する仮想化アーティファクトを生成します。

--refer=**フレーム ID**

キー トランザクション フレームとして指定される特定のトランザクション フレームの識別子を指定します。このオプションが指定されない場合、クエリ結果内の最初のトランザクション フレームがキー トランザクション フレームとして指定されます。

--force-stateless

サービス イメージを生成するとき、VSE 会話式処理が適用されることを指定します。

--unknown-request-action=no_match|no_hijack

不明な要求に対して実行されるアクションを指定します。 このオプションは、Java VSE に基づいて生成された仮想化アーティファクトに適用されます。 有効な値は、**no_match** （「一致なし」を返す）および **no_hijack** （仮想サービスを省略）です。

--to-dir=**出力 PFI ファイル名**

生成された PFI ファイルが書き込まれるディレクトリの名前を指定します。

--to-project=LISA **プロジェクト ディレクトリ**

中間のインポート ファイルを必要とせずに、生成されたアーティファクトが書き込まれるプロジェクトのルート ディレクトリを指定します。 **--to-dir** オプションの代わりに、または追加してこのオプションを指定できます。

エージェントの状態

以下のオプションでは、エージェントの指定された状態を確認できます。サポートされている唯一の状態は、エージェントが現在ディスパッチしているかどうかです。 「*ディスパッチ*」とは、トランザクションをキャプチャするアクションを表します。

--check

エージェントの指定された状態を確認します。

--agent=**エージェント名**

エージェントの名前を指定します。

--condition=**状態**

確認する状態を指定します。

値： Dispatching

--check-timeout=**秒数**

タイムアウトになるまで待機する秒数を指定します。

キャプチャ レベル

以下のオプションでは、Java エージェントがキャプチャできる各プロトコルのキャプチャ レベルを変更できます。

注: WebSphere MQ、JMS など、キューベースのクライアント/サーバ通信では、異なるキャプチャ レベルの設定はサポートされていません。

--set-weights

1 つ以上のプロトコルのキャプチャ レベルを変更します。

--agent=**エージェント名**

エージェントの名前を指定します。

--protocols="**プロトコル[,プロトコル]**"

プロトコル名を指定します。 複数の値を含める場合は、値をカンマで区切ります。

値： ALL、HTTP Client、HTTP Server、Logging、Category、Exception、GUI、EJB、JMS、MQ、JCA、RCP、RMI、SAP、Tibco、WPS、WebMethods、JDBC

--weights="**重み付け[,重み付け]**"

プロトコルの重み付けを指定します。 複数の値を含める場合は、値をカンマで区切ります。

値： 0、4、8。 値 0 は「カウント」レベルに対応します。 値 4 は「カウント数およびパス」レベルに対応します。 値 8 は「全データ」レベルに対応します。

その他

以下のオプションも使用できます。

--broker=**ブローカ URL**

ブローカ接続文字列を指定します。 デフォルト値は **tcp://localhost:2009** です。

--search-frame

指定された検索条件およびフレーム名に一致するトランザクション フレームを検索します。

--frame-name=**フレーム名**

検索するトランザクション フレームの名前を指定します。

--help -h

ヘルプ テキストを表示します。

--version

バージョン番号を出力します。

例: ルート トランザクション フレームの表示

この例では、指定したエージェントおよび開始時刻のルート トランザクション フレームを表示する方法を示します。

```
PFCmdLineTool --roots --agent=JBoss_LISABank --from=2014-03-14T12:28:00

Frame ID                          Time                  Category  Local IP      Remote IP     Name
Exec Time
--------------------------------- --------------------- -------- ------------ -------------
-------  ---------
c3a9c830-abae-11e3-b937-0024d6ab5ce2 2014-03-14 12:28:04 660 web_http 10.132.92.143 10.132.92.143
Unknown  984 ms
```

例: トランザクション フレーム階層の表示

この例では、1 セットのルート トランザクション フレームのトランザクション フレーム階層を表示する方法を示します。

```
PFCmdLineTool --paths --agent=JBoss_LISABank --from=2014-03-14T12:28:00

984 ms /lisabank/buttonclick.do (c3a9c830-abae-11e3-b937-0024d6ab5ce2)
   50 ms $Proxy93.getUser (c3c73b40-abae-11e3-b937-0024d6ab5ce2)
      37 ms EJB3UserControlBean.getUser (c3c8c1e0-abae-11e3-b937-0024d6ab5ce2)
         3 ms SQL Activity (1) (c3c8c1e0-abae-11e3-b937-0024d6ab5ce2-SQL)
```

例: ベースライン スイートの生成

この例では、ベースライン スイートを生成する方法を示します。

```
PFCmdLineTool --baseline=BaselineName --refer=c3c8c1e0-abae-11e3-b937-0024d6ab5ce2
--to-dir=C:¥DevTest
```

例: ステートフル ベースラインの生成

この例では、ステートフル ベースラインを生成する方法を示します。

```
PFCmdLineTool --baseline=BaselineName --stateful --from=2014-03-14T12:28:00
--refer=c3c8c1e0-abae-11e3-b937-0024d6ab5ce2 --to-dir=C:¥DevTest
```

例: 仮想化アーティファクトの生成

この例では、仮想化アーティファクトを生成する方法を示します。

```
PFCmdLineTool --virtualize=VSEServiceName --from=2014-03-14T12:28:00
--refer=c3c8c1e0-abae-11e3-b937-0024d6ab5ce2 --category=ejb --to-dir=C:¥DevTest
```

例: エージェントの状態の確認

この例では、エージェントがトランザクションをキャプチャしているかどうかを確認する方法を示します。 この出力は、エージェントがトランザクションをキャプチャしていないことを示しています。

```
PFCmdLineTool --check --agent=JBoss_LISABank --condition=Dispatching --check-timeout=10

Agent has not yet started dispatching.
```

例: キャプチャ レベルの設定

この例では、1 つのプロトコルのキャプチャ レベルを変更する方法を示します。

```
PFCmdLineTool --set-weights --agent=JBoss_LISABank --protocols=EJB --weights=8
```

この例では、複数のプロトコルのキャプチャ レベルを変更する方法を示します。 引用符が使用されていることに注目してください。

```
PFCmdLineTool --set-weights --agent=JBoss_LISABank --protocols="EJB,JMS" --weights="8,4"
```

例: 名前によるトランザクション フレームの検索

この例では、CAI が特定のトランザクション フレームをキャプチャしたかどうかをフレーム名によって確認する方法を示します。 検索によって複数の一致するフレームが見つかった場合、最新のフレームが表示されます。この出力は名前、期間、および一意の識別子で構成されています。

```
PFCmdLineTool --search-frame --frame-name=EJB3AccountControlBean.addAccount
--from=2014-03-20T10:20:00

EJB3AccountControlBean.addAccount, 141 ms, 27ca1bd0-b03c-11e3-a8f1-005056ba138a
```

第 12 章: VS Traffic to CA Application Insight Companion（CA Application Insight への VS トラフィック）コンパニオン

CA Application Insight への VS トラフィック コンパニオンは、仮想サービスを生成してアプリケーションを可視化するために、データを CAI データベースにプッシュします。 このコンパニオンを仮想サービス モデルに追加すると、以下のアクションを実行します。

- モデルによって処理された要求およびそれに対応する応答をキャプチャする。
- 要求をトランザクション フレームとして CAI データベースに保存する。

このコンパニオンは、HTTP、JMS、および WebSphere MQ モデルをサポートしています。

前提条件

- アプリケーションからのメッセージが VSE レコーダによってキャプチャされます。
- 仮想サービス モデル ファイルが作成されました。 仮想サービス モデルの作成の詳細については、「*CA Service Virtualization の使用*」を参照してください。

次の手順に従ってください:

1. 仮想サービス モデルを開きます。
2. 右側のパネルで［Companions］を選択し、［Add］ をクリックします。

   ［Companions］メニューが開きます。
3. ［Virtual Service Environment］-［VS Traffic to CA Application Insight Companion］を選択します。
4. 仮想サービス モデルを保存します。
5. 仮想サービスを右クリックし、［Deploy/Redeploy to VSE@default］を選択します。 仮想サービスの展開の詳細については、「*CA Service Virtualization の使用*」を参照してください。

   仮想サービスが VSE@default VSE サーバに展開されたことを示すメッセージが表示されます。
6. 仮想サービスを呼び出します。

   トランザクション フレームが作成されます。

注: このコンパニオンによって作成されたトランザクション フレームが DevTest ポータルに表示されます。その際、仮想化を示すラベルが名前の最後に付加されます。HTTP トランザクションのコンポーネント名の一例は、**POST/itkoExamples/EJB3AccountControlBean(virtualized)** です。コンポーネント名の詳細については、「パス グラフ (P. 41)」を参照してください。

7. DevTest ポータルに移動し、ベースライン テストを作成します。HTTP、JMS、および WebSphere プロトコルのベースライン作成の詳細については、「ベースラインの作成 (P. 93)」を参照してください。
8. ベースラインから DevTest テストを作成します。
9. DevTest ワークステーション でテストを検証します。

第 13 章: CA Continuous Application Insight Agent Light

CAI Agent Light は、任意の外部システムに配置され、データを CAI にプッシュする軽量エージェントです。 データはアプリケーションに直接統合され、エージェントを導入することなく、ログ ファイル形式を使用してトランザクションがキャプチャされます。この軽量エージェントは、**LisaAgentLight.jar** という名前の Java の実行可能 JAR ファイルです。 この JAR ファイルは、コマンド ラインから実行されます。 コマンド ラインによる Agent Light の使用の詳細については、「コマンド ライン (P. 233)」を参照してください。

ログ ファイルによってキャプチャされたデータは、CAI データベースに追加され、DevTest コンソールに表示されます。

CAI には、以下の Agent Light があります。

- Agent Light for webMethods HTTP（P. 234）
- Agent Light for JMS（P. 238）
- Agent Light for WebSphere MQ（P. 244）

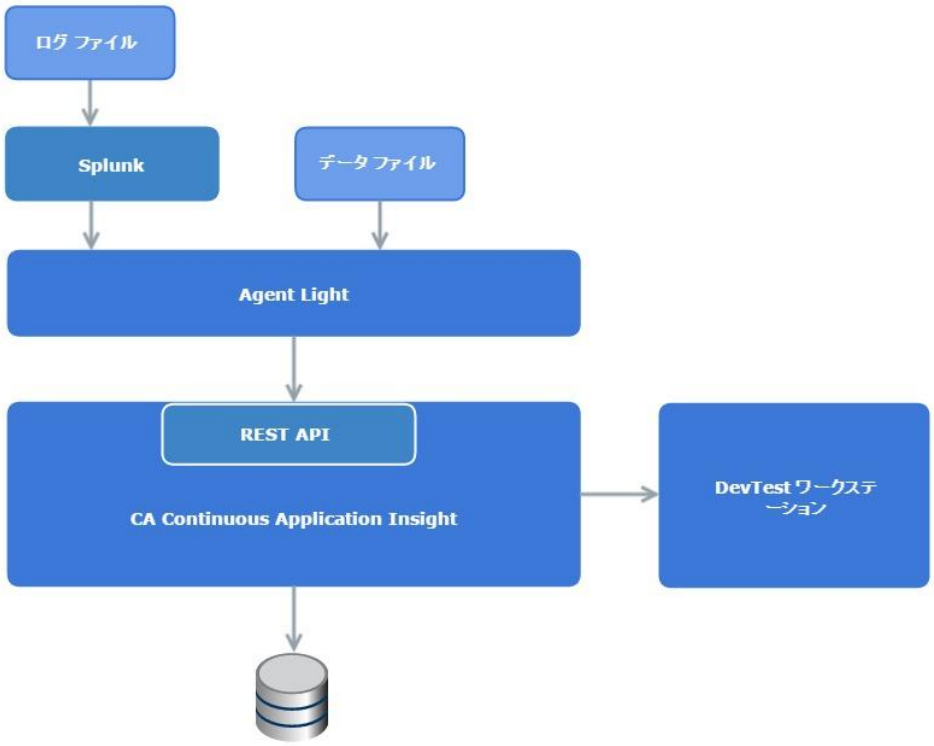
このセクションには、以下のトピックが含まれています。

Agent Light Architecture

CAI Agent Light アーキテクチャには、以下のコンポーネントがあります。

- サードパーティ ログ ファイルまたはデータ トランザクションからのデータ
- Agent Light
- CAI REST API
- DevTest ワークステーション
- データベース

以下の図は、コンポーネントおよびそれらのインタラクションを示しています。



Agent Light は、Splunk トランザクション クエリを使用して、データ トランザクション ファイルまたはサードパーティ ログ ファイルからデータを受信します。

REST API コールを通じてパスを作成するため、Agent Light はデータを CAI に送信します。

Agent Light 前提条件

Agent Light を使用するための要件は、以下のリストのとおりです。

- Java 1.7 以降
- LisaAgentLight.jar

Agent Light のインストール

LisaAgentLight.jar ファイルは、その他の CAI エージェント モジュールと同じ場所である、**LISA_HOME¥agent** ディレクトリにインストールされます。**LisaAgentLight.jar** ファイルは、任意のシステムのいかなる場所にもコピーできます。

REST API

CAI REST API は、トランザクション データを取得し、トランザクション パスを作成するためにそのデータを CAI に渡します。

URL

URL（http://LisaRegistryServer:1505/api/Pathfinder/convertTransaction?Protocol={protocolvalue}）を定義します。

メソッド

POST メソッドを定義します。

パラメータ

プロトコルを定義します。 HTTP、HTTPS、JMS、および MQ がサポートされています。

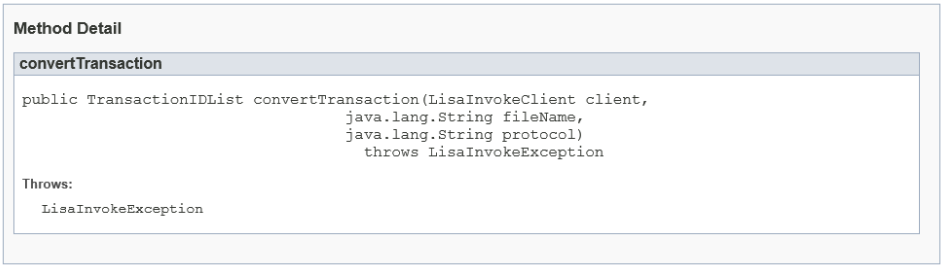
要求ボディ

（必須）トランザクション データの全内容をマルチパート形式で定義します。

戻り値

トランザクション ID を定義します。

CAI REST API は、Java SDK を使用して、REST API および **lisa-invoke2-client.jar** をラップします。 Java SDK では、クラス **com.itko.lisa.invoke.client.PathfinderServer** が CAI REST API をラップします。以下の図は、REST コールをラップする、このクラス内のメソッドを示しています。



Method Detail

convertTransaction

```
public TransactionIDList convertTransaction(LisaInvokeClient client,
                                            java.lang.String fileName,
                                            java.lang.String protocol)
                                              throws LisaInvokeException
```

Throws:

LisaInvokeException

コマンド ライン

コマンド ラインを使用して、CA Continuous Application Insight にエージェント ライトのトランザクション データをインポートします。 情報が解析され、CAI データベースに挿入されます。コマンド ラインは、CAI エージェントをカスタマイズまたは実装せずに、CAI にノードを挿入する方法を提供します。 この方法には、読み取りおよび解析できるデータのソース（ファイル ベースおよびその他のデータ）を指定する機能が含まれます。その後、データはフォーマットされ、CAI データベースに挿入されます。

コマンド ラインから、**LisaAgentLight.jar** ファイルを実行し、**-f** および **-t** パラメータを使用して、データ ファイルの場所とタイプを指定します。 パスが作成され、CAI コンソールに表示されます。

-f、-file

CAI にインポートされるトランザクションデータ ファイルを定義します。

-t、-type

データ プロトコル タイプを定義します。HTTP、JMS、および MQ がサポートされています。

デフォルト： HTTP

Agent Light for webMethods HTTP

Agent Light for webMethods HTTP は、REST API を使用して CA Continuous Application Insight と通信するエージェントです。この軽量エージェントの目的は、webMethods HTTP 要求/応答データ ファイルから入力を取得し、REST API コールを通じて、そのデータを CAI に送信することです。

Agent Light では、以下のソースから webMethods HTTP トランザクションを受信できます。

- HTTP トランザクション データ ファイル

  この軽量エージェントでデータ ファイルを入力として取得するには、**-f** パラメータを使用します。 このファイル内のデータは、標準的な HTTP 要求/応答形式にする必要があります。 標準データ フォーマットの詳細については、「HTTP の入力トランザクション データ形式 (P. 236)」を参照してください。

  データ ファイルからのトランザクション パラメータは、以下のとおりです。

  -f、-file

  CAI にインポートされるトランザクションデータ ファイルを定義します。

  -t、-type

  データ プロトコル タイプを示します。
  デフォルト： http

  以下のコード例を使用して、データ ファイルからパスを作成します。

  ```
  Java –jar lisaagentlight.jar –url http://<LisaServer>:1505 –f webmethods.log –t http
  ```

- Splunk クエリを使用する webMethods ログ ファイル

  webMethods 統合サーバが **trace** に設定されている場合、HTTP 要求/応答の Web サービス コール詳細がログ ファイルに記録されます。 Splunk によって、ログ ファイルがインデックス付けされ、HTTP トランザクション データが作成されます。 Agent Light により、Splunk に対してクエリが実行されて webMethods 統合サーバ HTTP トランザクションが取得され、その CAI パスが作成されます。

  Splunk からのトランザクション パラメータは、以下のとおりです。

  -splunk

  Splunk から取得されたトランザクションを示します。

-h、-hostname

Splunk サーバのホスト名または IP アドレスを定義します。

-port

Splunk のポート番号を定義します。

-u、-username

Splunk サーバのユーザ名を定義します。

-p、-password

Splunk サーバのパスワードを定義します。

-s、-search

（オプション） Splunk トランザクション検索ステートメントを定義します。

-t、-type

データ プロトコル タイプを定義します。HTTP、JMS、および MQ がサポートされています。

デフォルト： HTTP

-m、-maxtrans

インポートされる最大トランザクションを定義します。

デフォルト： 500

以下のコード例を使用して、Splunk からパスを作成します。

```
Java –jar lisaagentlight.jar –url http://<LisaServer>:1505 –splunk –hostname
10.130.151.105 –port 8089 –username admin –password admin
```

Splunk サーバに対して webMethods HTTP トランザクションに関するクエリを実行する場合、Splunk クエリ ステートメントを指定する必要があります。 このエージェントには、デフォルトの統合クエリ ステートメントがあります。 別のクエリが必要な場合は、**-search** パラメータを使用して、新しいクエリ ステートメントを渡します。

以下に示す例は、デフォルトのクエリ ステートメントです。

```
"search index=main ((CASE(GET) OR CASE(POST) OR CASE(PUT) OR CASE(DELETE)) /*) OR
((Accept OR User-Agent OR Accept-Encoding OR ¥"-- Host¥" OR Authorization OR Cookie
OR Accept-Language OR ¥"-- Connection¥" OR lisaFrameRoot OR lisaFrameRemoteIP OR
lisaFrameID OR Authorization OR Set-Cookie OR SOAPAction OR Content-Type OR
Content-Length): *) OR (¥" SOAP Request:¥") OR (¥" SOAP Response:¥") OR (HTTP/1.*)
| transaction startsWith=(CASE(GET) OR CASE(POST) OR CASE(PUT) OR CASE(DELETE))
endsWith=(¥"--> Content-Length: ¥")  | where !searchmatch(¥"-- User-Agent:
webMethods¥")"
```

注: Agent Light for webMethods HTTP がリモート システムから実行されている場合、DevTest サーバ REST URL を指定する必要があります。指定しない場合、エージェントでは DevTest サーバ がローカルであると仮定されます。 URL の形式は、http://<lisa-server>:1505 です。

この Agent Light には、以下の引数があります。

-url

CAI REST ベース URL を定義します。

デフォルト： http://localhost:1505

-wsuser

（オプション）CAI REST ユーザ名を定義します。

-wspassword

（オプション） CAI REST パスワードを定義します。

webMethods HTTP の入力トランザクション データ形式

webMethods HTTP の入力トランザクション データは、標準的な HTTP 要求/応答データ形式にする必要があります。 この Agent Light によって、取得された入力データは標準的な HTTP 形式に変換されます。

HTTP 要求メソッド URL {バージョン}［改行］
HTTP 要求ヘッダ［改行］
….
HTTP 要求ヘッダ［改行］
［改行 – 空白行］
{HTTP 要求ボディ – オプション}［改行］
HTTP 応答ステータス行［改行］
HTTP 応答ヘッダ［改行］
…
HTTP 応答ヘッダ［改行］
［改行 – 空白行］
{HTTP 応答ボディ – オプション}［改行］

Agent Light for webMethods HTTP におけるベースラインのサポート

この Agent Light によって作成されるパスでは、DevTest テスト ステップのみがサポートされます。 ［トランザクション フレーム ステップの使用］オプションは、Web インターフェースでは無効です。 Web サービスが HTTP ヘッダによる許可メカニズムを備えている場合、生成されたベースラインに対応して HTTP 許可ヘッダを変更する必要があります。Web HTTP ベースラインの生成の詳細については、「*Web HTTP ベースラインの生成* (116P.)」を参照してください。

Agent Light for webMethods HTTP における仮想化のサポート

この Agent Light によって作成されたパスは、仮想化できます。 再生は、CAI VSE ではなく VSE で実行されます。Web HTTP トランザクションからの仮想サービス作成の詳細については、「*Web HTTP トランザクションからの仮想サービスの作成* (177P.)」を参照してください。

Agent Light for JMS

Agent Light for JMS は、REST API を使用して CA Continuous Application Insight と通信するエージェントです。 この軽量エージェントの目的は、JMS トランザクション データ XML ファイルから入力を取得し、REST API コールを通じて、そのデータを CAI に送信することです。この Agent Light では、JMS トランザクション データ ファイルから JMS トランザクションを受信できます。

この軽量エージェントでデータ ファイルを入力として取得するには、**-f** パラメータを使用します。 このファイル内のデータは、CAI によって定義される XML 形式にする必要があります。 標準データフォーマットの詳細については、「JMS の入力トランザクション データ形式 (P. 239)」を参照してください。

データ ファイルからのトランザクション パラメータは、以下のとおりです。

-f、-file

CAI にインポートされるトランザクションデータ ファイルを定義します。

-t、-type

データ プロトコル タイプを示します。
デフォルト： http

以下のコード例を使用して、データ ファイルからパスを作成します。

```
Java –jar lisaagentlight.jar –url http://<LisaServer>:1505 –f sampleJmsDataFile.xml
–t jms
```

JMS の入力トランザクション データ形式

JMS の入力トランザクション データは、CA Continuous Application Insight によって指定された XML 形式にする必要があります。

JmsMessages タグ

JMS メッセージは、<JmsMessage> タグで開始し、対応する </JmsMessage> タグで終了します。JMS メッセージには、子 JMS メッセージを含めることができます。

```
<JmsMessages>
 <JmsMessage>
  JMS1
 </JmsMessage>
 <JmsMessage>
  JMS2
  <JmsMessage>
   JMS2.1
  </JmsMessage>
 </JmsMessage>
 . . .
</JmsMessages>
```

MessageMethod タグおよび MessageClass タグ

<MessageMethod> タグは、メッセージ メソッドを指定するために使用されます。 有効なメッセージ メソッドは、send、receive、および onMessage です。 タグがない場合、デフォルト値は send です。

<MessageClass> タグにより、メッセージ クラスを指定します。有効なメッセージ クラスは、javax.jms.TextMessage のみです。タグがない場合、デフォルト値は javax.jms.TextMessage です。

```
<JmsMessage>
 <MessageMethod>send</MessageMethod>
 <MessageClass>javax.jms.TextMessage</MessageClass>
 <Headers>
 <Destination>
 <Producer>
 <Body>
</JmsMessage>
```

ヘッダ タグ

<Header> タグにより、メッセージ ヘッダを指定します。有効なヘッダは、JMSCorrelationID、JMSDeliveryMode、JMSExpiration、JMSMessageID、JMSPriority、JMSRedelivered、JMSTimestamp、および JMSType です。

```
<JmsMessage>
 <MessageMethod>send</MessageMethod>
 <MessageClass>javax.jms.TextMessage</MessageClass>
 <Headers>
  <JMSCorrelationID></JMSCorrelationID>
  <JMSDeliveryMode>2</JMSDeliveryMode>
  <JMSExpiration>0</JMSExpiration>
  <JMSMessageID>ID:linch05win7-57515-1397847282288-1:1:1:1:1</JMSMessageID>
  <JMSPriority>4</JMSPriority>
  <JMSRedelivered>false</JMSRedelivered>
  <JMSTimestamp>1397847287643</JMSTimestamp>
  <JMSType></JMSType>
 </Headers>
 <Destination>
 <Producer>
 <Body>
</JmsMessage>
```

送信先タグ

<Destination> タグにより、メッセージ送信先を指定します。有効な送信先は、JNDI 送信先（JNDI 登録済みの Queue または Topic）です。 send メソッドが含まれる JMS メッセージにより、メッセージ送信先を指定します。

```
<JmsMessage>
 <MessageMethod>send</MessageMethod>
 <MessageClass>javax.jms.TextMessage</MessageClass>
 <Headers>
 <Destination>
  <JndiDestination name="dynamicQueues/ORDERS.REQUEST" type="Queue">
   <Context>
    <CanonicalUrl>tcp://localhost:2009</CanonicalUrl>
    <Prop name="java.naming.provider.url">tcp://localhost:2009</Prop>
    <Prop name="java.naming.factory.initial">org.apache.activemq.jndi.ActiveMQInitialContextFactory</Prop>
   </Context>
  </JndiDestination>
 </Destination>
 <Producer>
 <Body>
</JmsMessage>
```

プロデューサ タグ

<Producer> タグは、メッセージ プロデューサを指定するために使用されます。 プロデューサには、送信先（JNDI）およびセッションを含めることができます。 セッションには、セッション情報（Transacted、AcknowledgeMode）および接続ファクトリ（JNDI）が含まれます。 **send** メソッドが含まれる JMS メッセージにより、メッセージ プロデューサを指定します。

```
<JmsMessage>
 <MessageMethod>send</MessageMethod>
 <MessageClass>javax.jms.TextMessage</MessageClass>
 <Headers>
 <Destination>
 <Producer>
  <Destination>
  <Session>
   <Transacted>false</Transacted>
   <AcknowledgeMode>1</AcknowledgeMode>
   <ConnectionFactory>
    <JndiConnectionFactory name="ConnectionFactory">
     <Context>
      <CanonicalUrl>tcp://localhost:2009</CanonicalUrl>
      <Prop name="java.naming.provider.url">tcp://localhost:2009</Prop>
      <Prop name="java.naming.factory.initial">org.apache.activemq.jndi.ActiveMQInitialContextFactory</Prop>
     </Context>
    </JndiConnectionFactory>
   </ConnectionFactory>
  </Session>
 </Producer>
 <Body>
</JmsMessage>
```

コンシューマ タグ

<Consumer> タグは、メッセージ コンシューマを指定するために使用されます。 コンシューマの構造は、プロデューサと同じです。 receive/onMessage メソッドが含まれる JMS メッセージにより、メッセージ コンシューマを指定します。

ボディ タグ

<Body> タグは、メッセージ ペイロード（テキスト）を指定するために使用されます。 テキスト ペイロード自身に XML タグが含まれる場合、テキスト ペイロードは CDATA セクションの内部にある必要があります。 そのペイロードは、パーサでは無視されます。

```
<JmsMessage>
 <MessageMethod>send</MessageMethod>
 <MessageClass>javax.jms.TextMessage</MessageClass>
 <Headers>
 <Destination>
 <Producer>
 <Body>
  <![CDATA[
   <order>
       <id>2-22</id>
       <name>ouid-2</name>
       <product>prod200</product>
   </order>
  ]]>
 </Body>
</JmsMessage>
```

Agent Light for JMS におけるベースラインのサポート

この Agent Light によって作成されるパスでは、DevTest テスト ステップのみがサポートされます。 統合ベースラインおよび展開されたベースラインのみがサポートされます。 JMS ベースラインの生成の詳細については、「*JMS ベースラインの生成* (102P.)」 を参照してください。

Agent Light for JMS における仮想化のサポート

この Agent Light によって作成されたパスは、仮想化できます。MQ トランザクションの仮想サービス作成の詳細については、「*JMS トランザクションからの仮想サービスの作成* (157P.)」を参照してください。

Agent Light for WebSphere MQ

Agent Light for WebSphere MQ は、REST API を使用して CA Continuous Application Insight と通信するエージェントです。この軽量エージェントの目的は、MQ 要求/応答データ ファイルから入力を取得し、REST API コールを通じて、そのデータを CAI に送信することです。

この Agent Light では、以下のソースから MQ トランザクションを受信できます。

- MQ トランザクション データ ファイル

  この軽量エージェントでデータ ファイルを入力として取得するには、**-f** パラメータを使用します。データ ファイルからのトランザクション パラメータは、以下のとおりです。

  -f、-file

  CAI にインポートされるトランザクションデータ ファイルを定義します。

  -t、-type

  データ プロトコル タイプを定義します。HTTP、JMS、および MQ がサポートされています。

  デフォルト： HTTP

  以下のコード例を使用して、データ ファイルからパスを作成します。

  ```
  Java –jar lisaagentlight.jar –url http://<LisaServer>:1505 –f mqSample.xml –t mq
  ```

- Splunk クエリからのサードパーティ ログ ファイル

  サードパーティ ログ ファイルは、Splunk クエリを使用してサポートされています。Splunk によって、ログ ファイルがインデックス付けされ、MQ トランザクション データが作成されます。 この Agent Light により、Splunk に対してクエリが実行されて MQ トランザクションが取得され、その CAI パスが作成されます。サードパーティ ログ ファイルが MQ トランザクションによって必要なすべての情報を含むとは限りません。拡張 XML データ ファイルは、ユーザが欠落データを入力可能とするために定義されます。 この拡張ファイル内のデータは、DevTest Solutions によって定義された XML 形式にする必要があります。

  Splunk からのトランザクション パラメータは、以下のとおりです。

  -splunk

  Splunk から取得されたトランザクションを示します。

-h、-hostname

Splunk サーバのホスト名または IP アドレスを定義します。

-port

Splunk のポート番号を定義します。

-u、-username

Splunk サーバのユーザ名を定義します。

-p、-password

Splunk サーバのパスワードを定義します。

-s、-search

（オプション） Splunk トランザクション検索ステートメントを定義します。

-t、-type

データ プロトコル タイプを定義します。HTTP、JMS、および MQ がサポートされています。

デフォルト： HTTP

-m、-maxtrans

インポートされる最大トランザクションを定義します。

デフォルト： 500

-source

Splunk データ ソース（webMethods、swamq、または JMS のいずれか）を定義します。

-extf、-extfile

拡張トランザクションデータ ファイルを定義します。

以下のコード例を使用して、Splunk からパスを作成します。

```
Java –jar lisaagentlight.jar –url http://<LisaServer>:1505 –splunk –hostname
10.130.151.105 –port 8089 –username admin –password admin -source swamq -type mq
-extfile extendedDataFile.xml
```

Splunk に対してサードパーティ トランザクションに関するクエリを実行する場合、Splunk クエリ ステートメントを指定して、このトランザクションに関するクエリを実行する必要があります。 このエージェントには、デフォルトのビルトイン クエリ ステートメントがあります。 別のクエリが必要な場合は、**-search** パラメータを使用して、新しいクエリ ステートメントを渡すことができます。

デフォルトのビルトイン クエリ ステートメントは、以下のとおりです。

```
"search index=main CASE(¥"- REQUEST: <?xml¥") OR CASE(¥"- RESPONSE:
<?xml¥") | transaction startsWith=CASE(REQUEST:)
endsWith=CASE(RESPONSE:)";
```

Agent Light for WebSphere MQ がリモート システムから実行されている場合、DevTest サーバ REST URL を指定する必要があります。指定しない場合、エージェントでは DevTest サーバ がローカルであると仮定されます。 URL の形式は、**http://<lisa-server>:1505** です。

この Agent Light には、以下に示すオプションのパラメータがあります。

-url

CAI REST ベース URL を定義します。

デフォルト: http://localhost:1505

-wsuser

（オプション）CAI REST ユーザ名を定義します。

-wspassword

（オプション）CAI REST パスワードを定義します。

注: トランザクション データ ファイルおよび拡張 XML データ ファイル内のデータは、DevTest Solutions によって定義される XML 形式にする必要があります。 標準データフォーマットの詳細については、「MQ の入力トランザクション データ形式 (P. 247)」を参照してください。

MQ の入力トランザクション データ形式

MQ の入力トランザクション データは、DevTest Solutions によって指定された形式にする必要があります。すべての MQ トランザクションは、以下の例のようにする必要があります。

```
<MQTransactions>
        <MQTransaction>
                <MQMessage>
                        ....
                        <QueueConnection>
                        ...
                        </QueueConnection>
                        <request>request content</request>
                </MQMessage>
                <MQMessage>
                ...
                </MQMessage>
                <MQMessage>
                ...
                </MQMessage>
                ....
    </MQTransaction>
    ...
    <MQTransaction/>
</MQTransactions>
```

拡張データ ファイル内のデータは、DevTest Solutions によって定義された XML 形式にする必要があります。

```
<Platforms>
        <SWA>
                <!-- The log is from mq client or not, default value is true -->
                <IsClient>true</IsClient>
                <QueueConnection>
                        <!-- required -->
                        <RequestQueue>ORDERS.REQUEST</RequestQueue>
                        <ResponseQueue>ORDERS.RESPONSE</ResponseQueue>
                        <QueueManager>QueueManager</QueueManager>
                        <Host>HostName</Host>
                        CA Portal1414</Port>
                        <Channel>SERVERCON</Channel>
                        <!-- often appear -->
                        <Username>administrator</Username>
                        <Password>dcam09@CTC</Password>
                </QueueConnection>
        </SWA>
</Platforms>
```

Agent Light for WebSphere MQ におけるベースラインのサポート

この Agent Light によって作成されるパスでは、DevTest テスト ステップのみがサポートされます。 統合ベースラインおよび展開されたベースラインのみがサポートされます。WebSphere MQ ベースラインの生成の詳細については、「*WebSphere MQ ベースラインの生成* (133P.)」を参照してください。

Agent Light for WebSphere MQ における仮想化のサポート

この Agent Light によって作成されたパスは、仮想化できます。MQ トランザクションの仮想サービス作成の詳細については、「*WebSphere MQ トランザクションからの仮想サービスの作成*」を参照してください。

用語集

アサーション

*アサーション*は、1 つのステップとそのすべてのフィルタが実行された後に実行されるエレメントです。 アサーションにより、ステップの実行結果が予測と一致することが検証されます。 アサーションは、通常、テスト ケースまたは仮想サービス モデルのフローを変更するために使用されます。 グローバル アサーションは、テスト ケースまたは仮想サービス モデルの各ステップに適用されます。詳細については、「*CA Application Test の使用*」の「アサーション」を参照してください。

アセット

*アセット*は、1 つの論理的な単位にグループ化される設定プロパティのセットです。 詳細については、「*CA Application Test の使用*」の「アセット」を参照してください。

一致許容差

*一致許容差*は、CA Service Virtualization が受信要求をサービス イメージ内の要求と比較する方法を制御する設定です。 オプションは、EXACT、SIGNATURE、および OPERATION です。 詳細については、「*CA Service Virtualization の使用*」の「一致許容差」を参照してください。

イベント

*イベント*は、発生したアクションに関するメッセージです。 テスト ケースまたは仮想サービス モデル レベルでイベントを設定できます。 詳細については、「*CA Application Test の使用*」の「イベントについて」を参照してください。

会話ツリー

*会話ツリー*は、仮想サービス イメージにおいてステートフル トランザクションの会話パスを表すリンクされたノードのセットです。 各ノードは、withdrawMoney などの操作名でラベル付けされます。 getNewToken、getAccount、withdrawMoney、deleteToken は、金融機関システムの会話パスの一例です。 詳細については、「*CA Service Virtualization の使用」を参照してください。*

仮想サービス モデル（VSM）

*仮想サービス モデル*は、実際のサービス プロバイダなしでサービス要求を受信および応答します。 詳細については、「*CA Service Virtualization の使用*」の「仮想サービス モデル（VSM）」を参照してください。

監査ドキュメント

*監査ドキュメント*では、1 つのテスト、またはスイート内の 1 つのテスト セットに対する成功条件を設定できます。 詳細については、「*CA Application Test の使用*」の「監査ドキュメントの作成」を参照してください。

クイック テスト

*クイック テスト*機能を使用すると、最小のセットアップでテスト ケースを実行できます。 詳細については、「*CA Application Test の使用*」の「クイック テストのステージング」を参照してください。

グループ

グループ、または仮想サービス グループは、VSE コンソールでまとめてモニタできるように、同じグループ タグでタグ付けされている仮想サービスのコレクションです。

継続的検証サービス（CVS）ダッシュボード

*継続的検証サービス（CVS）ダッシュボード*では、長期間にわたって定期的に実行するテスト ケースおよびテスト スイートをスケジュールできます。 詳細については、「*CA Application Test の使用*」の「継続的検証サービス（CVS）」を参照してください。

コーディネータ

*コーディネータ*はテスト ランの情報をドキュメントとして受け取り、1 つ以上のシミュレータ サーバで実行されるテストをコーディネートします。詳細については、「*CA Application Test の使用*」の「コーディネータ サーバ」を参照してください。

コンパニオン

*コンパニオン*は、すべてのテスト ケースの実行の前後に実行されるエレメントです。 コンパニオンは、単一のテスト ステップではなく、テスト ケース全体に適用されるフィルタとして理解できます。 コンパニオンはテスト ケース内で（テスト ケースに対して）グローバルな動作を設定するために使用されます。 詳細については、「*CA Application Test の使用*」の「コンパニオン」を参照してください。

サービス イメージ（SI）

*サービス イメージ*は、CA Service Virtualization で記録されたトランザクションの正規化バージョンです。 各トランザクションは、ステートフル（会話型）またはステートレスです。 サービス イメージを作成する方法の 1 つは、仮想サービス イメージ レコーダを使用することです。 サービス イメージは、プロジェクトに格納されます。 サービス イメージは、*仮想サービス イメージ*（VSI）とも呼ばれます。詳細については、「*CA Service Virtualization の使用*」の「サービス イメージ」を参照してください。

サブプロセス

*サブプロセス*は、別のテスト ケースによってコールされるテスト ケースです。 詳細については、「*CA Application Test の使用*」の「サブプロセスの作成」を参照してください。

シミュレータ

*シミュレータ*は、コーディネータ サーバの管理下でテストを実行します。詳細については、「*CA Application Test の使用*」の「シミュレータ サーバ」を参照してください。

ステージング ドキュメント

*ステージング ドキュメント*には、テスト ケースを実行する方法に関する情報が含まれます。詳細については、「*CA Application Test* の使用」の「ステージング ドキュメントの作成」を参照してください。

設定

*設定*は、プロパティの名前付きのコレクションであり、通常はテスト中のシステムの環境に固有の値を指定します。 ハードコードされた環境データをなくすことにより、設定を変更するだけで、異なる環境内のテスト ケースまたは仮想サービス モデルを実行できます。 プロジェクトのデフォルト設定の名前は project.config です。 プロジェクトは多数の設定を持つことができますが、一度にアクティブになるのは 1 つの設定のみです。詳細については、「*CA Application Test の使用*」の「設定」を参照してください。

対話型テスト ラン（ITR）

*対話型テスト ラン（ITR）*ユーティリティを使用すると、テスト ケースまたは仮想サービス モデルをステップごとに実行できます。 テスト ケースまたは仮想サービス モデルを実行時に変更し、結果を確認できます。 詳細については、「*CA Application Test の使用*」の「対話型テスト ラン（ITR）ユーティリティの使用」を参照してください。

ディセンシタイズ

*ディセンシタイズ*は、機密データをユーザ定義の代替データに変換するために使用されます。 クレジット カード番号や社会保障番号は機密データの例です。 詳細については、「*CA Service Virtualization の使用*」の「データのディセンシタイズ」を参照してください。

データ セット

*データ セット*は、実行時にテスト ケースまたは仮想サービス モデルにプロパティを設定するために使用できる値のコレクションです。 データ セットによって、テスト ケースまたは仮想サービス モデルに外部のテスト データを使用することができます。データ セットは、DevTest の内部または外部（たとえば、ファイルやデータベース テーブル）に作成できます。 詳細については、「*CA Application Test の使用*」の「データ セット」を参照してください。

データ プロトコル

*データ プロトコル*は、データ ハンドラとも呼ばれます。 CA Service Virtualization では、データ プロトコルは、要求の解析処理を行います。一部のトランスポート プロトコルは、要求を作成するジョブの委任先のデータ プロトコルを許可（または要求）します。 結果として、プロトコルは要求ペイロードを認識する必要が生じます。 詳細については、「*CA Service Virtualization の使用*」の「データ プロトコルの使用」を参照してください。

テスト ケース

*テスト ケース*は、テスト中のシステムのビジネス コンポーネントをテストする方法の仕様です。 各テスト ケースには、1 つ以上のテスト ステップが含まれます。 詳細については、「*CA Application Test の使用*」の「テスト ケースの作成」を参照してください。

テスト スイート

*テスト スイート*は、順番に実行されるようにスケジュールされたテスト ケース、その他のテスト スイート、またはその両方のグループです。 スイート ドキュメントは、スイートのコンテンツ、生成するレポート、および収集するメトリックを指定します。詳細については、「*CA Application Test の使用*」の「テスト スイートの作成」を参照してください。

テスト ステップ

*テスト ステップ*は、実行される単一のテスト アクションを表すテスト ケース ワークフローのエレメントです。 テスト ステップの例としては、Web サービス、Java Bean、JDBC、JMS メッセージングなどがあります。 テスト ステップには、フィルタ、アサーション、データ セットなどの DevTest エレメントを含めることができます。 詳細については、「*CA Application Test の使用*」の「テスト ステップの作成」を参照してください。

トランザクション フレーム

*トランザクション フレーム*は、DevTest Java エージェントまたは CAI Agent Light がインターセプトしたメソッド コールに関するデータをカプセル化します。詳細については、「*CA Continuous Application Insight の使用*」の「ビジネス トランザクションおよびトランザクション フレーム (P. 16)」を参照してください。

ナビゲーション許容差

*ナビゲーション許容差*は、CA Service Virtualization が会話ツリーを検索して次のトランザクションを見つける方法を制御する設定です。 オプションは、CLOSE、WIDE、および LOOSE です。 詳細については、「*CA Service Virtualization の使用*」の「ナビゲーション許容差」を参照してください。

ネットワーク グラフ

ネットワーク グラフは、DevTest クラウド マネージャおよび関連するラボをグラフで表示するサーバ コンソールの領域です。詳細については、「*CA Application Test の使用*」の「ラボの開始」を参照してください。

ノード

DevTest の内部では、テスト ステップは*ノード*とも呼ばれます。これが、一部のイベントがイベント ID 内にノードを持つ理由です。

パス

*パス*には、Java エージェント がキャプチャしたトランザクションに関する情報が含まれます。詳細については、「*CA Continuous Application Insight の使用*」を参照してください。

パス グラフ

*パス グラフ*には、パスおよびそのフレームのグラフ表示が含まれています。 詳細については、「*CA Continuous Application Insight の使用*」の「パス グラフ」を参照してください。

反応時間

*反応時間*は、テスト ステップを実行する前にテスト ケースが待機する時間です。詳細については、「*CA Application Test の使用*」の「テスト ステップの追加 - 例」および「ステージング ドキュメント エディタ - ［ベース］タブ」を参照してください。

フィルタ

*フィルタ*は、ステップの前後に実行されるエレメントです。フィルタは、結果のデータを処理、またはプロパティに値を格納する機会を提供します。グローバル フィルタは、テスト ケースまたは仮想サービス モデルの各ステップに適用されます。 詳細については、「*CA Application Test の使用*」の「フィルタ」を参照してください。

プロジェクト

*プロジェクト*は、関連する DevTest ファイルのコレクションです。 ファイルには、テスト ケース、スイート、仮想サービス モデル、サービス イメージ、設定、監査ドキュメント、ステージング ドキュメント、データ セット、モニタ、および MAR 情報ファイルなどが含まれます。 詳細については、「*CA Application Test の使用*」の「プロジェクト パネル」を参照してください。

プロパティ

*プロパティ*は、ランタイム変数として使用できるキー/値ペアです。 プロパティには、さまざまなタイプのデータを格納できます。 一般的なプロパティには、LISA_HOME、LISA_PROJ_ROOT、LISA_PROJ_NAME などがあります。 設定は、プロパティの名前付きのコレクションです。 詳細については、「*CA Application Test の使用*」の「プロパティ」を参照してください。

マジック ストリング

*マジック ストリング*は、サービス イメージの作成中に生成される文字列です。 マジック ストリングは、仮想サービス モデルによって応答内で意味のある文字列値が提供されることを確認するために使用されます。 {{=request_fname;/chris/}} は、マジック ストリングの一例です。 詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

マジック デート

レコーディング中、日付パーサは要求および応答をスキャンします。 日付表示形式の広範な定義に一致する値は、*マジック デート*に変換されます。 マジック デートは、仮想サービス モデルによって応答内で意味のある日付値が提供されることを確認するために使用されます。 {{=doDateDeltaFromCurrent("yyyy-MM-dd","10");/*2012-08-14*/} は、マジック デートの一例です。詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

メトリック

*メトリック*により、テストおよびテスト中のシステムのパフォーマンス/機能面に定量的手法および測定単位を適用できます。 詳細については、「*CA Application Test の使用*」の「メトリックの生成」を参照してください。

モデル アーカイブ（MAR）

*モデル アーカイブ（MAR）*は、DevTest Solutions における主要な展開アーティファクトです。 MAR ファイルには、プライマリ アセット、プライマリ アセットを実行するために必要なすべてのセカンダリ ファイル、情報ファイル、および監査ファイルが含まれます。 詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

モデル アーカイブ（MAR）情報

*モデル アーカイブ（MAR）情報*ファイルは、MAR を作成するために必要な情報が含まれるファイルです。詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

ラボ

*ラボ*は、1 つ以上のラボ メンバの論理コンテナです。 詳細については、「*CA Application Test の使用*」の「ラボとラボ メンバ」を参照してください。

レジストリ

*レジストリ*は、すべての DevTest サーバおよび DevTest ワークステーション コンポーネントの登録を一元的に行うための場所です。 詳細については、「*CA Application Test* の使用」の「レジストリ」を参照してください。

仮想サービス環境（VSE）

*仮想サービス環境（VSE）*は、仮想サービス モデルを展開して実行するために使用する DevTest サーバ アプリケーションです。 VSE は CA Service Virtualization とも呼ばれます。 詳細については、「*CA Service Virtualization の使用*」を参照してください。