DevTest Solutions

CA Service Virtualization 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 Service Virtualization

このセクションには、以下のトピックが含まれています。

仮想化の概要

「*仮想化*」は、通常、ハードウェアの仮想化を意味します。ハードウェアの仮想化では、物理アセットの動作がシミュレートされ（ソフトウェア エミュレータでのサーバやアプリケーションなど）、エミュレータは仮想環境でホストされます。 仮想環境は、エミュレートされたアセットに対して物理的環境と同じ通信を提供します。

仮想化には以下の利点があります。

- 物理アセットの管理に優れており、変更および設定の管理が容易
- 物理容量の使用率が改善され、物理アセットの容量の活用状態が向上
- 俊敏性が向上することにより、IT 部門によるサーバの再設定や切り替え時に発生するコストの高い遅延の回避

CA Service Virtualization は、サービスの仮想化を提供します。 そのプロセスは、基本的にはハードウェアの仮想化と同じです。

サービス仮想化のタイプ

CA Service Virtualization には、特定のカスタマ アプリケーションに対して最適化された以下の 2 つの製品構成があります。

- CA Service Virtualization
- CA Service Virtualization for Performance

CA Service Virtualization は、開発、統合、テスト、およびユーザ受け入れのユース ケースに最適です。 これらの製品のインスタンスは、一度に 10 個の並列トランザクション、または 1 秒間に約 10 個のトランザクションを処理します。

CA Service Virtualization for Performance は、パフォーマンス テスト アプリケーション専用であり、基盤となるハードウェアおよびネットワークによってのみ制限を受ける、より拡張性の高い製品です。

サービスの仮想化の概要

*サービスの仮想化*とは、ソフトウェアのサービスの動作のイメージ化、および開発/テスト時に実際のサービスの代わりとなる仮想サービスのモデル化のことです。 サービス仮想化はハードウェア仮想化を補完し、その制限に対応します。 *仮想化*という用語は、このドキュメントではサービス仮想化を意味します。

品質保証タスクのために、サーバに接続したままにすることを望まない場合や、接続したままにできない場合があります。CA Service Virtualization は、サーバの動作をエミュレートします。

CA Service Virtualization 内では、仮想サービス環境（CA Service Virtualization）をクライアントとして使用して、記録されるサービスを実行できます。ただし、レコーディングのためのサービスは、ユーザのクライアント（ブラウザや社内アプリケーションなど）を使用して実行する場合が多いと考えられます。

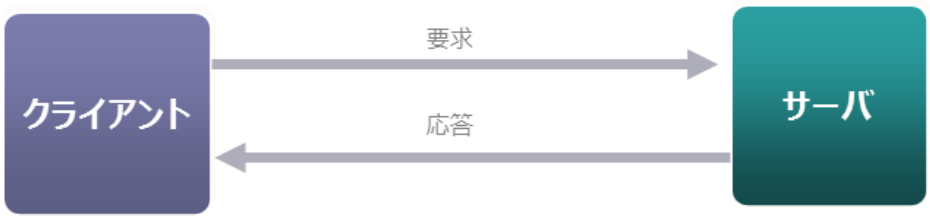
仮想化の手順の概要

CA Service Virtualization での手順の概要を以下に示します。

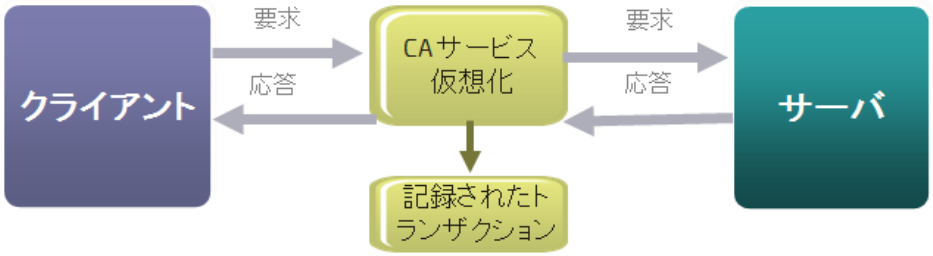
1. サービスの動作のイメージを作成（サービス イメージ）してサーバが処理するトランザクションを記録します。
2. その動作から仮想サービスを作成します。これは、「VSM」（Virtual Service Model、仮想サービス モデル）と呼ばれます。
3. 仮想サービス環境（VSE）に VSM を展開します。 次に、VSM は、キャプチャされたサービス イメージを確認して、VSE が受信した要求に対する適切な応答を検索します。

以下の図は、イメージのレコーディング時に VSE がクライアントとサーバ間のパススルー メカニズムとして動作することを示しています。 VSE は、要求と応答を渡し、トランザクションを記録します。

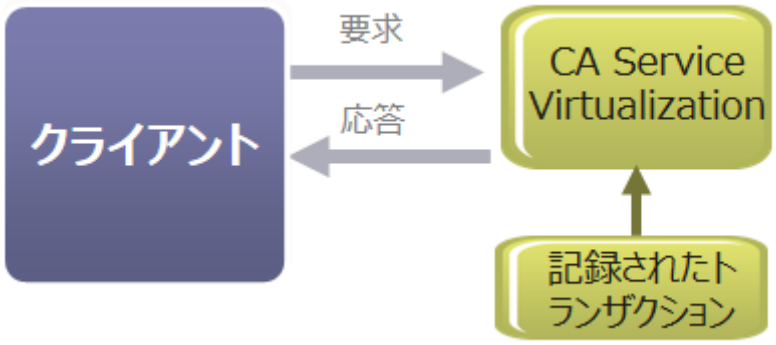
通常動作



レコーディング



仮想化時にサーバが存在しない場合、VSE は、記録されたトランザクションを確認することによって、クライアントの要求に応答します。


クライアント
要求
応答
CA Service Virtualization
記録されたトランザクション

メッセージング システムの仮想化

メッセージ指向ミドルウェア（MOM）と呼ばれるメッセージング システムは、2 つ以上のソフトウェア アプリケーション間の非同期通信を可能にする手段を提供するサービスです。 この通信は、常にメッセージの形式で発生します。 メッセージは、MOM に設定されたメッセージ送信先にポストされます。

メッセージ送信先のタイプは次のとおりです。

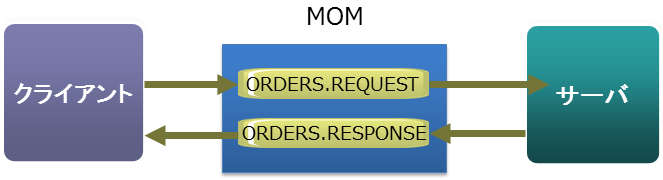
キュー

パブリッシャは、キューにメッセージを追加します。また、サブスクライバは、「先入れ先出し」方式で、キューからメッセージを取得します。

トピック

パブリッシャは、トピックにメッセージをパブリッシュします。また、トピックをサブスクライブしているすべてのサブスクライバはメッセージを受信します。

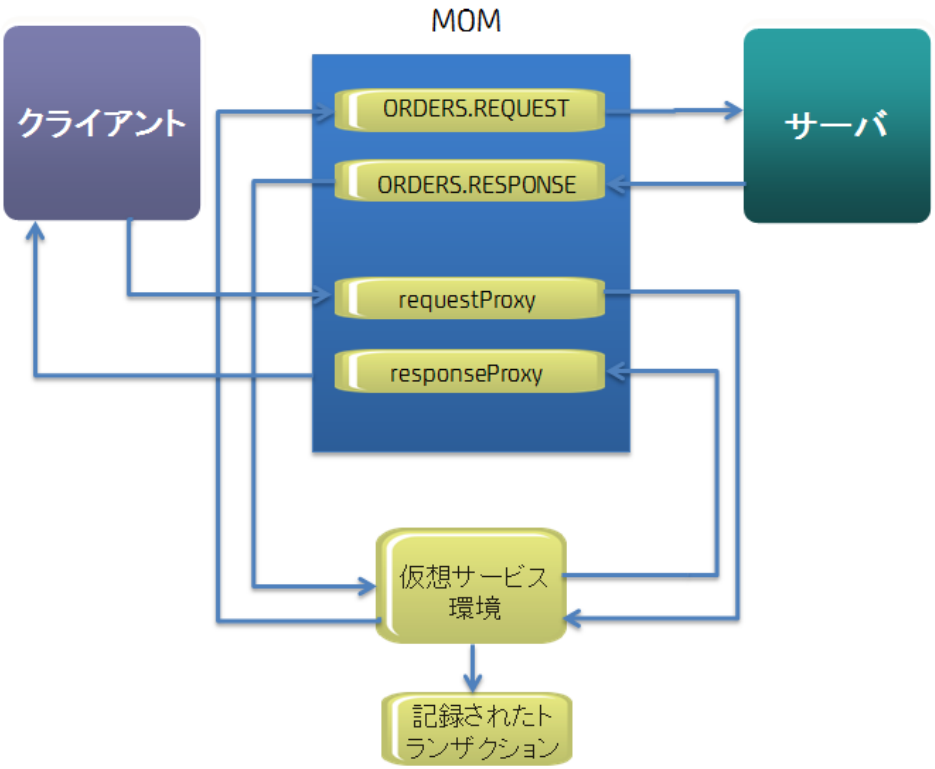
以下の図は、単純なメッセージ ベースのサービスを示しています。 このシナリオでは、クライアントはキュー（**ORDERS.REQUEST**）へメッセージを追加します。サーバはそれを取得します。 サーバからの応答は、別のキュー（**ORDERS.RESPONSE**）に追加されるメッセージの形式です。 その後、クライアントがそれを取得します。



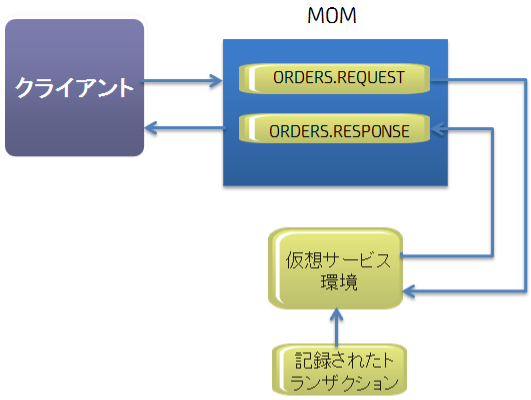
可能なバリエーションには以下のものがあります。

- キューの代わりにトピックを使用
- 単一の要求に対する複数の応答（別の送信先をターゲットにすることも可能）

CA Service Virtualization は、サーバの仮想化を目的としています。 レコーディング モードでは、VSE は、クライアントがその通信相手の代わりに使用する追加のプロキシ送信先（以下の図内の **requestProxy** キューと **responseProxy** キュー）を必要とします。 サーバは引き続きリスンし、実際の送信先にポストします。 VSE は、これらのプロキシと実際の送信先の間のパス スルーとして動作します。 VSE は、トラフィックを記録して、仮想化に必要な VSM およびサービス イメージを作成します。



サーバを仮想化すると、VSE は実際の送信先と動作します。 VSE はプロキシ送信先を必要としません。


MOM
クライアント
ORDERS.REQUEST
ORDERS.RESPONSE
仮想サービス
環境
記録されたト
ランザクション

第 2 章: インストール

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 Service Virtualization のインストール方法

CA Service Virtualization ソフトウェアは、DevTest サーバとともにインストールされます。 DevTest Solutions のインストールと設定の詳細については、「*インストール*」を参照してください。

CA Service Virtualization を使用するには、以下のプロセス（またはサービス）が実行中である必要があります。

- Registry（DevTest レジストリ サービス）
- VirtualServiceEnvironment（VSE サービス）

サーバ レベルのサービスとして、CA Service Virtualization は、接続されたコーディネータおよびシミュレータがあるレジストリと共存できます。シミュレータおよびコーディネータは CA Service Virtualization の実行に必須ではありません。

システム要件

CA Service Virtualization に対する以下のシステム リソースは、ベースラインの要件です。

- **CPU：** 2 GHz 以上
- **RAM：** 2 GB 以上
- **ディスク空き容量：** 5 GB
- **OS：** 推奨： 64 ビット オペレーティング システム。 サポート対象： Windows 2008、7、8、Linux、Solaris、AIX 6.1 （LISA 5.0 以降）

大規模な CA Service Virtualization 展開には、以下のリソースを推奨します。

- VSE インスタンスごとに 256 の仮想サービス スレッド
- VSE インスタンスごとに、1 つのプロセッサ コアおよび 2 GB の RAM

例: 毎日 1,000,000 のトランザクション

- 機能テストをサポートするにはサービスごとに 1 スレッド
- 負荷およびパフォーマンス テスト用の仮想化をサポートするにはサービスごとに約 6 スレッド。
- 8 つのコアは 2,048 の同時仮想サービス スレッドに相当します。
- 16 GB RAM （DevTest の場合）。

その他のシステム要件の詳細については、「*インストール*」の「システム要件および前提条件」を参照してください。

ソフトウェア ディレクトリ構造およびファイル

以下のリストは、DevTest Solutions のディレクトリ構造を示しています。このディレクトリは、DevTest ルート インストール フォルダに含まれています。

bin

TestRegistry.exe、Workstation.exe、VirtualServiceEnvironment.exe、VSEManager.exe などの実行可能ファイルが含まれます。

DemoServer

DevTest デモ サーバが含まれます。

doc

DevTest ドキュメントが含まれます。

examples/vse

このディレクトリには、VSE に関連するサンプルが含まれます。

tmp

このディレクトリには、一時的に会話とステートレス トランザクションを格納する VSE ワークスペースが含まれます。

vseDeploy

このディレクトリには、展開された VSM および関連データが含まれます。

CA Service Virtualization の設定方法

CA Service Virtualization を正しく設定するには、以下の手順に従います。

1. ユーザおよびモニタ使用状況をセットアップします (P. 22)。
2. ローカル ホストに対してプロキシを設定します (P. 22)。
3. local.properties でその他の設定を定義します（オプション） (P. 23)。

ユーザの設定および使用状況のモニタ

DevTest Solutions ライセンスは、製品全体向けです。 使用許諾契約によって、（ほかにもユーザ タイプはありますが） SV パワー ユーザ ユーザ タイプの同時ユーザの最大数が指定されます。 管理者は、SV パワー ユーザ ユーザ タイプに関連付けられた各種権限を CA Service Virtualization ユーザに付与します。 定期的に、管理者は、同時使用の最大数の遵守をモニタするため、使用状況レポートを生成します。 「*管理*」を参照します。

ローカル ホストに対するプロキシの設定

DevTest が VSE の HTTP クライアントとして動作する場合、DevTest からの HTTP トラフィックを VSE に渡す必要があります。HTTP トラフィックを渡す一般的な方法は、VSE を Web プロキシとして設定することです。 ただし、「localhost」のような単純な名前に対しては、プロキシの使用がデフォルトで無効になっています。

local.properties ファイルでこの動作を上書きできます。

次の手順に従ってください:

1. DevTest ルート インストール フォルダの local.properties ファイルを開きます。
2. **lisa.http.webProxy.nonProxyHosts.excludeSimple** プロパティのコメントを外します。
3. このプロパティの値を false に設定します。

   lisa.http.webProxy.nonProxyHosts.excludeSimple=**false**
4. ファイルを保存して閉じます。

local.properties のその他の設定の定義

必要に応じて、local.properties で以下の追加設定を行うことができます。

- **lisa.vseName**=VSENAME

  VSE サーバの名前を変更するには、このプロパティを追加し、VSENAME を VSE サーバの名前の値に変更します。

- **lisa.registryName**=REGISTRY or
- **lisa.registryName**=tcp://111.666.11.198:2010/REGISTRY

別のテスト レジストリに接続する方法

1. **local.properties** に **lisa.registryName** プロパティを追加します。
2. REGISTRY の値を新しいテスト レジストリの名前に変更します。

データベース シミュレータのインストール

JDBC トラフィックを記録するには、DevTest シミュレーション JDBC ドライバをデータベース クライアントにインストールします。 データベース クライアントは、実際のドライバの代わりに DevTest ドライバを使用します。

次の手順に従ってください:

1. データベースのクラスパスに、LISA_HOME¥lib ディレクトリ内の JAR ファイル **lisajdbcsim.jar** をコピーします。

   この JAR ファイルには、DevTest シミュレーション JDBC ドライバが含まれます。 データベース クライアントとしてのデモ サーバの使用を支援するため、ドライバは、デモ サーバの DEMO_HOME¥jboss¥server¥default¥lib ディレクトリにすでにコピーされています。

2. ドライバ クラスを **com.itko.lisa.vse.jdbc.driver.Driver** に設定します。 データベース クライアントが使用する JDBC のスタイルに応じて、ドライバ クラスを設定します。

   DriverManager スタイル

   アプリケーション サーバでは、データベース クライアントが接続を取得するために Java DriverManager を使用することはほとんどありません。 アプリケーション サーバがそれを実行する場合は、データベース クライアントの起動コマンドに以下のコマンドを追加します。

   ```
   -Djdbc.drivers=com.itko.lisa.vse.jdbc.driver.Driver
   ```

   このプロパティがすでに使用されている場合は、DevTest ドライバを前に追加します。 DevTest ドライバを、残りのドライバ クラス名からコロン（:）で分離します。

   DataSource スタイル

   データベース クライアントが接続を取得するために DataSource スタイルを使用する場合（デモ サーバと同様）は、その設定を更新します。 設定でデータ ソース定義を指定する場所に、使用する JDBC ドライバとして **com.itko.lisa.vse.jdbc.driver.Driver** を指定し、接続 URL を指定します。

3. パススルーを行うには、DevTest シミュレーション JDBC ドライバが実際のドライバ情報を識別できるように、接続 URL を変更します。 接続 URL は以下の形式にします。

   名前=値[;*名前*=値...]

name

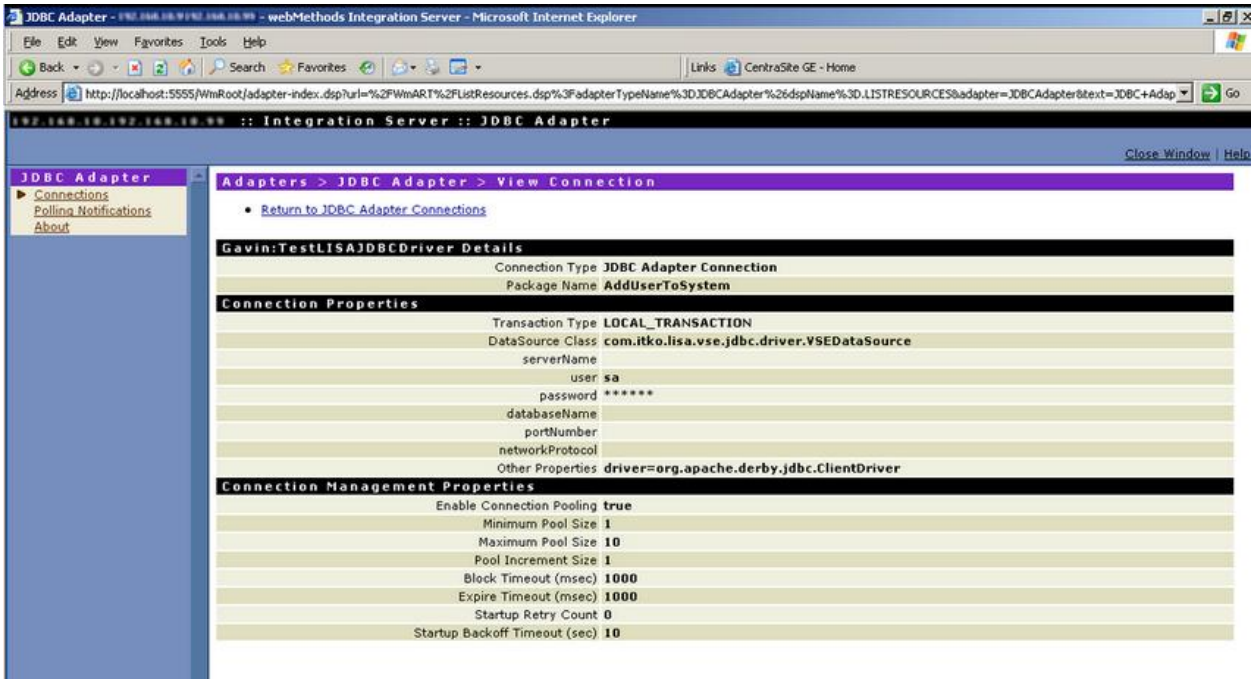
jdbc:lisasim:driver： 値は、使用する実際の JDBC ドライバの完全修飾クラス名である必要があります。

url： 値は、実際のドライバが予期する接続 URL に設定する必要があります（最後のプロパティとして定義する必要があるため、セミコロンを含めることができます）。

注: VSE は、パススルー ドライバとして Oracle シン ドライバをサポートしません。 Oracle シン ドライバは、完全な JDBC 実装を提供しません。 データベースとして Oracle を使用する場合は、仮想化にその他の JDBC ドライバを使用します。

接続 URL の設定例については、**DEMO_HOME¥jboss¥server¥default¥deploy¥itko-example-ds.xml** を参照してください。

以下の図は、WebMethods 環境でデータベースを仮想化する例を示しています。



シミュレーション ドライバと CA Continuous Application Insight ドライバの両方を使用するには、シミュレーション ドライバを「外部」ドライバにします。 CAI クラスおよび URL を指定します。 DevTest デモ サーバに対して DevTest データ ソースを JBoss に定義する例については、**DEMO_HOME¥jboss¥server¥default¥deploy** フォルダの **itko-example-ds.xml** ファイルを参照してください。

その他の起動プロパティ

データベース クライアントの接続タイプに関係なく、データベース クライアントの起動コマンドに以下のプロパティを追加して、シミュレーション ドライバに影響を及ぼすことができます。

lisa.jdbc.sim.require.remote

ドライバが動作するために DevTest ワークステーション または VSE サーバとのアクティブな接続が必要かどうかを指定します。 これは、データベース クライアントと VSE 同期させて、サーバが実行するすべてのデータベース起動アクティビティを記録または再生する最適な方法です。

値

- **true**： DevTest ワークステーション または VSE サーバとのアクティブな接続が行われるまで、ドライバがブロックされます。
- **false**： ドライバはアクティブな接続を必要としません。

デフォルト： false

lisa.jdbc.sim.port

ドライバがレコーダまたは実行中の仮想サービス モデルから接続をリスンする IP ポートを定義します。

デフォルト： 2999

主なデータベース用の DDL

DevTest にファイルの DDL を生成させるには、**local.properties** ファイルに以下の行を追加します。

```
eclipselink.ddl-generation=create-tables
eclipselink.ddl-generation.output-mode=sql-script
eclipselink.target-database=Oracle
```

これらのプロパティを追加して DevTest を起動すると、必要な DDL が含まれる以下のファイルが作成されます。

- createDDL.jdbc
- dropDDL.jdbc

target-database の値を変更することにより、別の DBMS 用の DDL を生成できます。 詳細については、以下を参照してください。

- セッションの EclipseLink 永続性ユニット プロパティ（P. 28）
- スキーマ生成の EclipseLink 永続性ユニット プロパティ（P. 33）

セッションの EclipseLink プロパティ

セッションの EclipseLink 拡張を設定するため、およびターゲット データベースおよびアプリケーション サーバとして、**persistence.xml** ファイルで定義できる EclipseLink JPA 永続性ユニット プロパティを、以下に示します。

eclipselink.session-name

EclipseLink セッションが静的なセッション マネージャに格納される名前を指定します。 このオプションは、JPA のコンテキスト外の EclipseLink 共有セッションにアクセスする必要がある場合に使用します。 また、このオプションは、EclipseLink **sessions.xml** ファイルを介して設定される既存の EclipseLink セッションを使用するためにも使用します。

有効な値： サーバ展開で一意の有効な EclipseLink セッション名。

例：

```
persistence.xml file<property value="MySession"/>
```

例：

```
property Mapimport
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.SESSION_NAME, "MySession");
```

デフォルト： EclipseLink で生成された一意の名前。

eclipselink.sessions-xml

EclipseLink セッション設定ファイル **sessions.xml** からロードされる永続的な情報を指定します。

このオプションは、アノテーションおよび展開 XML の代わりとして使用できます。 このプロパティを指定すると、EclipseLink は、persistence.xml、ORM.xml、およびその他のマッピング ファイルからすべてのクラス アノテーションおよびオブジェクト リレーショナル マッピングを上書きします。

eclipselink.session-name プロパティを設定することによりセッションを指定します。

注: このプロパティの値を指定しないと、**sessions.xml** ファイルは使用されません。

有効な値： セッション XML ファイルのリソース名。

例：

persistence.xml file<property value="mysession.xml"/>例： property Mapimport org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.put (PersistenceUnitProperties.SESSIONS_XML, "mysession.xml");

eclipselink.session-event-listener

ブートストラップ中に追加される記述子イベント リスナを指定します。

有効な値： org.eclipse.persistence.sessions.SessionEventListener インターフェースを実装するクラスの修飾クラス名。

例：

Persistence.xml file<property value="mypackage.MyClass.class"/>例： property Mapimport org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.put (PersistenceUnitProperties.SESSION_EVENT_LISTENER_CLASS, "mypackage.MyClass.class");

eclipselink.session.include.descriptor.queries

記述子からセッションへのすべての指定されたクエリのデフォルト コピーを有効または無効にします。 これらのクエリには、EclipseLink API、記述子修正メソッドなどを使用して定義されるクエリが含まれます。

有効な値：

- **true** - 記述子からセッションへのすべての指定されたクエリのデフォルト コピーを有効にします。
- **false** - 記述子からセッションへのすべての指定されたクエリのデフォルト コピーを無効にします。

例：

```
Persistence.xml file<property value="false"/>例： property Mapimport
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.INCLUDE_DESCRIPTOR_QUERIES, "false");
```

デフォルト： true。

eclipselink.target-database

JPA アプリケーションで使用するデータベースのタイプを指定します。

有効な値：

persistence.xml ファイルの **org.eclipse.persistence.config.TargetDatabase** で使用する有効な値は、以下のとおりです。

- **Attunity** - Attunity データベースを使用するように永続性プロバイダを設定します。
- **Auto** - EclipseLink はデータベースにアクセスし、JDBC が提供するメタデータを使用してターゲット データベースを決定します。 このメタデータをサポートする JDBC ドライバに適用可能です。
- **Cloudscape** - Cloudscape データベースを使用するように永続性プロバイダを設定します。
- **Database** - ターゲット データベースがここにリストされていない場合で、JDBC ドライバが Auto オプションで必要なメタデータの使用をサポートしない場合は、永続性プロバイダを設定して一般的な選択肢を使用します。
- **DB2** - DB2 データベースを使用するように永続性プロバイダを設定します。
- **DB2Mainframe** - DB2Mainframe データベースを使用するように永続性プロバイダを設定します。
- **DBase** - DBase データベースを使用するように永続性プロバイダを設定します。
- **Derby** - Derby データベースを使用するように永続性プロバイダを設定します。
- **HSQL** - HSQL データベースを使用するように永続性プロバイダを設定します。
- **Informix** - Informix データベースを使用するように永続性プロバイダを設定します。
- **JavaDB** - Java DB データベースを使用するように永続性プロバイダを設定します。
- **MySQL** - MySQL データベースを使用するように永続性プロバイダを設定します。
- **Oracle** - Oracle データベースを使用するように永続性プロバイダを設定します。
- **PointBase** - PointBase データベースを使用するように永続性プロバイダを設定します。
- **PostgreSQL** - PostgreSQL データベースを使用するように永続性プロバイダを設定します。
- **SQLAnywhere** - SQL Anywhere データベースを使用するように永続性プロバイダを設定します。

- **SQLServer** - SQL Server データベースを使用するように永続性プロバイダを設定します。
- **Sybase** - Sybase データベースを使用するように永続性プロバイダを設定します。
- **TimesTen** - TimesTen データベースを使用するように永続性プロバイダを設定します。 また、値として org.eclipse.persistence.platform.DatabasePlatform クラスのサブクラスの完全修飾クラス名を設定できます。

例：

```
Persistence.xml file<property value="Oracle"/>例： property Mapimport
org.eclipse.persistence.config.TargetDatabase;import
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.TARGET_DATABASE, TargetDatabase.Oracle);
```

デフォルト値： Auto。

eclipselink.target-server

JPA アプリケーションが使用するアプリケーション サーバのタイプを指定します。

有効な値：

persistence.xml ファイルの **org.eclipse.persistence.config.TargetServer** で使用する有効な値は、以下のとおりです。

- **None** - アプリケーション サーバを使用しないように永続性プロバイダを設定します。
- **WebLogic** - Oracle WebLogic Server を使用するように永続性プロバイダを設定します。 このサーバは、自動的にこのプロパティを設定します。 無効な場合にのみ、設定してください。
- **WebLogic_9** - Oracle WebLogic Server バージョン 9 を使用するように永続性プロバイダを設定します。
- **WebLogic_10** - Oracle WebLogic Server バージョン 10 を使用するように永続性プロバイダを設定します。
- **OC4J** - OC4J を使用するように永続性プロバイダを設定します。
- **SunAS9** - Sun Application Server バージョン 9 を使用するように永続性プロバイダを設定します。 このサーバは、自動的にこのプロパティを設定します。 無効な場合にのみ、設定してください。
- **WebSphere** - WebSphere Application Server を使用するように永続性プロバイダを設定します。

- **WebSphere_6_1** - WebSphere Application Server バージョン 6.1 を使用するように永続性プロバイダを設定します。
- **JBoss** - JBoss Application Server を使用するように永続性プロバイダを設定します。
- **org.eclipse.persistence.platform.ServerPlatform** インターフェースを実装するカスタム サーバ クラスの完全修飾クラス名。

例：

```
persistence.xml file<property value="OC4J_10_1_3"/>例： property Mapimport
org.eclipse.persistence.config.TargetServer;import
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.TARGET_SERVER, TargetServer.OC4J_10_1_3);
```

デフォルト値： None。

スキーマの EclipseLink プロパティ

スキーマ生成を設定するために、persistence.xml ファイルで EclipseLink JPA 永続性ユニット プロパティを定義できます。

eclipselink.ddl-generation

JPA エンティティに対して使用するデータ定義言語（DDL）生成アクションを指定します。 DDL 生成ターゲットを指定するには、「**eclipselink.ddl-generation.output-mode**」を参照してください。

値

persistence.xml ファイルでは以下の値を使用できます。

- **none**： EclipseLink は DDL を生成しません。スキーマは生成されません。
- **create-tables**： EclipseLink は、各テーブルに対して CREATE TABLE SQL コマンドを実行しようとします。 既存のテーブルに対して CREATE TABLE SQL コマンドを発行すると、EclipseLink は、お使いの特定のデータベースおよび JDBC ドライバの組み合わせのデフォルトの動作に従います。 ほとんどの場合、例外がスローされ、テーブルは作成されません。また、EclipseLink は以下のステートメントを続行します。
- **drop-and-create-tables** - EclipseLink は、すべてのテーブルの DROP を試行し、次にすべてテーブルの CREATE を試行します。EclipseLink は、問題が発生した場合、お使いの特定のデータベースおよび JDBC ドライバの組み合わせのデフォルトの動作に従います。 その後、EclipseLink は以下のステートメントを続行します。

以下の値が **org.eclipse.persistence.config.PersistenceUnitProperties** に対して有効です。

- **NONE**
- **CREATE_ONLY**
- **DROP_AND_CREATE**

Java SE 環境で永続性を使用する場合で、テーブルを作成することなく DDL ファイルを作成する場合は、Java システム プロパティ **INTERACT_WITH_DB** を定義し、その値を **false** に設定します。

デフォルト： 以下のいずれかです。

- なし
- PersistenceUnitProperties.NONE

例：

persistence.xml file<property value="create-tables"/>例： property Mapimport org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.put (PersistenceUnitProperties.DDL_GENERATION, PersistenceUnitProperties.CREATE_ONLY);

eclipselink.application-location

EclipseLink が、生成された DDL ファイルを書き込む場所を指定します。**eclipselink.ddl-generation** を **none** 以外に設定すると、ファイルが書き込まれます。

値： ユーザが書き込みアクセス権を持つディレクトリへのファイル指定。 ファイル指定は、現在の作業ディレクトリに対して相対的に指定するか、または絶対的に指定できます。 ファイル区切り記号で終了していない場合、EclipseLink は、お使いのオペレーティング システムで有効な区切り記号を追加します。

デフォルト： 以下のいずれかです。

"."+File.separator

または

<tt>PersistenceUnitProperties.DEFAULT_APP_LOCATION</tt>

例：

persistence.xml file<property value="C:¥ddl¥"/>例： property Mapimport org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.put (PersistenceUnitProperties.APP_LOCATION, "C:¥ddl¥");

eclipselink.create-ddl-jdbc-file-name

EclipseLink が生成する DDL ファイルのファイル名を指定します。このファイルには、JPA エンティティ用のテーブルを作成するための SQL ステートメントが含まれます。eclipselink.ddl-generation に create-tables または drop-and-create-tables が設定されている場合、このファイルは、eclipselink.application-location によって指定されている場所に書き込まれます。

有効な値： お使いのオペレーティング システムで有効なファイル名。eclipselink.application-location + eclipselink.create-ddl-jdbc-file-name の連結が、お使いのオペレーティング システムで有効なファイル指定の場合、必要に応じて、ファイル名の前にファイル パスを付けます。

例：

```
persistence.xml file<property value="create.sql"/>例： property Mapimport
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.CREATE_JDBC_DDL_FILE, "create.sql");
```

デフォルト値： createDDL.jdbc または PersistenceUnitProperties.DEFAULT_CREATE_JDBC_FILE_NAME

eclipselink.drop-ddl-jdbc-file-name

EclipseLink が生成する DDL ファイルのファイル名を指定します。このファイルには、JPA エンティティ用のテーブルをドロップするための SQL ステートメントが含まれます。 eclipselink.ddl-generation に drop-and-create-tables が設定されている場合、このファイルは、eclipselink.application-location によって指定されている場所に書き込まれます。

有効な値： お使いのオペレーティング システムで有効なファイル名。eclipselink.application-location + eclipselink.drop-ddl-jdbc-file-name の連結が、お使いのオペレーティング システムで有効なファイル指定の場合、ファイル名の前にファイル パスを付けることができます。

例：

```
persistence.xml file<property value="drop.sql"/>例： property Mapimport
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.DROP_JDBC_DDL_FILE, "drop.sql");
```

デフォルト値： dropDDL.jdbc または PersistenceUnitProperties.DEFAULT_DROP_JDBC_FILE_NAME

eclipselink.ddl-generation.output-mode

DDL 生成ターゲットを指定するには、このプロパティを使用します。

有効な値：

persistence.xml ファイルで使用される有効な値は以下のとおりです。

both

SQL ファイルを生成し、データベースに対して実行します。

eclipselink.ddl-generation に「create-tables」が設定されている場合、eclipselink.create-ddl-jdbc-file-name が **eclipselink.application-location** に書き込まれ、データベースに対して実行されます。

eclipselink.ddl-generation に「drop-and-create-tables」が設定されている場合、**eclipselink.create-ddl-jdbc-file-name** と **eclipselink.drop-ddl-jdbc-file-name** が **eclipselink.application-location** に書き込まれます。 両方の SQL ファイルがデータベースに対して実行されます。

database

データベースに対してのみ SQL を実行します（SQL ファイルを生成しません）。

sql-script

SQL ファイルの生成のみ行います（データベースに対して実行しません）。

eclipselink.ddl-generation に「create-tables」が設定されている場合、**eclipselink.create-ddl-jdbc-file-name** が **eclipselink.application-location** に書き込まれます。 コマンドはデータベースに対して実行されません。

eclipselink.ddl-generation に「drop-and-create-tables」が設定されている場合、**eclipselink.create-ddl-jdbc-file-name** と **eclipselink.drop-ddl-jdbc-file-name** が **eclipselink.application-location** に書き込まれます。 どちらもデータベースに対して実行されません。 以下の値が **org.eclipse.persistence.config.PersistenceUnitProperties** に対して有効です。

- DDL_BOTH_GENERATION - 「both」を参照してください。
- DDL_DATABASE_GENERATION -「database」を参照してください。
- DDL_SQL_SCRIPT_GENERATION - 「sql-script」を参照してください。

例：

```
persistence.xml file<property value="database"/>例： property Mapimport
org.eclipse.persistence.config.PersistenceUnitProperties;propertiesMap.pu
t(PersistenceUnitProperties.DDL_GENERATION_MODE,
PersistenceUnitProperties.DDL_DATABASE_GENERATION);
```

コンテナまたは Java EE モードのデフォルトは {{database}} です。

注: この設定は、コンテナと特定の EclipseLink サポートによって上書きされます。詳細については、コンテナのドキュメントを参照してください。ブートストラップまたは Java SE モード： **both** または **PersistenceUnitProperties.DDL_BOTH_GENERATION**。

APPC エージェント

APPC エージェントにより、APPC トランザクション プログラムを仮想化できます。 このエージェントは、APPC クライアントとサーバ プログラムの間のプロキシとして機能する APPC の CPI-C API ベースの Java プログラムです。エージェントは、APPC クライアントが APPC サーバに対して送受信するメッセージを転送します。

エージェントは、APPC CPI-C 送信先名で定義されているトランザクション プログラムを仮想化するように設定できます。 VSE レコーダを持つエージェントを使用して、APPC クライアントとサーバ トランザクション プログラムの間で交換される要求および応答を記録できます。 レコーディング モードでは、ライブ クライアントおよびサーバと通信し、交換されるすべてのメッセージを記録します。 エージェントは、VSE に展開できる APPC 仮想サービスを作成します。

再生モードでは、記録された APPC 仮想サービスを展開すると、ライブ クライアントが引き続き APPC エージェントと通信します。 応答は、サーバからではなく、VSE で実行中の APPC 仮想サービスから受信されます。 レコーディングまたは再生が進行中でない場合、エージェントはパススルー モードで動作します。 パススルー モードでは、エージェントは、APPC クライアントとサーバ間で単にメッセージを転送します。

APPC エージェントには以下の制限があります。

- APPC エージェントは、エージェントごとに 1 つの送信先名をサポートします。
- APPC プロトコルは、ライブ呼び出しモードおよびモデルの修正をサポートしません。

APPC プロトコルを使用する方法

1. 「APPC エージェントのインストール (P. 40)」の手順に従って、APPC エージェントをインストールおよび設定します。
2. 「APPC エージェントの設定 (P. 43)」の説明に従って、APPC エージェントの SNA を設定します。
3. 「APPC サービス イメージの記録」のプロセスを使用して、APPC 仮想サービスを記録します。
4. APPC サービスの再生を使用するには、「仮想サービスの展開および実行」の標準の手順に従います。

以下のトピックが含まれます。

APPC エージェントの展開の技術情報

前提条件

IBM Communication Server

システム リソース

メモリ

512 MB 以上の使用可能な RAM

CPU

IBM が推奨する CPU 要件は、APPC エージェントの実行に適切です。

ハード ディスク ストレージ

APPC エージェントは、ファイル システム ログを生成します。 常にログに十分な空き容量を確保するには、少なくとも 500 MB のディスク領域が必要です。 30 ～ 90 日ごとにログ アーカイブを削除します。

Java バージョン

APPC エージェントには、Oracle JDK 1.7.0 以上が必要です。

オペレーティング システム

APPC エージェントは、IBM が Communication Server をサポートするすべての Linux プラットフォームでサポートされています。

ネットワーク

- APPC エージェントは、DevTest 内部の通信ではポート 9000（デフォルト）にバインドされます。
- APPC エージェントは、DevTest レジストリ（デフォルト ポート： 2010）と通信します。
- APPC エージェントは、AMQ を介して DevTest ワークステーション で実行される VSE レコーダと通信します。これは、DevTest レジストリ（デフォルト ポート： 2010）を介して実行されます。
- APPC エージェントは、AMQ を介して VSE と通信します。これは、DevTest レジストリ（デフォルト ポート： 2010）を介して実行されます。

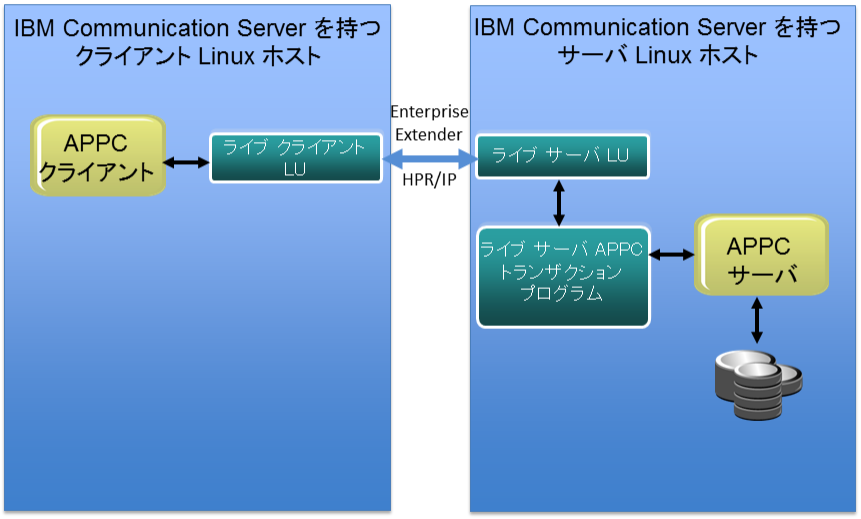
APPC エージェントのインストール

APPC エージェントは UNIX でのみサポートされており、IBM Communication Server が実行されているシステムにインストールする必要があります。このエージェントが正しく動作するためには、APPC CPI-C ライブラリが必要です。

エージェントは、APPC クライアント、サーバ、または独立した OS 上で動作します。 最小限のセットアップが必要であること、および再生時には APPC サーバを完全に切り離せるため、APPC クライアントへのインストールが推奨されています。

以下の図は、APPC エージェントのアーキテクチャを示しています。

APPC トランザクション アーキテクチャの例



エージェントをインストールする方法

1. LISA_HOME に appc-agent という名前のディレクトリを作成します。

   エージェントは、LISA_HOME/addons/appc-agent/appc-agent.zip にあります。

注: APPC エージェントでは、ターゲット コンピュータに DevTest Solutions がインストールされている必要はありません。 既存の DevTest インストールからエージェントの ZIP ファイルをコピーし、それを任意のコンピュータにインストールできます。

2. appc-agent ディレクトリにエージェントを解凍します。

   appc-agent.zip を抽出すると、**lisa-sna-appc.properties** ファイルが LISA_HOME/appc-agent/ のトップ レベルにあります。

3. appc-agent/bin ディレクトリの cpic.jar ファイルをコピーします。

4. 環境変数を使用してエージェントを設定するには、appc-agent ディレクトリの **set-appc-env.sh** シェル スクリプトを編集します。

   シェル スクリプト内の環境変数により、設定プロパティが設定されます。設定プロパティは、プロパティ ファイルで設定されているプロパティを上書きします。 シェル スクリプトで **LIVE_DESTINATION**、**VSE_REGISTRY**、**VSE_LAB**、**APPC_AGENT_PORT**、または **lisa.sna.appc.agent.registry.ping.timeout.millis** プロパティが設定されていない場合、lisa-sna-appc.properties ファイルの値が使用されます。

   プロパティ ファイルを使用してエージェントを設定するには、bin ディレクトリの **lisa-sna-appc.properties** を編集します。 シェル スクリプトで、**LIVE_DESTINATION**、**VSE_REGISTRY**、**VSE_LAB**、**APPC_AGENT_PORT**、および **lisa.sna.appc.agent.registry.ping.timeout.millis** フィールドのパラメータが指定されていない場合は、使用するパラメータを指定します。

   APPC エージェントは、以下の環境変数を使用します。

   LD_LIBRARY_PATH

   SNA 共有ライブラリが存在するディレクトリを定義します。

   LD_PRELOAD

   システムの SNA ライブラリ共有オブジェクトを定義します。

   LIVE_DESTINATION

   エージェントが仮想化するトランザクション プログラムの送信先を定義します。APPC エージェントに含まれるサンプル プログラムでは、ライブ送信先は VSE_SAM です。 VSE_SAM は、サンプルの送信先 CPI-C 名です。

   VSE_REGISTRY=tcp://*host*:*port*/Registry

VSE と DevTest ワークステーション が接続するレジストリ URL を定義します。 DevTest レジストリが実行されている IP アドレスおよびポートを入力します。 ホスト名が解決可能であることと、ホスト名に DNS に関する問題がないことを確認します。

VSE_LAB

エージェント インストールのラボ名を定義します。「Default」は、標準インストールで動作します。

デフォルト: Default

APPC_AGENT_PORT

エージェント サービスがバインドされるポートを定義します。

デフォルト： 9000

lisa.sna.appc.agent.registry.ping.timeout.seconds

エージェントが DevTest レジストリへの接続の待機を停止し、パススルー モードで操作を続行するまでのタイムアウト（秒）を定義します。

デフォルト： 5

APPC エージェントの設定

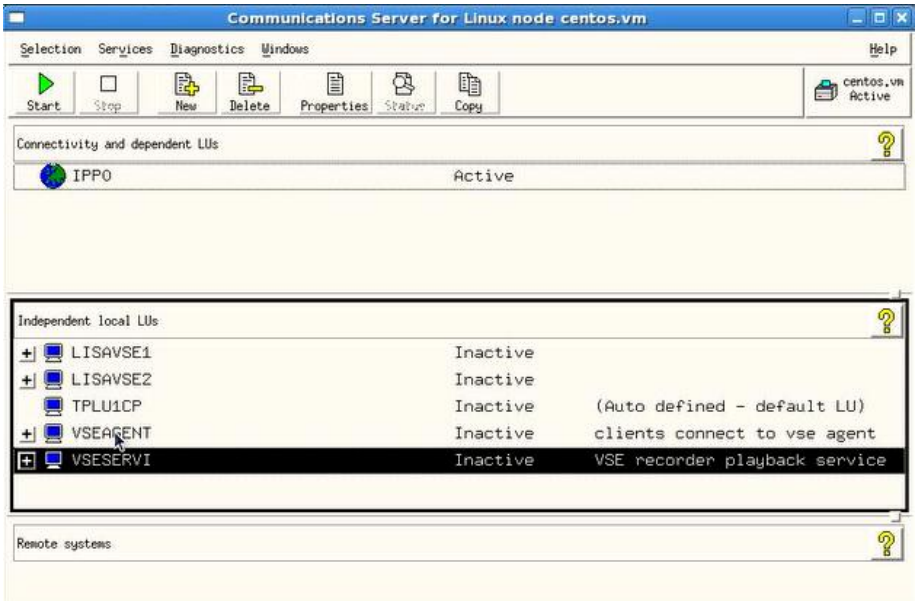
次の手順に従ってください:

1. ログ オプションを指定するには、APPC エージェントの bin ディレクトリの **appc-agent-logging.properties** ファイルを編集します。

   ログ オプションを log4j に設定する方法の詳細については、log4j のドキュメントを参照してください。

2. システムの SNA プロパティを設定します。 SNA Communications Manager にアクセスするには、コマンドラインに「*snaconf*」と入力します。

3. エージェント トランザクション プログラムおよび送信先の設定に使用できるローカルの論理ユニット ペアを設定します。システムに対する適切な値については、SNA 管理者に問い合わせてください。

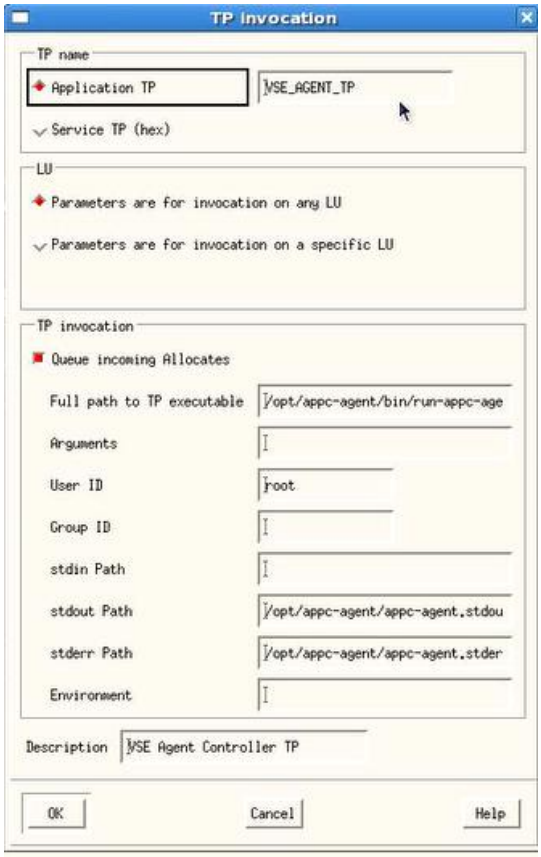


4. ［Services］-［APPC］-［Transaction program］-［Properties］を選択します。

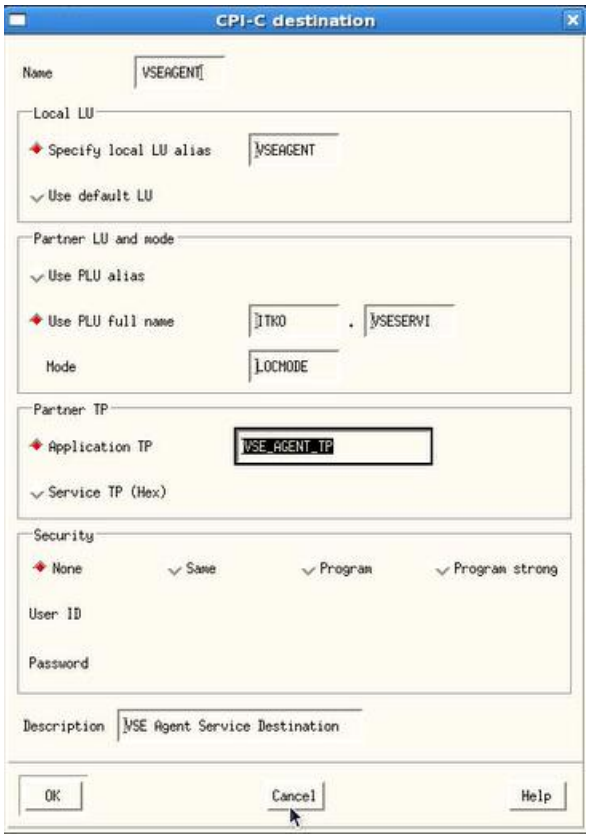
5. エージェント ホーム ディレクトリ /bin/run-appc-agent.sh を指すようにトランザクション プログラムを設定します。

6. すべてのログ ファイルが同じ場所になるように、stdout と stderr のパスをエージェント ディレクトリのいずれかのファイルに設定します。

7. このプログラムがエージェント トランザクション プログラムであることを示す説明を入力します。


TP invocation
TP name
Application TP
VSE_AGENT_TP
Service TP (hex)
LU
Parameters are for invocation on any LU
Parameters are for invocation on a specific LU
TP invocation
Queue incoming Allocates
Full path to TP executable
/opt/appc-agent/bin/run-appc-age
Arguments
User ID
root
Group ID
stdin Path
stdout Path
/opt/appc-agent/appc-agent.stdou
stderr Path
/opt/appc-agent/appc-agent.stder
Environment
Description
VSE Agent Controller TP
OK
Cancel
Help

8. エージェント トランザクション プログラムを設定したら、［Services］-［APPC］-［CPI-C］-［Properties］を選択して、CPI-C 送信先を定義します。



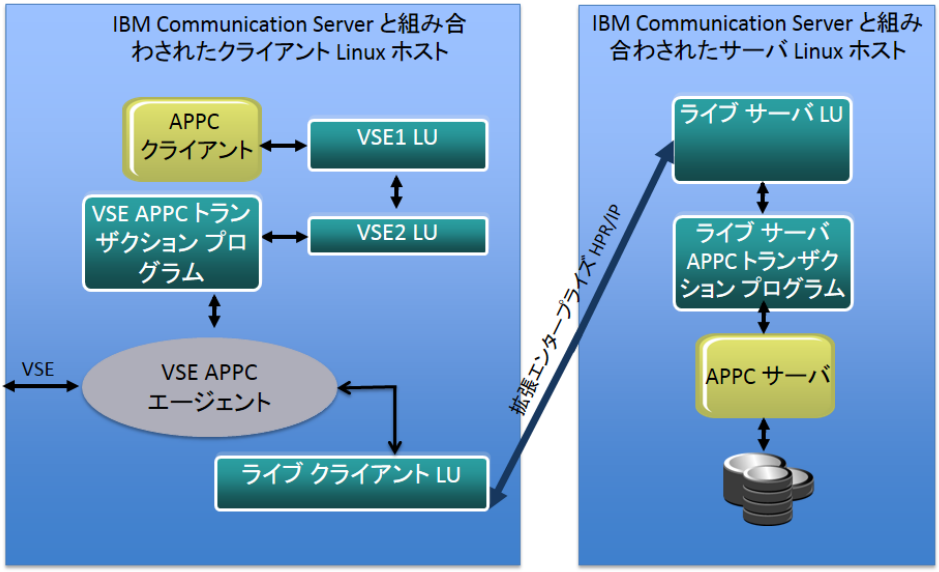
エージェントの設定が完了している場合は、レコーディングを開始するか、記録された仮想サービスを VSE に展開することができます。エージェント送信先名を指すには、クライアント プログラムを使用します。

クライアント プログラムがライブ送信先名を指していている場合は、それを VSEAGENT に切り替えることができます。 送信先名がクライアント プログラムに渡される方法に応じて、クライアント コマンド ライン パラメータ、設定ファイル、またはクライアント コードを変更できます。

DevTest ワークステーション を使用して開始したレコーディングが存在する場合、エージェントはレコーディング モードで記録します。サービスがこのエージェントによるレコーディングに基づいて展開されている場合、再生が行われます。 レコーディングまたは再生が展開されていない場合、サービスはパススルー モードで動作します。パススルー モードでは、ライブ送信先と通信し、ライブ クライアントとライブ サーバ間のすべての要求と応答を転送します。

以下の図は、VSE を持つ APPC エージェントのアーキテクチャを示しています。

VSE でのサンプル APPC トランザクション アーキテクチャ



第 3 章: CA Service Virtualization について

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 Service Virtualization での作業方法

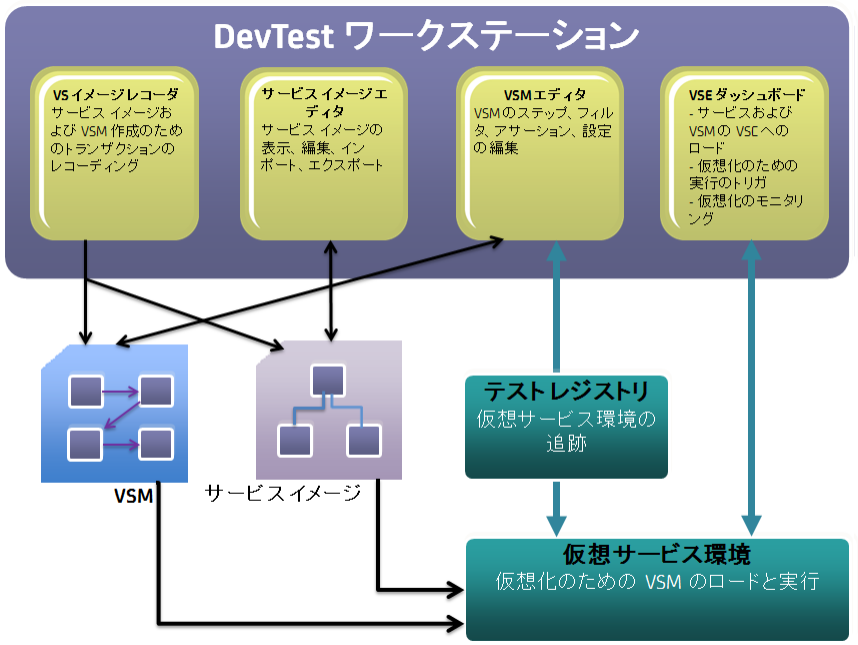
CA Service Virtualization での作業の一般的なプロセスには、以下の手順が含まれます。

1. 仮想サービス イメージ レコーダを起動します。
2. 仮想サービス イメージ レコーダで、記録する項目についての基本的な情報を指定します。適切なプロトコルを選択します。そのプロトコルで必要となるすべての情報を入力します。
3. 仮想サービス イメージ レコーダで、レコーディングを開始します。
4. VSE を介してルーティングされたサーバとのクライアント通信を実行します。

   VSE はトラフィックを記録します。
5. 仮想サービス イメージ レコーダで、レコーディングを完了します。
6. VSE コンソールで、仮想サービス モデルを展開し、仮想サービスを開始します。
7. CA Service Virtualization に対して、ライブ要求を実行します。

CA Service Virtualization のコンポーネント

以下の図は、CA Service Virtualization のコンポーネントの相互関係を示しています。



仮想サービス イメージ レコーダは、サービス イメージと VSM を作成します。 これらは、サービス イメージ エディタおよび VSM エディタで表示および編集できます。

仮想化時には、VSM とサービス イメージが、サービスとして実行される VSE にロードされ実行されます。 テスト レジストリ サービスは、実行中の 1 つ以上の VSE を追跡します。また、DevTest ワークステーション は、テスト レジストリ サービスを使用して VSE に接続します。 VSE ダッシュボードは、VSE にロードされる VSM とサービス イメージのモニタおよび制御に使用される Web UI です。

仮想サービス モデル

仮想サービス モデルは、要求の受信時に実行される一連のステップと考えることができます。 VSM のステップは、要求に対する応答を作成して返します。 仮想サービス モデルは .vsm ファイルに格納されます。

VSE に展開可能にするために、VSM は、DevTest ワークステーション の仮想サービス環境ステップ リストから少なくとも 1 つの VSE ステップを含む必要があります。

サービス イメージを記録した後、VSE によってプロトコル固有のステップが VSM に自動生成されます。生成されたステップは任意に変更できます。以下の操作も実行できます。

- Excel スプレッドシートから応答を入力するか、またはデータベース テーブルを相互参照して入力を計算することで応答を入力する
- 別のステップ タイプのステップを追加する
- 先に進む前に、要求および応答ステップを操作する
- 使用するサービス イメージを応答選択ステップから指定する

仮想化の後、VSM を VSE に展開する必要があります。VSM は、動作パターンの使用方法を定義し、サービス イメージにクエリを実行して応答方法を決定します。 VSM は、サービス イメージをナビゲートする方法を知っています。一般的に、VSM は VSE にのみ展開可能です。一方、テスト ケースはコーディネータにステージングできますが、VSE にはステージングできません。

サービス イメージ

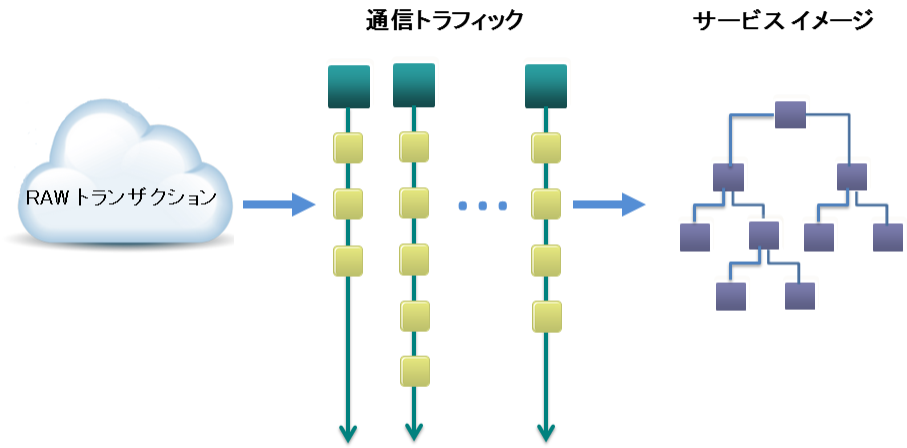
*サービス イメージ*は、CA Service Virtualization によって作成されるクライアントとサーバ間の通信の記録です。 仮想サービス モデルはサービス イメージを参照します。 サービス イメージは、記録された後、サーバが存在しない状態で、クライアントに適切な応答を配信するために使用されます。

サービス イメージには、以下の情報が含まれます。

- 会話ツリーとして記録された会話（要求と応答）のリスト
- ステートレス トランザクション（要求と応答）のリスト
- 不明な会話型またはステートレス要求が発生した場合に送信される応答

サービス イメージ エディタを使用して、サービス イメージを表示、編集、作成できます。

会話は、一連のステートフル トランザクションとして視覚化できます。ただし、（複数のセッションからの）複数の会話を同じサービス イメージに記録できます。 類似の要求構造は単一のトランザクションにマージされ、以下の図に示すようなツリーが作成されます。



たとえば、複数のユーザが **login()** トランザクションでシステムにログインする場合、これらのすべてのトランザクションが単一のトランザクションにマージされます。ただし、1 人のユーザが **login()** を使用してログインし、別のユーザが **acquireAuthToken()** でログインした場合、これらのトランザクションはマージされません。

インポートされたトランザクション

以下の XML ドキュメントを、記録中のサービス イメージにインポートできます。

RAW トランザクション

ネットワークから直接取得されたトラフィックような RAW トラフィックを表す XML ドキュメント。ルート エレメントは、<rawTraffic> です。

例については、**LISA_HOME¥examples¥VServices¥raw-traffic.xml** を参照してください。

会話型トラフィック

会話型トランザクション セット、ステートレス トランザクション セット、またはその両方に再整理されたトラフィックを表す XML ドキュメント。 これらのトラフィックは、トランザクションの線形リストへ整理されます、各トランザクションは「実際」の会話を表し、ルート エレメントは <traffic> です。

例については、**LISA_HOME¥examples¥VServices¥traffic.xml** を参照してください。

仮想化の動作の仕組み

サーバが存在しない場合、CA Service Virtualization は、クライアントのためにサーバの動作をシミュレートします。 このプロセスは、サーバの仮想化と呼ばれています。このプロセスでは、VSM が参照するサービス イメージをロードして、VSE ダッシュボードで実行する必要があります。

VSE が要求を受信すると、その要求が確認され、VSE 内の既存の会話状態（セッション）と照合されます。 たとえば、Cookie の ID またはその他のいくつかのセッション識別子は、HTTP などのステートレス プロトコルの要求を「関連付け」ます。 VSE では会話状態を使用して、「現在のトランザクション」が会話ツリーのどこに位置するか、また以前サブミットされた認証トークンなど、現在の要求の一部ではないが後続の応答で使用されているその他の「状態」を特定します。 例としてユーザ名があります。

既存のセッションが見つからない場合、VSE は要求をイメージの各会話のスタータ トランザクションと照合します。 一致するものがあった場合、セッションが作成され、セッションから関連する応答が返されます。セッションは、VSE によって「参照」された後、2 分間保持されます。 この動作は、**lisa.vse.session.timeout.ms** プロパティによって変更できます。 会話スタータが一致しない場合、セッションは作成されず、定義された順にステートレス トランザクションのリストが確認されます。 一致が見つかった場合は、適切な応答が返されます。 それでも一致が見つからない場合は、「不明な要求」応答が送信されます。

VSE が以前のセッションを検出し、会話の途中の場合、次のトランザクションの一致は多くの要因に依存します。 これらの要因には、ナビゲーションおよび一致許容差が含まれます。

VSE が会話型およびステートレス トランザクションに一致するトランザクションを検出しない場合は、一致しない会話型またはステートレス要求に対して送信する必要のある応答の種類について、もう一度サービス イメージを確認します。

詳細:

会話型要求が処理される方法 (P. 53)

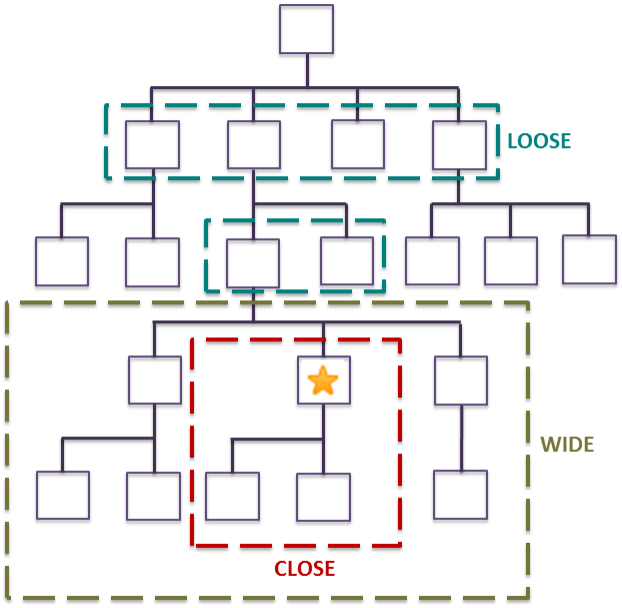
会話型要求が処理される方法

ツリー内のすべてのノードに対して指定できるナビゲーション許容差は、VSM が会話型要求を処理する方法において重要な役割を果たします。 ナビゲーション許容差は、会話ツリーにおいて、指定されたトランザクションに続くトランザクションを VSM が検索する場所を特定するために使用されます。

ナビゲーション許容差レベル

- **CLOSE：** 現在のトランザクションの子が検索されます。
- **WIDE：** 現在のトランザクションとその兄弟および姪/甥が含まれる CLOSE 検索。
- **LOOSE：** 現在のトランザクションの親および兄弟に続いて、開始トランザクションの子が対象となる WIDE 検索。 両方で一致が見つからない場合、すべての会話の開始トランザクションが確認され、会話全体が検索されることになります。

以下の図は、会話ツリーで検索されるトランザクションが、ナビゲーション許容差によってどのような影響を受けるかを示しています。星印 が付いているのは、現在のトランザクションです。



レコーディング時に、VSE レコーダは、以下の設定でトランザクションに対するナビゲーション許容差を初期化できます。

デフォルト ナビゲーション

子のメタ トランザクションを持つすべてのメタ トランザクションに対するデフォルトの許容差を定義します。

デフォルト： Wide

最後

子のメタ トランザクションを持たない「リーフ」トランザクションであるメタ トランザクションに対するデフォルトの許容差を定義します。

デフォルト： Loose

これらのパラメータは、後で DevTest ワークステーション のサービス イメージ エディタを使用して、各ノードに対して変更できます。

デフォルト設定は、「正しい」動作によりよく適合します。 VSE は、現在のランタイム セッションが新しい会話を開始する必要のない場合に会話を再起動する状況で、より正しく応答します。

不明な要求の処理

不明な要求は以下の状況で発生します。

- アクティブな会話が存在する場合
- アクティブな会話が存在しない場合

  アクティブな会話が存在しない場合で、要求を満たすステートレス トランザクションがない場合に、要求は不明として識別されます。 この場合、不明なステートレス要求のサービス イメージ応答が返信になります。

要求が後続のトランザクションに一致しない場合は、以下のようになります。

- ナビゲーション許容差が CLOSE でない場合、会話スタータ トランザクションに要求を満たす機会が与えられます。
- 要求がまだ一致しない場合、ステートレス トランザクションが応答を生成できるときは、それが送信されます。 現在のセッションは、引き続き会話ツリーのどこに存在するかを記憶しています。

失敗した場合、不明な会話型要求に対するサービス イメージ応答が返信になります。

マジック ストリングとマジック デート

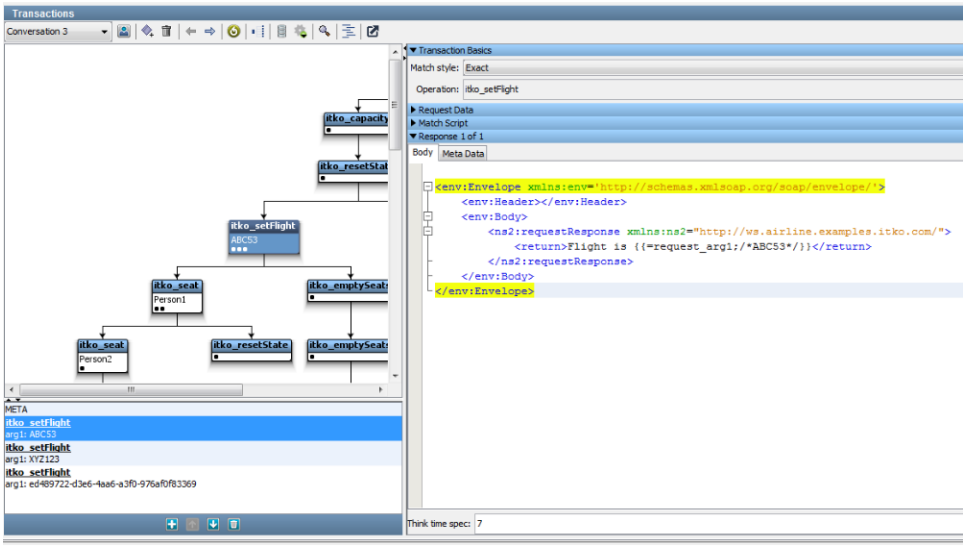
マジック ストリングとマジック デートは、仮想化されたサービスの動的な性質の中心となります。 これらによって、記録されなかった要求パラメータに対して仮想化されたサービスから意味のある結果が返されるようになります。

マジック ストリング

レコーディング時には、VSE によって各要求の引数またはパラメータが確認されます。 たとえば、天気予報 Web サービス コールには都市名が含まれる可能性があり、航空券予約システム要求にはフライト番号が含まれる可能性があります。 記録された応答にフライト番号が含まれている場合、それはマジック ストリングとして分類されます。

たとえば、VSE が再生モードで、記録されていないフライト番号に対する要求を受信した場合、正しいフライト番号は応答に含まれています。

以下の例では、会話で何らかの「状態」を設定する要求を記録しました（この場合はフライト番号（ABC53））。 レコーダは、文字列 ABC53 も応答に含まれていたことを認識して、ABC53 をマジック ストリングに変換します。マジック ストリングは '=request_arg1;/ABC53/ としてサービス イメージ データベースに保存されます。



{{ }} 表記法は、標準の VSE プロパティ代入構文です。この表記法は、「{{ }} テキストに要求の最初の引数のランタイム値を代入する」ことを意味しています。最初の引数が存在しない場合は、デフォルトとして ABC53 を使用します。

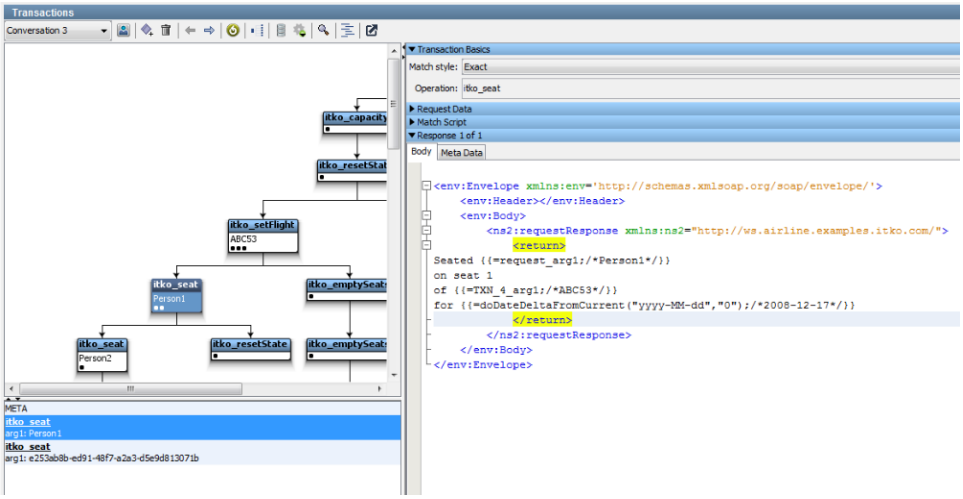
実用的な重要性は、将来のクライアントがフライト ZZ99 を要求した場合、仮想化されたサービスが正しいフライト番号 ZZ99 で応答することです。

VSE は、マジック ストリングを単純な要求/応答ペア以外でも検出します。会話の後続のいずれかの応答に引数がある場合、VSE はそれをマジックと見なし、引数値を会話状態に格納します。

デフォルトでは、VSE は XML タグ内のテキストをマジック ストリングと見なしません。 この除外には、XML タグ、属性名、および属性値が含まれます。 この動作は、**local.properties** ファイルで **lisa.magic.string.xml.tags=true** を設定することによって変更できます。この場合には、通常のマジック ストリング ルールが適用されます。

言いかえれば、**<foo bar="baz"/>** が存在する場合、そのいずれもマジック ストリングとは見なされません。 対照的に、構造が **<foo>bar</foo>** である場合、「bar」（「foo」ではない）はマジック ストリングと見なされます。

以下の例は、前の会話の続きです。 要求操作は「itko_seat」であり、単一の引数は、この場合には「Person1」という名前です。 応答には、要求内の名前、以前に設定されたフライト番号、および日付が含まれます。



これは、ステートレス プロトコル（この場合は、SOAP over HTTP）での会話状態の標準的な例です。 VSE の再生モードで、フライトが ZZ99 に設定され、PersonSomeoneElse の座席を予約する要求があった場合、これらの詳細が記録されなかったとしても、応答には「フライト番号 ZZ99 の座席番号 1 で PersonSomeoneElse 様の席を予約いたしました」という文字列が含まれます。

VSE は、座席番号 (この場合は 1) をマジック ストリングと見なしません。座席番号は、この応答に至るまでの一連の要求では出現していません。また、マジック ストリングと見なされるほど長くもありません。マジック ストリングと見なされるには、少なくとも 3 文字の長さが必要であり、必要に応じて、文字列の左側、右側、または両側に空白を持つことができます。長さおよび空白のパラメータは、**local.properties** ファイルで調整できます。

{{ }} プロパティ表記法は強力かつ柔軟であり、マジック ストリングの使用のみに制限されません。 たとえば、例の応答の「1」を「DataSetValue」に変更することができます。また、仮想サービス モデルにデータ セットを定義することもできます。 その後、そのデータ セットは、ランタイム応答値の生成に使用されます。 データ セットは、ランダム文字列ジェネレータ、カウンタ、データベースへのコール、Excel スプレッドシートの行への参照などにすることができます。 }} は、任意の Java コードを実行し、有効なクレジット カード番号 {{=[:Credit Card:] などの現実的な「日常の」データも生成できます。

詳細については、「*使用*」の「DevTest での BeanShell の使用」を参照してください。

マジック ストリングの除外

VSE は、同一の要求および応答 (特定のルールによって決まります) でトークンを識別し、応答の値を「マジック ストリング」に変換します。マジック ストリングの値は、要求の値によって異なります。

ただし、DevTest には、それらの値が設計によって一致したのか、または偶然に一致したのかを知る方法がありません。 この一致は、以下の値の場合に最もよく見られます。

- ブール値
- Java VSE 用の DevTest エージェントおよび CAI が使用する **__NULL** トークン。 以下に例を示します。

要求

```
<GetUserRequest>
<userId>lisaitko</userId>
<includeDetails>true</includeDetails>
</GetUserRequest>
```

応答

```
<GetUserResponse>
<userId>lisaitko</userId>
<isActive>true</isActive>
<isEmailVerified>true</isEmailVerified>
</GetUserResponse>
```

応答の 2 つの「true」値は、互いに関係がなく、また要求の値とも関係がありません。 実際のシナリオでは、生成される不要な「マジック ストリング」の数は、はるかに多くなります。この問題を回避する方法の 1 つは、サービス イメージが記録された後に、これらのマジック ストリングを手動で検索および置換することです。

ただし、より簡単な方法は、**lisa.properties** で **lisa.magic.string.exclusion** プロパティを設定することです。

このプロパティでは、マジック ストリングの識別の候補にしない値を指定できます。 DevTest は、レコーディング時に、応答のこれらの値を要求の値に関連付けません。 必要に応じて、サービス イメージを手動で編集してマジック ストリングを追加することができます。

マジック ストリングの大文字と小文字の区別

*マジック ストリング*とは、その会話の後続の応答に出現する要求引数の文字列です。 マジック ストリングを検出する場合、一致では大文字と小文字が区別されません。マジック ストリングの大文字と小文字の処理は、以下のロジックによって示されます。

- レコーディング中に要求で「dallas」、応答で「DALLAS」が出現した場合、再生中に「austin」が出現すると、「AUSTIN」で応答します。
- レコーディング中に要求で「DALLAS」、応答で「dallas」が出現した場合、再生中に「AUSTIN」が出現すると、「austin」で応答します。
- レコーディング中に要求で「dallas」、応答で「Dallas」が出現した場合、再生中に「austin」が出現すると、「Austin」で応答します。
- レコーディング中に要求で「dallas」、応答で「DaLLas」が出現した場合、再生中に「austin」が出現すると、「AuSTin」ではなく「austin」で応答します。 サービス イメージで大文字と小文字が混在する応答を明示的に処理する必要がある場合は、メタトランザクションの応答を変更して、カスタム Java コードに以下のようなコールアウトが含まれるようにします。

  ```
  {{=MyHelperClass.mixedCase(request_city);}}
  ```

マジック デート

強力な日付パーサによって要求がスキャンされ、応答もレコーディング中にスキャンされます。 日付形式の広範な定義に一致するものがすべて認識され、マジック デートに変換されます。 「マジック ストリング (P. 56)」に示した例では、マジック デートは以下のようになります。

```
{{ =doDateDeltaFromCurrent("yyyy-MM-dd","0D");/2008-12-17}}
```

ここでは、表記法 {{}} の使用が重要です。

実行時に、この文字列は、「現在の日付から 0 日後である yyyy-MM-dd 形式の日付を生成する」という意味になります。 つまり、現在の日付を生成します。

元のレコーディングが 2009 年 2 月 1 日に行われた場合で、応答に日付 2009-02-10 が含まれる場合には、マジック デート ストリングは以下のようになります。

```
=doDateDeltaFromCurrent("yyyy-MM-dd","10D");/2009-02-10/
```

マジック ストリング内の **10D** は、VSE が現在の時刻の 10 日後の応答の日付を生成することを意味しています。 したがって、VSE が 2010 年 6 月 12 日に再生モードになれば、応答には **2010-06-22** という文字列が含まれます。

日付デルタの有効なパラメータは以下のとおりです。

- **D：** 日
- **H：** 時
- **M：** 分
- **S：** 秒
- **Ms：** ミリ秒

別のタイプのマジック デートとして、以下のものがあります。

```
doDateDeltaFromRequest
```

doDateDeltaFromRequest タイプは、要求で日付がパラメータとして使用され、応答でその日付が参照される場合に使用します。 たとえば、航空券予約システムは、特定の日の特定のフライトの予約要求を受け取る場合があります。 応答でその日付が参照される場合、VSE は、後続の応答でその日付を正しく代入します。

より高度な例として、フライト要求によって、国際日付変更線を越えるフライトを詳細に示す応答が生成される場合があります。 飛行時間が 14 時間であっても、ロサンゼルスからシドニーへのフライトは、出発より 2 日後の日付に到着します。 この例では、応答には以下のようなものが含まれます。

```
doDateDeltaFromRequest("yy-MM-dd", "2D")
```

19-June-2013 に LAX を出発するフライトについて同様の要求を VSE が処理する場合、応答には正しい到着日の 21-June-2013 が含まれます。

注: タイムゾーンを含む任意の有効な日時形式を、マジック デートの計算用に **lisa.properties** に追加できます。

VSE トランザクションについて

CA Service Virtualization では、*トランザクション*は、サービスが作業を実行する単位です。 トランザクションには、クライアントがサービスに送信する要求、およびサービスがクライアントに送信する応答が含まれます。

Web サービス、HTTP、および Java を含むほとんどのサービス プロトコルで、トランザクションは同期して実行されます。 要求と応答は、互いに直接関連付けられています。

Web サービスおよび HTTP では、要求と応答は単一のソケット接続に含まれており、通常は要求に応答が続きます。 Java では、要求と応答は単一のスレッドに含まれています。 応答（戻り値）は、常に要求（メソッド コール）に続きます。

メッセージングは非同期であるため、異なります。 要求メッセージと応答メッセージは、同じソケット接続、同じスレッド、または同じ日に発生する必要はありません。 クライアントは要求メッセージを送信し、応答を待つ間に動作し続けます。 サービスから応答が送信されると、クライアントはそれを受信し、元の応答と一致させてトランザクションを完了します。

HTTP レスポンダの場合のように、通常、トランザクションには単一の応答のみがあります。 ただし、メッセージング プロトコルは、単一の要求に複数の応答を返すことができます。

VSE のステートレス トランザクションと会話型トランザクション

VSE のトランザクションは、*ステートレス*と*会話型*に分類されます。

ステートレス トランザクション

ステートレス トランザクションでは、トランザクション間に論理的な関係が含まれません。 たとえば、HTTP と SOAP はステートレス プロトコルです。 ステートレス トランザクションは、以前行われたコールに関係なく、常に静的な応答を行います。

ステートレスな会話では、各サービス コールは互いに独立しています。たとえば、以下のようになります。

- テキサス州ダラスの天候はいかがですか。 （操作： 天候、引数 1 - 都市）
- ジョージア州アトランタの天候はいかがですか。 （操作： 天候、引数 1 - 都市）
- テキサス州ダラスの明日の天候はどのようになりますか。 （操作： 天候、引数 1 - 都市、引数 2 - 日付）

会話型/ステートフル トランザクション

会話は、ステートフル トランザクションで構成されます。 会話は、一致した場合に一意のセッションを開始する論理トランザクション、およびそのセッションの作成と識別に必要な情報から構成されます。 会話型のトランザクションは、常に以前の会話で作成されたコンテキストに依存しています。

以下に、ATM を使用するステートフルな会話の例を示します。

1. ATM に接続します。 （操作： ログオン）
2. どの口座を使用しますか。 （引数 1 - 照合）
3. 残高はいくらですか。 （操作： 残高）
4. 応答。
5. 現金を引き出します。 （操作： 引き出し）
6. 金額はいくらですか。 （引数 2 - 金額）
7. 残高はいくらですか。 （操作： 残高）
8. 応答（以前の要求で引き出された金額によって異なる）。
9. セッション終了（操作： ログアウト）

会話スタータ

会話の最初のトランザクションは、*会話スタータ*です。 VSE は、受信要求を取得すると、トランザクションが新しい会話の開始を意味しているかどうかを判断するため、会話スタータを確認します。

会話でのトランザクションは、すべて互いに関連しているので、VSE には、この関係を確定する方法が必要です。 この特定は、通常、「jsessionid」などの何らかの特別なトークン、Web トランザクションの Cookie、またはメッセージ トラフィックのカスタム識別子を使用して行われます。

DevTest は、以下のタイプの会話をサポートしています。

インスタンス ベースの会話

プロトコル層で一意の文字列の識別が行われます。これは、クライアントの各インスタンスに基づいています。 その文字列により、サービス イメージのレコーディングと再生の両方に対するサーバ側セッションが識別されます。

トークン ベースの会話

VSE は、サービス イメージの会話に格納されている文字列生成パターンを使用して、トークンを生成します。 トークンが生成された後は、インスタンス ベースの会話と同じ操作が行われます。 トークン ベースの会話は、レコーディング中に自動的に推定することはできません。トークンを検索する場所を指定するには、仮想サービス イメージ レコーダ (P. 322)を使用します。

ナビゲーション許容差

会話型の場合、VSE は、特定のルールを使用して次の会話型トランザクションを検索する方法を決定します。 *ナビゲーション許容差*と呼ばれるルールのセットには、以下のものがあります。

CLOSE

トランザクションは、ツリーを下方向に直接移動する必要があります。次の会話型トランザクションの候補は、現在のトランザクションの子のみです。

WIDE

デフォルトです。 WIDE の許容値では、現在のトランザクション、現在のトランザクションの子、現在のトランザクションの兄弟、および兄弟の直接の子（「甥」）へのナビゲーションが許可されます。 以下に、優先順位を示します。

1. 現在のトランザクションの子
2. 兄弟の子
3. 現在のトランザクションの兄弟

LOOSE

最も許容度の高いナビゲーション許容差。 VSE は、まず CLOSE 許容差と WIDE 許容差を試行し、次に現在のトランザクションの親、親の兄弟（「おじ」）、親の兄弟の子（「いとこ」）の誰かに一致させる機能を追加します。 これが失敗した場合は、ツリーの 2 番目または 3 番目のレベルのトランザクションに対するナビゲーションが許可されます。 以下に、優先順位を示します。

1. 現在のトランザクションの子（CLOSE 許容差）
2. 兄弟の子（WIDE 許容差）
3. 兄弟、または現在のトランザクション（WIDE 許容差）
4. 親のトランザクションの兄弟（"おじ"）、ただしその子（いとこ）は除く
5. 親のトランザクションまたはその兄弟（親または「おじ」）
6. 現在の会話のスタータ トランザクションの子（ツリーのルートの直接の子）
7. SI のすべての会話に対するスタータ トランザクション

論理トランザクション

論理トランザクションは、会話において単一のノードとして出現します。論理トランザクションは、1 つ以上の特定のトランザクションを持つ 1 つのメタ トランザクションで構成されます。

特定の要求に応答するための検索で論理トランザクションが見つかった場合、メタ トランザクションが確認されます。 メタ トランザクションによって、特定の要求にこのトランザクションが応答するかどうかが判断されます。 メタ トランザクションが要求に応答すると判断された場合、すべての特定のトランザクションにクエリが実行されます (それらが要求に応答する場合)。 特定のトランザクションのいずれも要求に応答しない場合、応答はメタ トランザクションから取得されます。

このロジックは、以下の擬似コードで表されます。

```
for 各論理トランザクション ¥{


if (メタ トランザクション要求が特定の要求に一致) ¥{
// このノードは、このノードの特定の各トランザクションの要求を処理 ¥{
if (特定のトランザクションが特定の要求に一致) ¥{
特定のトランザクションから応答を返す
¥}
¥}
// 特定の要求に対して特定のトランザクションが見つからない
メタ トランザクションから応答を返す
¥}
¥}
```

また、物理メタ トランザクションは、特定のトランザクションのリストおよび子のメタ トランザクションのリストを持っています。 後者では、メタ トランザクションをデシジョン ツリーに構造化できます。

一致許容差

*一致許容差*は、特定のトランザクションが受信トランザクションに一致するかどうかを VSE が判断する方法を定義します。

一致許容差のレベルは以下のとおりです。

操作

最も許容度の高い一致許容差です。受信トランザクションの操作名は、記録されたトランザクションの名前に一致する必要があります。

シグネチャ

操作名が一致する必要があり、また引数の名前が追加や削除なしで正確に一致する必要があります。引数の順序は同じである必要はありません。

完全

シグネチャの一致に加えて、引数一致演算子の定義に従って、各引数の値が記録された値と一致する必要があります。

引数一致演算子

要求のすべての引数には、*一致演算子*があります。 完全一致操作では、この演算子によって、受信要求の引数の値をサービス イメージの対応する引数の値とどのように一致させるかを制御します。

使用可能な一致演算子は以下のとおりです。

すべて

常に true を返します。 仮想サービス レコーダは、引数が日付であると判断した場合、デフォルトで「**すべて**」による比較を行います。 この引数の値には、すべての値が一致します。 シグネチャ/メタの一致が機能するには引数が存在する必要がありますが、値は無視され、空白または NULL になります。

= 等しい

値が同じ場合に true を返します。

!= 等しくない

値が異なる場合に true を返します。

<より小さい

受信した値がサービス イメージの値より小さいか、前か、または早い場合に true を返します。

<= 以下

受信した値がサービス イメージの値より小さいか、前か、早いか、または等しい場合に true を返します。

> より大きい

受信した値がサービス イメージの値より大きいか、後か、または遅い場合に true を返します。

>= 以上

受信した値がサービス イメージの値より大きいか、後か、遅いか、または等しい場合に true を返します。

正規表現

受信した値（文字列）が、サービス イメージの値に正規表現として一致する場合に true を返します。

プロパティ式

値は、二重中かっこで囲まれたスクリプト式（{{ }}）の形式である必要があります。このプロパティまたは式は評価されます。結果が Y、y、T、t、または ON で始まる場合、引数は一致したと見なされます。 スクリプトで参照されていない場合、インバウンド要求の引数値は無視されます。

注: 引数が日付としてマークされている場合、比較を行う前に、比較される要求の値が日付に変換されます。

メタトランザクションおよび特定の応答

VSE は、会話の一致を検索する場合、メタ トランザクションのみを検索します。 メタ トランザクションは、「完全」の一致許容差を持つことはできません（デフォルトは「シグネチャ」）。 各メタ トランザクションには、1 つ以上の特定の応答があります。これらの応答は、任意の一致許容差を持つことができます（デフォルトは「完全」）。

メタ トランザクションの特定の応答がいずれも一致しない場合、メタ トランザクションに対して指定された応答が使用されます。

次の応答の選択方法

1. 会話に受信要求が存在する場合、ナビゲーション許容差および上記で説明したその他のルールに基づいて一致を検索します。
2. 現在の会話で何も見つからない場合は、別の会話を検索します（ナビゲーション許容差が CLOSE でない場合）。
3. 会話の一致が存在し、（単なるメタ一致ではなく）特定の応答が見つかった場合は、それを使用します。
4. それ以外の場合には、ステートレス トランザクションで特定の一致を検索し、見つかった場合はそれを使用します。
5. ステートレス リストに特定の一致が存在しない場合で、会話リストでメタ一致が見つかった場合は、そのメタ一致を使用します。
6. 会話の一致がまったく存在しない場合で、ステートレス リストでメタ一致が見つかった場合は、メタ一致を使用します。
7. 「一致が見つかりません」で失敗した場合、サービス イメージで指定された「不明な会話型応答」または「不明なステートレス応答」のいずれかを返します。 使用するプロトコルまたはハンドラによって、応答が上書きされる場合があります。

一致の失敗をデバッグする方法

一致に失敗した場合、**LISA_HOME¥logging.properties** ファイルを開き、**log4j.logger.VSE** を **DEBUG** または **TRACE** に設定します。 この設定により、VSE は **vse_*xxx*.log** ファイル（*xxx* はサービス イメージ名）に詳細なログを記録します。 **log4j.logger.VSE** プロパティにより、何が一致し、何が一致しなかったかも正確に報告されます。

実稼働で使用する場合は、**log4j.logger.VSE** プロパティを **INFO** または **WARN** に設定します。 必要以上に、**DEBUG** または **TRACE** に設定したままにしないでください。

トランザクションの追跡

VSE では、ITR 機能を介して実行されたトランザクションを追跡できます。

ITR で仮想サービスを使用する場合、プロジェクトが使用する設定ファイルで実行モードを設定します。

次の手順に従ってください:

1. ［アクション］-［ITR ランからトラッキング対象の応答を表示］を選択します。
2. 特定の仮想サービスのトランザクションを追跡するには、**lisa.vse.execution.mode** プロパティを TRACK に設定することにより、アクティブな設定ファイルで実行モードを設定します。

   エディタの新しいウィンドウに、追跡されるトランザクションのリストが示されます。

第 4 章: CA Service Virtualization による DevTest ポータルの使用

このセクションには、以下のトピックが含まれています。

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。

第 5 章: 仮想サービスの作成

［Create a Virtual Service ］ウィンドウでは、仮想サービスを記録できます。

仮想サービスを作成するには、CA Service Virtualization レコーダの上部ペインで、［Recording Information］を入力します。

次の手順に従ってください：

1. ［Recording Name］を入力します。 以前に保存されたレコーディングの設定のリストを表示するには、［リスト］をクリックします。レコーダから選択したレコーディング情報をクリアして新規レコーディングを開始するには、［New Recording］をクリックします。

   レコーディングのリストを表示すると、名前、ID、またはステータスによってリストを並べ替えることができます。 レコーディングを削除するには、ハイライト表示し、［削除］  をクリックします。
2. ドロップダウン リストから［VSE サーバ］を選択します。
3. この仮想サービスの説明を入力します。
4. 仮想サービスの記録中にトランザクションを表示するには、［トランザクション］ 

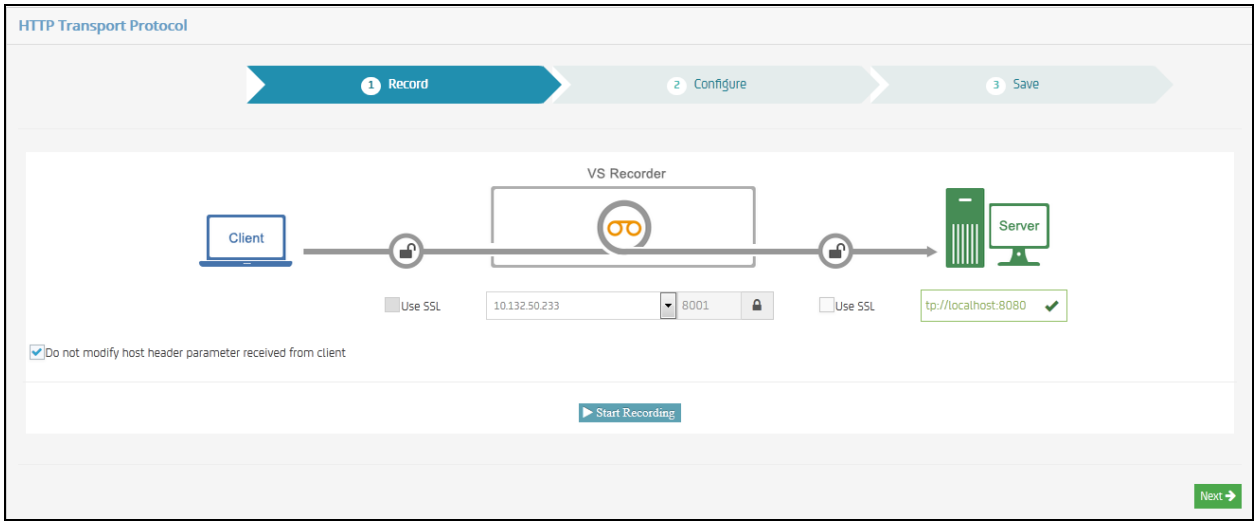
をクリックします。
5. レコーディングのステータスに関する詳細を参照するには、［ステータス］ 

をクリックします。

このセクションには、以下のトピックが含まれています。

Web サイト(HTTP または HTTP/S)の記録

HTTP/S Web サイトからの仮想サービスを記録するには、［Create Virtual Service］ウィンドウの下部ペインに HTTP トランスポート プロトコルに関する情報を入力します。



次の手順に従ってください：

1. CA Service Virtualization が VS レコーダのデフォルト ホストを選択すると、その IP アドレスが表示されます。 レコーダが実行される、このホストの名前または IP アドレスを変更するには、ドロップダウンから別の IP アドレスを選択します。 複数のホスト IP アドレスがあり、それらの IP アドレスのすべてを記録する場合、「すべて」がドロップダウン内の最初のオプションとして提供され、デフォルトで選択されています。
2. VS レコーダのターゲット ポートを入力または変更するには、ロック アイコンをクリックして入力用のフィールドを有効にします。 レコーダが受信要求のリスン ポートとして使用するポート番号を入力します。
3. 記録する Web サイトを指定するには、［Enter Target URL］フィールドにその URL を入力します。 以下の形式を使用できます。
   - <ip>:<port>
   - <hostname>:<port>
   - <server>.<domain>

注: ターゲット URL がこのフィールドへ入力されると、VSE は URL の形式を検証します。 フィールドを完全に入力し、Tab キーまたは Enter キーを押すと、VSE は、入力された URL がアクセス可能であることを確認するためにターゲット サーバへのコールを行います。該当するエラー メッセージが表示されます。

4. ［クライアントから受信したホスト ヘッダ パラメータは変更しないでください］チェック ボックスをオンまたはオフにします。

   オンの場合、このオプションは、クライアント アプリケーションから受信したホスト ヘッダをターゲット サーバに転送するようにライブ呼び出しに指示します。オフの場合、ライブ呼び出しは、ホスト ヘッダ パラメータを再生成して、**host: <***ターゲット ホスト***>:<***ターゲット ポート***>** にします。

   注: 情報を入力するときは、ウィンドウの上部の、無効なエントリが強調表示されるエラー メッセージに注目します。

   注: コンポーネントが検証されるとレコーディングの画像の色が変わることに注目します。上記の図では、VS レコーダおよびサーバは両方とも有色であり、それらが検証されたことを示しています。

   レコーディングのための必須情報がすべて入力されると、レコーディングのステータスが「Draft」から「Ready」に変わります。 ステータスは、青い［ステータス］ボタンに表示されます。 レコーディングに関する詳細情報を取得するには、［ステータス］をクリックします。

5. 以下のいずれかのオプションを選択します。

   - SSL を使用せずに記録するには、［SSL を使用］チェック ボックスをオフにしておきます。 ［次へ］をクリックします。
   - SSL を使用して記録するには、［SSL を使用］チェック ボックスのいずれかまたは両方を選択します。 左側の［SSL を使用］チェック ボックスでは、クライアントに送信される SSL 証明書に関する情報を入力できます。 右側の［SSL を使用］チェック ボックスでは、サーバに送信される SSL 証明書に関する情報を入力できます。

     注: SSL の詳細については、「CA Service Virtualization と SSL (P. 79)」を参照してください。

6. （オプション）［SSL を使用］チェック ボックスのいずれかまたは両方を選択した場合は、以下のフィールドに入力します。

   SSL キーストア ファイル

   キーストア ファイルの名前。

ファイル システムでキーストア ファイルを検索するには、［参照］をクリックします。

SSL キーストア パスワード

指定したキーストア ファイルに関連付けられたパスワードを指定します。

SSL キー エイリアス

公開鍵のエイリアスを指定します。

SSL キー パスワード

キーストア ファイル内の指定したエイリアスに関連付けられたパスワードを指定します。

7. （オプション） SSL パラメータを検証するには、［検証］をクリックします。 SSL フィールドにデータを入力すると、検証が順次実行されます。
8. ［レコーディングを開始］をクリックします。

   注: レコーディング パネルでレコーディングを開始して、いくつかのトランザクションを記録し、［次へ］をクリックしてから、レコーディング パネルに戻るために［前へ］をクリックすると、設定ステップに戻ります。
9. レコーディングが完了したら、［次へ］をクリックします。

CA Service Virtualization と SSL

CA Service Virtualization レコーディング中、SSL サーバ アプリケーションはテスト中のシステムです。

SSL の使用（レコーダとターゲット サーバの間）

レコーダとサーバの間の［SSL を使用］チェック ボックスをオンにすると、レコーダは SSL 接続を使用してターゲット サーバと通信します。 デフォルトでは、レコーダはサーバ証明書をすべて信頼します。

ターゲット サーバがクライアント認証（双方向 SSL）を要求する場合、レコーダは［SSL を使用］チェック ボックスの下のフィールドで指定されている証明書を送信します。 証明書情報が指定されていない場合、レコーダは、レコーディングが実行される VSE サーバの local.properties（**ssl.client.cert.*** プロパティ）ファイルで指定されている値を使用します。

クライアントとレコーダの間での SSL の使用

クライアントとレコーダの間の［**SSL を使用**］チェック ボックスをオンにすると、クライアントは SSL 接続を使用してレコーダと通信します。 レコーディングのこの部分で、レコーダはターゲット サーバの委任であり、クライアントが要求を開始したときにサーバ証明書を送り返す必要があります。 クライアントに送り返される証明書は、［SSL を使用］チェック ボックスの下のフィールドで指定した証明書です。

VSM の再生

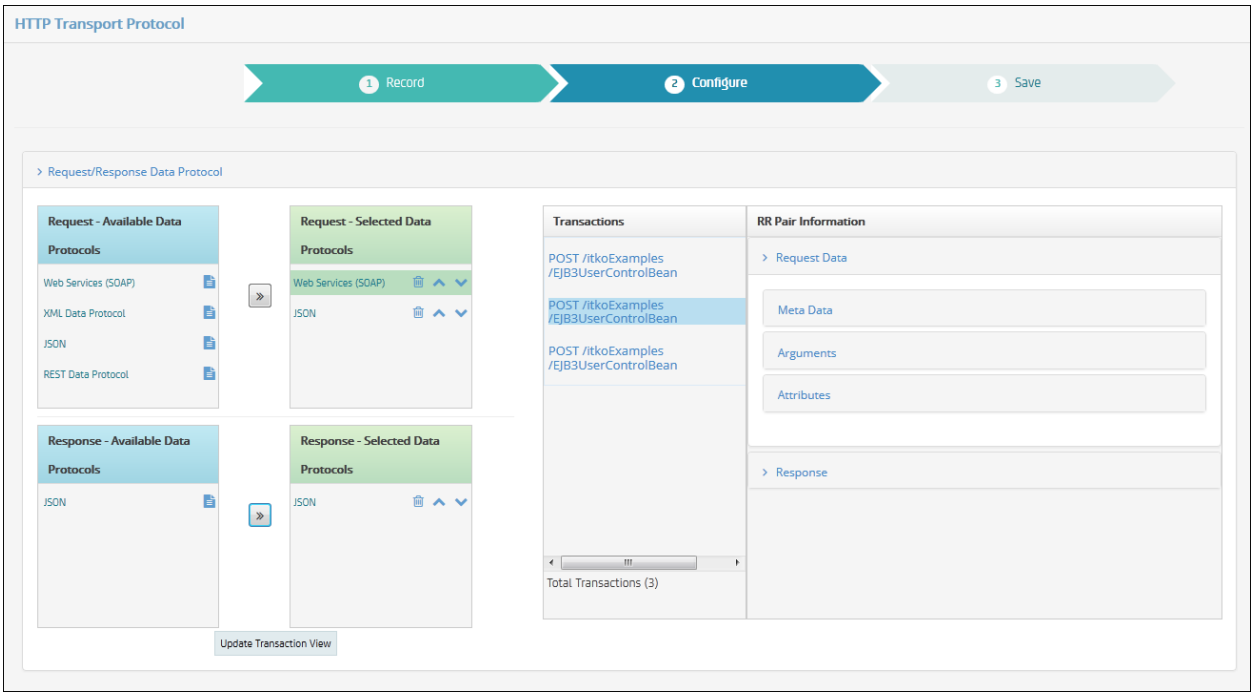
［クライアントに SSL を使用］を選択した場合、SSL ハンドシェイクが、クライアント アプリケーションまたはテスト ケース クライアントと VSM の間で行われます。VSM リスン ステップで提供されるキーストアは、サーバ証明書として一方向認証に使用されます。 クライアントと VSM の間では、SSL ハンドシェイクは行われません。 そのハンドシェイクは通常の HTTP です。

ライブ呼び出しステップが実行された場合、SSL ハンドシェイクは VSM クライアントと実際のサーバの間で行われます。 実際のサーバがクライアント認証を要求する場合、HTTP/S プロトコル ライブ呼び出しステップのキーストアが使用されます。

キーストアに複数の証明書が含まれる場合、CA Service Virtualization は最初のものを使用します。

仮想サービスの設定

［設定］ペインでは、完了したレコーディングで、データ プロトコルを受け入れるか、追加するか、または削除することができます。 また、データ プロトコルがトランザクションに与える影響を確認することもできます。



左側の列には、要求側および応答側で利用可能なデータ プロトコルのリストが表示されます。 自動検出されたデータ プロトコルは緑色で強調表示され、選択リストに表示されます。 要求および応答に利用可能なデータ プロトコルがすべて表示されます。

データ プロトコルを追加するには、データ プロトコルを強調表示して［移動］  をクリックし、［Selected Data Protocols］列に移動します。 選択した列に複数のデータ プロトコルを移動することができます。 選択内容の削除、上への移動、または下への移動を実行するには、列の一番上にあるアイコンを使用します。

トランザクションは、要求および応答のデータと共にペインの右側に表示されます。 ［Update Transaction View］をクリックすると、記録されたトランザクションにこれらのデータ プロトコルが適用されます。 データ プロトコルがトランザクションに適用されると、プロトコルに応じて、要求および応答が別々に解析されます。 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。

更新されたトランザクションは、解析された要求および応答と共に VSE サーバから取得され、［設定］ペインの右側にある［トランザクション］ペインにプレビュー表示されます。

注: ［Update Transaction View］をクリックした場合に限り、選択したデータ プロトコルが適用され、仮想サービスに対して保存されます。 利用可能リストおよび選択リストで加える変更は、UI 変更のみです。

仮想サービスの保存

CA Service Virtualization レコーダの［保存］ペインには、以下のオプションが表示されます。

- Save & Close
- Save & Deploy Virtual Service
- Save & Edit Virtual Service

各オプションについて、以下のフィールドに入力します。

Select Project

仮想サービスが格納されるプロジェクトを指定します。ドロップダウン リストからプロジェクトを選択します。ドロップダウンでリスト表示されるプロジェクトは、Projects ディレクトリに存在するプロジェクトです、

すべてのトランザクションをステートレスとして処理

このオプションは、記録されたトランザクションをすべてステートレスとして処理する特殊な場合に対して提供されています。ほとんどの場合は、このチェック ボックスをオフにします。

デフォルト： 選択済み

特定トランザクションの重複を許可

重複した特定のトランザクションを記録するかどうかを指定します。

デフォルト： オフ

より多くのオプションを表示するには、ウィンドウ右上隅の［Advanced Mode］の［はい］をクリックします。 オプションで、以下のフィールドに入力します。

VSM スタイル

準備ステップを含めて VSM を生成するかどうかを指定します。

値

- **フレキシブル**： 準備ステップを含めます（5 ステップ モデルになります）。
- **効率重視**： 準備ステップは含めません（3 ステップ モデルになります）。

デフォルト： 効率重視

デフォルト ナビゲーション

会話ツリーにおいて、特定のトランザクションに続くトランザクションを VSM が検索する場所を特定するナビゲーション許容差を指定します。最後 (リーフ) のトランザクション以外のすべてに対するデフォルトのナビゲーション許容差を選択します。

値

- CLOSE： 現在のトランザクションの子が検索されます。
- WIDE: 現在のトランザクションとその兄弟および姪/甥が含まれる CLOSE 検索。
- LOOSE: 現在のトランザクションの親および兄弟に続いて、開始トランザクションの子が対象となる WIDE 検索。両方で一致が見つからない場合、すべての会話の開始トランザクションが確認され、会話全体が検索されることになります。

デフォルト： Wide

リーフ ナビゲーション

会話ツリーにおいて、最後（リーフ）のトランザクションに続くトランザクションを VSM が検索する場所を特定するナビゲーション許容差を指定します。

値

- CLOSE： 現在のトランザクションの子が検索されます。
- WIDE: 現在のトランザクションとその兄弟および姪/甥が含まれる CLOSE 検索。
- LOOSE: 現在のトランザクションの親および兄弟に続いて、開始トランザクションの子が対象となる WIDE 検索。両方で一致が見つからない場合、すべての会話の開始トランザクションが確認され、会話全体が検索されることになります。

デフォルト： Loose

Save and Close

［Save & Close］では、仮想サービスが保存されて、［レコーダ］タブが閉じます。

Save and Deploy Virtual Service

［Save & Deploy Virtual Service］では、仮想サービスが保存されて、すぐに展開されます。

以下のフィールドに入力します。

グループ タグ

この仮想サービスの仮想サービス グループ (P. 468)の名前を指定します。展開した仮想サービスにグループ タグがある場合、それらのタグがドロップダウン リストに表示されます。 グループ タグは英数字で始まる必要があり、英数字と以下の特殊文字を含めることができます。

- ピリオド（.）
- ダッシュ（-）
- アンダースコア（_）
- ドル記号（$）

同時実行数

負荷容量を示す数値を指定します。 *容量*は、VSM で同時に何人の仮想ユーザ (インスタンス) が実行できるかを示します。ここでの容量は、このサービス モデルに対する要求を処理するスレッドの数を表します。

VSE は、同時実行数の合計と同数のスレッドを割り当てます。 休止している場合でも、各スレッドはシステム リソースを多少消費します。したがって、システム全体のパフォーマンスを最適化するため、この設定は可能な限り小さな値にします。必要なパフォーマンスを達成するか、または設定値を大きくしてもさらなるパフォーマンス向上が見られなくなるまで設定を調整することにより、適切な設定値を経験的に決定します。

標準装備のプロトコルは、スレッド使用を最小化するため、フレームワーク レベル タスク実行サービスを使用します。 このようなプロトコルの場合、VSM が高度にカスタマイズされていない限り、同時実行数が 1 コア当たり 2 ～ 3 を超えていても、それが役立つことはほとんどありません。

拡張、および標準装備のプロトコルを使用しない VSM では、反応時間を長く設定すると、その間にスレッドを消費する可能性があります。この場合、同時実行数を大きくすることができます。

このような場合のおおよその初期設定値は、以下の数式で与えられます。

同時実行数 = （必要な 1 秒あたりのトランザクション数/1000） * 平均反応時間（ミリ秒) * （反応時間スケール/100）

例：

反応時間を処理するためにフレームワーク タスク実行サービスを使用しない、カスタム プロトコルを使用していると仮定します。全体的なスループットとして必要な、1 秒あたりのトランザクション数は 100 です。 サービス イメージ全体にわたっての平均反応時間は、200 ミリ秒です。また、仮想サービスは、反応時間スケール 100 （%）で展開されます。

（1 秒あたりのトランザクション数 100/1000） * 200 ミリ秒 * （100/100） = 20

この場合、各スレッドは応答前に平均約 200 ミリ秒ブロックされ、その間は新しい要求を処理できません。そのため、100 トランザクション/秒に対応するには、同時実行数 20 が必要です。スレッドは平均で 10 ミリ秒ごとに利用可能になり、100 トランザクション/秒を実現するのに十分となります。

デフォルト： 1

反応時間スケール

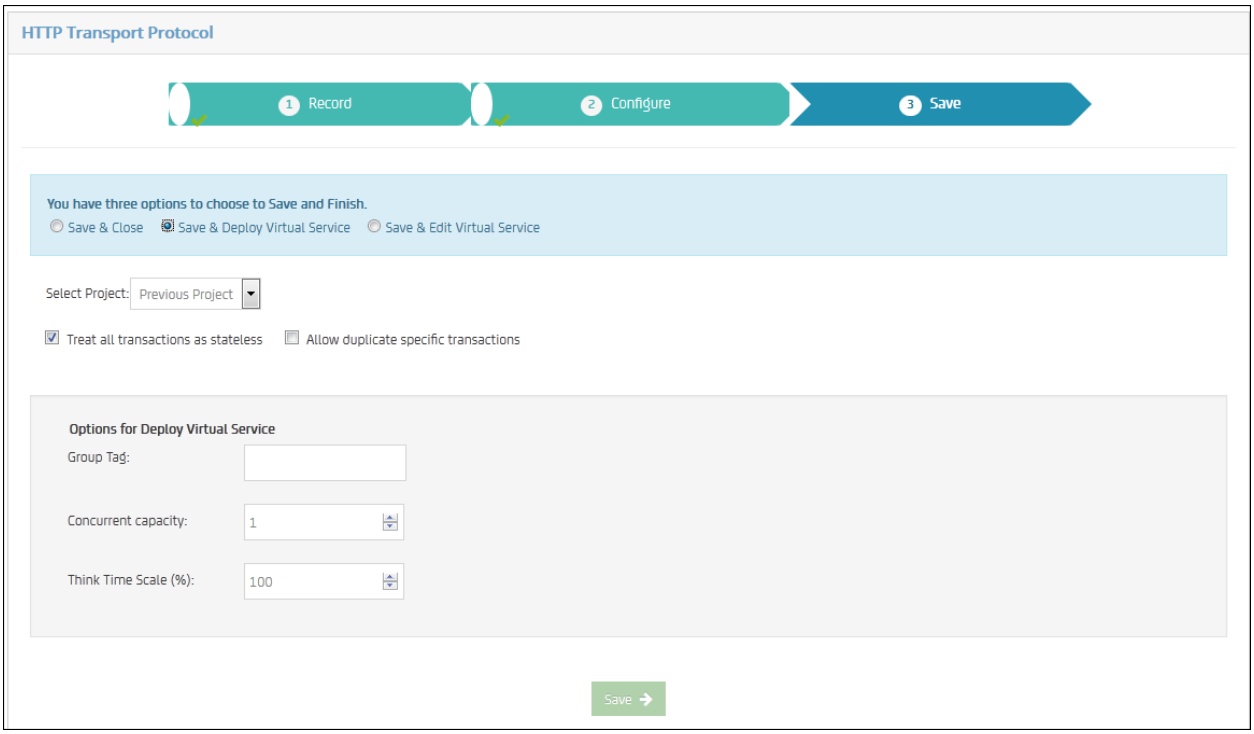
記録された反応時間に対する反応時間の割合（%）を指定します。

注: テストの実行でのペースの整合性を保つため、ステップ自体の処理時間は反応時間から引かれます。

デフォルト： 100

例：

- 反応時間を 2 倍にするには、「200」と入力します。
- 反応時間を半分にするには、「50」と入力します。



Save and Edit

［Save & Edit Virtual Service］では、仮想サービスが保存されて、新しいエディタ タブ内に編集モードで開かれます。

第 6 章: 仮想サービスの編集

DevTest ポータルでは、仮想サービスを開いて、仮想サービスの各種コンポーネントに変更を加えることができます。

DevTest ポータルの基本情報については、「*はじめに*」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。

仮想サービスを開く

DevTest ポータルから仮想サービスを開きます。

次の手順に従ってください:

1. DevTest ポータルのホーム ページに移動します。
2. 左ペインで、［Manage］をクリックします。
3. （オプション）別のプロジェクトを選択します。
4. ［仮想サービス］カテゴリを展開します。
5. 仮想サービス名をクリックします。

ステートレス トランザクション ビュー

ステートレス トランザクションを表示するには、仮想サービスを開いて、［ビュー］ドロップダウン リストから［ステートレス トランザクション］を選択します。

ステートレス トランザクション ビューは、以下の領域に分割されます。

- ［Stateless Signatures］ペイン
- ［Specifics］タブ
- ［Signature Definition］タブ

デフォルトでは、基本機能のみが表示されます。 上級ユーザは、右上領域の［詳細］をクリックすると拡張機能にアクセスできます。

以下の図は、［基本］および［詳細］のリンクを示しています。

Stateless Signatures

VSE がステートレス トランザクションでインバウンド要求との一致を検索しようとする場合、インバウンド要求に一致する *シグネチャ*を最初に検索します。

シグネチャには、以下のコンポーネントがあります。

- 操作名
- （オプション） 一連の引数名

仮想サービスで、操作は、実行されるアクションを表します。「**depositMoney**」は操作の一例です。

仮想サービスで、引数は、操作に含めることができる名前/値ペアです。名前「**金額**」および値「**100.00**」は引数の一例です。

注: VSE は、一致プロセスのこの段階では引数値を考慮しません。

各シグネチャには、一意の識別子もあります。 識別子は仮想サービス内で一意です。

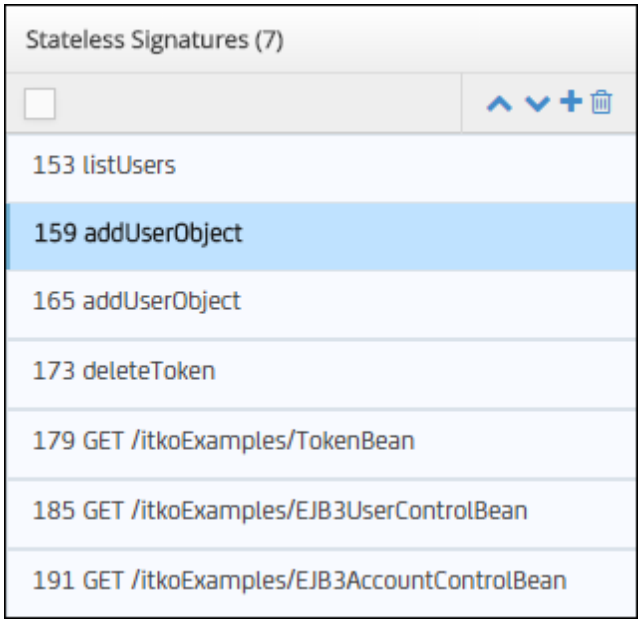
複数のシグネチャには、同じ操作名が存在する可能性があります。 以下に例を示します。

- シグネチャ A には、操作「**depositMoney**」と、1 つの引数「**金額**」があります。
- シグネチャ B には、操作「**depositMoney**」と、2 つの引数「**金額**」および「**日付**」があります。

一意の識別子は、同じ操作名があるシグネチャを区別するのに役立ちます。

同じ操作名があるシグネチャを区別する別の方法は、注 (P. 99)の追加です。

以下の図は、［Stateless Signatures］ペインを示しています。各シグネチャについて、一意の識別子と操作名が表示されます。選択されたシグネチャには、一意の識別子 **159** と操作名 **addUserObject** があります。



引数名を表示するには、シグネチャを選択してから、［Signature Definition］タブを選択します。

要求データ引数および応答データ

VSE が、一致するシグネチャを検出すると、インバウンド要求に一致する*特定のトランザクション*を検索します。

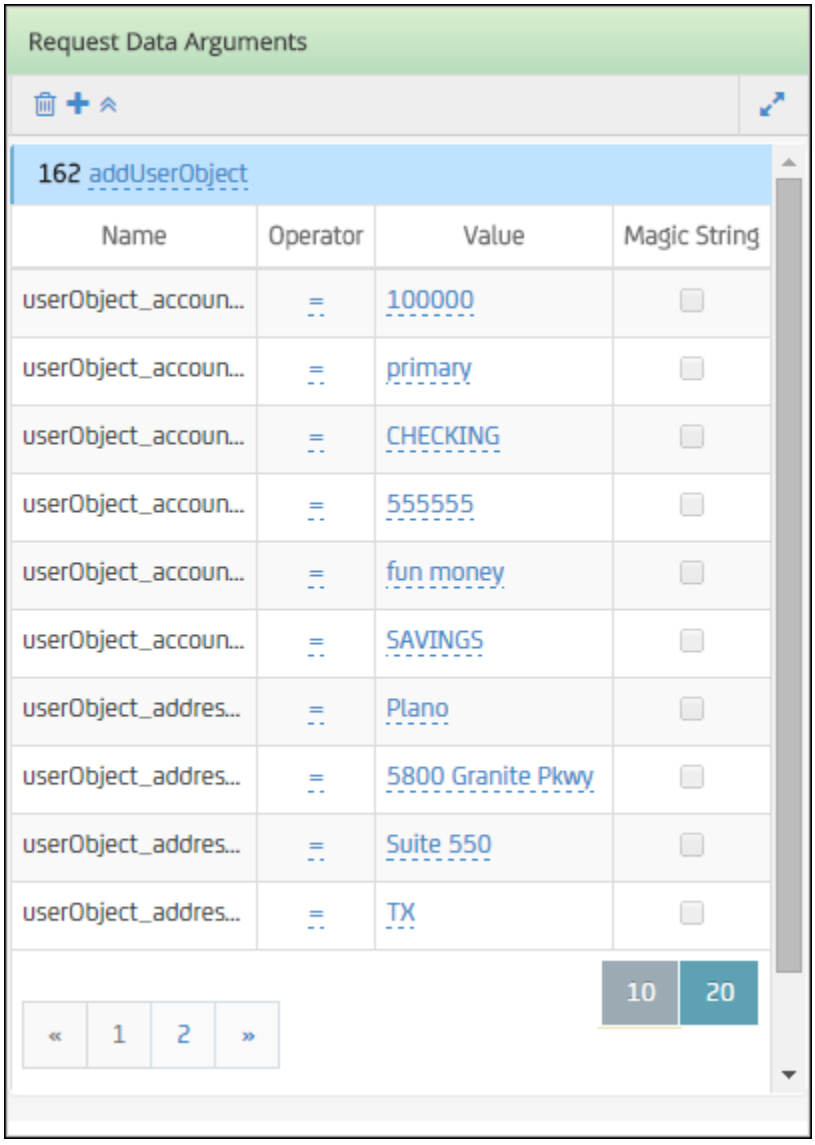
特定のトランザクションには、以下のコンポーネントがあります。

- 操作名
- （オプション）一連の引数名および値と、演算子などのその他の一致基準

特定のトランザクションのそれぞれには、仮想サービス内で一意の識別子もあります。

［Request Data Arguments］ペインには、選択されたシグネチャの特定のトランザクションが表示されます。

以下の図は、特定のトランザクションを示しています。 一意の識別子は **162** です。操作名は **addUserObject** です。引数がテーブルに表示されます。



VSE が、一致する特定のトランザクションを検出すると、そのトランザクションに関連付けられた応答を送信します。

［Response Data］ペインには、選択された特定のトランザクションの応答が含まれています。

以下の図は、［Response Data］ペインを示しています。

応答内でテキストを検索できます。 また、応答内の構文の強調表示を変更することもできます。

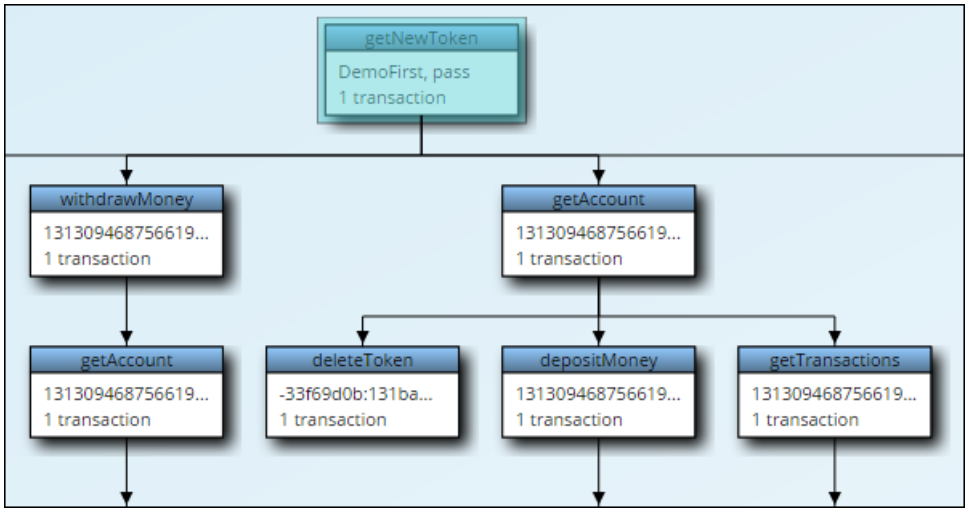
一致する特定のトランザクションが検出されない場合、VSE は、*デフォルト トランザクション*に関連付けられた応答を送信します。

デフォルト トランザクションが［Request Data Arguments］ペインの下部に表示されます。［Response Data］ペインには、選択されたデフォルト トランザクションの応答が含まれています。

Conversation View

会話を表示するには、仮想サービスを開いて、［ビュー］ドロップダウンリストから会話名を選択します。

以下の図は、会話の一部を示しています。 スタータ トランザクションが選択されています。



会話の各ノードは、論理トランザクション (P. 67)です。

ノードの上部領域には、操作名が表示されます。

ノードの主要領域には、以下の情報が表示されます。

- 最初の特定のトランザクションの引数値
- 特定のトランザクションの数

会話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。 この機能は、大量の会話を表示する場合に有用です。

［Request Data Arguments］ペインおよび［Response Data］ペインには、選択したノードに関する詳細情報が含まれています。 これらのペインの動作は、ステートレス トランザクションの場合と同じです (P. 92)。

不明な応答

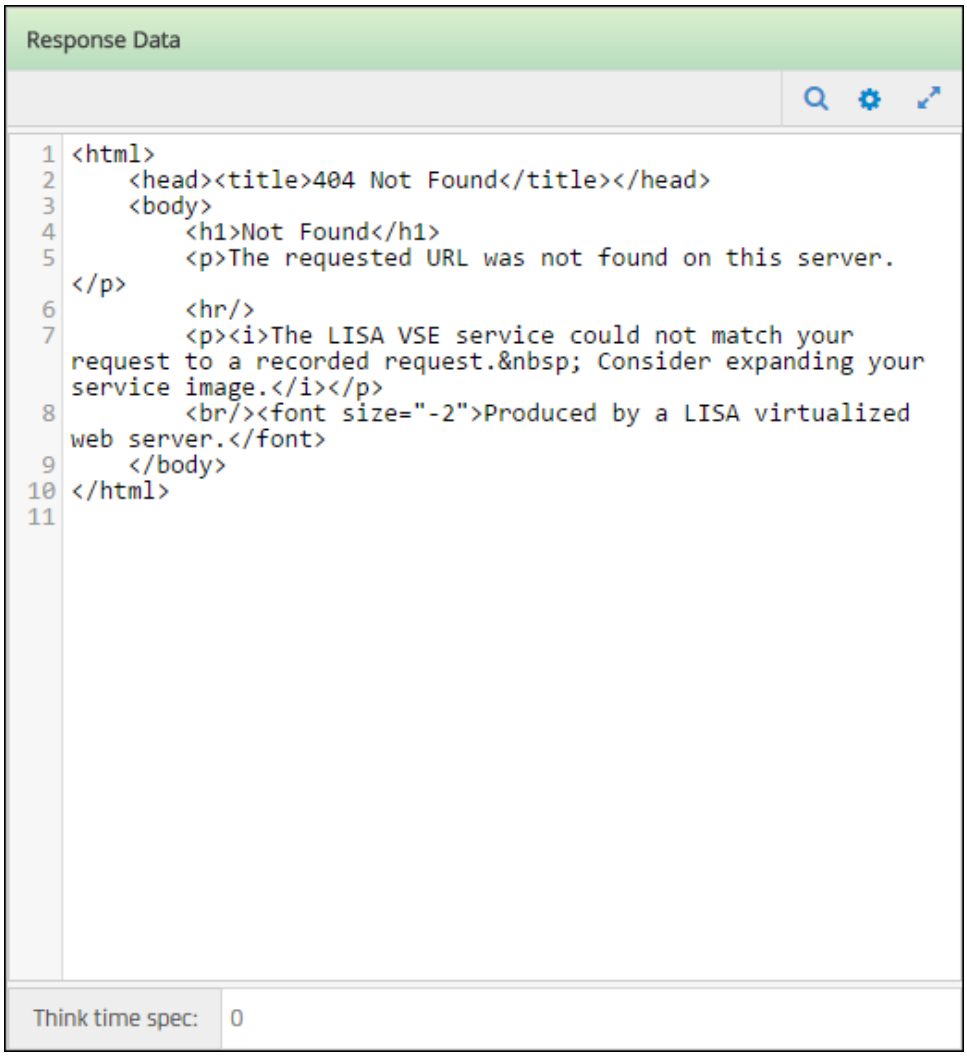
VSE が、インバウンド要求との一致を検出できない場合、以下のいずれかの応答を送信します。

- 不明な会話型要求用の応答
- 不明なステートレス要求用の応答

不明な応答を表示するには、仮想サービスを開いて、［ビュー］ドロップダウン リストから［Service Image Properties］を選択します。

応答内でテキストを検索できます。 また、応答内の構文の強調表示を変更することもできます。

以下の図は、不明な応答を示しています。


Response Data
<html>
<head><title>404 Not Found</title></head>
<body>
<h1>Not Found</h1>
The requested URL was not found on this server.

<hr/>
The LISA VSE service could not match your
request to a recorded request. Consider expanding your
service image.

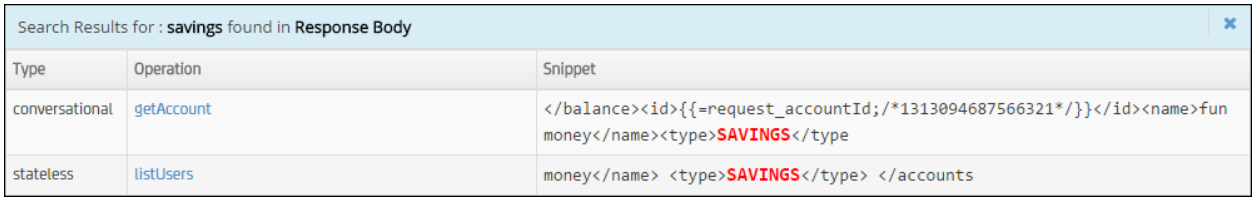
Produced by a LISA virtualized
web server.
</body>
</html>
Think time spec:
0

仮想サービス内のテキストの検索

仮想サービスの以下のコンポーネントにわたってテキストを検索できます。

- シグネチャ名
- 特定のトランザクションの名前
- 要求データ
- 応答データ
- 一意の識別子
- 注記

以下の図は、［検索結果］テーブルを示しています。



［タイプ］列は、結果がステートレス トランザクションまたは会話のどちらの一部であるかを示しています。

結果が応答ボディにある場合、テキストの場所が［Snippet］列に強調表示されます。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. 検索フィールドに、テキストを入力します。

   入力すると、候補が検索フィールドの下に表示されます。
3. 候補をクリックするか、または Enter キーを押します。

   ［検索結果］テーブルに、一連の結果が表示されます。
4. テーブル内のエントリをクリックします。

   そのテキストが含まれるコンポーネントがエディタに表示されます。

シグネチャへの注の追加

注の追加により、シグネチャに注釈を付けることができます。

たとえば、シグネチャが、まれに発生するシナリオを表すと仮定します。シナリオの説明を提供する注を追加できます。

また、シグネチャのデフォルト設定を変更したことを示す注を追加することもできます。

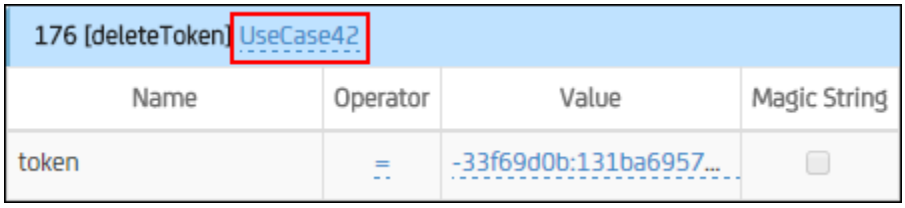
次の手順に従ってください:

1. 仮想サービスを開きます。
2. シグネチャを右クリックし、［Add Note］をクリックします。
3. テキストを入力します。 注の内部でコピーと貼り付けが可能です。
4. ［保存］アイコンをクリックします。

特定のトランザクションへのラベルの追加

ラベルの追加により、特定のトランザクションに注釈を付けることができます。

以下の図は、ラベルが含まれる特定のトランザクションを示しています。ラベルが強調表示されています。



次の手順に従ってください:

1. 仮想サービスを開きます。
2. 特定のトランザクションの操作名をクリックします。

   テキスト フィールドが表示されます。
3. ラベルを入力し、チェック マークをクリックします。

   操作名の右側にラベルが表示されます。

手動での仮想サービスの更新

DevTest ポータルから仮想サービスを手動で更新できます。

仮想サービスを更新する理由の一部を以下に示します。

- 仮想サービスの記録中に、操作を呼び出すことを忘れる。
- 開発チームが、仮想化されているサービスに新しい操作を追加する。
- サービスの変更案を、変更が実装される前にテストする。

広範な更新が必要な場合は、別の方法を考慮します。

- 仮想サービスと別の仮想サービスの組み合わせ (P. 119)
- 実行モードがイメージの検証に設定された仮想サービスの実行
- 仮想サービス全体の再記録

シグネチャの追加、変更、および削除

DevTest ポータルでは、以下のアクションのいずれも実行できます。

- シグネチャの追加
- シグネチャの上下への移動
- シグネチャの定義の変更
- シグネチャの削除

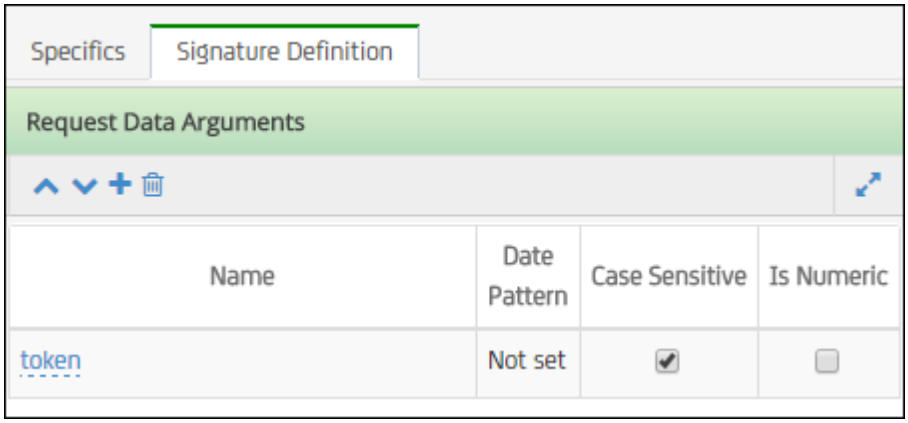
また、要求/応答ペアをインポート (P. 103)してシグネチャを追加することもできます。

VSE がインバウンド要求に一致するシグネチャを検索する場合、シグネチャのリストの一番上から開始し、下方に続行します。 そのため、シグネチャの上下移動は、一致が発生する順序に影響します。

シグネチャ定義の変更は、シグネチャに対応する特定のトランザクションのすべてに自動的に適用されます。たとえば、**NewArg** 引数を追加すると、特定のトランザクションにこの新しい引数が含まれるようになります。

デフォルトでは、一致は大文字と小文字を区別します。比較演算子 (P. 105)で引数の大文字と小文字を区別しないようにするには、［Signature Definition］タブに移動し、［大文字と小文字を区別］列のチェック ボックスをオフにします。

以下の図は、［Signature Definition］タブを示しています。



デフォルトでは、引数値は文字列として処理されます。 たとえば、アルファベット順で「1」は「9」より前に来るため、「10000」は「9」より小さいと見なされます。 引数値が数値として処理されるようにするには、［Signature Definition］タブに移動し、［数値］列のチェック ボックスをオンにします。

日付パターンは、日付情報を解析するために使用されます。［日付パターン］列は読み取り専用です。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. シグネチャを追加するには、以下の手順に従います。
   a. 詳細ビューが表示されていることを確認します。
   b. ［Stateless Signatures］ペインで、［追加］ボタンをクリックして［Add New Signature］を選択します。
   c. 操作名を入力します。
   d. ［続行］をクリックします。
   e. 新しいシグネチャを選択します。
   f. （オプション）［Signature Definition］タブに移動して、1 つ以上の引数を追加します。
   g. ［Specifics］タブに移動して、1 つ以上の特定のトランザクションを追加します。
3. シグネチャを上下に移動するには、シグネチャを選択し、［上へ］アイコンまたは［下へ］アイコンをクリックします。 また、新しい場所にシグネチャをドラッグすることもできます。
4. シグネチャの定義を変更するには、以下の手順に従います。
   a. シグネチャを選択します。
   b. ［Signature Definition］タブに移動します。
   c. 変更を加えます。 引数を上下に移動できます。 引数を追加または削除できます。 ［大文字と小文字を区別］列および［数値］列の設定を変更できます。 操作名を変更できます。
5. シグネチャを削除するには、シグネチャを選択して［削除］アイコンをクリックします。 Ctrl キーを使用して、複数のシグネチャを選択することができます。

要求/応答ペアのインポート

要求/応答ペアは、1 つの要求ファイルおよび 1 つ以上の応答ファイルから構成されます。

要求/応答ペアをインポートすると、要求は仮想サービスに対して処理されます。

仮想サービスに同じ引数を持ったシグネチャが含まれる場合、特定のトランザクションがシグネチャに追加されます。

仮想サービスに同じ引数を持ったシグネチャが含まれない場合、シグネチャが追加されます。 要求および応答の詳細が、特定のトランザクションとしてシグネチャに追加されます。 デフォルト トランザクションも追加されます。

要求/応答ペアは、テキスト形式または XML 形式にする必要があります。

以下のリストは、要求/応答ペアに含まれるファイル名の例です。

- **depositMoney-req.xml**
- **depositMoney-rsp1.xml**
- **depositMoney-rsp2.xml**
- **depositMoney-rsp3.xml**

要求ファイル名には、後に文字列 **-req** が続くプレフィックスが含まれている必要があります。

応答ファイル名には、後に文字列 **-rsp** が続く、同じプレフィックスが含まれている必要があります。 上記の例が示すように、文字列 **-rsp** の後に数を追加して、複数の応答ファイルを提供できます。 複数の応答ファイルは、メッセージング シナリオに適用可能です。

1 つ以上のサイドカー ファイルを含めることにより、要求および応答のメタデータを指定できます。 詳細については、「要求/応答ペアを持ったサイドカー ファイル (P. 135)」を参照してください。

注: 仮想サービスが HTTP 以外のトランスポート プロトコルに基づいている場合、この機能には、「DevTest ワークステーションで対応する仮想サービス モデルを開く」および「レコーディング セッション ファイルへのパスを[ドキュメント]フィールドに入力する」という前提条件があります。 レコーディング セッション ファイルは、以下のいずれかのファイルです。

VRS ファイル

DevTest ワークステーションの仮想サービス イメージ レコーダによって生成されます。

VR2 ファイル

DevTest ポータルの CA Service Virtualization レコーダによって生成されます。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. ［Stateless Signatures］ペインで、［追加］ボタンをクリックして［Import Request/Response Pairs］を選択します。

   ［Import Request/Response Pairs］ダイアログ ボックスが表示されます。
3. 要求/応答ペアが含まれるファイルを指定します。 ファイルをダイアログ ボックスにドラッグできます。または、ボタンをクリックしてファイル システムからファイルを選択することができます。
4. ［完了］をクリックします。

引数の比較演算子

特定のトランザクションの各引数には、比較演算子があります。 演算子は、引数値をインバウンド要求と一致させる方法を示します。

デフォルトでは、演算子は等号（=）に設定されます。 そのため、インバウンド要求での引数値が特定のトランザクションの引数値と同じである場合、一致が発生します。

以下の数学記号もサポートされています。

- 等しくない（!=）
- より小さい（<）
- 以下（<=）
- より大きい（>）
- 以上（>=）

また、正規表現またはプロパティ式も指定できます。これらの演算子は、［値］列と共に使用します。

たとえば、**750** で始まる任意の 5 桁の郵便番号が有効と仮定します。 演算子を［正規表現］に設定し、値を以下のテキストに設定することができます。

```
750¥d¥d
```

引数として任意の値を許可する場合は、演算子を［すべて］に設定します。

以下の図は、各種演算子の使用を示しています。 このシナリオでは、インバウンド要求の引数値は特定のトランザクションの基準にすべて一致しています。 たとえば、インバウンド要求の勘定残高は 5000 です。 特定のトランザクションは、勘定残高が 100 を超えている必要があることを示しています。

インバウンド要求

名前	値
AccountType	当座預金
AccountBalance	5000
AccountName	名前 2

特定のトランザクション

名前	演算子	値
AccountType	=	当座預金
AccountBalance	>	100
AccountName	すべて	名前 1

マジック ストリングの無効化および有効化

特定のトランザクションの各引数は、マジック ストリング (P. 56)として分類できます。

DevTest ポータルで特定のトランザクションを表示すると、［マジック ストリング］列に、各引数がマジック ストリングとして分類されたかどうかが示されます。

［マジック ストリング］列は読み取り専用です。 DevTest ワークステーション では、設定を変更する方法が提供されます。詳細については、「要求データ エディタ (P. 324)」を参照してください。

要求属性の編集

仮想サービスで、*属性*は、要求ボディに関する情報が含まれる名前/値ペアです。

HTTP トランスポート プロトコルを使用して仮想サービスを記録すると仮定します。 仮想サービスの属性には、XML ネームスペースおよび記録された RAW 要求が含まれます。

要求属性は、一致では使用されません。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. 詳細ビューが表示されていることを確認します。
3. ［要求データ］ラベルの右側にある［属性］リンクをクリックします。
4. 必要に応じて、属性を追加、変更、または削除します。

要求メタデータの編集

仮想サービスで、用語「メタデータ」は、要求または応答ボディの一部ではない情報が含まれる名前/値ペアを指します。

HTTP トランスポート プロトコルを使用して仮想サービスを記録すると仮定します。 仮想サービスの要求メタデータには、HTTP メソッドおよびコンテンツ タイプが含まれます。

要求メタデータは、一致では使用されません。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. 詳細ビューが表示されていることを確認します。
3. ［要求データ］ラベルの右側にある［Metadata］リンクをクリックします。
4. 必要に応じて、メタデータを追加、変更、または削除します。

応答メタデータの編集

仮想サービスで、用語「メタデータ」は、要求または応答ボディの一部ではない情報が含まれる名前/値ペアを指します。

HTTP トランスポート プロトコルを使用して仮想サービスを記録すると仮定します。 仮想サービスの応答メタデータには、HTTP 応答コードおよびコンテンツ タイプが含まれます。

応答メタデータは、一致では使用されません。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. 詳細ビューが表示されていることを確認します。
3. ［応答］ラベルの右側にある［Metadata］リンクをクリックします。
4. 必要に応じて、メタデータを追加、変更、または削除します。

応答の反応時間の変更

特定のトランザクションの各応答には、［反応時間］という名前のフィールドが含まれます。 このフィールドでは、応答を送信するまでの待機時間を指定します。 サービス イメージがレコーディングから作成された場合、初期値はレコーディング中に観察された応答動作に基づきます。

パフォーマンス向上をシミュレートするには、値を減らします。

パフォーマンス低下をシミュレートするには、値を増やします。

デフォルトでは、値はミリ秒単位です。 時間の単位を指定するには、サフィックスを数値に追加します。 大文字と小文字は区別されません。 以下のサフィックスを使用できます。

- **t：** ミリ秒
- **s：** 秒
- **m：** 分
- **h：** 時

数値の範囲を指定できます。 反応時間は、指定した範囲からランダムに選択されます。 たとえば、値「**100-1000**」は、100 ～ 1000 ミリ秒のランダムな反応時間を指定します。

反応時間が 10 ミリ秒で、要求を処理して応答を検索するのに 5 ミリ秒かかる場合、5 ミリ秒の遅延が余分に加算されるのみです。

反応時間が 0 であるか、または要求の処理に要した時間より短い場合、VSE は応答を可能な限り迅速に送信します。

仮想サービスの不明な応答にも同様のフィールドがあります。

仮想サービスを展開する (P. 111)場合、［反応時間スケール］という名前の関連フィールドを設定できます。このフィールドでは、サービス イメージ内の［反応時間］フィールドすべてに掛ける割合（%）を指定します。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. 特定のトランザクションの応答、または不明な応答を見つけます。
3. ［反応時間］フィールドの値を変更します。

要求に一致するトランザクションの検索

仮想サービスに要求を送信すると、予期されるものとは応答が異なると仮定します。

仮想サービスを開いて、要求を入力することができます。 DevTest ポータルでは、仮想サービスのどのコンポーネントが要求に一致するかが示されます。 この結果は、問題の修正方法を決定するのに役立ちます。

要求は、テキスト形式または XML 形式にする必要があります。

DevTest ポータルでは、要求の実際のコンテンツを入力するか、または要求ファイルを指定することができます。 要求ファイル名には、後に文字列 **-req** が続くプレフィックスが含まれている必要があります（たとえば、**depositMoney-req.txt**）。

結果にステートレス シグネチャ、またはステートレスの特定のトランザクションが含まれる場合、表示される順序は、再生中に一致が発生する順序と同じです。

会話型結果の順序は、重大ではありません。

注: 仮想サービスが HTTP 以外のトランスポート プロトコルに基づいている場合、この機能には、「DevTest ワークステーションで対応する仮想サービス モデルを開く」および「レコーディング セッション ファイルへのパスを［ドキュメント］フィールドに入力する」という前提条件があります。 レコーディング セッション ファイルは、以下のいずれかのファイルです。

VRS ファイル

DevTest ワークステーションの仮想サービス イメージ レコーダによって生成されます。

VR2 ファイル

DevTest ポータルの CA Service Virtualization レコーダによって生成されます。

次の手順に従ってください:

1. 仮想サービスを開きます。
2. ［Find a Match by Request］をクリックします。
3. 以下のいずれかの操作を実行します。
   - 要求をテキスト領域に貼り付ける。

- 要求ファイルをダイアログ ボックスにドラッグする。
- ［Select Request File］をクリックして、ファイル システムから要求ファイルを選択する。

4. ［Find a Match］をクリックします。

   ［検索結果］テーブルに、見つかった一致がすべて表示されます。

5. 元の要求を表示するには、［View Request］をクリックします。
6. ［検索結果］テーブル内のエントリをクリックします。

   一致するコンポーネントが表示されます。

仮想サービス URL

DevTest ポータルで仮想サービスを開くと、その URL は、表示される仮想サービス コンポーネントへのパスを指定します。

以下の例では、［ステートレス トランザクション］ビューにおける、特定のトランザクションの URL 形式を示しています。

```
http://localhost:1507/devtest/#/main/serviceimageeditor/0/bmS18CcJu-b74PB4tcz/bmS18CcJu/b74PB4tcz/kioskV6/0?signatureId=153&specificId=156&view=stateless
```

指定された URL をブックマークすることができます。後でブラウザに URL を入力することにより、同じコンポーネントに仮想サービスを開くことができます。

仮想サービスの展開

DevTest ポータルから仮想サービスを展開できます。

次の手順に従ってください:

1. DevTest ポータルのホーム ページに移動します。
2. 左側のナビゲーション ペインで、［Manage］をクリックします。
3. （オプション）別のプロジェクトを選択します。
4. 展開する仮想サービスを右クリックし、［展開］を選択します。

   ［仮想サービスの展開］ダイアログ ボックスが表示されます。
5. 展開先の VSE サーバを選択します。
6. 必要に応じてフィールドを変更します。

   グループ タグ

   この仮想サービスの仮想サービス グループ (P. 468)の名前を指定します。 展開した仮想サービスにグループ タグがある場合、それらのタグがドロップダウン リストに表示されます。グループ タグは英数字で始まる必要があり、英数字と以下の特殊文字を含めることができます。

   - ピリオド（.）
   - ダッシュ（-）
   - アンダースコア（_）
   - ドル記号（$）

   同時実行数

   負荷容量を示す数値を指定します。 *容量*は、VSM で同時に何人の仮想ユーザ（インスタンス）が実行できるかを示します。 ここでの容量は、このサービス モデルに対する要求を処理するスレッドの数を表します。

   VSE は、同時実行数の合計と同数のスレッドを割り当てます。 休止している場合でも、各スレッドはシステム リソースを多少消費します。 したがって、システム全体のパフォーマンスを最適化するため、この設定は可能な限り小さな値にします。必要なパフォーマンスを達成するか、または設定値を大きくしてもさらなるパフォーマンス向上が見られなくなるまで設定を調整することにより、適切な設定値を経験的に決定します。

標準装備のプロトコルは、スレッド使用を最小化するため、フレームワーク レベル タスク実行サービスを使用します。このようなプロトコルの場合、VSM が高度にカスタマイズされていない限り、同時実行数が 1 コア当たり 2 ～ 3 を超えていても、それが役立つことはほとんどありません。

拡張、および標準装備のプロトコルを使用しない VSM では、反応時間を長く設定すると、その期間中にスレッドを消費する可能性があります。この場合、同時実行数を大きくすることをお勧めします。

このような場合のおおよその初期設定値は、以下の数式で与えられます。

同時実行数 = （必要な 1 秒あたりのトランザクション数/1000） * 平均反応時間（ミリ秒） * （反応時間スケール/100）

例：

反応時間を処理するためにフレームワーク タスク実行サービスを使用しない、カスタム プロトコルを使用していると仮定します。全体的なスループットとして必要な、1 秒あたりのトランザクション数は 100 です。 サービス イメージ全体にわたっての平均反応時間は、200 ミリ秒です。また、仮想サービスは、反応時間スケール 100 （%）で展開されます。

（1 秒あたりのトランザクション数 100/1000） * 200 ミリ秒 * （100/100） = 20

この場合、各スレッドは応答前に平均約 200 ミリ秒ブロックされ、その間は新しい要求を処理できません。そのため、100 トランザクション/秒に対応するには、同時実行数 20 が必要です。スレッドは平均で 10 ミリ秒ごとに利用可能になり、100 トランザクション/秒を実現するのに十分となります。

デフォルト： 1

反応時間スケール

記録された反応時間に対する反応時間の割合（%）を指定します。

注: テストの実行でのペースの整合性を保つため、ステップ自体の処理時間は反応時間から引かれます。

デフォルト： 100

例：

- 反応時間を 2 倍にするには、「200」と入力します。
- 反応時間を半分にするには、「50」と入力します。

7. ［展開］をクリックします。

注: 仮想サービスのステータスには、以下のものがあります。

展開

入力した名前のサービスはまだ展開されていません。 サービスは展開されています。

再展開

入力した名前のサービスは、入力したサービスと同じ .vsm ファイルで展開されています。 サービスは再展開されています。

オーバーライド

入力した名前のサービスは、入力したサービスに関連付けられたものとは異なる .vsm ファイルで展開されています。 展開されているサービスのオーバーライドを促すメッセージが表示されます。

第 7 章: CA Service Virtualization でのワークステーションおよびコンソールの使用

このセクションには、以下のトピックが含まれています。

第 8 章: サービス イメージの作成

このセクションには、以下のトピックが含まれています。

サービスを開く

仮想サービス イメージ レコーダはサービス イメージを生成します。サービス イメージは、ユーザが記録したもの（記録された RAW トラフィックの操作または変更されたバージョン）のように振る舞います。

注: LISA 6.0 より前の VSE リリースのサービス イメージが存在する場合は、それらのサービス イメージをエクスポートします。 エクスポートを行うことにより、それらが以前のバージョンのデータベースから、現在のリリースで格納されるファイル システムに移動します。 詳細については、「レガシー サービス イメージ (P. 313)」を参照してください。

サービス イメージを開くと、サービス イメージ エディタでイメージを変更できます。

次の手順に従ってください:

1. プロジェクト パネルでサービス イメージを右クリックし、［開く］を選択します。

   サービス イメージ エディタが開きます。

2. 選択したサービス イメージを確認し、必要な変更を行います。

サービス イメージの組み合わせ

既存のサービス イメージにより多くの機能を追加する場合、サービス イメージを組み合わせると役立ちます。

次の手順に従ってください:

1. プロジェクト パネルでサービス イメージを右クリックし、[組み合わせ] を選択します。

   [サービス イメージの組み合わせ] ダイアログ ボックスが表示されます。

2. 元のイメージ（ターゲット）に組み合わせる 1 つ以上のサービス イメージを選択します。

3. 一致するターゲット データにあるサービス イメージを置換するには、[ソース イメージを優先] チェック ボックスをオンにします。

   サービス イメージを組み合わせると、各ソース サービス イメージのステートレス トランザクション（メタ レベル）とターゲット サービス イメージのトランザクションが比較されます。一致するものについては、ソースの特定のトランザクションがターゲットのトランザクションと比較されます。一致が見つからない場合は、ソース トランザクションがターゲット イメージに追加されます。一致する場合、トランザクションをマージする必要があります。

   ソース データ（応答ボディなど）を、一致するターゲット トランザクションのものと置換するよう選択できます。また、ターゲット データをそのまま残すことができます。

注: 会話を組み合わせるプロセスも同様です。 各ソース サービス イメージの各会話については、そのスタータ トランザクションがターゲットの会話の各スタータと比較されます。スタータが一致しない場合、ターゲット イメージに新しい会話が作成されます。一致する場合、ステートレス リストの同じプロセスが、ソースの会話ツリーの各メタ ノードに適用されます。 このプロセスは、新しいトランザクションを追加し、必要に応じて一致するトランザクションをマージします。

サービス イメージの削除

次の手順に従ってください:

1. プロジェクト パネルでサービス イメージを右クリックし、［削除］を選択します。

   確認ダイアログ ボックスで、選択したサービス イメージを削除するかどうかの確認を求められます。

2. ［OK］をクリックします。

   選択したサービス イメージが、プロジェクトから永久に削除されます。

   注: サービス イメージは一時的なものと考えて、未使用または古いサービス イメージを削除することをお勧めします。

サービス イメージの作成

次の手順に従ってください:

1. プロジェクト パネルで、［VirtualServices］-［Images］を右クリックします。
2. ［新規 VS イメージの作成］を選択します。

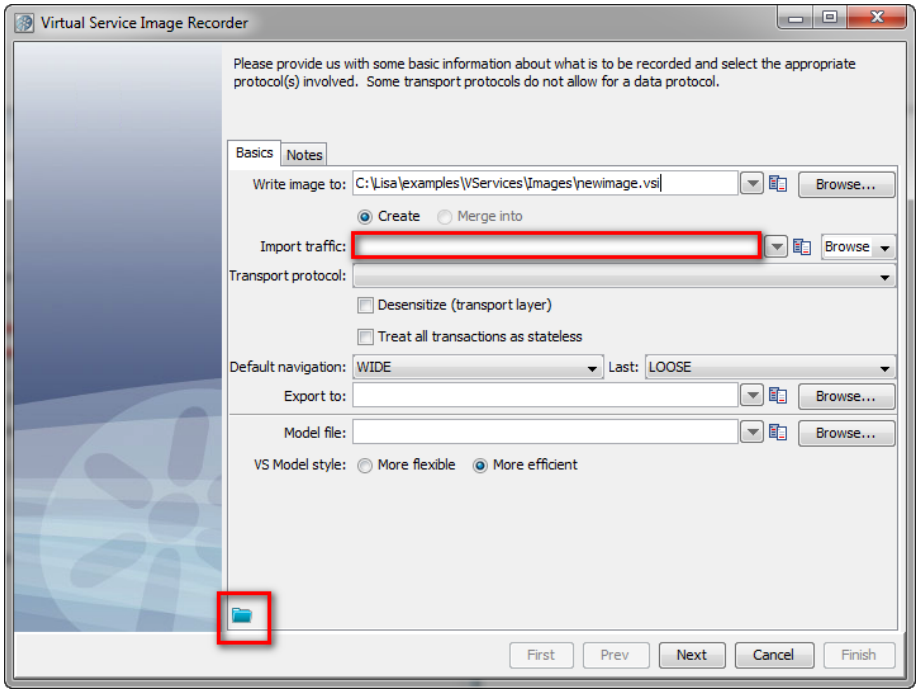
RAW トラフィックのインポート

RAW トラフィックは、VSE サービス イメージ レコーダによってのみインポートできます。

次の手順に従ってください:

1. サービス イメージを作成し、［仮想サービス イメージ レコーダ］を選択します。
2. RAW トラフィック ファイル名を入力するか、またはファイル ブラウザから選択します。
3. 以前に保存されたレコーディング セッションからサービス イメージのパラメータをロードするには、パネルの下部の隅の青いフォルダをクリックします。

   パラメータのロードに使用する .vrs ファイルを参照するよう促されます。



4. 通常のレコーディング プロセスを続行するには、トランスポート プロトコルを選択します。

   レコーディング プロセスを開始すると、VSE は、新しいサービス イメージに RAW トラフィックをインポートします。

詳細:

サービス イメージの新規作成 (P. 122)
WSDL からのサービス イメージの作成 (P. 122)
要求/応答ペアからのサービス イメージの作成 (P. 131)
PCAP からのサービス イメージの作成 (P. 136)
VSM の使用 (P. 241)
データ プロトコルの使用 (P. 243)

サービス イメージの新規作成

次の手順に従ってください:

1. プロジェクト パネルで［VirtualServices］を右クリックし、［新規 VS イメージの作成］-［新規に作成］を選択します。
2. 識別情報とプロトコル情報を入力し、［OK］をクリックします。

   ［サービス イメージ エディタ］ウィンドウが表示されます。
3. 新しいサービス イメージの固有のパラメータと情報を入力します。

WSDL からのサービス イメージの作成

以下の方法で、WSDL から仮想 Web サービス イメージを生成できます。

- ［クイック スタート］メニューから（P. 123）
- ［新規 VS イメージの作成］オプションから（P. 124）

[クイック スタート]メニューから

この手順では、UI を使用して、WSDL から仮想サービス イメージを作成する方法について説明します。

次の手順に従ってください:

1. ［クイック スタート］メニューから［WSDL を使用した SI の作成］を選択します。

2. ［接続］タブで、WSDL の名前を入力します。

   ［ユーティリティ］ をクリックして、お使いのシステムで WSDL を検索します。

   WSDL 名を入力した後、［サービス］および［ポート］フィールドに入力します。［操作］タブに、関連する操作がリスト表示されています。［すべて］または［なし］を選択するか、チェック ボックスを使用して、テストする特定の操作を選択します。

3. ウィンドウの下部の[保存先]フィールドに、作成するサービス イメージの名前を入力します。

   注: デフォルトのサービス イメージ名がすでに使用されている場合は、警告アイコン が表示されます。

4. ウィンドウの下部の緑の矢印 をクリックします。

   サービス イメージ エディタが開き、サービス イメージが表示されます。

［新規 VS イメージの作成］から

この手順では、新規 VS イメージ オプションを使用して、WSDL から仮想サービス イメージを作成する方法について説明します。

次の手順に従ってください:

1. プロジェクト パネルで［VirtualServices］-［Images］を右クリックし、［新規 VS イメージの作成］-［WSDL から作成］を選択します。

   ［WSDL からの仮想サービス］ウィンドウが表示されます。

2. サービス イメージ名および VS モデル ファイルの名前を入力します。
3. このウィンドウの残りのフィールドについては、デフォルト値を使用します。

   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］ をクリックします。

4. ［次へ］をクリックします。

   ［接続］タブが表示されます。

5. ウィンドウの下部の［リスン ポート］フィールドに、仮想サービスがリスンするポート番号を入力します。
6. ［WSDL URL］フィールドに、仮想化する WSDL を追加します。

   このエントリは、ローカル ファイルまたは URL にすることができます。

7. ［サービス］フィールドで、仮想化する必要がある WSDL 内のサービスを選択します。

   通常、1 つのみ選択できます。

8. 仮想化するサービス内の操作を選択します。

   デフォルトでは、すべての操作が選択されます。

9. ［次へ］をクリックします。

   要求/応答側データ プロトコル オプションが開きます。

10. ［**Web サービス（SOAP）**］を選択します。

    ［要求側データ プロトコル］リストは、［Web サービス（SOAP）］および［XML］データ プロトコル ハンドラであらかじめ入力されています。［応答側データ プロトコル］リストは、［区切りテキスト データ プロトコル］で自動的に入力されます。

11. ［次へ］をクリックします。
12. 区切りテキスト データ プロトコルを設定する方法の詳細については、「*CA Service Virtualization の使用*」の「区切りテキスト データ プロトコル (P. 271)」を参照してください。区切りテキスト データ プロトコルを設定した場合、［次へ］をクリックします。

    次のウィンドウで、サービス イメージが生成され、ウィザードが完了します。
13. ［終了］をクリックします。

    注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

    この手順で、空の仮想サービス モデルが生成されます。
14. 仮想サービス モデルを保存します。

    生成されたサービス イメージはスタブ サービスです。 このサービス イメージは正しく整形された応答を返しますが、値はデフォルト値です。

    保存されたサービス モデル（VSM）は、仮想サービス環境に展開されるモデルです。

WADL からのサービス イメージの作成

この手順では、WADL（Web Application Description Language）から仮想サービス イメージを作成する方法について説明します。

次の手順に従ってください:

1. 以下のいずれかのオプションを実行します。
   - プロジェクト パネルで［VirtualServices］を右クリックし、［新規 VS イメージの作成］-［From WADL］を選択します。
   - ［ファイル］メニューから［新規］-［VS イメージ］-［From WADL］を選択します。

   ［Virtual Service From WADL］ウィンドウが表示されます。
2. サービス イメージ名および VS モデル ファイルの名前を入力します。

   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］ をクリックします。

   フィールド説明の詳細については、「［基本］タブ (P. 140)」を参照してください。
3. ［次へ］をクリックします。
4. ウィンドウの下部にある［リスン ポート］フィールドに、仮想サービスがリスンするポート番号を入力します。
5. ［WADL URL］フィールドに、仮想化する WADL を追加します。

   このエントリは、ファイル システム上の WADL ファイルまたは URL にすることができます。
6. ［Refresh WADL Cache］ をクリックします。
7. ［エンドポイント］フィールドで、仮想化する必要がある、WADL 内のエンドポイントを選択します。

   注: 通常、1 つのみを選択できます。
8. ［メソッド］ペインで、仮想化する、エンドポイント内のメソッドを選択します。

   デフォルトでは、すべてのメソッドが選択されます。 必要に応じて、［すべて選択］または［Select None］をクリックできます。
9. ［次へ］をクリックします。

10. 必要に応じて、その他のデータ プロトコル ハンドラを追加するか、またはチェーンします。 デフォルトでは、REST データ プロトコル ハンドラが選択されます。
11. ［次へ］をクリックします。

    次のウィンドウで、サービス イメージが生成され、ウィザードが完了します。
12. ［終了］をクリックします。

    このレコーディングに関する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］ をクリックします。

注: パラメータ lisa.vse.rest.max.optionalqueryparams は、WADL ファイルにおいて 1 つのメソッドで処理する、オプションのクエリ パラメータの最大数を指定します。デフォルトは 5 です。6 番目以降のオプション パラメータは、すべて無視されます。値を 6 以上に変更しないことをお勧めします。値を 6 以上に変更すると、6 番目以降のオプション パラメータによって生成される応答の数が、急激に増加する可能性があります。

RAML からのサービス イメージの作成

この手順では、RAML（RESTful API Modeling Language）から仮想サービス イメージを作成する方法について説明します。

RAML では、Schema プロパティと Example プロパティの組み合わせを使用して、メッセージ ボディを定義できます。Example プロパティは、DevTest のトランザクション ボディに使用されます。 メッセージ ボディには、Example プロパティを必ず指定します。指定しない場合、Schema プロパティが代わりに使用されます。

重要: Schema プロパティは、まったく解釈されず、DevTest 内のトランザクション ボディに指定されたとおりに表示されます。

次の手順に従ってください:

1. 以下のいずれかのオプションを実行します。
   - プロジェクト パネルで［VirtualServices］を右クリックし、［新規 VS イメージの作成］-［From RADL］を選択します。
   - ［ファイル］メニューから［新規］-［VS イメージ］-［From RADL］を選択します。

   ［Virtual Service From RAML］ウィンドウが表示されます。
2. サービス イメージ名および VS モデル ファイルの名前を入力します。

   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］ をクリックします。

   フィールド説明の詳細については、「［基本］タブ (P. 140)」を参照してください。
3. ［次へ］をクリックします。
4. ウィンドウの下部にある［リスン ポート］フィールドに、仮想サービスがリスンするポート番号を入力します。
5. ［RAML URL］フィールドに、仮想化する Web サービスの RAML を追加します。

   ドロップダウン リストから RAML を選択するか、または［参照］ボタンをクリックしてファイル システムから RAML を見つけることもできます。

このエントリは、ファイル システム上の RADL ファイルまたは URL にすることができます。

6. ［Refresh RAML Cache］ をクリックします。

   DevTest で RAML が解析され、［エンドポイント］フィールドおよび［メソッド］ペインにデータが入力されます。

   注: DevTest で RAML の解析に失敗すると、警告アイコンが ［エンドポイント］フィールドの後に表示されます。エラー メッセージを表示するには、警告アイコンをクリックします。

7. ［メソッド］ペインで、仮想化するメソッドを選択します。

   デフォルトでは、すべてのメソッドが選択されます。 必要に応じて、［すべて選択］または［Select None］をクリックできます。 少なくとも 1 つのメソッドが必要です。

8. ［次へ］をクリックします。

9. 必要に応じて、その他のデータ プロトコル ハンドラを追加するか、またはチェーンします。 デフォルトでは、REST データ プロトコル ハンドラが選択されます。

10. ［次へ］をクリックします。

    次のウィンドウで、サービス イメージが生成され、ウィザードが完了します。

11. ［終了］をクリックします。

    このレコーディングに関する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］ をクリックします。

注: パラメータ lisa.vse.rest.max.optionalqueryparams は、RAML ファイルにおいて 1 つのメソッドで処理する、オプションのクエリ パラメータの最大数を指定します。デフォルトは 5 です。6 番目以降のオプション パラメータは、すべて無視されます。値を 6 以上に変更しないことをお勧めします。値を 6 以上に変更すると、6 番目以降のオプション パラメータによって生成される応答の数が、急激に増加する可能性があります。

Layer 7 からのサービス イメージの作成

［Layer7 からの仮想サービス］では、Layer 7 から仮想サービス イメージを生成します。

前提条件

1. Layer 7 サポート Web サイトから Layer 7 Command-line Migration Tool 2.2 をダウンロードします。
2. Jar ファイルを解凍します。

   注: この Jar ファイルには、手順 3 で必要な cmt2.jar ファイルが含まれています。 このファイルは、任意の場所に配置できます。

次の手順に従ってください:

1. 以下のいずれかの操作を実行します。
   - ［ファイル］-［新規］-［VS イメージ］-［Layer7 から作成］を選択します。
   - プロジェクトのルート ノードを右クリックし、［新規 VS イメージの作成］-［Layer7 から作成］を選択します。
   - Virtual Services Images フォルダを右クリックし、［新規 VS イメージの作成］-［Layer7］を選択します。

   ［Layer7 からの仮想サービス］ウィンドウが表示されます。
2. ［Layer7 接続情報］フィールドに入力します。
3. フォルダ アイコンをクリックして、cmt2.jar ファイルを見つけ、［Layer7 サービスを取得］をクリックします。

   ［Layer7 サービス］タブに使用可能なサービスが表示されます。
4. サービスを選択して、［次へ］をクリックします。
5. サービス イメージ名および VS モデル ファイルの名前を入力します。

   このウィンドウの残りのフィールドについては、デフォルト値を使用します。
   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］アイコンをクリックします。
6. ［次へ］をクリックします。

   ［接続］タブが表示されます。
7. 仮想化するサービス内の操作を選択します。

デフォルトでは、すべての操作が選択されます。

8. ［次へ］をクリックします。

   要求/応答側データ プロトコル オプションが開きます。

9. ［**Web サービス（SOAP）**］を選択します。

   ［要求側データ プロトコル］リストは、［Web サービス（SOAP）］および［XML データ プロトコル］データ プロトコル ハンドラであらかじめ入力されています。［応答側データ プロトコル］リストは、［区切りテキスト データ プロトコル］で自動的に入力されます。

10. ［次へ］をクリックします。

11. 区切りテキスト データ プロトコルを設定する方法の詳細については、「区切りテキスト データ プロトコル（P. 271）」を参照してください。設定したら、［次へ］をクリックします。

    次のウィンドウで、サービス イメージが生成され、ウィザードが完了します。

12. ［終了］をクリックします。

    注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

    この手順で、空の仮想サービス モデルが生成されます。

13. 仮想サービス モデルを保存します。

    生成されたサービス イメージはスタブ サービスです。 このサービス イメージは正しく整形された応答を返しますが、値はデフォルト値です。

    保存されたサービス モデル（VSM）は、仮想サービス環境に展開されるモデルです。

要求/応答ペアからのサービス イメージの作成

以下の方法で、要求/応答ペアから仮想サービス イメージを生成できます。

- UI から（P. 132）
- コマンドラインから（P. 134）

要求/応答ペアからのメタデータをカスタマイズするためにサイドカー (P. 135) ファイルを使用できます。

UI での要求/応答ペアからのサービス イメージの作成

この手順では、UI を使用して、要求/応答ペアから仮想サービス イメージを作成する方法について説明します。

次の手順に従ってください:

1. ［VirtualServices］-［Images］を右クリックし、［新規 VS イメージの作成］-［要求/応答ペアから作成］を選択します。

   ［要求/応答ペアからの仮想サービス］ページが表示されます。

2. サービス イメージ名および VS モデル ファイルの名前を入力します。
3. このウィンドウの残りのフィールドについては、デフォルト値を使用し、［次へ］をクリックします。

   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］ をクリックします。

4. ファイル システムで、要求/応答ペアが含まれるディレクトリを参照します。

   要求/応答ペアに一意の識別子で名前を付け、次に要求側に「-req」、応答側に「-rsp」を追加し、**.xml** または **.txt** 拡張子を付けます。 たとえば、**addUserObject-req.xml** および **addUserObject-rsp.xml** などです。

   注: 単一の要求に複数の応答が存在する場合があります。 上記の例を使用すると、1 つの要求と 3 つの応答は以下のようなファイル名になります。

   - addUserObject-req.xml
   - addUserObject-rsp1.xml
   - addUserObject-rsp2.xml
   - addUserObject-rsp3.xml

   VSE は、指定したディレクトリに各要求/応答ペアのトランザクションを生成します。 HTTP/S の場合は、ファイルに SOAP エンベロープおよびヘッダ全体が含まれる必要があります。

5. 次の情報を指定します。

   - トランスポート プロトコル
   - 適切なエンコーディング
   - バイナリ要求/応答ペアを使用するかどうか

6. ［設定］をクリックします。

   選択したトランスポート プロトコル用の設定ウィンドウが表示されます。

7. 要求に関する設定情報を入力し、［終了］をクリックします。

   ［要求/応答ペアからの仮想サービス］ウィンドウが開きます。

8. もう一度［設定］をクリックし、入力した値を保持します。

   別のプロトコルを選択すると、そのプロトコルが表示されます。 その設定情報を入力できます。

9. ［次へ］をクリックします。

   ［データ プロトコル］パネルが開きます。このパネルでは要求/応答ペアが分析されており、要求/応答に対する適切なデータ プロトコルがデフォルトに設定されています。

10. 変更を行うか、またはデータ プロトコルを追加して、［次へ］をクリックします。

    トークン識別および会話の仮想サービス要求/応答ペアが表示されます。 要求/応答ペアからは、ステートレス トランザクションのみがサポートされています。

11. ［次へ］をクリックします。

    注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

    VS イメージの処理が完了した後、サービス イメージと仮想サービス モデルを開くことができます。

コマンド ラインでの要求/応答ペアからのサービス イメージの作成

UI を使用して要求/応答ペアからサービス イメージを作成するほかに、ServiceImageManager コマンド ラインを使用してイメージを作成できます。

次の手順に従ってください:

1. UI でサービス イメージを作成します。
2. イメージが作成された後、［終了］ボタンの上の［保存］ をクリックし、［終了］をクリックします。

   設定は、**.vrs** 拡張子を持つファイルに保存されます。
3. LISA_HOME¥bin ディレクトリに移動して、以下のコマンドを入力します。

   ```
   ServiceImageManager -vrs recording-session-file
   vsi-file=vsi-file --vsm_file=vsm-file --record
   ```

   recording-session-file

   上記で作成した **.vrs** ファイルのパスを定義します。

   vsi-file

   作成されるサービス イメージ ファイルを定義します。

   vsm_file

   作成される仮想サービス モデル ファイルを定義します。

要求/応答ペアを持ったサイドカー ファイル

VSE が要求/応答ペアからサービス イメージを作成する場合、システムはサイドカー ファイルという名前の要求/応答ペアを持った別のファイルを使用できます。 サイドカー ファイルは、特定の要求、応答、またはその両方のメタデータに追加する必要のあるキー/値ペアを追加できるプロパティ ファイルです。 すべての要求および応答に対して、一般的ないくつかのファイルを追加できます。

例:

soap という名前のディレクトリに **abc-req.xml** と **abc-rsp.xml** という名前の要求/応答ペア ファイルがあります。 **abc-req-meta.properties**、**abc-rsp-meta.properties** のような命名規則を使用してサイドカー ファイルを追加します。 そのディレクトリ内のすべての要求および応答のメタにプロパティを追加する場合は、ファイル名 **meta-req.properties** と **meta-rsp.properties** を使用します。 トランザクションに固有のサイドカー ファイル（**abc-req-meta.properties** および **abc-rsp-meta.properties**）のエントリは、グローバル サイドカー ファイル（**meta-req.properties** および **meta-rsp.properties**）にあるエントリをオーバーライドします。そのため、グローバル ファイルにすべてのデフォルトのセットを設定し、トランザクションに固有のサイドカーを使用して特定のトランザクションのデフォルトをオーバーライドできます。

たとえば、3 つの要求/応答（**abc1-req.xml/abc1-rsp.xml**、**abc2-req.xml/abc2-rsp.xml**、および **abc3-req.xml/abc3-rsp.xml**）がある場合、以下を持つ共通のサイドカー ファイル **rsp-meta.properties** を使用できます。

```
Content-type=text/plain
```

また、以下を持つ abc3 固有のサイドカー ファイル **abc3-rsp-meta.properties** を使用できます。

```
Content-type=text/html
```

サービス イメージで、応答 **abc1** および **abc2** のメタデータには、「text/plain」に設定された Content-type メタ プロパティがありますが、応答 **abc3** の値は「text/html」です。

PCAP からのサービス イメージの作成

Wireshark などのパケット キャプチャ ソフトウェアを使用してトラフィックのログを作成する場合、VSE は、それらのログを使用してパケット キャプチャ ファイル（PCAP）から仮想サービス イメージを作成できます。

前提条件

1. http://jnetpcap.com/download から、お使いのオペレーティング システム用の適切なバイナリ パッケージをダウンロードします。
2. バイナリ パッケージから **jnetpcap.jar** を **LISA_HOME¥lib** ディレクトリに追加し、jnetpcap ネイティブ ライブラリを **LISA_HOME¥lib** ディレクトリに追加します。

   ネイティブ ライブラリは、オペレーティング システムによって異なります。 Windows の場合は、jnetpcap.dll です。

   これで、PCAP 機能が設定されます。
3. DevTest ワークステーション を再起動して（実行中の場合）、設定変更を反映します。

次の手順に従ってください:

1. ［VirtualServices］-［Images］フォルダを右クリックし、［新規 VS イメージの作成］-［PCAP から］を選択します。

   サービス イメージ名および仮想サービス モデル ファイルの名前を入力します。

   このウィンドウの残りのフィールドについては、デフォルト値を使用します。

   注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］ をクリックします。
2. （オプション）この仮想サービスに関するドキュメントを追加するには、［メモ］タブをクリックします。
3. ［次へ］をクリックします。

   データ プロトコル ウィンドウが開きます。
4. ［次へ］をクリックします。

5. 入力に使用するパケット キャプチャ ファイルを選択するには、ファイル名を入力するか、またはファイル システムを参照します。
6. トランスポート プロトコルとして HTTP/S を選択し、［設定］をクリックします。

   ［PCAP トランスポート プロトコル設定からの仮想サービス］ウィンドウが開きます。
7. 以下の設定オプションを入力します。

   リスン/記録ポート

   クライアントが DevTest と通信するポートを定義します。

   ターゲット ホスト

   サーバが実行されているターゲット ホストの名前または IP アドレスを定義します。

   ターゲット ポート

   サーバがリスンするターゲット ポート番号を定義します。 プロキシ パススルー スタイルを選択する場合は、このフィールドを空白のままにします。

   デフォルト： 80 （HTTP）および 443 （HTTPS）

   注:［ターゲット ホスト］と［ターゲット ポート］は重要です。これらは、DevTest がパケットを照合する方法を決定します。［ゲートウェイ］を選択し、仮想化するサービスをホストしているサーバの IP アドレスとポートを入力します。PCAP キャプチャがどのように実行されたかによって、キャプチャが実行されたサブネットのすべてのコンピュータに対するネットワーク トラフィックが発生する場合があります。 IP アドレスとポートを指定すると、特定の IP およびポートとの間で送受信されるすべてのパケットに関して、データをファイルからフィルタできます。 その後、それらのパケットが再構築されて有効な TCP ストリームが形成されます。そこでは、重複が排除され、パケットが正しい順序に並べ直されるなどの処理が行われます。 この作業は、実際のレコーディング中に、オペレーティング システムの TCP スタックによってすべて実行されます。 これらのストリームは、実際のプロトコル（HTTP）へと再生されます。また、プロトコルに関する限り、データは有効な TCP ストリームを介して到達します。

   レコーダ転送方式

   レコーディング中に仮想サービス イメージ レコーダが動作する方法を指定します。 ［**ゲートウェイ**］を選択します。

サーバに SSL を使用

DevTest がサーバに要求を送信するために HTTPS を使用するかどうかを指定します。

- **オン：** DevTest は HTTPS（セキュア レイヤ）要求をサーバに送信します。

  ［サーバに SSL を使用］を選択した場合で、［クライアントに SSL を使用］を選択しない場合には、DevTest はレコーディングに HTTP 接続を使用します。 その後、DevTest は HTTPS を使用して、サーバにそれらの要求を送信します。

- **オフ：** DevTest は HTTP 要求をサーバに送信します。

クライアントに SSL を使用

クライアントからの SSL 要求を再生するためにカスタム キーストアを使用するかどうかを指定します。 このオプションは、［サーバに SSL を使用］が選択されている場合のみ有効です。

値

- **オン：** カスタム クライアント キーストアおよびパスフレーズを指定できます。これらのパラメータを入力した場合、ハードコードされたデフォルトの代わりに使用されます。
- **オフ：** カスタム クライアント キーストアおよびパスフレーズを指定できません。

SSL キーストア ファイル

キーストア ファイルの名前を指定します。

キーストア パスワード

指定したキーストア ファイルに関連付けられたパスワードを指定します。

注: 双方向 SSL 環境で VSE を設定する方法の詳細については、「双方向 SSL 接続の仮想化 (P. 155)」を参照してください。

特定トランザクションの重複を許可する（NTLM で推奨）

重複した特定のトランザクションを記録するかどうかを指定します。

8. ［終了］をクリックして、前のウィンドウに戻ります。
9. ［次へ］をクリックして、レコーディングを開始します。

レコーディングによるサービス イメージの作成

仮想サービス イメージ レコーダは、サービス イメージを生成します。サービス イメージは、ユーザによって記録されたように振る舞います。

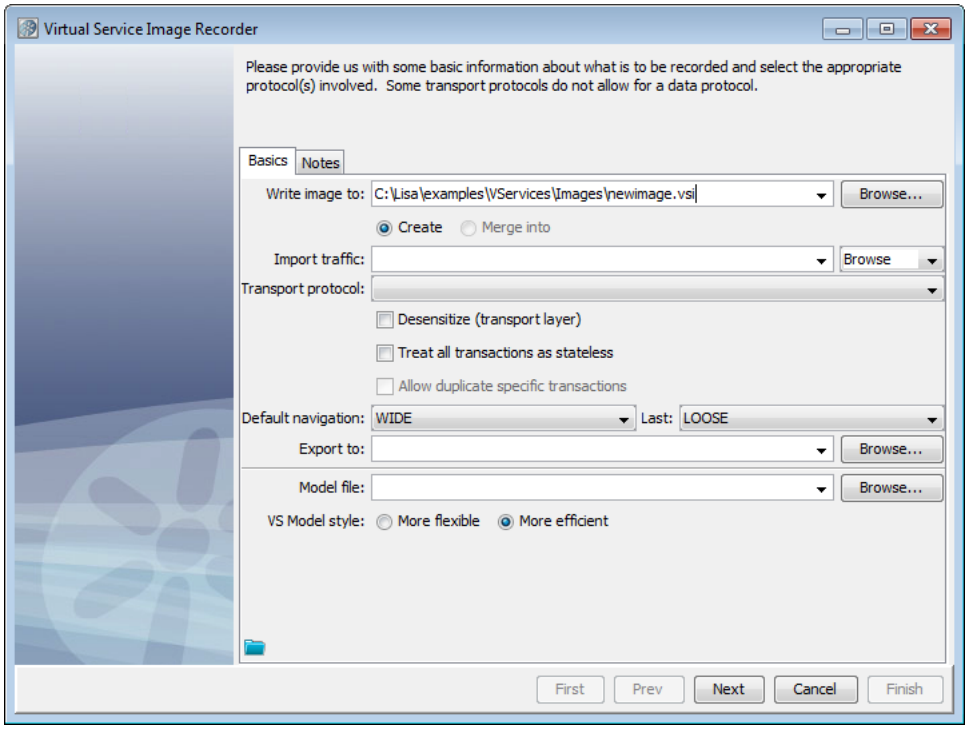
次の手順に従ってください:

1. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］ -［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 レコーダが開きます。
2. 必要に応じて、以下の仮想サービス イメージ レコーダの各タブに入力します。
   - 基本 (P. 140)
   - データ プロトコル (P. 144)
   - トランスポート プロトコルによるレコーディング (P. 146)（仮想サービス イメージを記録する各方法の手順について説明しています）

[基本]タブ

仮想サービス イメージ レコーダ ウィザードの最初のウィンドウにある［基本］タブは、イメージの名前、プロトコル、およびナビゲーション オプションを提供します。 使用するプロトコルに応じて、必要なフィールドに情報を入力します。 それ以降のウィンドウは、すべてプロトコルに固有です。 特定のプロトコルの詳細については、「トランスポート プロトコル (P. 146)」を参照してください。



［基本］タブのオプションを以下に示します。

イメージの書き込み先

一意のサービス イメージ名を指定します。

VSI パスは、デフォルトで DevTest ワークステーション を起動したときに開いていたプロジェクトになります。 別のプロジェクトに変更した場合でも、VSI パスはデフォルトで DevTest ワークステーション を起動したときに開いていたプロジェクトになります。

インポート トラフィック

RAW または会話型 XML トラフィック ファイルをインポートします。そのようなファイルが存在しない場合は、このフィールドを空白のままにできます。ファイルを指定すると、参照された XML ドキュメント内のトランザクションは、レコーディングの確認から発生する内容にマージされます。

トランスポート プロトコル

使用するトランスポート プロトコルを指定します。 詳細については、「トランスポート プロトコル (P. 146)」を参照してください。

ディセンシタイズ

レコーディング中に、機密データを認識し、代わりにランダムな値を代入します。 詳細については、「データのディセンシタイズ (P. 409)」を参照してください。

すべてのトランザクションをステートレスとして処理

このオプションは、記録されたトランザクションをすべてステートレスとして処理する特殊な場合に対して提供されています。 ほとんどの場合は、このチェック ボックスをオフにします。

特定トランザクションの重複を許可

DevTest が別の応答を選択して同じコールに 2 回以上応答できるかどうかを指定します。このチェック ボックスをオンにすると、ラウンドロビン一致のみが発生します。 このオプションは、重複した特定のトランザクションを許可しないトランスポート プロトコルに対しては無効です。

デフォルト ナビゲーション

会話ツリーにおいて、特定のトランザクションに続くトランザクションを VSM が検索する場所を特定するナビゲーション許容差を指定します。最後 (リーフ) のトランザクション以外のすべてに対するデフォルトのナビゲーション許容差を選択します。

値

- CLOSE： 現在のトランザクションの子が検索されます。
- WIDE : 現在のトランザクションとその兄弟および姪/甥が含まれる CLOSE 検索。
- LOOSE : 現在のトランザクションの親および兄弟に続いて、開始トランザクションの子が対象となる WIDE 検索。両方で一致が見つからない場合、すべての会話の開始トランザクションが確認され、会話全体が検索されることになります。

デフォルト： WIDE

最後

会話ツリーにおいて、最後（リーフ）のトランザクションに続くトランザクションを VSM が検索する場所を特定するナビゲーション許容差を指定します。

値

- CLOSE： 現在のトランザクションの子が検索されます。
- WIDE: 現在のトランザクションとその兄弟および姪/甥が含まれる CLOSE 検索。
- LOOSE: 現在のトランザクションの親および兄弟に続いて、開始トランザクションの子が対象となる WIDE 検索。両方で一致が見つからない場合、すべての会話の開始トランザクションが確認され、会話全体が検索されることになります。

デフォルト： LOOSE

エクスポート先

RAW トラフィックをログ記録するファイルのフル パスを指定します。指定した場合、トランザクションは、（実際のレコーディング時のトランスポート プロトコルから、またはインポート プロセスから）レコーダに提供されるごとにこのファイルに書き込まれます。 レコーディング セッションは、後のインポート用にキャプチャでき、またデータ プロトコルの詳細な設定に再利用できます。

モデル ファイル

このサービス イメージの仮想サービス モデル ファイルのフル パスを入力します。 このフィールドでファイル名を指定すると、レコーダは自動的に VSM を生成します。 モデル スタイルは、VSM が要求されている場合にのみ有効です。

VS モデル スタイル

準備ステップを含めて VSM を生成するかどうかを指定します。

値

フレキシブル： 準備ステップを含めます（HTTP/S プロトコルの場合は 5 ステップ モデルになります）。

効率重視： 準備ステップを含めません（HTTP/S プロトコルの場合は 3 ステップ モデルになります）。

デフォルト： フレキシブル

注: 以前に保存したサービス イメージからパラメータをロードするには、ウィンドウの下部の［ファイルからロード］  をクリックします。

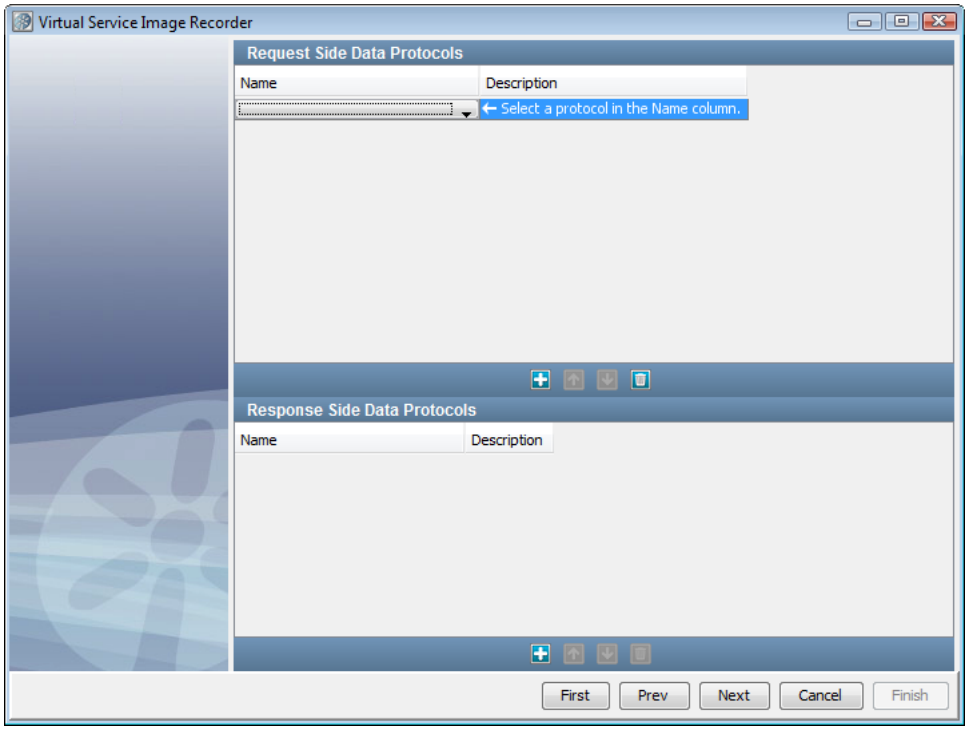
［メモ］タブでは、このサービス イメージの説明を入力できます。

注: VSE レコーダで RAW トラフィック ファイルをインポートする場合、［戻る］ボタンをクリックして最初のパネルに戻ると、VSE レコーダはトラフィック ファイルを再インポートして再びトランザクションを処理するため、2 倍の数のトランザクションが生成されます。

また、このプロセスを実行した場合で、最初にトラフィック ファイルを指定せずにレコーディング パネルに進み、次に最初のパネルに戻ってトラフィック ファイルを選択した場合、再度 VSE レコーダを実行しても、トランザクションは処理されません。

データ プロトコル

仮想サービス イメージ レコーダ ウィザードの 2 番目のウィンドウでは、仮想サービスのデータ プロトコルに関する情報を入力できます。



このレコーダは、以下のデータ プロトコル ハンドラを使用できます。 適切なデータ プロトコルを選択することは、レコーダが記録する情報を分析してそれぞれの会話を識別するために役立ちます。また、これらの会話に属するトランザクションを識別するためにも役立ちます。 これらのデータ プロトコル ハンドラを組み合わせて、互いに使用することができます。

自動ハッシュ トランザクション ディスカバリ

データのハッシュ コードによってメッセージを識別します。データがわずかに違うだけでハッシュ コードは異なるため、すべての要求を効果的に一意にすることができます。 このプロトコルは、サービスに対して同じ少数の要求のセットを実行する場合に役立ちます。

CICS Copybook データ プロトコル

記録された要求をそれぞれのコンテナ チャンクに分割します。 次に、Copybook データ プロトコルに各チャンクを送信し、対応する XML を集約します。

Copybook データ プロトコル

Copybook テキストを XML に変換します。

CTG Copybook データ プロトコル

記録された要求をそれぞれのコンテナ チャンクに分割します。 次に、Copybook データ プロトコルに各チャンクを送信し、対応する XML を集約します。

データ ディセンシタイザ

レコーディング中に機密データを認識し、ランダムな値を代入します。詳細については、「データのディセンシタイズ (P. 409)」を参照してください。

区切りテキスト データ プロトコル

区切られた文字列を XML に変換します。

DRDA データ プロトコル

レコーディング中にバイナリ DRDA ペイロードを XML に変換して、ネイティブ DevTest 機能との整合性、可読性、動的データのサポートを促進します。 応答を再生時にネイティブ形式に再変換します。

EDI X12 データ プロトコル

ANSI X12 EDI ドキュメントを要求のボディ内の XML 表現に変換します。

ジェネリック XML ペイロード パーサ

要求および応答が XML 文字列であるかどうかを識別します。このプロトコルを使用すると、レコーダが使用する XML メッセージ内の変数を識別できます。

JSON データ プロトコル

JSON データを同等の XML に変換し、XML データを JSON 形式に変換します。

要求データ コピー

現在のインバウンド要求から現在のテスト コンテキストにデータをコピーします。

要求データ マネージャ

VSE 要求の記録時または再生時に、それらを操作します。

REST データ プロトコル

REST アーキテクチャ スタイルに従う HTTP 要求を分析します。

スクリプタブル データ プロトコル

要求側、応答側、またはその両方にスクリプトを提供して、要求または応答を処理します。

SWIFT データ プロトコル

SWIFT メッセージを同等の XML に変換し、XML データを SWIFT メッセージに変換します。

Web サービス ブリッジ

サンプルの DevTest Travel にのみ適用されます。 このデータ プロトコルはこのサンプルに固有であり、一般的なケースには役立たないため、無視できます。

Web サービス（SOAP）

Web サービス クライアントによって使用するために適用されます。

Web サービス（SOAP ヘッダ）

SOAP ヘッダ エレメントを要求の引数に変換します。

WS-Security 要求

仮想化フレームワークに従って SOAP 要求を送信する前にセキュリティを解除し、送信する SOAP 応答にセキュリティを適用します。

XML データ プロトコル

XML ドキュメントを適切な要求の操作/引数タイプに変換します。

注: JDBC は、データ プロトコルを許可しません。

これらのデータ プロトコルの詳細については、「データ プロトコルの使用」を参照してください。

動的データ プロトコルの使用方法の詳細については、「ジェネリック XML ペイロード パーサ (P. 279)」を参照してください。

トランスポート プロトコル

使用可能なトランスポート プロトコルについては、それぞれ以下のセクションで説明しています。

HTTPS

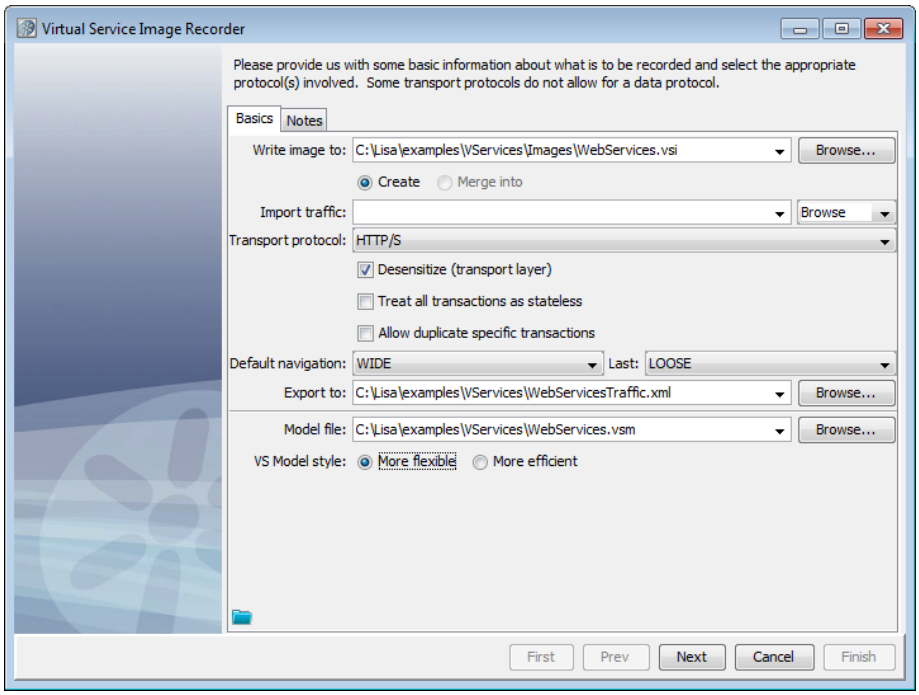
次の手順に従ってください:

1. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。

   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。

2. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



3. ［次へ］をクリックします。

   ウィザードの次のウィンドウが表示されます。

4. この手順では、ポートおよびホストの情報を入力します。

   リスン/記録ポート

   クライアントが DevTest と通信するポートを定義します。 通常、8001 を選択します。ただし、別のポート番号も使用できます。

   ターゲット ホスト

サーバが実行されているターゲット ホストの名前または IP アドレス。プロキシ パススルー スタイルを選択する場合は、空白のままにします。

ターゲット ポート

サーバがリスンするターゲット ポート番号を定義します。プロキシ パススルー スタイルを選択する場合は、このフィールドを空白のままにします。

デフォルト： 80 （HTTP）および 443 （HTTPS）

レコーダ転送方式

レコーディング中に仮想サービス イメージ レコーダが動作する方法を示します。選択肢は［ゲートウェイ］と［プロキシ］です。［プロキシ］を選択すると、［ターゲット ホスト］フィールドと［ターゲット ポート］フィールドのコンテンツがクリアされ、フィールドが無効になります。この選択肢は、クライアントがレコーディング モードで接続する方法に影響します。

- 仮想サービス イメージ レコーダがゲートウェイ モードでリスンする場合、クライアントは、サーバではなくレコーダに HTTP 要求を直接送信する必要があります。クライアントがブラウザの場合、URL には、サーバのホストおよびポートではなく、レコーダのホストおよびポートが含まれます。
- 仮想サービス イメージ レコーダがプロキシ モードでリスンする場合、クライアントは、プロキシとしてレコーダのホストおよびポートを指定する必要があります。クライアントがブラウザの場合、URL にはサーバのホストおよびポートが含まれます。プロキシ設定は、レコーダを介して要求をルーティングするように設定されている必要があります。

ほとんどの HTTP クライアントには、ローカル ホストにプロキシを使用しないようにする設定があります。仮想サービス イメージ レコーダがローカル ホスト上でプロキシ モードで実行されている場合は、トラフィックがレコーダを正しく通過するように、この設定を無効にします。

クライアントから受信したホスト ヘッダ パラメータは変更しないでください

Host パラメータの値をパス スルーするかどうかを指定します。このオプションは、ゲートウェイ モードで記録する場合にのみ使用可能です。パススルー オプションは、ターゲット エンドポイントにトラフィックを再送信する場合に Host ヘッダ パラメータを再書き込みしないようにレコーダに指示します。

サーバに SSL を使用

DevTest がサーバに要求を送信するために HTTPS を使用するかどうかを指定します。

- **オン：** DevTest は HTTPS（セキュア レイヤ）要求をサーバに送信します。

  ［サーバに SSL を使用］を選択した場合で、［クライアントに SSL を使用］を選択しない場合には、DevTest はレコーディングに HTTP 接続を使用します。 その後、DevTest は HTTPS を使用して、サーバにそれらの要求を送信します。

オフ： DevTest は HTTP 要求をサーバに送信します。

クライアントに SSL を使用

クライアントからの SSL 要求を再生するためにカスタム キーストアを使用するかどうかを指定します。 このオプションは、［サーバに SSL を使用］が選択されている場合のみ有効です。

値

- **オン：** カスタム クライアント キーストアおよびパスフレーズを指定できます。これらのパラメータを入力した場合、ハードコードされたデフォルトの代わりに使用されます。

オフ： カスタム クライアント キーストアおよびパスフレーズを指定できません。

SSL キーストア ファイル

キーストア ファイルの名前。

キーストア パスワード

キーストア ファイルのパスワード。

注: 双方向 SSL 環境で VSE を設定する方法の詳細については、「双方向 SSL 接続の仮想化 (P. 155)」を参照してください。

5. ［次へ］をクリックします。

   仮想サービス イメージ レコーダが、トラフィックのレコーディングを開始します。 このウィンドウに、割り当てられたポートおよびサービス ターゲットが表示されます。

6. 仮想サービス イメージ レコーダを介してルーティングされ、トラフィックのレコーディングを開始する要求をサーバに送信するには、HTTP クライアントを使用します。

仮想サービス イメージ レコーダがトランザクションを記録すると、ウィンドウの下部の統計表示が動的に増加します。オプションと動的な統計表示には、以下のものが含まれます。

合計セッション数

記録された会話の数。

合計トランザクション数

記録されたトランザクションの数。

クリア

現在記録されているトランザクションをクリアするには、このボタンをクリックします。

7. レコーディングを完了したら、［次へ］をクリックして次の手順に進みます。

   ［次へ］をクリックした場合で、トランザクションが記録されていない場合には、エラー メッセージが表示されます。［OK］をクリックしてレコーディングを続行します。

   注: トランザクションが記録されない場合、ポートの競合が発生している可能性があります。クライアントは、仮想サービス レコーダの代わりに、アプリケーションにトランザクションを送信します。 別のサービスがそのポートを使用している場合は、そのサービスを停止するか、またはポート設定を変更して競合を解消します。

   ［トランザクション］タブには、記録された最新のトランザクションのリストが表示されます。 このトランザクションのリストで、トランザクションをダブルクリックして、トランザクションのコンテンツを示すダイアログ ボックスを表示できます。

8. ベース パスを確認し、必要に応じて更新します。
9. 要求を処理する前にポートへのバインド ステップを要求するには、［ポートへの個別のバインド **ステップが必要**］チェック ボックスをオンにします。
10. ［次へ］をクリックします。
11. 次のウィンドウではデータ プロトコルにいずれの値も選択せずに、［次へ］をクリックします。

12. レコーディング プロセス中に会話が検出されなかった場合は、会話を開始するトランザクションを選択します。トークン ベースの会話では、トークンを検出する場所を指定します。 仮想サービス イメージ レコーダの［トークン識別］領域を使用します。 会話を開始するトランザクションを選択し、セッション トークンを識別します。会話スタータとしてリストされた **getNewToken** 会話スタータ トランザクションを指定するには、それを選択して青い矢印をクリックします。

    この手順には、以下のコンポーネントが含まれます。

    会話スタータ トランザクション

    会話スタータとして選択したトランザクションをリスト表示します。 トランザクションを［残りのトランザクション］リストに移動する（会話スタータにしない場合）には、トランザクションを選択して矢印をクリックします。

    残りのトランザクション

    記録されたトランザクションをリスト表示します。 トランザクションを［会話スタータ トランザクション］リストに移動するには、トランザクションを選択して矢印をクリックします。

    プラス アイコン

    選択したトランザクションと同様に、リスト内のトランザクション（会話スタータまたは残りのいずれか）をすべて選択します。選択したトランザクションをすべて移動するには、適切な矢印ボタンを使用します。

    会話数

    レコーディングでの会話数を表示します。 会話を作成すると、数が増加します。

    ステートレス強制

    ［残りのトランザクション］リストから、引き続きステートレスにする必要があり、チェック ボックスをオンにしておく必要があるトランザクションを選択します。 たとえば、会話スタータ トークンが含まれている場合であっても、イメージが含まれるトランザクションをステートレスのままにするように指定できます。

    ステートレス トランザクション

    クリックすると、このパネルで識別された会話で、ステートレスのままにするすべてのトランザクションのリストを参照できます。このリストを使用して、すべての会話スタータ トランザクションを識別したことを確認できます。

保存

クリックすると、記録された RAW トランザクションが保存されます。［参照］をクリックして、ファイルを保存する場所に移動します。新しいレコーディングを開始する前に、［基本］タブで RAW トラフィック レコーディングをインポートできます。

応答

現在選択されているトランザクションに対して、このフィールドは、その応答のどれを検索するかを識別します。 通常は、1 が唯一のオプションです。

検索対象

このフィールドは、会話トークンを検索する際に確認する応答を識別します。ドロップダウン リストには、応答の各メタ データ エントリに対するエントリと、応答のボディに対するエントリが 1 つ含まれます。

［トークン識別］領域

選択したトランザクションおよび応答に基づいて、選択した応答の検索対象セクションのコンテンツが表示されます。

- 会話トークンとしてテキストの一部をマークするには、テキストを選択して、赤いスタンプ アイコンをクリックします。 テキストが黄色で強調表示されます。
- 会話トークンではないものとしてテキストをマークするには、テキストの別の部分をマークするか、または消去アイコンをクリックします。
- トークンをマークした後、検索アイコンを使用して同様のトランザクションを検索し、そのトークンをマークすることができます。 会話トークンにバインドするテキスト（XML タグなど）を選択できるダイアログ ボックスを開くには、検索アイコンをクリックします。検索する先頭と末尾のテキストを指定するには、この方法を使用します。

13. ［次へ］をクリックします。

後処理中には、仮想サービス イメージ レコーダに処理ステータスが表示されます。.vsi ファイルを作成するための準備の一部として、レコーダは、要求と応答のボディを検証し、それらがテキストとしてマークされている場合はテキストであることを確認します。 テキストではない場合、タイプがバイナリに切り替わります。

レコーダは、レコーディングの後処理を完了します。

注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

14. ［終了］をクリックして、イメージを格納します。
15. DevTest ワークステーション で仮想サービス モデルを確認および保存します。

双方向 SSL 接続の仮想化

双方向 SSL 接続を仮想化するには、DevTest に以下のいずれかが必要です。

- クライアント キーストアとサーバ キーストアの両方。
- クライアント キーストアと DevTest キーストア（インストール ディレクトリの **webreckeys.ks** を参照）。 DevTest キーストアから DevTest 証明書を抽出し、それをクライアントの信頼ストアに追加します。 DevTest 証明書は自己署名証明書であり、認証局が発行する証明書ではありません。 この回避策は、クライアントが自己署名証明書を受け入れる場合にのみ機能します。

いずれの場合にも、クライアント キーストアとサーバ キーストア（または DevTest キーストア）の 2 つのキーストアが必要です。

local.properties ファイル（インストール ディレクトリにある）で、クライアント キーストアを以下のように使用するように SSL プロパティを設定します。

ssl.client.cert.path

キーストアのパスを定義します。例：

c:/mykeystore.jks。

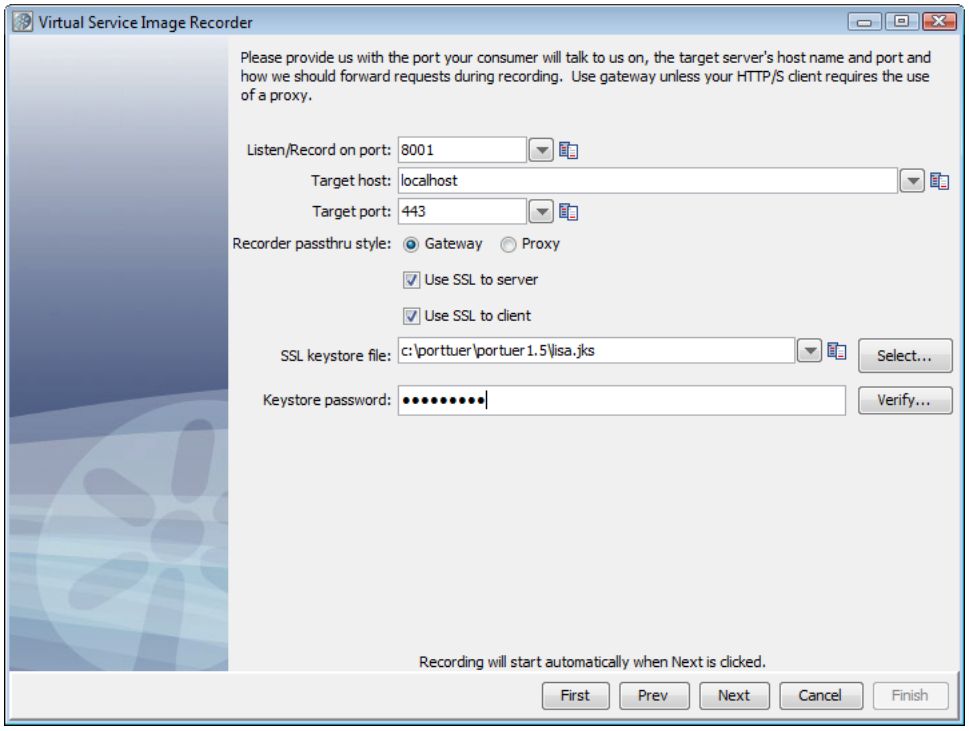
ssl.client.cert.pass

キーストアのパスワードを定義します。

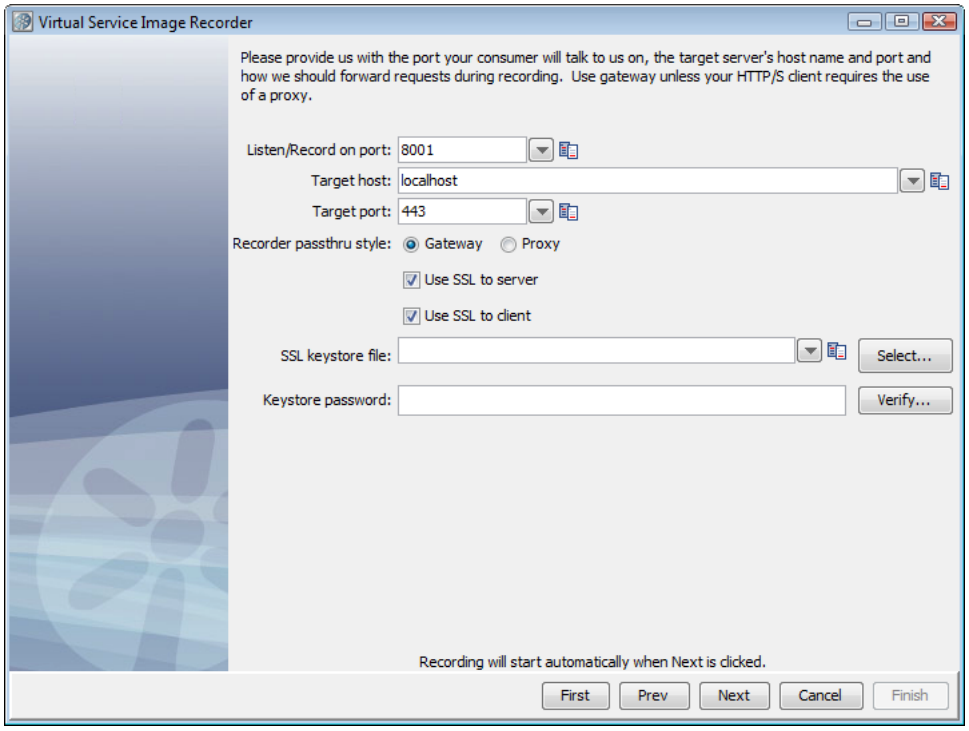
ssl.client.key.pass

証明書のパスワードを定義します。

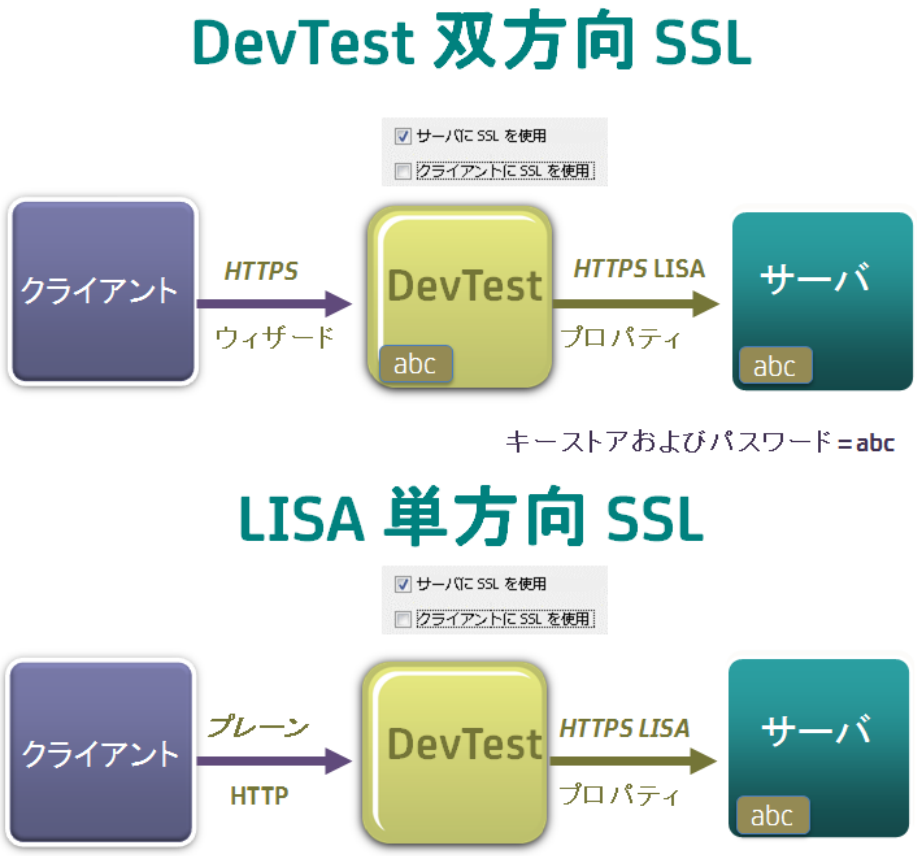
VSE レコーダを開始し、双方向 SSL を使用するように設定します。 クライアントおよびサーバ キーストアを使用する場合、レコーダは以下の図のようになります。



実際のサーバ キーストアの代わりに DevTest キーストアを使用する場合、そのパスを指定する必要はありません。 DevTest キーストアはデフォルトで使用されます。レコーダは、以下の図のように設定する必要があります。



以下の図は、単方向および双方向 SSL の仮想化を示しています。


DevTest 双方向 SSL
サーバに SSL を使用
クライアントに SSL を使用
クライアント
HTTPS
ウィザード
DevTest
abc
HTTPS LISA
プロパティ
サーバ
abc
キーストアおよびパスワード = abc
LISA 単方向 SSL
サーバに SSL を使用
クライアントに SSL を使用
クライアント
プレーン
HTTP
DevTest
HTTPS LISA
プロパティ
サーバ
abc

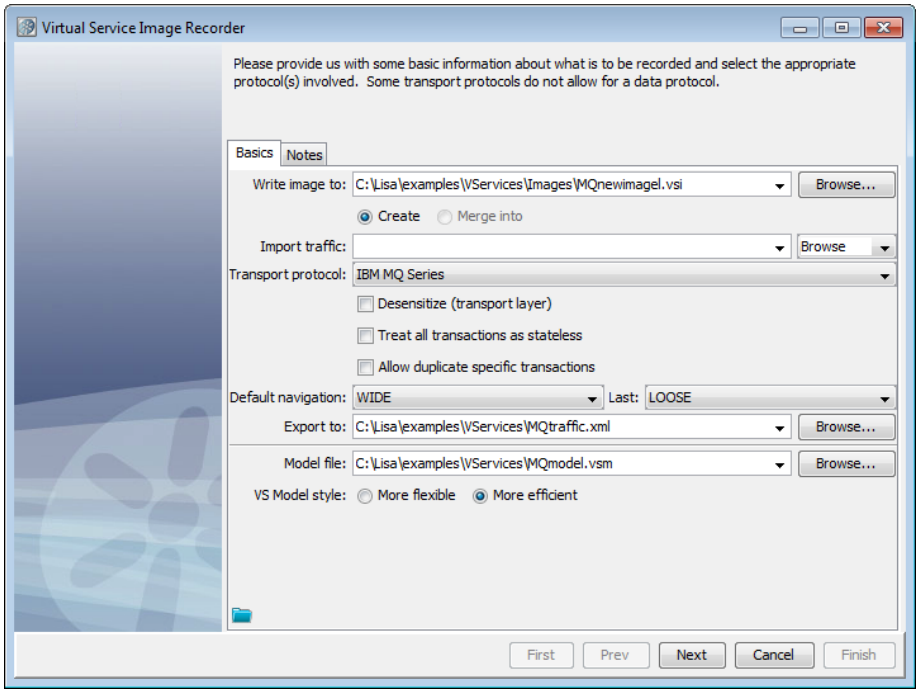
IBM WebSphere MQ

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. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。
2. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



3. ［次へ］をクリックします。

   レコーディング モードの選択手順が表示されます。以下のオプションのいずれかが選択されています。

   プロキシ モード

プロキシ モードは、VSE がサポートする唯一のレコーディング モードです。 プロキシ モードには以下のオプションがあります。

- ライブ キューからのサービスの生成
- プロキシ キューを使用したサービスの生成

プロキシ モードでは、メッセージ バスで設定するプロキシ送信先を使用できます。 クライアント アプリケーションは、これらのプロキシ送信先にメッセージを送信するように設定されます。 DevTest は、それらを記録し、実際の送信先に転送します。 応答の際も同様の動作になります。DevTest は実際の応答先をリスンするように設定され、メッセージを取得すると、応答プロキシの送信先に転送します。 一時送信先を有効にして、応答側を自動化していると、異なった動作になる場合があります。

インポート モード

初期レコーディング ウィンドウの［インポート トラフィック］フィールドで RAW トラフィック ファイルを指定すると、仮想サービスを記録するためのこのモードが使用可能になります。 このレコーディング モードでは、RAW トラフィック ファイルで検出された要求および応答キューに関する追加の情報を設定できます。 また、要求応答ステップ全体をスキップするように選択することもできます。

4. プロキシ モードで記録する場合は、以下のフィールドに入力します。

ライブ/プロキシ キューを使用したサービスの生成

最終的な仮想サービス モデルおよびイメージを生成する場合に、ライブ キューまたはプロキシ キューを使用するかどうか。

最大保留トランザクション数

各応答キューで実行される応答リスナの最大数。 これらの応答リスナは、トランザクションが保留中である限り実行されます。 保留中のトランザクション数がこの最大数を超えると、最も古いトランザクションが自動的に閉じられます。 このフィールドに「0」を入力した場合、最大数は設定されません。

複数の応答を無効化

各トランザクションで複数の応答をサポートするかどうか。 デフォルトでは、トランザクションは最初の応答の後に保留中となり、同じトランザクションに対する追加の応答の受信を待機します。 このオプションを選択すると、トランザクションは最初の応答の後に自動的に閉じられます。 多数のトランザクションを立て続けに受信する場合、このオプションはパフォーマンスを向上させることができます。これは、MQ に対して特に有効です。

相関

個別のトランザクションの要求側と応答側を関連付けるためのスキームが含まれます。 JMS は非同期です。これは、要求と応答が個別に受信されることを意味しています。 このドロップダウン リストでは、VSE レコーダに対して、どの要求をどの応答に関連付けるかを定義できます。 ［相関］フィールドには以下のオプションがあります。

- ［シーケンシャル］: すべての応答は、受信した最終要求に時系列で関連付けられます。相関スキームはありません。そのため、VSE MQ レコーダは、ライブ応答キューに対して排他的読み取りロックを取得し、別の MQ リスナがそのキューから応答メッセージを受信できないようにします。

  ［相関 ID］または［メッセージ ID と相関 ID］などの相関スキームを指定した場合、VSE MQ レコーダはライブ応答キュー上のその他のリスナも相関スキームを使用すると想定します。ライブ応答キュー上のすべてのリスナが相関スキームを使用する場合、VSE MQ レコーダはその応答を排他的読み取りロックを使用せずに個別に受信できるため、共有の入力フラグを使用してキューを開きます。

- 相関 ID： 要求と応答は同じ相関 ID を持つ必要があります。
- メッセージ ID と相関 ID: 要求のメッセージ ID と応答の相関 ID が同じである必要があります。
- メッセージ ID： 要求と応答が同じメッセージ ID を持ちます。

5. インポート モードで記録する場合は、以下のフィールドに入力します。

クライアント モード

WebSphere MQ サーバと通信する方法を指定します。

値

- ネイティブ クライアント: IBM 固有の API を使用する Pure Java 実装

- JMS：JMS 仕様に基づく Pure Java 実装。この実装が必要な場合は、MQ の代わりに JMS トランスポート プロトコルを使用することをお勧めします。
- バインディング：このオプションは、WebSphere MQ クライアント インストールからネイティブ ライブラリへのアクセスを必要とします。DevTest アプリケーション ランタイムがこれらのライブラリにアクセス可能であることを確認します。ほとんどの場合、これらのライブラリを PATH 環境変数で使用可能にしておくことは適切です。

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

このチェック ボックスをオンにすると、要求ステップと応答ステップ全体をスキップできます。 RAW トラフィック ファイルを CAI から受信した場合、このオプションはオフになります。CAI は、キューとトランザクションを自動的に検出します。 その他の場所から RAW トラフィック ファイルを受信した場合、このオプションはデフォルトでオンになり、送信先とトラッキングの設定を確認できます。

相関

個別のトランザクションの要求側と応答側を関連付けるためのスキームが含まれます。 JMS は非同期です。これは、要求と応答が個別に受信されることを意味しています。 このドロップダウン リストでは、VSE レコーダに対して、どの要求をどの応答に関連付けるかを定義できます。 ［相関］フィールドには以下のオプションがあります。

- ［シーケンシャル］：すべての応答は、受信した最終要求に時系列で関連付けられます。相関スキームはありません。そのため、VSE MQ レコーダは、ライブ応答キューに対して排他的読み取りロックを取得し、別の MQ リスナがそのキューから応答メッセージを受信できないようにします。

  ［相関 ID］または［メッセージ ID と相関 ID］などの相関スキームを指定した場合、VSE MQ レコーダはライブ応答キュー上のその他のリスナも相関スキームを使用すると想定します。ライブ応答キュー上のすべてのリスナが相関スキームを使用する場合、VSE MQ レコーダはその応答を排他的読み取りロックを使用せずに個別に受信できるため、共有の入力フラグを使用してキューを開きます。
- 相関 ID： 要求と応答は同じ相関 ID を持つ必要があります。

- メッセージ ID と相関 ID：要求のメッセージ ID と応答の相関 ID が同じである必要があります。
- メッセージ ID： 要求と応答が同じメッセージ ID を持ちます。

6. ［次へ］をクリックします。

   ［送信先情報］タブが表示されます。

7. プロキシおよびライブ キュー名を入力し、キュー タイプを選択します。

   ［プロキシ キューを作成する］チェック ボックスでは、プロキシ キューとして使用される一時キューを適宜作成できます。 WebSphere MQ に手動でプロキシ キューを作成していない場合は、このオプションを選択します。

8. ［接続のセットアップ］タブをクリックします。

9. MOM への接続に使用する接続パラメータを入力します。

   これらの接続パラメータは内部に保存されます。

   ［接続のセットアップ］タブの下部にある［詳細］タブには、以下のサブタブが含まれます。

   環境

   より多くの MQ 環境設定に対して値を追加、削除、および指定できます。

   MQ Exits

   セキュリティ、送信、および受信用の MQ 出口をポイントできます。

10. プロキシ キューがライブ キューとは別のキュー マネージャ上にある場合、オプションの個別の接続情報セットを定義するには、［プロキシ接続のセットアップ］タブをクリックします。

11. 応答の送信先がメッセージをリスンするための接続の詳細を定義します。

12. クライアント アプリケーションがそれらの応答を受信するプロキシ キューを定義します。

    - 単一の要求に対して複数の応答が可能であることに留意してください。応答プロキシ キューのセットを登録するには、［追加］  をクリックします。
    - 不明な要求に対して使用する応答キューを登録するには、［不明な応答に使用する］チェック ボックスをオンにします。

- 一時送信先を定義するには、［一時キュー/トピックを使用する］チェック ボックスをオンにします。 このオプションを選択すると、送信先名およびプロキシ送信先名フィールドが無効になり、追加する応答リスナが一時応答としてマークされます。 送信先名は空白です。

  メッセージングでの「一時」送信先は、メッセージング クライアント用にオンデマンドで作成される送信先です。 一時送信先は、通常、要求/応答シナリオで使用されます。 DevTest は、応答に対して一時キューの使用をサポートしていますが、一時キューでは一度に 1 つの同時トランザクションのみをサポートします。

13. ［次へ］をクリックします。

    ［送信先リスト］タブが開きます。

14. ［現在の接続情報］タブをクリックして、接続情報が正しいことを確認します。

15. ［現在の接続情報］タブに表示される情報は、以前に指定した接続情報からコピーされます。 まれに、応答接続情報が異なる場合には、それを変更できます。

16. ［次へ］をクリックして、レコーディングを開始します。

    VSE がリスンしているキューの名前が表示されます。ステータスは［待機中］です。

    WebSphere MQ に接続する場合で、プロキシ キューの作成を選択した場合、そのプロキシ キューが作成されます。

17. 要求プロキシ キューにメッセージを追加するクライアントを実行します。

    VSE は、実際の要求キューにメッセージをコピーします。 サーバは、そこからそれらのメッセージを取得し、1 つまたは複数の応答キューに応答を送信します。 VSE は、再度それらのメッセージを取得し、クライアントがリスンしている応答プロキシ キューにコピーします。

    トランザクションが記録されるに従ってメッセージ数が増加し、［仮想サービス イメージ レコーダ］ウィンドウの［合計セッション数］と［合計トランザクション数］が増加します。 レコーディングの終了時には、すべての要求が同じ要求キューを経由しています。 応答の約半分は一時キューから返され、別の半分は非一時応答キューを介して返されています。

実行が完了すると、応答キューのメッセージ数が予想よりも 1 つ少ないことがあります。 単一の要求には複数の応答が存在する場合があるため、VSE は最後のトランザクションが完了したことをまだ認識していません。 したがって、最後のトランザクションに対応するメッセージはカウントされません。

18. ［次へ］をクリックすると、最後のトランザクションを閉じて必要なクリーニングを実行するトリガとなります。（動的なデータ プロトコルを使用している場合には、中間ウィンドウが表示される場合があります。）.vsi ファイルを作成するためのレコーダの準備の一部として、レコーダは、要求と応答のボディを検証して、それらがテキストであるか、テキストとしてマークされているかを検証します。 テキストではない場合、タイプがバイナリに切り替わります。

    要求データ マネージャ データ プロトコルが要求側に追加されています。 このプロトコルの設定の詳細については、「要求データ マネージャ (P. 287)」を参照してください。 さらに多くのデータ プロトコルを変更または追加できます。

19. ［次へ］をクリックします。

    注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

20. ［終了］をクリックしてウィンドウを閉じ、イメージを格納します。

21. メイン ウィンドウで生成された VSM を確認して保存します。

JMS

以下のトピックでは、JMS トランスポート プロトコルを使用してサービス イメージを記録する詳細な手順について説明します。

- 基本的なプロキシ レコーディング（P. 166）
- TIBCO モニタ レコーダ（P. 171）

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。

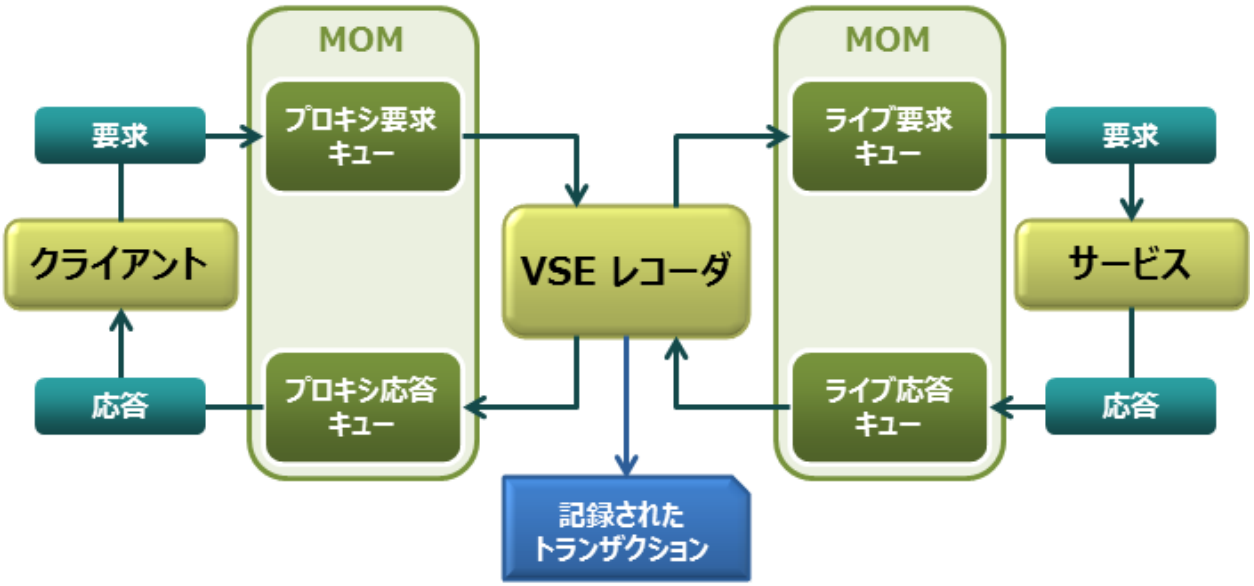
基本的なプロキシ レコーディング

*プロキシ レコーディング*は、メッセージング アプリケーションを記録する一般的な方法です。

以下の図は、プロキシ レコーディングの主要なコンポーネントを示しています。

- クライアント
- サービス
- プロキシ キュー
- ライブ キュー
- VSE レコーダ

メッセージ指向ミドルウェア（MOM）は、メッセージが交換されるプラットフォームです。



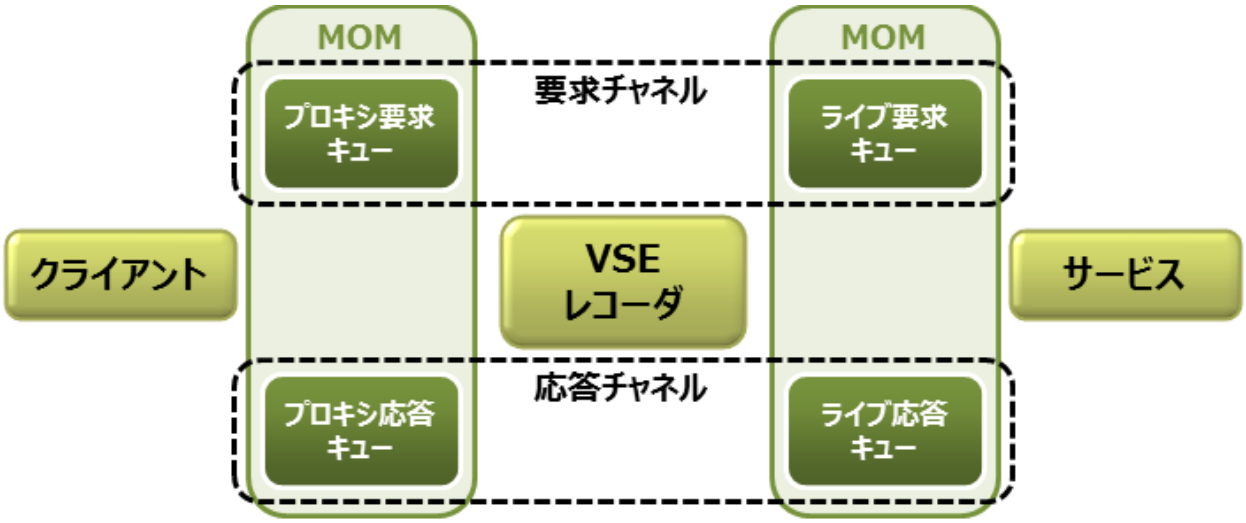
VSE レコーダは、クライアントとサービスの間のメッセージ フローに挿入されます。 各要求メッセージは、サービスに到達する前に、レコーダを経由します。各応答メッセージは、クライアントに到達する前に、レコーダを経由します。

*チャネル*は、プロキシ キューとライブ キューの組み合わせです。

プロキシ要求キューとライブ要求キューは、要求チャネルを形成します。

プロキシ応答キューとライブ応答キューは、応答チャネルを形成します。

以下の図は、プロキシ レコーディングの要求チャネルと応答チャネルを示しています。



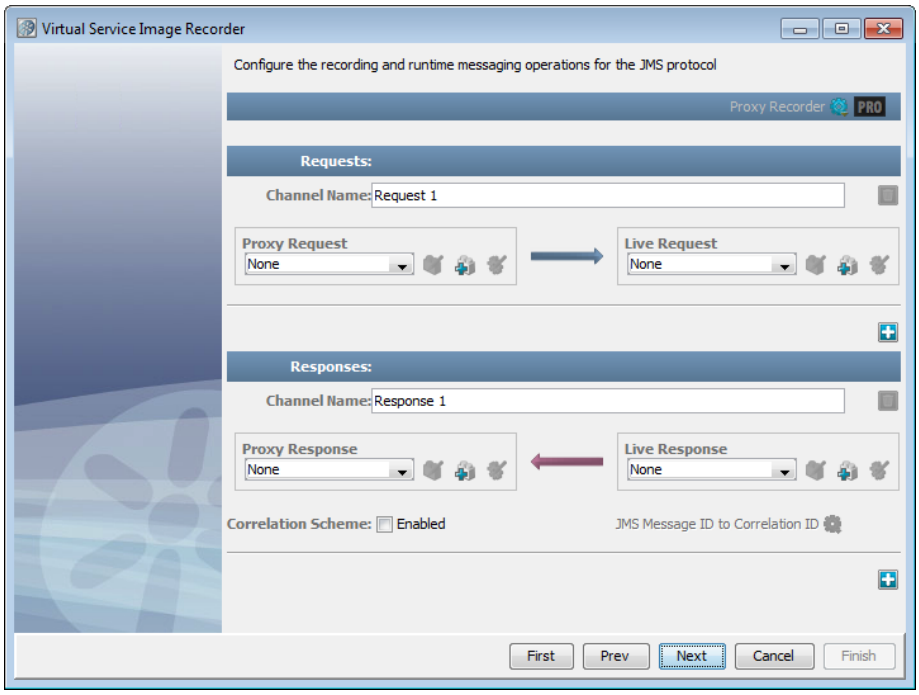
プロキシ レコーディングには、複数の要求チャネルおよび複数の応答チャネルが存在できます。

この手順で、各キューのアセットを選択します。 *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。 アセットの詳細については、「*CA Application Test 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 イメージ レコーダの［基本］タブで、トランスポート プロトコルを［JMS］に設定します。 サービス イメージと仮想サービス モデルを必ず設定してください。
2. ［次へ］をクリックします。

   プロキシ レコーダ セットアップ ページが表示されます。

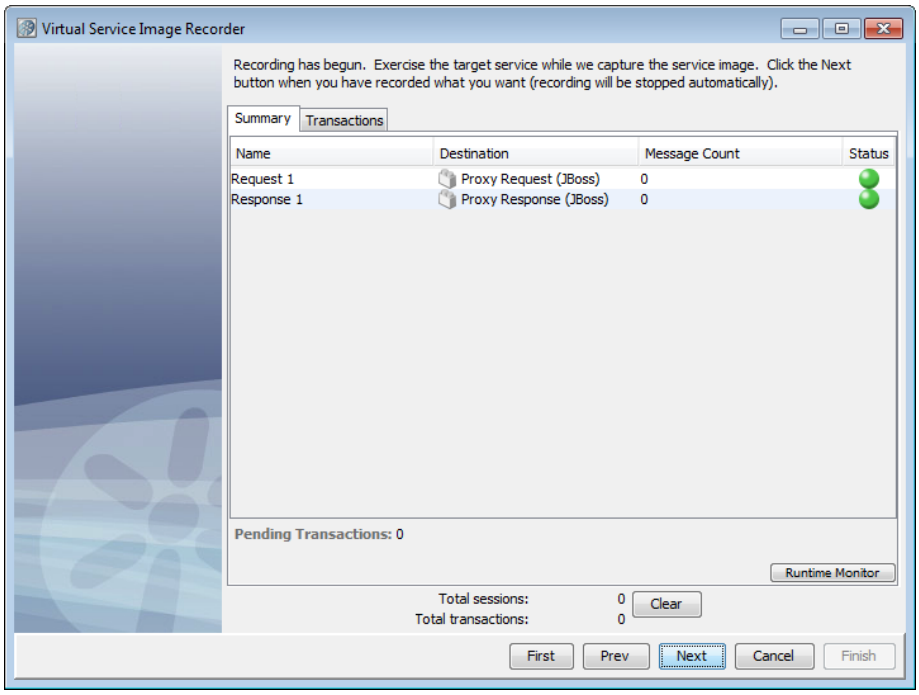


プロキシ レコーダ セットアップ ページには、基本パラメータと詳細パラメータがあります。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。

各チャネルには、一意の名前を付ける必要があります。 このチャネル名は、どのキューがトランザクションで使用されているかについて、仮想サービス モデルとサービス イメージが通信する手段です。

3. ［プロキシ要求］、［ライブ要求］、［プロキシ応答］、および［ライブ応答］リストでキューのアセットを選択します。 アセットが作成されていない場合、このページから作成できます。 また、このページからのアセットを編集できます。
4. （オプション）プラス（+）アイコンをクリックして、要求チャネルおよび応答チャネルを追加します。
5. （オプション）［相関スキーム］チェック ボックスをオンにし、目的のスキームを指定します。 応答チャネルにはそれぞれ別の相関スキームを指定できます。 相関スキームの説明については、「*CA Application Test の使用*」の JMS 送信/受信ステップのドキュメントを参照してください。
6. ［次へ］をクリックして、レコーディング セッションを開始します。

   レコーディング フィードバック ページが表示されます。



［サマリ］タブのテーブルには、各要求チャネルおよび応答チャネルがリスト表示されます。 各行には、チャネル名、選択された送信先アセット、チャネルを通過したメッセージ数、およびステータスが含まれます。

［送信先］列のセルをクリックすると、チャネルと関連付けられているプロキシ送信先を参照できます。

ページの下部には、保留中のトランザクション数、セッション数、およびトランザクションの総数が表示されます。

エラーによってレコーディング セッションを開始できない場合、ダイアログ ボックスが表示され、プロキシ レコーダ セットアップ ページに戻ります。

エラーは、レコーディング セッション中にも発生する可能性があります。 特定の要求チャネルまたは応答チャネルと関連付けられたエラーが[ステータス]列に表示されます。ステータス アイコンが赤くなり、ツールヒントにエラー メッセージが表示されます。 より一般的なエラーがレコーディング フィードバック ページの上部に表示されます。

リアルタイムで使用されているアセットを表示するためにランタイム モニタを使用できます。ランタイム モニタがどのように動作するかの詳細については、「*CA Application Test の使用*」の JMS 送信/受信ステップのドキュメントを参照してください。

7. 目的のサービスを実行します。

8. サービスが完了したら、［次へ］をクリックしてレコーディングを終了します。
9. 任意のデータ プロトコルを指定し、［次へ］をクリックします。
10. （オプション）レコーディング セッション ファイルを保存します。
11. ［終了］をクリックします。

    サービス イメージ (P. 355)および仮想サービス モデル (P. 396)が作成されます。

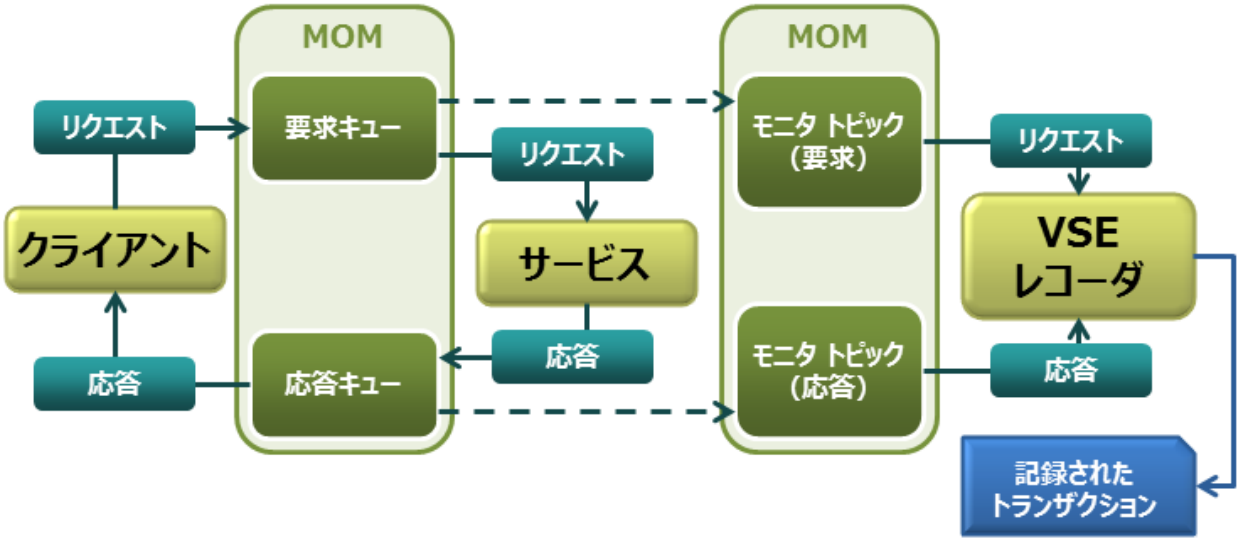
TIBCO モニタ レコーダ

TIBCO EMS サーバ上のすべてのキューで、管理者権限を持ったユーザは特別なモニタ トピックに接続できます。 このモニタ トピックをサブスクライブしている間、管理者は対応するキューを介して送信されるすべてのメッセージのコピーを受信できます。 この機能により、VSE レコーダを完全にアプリケーションのメッセージ フローの外側に置いて、メッセージ トラフィックのモニタのみを行えます。

以下の図は、TIBCO モニタ レコーディングの主要なコンポーネントを示しています。

- クライアント
- サービス
- 要求キューおよび応答キュー
- 要求モニタ トピックおよび応答モニタ トピック
- VSE レコーダ

メッセージ指向ミドルウェア（MOM）は、メッセージが交換されるプラットフォームです。



モニタトピック（要求）

アプリケーションの要求キューに関連付けられた TIBCO モニタ トピック。 TIBCO サーバは、要求キュー上で送信される各メッセージをこのトピックに自動的に転送します。

モニタトピック（応答）

アプリケーションの応答キューに関連付けられた TIBCO モニタ トピック。 TIBCO サーバは、応答キュー上で送信される各メッセージをこのトピックに自動的に転送します。

TIBCO モニタ レコーダは、TIBCO EMS 6.3 をサポートしています。

再生中にモニタ トピックを使用してサービスを仮想化することはできません。 以下のアクションのいずれかを実行できます。

- キューの個別のセットを作成し、それらを使用するようにクライアントおよび VSE サービスを設定する。
- VSE サービスがそれを置換できるように、ライブ サービスをシャットダウンする。

JMS 接続アセットでは、管理者ユーザは管理者権限が必要です。

次の手順に従ってください:

1. 仮想サービス イメージ レコーダの［基本］タブで、トランスポート プロトコルを［JMS］に設定します。 サービス イメージと仮想サービス モデルを必ず設定してください。
2. ［次へ］をクリックします。

   プロキシ レコーダ セットアップ ページが表示されます。
3. ［プロキシ レコーダ］から［TIBCO モニタ レコーダ］にレコーダ タイプを変更します。

   プロキシ レコーダ セットアップ ページには基本パラメータと詳細パラメータが含まれます。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。
4. ［受信先］リストおよび［送信先］リストでキュー アセットを選択します。 アセットが作成されていない場合、このページから作成できます。 また、このページからのアセットを編集できます。
5. （オプション）プラス（+）アイコンをクリックして、要求キューおよび応答キューを追加します。
6. （オプション）［相関スキーム］チェック ボックスをオンにし、使用するスキームを指定します。
7. ［次へ］をクリックして、レコーディング セッションを開始します。

   レコーディング フィードバック ページが表示されます。

［サマリ］タブ内のテーブルには、各要求チャネルおよび応答チャネルのリストが含まれます。 各行には、チャネル名、選択された送信先アセット、チャネルを通過したメッセージ数、およびステータスが含まれます。

8. 目的のサービスを実行します。
9. サービスが完了したら、［次へ］をクリックしてレコーディングを終了します。
10. 任意のデータ プロトコルを指定し、［次へ］をクリックします。
11. （オプション）レコーディング セッション ファイルを保存します。
12. ［終了］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 モデルが作成されます。

標準 JMS

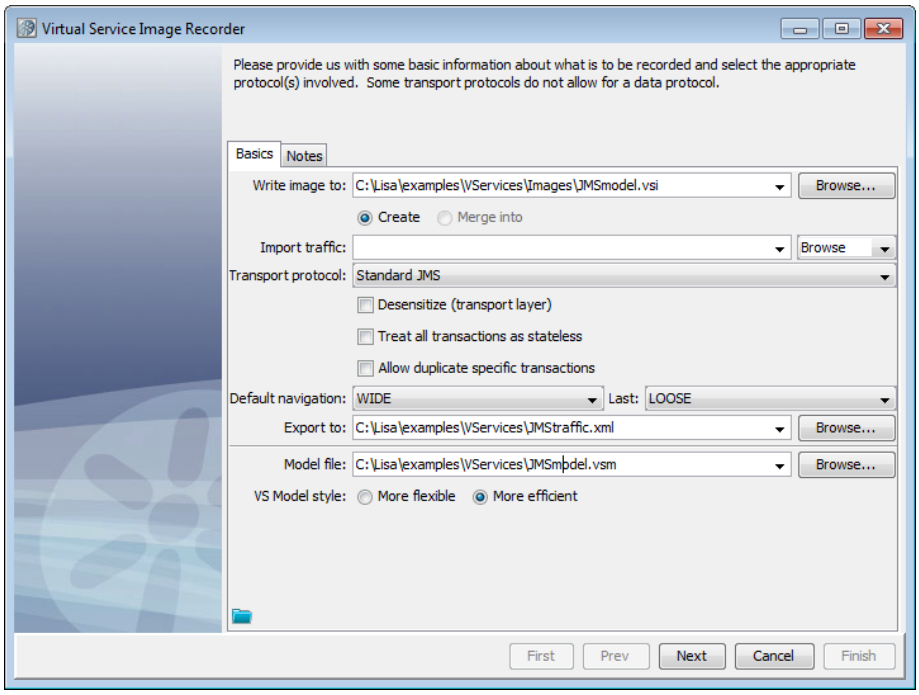
このトピックでは、標準 JMS トランスポート プロトコルを使用して仮想サービス イメージを記録する詳細な手順について説明します。

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. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。
2. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



3. ［次へ］をクリックします。

   以下のいずれかのオプションが選択された状態で、レコーディング モードの選択手順が表示されます。

プロキシ モード

プロキシ モードは、VSE がサポートする唯一のレコーディング モードです。 プロキシ モードには以下のオプションがあります。

- ライブ キューからのサービスの生成
- プロキシ キューを使用したサービスの生成

プロキシ モードでは、メッセージ バスで設定するプロキシ送信先を使用できます。 クライアント アプリケーションは、これらのプロキシ送信先にメッセージを送信するように設定されます。 DevTest は、それらを記録し、実際の送信先に転送します。 応答の際も同様の動作になります。DevTest は実際の応答先をリスンするように設定され、メッセージを取得すると、応答プロキシの送信先に転送します。 一時送信先を有効にして、応答側を自動化していると、異なった動作になる場合があります。

インポート モード

初期レコーディング ウィンドウの［インポート トラフィック］フィールドで RAW トラフィック ファイルを指定すると、仮想サービスを記録するためのこのモードが使用可能になります。 このレコーディング モードでは、RAW トラフィック ファイルで検出された要求および応答キューに関する追加の情報を設定できます。 また、要求応答ステップ全体をスキップするように選択することもできます。

4. プロキシ モードで記録するには、以下のフィールドに入力します。

ライブ/プロキシ キューを使用したサービスの生成

最終的な仮想サービス モデルおよびイメージを生成する場合に、ライブ キューまたはプロキシ キューを使用するかどうか。

最大保留トランザクション数

各応答キューで実行される応答リスナの最大数。 これらの応答リスナは、トランザクションが保留中である限り実行されます。 保留中のトランザクション数がこの最大数を超えると、最も古いトランザクションが自動的に閉じられます。 このフィールドに「0」を入力した場合、最大数は設定されません。

複数の応答を無効化

各トランザクションで複数の応答をサポートするかどうか。 デフォルトでは、トランザクションは最初の応答の後に保留中となり、同じトランザクションに対する追加の応答の受信を待機します。 このオプションを選択すると、トランザクションは最初の応答の後に自動的に閉じられます。 多数のトランザクションを立て続けに受信する場合、このオプションはパフォーマンスを向上させることができます。これは、MQ に対して特に有効です。

5. インポート モードで記録するには、以下のフィールドに入力します。

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

このチェック ボックスをオンにすると、要求ステップと応答ステップ全体をスキップできます。 RAW トラフィック ファイルを CAI から受信した場合、このオプションはオフになります。CAI は、キューとトランザクションを自動的に検出します。 その他の場所から RAW トラフィック ファイルを受信した場合、このオプションはデフォルトでオンになり、送信先とトラッキングの設定を確認できます。

相関

個別のトランザクションの要求側と応答側を関連付けるためのスキームが含まれます。 JMS は非同期です。これは、要求と応答が個別に受信されることを意味しています。 このドロップダウン リストでは、VSE レコーダに対して、どの要求をどの応答に関連付けるかを定義できます。 ［相関］フィールドには以下のオプションがあります。

- ［シーケンシャル］: すべての応答は、受信した最終要求に時系列で関連付けられます。相関スキームはありません。そのため、VSE MQ レコーダは、ライブ応答キューに対して排他的読み取りロックを取得し、別の MQ リスナがそのキューから応答メッセージを受信できないようにします。

  ［相関 ID］または［メッセージ ID と相関 ID］などの相関スキームを指定した場合、VSE MQ レコーダはライブ応答キュー上のその他のリスナも相関スキームを使用すると想定します。ライブ応答キュー上のすべてのリスナが相関スキームを使用する場合、VSE MQ レコーダはその応答を排他的読み取りロックを使用せずに個別に受信できるため、共有の入力フラグを使用してキューを開きます。

- 相関 ID： 要求と応答は同じ相関 ID を持つ必要があります。
- メッセージ ID と相関 ID: 要求のメッセージ ID と応答の相関 ID が同じである必要があります。

- メッセージ ID： 要求と応答が同じメッセージ ID を持ちます。

6. ［次へ］をクリックします。

   ［送信先情報］タブが表示されます。

7. プロキシとライブ送信先名を入力し、送信先タイプを選択します。
8. ［接続のセットアップ］タブをクリックします。
9. MOM への接続に使用する接続パラメータを入力します。

   これらの接続パラメータは内部に保存されます。

   ［接続のセットアップ］タブの下部にある［詳細］タブでは、サービス イメージのカスタム接続プロパティを定義できます。

10. ［次へ］をクリックします。

    ［送信先リスト］タブが開きます。

11. 応答の送信先がメッセージをリスンするための接続の詳細を定義します。
12. クライアント アプリケーションがそれらの応答を受信するプロキシ キューを定義します。

    - 単一の要求に対して複数の応答が可能であることに留意してください。応答プロキシ キューのセットを登録するには、［追加］をクリックします。
    - 不明な要求に対して使用する応答キューを登録するには、［不明な応答に使用する］チェック ボックスをオンにします。
    - 一時送信先を定義するには、［一時キュー/トピックを使用する］チェック ボックスをオンにします。 このオプションを選択すると、送信先名およびプロキシ送信先名フィールドが無効になり、追加する応答リスナが一時応答としてマークされます。 送信先名は空白です。

      メッセージングでの「一時」送信先は、メッセージング クライアント用にオンデマンドで作成される送信先です。 一時送信先は、通常、要求/応答シナリオで使用されます。 DevTest は、応答に対して一時キューの使用をサポートしていますが、一時キューでは一度に 1 つの同時トランザクションのみをサポートします。

13. ［現在の接続情報］タブをクリックして、接続情報が正しいことを確認します。

このタブに表示される情報は、以前に指定した接続情報からコピーされます。 まれに、応答接続情報が異なる場合には、それを変更できます。

14. ［次へ］をクリックして、レコーディングを開始します。

    VSE がリスンしている送信先の名前が表示されます。

    ［保留中トランザクション］フィールドには、追加の応答を待機している保留中のトランザクション バッファに格納されているトランザクションの数が表示されます。 このトランザクションは、保留中のトランザクション数の最大値に達するか、または［複数の応答を無効化］オプションを使用すると閉じられます。 トランザクションは、閉じられると、保留中のトランザクション バッファから合計トランザクション バッファに移動されます。

15. 要求プロキシ キューにメッセージを追加するクライアントを実行します。

    VSE は、実際の要求キューにメッセージをコピーします。 サーバは、そこからそれらのメッセージを取得し、応答キューに応答を送信します。 VSE は、再度それらのメッセージを取得し、クライアントがリスンしている応答プロキシ キューにコピーします。

    トランザクションが記録されるに従ってメッセージ数が増加し、［仮想サービス イメージ レコーダ］ウィンドウの［合計セッション数］と［合計トランザクション数］が増加します。 レコーディングの終了時には、すべての要求が同じ要求キューを経由しています。 応答の約半分は一時キューから返され、別の半分は非一時応答キューを介して返されています。

    実行が完了すると、応答キューのメッセージ数が予想よりも 1 つ少ないことがあります。 単一の要求には複数の応答が存在する場合があるため、VSE は最後のトランザクションが完了したことをまだ認識していません。 したがって、最後のトランザクションに対応するメッセージはカウントされません。

    このパネルでは、推奨されているデータ プロトコルがデフォルトになります。ここで、要求側および応答側データ プロトコルの両方を追加または編集できます。選択したデータ プロトコルの設定パネルが存在する場合、次にそれらのパネルが表示されます。 詳細については、「データ プロトコルの使用」を参照してください。

16. ［次へ］をクリックします。

    最後のトランザクションが閉じられ、必要なクリーンアップが実行されます。

注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

17. ［終了］をクリックしてウィンドウを閉じ、イメージを格納します。
18. メイン ウィンドウで生成された VSM を確認して保存します。

Java

Java サービス イメージを記録するには、DevTest Java エージェントが必要です。 LISA Bank アプリケーションでは、エージェントはデフォルトでインストールされます。 その他のアプリケーションでは、エージェントをインストールしてレジストリを起動します。

テストの場合は、エージェントがインストールされているデモ サーバを起動するか、または LISA Bank アプリケーションを実行します。 その他の Java アプリケーションで実行するには、エージェントを設定します。エージェントの詳細については、「エージェント」を参照してください。

同じワークステーションで複数のレコーダを使用して、さまざまなクラスから複数の EJB リモート コールを同時に記録する場合、各レコーダで作成されたサービス イメージおよび VSM によって、集約されたクラスのリストが表示されます。

次の手順に従ってください:

1. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。

   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］ -［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 レコーダが開きます。

2. ［基本 (P. 140)］タブで、トランスポート プロトコルとして Java を選択します。
3. ［次へ］をクリックします。

   ［仮想化する Java クラスの選択］ウィンドウが表示されます。

4. 以下の列の間で選択したエージェントを移動するには、提供されている矢印を使用します。

   利用可能なオンライン エージェント

   接続可能なオンライン エージェントをリスト表示します。

   接続されたエージェント

リストには、仮想サービス モデルに対して接続するオンライン エージェントまたはオフライン エージェントが含まれます。 オフライン エージェントは、灰色の斜体フォントで表示されます。［利用可能なオンライン エージェント］リストからエージェントを選択できます。

いずれかのリストからエージェントを選択すると、ホスト名とメインクラスが表示されます。

5. リストにないエージェントを追加するには、接続したエージェントのリストの上のフィールドにエージェント名を入力し、［エージェントの追加］  をクリックします。

   このメカニズムは、以前は接続リストに存在しなかったオフライン エージェントを追加するために提供されています。エージェント名は空にすることはできません。または、接続したエージェントのリストにすでに存在します。入力したエージェント名がオンライン エージェント リストに存在する場合、そのエージェントは、接続した利用可能なエージェントのリストから接続したエージェントのリストに移動されます。

6. 検索によってクラスを選択するには、以下の手順に従います。

   a. ［クラスの検索］をダブルクリックします。

   b. 正規表現を使用して、完全修飾名（パッケージを含む）として検索文字列を入力します。

   c. 検索アイコンをクリックします。

   d. クラスを選択します。 一部のクラスは 2 回以上現れる場合があります。 クラスは 1 回のみ選択します。

   e. 右方向矢印をクリックします。

7. 手動で名前を入力してクラスを選択するには、以下の手順に従います。

   a. ［クラス名の手動入力］をダブルクリックします。

   b. 完全修飾クラス名を入力します。

   c. 右方向矢印をクリックします。

8. エージェントが示すクラスのリストからクラスを選択するには、以下の手順に従います。

   a. ［エージェント提案］をダブルクリックします。

   b. 取得  をクリックします。

c. クラスを選択します。

d. 右方向矢印をクリックします。

9. レコーディングにプロトコルを追加するには、［プロトコル］をダブルクリックして記録するプロトコルを選択します。
10. テスト中のシステムをシャットダウンします。
11. ［次へ］をクリックします。

    「レコーディングが開始されました。」というページが表示されます。
12. テスト中のシステムを再起動します。

    注: テスト中のシステムのシャットダウンと再起動はオプションです。再起動を行うと、テスト中のシステムの初期化時に発生する初期トラフィックを、VSE が確実にキャプチャできるようになります。

    テスト中のシステムが起動されると、最新のトランザクションのリストが表示されます。選択したトランザクションのコンテンツを参照するには、トランザクションをダブルクリックします。
13. レコーディングが完了したら、［次へ］をクリックします。

    ［データ プロトコル］パネルが開き、事前に入力された適切なデータ プロトコルが表示されます。データ プロトコルを変更または追加できます。データ プロトコルを設定するための追加のパネルが存在する場合があります。データ プロトコルの設定の詳細については、「データ プロトコルの使用」を参照してください。
14. データ プロトコルを設定したら、［次へ］をクリックします。

    .vsi ファイルを作成する準備をしている間、レコーダは、要求と応答のボディがテキストであることを確認します（テキストとしてマークされている場合）。 テキストではない場合、タイプがバイナリに切り替わります。

    注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］ をクリックします。
15. ［終了］をクリックします。

JDBC

このトピックでは、JDBC （ドライバ ベース）トランスポート プロトコルを使用して仮想サービス イメージを記録する詳細な手順について説明します。

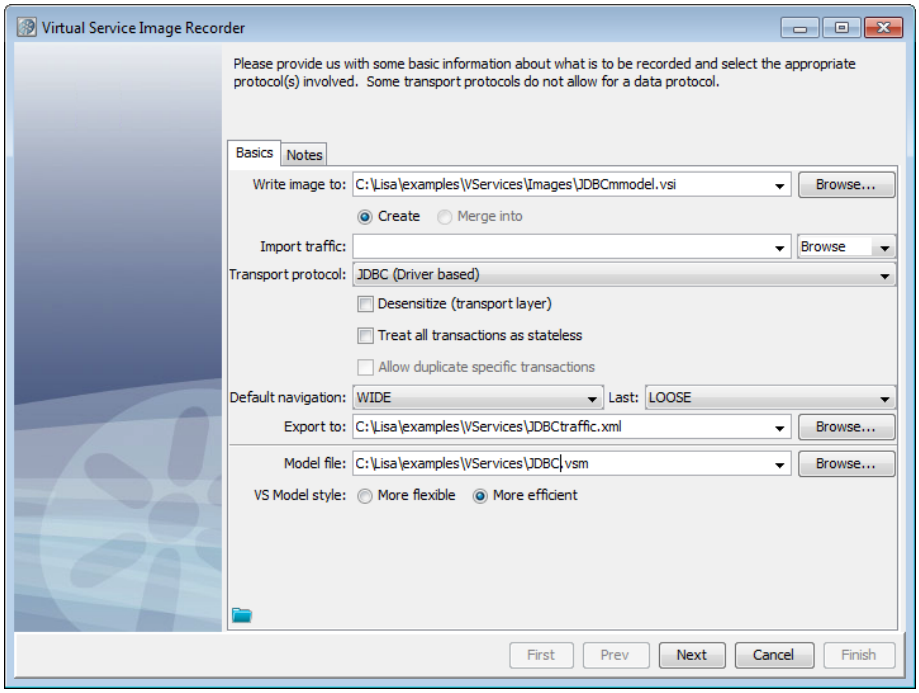
注: JDBC（ドライバ ベース）トランスポート プロトコルは、将来のリリースで提供終了予定です。 この機能はエージェントベースの JDBC 仮想化に置き換えられます。

前提条件と準備手順の詳細については、「JDBC の仮想化 (P. 188)」を参照してください。

次の手順に従ってください:

1. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。
2. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



3. ［次へ］をクリックします。

   エンドポイントの設定ウィンドウが開きます。

4. 必要に応じて、シミュレーション ホストとポートの範囲（デフォルトは 2999）を設定します。

   JDBC VSE は、レコーディングおよび再生時に複数のエンドポイントをサポートします。 レコーダおよび JDBC リスン ステップ エディタには、［ドライバ ホスト］、［ベース ポート］、および［最大ポート］のテーブルがあります。 ［ベース ポート］と［最大ポート］が異なる場合、ベース ポート、最大ポート、各ポート間で一意のエンドポイントが作成されます。

5. ［次へ］をクリックして、JDBC シミュレーション サーバに接続します。

   接続の試行を停止するには、ステータス ウィンドウで［キャンセル］をクリックします。

6. ［SQL アクティビティ］タブに記録される接続 URL とユーザを選択します。

   ［SQL アクティビティ］タブには、すべての一意の接続 URL とデータベース ユーザの組み合わせに対して実行された最新の 10 個のステートメント（および実行回数）が表示されます。 この表示は、約 5 秒ごとにリフレッシュされます。

   - 選択した接続文字列とユーザをパネル下部の URL およびユーザ フィールドに入力するには、［URL にコピー］をクリックします。
   - ［記録する URL］リストに URL を入力するには、［追加］をクリックします。

7. ［ロード済みドライバ］タブをクリックします。

   ［ロード済みドライバ］タブには、テスト中のシステムにインストールおよび登録されている JDBC ドライバがリスト表示されます。

   - ドライバを選択し、［URL にコピー］をクリックして、パネル下部の［URL］フィールド内のドライバを介した任意の接続に一致する正規表現を入力します。
   - ［追加］をクリックして、［記録する URL］リストにそのパターンを追加します。

   記録する URL

   ［SQL アクティビティ］リストまたは URL/ユーザ エントリ フィールドから追加された URL が表示されます。 エントリを削除するには、エントリを選択して［削除］をクリックします。

ユーザ

［記録する URL］フィールドの下の名前のない領域であり、データベース ユーザと URL を含めることができます。 これを空白のままにしてデータベース ユーザを含めない場合、その URL に接続するすべてのユーザに対してレコーディングが行われます。

注:［追加］ボタンを有効にするには、最初のフィールドに接続 URL を入力し、次のフィールドにユーザ名を入力します。

下部のセクションに、記録する接続 URL とデータベース ユーザの組み合わせのリストが表示されます。URL には、正規表現を使用できます。テスト中の DataSource スタイルのシステムに対するシミュレーション接続 URL で、state 属性が RECORD に設定されていない限り、通常、このリストは最初空です。

ドライバ リストからドライバが選択されている場合、［URL にコピー］ボタンを使用して、そのドライバの接続 URL と一致する汎用の正規表現を URL 入力フィールドにコピーできます。アクティビティ ツリーで接続 URL （トップ レベル）ノードが選択されている場合、［URL にコピー］ボタンを使用して、URL とユーザを URL およびユーザ入力フィールドにコピーできます。 また、レコーディング リストに接続 URL とユーザを直接追加するには、アクティビティ ツリーの右側の［追加］ボタンを使用します。

接続 URL には、正確な接続要求、または接続要求に一致する正規表現を使用でき、データベース ユーザあり、またはデータベース ユーザなしでレコーディング リストに追加できます。 データベース ユーザを指定しない場合、接続を試行する任意のユーザに一致します。 URL およびユーザフィールドの右側の［追加］ボタンが無効の場合は、その URL とユーザの組み合わせがすでにレコーディング リストに存在する可能性があります。

8. 記録する URL とユーザがすべてレコーディング リストに含まれたら、［次へ］をクリックします。

「レコーディングが開始されました。」というウィンドウが表示されます。

オプションと動的な統計表示には、以下のものが含まれます。

合計セッション数

記録された会話の数。

合計トランザクション数

記録されたトランザクションの数。

クリア

現在記録されているトランザクションをクリアするには、このボタンをクリックします。

記録されているトランザクションの数に従って、[合計セッション数]と[合計トランザクション数]の数が増加します。

注: トランザクションが記録されない場合、ポートの競合が発生している可能性があります。クライアントは、仮想サービス レコーダの代わりに、アプリケーションにトランザクションを送信します。 別のサービスがそのポートを使用している場合は、そのサービスを停止するか、またはポート設定を変更して競合を解消します。

パフォーマンスに問題がある場合、ターゲット データベースの応答が遅い可能性があります。 **user.home¥lisatmp¥tmanager.log** ファイルで、クエリの実行時間をレポートするデバッグ メッセージを確認してください。

以下に例を示します。

```
2009-07-01 15:35:39,248 [AWT-EventQueue-0] DEBUG
com.itko.lisa.vse.stateful.model.SqlTimer - Exec time 72ms :
SELECT TRAFFIC_ID, LAST_MODIFIED, SERVICE_INFO, CREATED_ON, NOTES,
UNK_CONV_RESPONSE, UNK_STLS_RESPONSE FROM SVSE_TRAFFIC
```

クエリ時間がしきい値を超えている場合、以下のような警告メッセージが生成されます。

```
2009-07-01 15:17:11,161 [AWT-EventQueue-0] WARN
com.itko.lisa.vse.stateful.model.SqlTimer - SQL query took a long
time (112 ms) : SELECT TRAFFIC_ID, LAST_MODIFIED, SERVICE_INFO,
CREATED_ON, NOTES, UNK_CONV_RESPONSE, UNK_STLS_RESPONSE FROM
SVSE_TRAFFIC
```

9. レコーディングが完了したら、[次へ]をクリックします。

   レコーダは、要求と応答のボディを検証することにより、.vsi ファイルを作成する準備をします。 テキストとしてマークされている場合、レコーダはそれがテキストであるかどうかを確認します。 テキストではない場合、タイプがバイナリに切り替わります。

   レコーダは、レコーディングの後処理を完了します。

   注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、[終了]ボタンの上の[保存] をクリックします。

10. [終了]をクリックして、イメージを格納します。

11. DevTest ワークステーション で仮想サービス モデルを確認および保存します。

JDBC の仮想化

このセクションでは、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、VSE フレームワークを使用する準備を行う方法の詳細について説明します。 JDBC の設定については、「データベース シミュレータのインストール (P. 24)」でも説明しています。 このセクションでは、ドライバ ベース JDBC 仮想化を組み合わせるさまざまな方法、およびサポートされる設定オプションについて説明します。 このセクションでは、特定のアプリケーション コンテナに対して設定を実装する方法については説明していません。

注: JDBC 仮想化は、Java 1.6 でのみ動作します。Java 1.5 では動作しません。

システム レベル プロパティ

以下のいずれかの方法で以下のプロパティを指定できます。

- システム変数の使用（テスト中のシステムの起動で **-Dvar = val** フラグを追加するか、またはその他のメカニズムを介して）
- **rules.properties** という名前のプロパティ ファイル。このファイルは、クラスパス上に存在する必要があります（多くの場合、**lisajdbcsim.jar** と同じディレクトリ内）。

プロパティが両方の場所に存在する場合、システム プロパティが設定ファイルよりも優先されます。

lisa.jdbc.sim.require.remote

コマンドを処理する前に、ドライバが VSE への接続を待機するかどうかを決定します。

デフォルト： false

lisa.jdbc.hijack.drivermanager

ドライバが、このアプリケーションからのデータベース接続をすべて引き継ごうとするかどうかを決定します。

デフォルト： false

lisa.jdbc.sim.port

デフォルトのリスン ポートを定義します。

デフォルト： 2999

lisa.log.level

デフォルトのログ レベルを指定します。

値

- OFF
- FATAL
- ERROR
- WARN
- INFO
- DEBUG
- DEV

デフォルト： WARN

lisa.log.target

ログを書き込む場所を指定します。

値

- stderr
- stdout
- 任意のファイルの場所

デフォルト： stderr

VSE ドライバ（スタンドアロン）

テスト中のシステムは、VSE ドライバを直接使用できます。または、VSE ドライバを Apache BasicDataSource などのデータ ソースでラップすることができます。 このような場合、設定（ユーザ名とパスワード以外）は URL を介して渡されます。 ドライバ プロパティには以下のものが含まれます。

URL

基盤となるドライバに渡される実際の URL を指定します。この URL は、最後のエレメントとして渡す必要があります。

State

ドライバの初期状態を指定します。

値

- watch

- record
- playback

JdbcSimPort

VSE からの接続をリスンするポートを指定します。 この値は、システム レベルのプロパティを上書きします。

Driver

使用する実際のドライバのクラス名を指定します。

User

Password

ドライバのクラス名として、**com.itko.lisa.vse.jdbc.driver.Driver** を指定します。

以下の例は URL の形式を示しています。 ドライバと URL 以外はすべてオプションです。

```
jdbc:lisasim:driver=<実際のドライバ クラス>;state=<初期状態
>;jdbcSimPort=<開始点>;url=<実際の URL>
```

注: URL エレメントは、最後に指定する必要があります。「url=」以降は、すべて変更されることなく基盤となるドライバに渡されます。 DevTest は、実際の URL に埋め込むことにより、基盤となるドライバに追加の情報を渡すことをサポートしています。

たとえば、Derby 接続には以下の URL が適切です。

```
jdbc:lisasim:driver=org.apache.derby.jdbc.ClientDriver;state=watch
;jdbcSimPort=4000;url=jdbc:derby://localhost:1527/sample;create=tr
ue
```

この URL は、以下のことを VSE ドライバに指示します。

- Derby クライアント ドライバを使用すること
- 「watch」モードで起動すること
- ポート 4000 を使用して VSE と通信すること
- URL jdbc:derby://localhost:1527/sample;create=true を実際の Derby ドライバに渡すこと

ドライバをラップする VSE データ ソース

ドライバをラップする VSE データ ソースは、VSE ドライバを直接使用するよりも、アプリケーション コンテナに対する一般的な設定の場合があります。 VSE は、実際のドライバをラップするデータ ソースを提供します。ほとんどのアプリケーション コンテナは、データ ソースのプロパティを直接指定するメカニズムを提供しています。

ドライバのクラス名として、**com.itko.lisa.vse.jdbc.driver.VSEDataSource** を指定します。

データ ソース プロパティ

Driver

使用する実際のドライバの名前を指定します（通常の設定）。

URL

接続する実際の URL を指定します（**jdbc:lisasim** で開始することはできません）。

User

Password

JdbcSimPort

VSE からの接続をリスンするポートを指定します。 この値は、システム レベルのプロパティを上書きします。

CustomProperties

セミコロンで区切られた「名前 = 値」ペアのリストを定義します。 これらのプロパティが存在する場合、解析され、ラップされたデータ ソースに委任されます。 最初の文字が英数字でない場合、その文字がセミコロンの代わりに区切り文字として使用されます。

データ ソースをラップする VSE データ ソース

アプリケーション コンテナがドライバを Class.forName() で 直接インスタンス化するのを許可しない場合、データ ソースのラップは一般的な設定ではありません。 これを使用するには、**driver=<実際のドライバのクラス名>**（**oracle.jdbc.OracleDriver** など）を指定する代わりに、**datasource=<実際のデータ ソースのクラス名>** （**oracle.jdbc.pool.OracleDataSource** など）を指定します。

ドライバのクラス名として、**com.itko.lisa.vse.jdbc.driver.VSEDataSource** を指定します。

複数のエンドポイントのサポート

1 つのアプリケーションに仮想化する JDBC 接続が複数存在する場合（または、1 台のアプリケーション サーバに、仮想化する別々の JDBC 接続を持つ複数のアプリケーションが存在する場合）、各接続に対して個別の **JdbcSimPort** を指定します。 このエンドポイントは、VSE データ ソースのプロパティ、またはドライバの URL 上のパラメータのいずれかとして指定できます。

例:

アプリケーションが 2 つの JDBC 接続を使用しており、その両方を仮想化すると仮定します。 また、アプリケーションが、ドライバを直接使用すると仮定します。

driver や URL など、それぞれに基盤となる適切なパラメータを付けて、各接続に **com.itko.lisa.vse.jdbc.driver.Driver** を指定します。 最初の接続に **jdbcSimPort=3000** を指定し、2 番目に **jdbcSimPort=3001** を使用することもできます。 その場合の設定は以下のようになります。

```
connection1.url=jdbc:lisasim:driver=org.apache.derby.jdbc.ClientDr
iver;jdbcSimPort=3000;url=jdbc:derby://localhost:1527/sample;creat
e=true
```

```
connection2.url=jdbc:lisasim:driver=org.apache.derby.jdbc.ClientDr
iver;jdbcSimPort=3001;url=jdbc:derby://localhost:1527/sample2;crea
te=true
```

引き続き個別に記録する必要がありますが、両方のサービスを同時に展開して、個別に開始および停止できます。

TCP

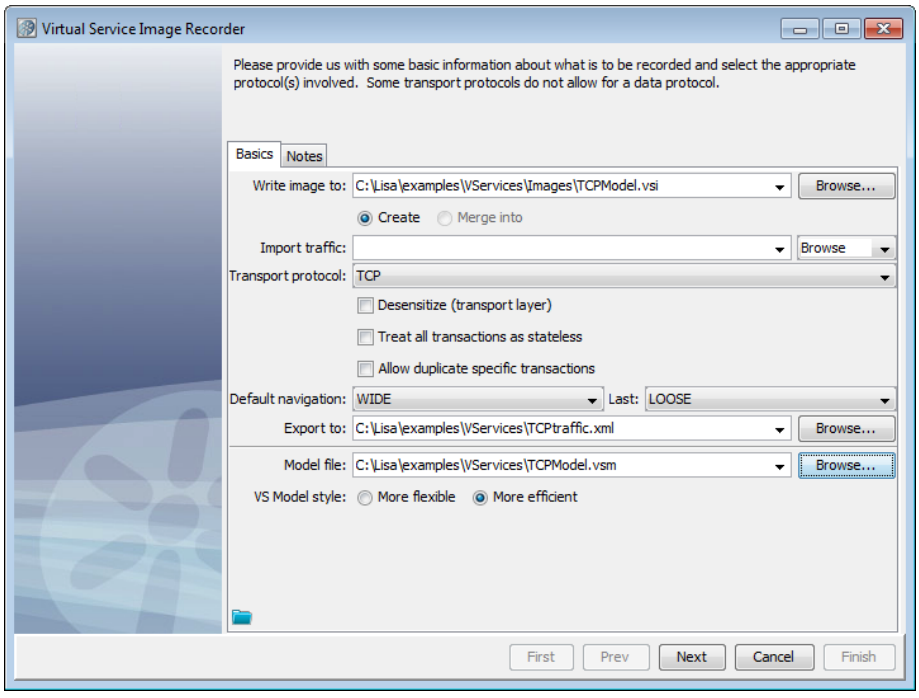
このトピックでは、TCP トランスポート プロトコルを使用して仮想サービス イメージを記録する詳細な手順について説明します。

前提条件と準備手順の詳細については、「TCP の仮想化 (P. 196)」を参照してください。

次の手順に従ってください:

1. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。
2. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



3. ［次へ］をクリックします。
4. クライアントとサーバの両方に情報を入力し、区切り文字を選択し、SSL パラメータを追加します。

   リスン/記録ポート

クライアントが DevTest と通信するポートを定義します。

ターゲット ホスト

サーバが実行されているターゲット ホストの名前または IP アドレスを定義します。

ターゲット ポート

サーバが実行されているターゲット ホストのポート番号を定義します。

要求をテキストとして処理

要求がテキストとして処理されるかどうかを指定します。 詳細については、「TCP の仮想化 (P. 196)」を参照してください。

Request Encoding

DevTest ワークステーション が実行されているマシンで使用可能な要求エンコーディングのリストを示します。 デフォルトは UTF8 です。

応答をテキストとして処理

応答がテキストとして処理されるかどうかを指定します。 詳細については、「TCP の仮想化 (P. 196)」を参照してください。

Response Encoding

DevTest ワークステーション が実行されているマシンで使用可能な応答エンコーディングのリストを示します。 デフォルトは UTF8 です。

サーバに SSL を使用

DevTest がサーバに要求を送信するために HTTPS を使用するかどうかを指定します。

- **オン：** DevTest は HTTPS （セキュア レイヤ）要求をサーバに送信します。

  ［サーバに SSL を使用］を選択した場合で、［クライアントに SSL を使用］を選択しない場合には、DevTest はレコーディングに HTTP 接続を使用します。 その後、DevTest は HTTPS を使用して、サーバにそれらの要求を送信します。

オフ： DevTest は HTTP 要求をサーバに送信します。

クライアントに SSL を使用

クライアントからの SSL 要求を再生するためにカスタム キーストアを使用するかどうかを指定します。 このオプションは、［サーバに SSL を使用］が選択されている場合のみ有効です。

値

- **オン：** カスタム クライアント キーストアおよびパスフレーズを指定できます。これらのパラメータを入力した場合、ハードコードされたデフォルトの代わりに使用されます。

オフ： カスタム クライアント キーストアおよびパスフレーズを指定できません。

SSL キーストア ファイル

キーストア ファイルの名前。

キーストア パスワード

キーストア ファイルのパスワード。

5. ［次へ］をクリックして、レコーディングを開始します。
6. レコーディングが完了したら、［次へ］をクリックします。
7. 応答が暗号化されるか、圧縮されるか、その他の方法でエンコードされている場合は、応答データ プロトコルを選択します。
8. レコーダは、完全な要求または応答を読み取った場合に DevTest に伝えるメッセージ区切り文字の検出を試みます。 この画面でこれらの区切り文字を確認または修正します。

   注: 要求区切り文字は必須です。 ライブ呼び出しを使用するには、応答区切り文字を選択する必要があります。
9. 要求と応答のボディがテキストとしてマークされている場合、レコーダはそれがテキストであるかどうかを検証します。 テキストではない場合、タイプがバイナリに切り替わります。

   注: このレコーディングに対する設定を保存して別のサービス イメージ レコーディングにロードするには、［終了］ボタンの上の［保存］  をクリックします。

   TCP/IP プロトコルは、ライブ呼び出しステップをサポートしています。ライブ呼び出しステップを有効にするには、応答プロトコルを選択します。 応答プロトコルを選択しない場合、VSM にはライブ呼び出しステップが含まれません。

TCP の仮想化

TCP トランスポート プロトコルでは、1 つ以上のクライアントと 1 つのサーバ間で RAW TCP/IP トラフィックを仮想化できます。 このトランスポートは HTTP プロトコルに類似しており、同じ機能の多くを使用します。

レコーディング時に、エンドポイント モードで HTTP を設定するのとまったく同様に TCP プロトコルを設定します。リスンするソケット、ターゲット サーバ、および送信先を指定します。 このプロトコルは、クライアントとサーバ間でデータを転送し、そのデータを「リスン」して VSE トランザクションを作成します。

バイトから VSE トランザクションへの変換

要求または応答を構成しているすべてのもの、およびそれらの要求と応答を関連付ける方法を知るのは簡単ではありません。TCP データには固有の構造がありません。 受信データ ストリームで要求を識別するには、要求区切り文字が必要です。 また、応答区切り文字は厳密には必要ありませんが、応答区切り文字も選択できます。

レコーダは、以下のルールを使用して要求と応答を特定し、要求と応答を一緒にトランザクションに関連付けます。

1. 要求には 0 または 1 つの応答が続きます。
2. 各応答は、完了した最新の要求とペアになります。
3. 要求が完了していないことを要求区切り文字が示している場合でも、応答データを受信すると完了していると見なされます。
4. 応答が完了していないことを応答区切り文字（存在する場合）が示している場合でも、要求データを受信すると完了していると見なされます。
5. 応答データの受信時に、応答区切り文字によって応答の完了が検出された場合、それ以降の応答データは破棄されます。

以下に例を示します。

要求 1 - 要求 2 - 応答 A - 部分的な要求 3 - 応答 B - 要求 4 - 部分的な応答 C - 要求 5 - 応答 D と後続データ

このデータは、以下のようにトランザクションへと解析されます。

- 応答のない要求 1
- 応答 A のある要求 2（ルール 2）

- 応答 B のある部分的な要求 3（ルール 3）
- 部分的な応答 C のある要求 4（ルール 4）
- 応答 D のある要求 5、ただし「後続データ」はドロップ（ルール 5）

クライアントとサーバが同期要求/応答パラダイムに従っており、応答のない要求を処理する場合には、このシステムが機能します。 このシステムは、非同期通信、または複数の応答を持つ要求を処理しません。

会話の作成

デフォルトでは、VSE は、TCP サービス イメージを記録し、マシン名（または IP アドレス）および送信ソケットに基づいて会話を作成します。 送信ソケットが変更されるたびに、新しい会話が開始されます。 この動作を変更するには、［すべてのトランザクションをステートレスとして処理］チェック ボックスをオンにします。

標準の区切り文字

DevTest には以下の区切り文字があります。

レコードは行末で終了する

1 つ以上の改行文字によって、各要求/応答が終了します。 区切り文字自体は、要求/応答に含まれません。

値

- ¥n
- ¥r
- Unicode 文字 0085、2028、2029

レコードは固定長

各要求/応答は、特定のバイト数の長さです。

レコードを特定の文字によって区切る

カンマなどの特定の文字または文字のシーケンスによって、各要求/応答が終了します。 シーケンス全体が一致する必要があります。 区切り文字自体は、要求/応答に含まれません。このオプションは、CSV ファイルのインポートに類似しています。

区切り文字は正規表現に一致する

正規表現に一致するバイト セットによって、各要求/応答が終了します。 区切り文字自体は、要求/応答に含まれません。

要求/応答に区切り文字が含まれていない場合（ほとんどの場合）、連続する区切り文字のセットは無視されます。

注: レコーディングが完了した後は、選択された区切り文字を編集できません。 モデル エディタでは、選択された区切り文字を変更できません。

［要求をテキストとして処理］および［応答をテキストとして処理］

［要求をテキストとして処理］チェック ボックスと［応答をテキストとして処理］チェック ボックスは、要求データと応答データの解析方法を制御します。

要求をテキストとして処理

要求を VSE 要求のボディとして使用するかどうかを指定します。

値

- **オン：** 要求全体を VSE 要求のボディ テキストとして設定します。 最初の「単語」（最初のスペース文字まで）は、操作名として使用されます。
- **オフ：** 要求ボディをバイナリとして処理し、操作名を OPERATION に設定します。

要求をテキストとして処理

要求を VSE 要求のボディとして使用するかどうかを指定します。

値

- **オン:** 応答ボディをテキストとして処理し、埋め込まれた NULL（¥0）をスペースに変換します。
- **オフ：** 応答ボディをバイナリとして処理します。

トランザクション数が正しくない理由

トランザクション数は、常に少なくとも 1 ずつ遅延します。 TCP プロトコルは、完了したトランザクションを検出すると、次のトランザクションが開始されるまでそのトランザクションをキャッシュします。その理由は、応答後に、サーバが意図的にソケットを閉じる場合があるためです。 この状況が発生すると、TCP プロトコルはそれを検出し、応答に **lisa.vse.close_socket_after_response=TRUE** を設定します。 このプロパティが設定されると、VSE は、再生時に応答を送信した後、サーバ側のソケットを閉じます。

応答区切り文字を使用しない場合、トランザクション数はさらに 1 つ遅延します。 TCP プロトコルは、新しい要求が開始されるまで、応答が完了したことを知りません。したがって、合計数に 2 つの差がある可能性があります。 レコーディングが完了すると、その数は正しくなります。

CICS の記録（LINK DTP MRO）

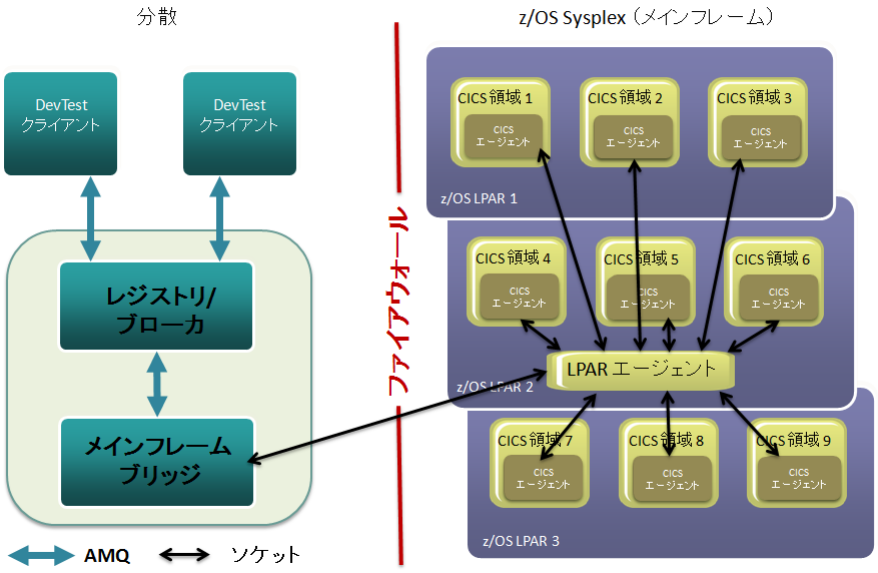
このセクションでは、CICS トランスポート プロトコルを使用して仮想サービス イメージを記録する詳細な手順について説明します。 このセクションには、以下のトピックが含まれます。

- CICS の概要（P. 200）
- CICS の例（P. 202）
- パネルの仮想化（P. 211）

IBM CICS の概要

IBM CICS サービス イメージのレコーディングを可能にするため、DevTest には、レジストリの一部であるメインフレーム ブリッジがあります。 メインフレーム ブリッジは、すべての DevTest ワークステーション にメインフレームに対する単一の接点を提供します。 メインフレーム上には、z/OS LPAR 上で実行される LPAR エージェントが存在します。 LPAR エージェントは、すべての CICS エージェントに単一の接点を提供します。

各 CICS 領域には、仮想化機能を提供する DevTest CICS エージェントが存在します。 CICS エージェントは、ソケットを介して LPAR エージェントと通信し、メインフレーム ブリッジはソケットを介して LPAR エージェントと通信します。 メインフレーム ブリッジは、レジストリを介してすべての DevTest クライアントに ActiveMQ インターフェースを提供します。



メインフレーム ブリッジは以下のモードで実行できます。

クライアント モード

メインフレーム ブリッジは、LPAR エージェントへの接続を開始します。

サーバ モード

メインフレーム ブリッジは、LPAR エージェントが接続を開始するのを待機します。

lisa.properties ファイルには、メインフレーム ブリッジを有効にするためのプロパティが含まれています。 詳細については、「メインフレーム プロパティ」を参照してください。

IBM CICS の例

この例では、VSE レコーダを使用して CICS LINK サービス イメージを記録します。

レジストリが起動されると、メインフレーム ブリッジがクライアント モードで開始されており、ブリッジが LPAR エージェントに接続していることが示されます。

この例は、リソースへの依存の解消を示します。 この例では、CICS アプリケーションは、口座残高情報フィールドへの入力に VSAM ファイルを必要としていますが、その VSAM ファイルは使用できません。 この例は、リソースへの依存を解消して、欠落した VSAM ファイルを保守するプログラムを仮想化する方法を示しています。

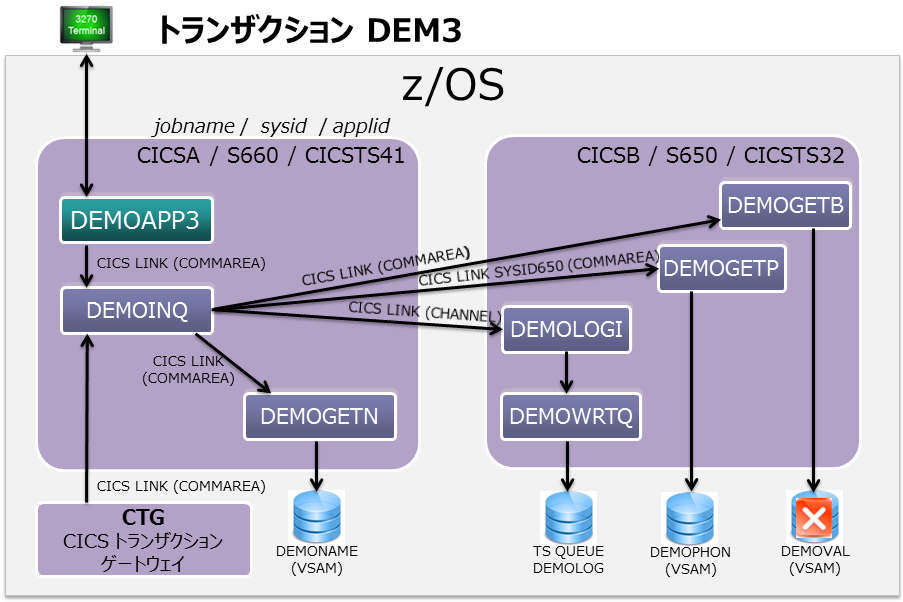
以下の図は、実行中の CICS トランザクション (DEM3) を示しています。このトランザクションは 3270 の端末から実行されますが、CICS トランザクション ゲートウェイから実行される場合もあります。

トランザクションを実行すると、DEMOAPP3 が開始され、DEMOINQ に CICS LINK を実行します。 その後、DEMOINQ は以下の CICS LINK を実行します。

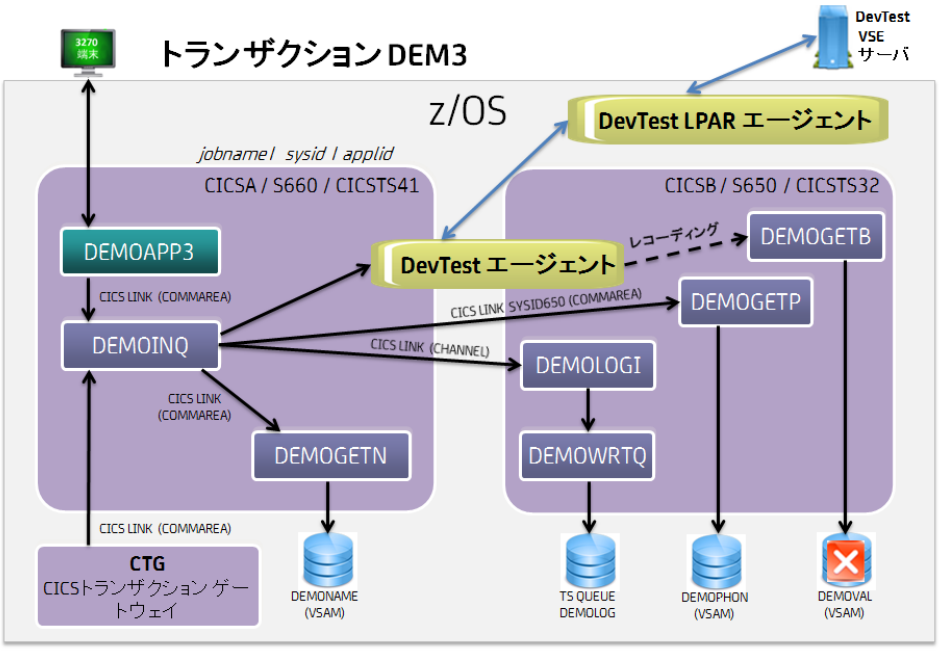
1. ローカル LINK は、DEMOGETN に COMMAREA を渡して、VSAM からカスタマ名とアドレスを取得します。
2. CICSB への CICS LINK は、DEMOGETB に COMMAREA を渡します。 DEMOGETB は、VSAM ファイル DEMOGETBAL を読み取りますが、これはこの例では使用できません。
3. CICS LINK は、DEMOGETP に COMMAREA を渡して電話番号情報を取得します。
4. CICS LINK は、DEMOLOGI に CONTAINERS を指定して CHANNEL を渡します。

以下の CICS 領域が含まれます。

- S660 のシステム ID および CICSTS41 の applid を持ったジョブ名 CICSA
- S650 のシステム ID および CICSTS32 の applid を持ったジョブ名 CICSB



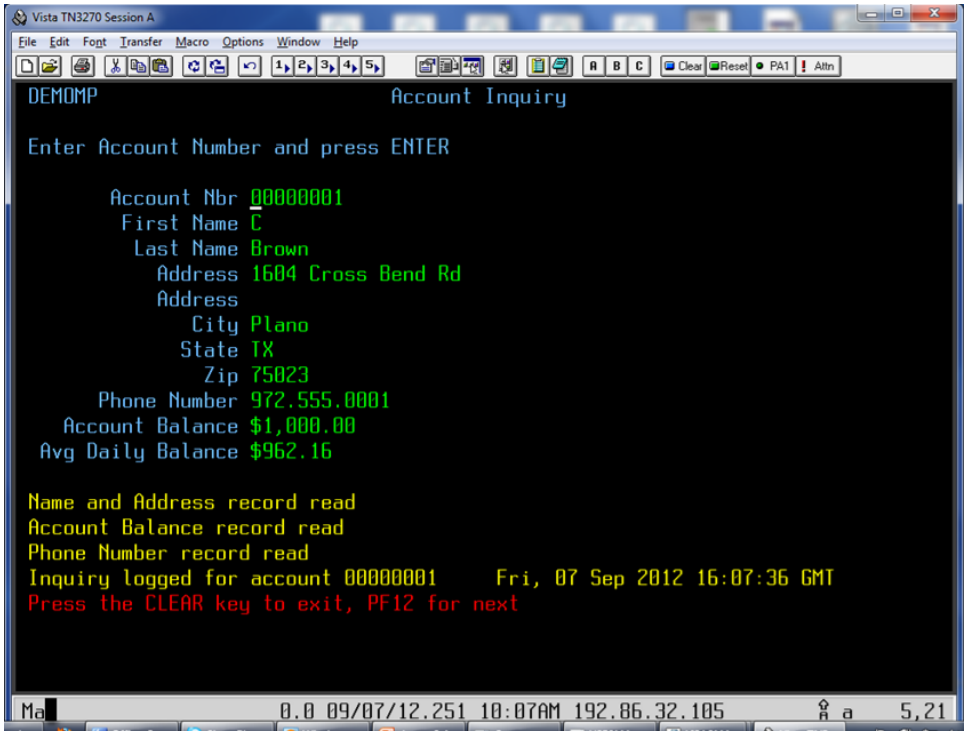
この例は、CICSTS41 上で DEMOGETB を仮想化します。 この例では、CICSA で実行され、LPAR エージェントと通信し、VSE サーバと通信する DevTest エージェントを使用します。 最初に、CICS トランザクションを実行して DEMOGETB への CICS LINK を記録します。 次に、そのレコーディングを使用して CICS LINK を仮想化します。



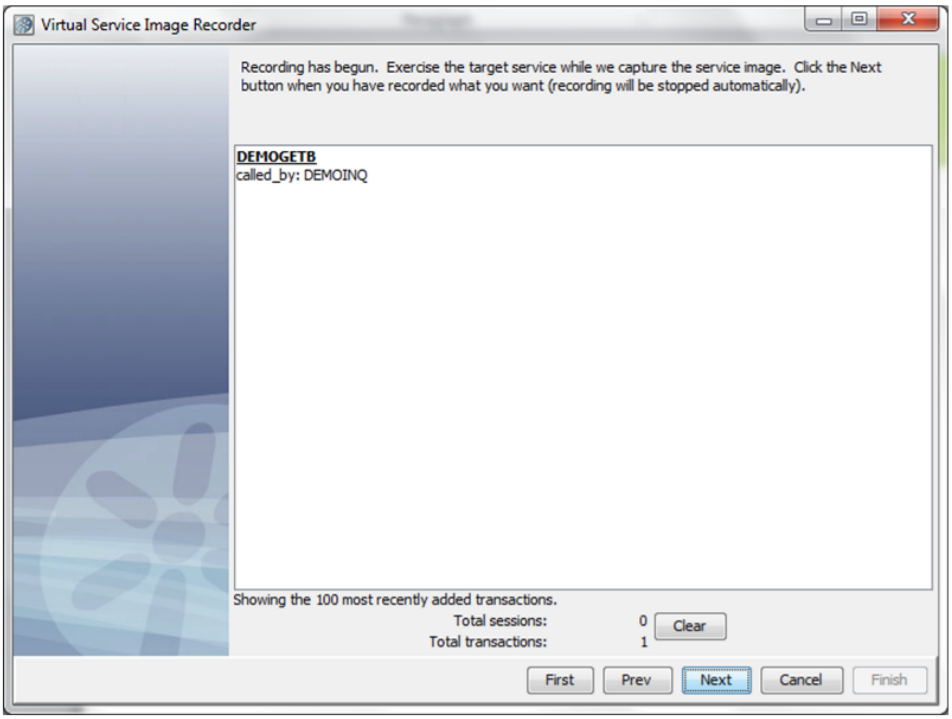
仮想サービス イメージ レコーダの［基本］タブには、サービス イメージの名前（DEMOGETB.vsi）、VSM（DEMOGETB.vsm）、および CICS LINK のトランスポート プロトコルが含まれます。

次に、CICS の［仮想化するプログラム］パネルが表示されます。 このパネルの詳細については、「パネルの仮想化 (P. 211)」を参照してください。

［次へ］ボタンをクリックすると、レコーディングが開始されます。 レコーディング中に、トランザクション DEM3 を実行して CICS LINK で DEMOGETB を呼び出します。



以下の図のように、1 回実行した後に、VSE は 1 つのトランザクションを記録します。



次のパネルでは、VSE は、CICS Copybook データ プロトコルを要求側と応答側の両方に追加します。

以下の図は、この例で使用する Copybook マッピング XML ファイルを示しています。 このファイルは、DEMOGETB のマッピングを定義します。また、プログラム DEMOGETB に対する要求が存在する場合、Copybook として DEMOGETB.txt を使用することを示しています。

payload-copybook-mapping-CICS.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<payloads>
    <!-- DEMOGETB -->
    <payload name="DEMOGETB" type="request" matchType="metaData"
             key="DEMOGETB" value="DEMOGETB" >
       <section name="Body">
            <copybook key="">DEMOGETB.txt</copybook>
        </section>
    </payload>
    <payload name="DEMOGETB" type="response" key="DEMOGETB" value="DEMOGETB">
        <section name="Body">
            <copybook key="">DEMOGETB.txt</copybook>
        </section>
    </payload>
</payloads
```

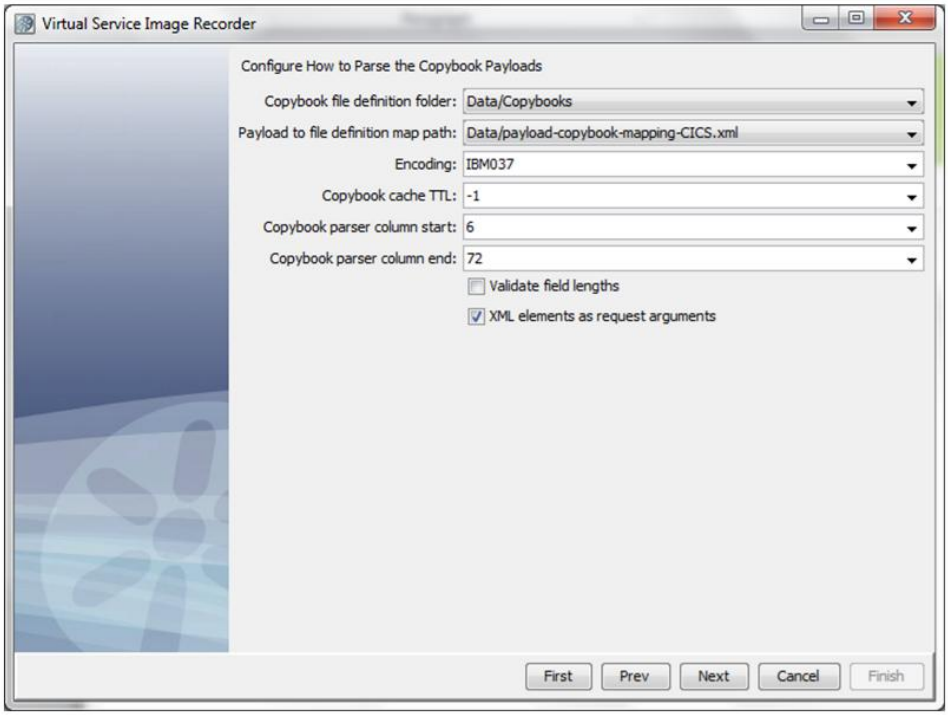
以下の例のように、DEMOGETB.txt には COBOL Copybook 定義が含まれています。

```
Copybooks/DEMOGETB.txt

   05  DEMOGETB-COMMAREA.

       10  DEMOGETB-RETURN             PIC X.
       10  DEMOGETB-MESSAGE            PIC X(70).
       10  BALANCE-REC.
           15  BALANCE-ACCOUNT-NBR     PIC X(8).
           15  FILLER                  PIC X.
           15  BALANCE-BALANCE         PIC X(14).
           15  FILLER                  PIC X.
           15  BALANCE-AVERAGE         PIC X(14).
```

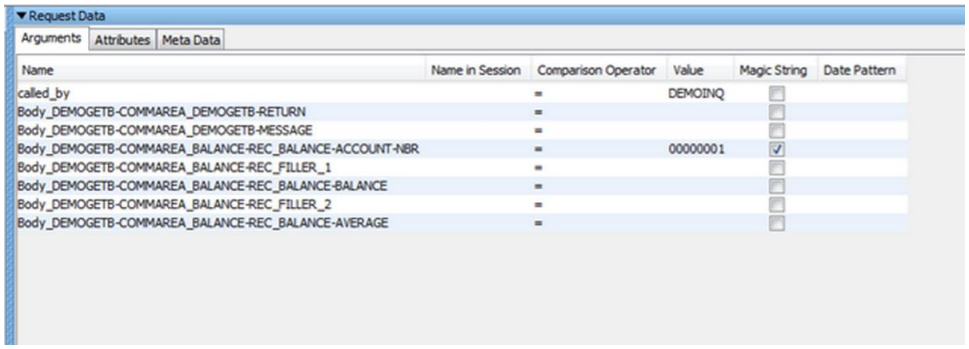
次のパネルでは、DEMOGETB.txt が存在するフォルダ、およびマッピング XML が指定されています。 コードページが IBMO37 に変更されています。



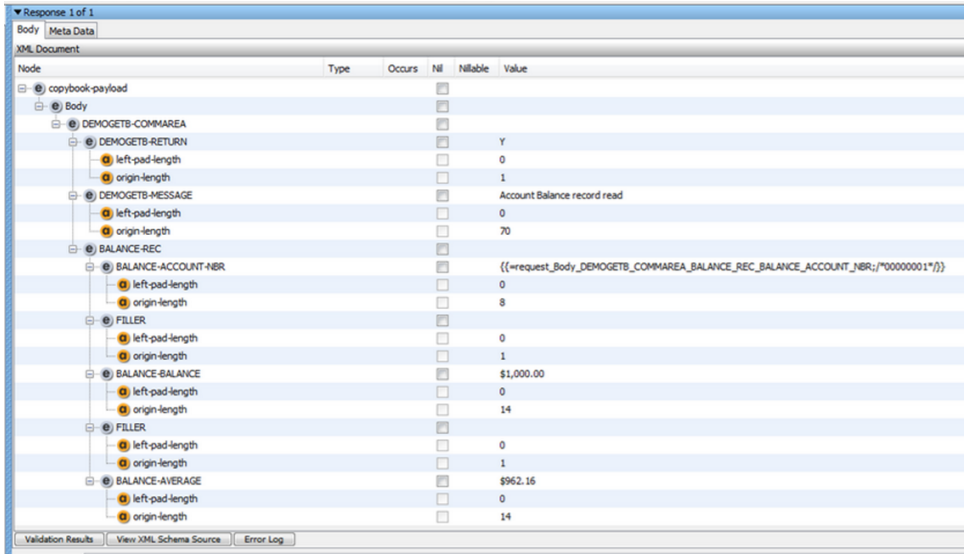
［次へ］をクリックします。 Copybook がデータを処理します。

［次へ］をクリックすると処理が停止し、サービス イメージのレコーディングが完了します。

サービス イメージ エディタで VSI を開くと、Copybook が要求 データをマップし、それを属性に変換したことが確認できます。 以下に例を示します。

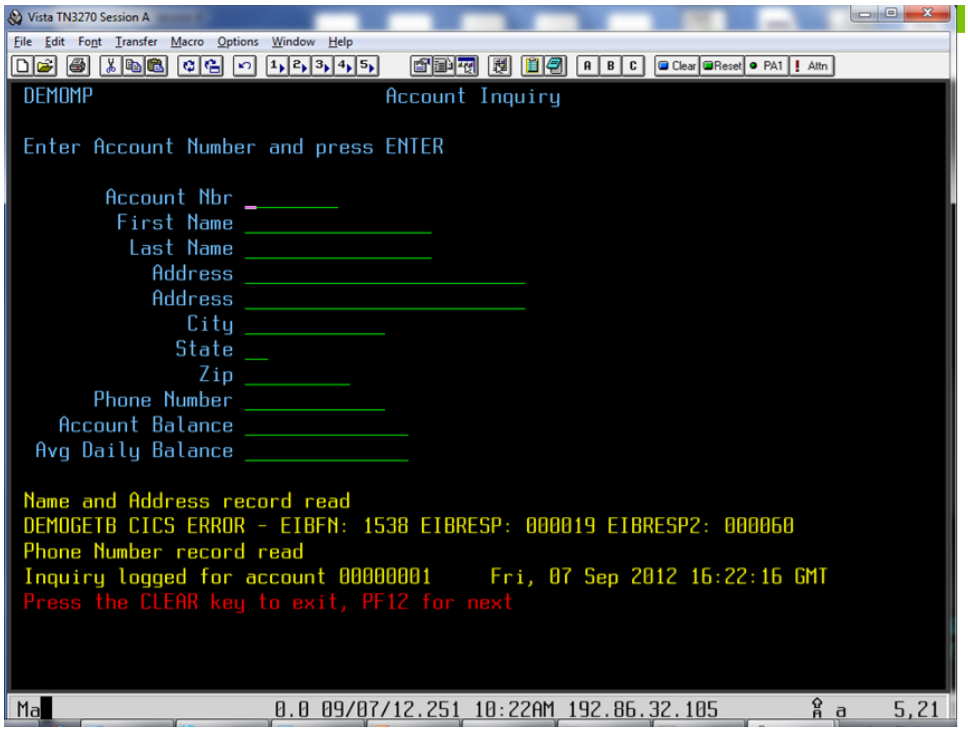


Copybook によって応答も整形されています。 以下の図は、仮想化を使用した場合に返信する応答を示しています。



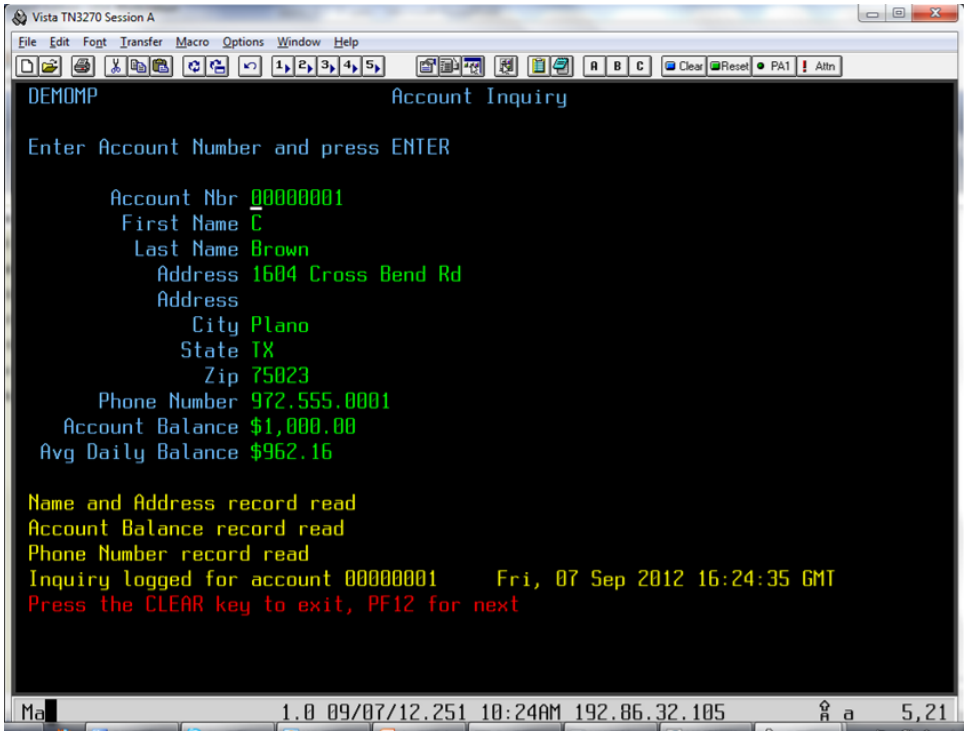
例

この例では、VSAM ファイル DEMOBAL は使用できません。また、プログラム DEMO3 を実行します。 DEMO3 は、DEMOBAL を使用できないために失敗します。



次は、仮想サービスが展開されています。

プログラム DEMO3 を再度実行すると、引き続き VSAM ファイルを使用できないにもかかわらず機能します。

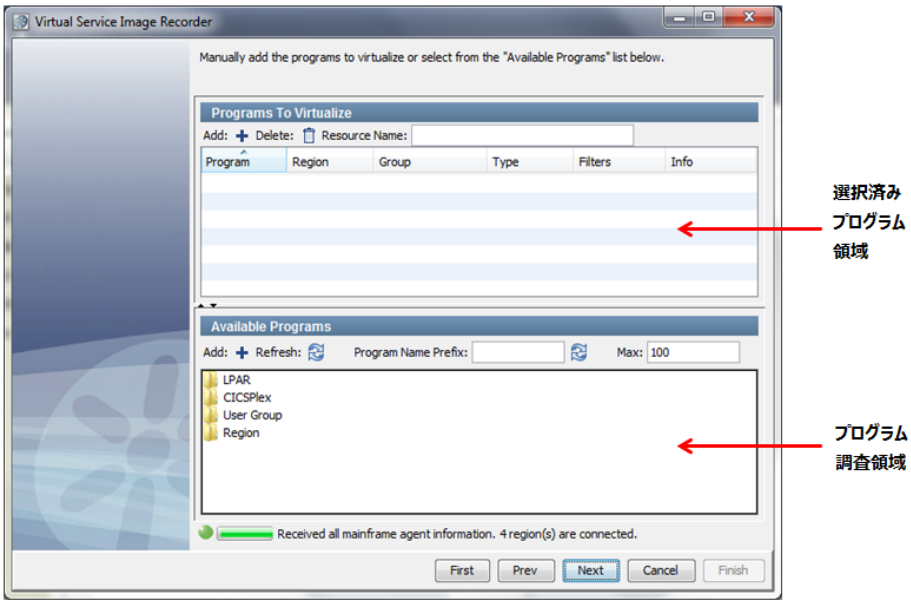

Vista TN3270 Session A
File Edit Font Transfer Macro Options Window Help
DEMOMP Account Inquiry
Enter Account Number and press ENTER
Account Nbr 00000001
First Name C
Last Name Brown
Address 1604 Cross Bend Rd
Address
City Plano
State TX
Zip 75023
Phone Number 972.555.0001
Account Balance $1,000.00
Avg Daily Balance $962.16
Name and Address record read
Account Balance record read
Phone Number record read
Inquiry logged for account 00000001 Fri, 07 Sep 2012 16:24:35 GMT
Press the CLEAR key to exit, PF12 for next
1.0 09/07/12.251 10:24AM 192.86.32.105
5,21

CICS の［仮想化するプログラム］パネル

CICS の［仮想化するプログラム］パネルを使用して、以下を実行します。

- DevTest LPAR エージェントに接続する CICS 領域の調査
- これらの CICS 領域に対して定義されているプログラムの調査
- 仮想化するプログラムの選択
- 仮想化するプログラムの手動での追加
- フィルタの設定による、仮想化する CICS LINK ステートメント、トランザクション、および CICS ユーザの絞り込み
- CICS 領域のグループで CICS プログラムを仮想化するためのグループメンバシップの確立

以下の図は、CICS の［仮想化するプログラム］パネルの主な領域を示しています。



利用可能なプログラム

［利用可能なプログラム］領域では、DevTest CICS エージェントを使用して DevTest に接続された CICS 領域を調査して、そこに定義されているプログラムを参照できます。

以下によって、既知の CICS 領域およびプログラムを調査できます。

- LPAR （z/OS の論理パーティション）
- CICSPlex
- ユーザ定義のグループ
- 領域

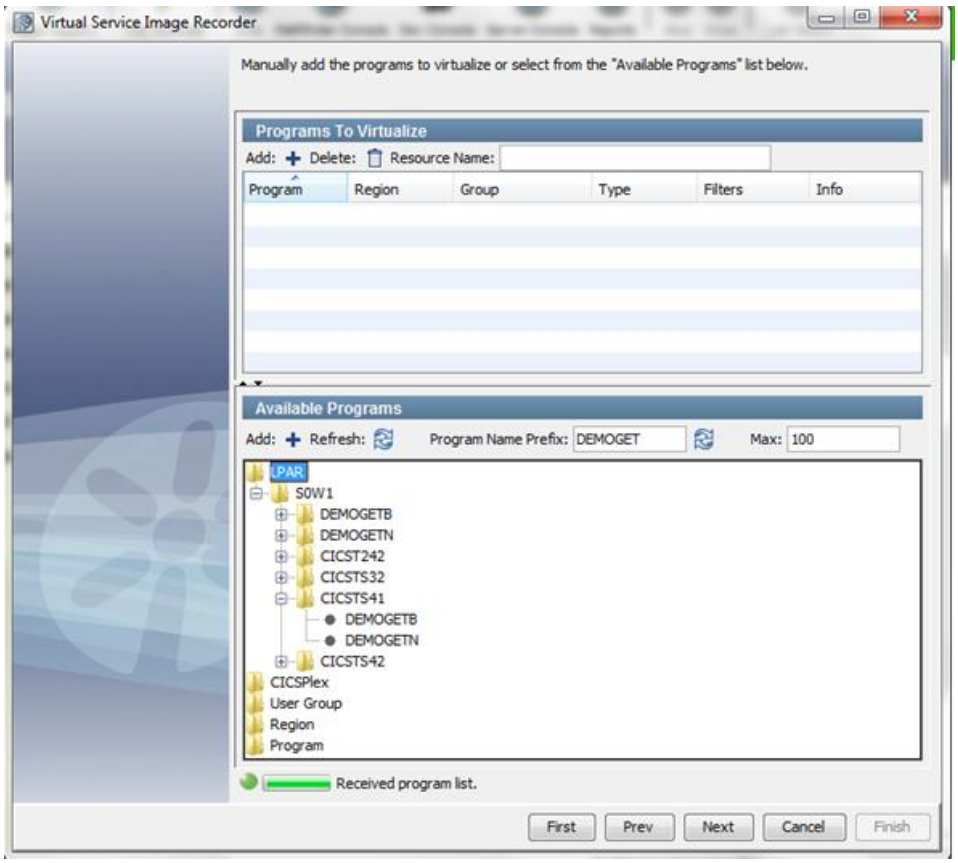
VTAM APPLID （アプリケーション ID）は CICS 領域を識別します。

プログラム名プレフィックス

表示するプログラムをフィルタするためにレコーダが使用するプレフィックスを定義します。 ［利用可能なプログラム］領域は、指定されたプレフィックスから始まるプログラムのみを表示します。 常に、プログラムのプレフィックスを追加します。

最大

1 つの CICS 領域から表示するプログラムの最大数を定義します。

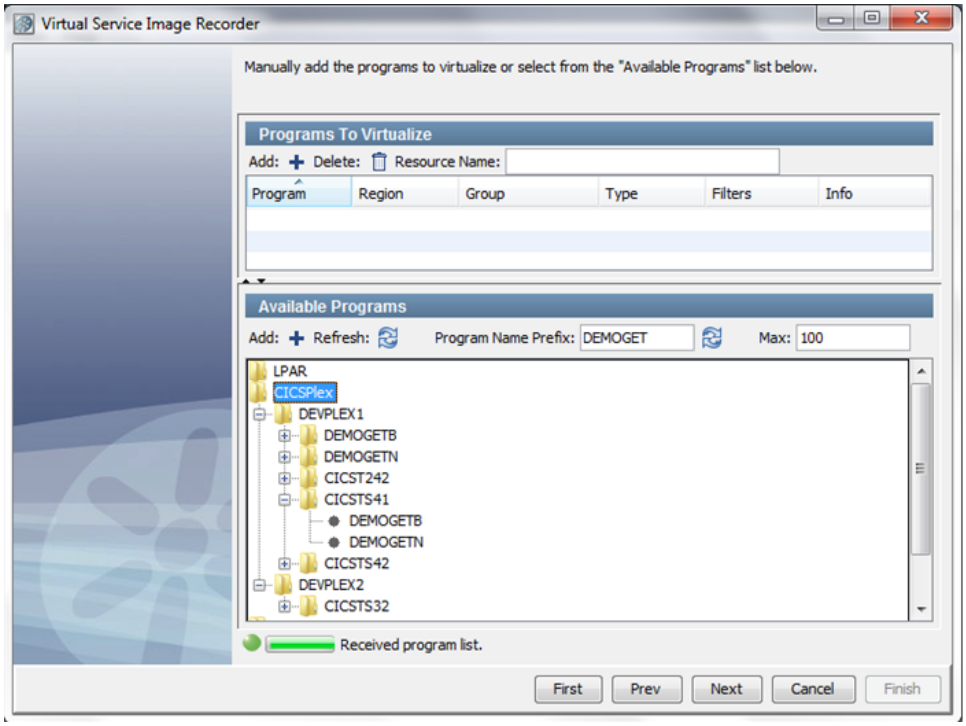


LPAR をクリックすると、既知の LPAR が表示されます（この場合は S0W1）。

S0W1 の横のプラス記号をクリックすると、SOW1 の既知の CICS 領域である CICST242、CICSTS32、CICSTS41、および CICSTS42 が表示されます。

CICSTS41 の横のプラス記号をクリックすると、CICSTS41 のプレフィックス DEMOGET に一致する既知のプログラム（DEMOGETB と DEMOGETN）が表示されます。

DEMOGETB と DEMOGETN は LPAR で検出されたプログラムであるため、SOW1 の下のリストにも表示されます。



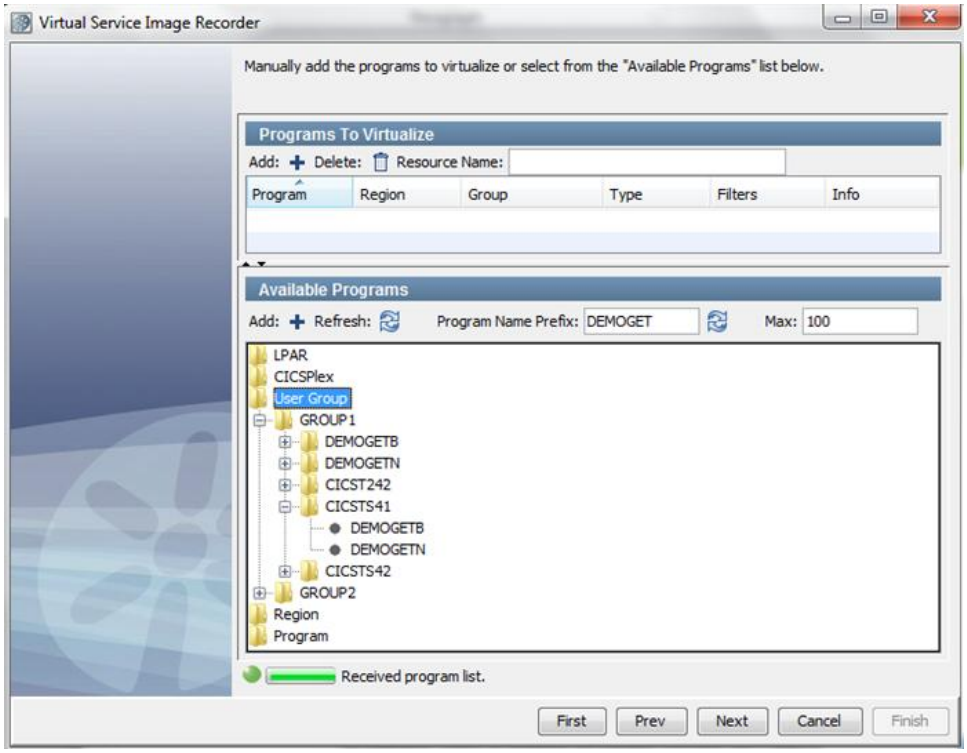
CICSPlex をクリックすると、既知の CICSPlex が表示され、DEVPLEX1 と DEVPLEX2 が表示されます。

DEVPLEX1 の横のプラス記号をクリックすると、既知の CICS 領域である CICST242、CICSTS41、および CICSTS42 が表示されます。

CICSTS41 の横のプラス記号をクリックすると、CICSTS41 のプレフィックスに一致する既知のプログラム（DEMOGETB と DEMOGETN）が表示されます。

DEMOGETB と DEMOGETN は CICSPlex で検出されたプログラムであるため、DEBPLEX1 の下のリストにも表示されます。

注: CICS 領域を CICSPlex グループのメンバにするためには、DevTest CICS エージェントのユーザ出口を変更する必要があります。 詳細については、「エージェント」の「CICS エージェントのユーザ出口」を参照してください。



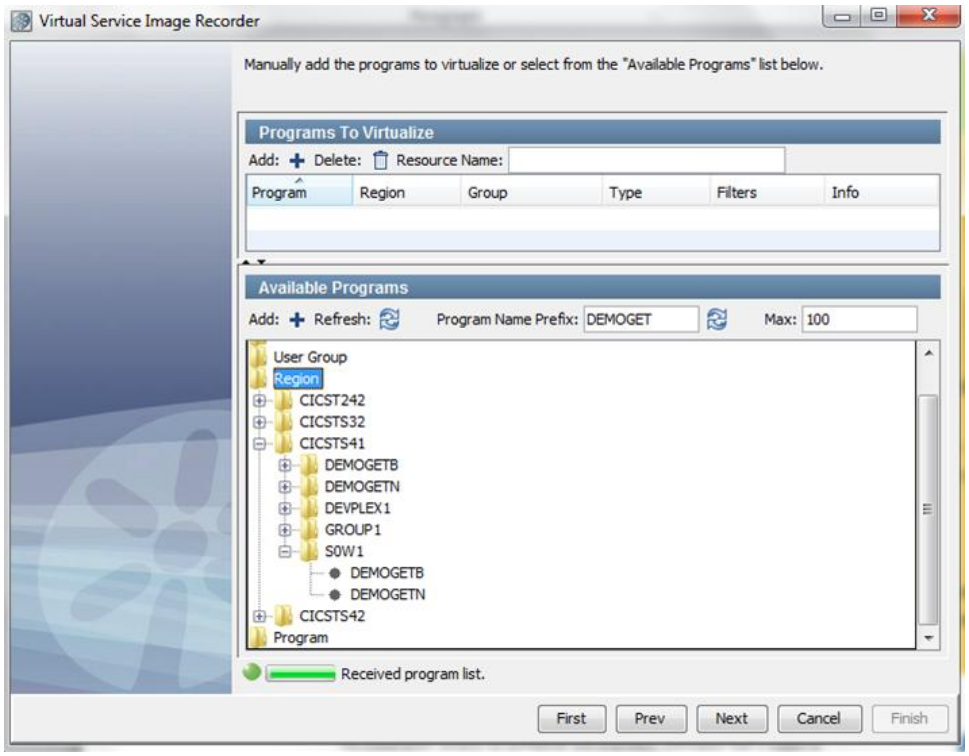
User Group をクリックすると、既知のユーザ グループが表示され、GROUP1 と GROUP2 が表示されます。

GROUP1 の横のプラス記号をクリックすると、既知の CICS 領域である CICST242、CICSTS41、および CICSTS42 が表示されます。

CICSTS41 の横のプラス記号をクリックすると、CICSTS41 のプレフィックスに一致する既知のプログラム (DEMOGETB と DEMOGETN) が表示されます。

DEMOGETB と DEMOGETN はユーザ グループで検出されたプログラムであるため、GROUP1 の下のリストにも表示されます。

注: CICS 領域をユーザ グループのメンバにするためには、DevTest CICS エージェントのユーザ出口を変更する必要があります。 詳細については、「エージェント」の「CICS エージェントのユーザ出口」を参照してください。

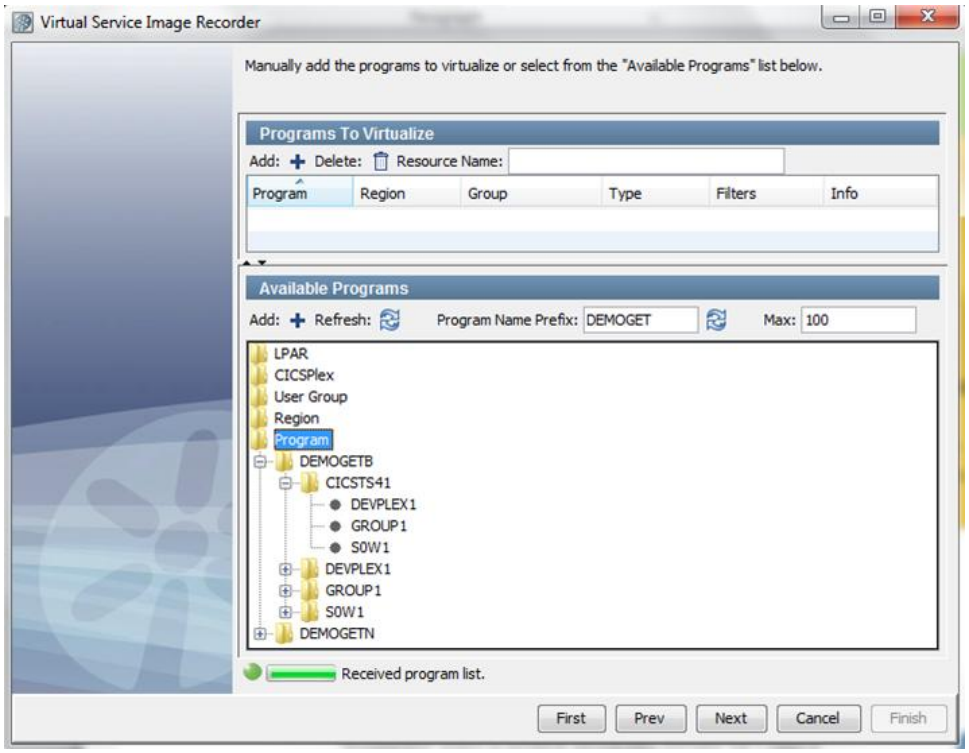


Region をクリックすると、既知の CICS 領域が表示され、CICST242、CICSTS32、CICSTS41、および CICSTS42 が表示されます。

CICSTS41 の横のプラス記号をクリックすると、この CICS 領域を含むグループが表示されます。

S0W1 （この領域がメンバである LPAR グループ）の横のプラス記号をクリックすると、CICSTS41 のプレフィックスに一致する既知のプログラム（DEMOGETB と DEMOGETN）が表示されます。

DEMOGETB と DEMOGETN はユーザ グループで検出されたプログラムであるため、GROUP1 の下のリストにも表示されます。



CICS 領域をクリックすることによってプログラム情報が取得されると、Program フォルダにグループが表示されます。

Program をクリックすると、既知の CICS プログラムがすべて表示されます。この場合は、DEMOGETB と DEMOGETN が表示されています。

DEMOGETB の横のプラス記号をクリックすると、DEMOGETB を含む既知の CICS 領域およびグループが展開されます。

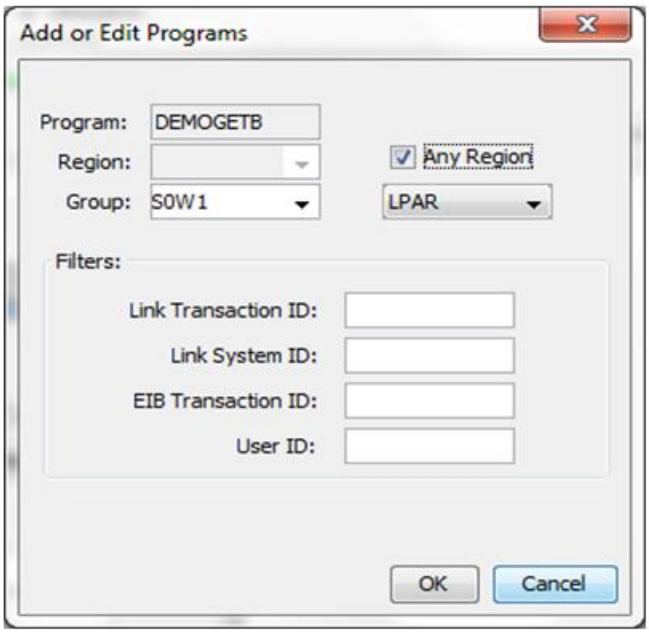
仮想化するプログラム

プログラムを追加するには、ダブルクリックして［仮想化するプログラム］パネルに表示します。 または、［仮想化するプログラム］パネルで［追加］ をクリックして、プログラムを手動で追加します。

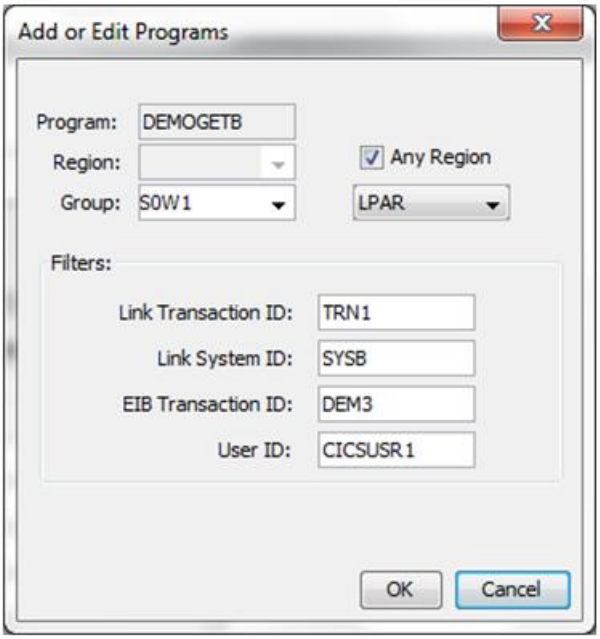
プログラムが［仮想化するプログラム］パネルに表示されたら、ダブルクリックしてフィルタとグループを追加します。

この例では、CICS 領域 CICSTS41 でプログラム DEMOGETB が仮想化されます。［任意の領域］チェック ボックスをオンにしない限り、グループ S0W1 と LPAR は無視されます。

フィルタによって、DEMOGETB への CICS LINK のうち、どれを仮想化するかを指定できます。



［任意の領域］チェック ボックスをオンにすると、［領域］テキスト ボックスがクリアされ、LPAR グループ S0W1 内のすべての CICS 領域（LPAR S0W1 内に存在するすべての CICS 領域）で DEMOGETB が仮想化されることが示されます。



上記の図では、フィルタを使用して仮想化するプログラムを限定しています。

リンクトランザクション ID

TRANSID('TRN1') でコード化された場合、DEMOGETB への CICS LINK のみが仮想化されることを示しています。

リンク システム ID

SYSID('SYSB') でコード化された場合、DEMOGETB への CICS LINK のみが仮想化されることを示しています。

EIB トランザクション ID

トランザクション DEM3 で実行された場合、DEMOGETB への CICS LINK のみが仮想化されることを示しています。

ユーザ ID

ユーザ ID CICSUSR1 で実行された場合、DEMOGETB への CICS LINK のみが仮想化されることを示しています。

CICS トランザクション ゲートウェイ（ECI）イメージの記録

CICS トランザクション ゲートウェイ（ECI）を記録する方法

1. 仮想サービス イメージ レコーダの［基本］タブで、トランスポート プロトコルとして CICS トランザクション ゲートウェイ（ECI）を選択します。
2. ［基本］タブのフィールドに入力し、［次へ］をクリックします。

   レコーダの次の手順が開きます。
3. この手順では、ポートおよびホストの情報を入力します。

   リスン/記録ポート

   クライアントが DevTest と通信するポートを定義します。

   ターゲット ホスト

   サーバが実行されているターゲット ホストの名前または IP アドレスを定義します。

   ターゲット ポート

   サーバがリスンするターゲット ポート番号。

   クライアントから SSL で受信

   選択すると、レコーダは、クライアントが SSL を使用して接続すると想定します。関連するキーストアとパスワードを提供した場合、セキュリティ マテリアル（証明書など）の取得に使用されます。

   サーバに SSL で送信

   選択すると、レコーダは、SSL を使用して実際のシステムに接続します。 関連するキーストアとパスワードを提供した場合、セキュリティ マテリアル（証明書など）の取得に使用されます。

   SSL キーストア ファイル

   キーストア ファイルの名前。

   キーストア パスワード

   キーストア ファイルのパスワード。
4. ［次へ］をクリックして、レコーディング ウィンドウを表示します。
5. CTG サーバと通信するアプリケーションを起動し、記録するアクティビティを実行します。
6. 仮想サービス イメージ レコーダ ウィンドウでトランザクションが記録されたこと確認したら、［次へ］をクリックします。

データ プロトコル ウィンドウが表示されます。

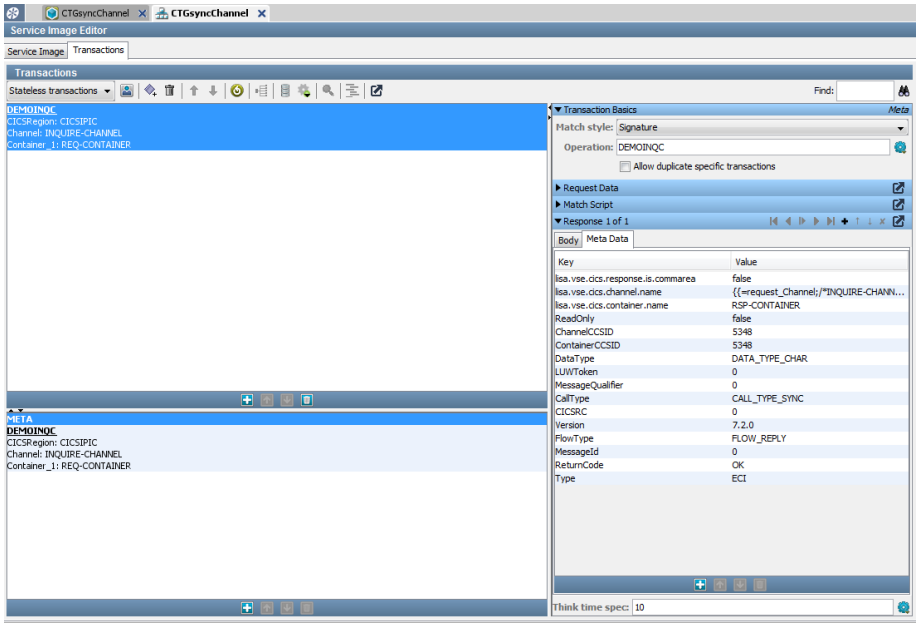
7. ［次へ］をクリックします。

   会話スタータ ウィンドウが表示されます。

8. ［次へ］をクリックします。

9. VSM と VSI を表示するオプションを選択し、［終了］をクリックます。

10. サービス イメージ エディタに CTG サービス イメージが表示されます。



IMS Connect サービス イメージの記録

IMS Connect サービス イメージを記録する方法

1. レコーダを開始する前に、必要に応じて **local.properties** に以下のプロパティを追加します。
   - テスト中のシステムが、応答メッセージの先頭に 4 バイトの LLLL 長フィールドを持つ応答を送信する場合は、以下のプロパティを設定します。

     ```
     lisa.vse.protocol.ims.response.includes.llll=true
     ```

   - テスト中のシステムが、要求メッセージの先頭に 4 バイトの LLZZ 長フィールドを持つ要求を受信する場合は、以下のプロパティを設定します。

     ```
     lisa.vse.ims.connect.llzz.request=true
     ```

   - テスト中のシステムが、応答メッセージの先頭に 4 バイトの LLZZ 長フィールドを持つ応答を送信する場合は、以下のプロパティを設定します。

     ```
     lisa.vse.ims.connect.llzz.response=true
     ```

   - テスト中のシステムがペイロードの解析に Copybook を使用する場合は、以下のプロパティを追加します。

     ```
     lisa.vse.copybook.unknown.passthrough=true
     ```

   これにより、確認応答メッセージ（ペイロード データを含まない）を表すバイナリ応答では、Copybook 処理をバイパスすることができます。
2. 仮想サービス イメージ レコーダの［基本］タブで、トランスポート プロトコルとして IMS Connect を選択します。
3. ［基本］タブのフィールドに入力し、［次へ］をクリックします。

   レコーダの次の手順が開きます。
4. ポートおよびホストの情報を入力します。

   リスン/記録ポート

   クライアントが DevTest と通信するポートを定義します。

   ターゲット ホスト

   サーバが実行されているターゲット ホストの名前または IP アドレスを定義します。

   ターゲット ポート

サーバがリスンするターゲット ポート番号を定義します。 プロキシ パススルー スタイルを選択する場合は、このフィールドを空白のままにします。

デフォルト： 80 （HTTP）および 443 （HTTPS）

IMS 形式ファイル

以下のいずれかを定義します。

- IBM サンプル アプリケーションが使用する 160 バイトのヘッダを使用する IMS Connect 要求をサポートするファイルの名前。
- 80 バイトのヘッダ。

サポートされるこれらの 160 バイトおよび 80 バイト ヘッダのフィールド定義は、LISA_HOME ディレクトリの **ims.format** ファイルで提供されています。

デフォルトで DevTest に含まれている IMS Connect サポートを使用するには、このフィールドを空白のままにします。

文字セット

仮想化されるアプリケーションによって、データがどのようにエンコードされるか定義します。

値

- ASCII - CP1252
- EBCDIC - CP037

5. ［次へ］をクリックして、レコーディング ウィンドウを表示します。
6. IMS Connect サーバと通信するアプリケーションを起動し、記録するアクティビティを実行します。

   このアプリケーションのホストとポートが、レコーダのホストとポート（通常は localhost:8001）であることを確認します。
7. 仮想サービス イメージ レコーダ ウィンドウでトランザクションが記録されたことを確認したら、［次へ］をクリックします。

   データ プロトコル ウィンドウが表示されます。

   デフォルトでは、TransactionCode ヘッダ フィールド（存在する場合）は、要求操作として設定されます。また、TransactionCode、DestinationId、および ClientId の各フィールドが形式ファイル定義に存在する場合、これらが引数として追加されます。
8. 必要に応じて、Copybook データ プロトコルなどのデータ プロトコルを追加し、［次へ］をクリックします。

会話スタータ ウィンドウが表示されます。

9. ［次へ］をクリックします。
10. VSM と VSI を表示するオプションを選択し、［終了］をクリックます。

    サービス イメージ エディタに IMS Connect サービス イメージが表示されます。

IMS Connect のカスタム形式ファイル

テスト中のシステムが DevTest がデフォルトでサポートするものとは異なる要求 ヘッダを使用する場合は、レコーディングの前に以下の手順に従います。

1. DevTest プロジェクトの Data フォルダに、**<custom-format>.format** ファイルを作成します。
2. テキスト エディタでファイルを編集し、以下のテキストを追加します。RequestUserHeader 領域でフィールド定義を指定します。

```
#--------------------------------------------------------------------------------
# The IMS™ request message (IRM) header contains a 28-byte fixed-format
# section that is common to all messages from all IMS Connect client applications
# that communicate with IMS TM.
#--------------------------------------------------------------------------------
RequestHeaderCommon {
    LLLL                        int;                    #total message length IRM + user data
    IRMFixedHeader {
        LL                     short;                #IRM_LEN, total length of the header segment
including user header portion
        ZZ                      byte;                   #IRM_ARCH
        Flag0                   byte;                   #IRM_F0
        UserExitId              string(8);              #IRM_ID
        NakReasonCode           byte(2);                #IRM_NAK_RSNCDE
        Reserved1               byte(2);                #IRM_RES1
        MessageType             byte;                   #IRM_MessageType
        WaitTime                byte;                   #IRM_TIMER
        SocketConnectionType    byte;                   #IRM_SOCT
        EncodingSchema          byte;                   #IRM_ES
        ClientId                byte(8);                #IRM_CLIENTID
    }
}

#--------------------------------------------------------------------------------
# Format of user portion of IRM some custom header
#
# Following the 4-byte length field and the 28-byte fixed portion of the
# IMS™ request message (IRM) header in IMS Connect client input messages,
# user-written client applications can include a user-defined section in the IRM.
#
#--------------------------------------------------------------------------------
RequestUserHeader {
    MyFlag1                     byte;
    MyFlag2                     byte;
    MyFlag3                     byte;
    MyFlag4                     byte;
```

```
    TransactionCode             string(8);
    DestinationId               string(8);
    LogicalTerminal             string(8);
    Miscellaneous               byte(20);
}

#-------------------------------------------------------------------------------
# Format of data segments of request message
#-------------------------------------------------------------------------------
RequestPayloadSegment  {
        LL                      short;
        ZZ                      byte(2);
        Data                    byte(LL:inclusive);
}

#-------------------------------------------------------------------------------
# Format of header for response message. Some responses will have it, some won't
#-------------------------------------------------------------------------------
ResponseMessageHeader {
        LLLL                  int;                # followed by multiple ResponsePayloadSegment
}

#-------------------------------------------------------------------------------
# Format of data segments of request message
#-------------------------------------------------------------------------------
ResponsePayloadSegment {
        LL                      short;
        ZZ                      byte(2);
        Data                    byte(LL:inclusive);
}
```

ファイルを作成して Data フォルダに保存すると、レコーダの［IMS 形式ファイル］フィールドに表示されます。 デフォルト以外の要求 ヘッダを使用するアプリケーションの形式ファイルを選択できます。

JCo 経由の SAP RFC の記録

次の手順に従ってください:

1. 仮想サービス イメージ レコーダの［基本］タブで、トランスポート プロトコルとして JCo 経由の SAP RFC を選択します。
2. ［基本］タブのフィールドに入力し、［次へ］をクリックします。

   レコーダの次の手順が開きます。
3. 接続の詳細フィールドに入力します。

   クライアント システム名

   クライアント SAP システムを識別する一意の名前を指定します。

   クライアント システム接続プロパティ

   クライアント システム接続プロパティ ファイルには、クライアント システム上の送信先に接続するための接続プロパティが含まれます。これはプロジェクトの Data ディレクトリの .properties ファイルである必要があり、.jcoServer ファイルで通常見つかるプロパティが含まれます。 このファイルでは、jco.server.repository_destination を指定しないでください。 サポートされているプロパティの詳細については、インストール ディレクトリの **doc** フォルダにある **com.sap.conn.jco.ext.ServerDataProvider** の JavaDoc を参照してください。

   RFC リポジトリ名

   RFC テンプレート用のリポジトリがある SAP システムを識別する一意の名前を指定します。 この名前は、多くの場合、送信先システムと同じです。

   RFC リポジトリ接続プロパティ

   リポジトリがあるシステムに接続するための接続プロパティが含まれるリポジトリ接続プロパティ ファイルを定義します。 これは、プロジェクトの Data ディレクトリの .properties ファイルである必要があり、.jcoDestination ファイルで通常見つかるプロパティが含まれます。サポートされているプロパティの詳細については、インストール ディレクトリの **doc** フォルダにある **com.sap.conn.jco.ext.DestinationDataProvider** の JavaDoc を参照してください。

   送信先システム名

   RFC が実行される SAP システムを識別する一意の名前を定義します。これは多くの場合、リポジトリ名と同じです。

送信先システム接続プロパティ

リポジトリがあるシステムに接続するための接続プロパティが含まれる送信先システム接続プロパティ ファイルを指定します。 これは、プロジェクトの Data ディレクトリの .properties ファイルである必要があり、.jcoDestination ファイルで通常見つかるプロパティが含まれます。サポートされているプロパティの詳細については、インストール ディレクトリの **doc** フォルダにある **com.sap.conn.jco.ext.DestinationDataProvider** の JavaDoc を参照してください。これは、リポジトリ接続プロパティと同じファイルにすることができます。

関数名

DevTest がインターセプトする必要がある RFC の名前を指定します。

4. ［次へ］をクリックして、レコーディング ウィンドウを表示します。
5. SAP サーバと通信するアプリケーションを起動し、記録するアクティビティを実行します。
6. 仮想サービス イメージ レコーダ ウィンドウでトランザクションが記録されたこと確認したら、［次へ］をクリックします。

   データ プロトコル ウィンドウが表示されます。
7. ［次へ］をクリックします。

   会話スタータ ウィンドウが表示されます。
8. ［次へ］をクリックします。
9. VSM と VSI を表示するオプションを選択し、［終了］をクリックます。

   サービス イメージ エディタに SAP RFC サービス イメージが表示されます。

JCo IDoc の記録

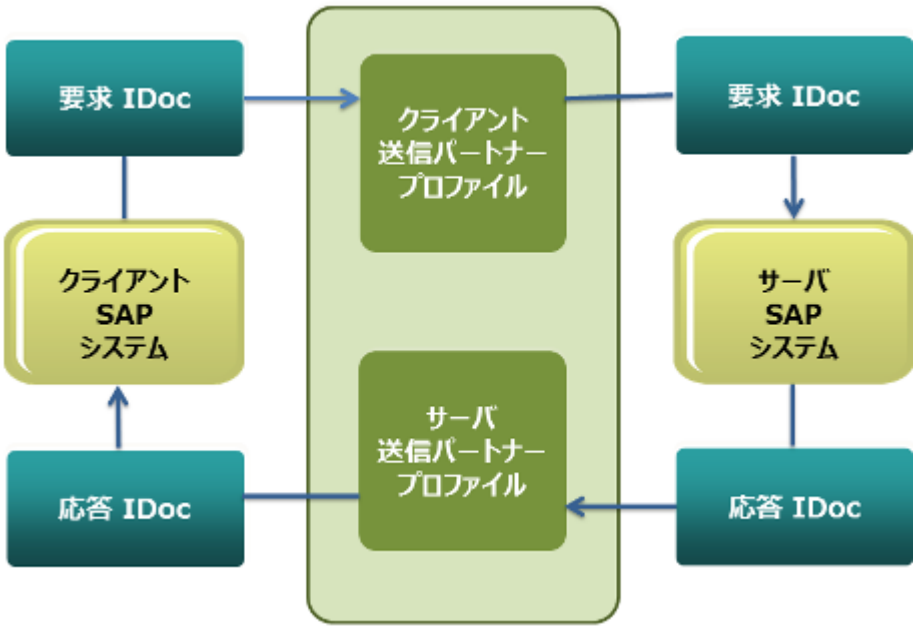
SAP JCo IDoc トランスポート プロトコルは、SAP JCo IDoc プロトコルを使用する仮想サービスを作成することを可能にします。

JCo IDoc プロトコルは、RFC 送信先を使用して、ある SAP システムから別の SAP システムに送信された IDoc を記録および再生する仮想サービスの作成をサポートします。

通信を開始して IDoc を送信する SAP システムは、クライアント SAP システムと呼ばれます。 送信された IDoc は、要求 IDoc と呼ばれます。

要求 IDoc を受信する SAP システムは、サーバ SAP システムと呼ばれます。サーバ SAP システムが返す IDoc は、応答 IDoc と呼ばれます。

以下の図は、2 つの SAP システムの間の IDoc の標準フローを示しています。



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。

JCo IDoc サービス イメージを記録する方法

1. 仮想サービス イメージ レコーダの［基本］タブで、トランスポート プロトコルとして JCo IDoc プロトコルを選択します。
2. ［基本］タブのフィールドに入力し、［次へ］をクリックします。

レコーダの次の手順が開きます。

3. 接続の詳細フィールドに入力します。

クライアント RFC 接続プロパティ

SAP ゲートウェイに対してプログラム ID で登録して IDoc を受信するために VSE が 使用する接続プロパティが含まれるクライアント RFC 接続プロパティ ファイルを定義します。プロパティは、.jcoServer ファイルで指定されたものと同じである必要があります。

クライアント RFC 送信先名

RFC 送信先を識別する一意の名前を指定します。

クライアント システム接続プロパティ

IDoc をクライアント SAP システムに返すための接続プロパティが含まれるクライアント システム接続プロパティ ファイルを指定します。これらのプロパティは、クライアント SAP システムに接続するために使用できる .jcoDestination ファイルで指定されたものと同じである必要があります。

クライアント システム名

クライアント SAP システムを識別する一意の名前を指定します。

要求識別子の XPath 式

識別子を生成するために要求 IDoc XML と一緒にプロトコルが使用する XPath 式を指定します。 要求識別子の XPath 式は、単一の XPath 式にすることができます。 この識別子は、要求 IDoc と応答 IDoc を関連付けるために使用されます。また、XPath 式は、XPath 式のカンマ区切りリストにすることもできます。その場合、複数の式からの結果の値は連結され（ダッシュで区切られ）、単一の識別子として使用されます。

サーバ RFC 接続プロパティ

SAP ゲートウェイに対してプログラム ID で登録して IDoc を受信するために VSE が 使用する接続プロパティが含まれるプロパティ ファイルを指定します。プロパティは、サーバ SAP システムから IDoc を受信する JCo サーバ プログラムを起動するために .jcoServer ファイルで指定されたものと同じである必要があります。

サーバ RFC 接続先名

サーバ RFC 接続先を識別する一意の名前を指定します。

サーバ システム接続プロパティ

サーバ システム接続プロパティ ファイルには、クライアント SAP システムに IDoc を送信するための接続プロパティが含まれます。これらのプロパティは、SAP サーバ システムに接続するために使用できる .jcoDestination ファイルで指定されたものと同じである必要があります。

サーバ システム名

サーバ SAP システムを識別する一意の名前。

応答識別子の XPath 式

識別子を生成するために応答 IDoc XML と一緒にプロトコルが使用する XPath 式を指定します。
応答識別子の XPath 式は、単一の XPath 式にすることができます。 この識別子は、要求 IDoc と以前に受信した応答 IDoc を関連付けるために使用されます。 また、XPath 式は、XPath 式のカンマ区切りリストにすることもできます。その場合、複数の式からの結果の値は連結され（ダッシュで区切られ）、単一の識別子として使用されます。

4. ［次へ］をクリックして、レコーディング ウィンドウを開きます。
5. SAP サーバと通信するアプリケーションを起動し、記録するアクティビティを実行します。
6. 仮想サービス イメージ レコーダ ウィンドウでトランザクションが記録されたこと確認したら、［次へ］をクリックします。

   データ プロトコル ウィンドウが表示されます。

   VSE は XML として IDoc を格納するため、XML データ プロトコルが要求側のデータ プロトコルに選択されています。 XML データ プロトコルは JCo IDoc プロトコルが設定する操作名を上書きするため、XML データ プロトコルを削除します。
7. ［次へ］をクリックします。

   会話スタータ ウィンドウが表示されます。
8. ［次へ］をクリックします。
9. VSM と VSI を表示するオプションを選択し、［終了］をクリックます。

   サービス イメージ エディタに SAP JCo IDoc サービス イメージが表示されます。

不透明データ処理

不透明データ処理（ODP）を使用すると、要求および応答の形式が不明な場合に、CA Service Virtualization でデータを十分詳細に仮想化することができます。 ODP により、新しいメッセージ フォーマットに遭遇するたびに新しいデータ ハンドラを必要とすることがなくなります。

要求および対応する応答を記録して、CA Service Virtualization がメッセージ構造を推定できます。 これは、CA Service Virtualization が、要求のバイトを対応する応答のバイトに関連付けて、その他のプロトコルで利用可能であるものと同じ「マジック ストリング」動作を提供することができることを意味します。また、CA Service Virtualization は、仮想サービスを再生したときに遭遇する、新しい要求にインテリジェントに一致する要求構造を十分に理解することもできます。

水面下では、ODP により、特許取得済みのアルゴリズムを使用して、受信要求が、記録された ODP サービス イメージ内のすべての要求と比較されます。最も一致する要求が選択され、動的なマジック ストリングの置換が実行された後、対応する応答が返されます。

ODP の一致アルゴリズムでは、一致プロセス中にエントロピーによるウェイトが適用されます。エントロピー重み付けプロセスにより、メッセージ内のどのバイトがより重要かが推定されます。 たとえば、操作タイプに対応するバイトは、残りのペイロードと比較して、一致プロセス中の重要性が高くなります。記録されたメッセージの多数（100 以上）のサンプルと、パラメータ値の多様なサンプリングを使用すると、エントロピー重み付けプロセスが最適に機能します。

ODP 一致および応答アルゴリズムは、最適な結果が取得されたバイナリおよびテキストのメッセージ プロトコルの両方で機能することが示されました。固定幅フィールドを使用するプロトコル（IMS など）または区切りフィールドを使用するプロトコル（XML ベースのプロトコルなど）で、最適な結果が得られました。長さのエンコードを使用するプロトコル形式（たとえば ASN.1）でも妥当な正確さが得られましたが、この点の改良が最大の課題となっています。

ODP では、RAW TCP/IP ソケットを使用してキャプチャできるトラフィックがサポートされています。 データを PCAP ファイルからインポートすることもできます。

ODP の使用方法

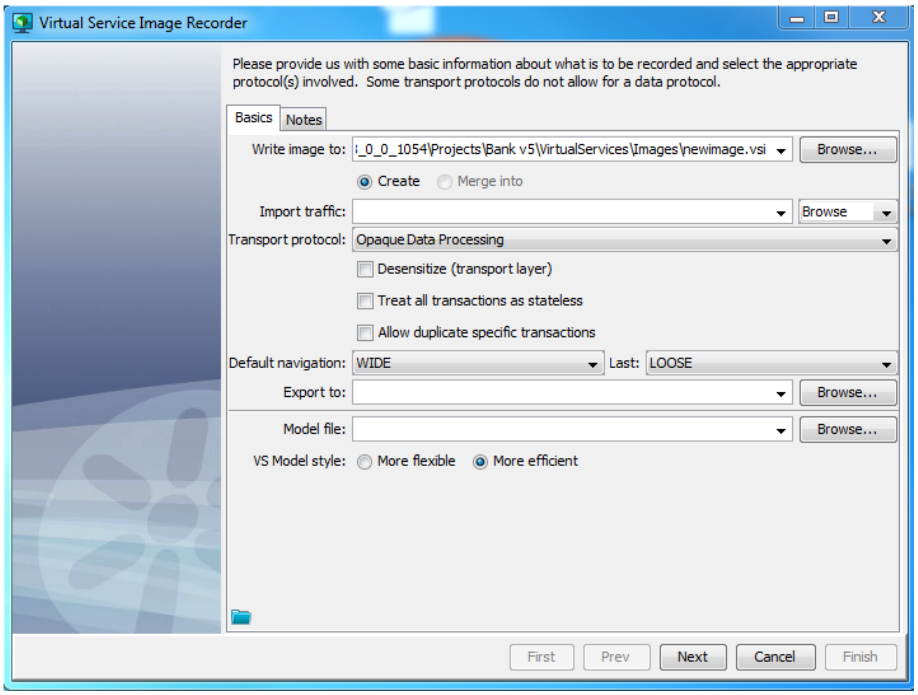
ODP を使用して仮想サービス イメージをレコーディングするには、以下の手順を使用します。

前提条件と準備手順の詳細については、「TCP の仮想化 (P. 196)」を参照してください。

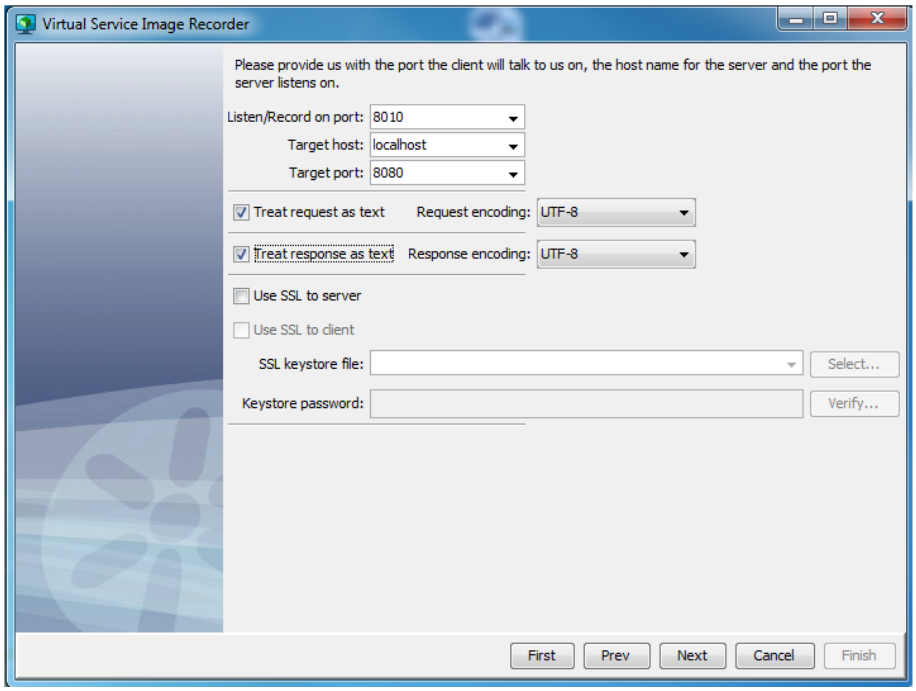
次の手順に従ってください:

1. DevTest ワークステーション を開きます。
2. 新しい仮想サービス イメージのレコーディングを開始するには、以下のいずれかの手順に従います。
   - メイン ツールバーの［VSE レコーダ］  をクリックします。
   - プロジェクト パネルの VirtualServices ノードを右クリックし、［新規 VS イメージの作成］-［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 レコーダが開きます。
3. 以下の図に示すように、［基本 (P. 140)］タブに入力します。



4. ［次へ］をクリックします。
5. クライアントとサーバの両方に情報を入力し、エンコーディングを選択して、SSL パラメータを追加します。



リスン/記録ポート

クライアントが DevTest と通信するポートを定義します。

ターゲット ホスト

サーバが実行されているターゲット ホストの名前または IP アドレスを定義します。

ターゲット ポート

サーバがリスンしているポート番号を定義します。

要求をテキストとして処理

要求がテキストとして処理されるかどうかを指定します。 詳細については、「TCP の仮想化 (P. 196)」を参照してください。

Request Encoding

DevTest ワークステーション が実行されているマシンで使用可能な要求エンコーディングのリストを示します。 デフォルトは UTF8 です。

応答をテキストとして処理

応答がテキストとして処理されるかどうかを指定します。 詳細については、「TCP の仮想化 (P. 196)」を参照してください。

Response Encoding

DevTest ワークステーション が実行されているマシンで使用可能な応答エンコーディングのリストを示します。 デフォルトは UTF8 です。

サーバに SSL を使用

DevTest がサーバに要求を送信するために HTTPS を使用するかどうかを指定します。

- **オン：** DevTest は HTTPS （セキュア レイヤ）要求をサーバに送信します。

  ［サーバに SSL を使用］を選択した場合で、［クライアントに SSL を使用］を選択しない場合には、DevTest はレコーディングに HTTP 接続を使用します。 その後、DevTest は HTTPS を使用して、サーバにそれらの要求を送信します。

- **オフ：** DevTest は HTTP 要求をサーバに送信します。

クライアントに SSL を使用

クライアントからの SSL 要求を再生するためにカスタム キーストアを使用するかどうかを指定します。 このオプションは、［サーバに SSL を使用］が選択されている場合のみ有効です。

値

- **オン：** カスタム クライアント キーストアおよびパスフレーズを指定できます。これらのパラメータを入力した場合、ハードコードされたデフォルトの代わりに使用されます。
- **オフ：** カスタム クライアント キーストアおよびパスフレーズを指定できません。

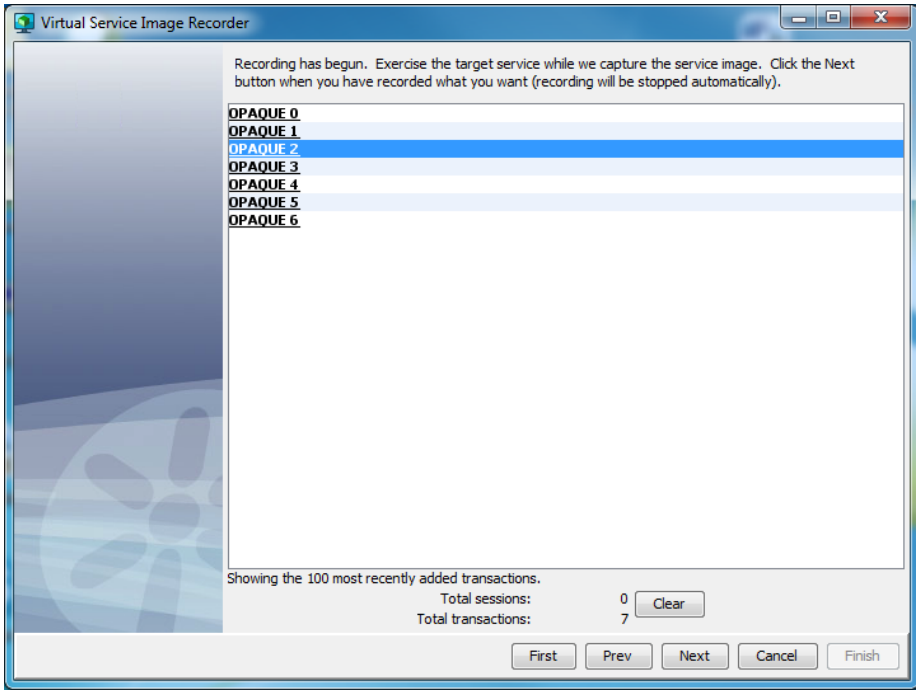
SSL キーストア ファイル

キーストア ファイルの名前。

キーストア パスワード

キーストア ファイルのパスワード。

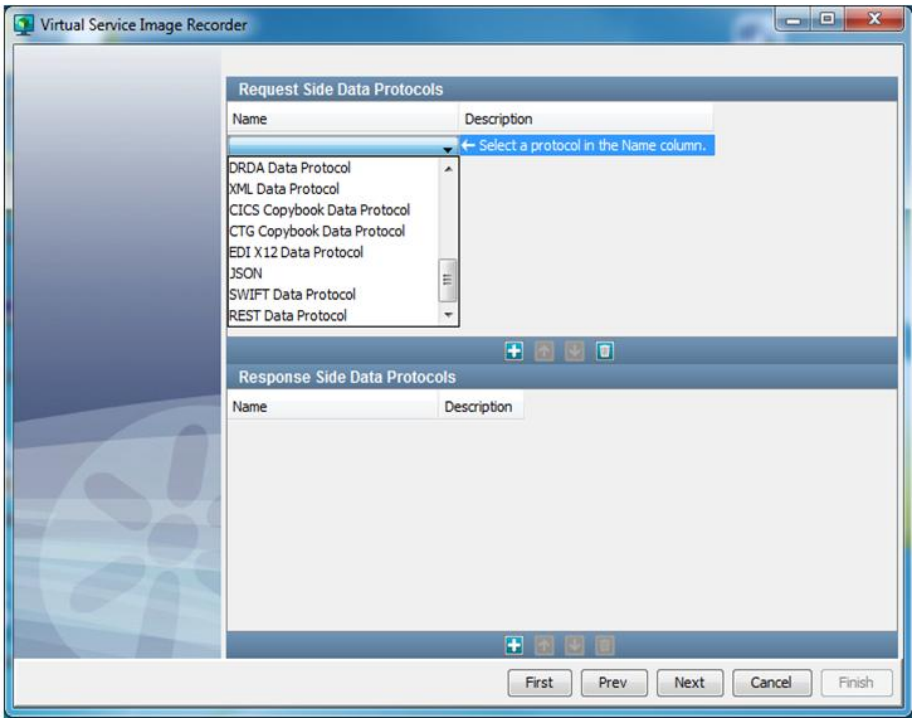
6. ［次へ］をクリックして、レコーディングを開始します。



7. レコーディングが完了したら、［次へ］をクリックします。

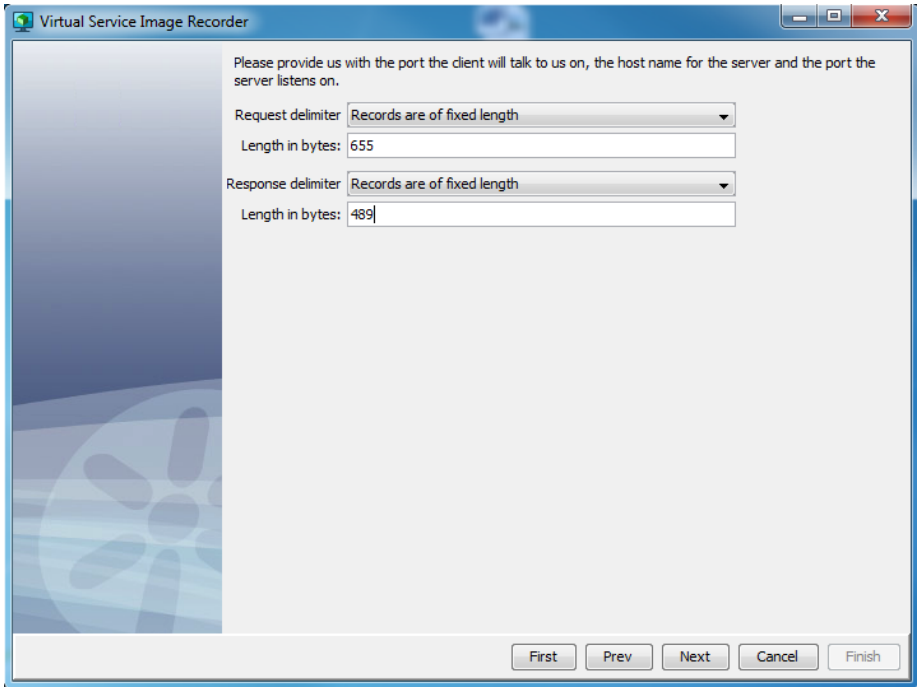
   ODP は、一致では、操作や引数ではなく要求ボディに依存します。 そのため、ほとんどの要求データ プロトコルは適切ではありません。

   一方、応答が暗号化、圧縮、またはそれ以外の形でエンコードされている場合、応答データ プロトコルは適切です。



8. レコーダは、完全な要求または応答を読み取った場合に DevTest に伝えるメッセージ区切り文字の検出を試みます。 この画面でこれらの区切り文字を確認または修正します。

   注: 要求区切り文字は必須です。 ライブ呼び出しを使用するには、応答区切り文字を選択する必要があります。



9. レコーディング後、要求と応答のボディがテキストとしてマークされている場合、レコーダはそれが実際にテキストであるかどうかを検証します。テキストではない場合、タイプがバイナリに切り替わります。

VSE レコーディングでの SSL の使用

VSE レコーディング中、SSL サーバ アプリケーションはテスト中のシステムです。

サーバに SSL を使用

レコーダのセットアップ時に［サーバに SSL を使用］のみが選択されている場合（［クライアントに SSL を使用］は選択されていない場合）、クライアントは DevTest にプレーン HTTP 接続を送信しますが、レコーダはサーバ アプリケーションに HTTPS （SSL ：Secure Sockets Layer）要求を送信します。この場合、*クライアント*とは、要求を送信するアプリケーションまたはテスト ケースを示します。

サーバ証明書を認証するには（または、一方向認証を提供するには）、SSL サーバ アプリケーションの証明書の公開鍵が、DevTest トラスト ストアに格納されている必要があります。

SSL サーバがクライアント認証（双方向認証）を要求する場合、LISA v6.0.8 以降では、**ssl.client.cert.path** で識別されたキーストアを使用して、レコーダから SSL サーバ アプリケーションにクライアント証明書を送信します。［サーバに SSL を使用］が選択されている場合、レコーダは、双方向クライアント認証を完了するためにクライアントをシミュレートします。

サーバおよびクライアントに SSL を使用

記録するために、クライアントまたはテスト ケースは、設定されたリスン/記録ポートに送信します。 レコーダのターゲット ホストは実際の SSL サーバで、ターゲット ポートは、SSL サーバが使用する SSL ポート（通常、443 または 8443）です。

レコーディング中に、SSL ハンドシェイクが、クライアント（レコーダ）と SSL サーバの間で行われます。サーバはその証明書を送信し、DevTest はそれを認証します。 サーバがクライアント認証を要求する場合、**local.properties** にある証明書が使用されます。 有効なキーストアが **ssl.client.cert.path** になく、サーバがクライアント認証を要求する場合、クライアント レコーダの証明書がサーバに返されないため、bad_certificate 状態が返されます。

レコーディング中に、SSL ハンドシェイクが、クライアント（アプリケーションまたはテスト ケース）とサーバ（レコーダ）の間で行われます。レコーダは、［クライアントに SSL を使用］キーストアで指定された証明書を送信します。［SSL キーストア ファイル］が空白の場合、デフォルトのキーストア（{LISA_HOME]}¥webreckeys.ks）が使用されます。レコーダ サーバはクライアント認証を要求しません。 そのハンドシェイクは一方向認証です。

VSM の再生

［クライアントに SSL を使用］を選択した場合、SSL ハンドシェイクが、クライアント アプリケーションまたはテスト ケース クライアントと VSM の間で行われます。VSM リスン ステップで提供されるキーストアは、サーバ証明書として一方向認証に使用されます。 クライアントと VSM の間では、SSL ハンドシェイクは行われません。 そのハンドシェイクは通常の HTTP です。

ライブ呼び出しステップが実行された場合、SSL ハンドシェイクは VSM クライアントと実際のサーバの間で行われます。 実際のサーバがクライアント認証を要求する場合、HTTP/S プロトコル ライブ呼び出しステップのキーストアが使用されます。

キーストアに複数の証明書が含まれる場合、VSE は最初のものを使用します。

VSEasy を使用した仮想サービスの作成と展開

VSEasy コンソールを使用して、仮想サービスを迅速に作成および展開できます。

VSEasy は、以下のモードをサポートします。

- HTTP プロトコル
- VRS ファイルから

VSEasy を使用して、SOAP、JSON、XML、または HTML ペイロードで HTTP のステートレス仮想サービスを作成できます。

非 HTTP 仮想サービスを設定するには、VRS ファイルを使用します。

VRS ファイルを使用する場合、データ プロトコルを組み合わせることができます。 たとえば、サービス イメージを変更するために、ジェネリック XML ペイロード パーサ データ プロトコルまたは要求データ マネージャ データ プロトコルを使用できます。

VRS ファイルは外部データを参照できません。たとえば、VRS ファイルは、Copybook や署名用の証明書を参照できません。

VSEasy コンソールで、仮想サービスを仮想サービス グループに割り当てる必要があります。 VSE サーバを選択すると、利用可能なグループが［**Service Group**］リストに表示されます。デフォルト グループは「**VSEasy**」と命名されます。 デフォルト グループの名前を変更するには、**lisa.properties** ファイル内の **lisa.vseasy.default.group.tag** プロパティを更新します。

注: VSE サーバですでに展開している仮想サービスと同じ名前およびポートを使用して仮想サービスを展開すると、既存の仮想サービスは上書きされます。

以下のブラウザがサポートされています。

- Internet Explorer 10 または 11
- Mozilla Firefox 24
- Google Chrome
- Safari 5

次の手順に従ってください:

1. VSE サーバが実行されていることを確認します。
2. DevTest ワークステーション から、ツールバーの［サーバ コンソール］アイコンを選択します。
3. ［サーバ コンソール］から、［VSEasy］を選択します。

   VSEasy コンソールが開きます。
4. 右側のパネルの［ヘルプ］内の手順に従います。 また、［ヘルプ］には以下のサンプル ファイルが含まれます。
   - HTTP 要求/応答ペア
   - JMS 要求/応答ペアおよび VRS ファイル

VSM の使用

仮想サービス モデル（VSM）は、仮想化されたサービスのエンドポイントになる特殊なタイプのテスト ケースです。仮想サービス イメージ レコーダを起動するには、VSM を作成します。

VSM の作成

次の手順に従ってください:

1. 既存のプロジェクトを開くか、またはプロジェクトを作成します。

   プロジェクトが開き、左側のプロジェクト パネルに project フォルダとそのサブフォルダが表示されます。
2. VirtualServices サブフォルダ ノードを右クリックし、［新規 VS モデルの作成］を選択します。

   別のフォルダに VS モデルを作成することもできますが、VirtualServices フォルダに作成することをお勧めします。

   VS モデル エディタ ウィンドウが開きます。
3. 新規 VSM の場所を参照します。
4. ［ファイル名］フィールドに、VSM の一意の名前を入力します。 拡張子は **.vsm** です。
5. ［保存］をクリックします。

   新規 VSM が開きます。

   VSM が開いている場合、［コマンド］メニューには、VSM に関連するメニュー項目が表示されます。

VSM を開く

次の手順に従ってください:

1. ツールバーの［開く］をクリックするか、または［クイック スタート］ウィンドウの［最近の使用履歴］プロジェクト リストからプロジェクトを選択します。
2. プロジェクト パネルで、そのフォルダから仮想サービス モデルを選択します。

   仮想サービスモデルは、通常、VServices フォルダまたは VirtualServices フォルダに存在します。

データ プロトコルの使用

*データ プロトコル*は、*データ ハンドラ*または*データ プロトコル ハンドラ*とも呼ばれます。 データ プロトコルは要求の解析を行います。 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。そのため、データ プロトコルは、要求のペイロードを知っている必要があります。1 つのトランスポート プロトコルに、複数のデータ プロトコルが存在する場合があります。 一部のトランスポート プロトコル（JDBC など）はデータ プロトコルを許可しません。

レコーディング中に、すべてのトランスポート プロトコルは、要求/応答ペイロードおよび要求側/応答側のデフォルトの適切なデータ プロトコルを検出しようとします。 指定されたデータ プロトコル ハンドラは、以下のペイロードに対して追加されます。

- **SOAP：** Web サービス（SOAP）データ プロトコル ハンドラ
- **XML：** XML データ プロトコル ハンドラ
- **HTML：** データ プロトコル ハンドラなし
- **署名 SOAP**：要求側の WS-Security 要求および応答側の WS-Security 応答データ プロトコル ハンドラ
- **暗号化 SOAP**：要求側の WS-Security 要求および応答側の WS-Security 応答データ プロトコル ハンドラ
- **SOAP セキュリティ**： 要求側の WS-Security 要求および応答側の WS-Security 応答データ プロトコル ハンドラ

さらにデータ プロトコルを追加するか、デフォルトのデータ プロトコルの順序を変更するか、またはデフォルトを削除できます。要求/応答ペア、RAW トラフィック ファイル、またはレコーディング以外の方法でサービス イメージを作成する場合で、指定されたデータ プロトコル ハンドラが存在する場合、デフォルトは追加されません。

データ プロトコル データ ハンドラを組み合わせて、互いに機能させることができます。 たとえば、要求側では、区切りテキスト データ プロトコル、ジェネリック XML ペイロード パーサ、および要求データ マネージャを一緒に使用できます。

使用可能なデータ プロトコルについては、それぞれ以下のセクションで説明しています。

自動ハッシュ トランザクション ディスカバリ

自動ハッシュ トランザクション ディスカバリ データ プロトコルは、データのハッシュ コードによってメッセージを識別します。 データがわずかに違うだけでハッシュ コードは異なるため、すべての要求を効果的に一意にすることができます。 このデータ プロトコルは、サービスに対して同じ少数の要求のセットを実行する場合に役立ちます。

自動ハッシュ トランザクション ディスカバリ データ プロトコルには、単純な用途があります。 処理する要求が指定されると、要求ボディのテキスト バージョンに標準の Java ハッシュ コード関数が適用されます。 結果のハッシュ コード値は、**lisa.vse.auto.hashDiscovery** という名前の要求に新しい引数として追加されます。

この機能は、特にメッセージングを仮想化する場合に、会話の識別に使用する合理的な一意の値を作成するために役立ちます。

このプロトコルは設定情報を必要としないため、レコーディング ウィザードではウィンドウが表示されません。

CICS Copybook

CICS Copybook データ プロトコルは、記録された要求を取得し、それぞれのコンテナ チャンクに分割します。 次に、Copybook データ プロトコルに各チャンクを送信し、対応する XML を集約します。 結果の XML は、コンテナの集約された XML です。

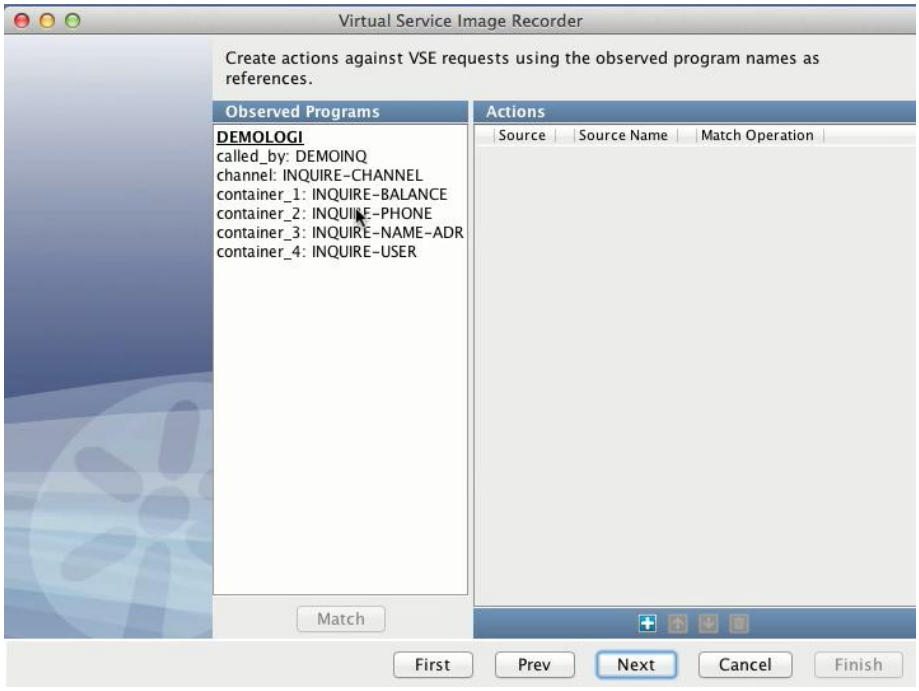
この例は、CICS Copybook データ プロトコルの使用方法を示しています。

次の手順に従ってください:

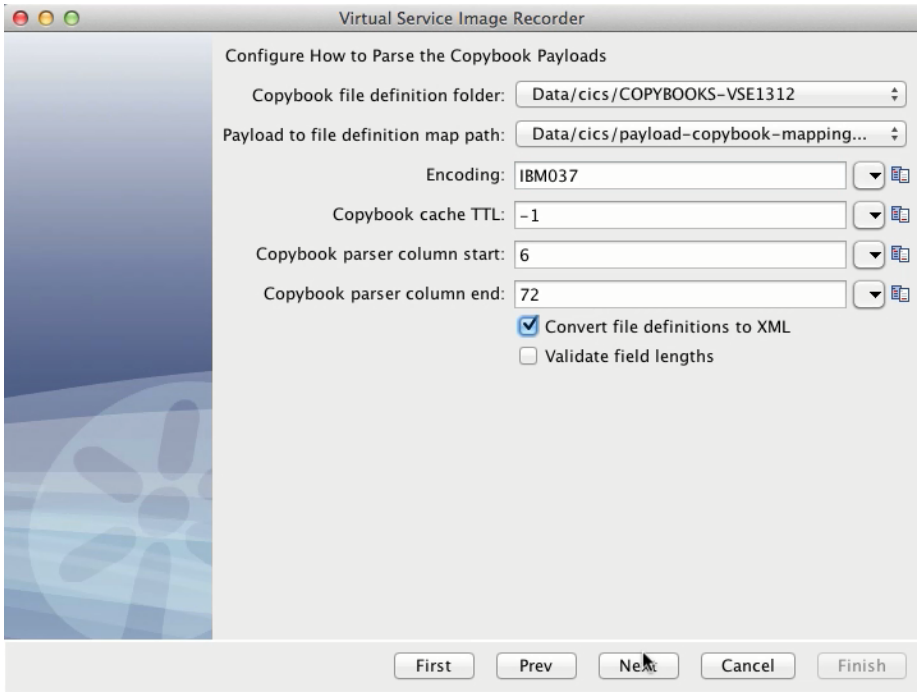
1. 仮想サービス イメージ レコーダの［基本 (P. 140)］タブで、必要なフィールドに入力します。
2. ［次へ］をクリックします。
3. IP アドレスとポートを入力し、［次へ］をクリックします。

   データ プロトコルの選択ウィンドウが表示されます。
4. 要求側で CICS Request Data Access データ プロトコルを選択します。
5. 要求側と応答側の両方に対して CICS Container Copybook データ プロトコルを選択して、［次へ］をクリックします。

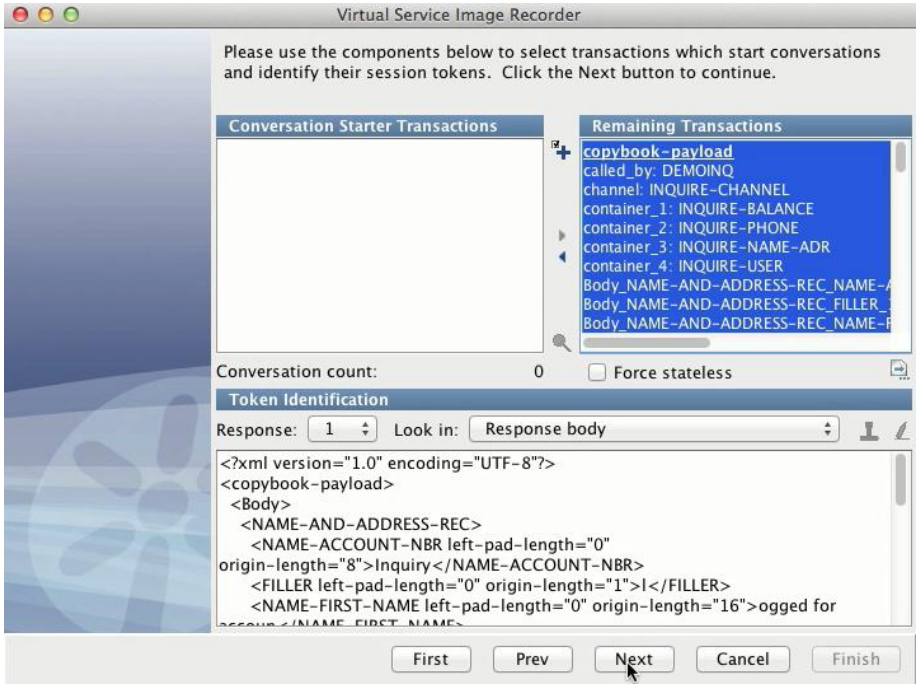
   ［観察されたプログラム］リストには、4 つの CICS コンテナが含まれています。



6. ［次へ］をクリックします。

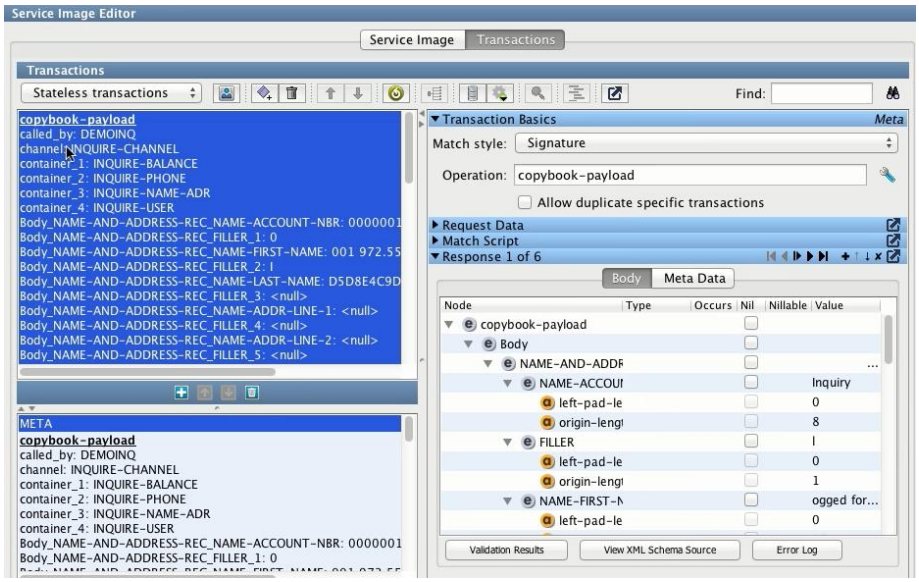


7. 要求側の定義で以下のパラメータを入力します。
   - Copybook ファイル定義用のフォルダ。
   - ペイロード ファイル定義マップのパス。
   - エンコーディング情報。
8. ［Convert file definitions to XML （ファイル定義を XML に変換）］を選択し、［次へ］をクリックします。



9. ［次へ］をクリックしてサービス イメージを完了します。
10. サービス イメージを表示するには、［View Service Image（サービス イメージの表示）］を選択して［終了］をクリックします。

    完了したサービス イメージが表示されます。



Copybook

Copybook データ プロトコルを使用すると、VSE は実行時にコール元には透過的な方法で Copybook ペイロードを XML との間で相互に変換できます。 XML にこれらのペイロードを表示できることにより、その他のメカニズム、標準メカニズム、VSE メカニズムで値を指定できます。ジェネリック XML ペイロード パーサを使用して、サービス イメージ内の一致とトランザクションの整理のために、要求の引数と操作を識別および選択できます。 標準の XML 構造を使用することにより、応答のマジック ストリングを処理して、より有効な方法で動的データを作成できます。

このセクションには、以下のトピックが含まれます。

用語

Copybook データ プロトコルをより深く理解するには、以下の用語を理解することが重要です。

- **Copybook**: データ レイアウトの定義に使用される階層構造。Copybook は、フィールド名、それらの長さ、および相互の関係を識別します。
- **Copybook ファイル定義**: Copybook が含まれるプレーン テキスト形式のファイル。
- **フィールド**: Copybook 内の単一のエレメント。フィールドは、レコード内のデータの単一の論理部分の名前、長さ、およびデータ型をカプセル化したものです。
- **ペイロード**: VSE が、単一の要求または単一の応答のいずれかで参照する、1 つ以上のレコードのコレクション。
- **ペイロード マッピング ファイル**: （「ペイロード Copybook マッピング」または「ペイロード ファイル定義マップ」とも呼ばれます）ペイロードの解析に必要な 1 つまたは複数の Copybook を識別する方法を記述した XML ドキュメント。 ペイロード マッピング ファイルは、結果の XML の基本構造を提供します。
- **レコード**: それ自体およびその中に構造エレメントを持たないデータのコレクション。 単一の Copybook を構成する 1 つ以上のフィールドでレコードを定義します。レコードには、Copybook が定義するすべてのフィールドのデータが含まれるとは限りませんが、レコードと Copybook の関係は常に 1 対 1 です。 特定のレコードは、単一の Copybook によってのみ定義されます。また、単一の Copybook は 1 つのレコードのみを定義します。

Copybook 仮想サービスを作成するための準備

Copybook 仮想サービスの作成準備を行うには、以下のタスクを完了します。

- ペイロードのレコードを定義する Copybook をすべて収集します。 これらの Copybook は、DevTest プロジェクト内のディレクトリにある必要があります。 使用するディレクトリには、使いやすい任意の階層を含めることができます。 通常、Copybook ファイル定義フォルダは、プロジェクトの Data フォルダに配置します。
- ペイロード マッピング ファイル (P. 256)を作成します。
- クライアントでレコーディングを準備するか、要求/応答ペアを収集するか、トラフィックの PCAP ファイルをキャプチャするか、またはその他の方法で仮想サービスに使用するトランザクションを収集します。

Copybook データ プロトコルを使用する方法

Copybook データ プロトコルは、VSE がサポートする任意のトランスポート プロトコルと共に使用できます。通常、Copybook データ プロトコル ハンドラは、JMS や MQ などのメッセージング プロトコルまたは CICS トランスポート プロトコルと共に使用されます。デモ サーバでは、HTTP を使用するサンプル Copybook アプリケーションを使用できます。

選択

Copybook データ プロトコルを選択する場合は、以下の点に注意してください。

- 要求側と応答側の両方でデータ プロトコル ハンドラを使用することを考慮します。片側でのみデータ プロトコルを使用することもできますが、一般的ではありません。 このドキュメントでは、片側での使用の詳細については説明していません。
- Copybook データ プロトコルは、要求（または応答）ボディを XML に変換します。 Copybook データ プロトコルは、要求のすべてのフィールドに対して引数を自動的に追加しません。また、Copybook データ プロトコルを活用するには、要求側に別のデータ プロトコル ハンドラを含める必要があります。最も一般的には、ジェネリック XML ペイロード パーサは、2 つ目のデータ プロトコル ハンドラです。 ただし、XML ペイロードを活用する任意のデータ プロトコル ハンドラを使用できます。

設定

トラフィックをキャプチャ（または要求/応答ペア、PCAP、RAW トラフィック ファイルからインポート）すると、Copybook DPH 設定ウィンドウが開きます。

以下のパラメータを入力します。

Copybook ファイル定義フォルダ

Copybook ファイル定義が格納されるプロジェクトのフォルダ。 このフォルダは、ペイロード マッピング ファイル内の相対パスのベース パスになります。

ペイロード ファイル定義マップ パス

この仮想サービスに対してペイロード マッピング ファイルの役割を果たす XML ドキュメント。

エンコーディング

任意の有効な Java 文字セットを指定します。 指定した場合、この値は、ペイロード内のバイトを出力 XML で使用されるテキストに変換するために使用されます。 VSE のデフォルト文字セットは UTF-8 です。 デフォルト文字セットは、local.properties 内の **lisa.vse.default.charset** を設定することによって設定できます。

Copybook キャッシュ TTL

実行時に VSE は指定された Copybook を参照する必要があるため、ファイルから読み取り、Copybook の XML 表現に変換する必要がある場合があります。 このパラメータは、変換された Copybook のキャッシュされたバージョンをどのくらいメモリに保持できるかを定義します。 TTL に到達すると、変換された Copybook はキャッシュから削除されます。 ファイルが再度必要な場合は、再度読み取られ、再変換されます。 TTL が 0 または負の数値の場合、キャッシュが無効であることを意味します。 ファイルは、必要になる度に読み取りおよび解析されます。 TTL が正の数値の場合、秒単位のタイムアウトとして使用されます。

Copybook パーサ開始列

多くの場合、Copybook には、各行の先頭に行番号が含まれます。通常、これらの行番号は 6 桁の数値であるため、このパラメータのデフォルト値は 6 です。 このオプションでは、Copybook ファイル定義を解析する場合、パーサがどの列から開始するかを指定できます。この数値はゼロベース インデックスで包含的です。ただし、「通常」の排他的な 1 ベース インデックスと見なすことができます。 この値に 6 を設定した場合、最初の 6 文字をスキップして、7 文字目から開始します。

Copybook パーサ終了列

場合によっては、Copybook の各行の末尾にその他の参照データが含まれることがあります。 その場合、パーサは、どの列で停止するかを知る必要があります。 ファイル内の行の末尾に「余分な」データがない場合、この数値は、ファイル内の最長行の長さより大きく設定することができます。 この数値はゼロベース インデックスで排他的です。 「通常」の包含的な 1 ベース インデックスと見なすことができます。この値に 72 を設定した場合、行内の 72 番目の文字を読み取った後に停止します (73 番目を読み取りません)。この数値が行の長さを超えている場合、パーサは行の末尾で停止します。

フィールド長の検証

このオプションは、応答側の VSE でのみ使用されます。レコーディング時には、ペイロードが XML に変換され、その後バイトに再変換されて、相互に変換できることが確認されます。 また、再生時には、コール元に応答する前に XML 応答がレコード/ペイロードに再変換されます。 それらの両方の操作で、VSE は、各フィールドの値が Copybook で指定された正確な長さであることを検証できます。 ただし、この検証を行うことは、必ずしも望ましいとは限りません。 たとえば、レコードの一部のフィールドにデータが含まれていない場合、その長さは 0 と見なされます。しかし、Copybook は、そのフィールドを 0 より大きい長さとして定義しています。 その場合、このオプションを選択すると、VSE は検証に失敗し、エラーをレポートします。データが Copybook で定義された長さに正しく一致しないことがわかっている場合は、このチェック ボックスをオフにします。 ただし、各レコードに格納するように指定されたデータが正確に含まれるように強制したい場合は、このオプションをオンにできます。

XML エレメントを要求引数として設定する

要求および応答が XML 文字列であることを検証するかどうかを指定します。

値

- **オン：** レコーダが使用する XML メッセージから変数を識別します。 これは、要求および応答が XML 文字列であることを検証します。
- **オフ：** 要求および応答が XML 文字列であることを検証しません。

要求および応答が XML 文字列であることを検証します。 このオプションを選択すると、レコーダが使用する XML メッセージから変数を識別できます。変数の識別の詳細については、「ジェネリック XML ペイロード パーサ (P. 279)」を参照してください。

同じエディタを、VSM のデータ プロトコル フィルタで使用できます（要求側と応答側で 1 つずつ）。このエディタでは、必要に応じて、レコーディング後に設定を変更できます。

ペイロード マッピング ファイル

*ペイロード マッピング ファイル*は、受信ペイロードが、対応する Copybook とどのように一致可能かを記述した XML ドキュメントです。 ペイロード マッピング ファイルは、*ペイロード Copybook マッピング*および*ペイロード Copybook ファイル定義マップ*とも呼ばれます。また、このファイルは、ユーザにわかりやすいように結果の XML を構造化する方法についてのヒントも提供します。

サンプル

サンプル マッピング ファイルは、ここで参照できます。 サンプル ファイルには、さまざまな設定の例、および各ノードの各属性について説明したコメントが含まれています。 このサンプル ファイルは、Copybook マッピング ファイルの作成時に参考として役立ちます。

構造

以下の例は、基本的なペイロード マッピング ファイルの構造を示しています。 わかりやすくするために、この例には、値の属性が含まれていません。

```
<payloads>
  <payload>
    <key></key>
    <section>
      <copybook></copybook>
      ...
    </section>
    ...
  </payload>
  <payload>
    ...
  </payload>
</payloads>
```

<payloads>

ドキュメントのルート XML ノード。このノードは、繰り返すことができません。

<payload>

このエレメントは、その全体で、一致するペイロードを完全に記述します。 このドキュメントで、ペイロードが <payload> エレメントに一致すると判断された場合、<payload> エレメントを使用して受信ペイロードを記述します。

<key>

（オプション）このオプションにより、XML ドキュメントの可読性が向上します。 このエレメントで指定するものは、すべて <payload> エレメントの属性としても指定できます。

<section>

ペイロードの一部を定義する Copybook の論理グループ。 複数の <section> が存在する場合は、それらが繰り返しなしで順番に処理されます。

<copybook>

ペイロード内のレコードを記述する場合も、記述しない場合もある単一の Copybook。 <section> には、複数の <copybook> エレメントを含めることができます。

<payload>

A <payload> エレメントには、以下の属性を含めることができます。

```
<payload name="TEST" type="request" matchType="all" key="reqKey"
value="reqVal" keyStart="3" keyEnd="6" headerBytes="0"
footerBytes="0" saveHeaderFooter="false" definesResponse="false">
    ...
</payload>
```

属性	必須/任意	説明
name	必須	ここで記述する要求のタイプを識別する一意の名前を定義します。 **name** は、同じタイプ（要求または応答）のペイロードのセット内で一意である必要があります。
type	必須	ペイロードのタイプを指定します。 **値：** 以下のいずれかです。 ■ request ■ response

matchType	オプション	matchType 属性は、異なる方法で「response」タイプのペイロード定義に適用されます。たとえば、応答には引数や操作名が含まれません。「response」タイプのペイロード定義でこの属性に **argument**、**attribute**、または **operation** を設定しても、何も一致しません（一致する要求の definesResponse 属性が **true** に設定されて、この属性を上書きしない限り）。 「response」タイプのペイロード定義でこの属性に **all** を設定すると、argument、attribute、および operation の一致はスキップされます。 **値：** 以下のいずれかです。 ■ **argument：** 指定された引数に対してのみ一致を試行します。 ■ **attribute：** 指定された属性に対してのみ一致を試行します。 ■ **metaData**： 指定されたメタデータに対してのみ一致を試行します。 ■ **operation：** 操作名に対してのみ一致を試行します。 ■ **payload：** 要求のボディに対して一致を試行します。 ■ **all：** 次の順番で一致を試行します： argument、attribute、metaData、operation、payload。 **デフォルト：** payload
key	必須	この属性の動作は、以下のように、matchType によって異なります。 ■ matchType が **operation** または **payload** の場合、一致に対してこの属性の値が使用されます。 ■ matchType が **argument**、**attribute**、**metaData**、または **all** の場合、この属性の値は argument、attribute、または metaData エントリのキーであると想定されます。 matchType の値が **all** の場合、このキー値は、操作名またはペイロード ボディに対する一致に使用されません。 代わりに attribute 値が使用されます。

value	オプション/必須	この属性の動作（および、必須かどうか）は、以下のように、matchType によって異なります。 ■ matchType が **operation** または **payload** の場合、この属性の値は無視され、除外することができます。 ■ matchType が **argument**、**attribute**、または **metaData** の場合、この属性は argument、attribute、または metaData エントリの値であると想定され、一致に対して必須となります。 ■ matchType が **all** の場合、この属性は必須であり、argument、attribute、および metaData の一致では、上記で説明したように動作します。ただし、operation および payload の一致では、この属性の値が一致に使用されます（matchType が **operation** または **payload** の場合のように、**key** 属性の値が使用されるのではなく）。
keyStart	オプション	ペイロード内で、キーの検索を開始する位置（1 ベースのインデックス）を指定します。 検索は、この値を**含めて**行われます。 ■ keyStart を 1 に設定すると、最初のバイトから検索が開始されます。 ■ keyStart を設定しないと、ペイロード全体が検索され、keyEnd は無視されます。 ■ keyStart を 1 より小さい数に設定すると、値は 1 として処理されます。 ■ keyStart をペイロードの長さより大きな数に設定すると、ペイロード全体が検索され、keyEnd は無視されます。 ■ keyStart を keyEnd と等しいか keyEnd より大きく設定すると、ペイロード全体が検索されます。 ■ keyEnd から keyStart を引いた値がキーの長さより小さい場合、ペイロード全体が検索されます。

keyEnd	オプション	ペイロード内で、キーの検索を終了する位置（1 ベースのインデックス）を指定します。 検索は、この値を**含めずに**行われます。 ■ keyEnd を 3 に設定すると、バイト 1 と 2 のみが検索されます。 ■ keyEnd を設定しないと、検索は keyStart で開始され、ペイロードの末尾で終了します。 ■ この値がペイロードの長さより大きい場合、検索はペイロードの末尾で終了します。 ■ この値が keyStart に等しいか keyStart より小さい場合、ペイロード全体が検索されます。 ■ keyEnd から keyStart を引いた値がキーの長さより小さい場合、ペイロード全体が検索されます。
headerBytes	オプション	ペイロードの先頭から削除するバイト数を指定します。値を指定しない場合、デフォルトは 0 になります。 この属性を指定する場合、値は有効な整数である必要があります。
footerBytes	オプション	ペイロードの末尾から削除するバイト数を指定します。値を指定しない場合、デフォルトは 0 になります。 この属性を指定する場合、値は有効な整数である必要があります。
saveHeaderFooter	オプション	削除されるヘッダとフッタのバイト数が XML バージョンの要求で **rawHeader** タグおよび **rawFooter** タグの下の 16 進エンコードされた文字列として保持されるかどうかを指定します。 **値** ■ true : 削除されるヘッダとフッタのバイト数が保持されます。 ■ false : 削除されるヘッダとフッタのバイト数が保持されません。 **デフォルト値：** false。
definesResponse	オプション	**true** の場合、この要求に対する応答は、同じ名前を持った **response** タイプのペイロードエレメントを検索します。タイプが **response** の場合、この属性は無視されます。 **デフォルト値：** false。

オプションの <key> エレメントを使用して、キー関連の属性を置換できます。 上記の <payload> エレメントの例は、必要に応じて、可読性を高めるために以下のように記述できます。

```
<payload name="TEST" type="request" headerBytes="0" footerBytes="0"
saveHeaderFooter="false" definesResponse="false">
        <key matchType="all" value="reqVal" keyStart="3"
    keyEnd="6">reqKey</key>
        ...
    </payload>
```

<section>

例： 以下の例は、すべての属性および属性の記述を持つサンプルのセクション エレメントです。

```
...
    <section name="Body">
        ...
    </section>
    ...
```

属性	必須	説明
name	必須	ペイロードの XML 出力バージョンでのグループの XML エレメントの名前を定義します。 その下に、1 つ以上の変換された Copybook エレメントが存在します。

<copybook>

例： 以下の例は、すべての属性および属性の記述を持つサンプルの Copybook エレメントです。

```
...
<copybook key="cpk" order="1" max="1" name="TESTRECORD"
length-field="SOME-ID">TESTIN.CPY.TXT</copybook>
...
```

属性	必須	説明
key	必須	Copybook を識別するレコードの一意の文字列を定義します。 技術的には、この属性は省略可能です。 ただし、キーを指定しない場合、その Copybook はペイロードに適用される唯一のものであることを意味します。ペイロードでバイトが使い果たされるまで、繰り返し適用されます。 複数の Copybook エレメントにキーが存在しない場合、max 属性が指定されない限り、常に最初のものが使用されます。
order	オプション	ペイロード内でレコードが見つかる順番に関するヒントを定義します。 使用される番号は無関係ですが、ペイロード内の「後ろ」のレコードには、より大きな整数を使用する必要があります。 複数の Copybook に同じ番号でタグ付けすることができます。これは、それらのレコードが任意の順番になることを意味します。特定の番号でレコードが見つかった場合、それ以降の検索では、その番号以上の Copybook のみ検索されます。 ペイロードに一致することがないグループの Copybook を含めることができます。 それらは単に無視されます。 ただし、各 Copybook を確認する必要があるため、パフォーマンスに影響があります。 **デフォルト：** 0
max	オプション	Copybook をペイロードに適用できる最大回数を定義します。 空白、0、負の数、数値以外、および存在しない値はすべて「制限なし」を意味します。

name	オプション	レコード名（Copybook のルート レベル）を上書きする値を定義します。 ■ この値を設定すると、この Copybook 用に XML に生成されたノードに、Copybook 定義のレコード名の代わりに、この名前を使用します。 ■ この値を指定しない場合、デフォルトでは、Copybook 定義からレコード名を検索し、それを使用します。 ■ この値を Copybook 定義のレコード名に設定すると、このファイルの可読性が向上する効果のみがあります。
length-field	オプション	次のレコード検索が正しい位置から開始されるようにペイロードを分割する方法を定義します。 この属性が存在しない場合、プロセッサは定義から Copybook の長さを決定しようとします。 何らかの理由で長さが判明しない場合、プロセッサは、ペイロードの残りがこの Copybook に該当すると仮定します。処理は、この Copybook を適用した後に終了します。 このフィールドが符号なしの Display 数値フィールドでない場合、プロセッサはこのフィールドを無視します。

一致ロジック

XML エレメントと引数の記述に基づいて、一致ロジックの機能を判断できます。 ただし、明確な全体像を示すために、このセクションでは一致について詳細に説明します。

VSE がペイロードを受信すると、一致が以下の段階で発生します。

- ペイロード
- セクション
- Copybook
- 最終処理

ペイロード

ペイロードは、VSE が新しいペイロードを受信したときに発生する一致の最初の段階です。 この段階では、ペイロード マッピング ファイルのどの <payload> エレメントが受信したペイロードに対応するかが判断されます。 VSE は、ペイロード マッピング ファイル内のエレメントを上から下まで順番に処理することにより、<payload> エレメントを照合します。 最初に一致したものが使用されます。

注: ペイロード エレメントがペイロードに一致しない場合、それは「不明なペイロード」として扱われます。 コンテンツは 16 進エンコードされ、汎用の XML 構造でラップされます。必要に応じて、不明なペイロードは、再生時に自動的にバイトに変換されます。

要求ペイロードの場合

1. このエレメントの type 属性が「request」の場合、処理を続行します。 それ以外の場合、このエレメントは照合されません。
2. matchType が argument の場合、キーがこのエレメントの key 属性に一致し、値がこのエレメントの value 属性に一致する要求引数が検索されます。 見つかった場合、このペイロード エレメントは一致します。
3. matchType が attribute の場合、キーがこのエレメントの key 属性に一致し、値がこのエレメントの value 属性に一致する要求属性が検索されます。 見つかった場合、このペイロード エレメントは一致します。

4. matchType が metaData の場合、キーがこのエレメントの key 属性に一致し、値がこのエレメントの value 属性に一致する要求メタデータ エントリが検索されます。 見つかった場合、このペイロード エレメントは一致します。
5. matchType が operation の場合、要求の操作名がこのエレメントの key 属性に一致するかどうかが確認されます。 一致した場合、このペイロード エレメントは一致します。
6. matchType が payload の場合、このエレメントの keyStart および keyEnd 属性で指定された境界内で key 属性の値が検索されます。 キーがそれらの境界内で見つかった場合、ペイロード エレメントは一致します。
7. matchType が all の場合、手順 2 ～ 6 が順番に処理され、一致が見つかると停止します。 異なる点は、手順 5 および 6 で key 属性の代わりに value 属性が使用されることのみです。

応答ペイロードの場合（レコーディング中）

1. 以前に参照された要求のペイロード エレメントの definesResponse 属性が **true** に設定されている場合、要求のエレメントと同じ名前でタイプが response のペイロード エレメントをすぐに返します。 そのようなエレメントが見つからない場合、一致は存在しません。
2. 以前に参照された要求のペイロード エレメントの definesResponse 属性が true に設定されて*いない*場合
   a. このエレメントの type 属性が response の場合、処理を続行します。それ以外の場合、このエレメントは照合されません。
   b. matchType が metaData の場合、キーがこのエレメントの key 属性に一致し、値がこのエレメントの value 属性に一致する応答メタデータ エントリが検索されます。 見つかった場合、このペイロード エレメントは一致します。
   c. matchType が payload の場合、このエレメントの keyStart および keyEnd 属性で指定された境界内で key 属性の値が検索されます。キーがそれらの境界内で見つかった場合、ペイロード エレメントは一致します。
   d. matchType が all の場合、手順 b と c が順番に実行され、一致が見つかると停止します。 異なる点は、手順 c で key 属性の代わりに value 属性が使用されることのみです。

応答ペイロードの場合（再生中）

再生中は、応答に対して一致は確認されません。 代わりに、レコーディング中に、一致したものがすべて応答オブジェクトの Metadata に保存されます。その後、再生中にその応答 オブジェクトが返されると、プロセッサは応答の Metadata から値を取得し、その名前（および type="response"）のペイロード エレメントを検索して使用します。 そのようなエレメントが存在しない場合、それは回復不可能なエラーと見なされます。

セクション

ペイロード エレメントが識別された後、プロセッサは各セクション エレメントを上から下まで順番に確認します。 この段階では、一致は確認されません。 プロセッサがセクションを完全に処理すると（セクション内のいずれの Copybook も一致しない場合）、次のセクションに進み、前のセクションに戻ることはありません。

Copybook

一致の最終段階では、ペイロード内のレコードがすべて処理されるまで、このセクション内のどの Copybook を最初に適用し、次に 2 番目以降に適用する順番を判断します。 この段階は、最も複雑な一致段階です。 処理は、以下のとおりです。

1. プロセッサは、セクション内のすべての Copybook エレメントを確認し、order 属性に指定された数値で並べ替えます。
2. 最も小さい order の数値を持つ Copybook エレメントのリストが最初に確認されます（ファイルに指定された順番）。残りのペイロードで、この Copybook エレメントの key 属性の値が検索され、見つかった位置（すべて）のインデックスが保存されます。 残りのペイロードで、この Copybook エレメントの key 属性の値が検索され、見つかった位置（すべて）のインデックスが保存されます。
3. 最も小さい order の数値のリスト内ですべての Copybook が確認され、いずれかが一致した場合、ペイロードで最初に見つかったものが使用されます。
4. Copybook が一致した場合、以下の処理を行います。

a. 一致する Copybook が、以前にこのペイロードに一致していた場合（ペイロード内の前のレコード）、もう一度適用できるかどうか max 属性が確認されます。 max の値に達している場合、これは一致とは見なされず、次に一致する Copybook が使用されます。

b. max の値に達していない場合、この Copybook がペイロードに適用され、Copybook によって指定されるバイトするが処理されます。Copybook に length-field フィールド属性が存在する場合は、レコード内のそのフィールドの値を使用して処理するバイト数が判断され、プロセッサは手順 2 に戻ります。

5. Copybook が一致しなかった場合、次に小さい order の数値を持つ Copybook のリストが、手順 2、3、4 と同じルールを使用して確認されます。プロセッサは、order の数値のリストに一致する Copybook がないと判断すると、このペイロードに対してその order の数値のリスト（このセクション）の処理を再度試行しません。

6. すべてのリストを確認して一致が見つからない（または、ペイロードがバイトを使い果たした）場合、プロセッサは次のセクションに進みます。

最終処理

ペイロードのすべてのセクションが処理された後、バイトがペイロード内に残されている場合、それらは切り捨てられます。

CTG Copybook

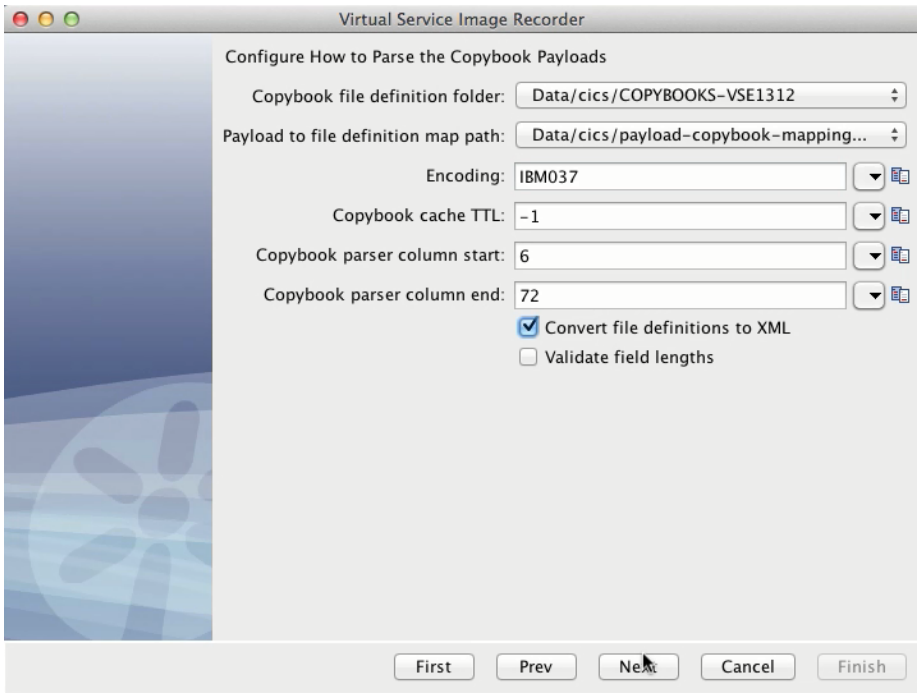
CTG Copybook データ プロトコルは、記録された要求を取得し、それぞれのコンテナ チャンクに分割します。 次に、Copybook データ プロトコルに各チャンクを送信し、対応する XML を集約します。 結果の XML は、コンテナの集約された XML です。

この例は、CTG Copybook データ プロトコルの使用方法を示しています。

次の手順に従ってください:

1. 仮想サービス イメージ レコーダの［基本 (P. 140)］タブで、必要なフィールドに入力します。
2. ［次へ］をクリックします。
3. IP アドレスとポートを入力し、［次へ］をクリックします。

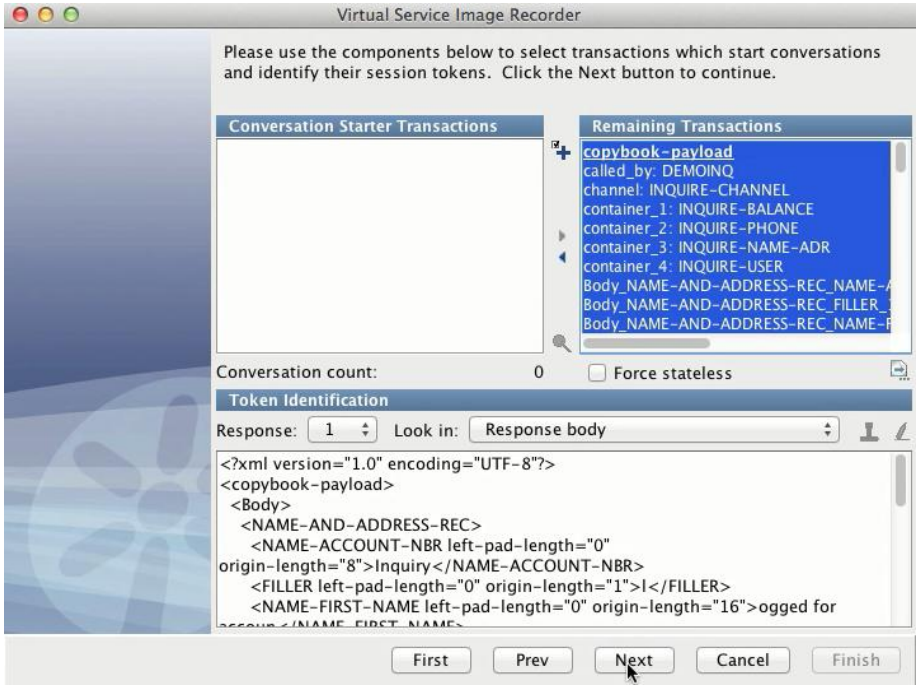
   データ プロトコルの選択ウィンドウが表示されます。
4. 要求側と応答側の両方に対して CTG Copybook データ プロトコルを選択して、［次へ］をクリックします。
5. ［次へ］をクリックします。



6. 要求側の定義で以下のパラメータを入力します。
   - Copybook ファイル定義用のフォルダ。

- ペイロード ファイル定義マップのパス。
- エンコーディング情報。

7. ［Convert file definitions to XML （ファイル定義を XML に変換）］を選択し、［次へ］をクリックします。



8. ［次へ］をクリックしてサービス イメージを完了します。
9. サービス イメージを表示するには、［View Service Image （サービス イメージの表示）］を選択して［終了］をクリックします。

   完了したサービス イメージが表示されます。

データ ディセンシタイザ

トランスポート層がディセンシタイズするデータを「参照」できない場合、VSE はデータ プロトコル ハンドラを提供します。 例として、SOAP over HTTP またはメッセージングでの gzip の使用について考えてみます。 VSE に gzip された SOAP 要求（または応答）が提供された場合、このレベルではデータのディセンシタイズを実行できません。 その場合、レコーダにデータ プロトコルを設定して、必要な gunzip 操作を実行できます。 SOAP ボディがプレーン テキストに変換された後にディセンシタイズを処理するには、データ ディセンシタイズ プロトコルを追加します。

通常、VSE は、できるだけ低いレベルでデータのディセンシタイズを試行します。VSE レコーディング ウィザードの最初のウィンドウにある［ディセンシタイズ］チェック ボックスは、トランスポート プロトコルができるだけ早くディセンシタイズを行うよう要求します。 ただし、トランスポート プロトコルにとって、要求のペイロードが不透明な場合があります。 gzip や FastInfoSet など、ペイロードが圧縮される HTTP メッセージについて考えてみます。 この場合、トランスポート プロトコルはディセンシタイズ ルールを適用できません。このデータ ディセンシタイザ データ プロトコルは、そのような状況に対処するために作成されています。

データ ディセンシタイザ データ プロトコルを、データ プロトコル チェーンの要求側、応答側、またはその両方に追加します。 このプロトコルは、別のプロトコルが不透明なペイロードを読み取り可能なものに変換した後に追加します。 このデータ プロトコルは、レコーディング専用に意図されており、再生では使用されません。 要求または応答がそれぞれプロトコル ハンドラに渡されると、**LISA_HOME¥desensitize.xml** で指定されたすべてのディセンシタイズ ルールが適用されます。

このプロトコルは設定情報を必要としないため、レコーディング ウィザードではウィンドウが表示されません。

区切りテキスト

区切りテキスト データ プロトコルにより、区切りテキスト データを引数および値に解析できます。 このプロトコルは、要求または応答のボディを取得し、それを解析して、ボディを解析されたデータの XML 表現に置き換えます。 以下に例を示します。

```
“name1=val1;name2=val2”
```

は、以下のようになります。

```
<name1>val1</name1><name2>val2</name2>
```

次の手順に従ってください:

1. 仮想サービス レコーダの［基本 (P. 140)］タブに入力します。
2. 要求側と応答側の両方で［区切りテキスト データ プロトコル］を選択します。
3. レコーディングが完了したら、区切りテキスト要求の形式を選択します。

   名前/値ペア

   名前/値ペアの間の区切り文字、および名前と値の間の区切り文字を定義します。 たとえば、データが「`name1=val1,name2=val2,name3=val3`」である場合、

   ペアの間の区切り文字はカンマであり、名前と値の間の区切り文字は等号です。

   値リスト

   値の間の区切り文字を定義します。2 種類のリストを使用できます。

   - テキスト区切り文字によって分離された値のリスト。

     たとえば、データが

     「`val1,val2,val3,val4`」である場合、

     区切り文字はカンマです。パラメータは、位置に基づいて命名されます。

   - 改行文字（キャリッジ リターン、ライン フィードなど）によって分離された値のリスト。

     パラメータは、位置に基づいて命名されます。

   固定幅

データ フィールドの幅を定義します（整数値）。 パラメータは、位置に基づいて命名されます。

正規表現区切り

値を検出する正規表現を定義します。 たとえば、データが

「xxxx123xxx456xx789」である場合、

「¥d¥d¥d」という正規表現によって「123」、「456」、および「789」が値として検出され、残りは破棄されます。パラメータは、位置に基づいて命名されます。

行区切り

4. 以下のフィールドに入力します。

フィールド名パス

番号付けされたフィールド名のリストを指定する行区切りドキュメントである、フィールド名ドキュメントの位置を定義します。デフォルトでは、フィールド名は **value1**、**value2**、**value3** などになります。 それらの XML エレメントに異なる名前を指定するには、フィールド名ドキュメントを使用します。

区切り文字タイプ

名前/値ペアおよび値リストを分離するために使用される区切り文字のタイプを定義します。 指定された区切り文字に基づいて、テキストまたは 16 進の区切り文字が自動的に選択されます。

値： 任意の英数字と以下の表現

- **¥r**（キャリッジ リターン）
- **¥n**（改行）
- **¥t**（タブ）

また、16 進表記を使用して区切り文字を指定することもできます。

注: XML 1.0 で有効な区切り文字のみを使用できます。 印刷できない制御文字を指定すると、サービス イメージを使用できなくなります。

XML エレメントを要求引数として設定する

パラメータおよび関連する値を引数として要求に追加するかどうかを指定します。

値

- **オン:** パラメータおよび関連する値が引数として要求に自動的に追加されます。
- **オフ：** ジェネリック XML ペイロード パーサ (P. 279)（または類似のプロトコル）を使用して、引数となる値を選択する必要があります。

注: 応答側で区切りテキスト データ プロトコルが使用される場合、このチェック ボックスは無効になります。

5. ［次へ］をクリックします。

   名前/値ペアは XML で表されます。 ここで、トランザクションをダブルクリックして、そのトランザクションのコンテンツを表示できます。

6. 要求側に対する方法と同じ方法で、応答側の区切り文字を設定します。

   処理が完了すると、XML に変換された仮想サービス イメージとペイロードを参照できます。

詳細情報:

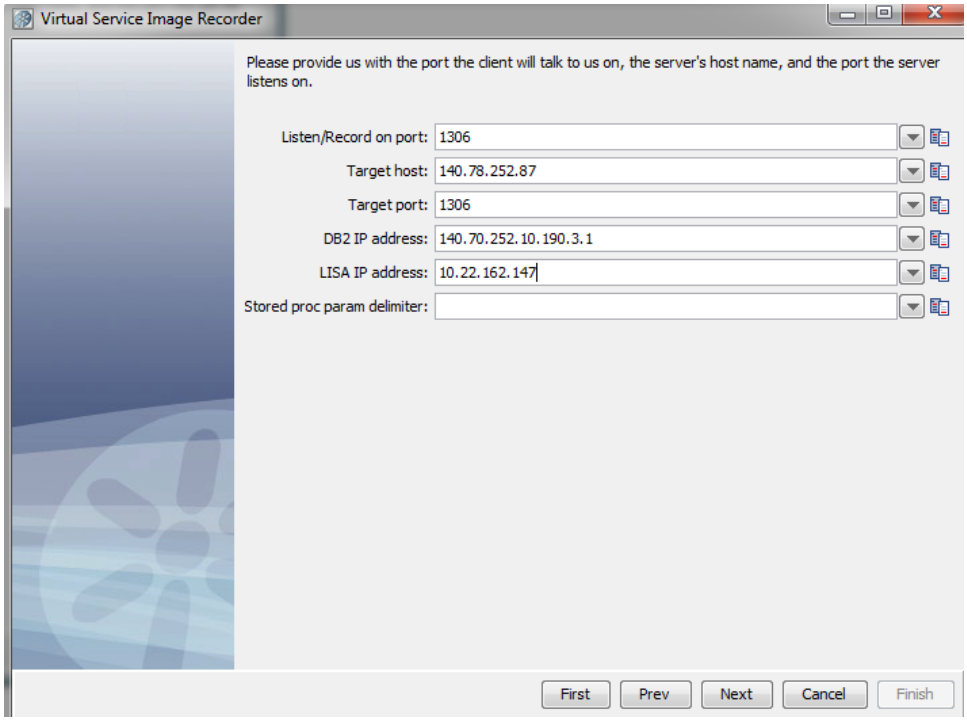
ジェネリック XML ペイロード パーサ データ プロトコル (P. 279)

DRDA

DRDA データ プロトコルは、レコーディング中に、バイナリの DRDA（Distributed Relational Database Architecture）ペイロードを XML に変換します。 このプロトコルは、ネイティブ DevTest 機能との整合性、可読性、動的データのサポートを促進します。 応答は、再生時にネイティブ形式に再変換されます。

DRDA データ プロトコルを選択すると、要求側と応答側のデータ プロトコル フィールドが、［DRDA データ プロトコル］の選択内容で自動的に入力されます。

レコーダの次のウィンドウでは、通信情報を指定します。



サービス イメージの通信パラメータを入力します。

注: ペイロードに非 ASCII 文字が含まれる場合、ペイロードはサービス イメージ エディタにバイナリで表示されます。

EDI X12 データ プロトコル

EDI X12 データ プロトコルは、ANSI X12 EDI ドキュメントを、要求のボディ内の XML 表現に変換します。また、DPH は、特定のドキュメント バージョン（たとえば 004010）と結合された EDI ドキュメント タイプ（たとえば 835）および一連の VSE 要求引数から構成される VSE 要求操作を作成します。これらの要求引数は、XML ボディのフラットな表現です。これらは、XML エレメントを組み合わせて形成されます。各エレメントは、タグ用の「_」、または各タグ属性の引数名およびその値のエレメント値用の「@」で区切られます。

たとえば、以下の ANSI X12 EDI 850 ドキュメントのようになります。

```
ISA*00* *00* *ZZ*0011223456 *ZZ*999999999
*990320*0157*U*00300*000000015*0*P*~$
GS*PO*0011223456*999999999*950120*0147*5*X*003040$
ST*850*000000001$
BEG*00*SA*95018017***950118$
N1*SE*UNIVERSAL WIDGETS$
N3*375 PLYMOUTH PARK*SUITE 205$
N4*IRVING*TX*75061$
N1*ST*JIT MANUFACTURING$
N3*BUILDING 3B*2001 ENTERPRISE PARK$
N4*JUAREZ*CH**MEX$
N1*AK*JIT MANUFACTURING$
N3*400 INDUSTRIAL PARKWAY$
N4*INDUSTRIAL AIRPORT*KS*66030$
N1*BT*JIT MANUFACTURING$
N2*ACCOUNTS PAYABLE DEPARTMENT$
N3*400 INDUSTRIAL PARKWAY$
N4*INDUSTRIAL AIRPORT*KS*66030$
PO1*001*4*EA*330*TE*IN*525*VN*X357-W2$
PID*F****HIGH PERFORMANCE WIDGET$
SCH*4*EA****002*950322$
CTT*1*1$
SE*20*000000001$
GE*1*5$
```

この結果の VSE 要求 オブジェクトには以下のものが含まれます。

操作： 850-00340

引数： 以下の名前/値ペアのサブセットになります。

```
interchange_sender_address <null>
interchange_sender_address@Id    0011223456
```

```
interchange_sender_address@Qual  ZZ
interchange_receiver_address <null>
interchange_receiver_address@Id  999999999
interchange_receiver_address@Qual ZZ
interchange_group_transaction_segment_element_1 00
...
```

ボディ： 以下のように構造化されます。 このドキュメントでは、XML ドキュメントは可読性を考慮して整形されています。 要求ボディには、改行やインデントなどの整形要素は含まれていません。

```
<?xml version="1.0" encoding="UTF-8"?>
<ediroot>
    <interchange Standard="ANSI X.12" Date="990320" Time="0157"
StandardsId="U" Version="00300"
        Control="000000015">
        <sender>
            <address Id="0011223456 " Qual="ZZ"/>
        </sender>
        <receiver>
            <address Id="999999999 " Qual="ZZ"/>
        </receiver>
        <group GroupType="PO" ApplSender="0011223456"
ApplReceiver="999999999" Date="950120"
            Time="0147" Control="5" StandardCode="X"
StandardVersion="003040">
            <transaction DocType="850" Name="Purchase Order"
Control="000000001">
                <segment Id="BEG">
                    <element Id="BEG01">00</element>
                    <element Id="BEG02">SA</element>
                    <element Id="BEG03">95018017</element>
                    <element Id="BEG06">950118</element>
                </segment>
                <loop Id="N1">
                    <segment Id="N1">
                        <element Id="N101">SE</element>
                        <element Id="N102">UNIVERSAL
WIDGETS</element>
                    </segment>
                    <segment Id="N3">
                        <element Id="N301">375 PLYMOUTH
PARK</element>
                        <element Id="N302">SUITE 205</element>
                    </segment>
                    <segment Id="N4">
                        <element Id="N401">IRVING</element>
                        <element Id="N402">TX</element>
                        <element Id="N403">75061</element>
```

```
                    </segment>
                </loop>
                <loop Id="N1">
                    <segment Id="N1">
                        <element Id="N101">ST</element>
                        <element Id="N102">JIT
MANUFACTURING</element>
                    </segment>
                    <segment Id="N3">
                        <element Id="N301">BUILDING 3B</element>
                        <element Id="N302">2001 ENTERPRISE
PARK</element>
                    </segment>
                    <segment Id="N4">
                        <element Id="N401">JUAREZ</element>
                        <element Id="N402">CH</element>
                        <element Id="N404">MEX</element>
                    </segment>
                </loop>
                <loop Id="N1">
                    <segment Id="N1">
                        <element Id="N101">AK</element>
                        <element Id="N102">JIT
MANUFACTURING</element>
                    </segment>
                    <segment Id="N3">
                        <element Id="N301">400 INDUSTRIAL
PARKWAY</element>
                    </segment>
                    <segment Id="N4">
                        <element Id="N401">INDUSTRIAL
AIRPORT</element>
                        <element Id="N402">KS</element>
                        <element Id="N403">66030</element>
                    </segment>
                </loop>
                <loop Id="N1">
                    <segment Id="N1">
                        <element Id="N101">BT</element>
                        <element Id="N102">JIT
MANUFACTURING</element>
                    </segment>
                    <segment Id="N2">
                        <element Id="N201">ACCOUNTS PAYABLE
DEPARTMENT</element>
                    </segment>
                    <segment Id="N3">
                        <element Id="N301">400 INDUSTRIAL
PARKWAY</element>
```

```
                    </segment>
                    <segment Id="N4">
                        <element Id="N401">INDUSTRIAL
AIRPORT</element>
                        <element Id="N402">KS</element>
                        <element Id="N403">66030</element>
                    </segment>
                </loop>
                <loop Id="PO1">
                    <segment Id="PO1">
                        <element Id="PO101">001</element>
                        <element Id="PO102">4</element>
                        <element Id="PO103">EA</element>
                        <element Id="PO104">330</element>
                        <element Id="PO105">TE</element>
                        <element Id="PO106">IN</element>
                        <element Id="PO107">525</element>
                        <element Id="PO108">VN</element>
                        <element Id="PO109">X357-W2</element>
                    </segment>
                    <loop Id="PID">
                        <segment Id="PID">
                            <element Id="PID01">F</element>
                            <element Id="PID05">HIGH PERFORMANCE
WIDGET</element>
                        </segment>
                    </loop>
                    <loop Id="SCH">
                        <segment Id="SCH">
                            <element Id="SCH01">4</element>
                            <element Id="SCH02">EA</element>
                            <element Id="SCH06">002</element>
                            <element Id="SCH07">950322</element>
                        </segment>
                    </loop>
                </loop>
                <loop Id="CTT">
                    <segment Id="CTT">
                        <element Id="CTT01">1</element>
                        <element Id="CTT02">1</element>
                    </segment>
                </loop>
            </transaction>
        </group>
    </interchange>
</ediroot>
```

ジェネリック XML ペイロード パーサ データ プロトコル

ジェネリック XML ペイロード パーサは、要求と応答が XML 文字列であることを識別します。 このプロトコルを使用すると、レコーダが使用する XML メッセージから変数を識別できます。

注: 区切りテキスト データ プロトコル (P. 271)、Copybook データ プロトコル (P. 249)、および DRDA データ プロトコル (P. 274)については、この機能は、データ プロトコル設定ウィンドウで［XML エレメントを要求引数として設定する］チェック ボックスをオンにすることにより有効になります。

会話およびトランザクションの識別

記録されたサービス イメージの品質は、会話およびその個別のトランザクションを識別するための情報を、VSE がどの程度持っているかに直接依存します。 具体的には、VSE には、トランザクションを互いに区別するための支援が必要です。

- **会話の一部として:** セッションを表す何らかの一意の ID を識別します。
- **個別の操作として：** 操作のセマンティックに固有の何らかの情報を識別します。 たとえば、「順番の作成」操作と「順番のリスト」操作を区別します。

VSE は、検索対象のパターンを内部に多数保持しています。たとえば、HTTP 仮想化では（サーバが Java サーバの場合）、VSE は、sessionid という名前の特殊な変数の値を含む Cookie を設定します。sessionid 変数は、セッションを一意に識別します。 VSE は、この変数を使用して、異なるセッションを区別できます。

ただし、VSE には、さらに支援が必要な場合があります。これらは、以下のようにして提供されます。

- HTTP レコーディングでは、VSE でトークンを指定できます。
- JMS プロトコルでは、この識別のために、VSE が JMS 相関 ID、カスタム JMS ヘッダ、またはすべての JMS ヘッダを使用するかどうかを指定できます。
- VSE では、メッセージ自体から有意な情報を取得するための動的データ プロトコルを指定できます。以下のセクションでは、この技術について説明します。

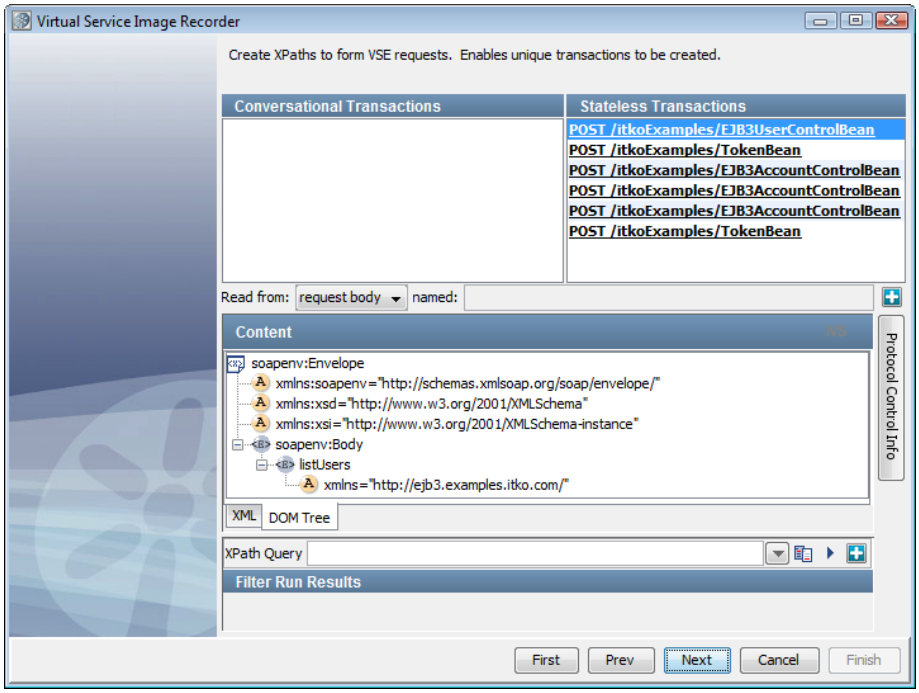
ジェネリック XML ペイロード パーサの使用は、VSE が記録されたメッセージ（ペイロード）のボディを確認し、トランザクションの識別に役立つ有意な情報を抽出することを支援する技術です。 特に、WebSphere MQ ネイティブ プロトコルのような不透明なプロトコルでは、この技術は、会話の有意な情報を取得するただ 1 つの方法である場合があります。

注: ジェネリック XML ペイロード パーサと区切りテキスト データ プロトコルの両方を使用する場合は、ジェネリック XML ペイロード パーサの前に区切りテキスト データ プロトコルを追加します。 そうしない場合、レコーダに解析された要求が表示されません。

次の手順に従ってください:

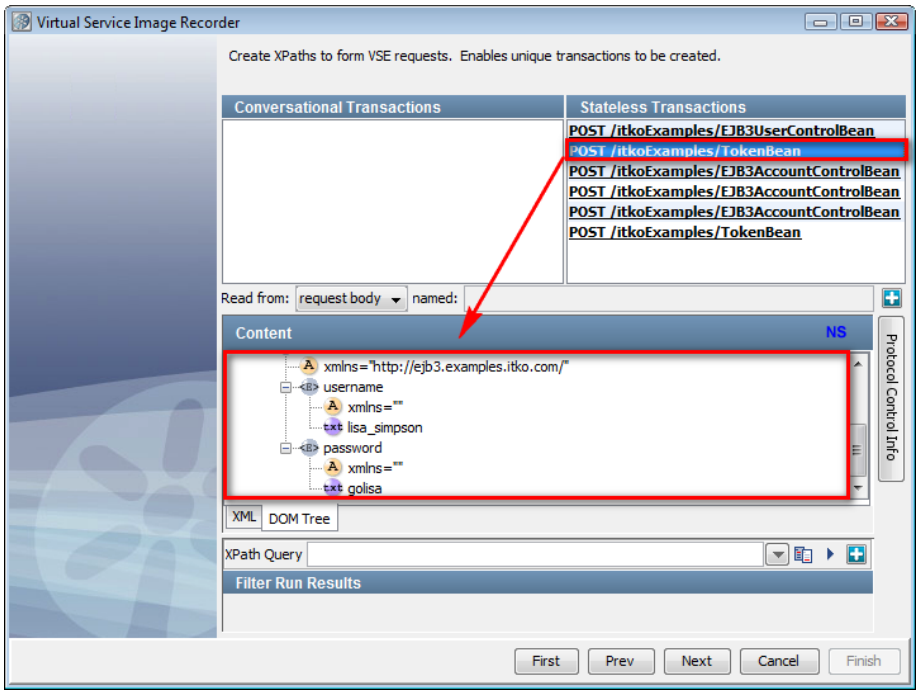
1. 動的データ プロトコルを使用するには、ペイロードが整形式 XML の場合、仮想サービス イメージ レコーダの［データ プロトコル］タブで［ジェネリック XML ペイロード パーサ］を選択します。
2. クリーンアップ ステップまで、残りのステップを実行します。

   クリーンアップ ステップの後、以下のウィンドウが表示されます。



   デフォルトでは、スタータ トランザクションは識別されません。そのため、DevTest は、どのデータを確認して会話を識別するのかわかりません。 ただし、［他のトランザクション］リストに、記録されたトランザクションが示されます。

3. ［コンテンツ］領域で XML ペイロードを表示するには、最初のトランザクションをクリックします。



［コンテンツ］領域の［XML］タブには、クラシック XML ビューが表示されます。［DOM ツリー］タブには、ペイロードがツリー形式で表示されます。

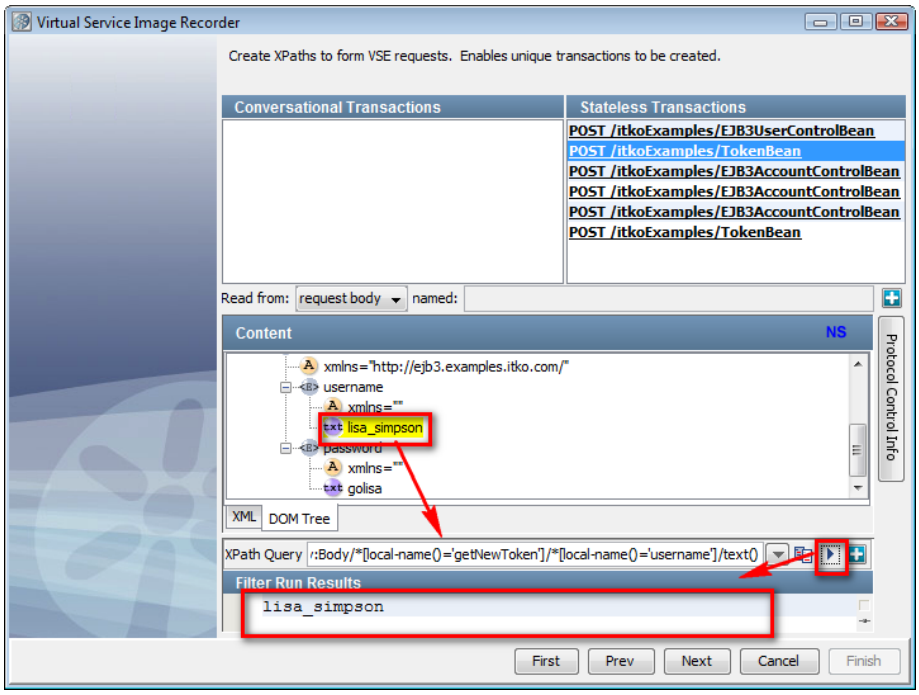
注: ウィンドウのこの部分は、XML XPath フィルタの作成時に表示されるパネルに似ています。「*使用*」の「XML XPath フィルタ」に説明されているように、**lisa.xml.xpath.computeXPath.alwaysUseLocalName** プロパティを使用してネームスペースを無視することができます。

4. VSE のパラメータとして特定のノードを指定するには、［DOM ツリー］タブのノードをクリックします。［XPATH クエリ］ボックスに、選択したノードに対応するクエリが表示されます。現在のペイロードに対して XPath クエリを評価するには、  をクリックします。

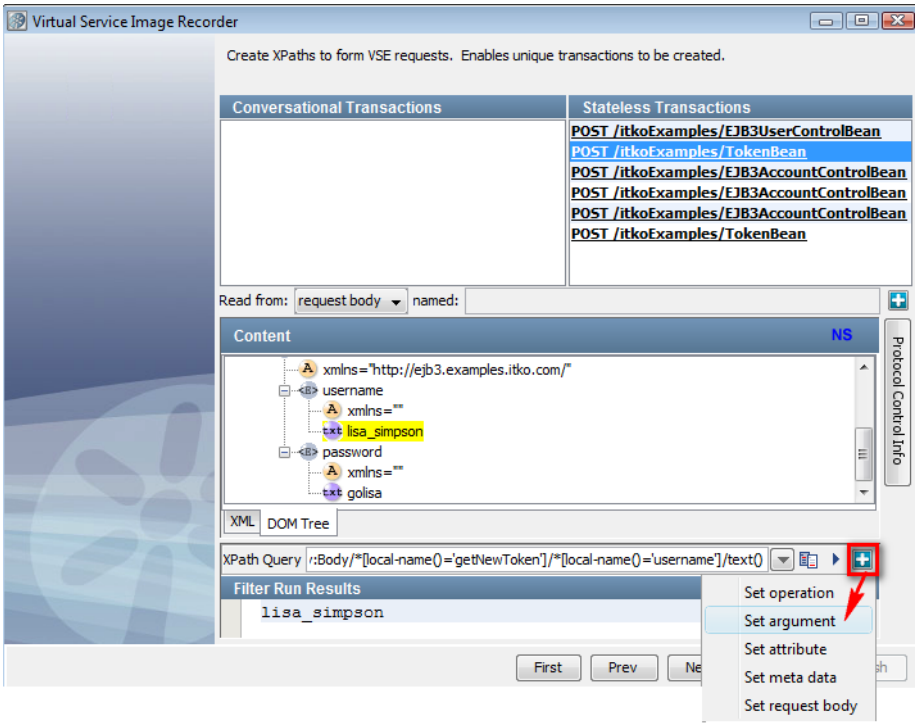
   評価結果が、［フィルタ実行結果］見出しの下に表示されます。

5. この XPath クエリをパラメータ リストに追加するには、［追加］  をクリックします。

   操作、引数、メタ パラメータ、または要求ボディを設定するメニューが表示されます。たとえば、このメニューを使用して要求ボディに埋め込まれた SOAP ドキュメントを選択し、WS SOAP プロトコルを使用してドキュメントを解析できるようにします。



6. ［プロトコル制御情報］をクリックします。指定した XPath 文字列を DevTest が記憶しており、それに引数名が指定されています。 引数名は変更できます。 また、このパネルの保存 ボタンおよびリストア ボタンを使用して、XPath のリストをファイルに保存するか、または XPath のリストをファイルからロードします。 保存およびロードによって、ある XML ペイロード パーサから別のパーサに容易にコピーして貼り付けることができます。



同様に、このトランザクションまたはほかのトランザクションから、その他の変数を取り出すことができます。すべてのトランザクションが特定の引数を持つ必要はありません。処理の時点で、ペイロードに特定の引数が存在しない場合、それは無視されます。

［プロトコル制御情報］パネルを右にスクロールすると、ネームスペースの定義を追加または編集できる［NS］列が表示されます。ファイルシステムで XML ファイルを検索してインポートし、そのファイルのネームスペース定義を使用するには、［参照］ボタンを使用します。

7. ［XML］タブでトランザクションをダブルクリックすると、ダイアログ ボックスにトランザクションのコンテンツが表示されます。
8. 変数の選択が完了したら、［次へ］をクリックします。次に、後処理ウィンドウが表示されます。

JSON データ プロトコル

JSON データ プロトコル ハンドラは、JSON データを同等の XML に変換するため、および XML データを JSON 形式に変換するために使用されます。

以下に、JSON の例を示します。

```
[
    {"organizationType":"W",
     "parentOrganizationType":"S",
     "parentUnitNbr":777,
     "positionName":"S",
     "positionType":"S",
     "positionTypeId":56,
     "unitNbr":433,
     "l":
            [{"Roles":["P","P1","P2","S","C","AC","ACF","ACM"]}],
     "groupKey":"P",
     "groupName":"P"
    }
]
```

以下のように変換されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <element class="object">
            <groupKey type="string">P</groupKey>
            <groupName type="string">P</groupName>
            <l class="array">
                    <element class="object">
                            <Roles class="array">
                                    <element type="string">P</element>
                                    <element type="string">P1</element>
                                    <element type="string">P2</element>
                                    <element type="string">S</element>
                                    <element type="string">C</element>
                                    <element type="string">AC</element>
                                    <element type="string">ACF</element>
                                    <element type="string">ACM</element>
                            </Roles>
                    </element>
            </l>
            <organizationType type="string">W</organizationType>
            <parentOrganizationType type="string">S</parentOrganizationType>
            <parentUnitNbr type="number">777</parentUnitNbr>
            <positionName type="string">S</positionName>
            <positionType type="string">S</positionType>
            <positionTypeId type="number">56</positionTypeId>
```

```
            <unitNbr type="number">433</unitNbr>
        </element>
    </root>
```

JSON データ プロトコル ハンドラでは、JSON に 1 つのエレメント配列が存在する場合、それは「element:」をキーとして持つオブジェクトに変換されます。 次に例を示します。

```
[{"question1":"answer1","question2":"answer2"}]
```

以下のように変換されます

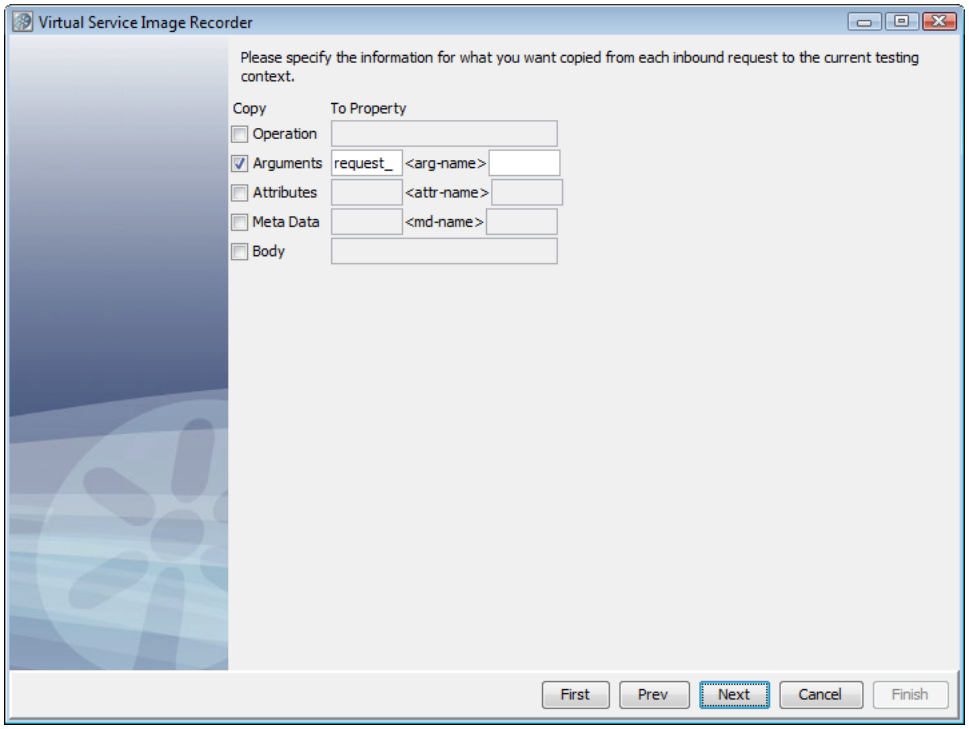
```
{"element":{"question1":"answer1","question2":"answer2"}}
```

JSON データ プロトコル ハンドラは、オブジェクト内のキー/値ペアを、キー名によってアルファベット順に並べ替えます。

注: アプリケーション タイプが、application/json、text/json、または text/javascript に正しく設定されている場合、JSON のレコーディングは正しく動作します。 間違ったアプリケーション タイプが設定されている場合、レコーディングは動作しません。

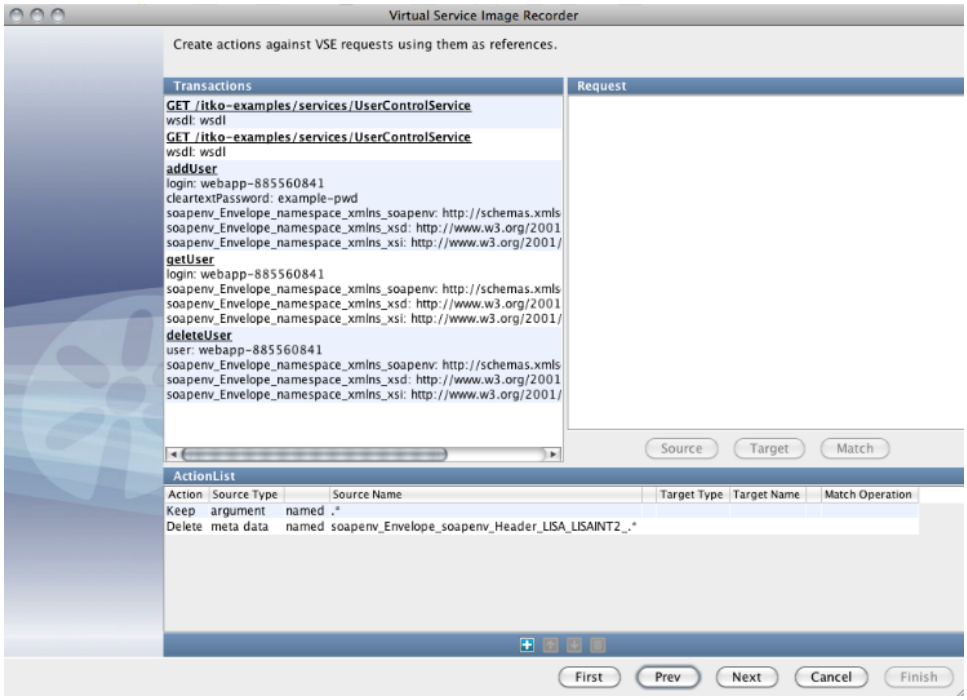
要求データ コピー

要求データ コピーでは、現在のインバウンド要求から現在のテスト コンテキストにデータをコピーできます。 この機能では、VS モデルの VSE 応答ルックアップ ステップの前にカスタム動作が必要な場合、より容易に要求情報を確認できます。



要求データ マネージャ データ プロトコル

［データ プロトコル］ウィンドウで、データ プロトコルに［要求データ マネージャ］を選択します。 レコーディングが完了すると、以下のウィンドウが表示されます。



要求データ マネージャ プロトコルでは、レコーディングまたは再生時に VSE 要求を変更できます。

基本的に、このプロトコルは、要求に対してアクションのリストを適用します。 このウィンドウの［ActionList （アクション リスト）］セクションで以下のアクションを追加できます。

コピー

要求のデータの一部を、要求の別の部分にコピーします。

移動

要求のデータの一部を、要求の別の部分に移動します。

削除

要求からデータの一部を削除（クリア）します。

保存

要求のデータの一部を保存し、そのグループのその他のデータ値を削除します。

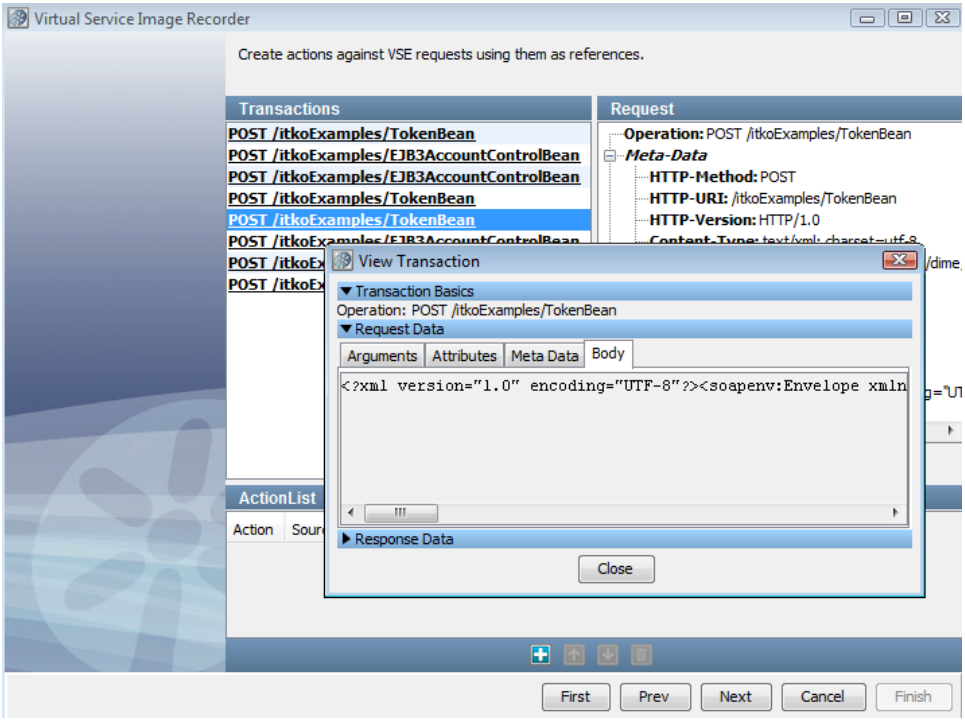
どのアクションも、要求操作の、任意の引数、属性、メタデータ エントリ、または要求ボディのすべてに適用できます。 たとえば、Java（引数として XML ドキュメントを取る）を仮想化する場合、要求ボディに引数の値を移動またはコピーできます。これにより、その他のデータ プロトコルで引数を処理できるようになります。

注: 引数、属性、およびメタデータには、「保存」アクションが最も大きな意味があります。これらの 3 つのグループの 1 つから特定の値を保存する場合、データ プロトコルのリスト内のいずれのアクションによっても参照されないそのグループ内のその他の値が削除されます。1 つの引数を保存する場合、移動またはコピーの対象でない限り、その他の引数が削除されます。 この技術は、無意味な引数を削除する良い方法です。

また、各アクションは、その操作が指定した正規表現に一致する要求にのみ適用するように制限できます。

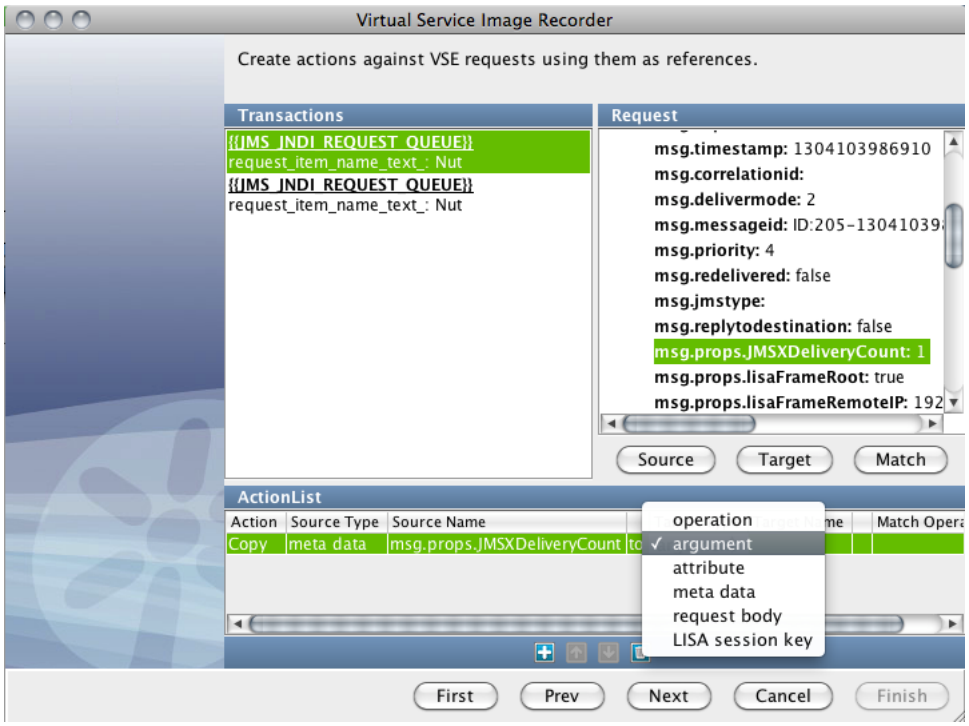
要求データ マネージャ DPH を含むレコーディング ウィザードまたはモデル エディタで、「保存」アクションまたは「削除」アクション、あるいはその両方を追加します。 ［引数］/［属性］/［メタデータ］を選択し、名前として一致する正規表現を指定します。 また、［*名前*］となっているセルを［*一致*］に変更する必要があります。 DPH が実行されると、名前がパターンに一致する引数、属性、またはメタデータ リストで、項目がすべて保存されるか削除されます。 アクションに関する操作一致パターンを空のままにすると、すべての要求に影響します。

このトランザクションのリストでトランザクションをダブルクリックすると、トランザクションのコンテンツを示すダイアログ ボックスが表示されます。



要求データ マネージャ データ プロトコルを使用した JMS および MQ メッセージ プロパティの設定

レコーディングが完了した後、要求データ マネージャ ウィンドウを使用して、要求引数にターゲットの JMS メッセージ プロパティを追加します。JMS メッセージ プロパティは、相関 ID などの標準 JMS プロパティに対する **msg.** プレフィックス、およびカスタム メッセージ プロパティに対する **msg.props.** プレフィックス を持つ要求メタ データの下にあります。 要求引数にプロパティをコピーするには、ドロップダウン リストから［引数］を選択します。



MQ は同じ方法で動作します。

引数の代わりにステートフル セッション キーを設定するには、ドロップダウンから［セッション キー］を選択します。

単一の要求データ マネージャ データ プロトコルを使用して、任意の数の引数とセッション キーを同時に設定できます。

REST データ プロトコル

REST は、HTTP プロトコルを使用して Web サービスを呼び出す方法です。

REST データ プロトコル ハンドラは、HTTP 要求が REST アーキテクチャ スタイルに従っていることを分析します。このデータ プロトコルは、URI 文字列の動的な部分を識別します。 結果は 1 セットのルールです。 再生中に、VSE は、同じ操作を呼び出す HTTP 要求を仮想化するルールを使用します。

VSE が自動的に REST データ プロトコルを選択しない場合、仮想サービス イメージ レコーダのデータ プロトコル ページで手動で選択できます。

ライブ トラフィックまたは要求/応答ペアに HTTP 要求を含めることができます。

要求/応答ペアに対して、HTTP/S トランスポート プロトコルを使用する必要があります。

分析プロセスの側面の一部を設定できます。

各ルールには、1 つ以上のパラメータが含まれます。これは、動的な部分を表します。 デフォルトでは、パラメータのテキストは **URLPARAM** で始まります。

以下のサンプル ルールには、**URLPARAM0** および **URLPARAM1** パラメータが含まれています。 パラメータは、ルール内で中括弧で囲む必要があります。 アナライザがルールを生成した後、**URLPARAM** 識別子を変更できます。

```
GET /Service/rest/user/{URLPARAM0}
GET /Service/rest/customer/{URLPARAM0}
GET /Service/rest/customer/{URLPARAM0}/order/{URLPARAM1}
```

再生では、以下の URI 文字列が最初のルールに一致します。

```
GET /Service/rest/user/100
GET /Service/rest/user/101
```

再生では、以下の URI 文字列が 3 番目のルールに一致します。

```
GET /Service/rest/customer/1234/order/5678
```

必要に応じて、パラメータとその他の固定テキストを混在させることもできます。 以下に例を示します。

```
GET /Service/rest/user/{USERNAME}/format.{type}
```

この例では、最後のトークンを「format.json」や「format.xml」とすることもできます。

任意の位置のトークンに、任意の数のパラメータを含めることができます。以下に例を示します。

```
GET
/Service/rest/users/format.{type}.sortorder.{order}.filter.{filter
}
```

レコーディング後に表示される REST ルール エディタを使用し、要件に応じてこのようなルールを追加または変更することができます。 このようなルールも、WADL ファイルまたは RAML ファイルから生成できます。

要求/応答ペアの形式

このトピックでは、REST データ プロトコルの要求/応答ペアの形式について説明します。

要求

要求ファイルには、有効な HTTP ヘッダが含まれている必要があります。URL はヘッダの最初の行です。 その他のヘッダ行はオプションです。

REST データ プロトコル ハンドラは、すべての HTTP メソッド/動詞（GET、HEAD、POST、PUT、DELETE、TRACE、OPTIONS、CONNECT、および PATCH）をサポートします。

URL 行の形式は次のとおりです。

<メソッド><空白文字><REST API パス><空白><HTTP バージョン>

以下に例を示します。

```
PUT /rest-example/control/users/save HTTP/1.1
```

要求にボディが含まれる場合は、空白行を使用してボディをヘッダから離します。

以下の例では、ボディが含まれる要求を JSON 形式で示します。

```
PUT /rest-example/control/users/save HTTP/1.1
accept: application/json
content-Type: application/json
Connection: Keep-Alive
User-Agent: LISA

   {
     "user": {
       "emailAddress": "test@test.com",
       "firstName": "first-9",
       "lastName": "last-9",
       "password": "aaaaaaaa",
       "username": "dmxxx-009"
     }
   }
```

JSON または XML ボディは、単一の行にまとめることができます。

```
{ "user": { "emailAddress": "test@test.com","firstName": "first-9",
"lastName": "last-9", "password": "aaaaaaaa", "username":
"dmxxx-009"  }}
```

応答

応答ファイルには、HTTP 応答コードが含まれます。 このファイルには、ヘッダおよびボディを含めることができます。 応答コードは、ファイル内の最初の行である必要があります。形式は以下のとおりです。

<HTTP バージョン><空白文字><HTTP 応答コード>

以下に、ボディがある応答の例を示します。

```
HTTP/1.1 200
```

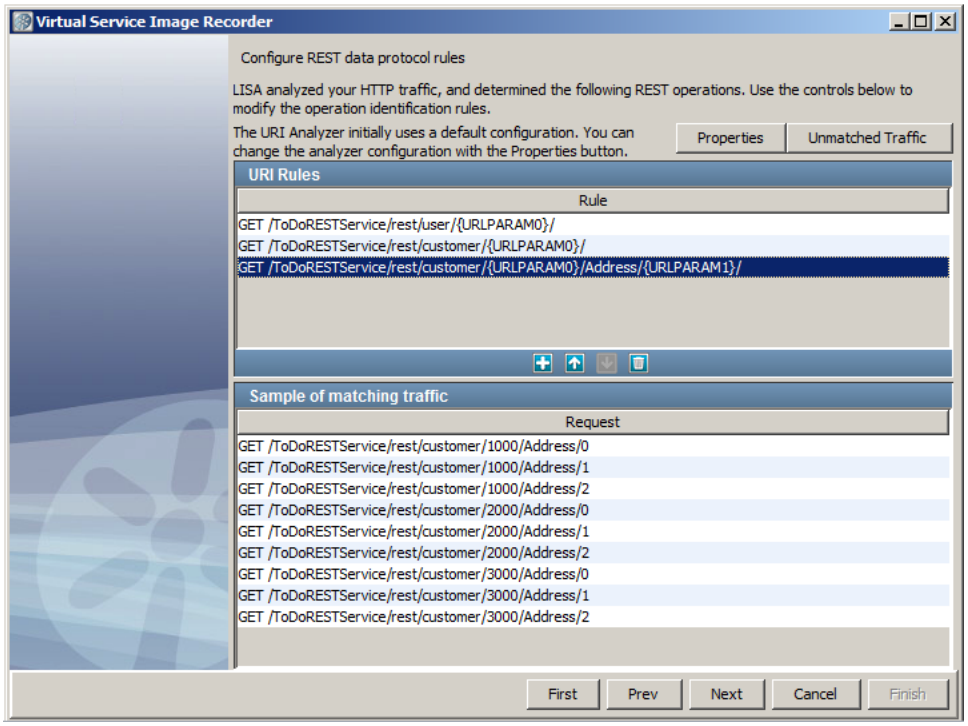
```
"user":{"emailAddress":"lisa.simpson@itko.com","firstName":"lisa",
"lastName":"simpson","password":"60fAFoq+W0R4HrLgsfPodkWRw9I=",
"phoneNumber":"","username":"lisa_simpson"}}
```

応答に応答コード行が含まれない場合、応答コードはデフォルトで **200**（**OK**）になります。

ルールの確認および変更

仮想サービス イメージ レコーダおよび［要求/応答ペアからの仮想サービス］インターフェースには、REST データ プロトコルが作成したルールを確認および変更できるページが含まれます。

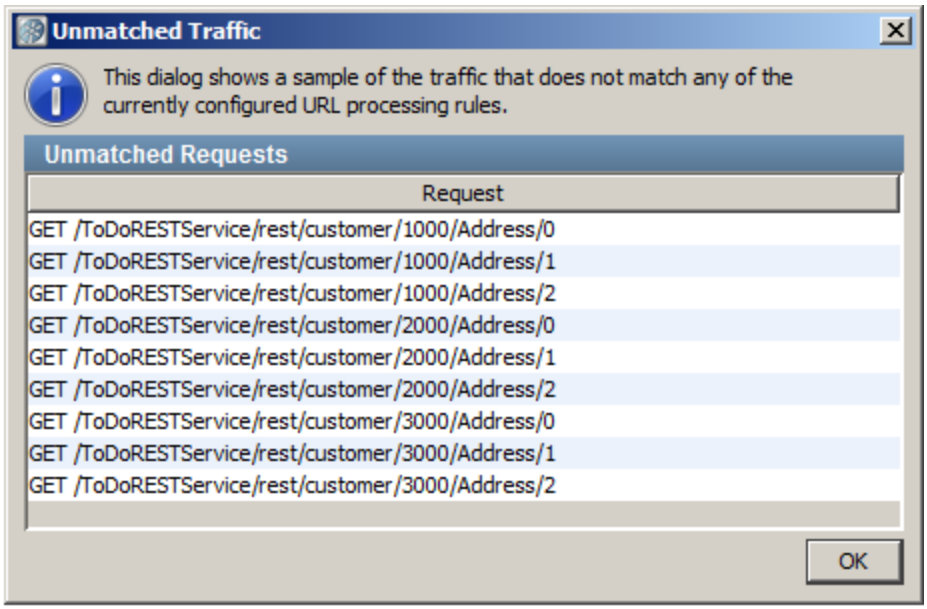
以下の図は、仮想サービス イメージ レコーダのウィンドウを示しています。



上部ペイン（［URI ルール］）には、ルールが表示されます。 ルールを選択すると、下部ペイン（［一致するトラフィックのサンプル］）にルールに一致するトラフィックのサンプルが表示されます。 表示される行の最大数を設定するには、**lisa.properties** ファイル内の **lisa.protocol.rest.editor.observedtraffic.max** プロパティを設定します。

ルールに一致しない HTTP 要求のリストを表示するには、［一致しないトラフィック］をクリックします。 ルールが収集されたトラフィックすべてに一致する場合、このリストは空です。 表示される要求の最大数を設定するには、**lisa.properties** ファイル内の **lisa.protocol.rest.editor.unmatchedtraffic.max** プロパティを設定します。

［URI ルール］ペインでは、ルールを追加、更新、並べ替え、および削除できます。



ルール内のパラメータをもっと意味のある名前に置換できます。 以下に例を示します。

```
GET /Service/rest/customer/{customerid}/order/{orderid}
```

同じ操作に一致する複数のルールを作成できます。 ルールは表示された順に照合されます。 そのため、リストの下の位置にあるルールが一致すると予想しているときに、リストの上の位置にあるルールが一致する可能性があります。 順序を変更するには、並べ替えボタンを使用します。

ルールを削除する場合は、［一致しないトラフィック］をクリックして、キャプチャされたトラフィックに対するルールを削除した場合の影響を確認します。

以下の設定プロパティの値を変更するには、［プロパティ］をクリックします。

［プロパティ］をクリックすることにより、トラフィックの再分析を実行できます。［プロパティ］ページが表示されたら、値を変更せずに［OK］をクリックします。 たとえば、スクリプタブル データ プロトコルと REST データ プロトコルをチェーン化した後、これを実行する場合があります。

最大変更数

この数を超えると変動が大きすぎるのでルールを生成する必要があると認識される、トークンに対して許可される変更の最大数を定義します。

URI を「/」文字で区切られた「トークン」のリストと考えます。 たとえば、「GET /rest/user/1234」という URI には以下のトークンが含まれます。

- GET
- rest
- user
- 1234

デフォルト値を変更するには、**lisa.properties** ファイル内の **lisa.protocol.rest.maxChanges** プロパティを設定します。

開始位置

REST データ プロトコルが変数トークンの検索を開始する URL 内の位置を定義します。

開始位置は、URI を構成しているトークンのリスト内のトークンのインデックスです。 たとえば、以下の URI があるとします。

```
GET /service/rest/customers"
```

「GET」は位置 0 にあります。また、「customers」は位置 3 にあります。

デフォルト値を変更するには、**lisa.properties** ファイル内の **lisa.protocol.rest.startPosition** プロパティを設定します。

ID 識別の正規表現

HTTP 要求のリソース識別子を検出するには、REST データ プロトコルが使用する正規表現文字列を定義します。デフォルト値を変更するには、**lisa.properties** ファイル内の **lisa.protocol.rest.idPattern** プロパティを設定します。

URL パラメータ プレフィックス

REST データ プロトコルがルールのパラメータに使用するプレフィックスを定義します。

この設定の目的は、ユーザは変数であることを知っていても、アナライザが自動検出しない可能性がある特定のパターンに従うトークンをアナライザが識別できるように支援することです。

たとえば、以下の URI があるとします。

```
GET /rest/user/person-1234-dev
```

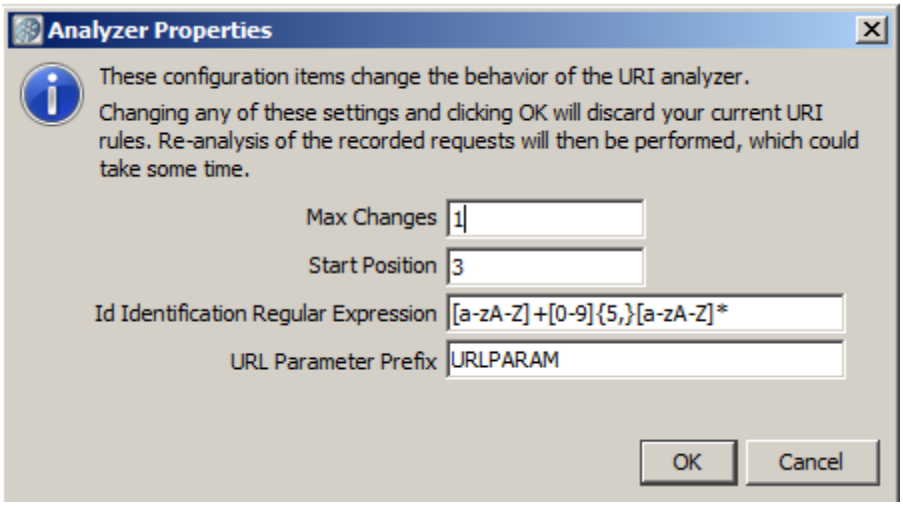
ユーザは、*person-nnnn-nnn* という形式のユーザ ID が必ず「user」の後に続くことを知っています。 この場合、このパターンを直接検出する正規表現を定義できます。この例では、正規表現は以下のとおりです。

```
person-[0-9]{4}-[a-z]{3}
```

デフォルト値を変更するには、**lisa.properties** ファイル内の **lisa.protocol.rest.parameterBaseName** プロパティを設定します。

これらのプロパティの 1 つ以上の値を変更した場合、このデータ プロトコルは記録されたトラフィックを再分析します。

以下の図は、［アナライザ プロパティ］ウィンドウを示しています。



スクリプタブル データ プロトコル

スクリプタブル データ プロトコルは、要求、応答、またはその両方に対して少しの処理が必要な状況で使用できます。 サンプル スクリプトは BeanShell で記述されています。スクリプトは、CA Application Test がサポートする任意のスクリプト言語で作成できます。

スクリプト用に BeanShell 以外の言語を指定するには、スクリプトの最初の行に言語を入力します。

値

- applescript （OS X 用）
- beanshell
- freemarker
- groovy
- javaScript
- velocity

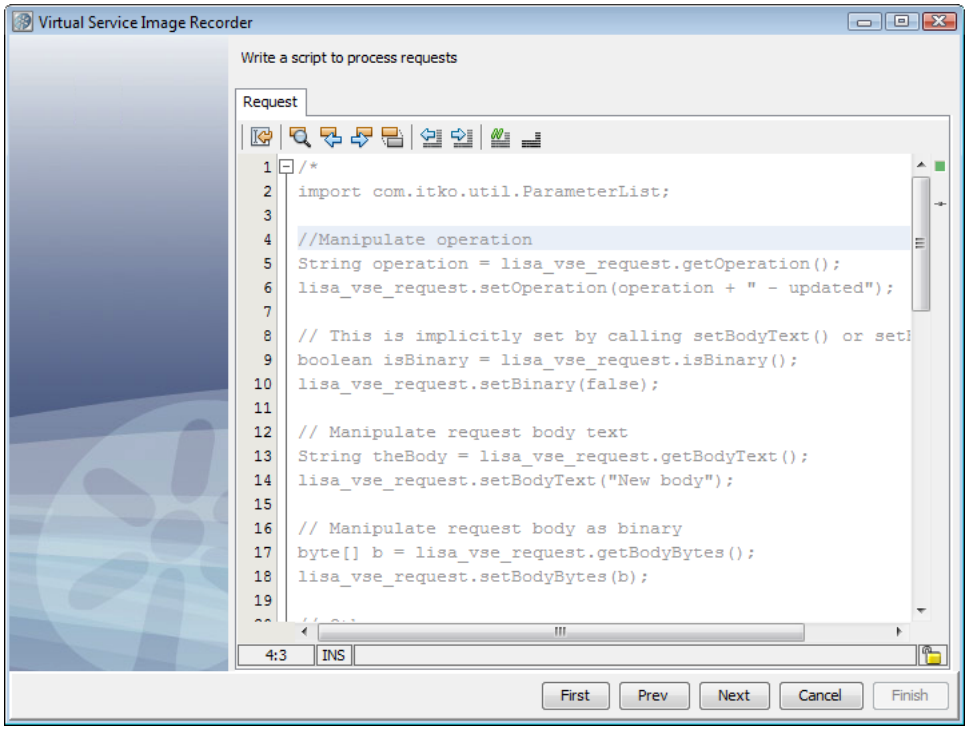
フォーマット：

%language%

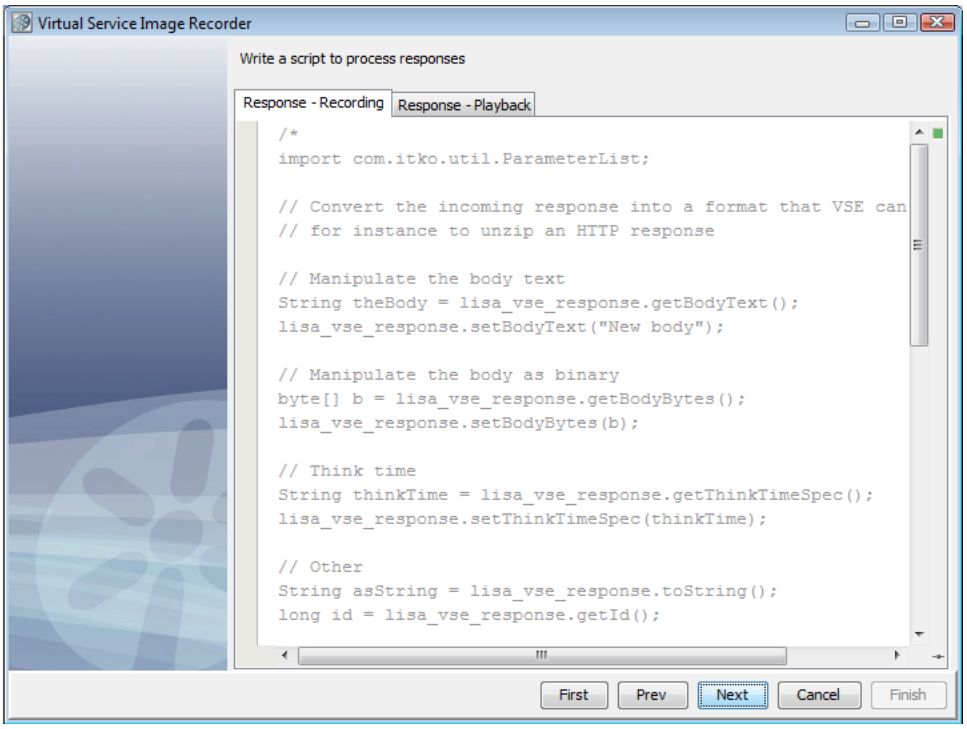
デフォルト： beanshell

追加のスクリプト言語を使用するには、「追加のスクリプト言語の有効化」を参照してください。

要求側でスクリプタブル データ プロトコルを指定すると、以下のウィンドウが表示されます。



応答側では、レコーディング用と再生用に 2 つのスクリプトを使用できます。「応答 - レコーディング」スクリプトを使用して、記録された応答を、VSE が処理できる形式に変換できます。 次に、処理された後、「応答 - 再生」スクリプトを使用して、テスト中のシステムが予期している形式に戻すことができます。



独自のスクリプトを追加して、要求、応答、またはその両方に対して任意のアクションを実行できます。

SWIFT データ プロトコル

SWIFT データ プロトコルは、以下のアクションを実行します。

- SWIFT メッセージを XML 表現に変換します
- SWIFT メッセージの XML 表現をそのネイティブ形式に変換します

たとえば、このデータ プロトコルは以下の SWIFT メッセージを変換します。

```
{1:F01BANKDEFMAXXX2039063581}{2:O1031609050901BANKDEFXAXXX89549829
458949811609N}{3:{108:00750532785315}}{4:

:16R:GENL

:20C::SEME//YOUR REFERENCE

:16S:GENL

:16R:SETDET

:22F::SETR//TRAD

:16R:SETPRTY

:97A::SAFE//YYYY

:16S:SETPRTY

:16S:SETDET

-}
```

以下の XML に変更します。

```
<message>
<block1>
   <applicationId>F</applicationId><serviceId>01</serviceId>
   <logicalTerminal>BANKDEFMAXXX</logicalTerminal>
   <sessionNumber>2039</sessionNumber>
   <sequenceNumber>063581</sequenceNumber>
</block1>
<block2 type=¥"output¥">
   <messageType>103</messageType>
   <senderInputTime>1609</senderInputTime>
   <MIRDate>050901</MIRDate>
   <MIRLogicalTerminal>BANKDEFXAXXX</MIRLogicalTerminal>
   <MIRSessionNumber>8954</MIRSessionNumber>
   <MIRSequenceNumber>982945</MIRSequenceNumber>
   <receiverOutputDate>894981</receiverOutputDate>
   <receiverOutputTime>1609</receiverOutputTime>
```

```
   <messagePriority>N</messagePriority>
</block2>
<block3>
   <BLOCK3_108>00750532785315</BLOCK3_108>
</block3>
<block4>
   <GENL>
      <GENL_20C>:SEME//YOUR REFERENCE</GENL_20C>
   </GENL>
   <SETDET>
      <SETDET_22F>:SETR//TRAD</SETDET_22F>
      <SETPRTY>
         <SETPRTY_97A>:SAFE//YYYY</SETPRTY_97A>
      </SETPRTY>
   </SETDET>
</block4>
</message>
```

SWIFT メッセージの一部のフィールドに日付が含まれる場合、このデータ プロトコルは DevTest がマジック デートの候補として識別できる形式で日付を再フォーマットします。

以下の例は、このデータ プロトコルが日付を確認するフィールドを示しています。

:98A::SETT//**19911130**

:98C::TRAD//**20140117125901**

:98E::PREP//**20091107093238,02/N0230**

:32A:**870902**JPY3520000,

:30:**640123**

このデータ プロトコルは、これらの行を以下の XML に変換します。

<BLOCK4_98A>:SETT//**1991-11-30**</BLOCK4_98A>

<BLOCK4_98C>:TRAD//**2014-01-17 12:59:01**</BLOCK4_98C>

<BLOCK4_98E>:PREP//**2009-11-07T09:32:38.020-0230**</BLOCK4_98E>

<BLOCK4_32A>**1987-09-02** JPY3520000,</BLOCK4_32A>

<BLOCK4_30>**2064-01-23**</BLOCK4_30>

サービス イメージには、マジック デートとして表される日付が含まれます。

SWIFT 通信

SWIFT トランザクション用のセッション キーを生成するために、SWIFT データ プロトコルはメッセージ ボディから情報を抽出します。 このプロトコルは、SWIFT メッセージの参照 (mesg_ref) および処理の参照 (deal_ref) を抽出する以下のルールを使用し、それらを組み合わせてセッション キーを生成します。

1. フィールド 70E がフォーム **:SPRO///xxxREF/<mesg_ref>/<deal_ref>** で見つかった場合、それらはここから抽出されます。
2. フィールド 20C がフォーム **:RELA//<mesg_ref>** で見つかり、2 番目のフィールド 20C がフォーム **:TRRF//<deal_ref>** で見つかった場合、それらはここから抽出されます。
3. フィールド 26H がフォーム **<msg_ref>/<deal_ref>** で見つかった場合、それらはここから抽出されます。

このデータ プロトコルは、順番にこれらのルールを適用します。 それらの 1 つが満たされた場合、フォーム **<message ref>/<deal ref>** でセッション キーを作成します。 ルールのどれも満たされない場合、このデータ プロトコルはトランザクションをステートレスとして処理し、セッション キーを作成しません。

Web サービス ブリッジ データ プロトコル

Web サービス ブリッジ データ プロトコルは、サンプルの DevTest Travel 専用です。 このプロトコルはこのサンプルに固有であり、一般的なケースには役立ちません。 サンプルの DevTest Travel 以外では、このプロトコルは無視してもかまいません。

Web サービス（SOAP）

Web サービス SOAP データ プロトコル ハンドラは、XML 形式の SOAP ドキュメントを要求の適切な操作/引数タイプに変換するために使用されます。

データ プロトコルに提示された各要求では、テキスト形式の要求ボディを XML ドキュメントとして解析するよう試行されます。 ボディが有効な XML ドキュメントの場合で、トップレベルのエレメントが**エンベロープ**の場合、子エレメントの**ヘッダ**と**ボディ**の両方が検索されます。

SOAP エンベロープにヘッダ エレメントが含まれる場合、データ プロトコルは、ヘッダ内の**返信先**エレメントを検索します。 次に、そのエレメント下で、このデータ プロトコルは**アドレス** エレメントを確認します。 そのようなアドレス エレメントが存在する場合、その値は、現在の VSE 要求に対するメタデータ リストに「**lisa.vse.reply.to**」という名前で設定されます。

SOAP エンベロープにボディ エレメントが含まれる場合、その最初の子のタグ名が VSE 要求の操作名になります。何らかの理由で操作を特定できない場合、現在の要求に対して以下のいずれのアクションも発生しません。

操作エレメントにいずれかの XML 属性が含まれる場合、これらの属性が、現在の VSE 要求の属性リストに追加されます。

その後、操作エレメントの下の XML エレメントのツリー全体が確認されます。子エレメントを持たないすべてのエレメントが、引数として VSE 要求に追加されます。 各引数の名前は、すべての親エレメント タグ（操作エレメントまで）を使用して構成することにより、その一意性が確保されます。同じ名前が 2 回以上出現する場合は、数値のサフィックスが追加されます。 数値のサフィックスは、配列の構造体を示し、特定の引数の一意性を確保します。

最後に、元の XML ドキュメント（要求ボディ）が、「**recorded_raw_request**」という名前の VSE 要求内の新しい属性にコピーされます。

このデータ プロトコル ハンドラは設定情報を必要としないため、レコーディング ウィザードではページが表示されません。

Web サービス（SOAP ヘッダ）

SOAP ヘッダ データ プロトコル ハンドラは、SOAP メッセージのヘッダのエレメントを、仮想サービス イメージ内の要求の引数に変換します。 このデータ プロトコルは、SOAP メッセージを生成する任意のトランスポート プロトコル（通常は HTTP/S）と互換性があります。

SOAP ヘッダ データ プロトコルを使用するには、レコーディングまたは要求/応答ペアから VS イメージを生成します。 通常、HTTP/S トランスポート プロトコルを使用します。要求側データ プロトコルとして、SOAP ヘッダ データ プロトコルを追加します。 このデータ プロトコルは、追加の設定を必要としません。

このデータ プロトコルは、SOAP ヘッダと SOAP ボディの両方の処理に SOAP データ プロトコルと共に使用できます。

引数には、XML の構造に基づいて名前が付けられます。

例

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header>
<wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurit
y-secext-1.0.xsd">
<wsse:UsernameToken>
<wsse:Username>username</wsse:Username>
<wsse:Password>password</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
<n1:ServiceControl xmlns:n1="http://localhost:8080/examples.xsd">
<n1:VersionID>2.0</n1:VersionID>
<n1:Asynchronous>
<n1:ReplyRequiredIndicator>false</n1:ReplyRequiredIndicator>
<n1:PassThroughData>
<n1:Key>InteractionID</n1:Key>
<n1:Value>444831</n1:Value>
</n1:PassThroughData>
</n1:Asynchronous>
</n1:ServiceControl>
</soapenv:Header>
...
```

この例は、以下の名前のエレメントに解析されます。

- Security_UsernameToken_Username

- Security_UsernameToken_Password
- ServiceControl_VersionID
- ServiceControl_Asynchronous_ReplyRequiredIndicator
- ServiceControl_Asynchronous_PassThroughData_Key
- ServiceControl_Asynchronous_PassThroughData_Value

重複したエレメントには、名前に _1、_2 などが追加されます。

WS セキュリティ要求データ プロトコル

WS セキュリティ要求データ プロトコルは、WS-Security ヘッダを含む SOAP メッセージをサポートします。 このデータ プロトコルは、仮想化フレームワークに従って SOAP 要求を送信する前にすべてのセキュリティを解除します。 次に、WS セキュリティ要求データ プロトコルは、送信する SOAP 応答にセキュリティを適用します。

WS-Security ヘッダを持つ Web サービスを記録する場合、WS セキュリティ要求（要求側）データ プロトコル（通常は Web サービス SOAP データ プロトコルの前）と WS セキュリティ応答（応答側）データ プロトコルを追加します。

記録する前に、設定パネル セットが表示されます。

- WS セキュリティ要求用に 1 つ
- WS セキュリティ応答用に 2 つ

要求データ プロトコル

要求データ プロトコルに対して、クライアントが送信する要求メッセージを処理するハンドラを設定します。 ヘッダのデコードおよび検証に使用する受信アクションを入力します。 この設定は、レコーディングと再生の両方に使用されます。

受信（応答）メッセージに使用可能なオプションは、以下のとおりです。

- 復号化
- 署名検証
- SAML 検証
- ユーザ名トークン検証
- タイムスタンプ受信
- シグネチャ確認

送信（要求）メッセージに使用可能なオプションは、以下のとおりです。

- タイムスタンプ
- ユーザ名トークン
- SAML トークン
- シグネチャ トークン

- 暗号化

暗号化を使用するには、以下の情報を入力します。

キーストア ファイル

暗号化に使用するキーストア ファイルをドロップダウンから選択するか、またはファイル システムから選択します。

キーストア タイプ

- ［Java キー ストア］または［Personal Information Exchange（PKCS #12）］のいずれかを選択します。

キーストア パスワード

指定したキーストア ファイルに関連付けられたパスワードを定義します。

キーストア エイリアス

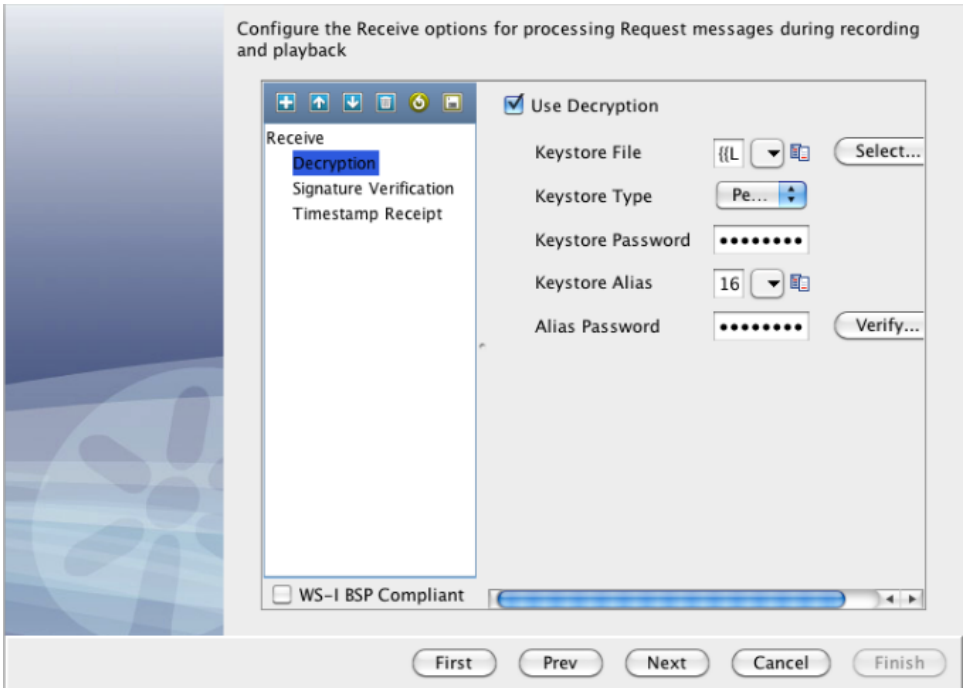
公開鍵のエイリアス。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同様に入力します。

［WS-I BSP 準拠］チェック ボックスは、WS-I Basic Security Profile への準拠を検証するかどうかを示します（SignedInfo の InclusiveNamespaces と CanonicalizationMethod の使用を含む）。

キーストア情報を検証するには、［検証］ボタンをクリックします。



応答データ プロトコル

応答データ プロトコルに対しては、レコーディング中にライブ サービスから返信される応答メッセージ、および VSM から返信される応答メッセージを処理するようにハンドラを設定します。 レコーディング段階では、クライアントが実行するように応答 メッセージを処理する必要があります。

［タイムスタンプの追加］チェック ボックスをオンにします。

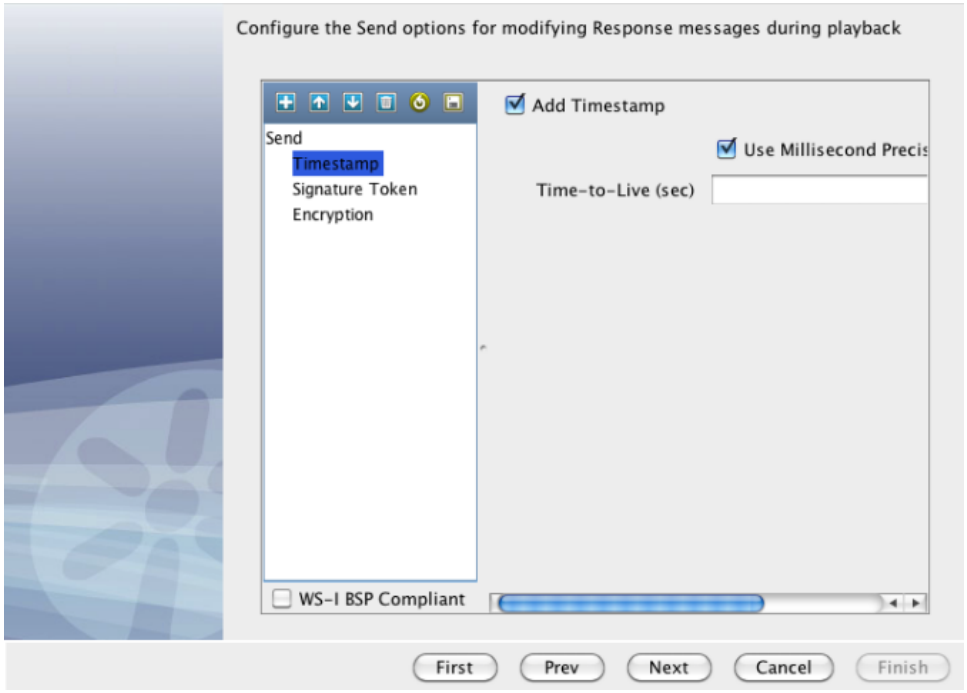
有効期間（秒）

メッセージの有効期間を秒単位で定義します。 有効期限エレメントを含まないようにするには、「**0**」を入力します。

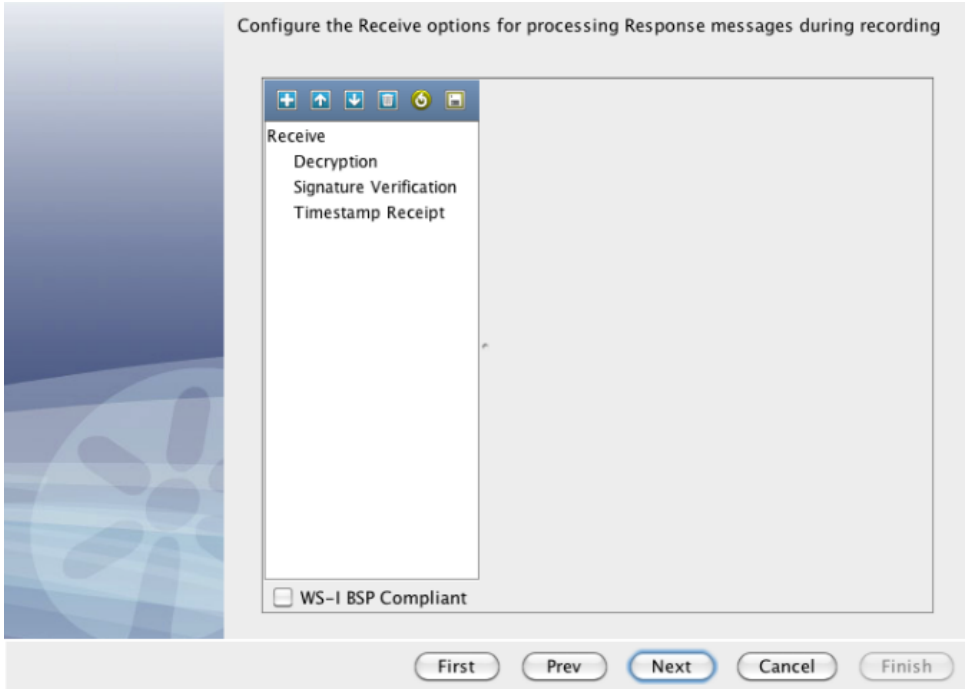
タイム スタンプでミリ秒を使用する

タイム スタンプをミリ秒単位で出力するかどうかを指定します。

注: 一部の Web サービス（WSE 2.0 を使用する .NET 1.x/2.0 など）は標準のタイムスタンプ形式に準拠しておらず、ミリ秒を使用できません。 これらの Web サービスについては、［タイムスタンプでミリ秒を使用する］チェック ボックスをオフにします。

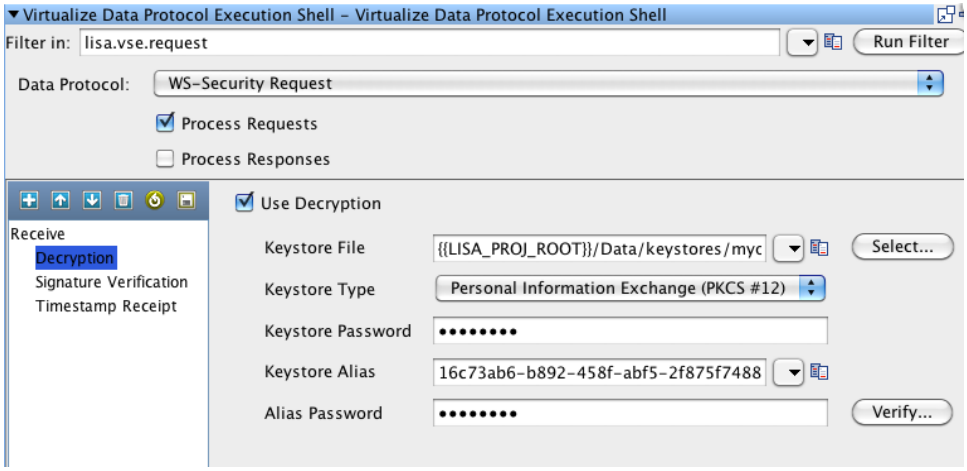


再生中には、サーバが送信するのに伴ってメッセージを処理する必要があります。 VSM からの SOAP メッセージには、セキュリティ ヘッダがありません。この設定は、セキュリティ ヘッダを適用します。



レコーディングが完了した後、仮想サービス モデルが作成されます。 このモデルでは、WS セキュリティ要求データ プロトコルの HTTP/S リスン ステップに、データ プロトコル フィルタが追加されています。

再生用のすべてのセキュリティ設定情報を更新します。 たとえば、サービスで WS セキュリティ設定が異なる場合、仮想サービスを再度記録する代わりに、ここでそれらを更新できます。



VSM では、WS セキュリティ応答データ プロトコルの HTTP/S 応答ステップにもフィルタが追加されます。

再生用のすべてのセキュリティ設定情報を応答メッセージ用に更新できます。

ファイルにセキュリティ設定を保存するか、またはセキュリティ設定が含まれる保存ファイルをロードするには、［ロード］ および［保存］ を使用します。

XML

XML データ プロトコル ハンドラは、XML ドキュメントを要求の適切な操作/引数タイプに変換するために使用されます。

このデータ プロトコル ハンドラは、Web サービス（SOAP） (P. 305)データ プロトコルと同様に動作します。 このデータ プロトコルは設定情報を必要としないため、レコーディング ウィザードではウィンドウが表示されません。

第 9 章: サービス イメージの編集

このセクションには、以下のトピックが含まれています。

レガシー サービス イメージ

LISA 6.0 から、サービス イメージはデータベースに格納されなくなりました。 LISA 6.0 以降で LISA 5.0 サービス イメージを使用する必要がある場合は、LISA 5.0 を使用してそれらをエクスポートします。 次に、後継の LISA バージョンのプロジェクト ツリー コンテキスト メニューで、［インポート］を使用してそれらをインポートします。

注: LISA 5.0 でエクスポートされたサービス イメージの拡張子は **.xml** です。このエクスポートされたバージョン 5.x のサービス イメージは、拡張子を **.vsi** に変更することにより、LISA 6.0 以降の有効なサービス イメージに変更できます。

編集するサービス イメージを開きます

仮想サービス イメージ レコーダはサービス イメージを生成します。サービス イメージは、ユーザが記録したもの（記録された RAW トラフィックの操作または変更されたバージョン）のように振る舞います。

LISA 6.0 より前の VSE リリースのサービス イメージが存在する場合は、それらのサービス イメージをエクスポートします。 エクスポートを行うことにより、それらが以前のバージョンのデータベースから、現在のリリースで格納されるファイル システムに移動します。 詳細については、「レガシー サービス イメージ (P. 313)」を参照してください。

サービス イメージを開くと、サービス イメージ エディタでイメージを変更できます。

次の手順に従ってください:

1. プロジェクト パネルでサービス イメージを右クリックし、［開く］を選択します。

   サービス イメージ エディタが開きます。

2. 選択したサービス イメージを確認し、必要な変更を行います。

注: サービス イメージ名の横の［開く］をクリックすることにより、VSM の応答選択ステップからエディタにアクセスすることもできます。

[サービス イメージ]タブ

サービス イメージ エディタの［サービス イメージ］タブには、以下のフィールドがあります。

イメージ名

現在のサービス イメージ名。

作成日

サービス イメージが作成された日時。

最終変更日

サービス イメージが最後に変更された日時。

メモ

サービス イメージに関するドキュメント。

推定メモリ使用量

サービス イメージに必要な推定メモリ量。

不明な会話型要求用の応答

再生での不明な会話型要求に対する応答用のボディ、メタデータ、および反応時間を詳細に設定します。

不明なステートレス要求用の応答

再生での不明なステートレス要求に対する応答用のボディ、メタデータ、および反応時間を詳細に設定します。

このウィンドウの特定エレメントの詳細については、以下を参照してください。

- 不明な要求に対する応答の編集 (P. 316)
- 応答エディタのカスタマイズ (P. 318)

不明な要求に対する応答の編集

矢印アイコン  を使用して、パネルを最大サイズにズームします。 元のサイズに戻すには、同じアイコンを使用します。

応答パネルで、必要に応じて以下のフィールドに入力します。

ボディ

再生中に不明なステートレス要求に対して返される応答が含まれています。

メタ データ

［メタ データ］領域では、必要に応じて下部のツールバーを使用して、キー値ペアを追加、移動、または削除します。

反応時間

必要な反応時間をミリ秒単位で定義します。 *反応時間*とは、要求に対して応答を送信するまでにかかる時間です。

値： ミリ秒での数値または数値の範囲。

範囲を入力すると、アプリケーションによって反応時間がその範囲からランダムに選択されます。

数値にサフィックスを追加すると、時間の単位を指定できます。大文字と小文字は区別されません。 有効なサフィックスは以下のとおりです。

- **t：** ミリ秒
- **s：** 秒
- **m：** 分
- **h：** 時

デフォルト： 0

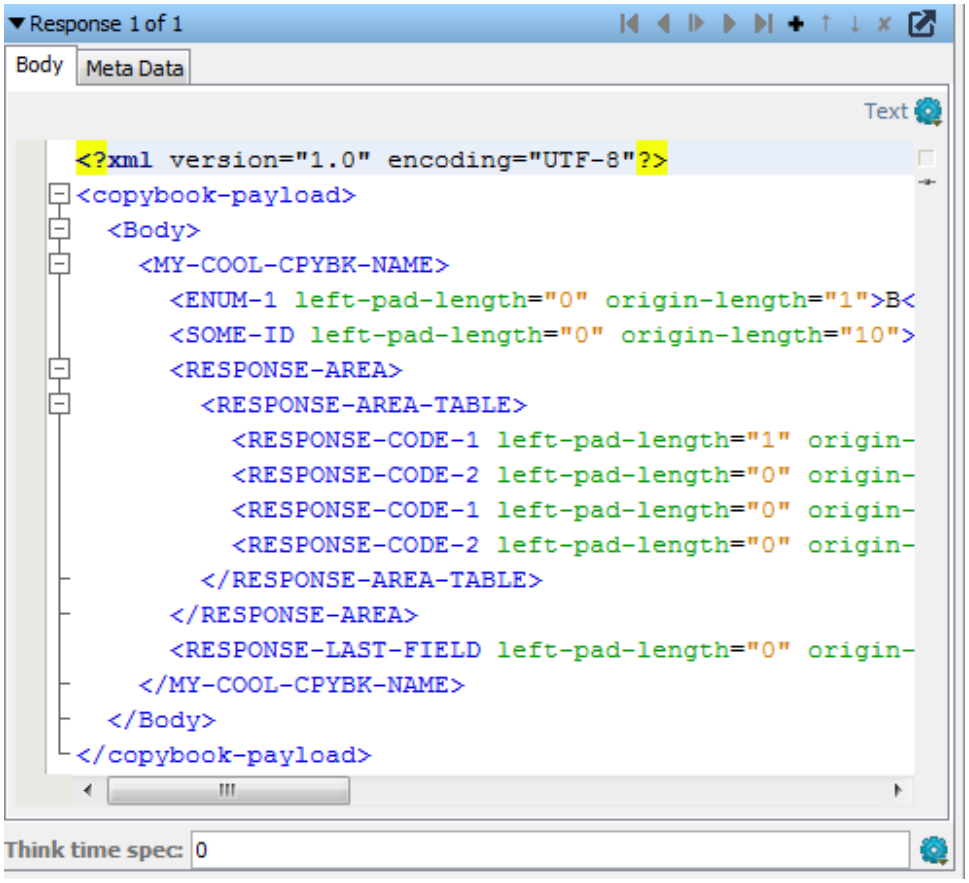
例：

- 「100」は 100 ミリ秒の反応時間を示します。
- 「100s」は 100 秒の反応時間を示します。
- 「100-1000」は 100 ～ 1000 ミリ秒の間のランダムな反応時間を示します。
- 「10t-5s」は、10 ミリ秒から 5 秒の間のランダムな反応時間を示します。

注: テストの実行でのペースの整合性を保つため、ステップ自体の処理時間は反応時間から引かれます。

応答エディタのカスタマイズ

応答エディタをカスタマイズするには、パネルの右下隅の歯車アイコン  を使用します。



応答エディタ メニューのオプションには、以下のものが含まれます。

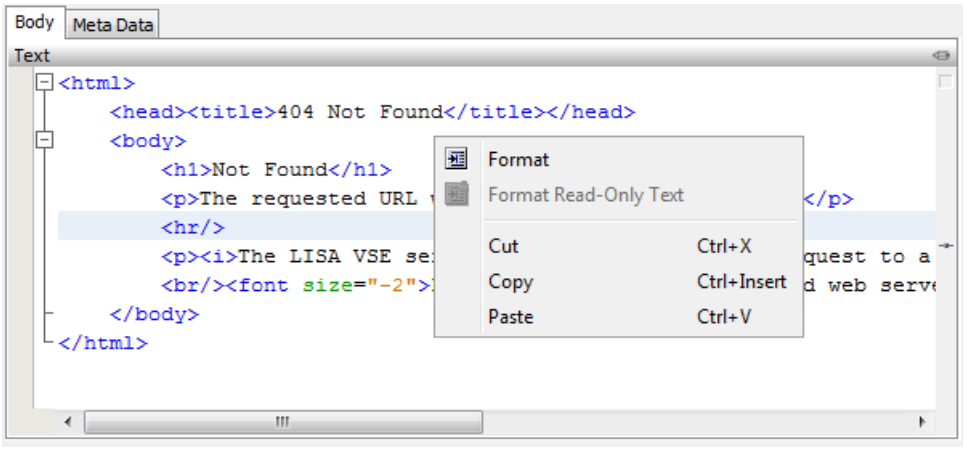
参照プロトコルの設定

［プロトコルを指定しない］または［JDBC（ドライバ ベース）］。

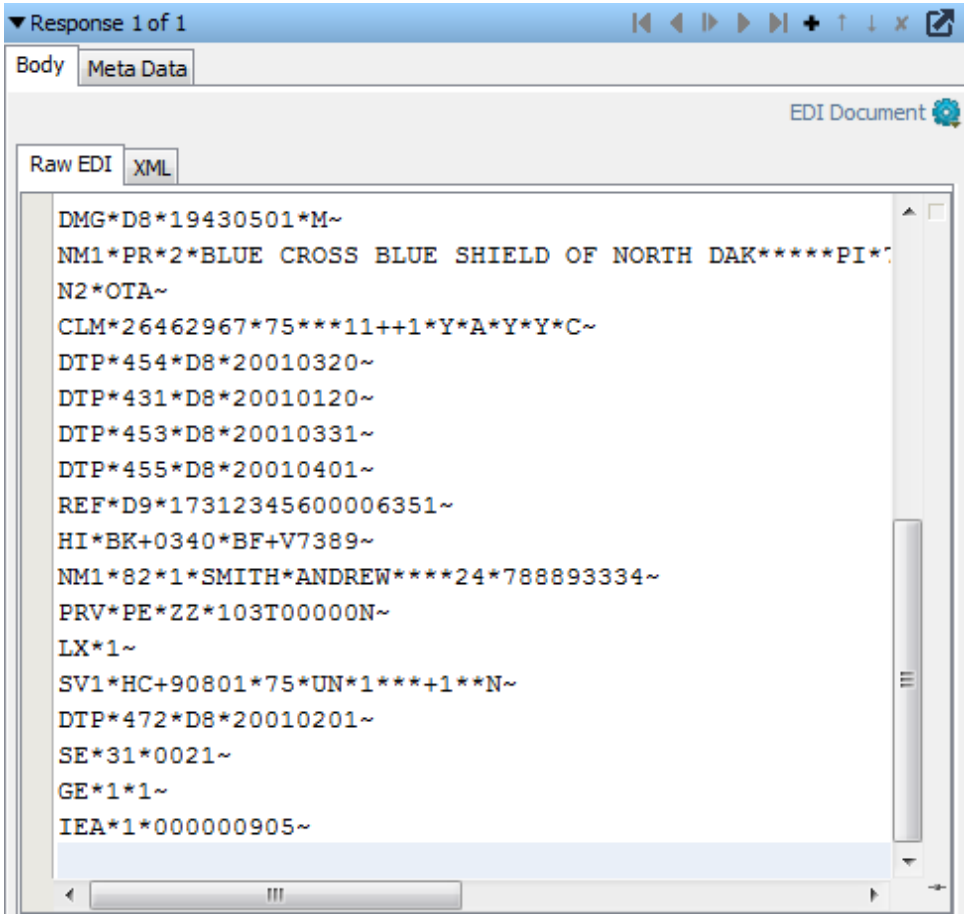
応答エディタの［ボディ］タブのタイトル バーは、どのカテゴリの応答が返されたかを示します。上記の図は、テキスト応答を示しています。その他のカテゴリの応答ペイロードには、XML、JSON、文字列、長い文字列、非常に長い文字列、グラフィック イメージ、および RAW バイトがあります。

別のエディタを使用するか、またはペイロードのタイプを変更するには、パネルの右上隅の歯車アイコンを選択します。 現在選択しているエディタに基づいて、その他の有効なエディタのみを示すメニューが動的に構築されます。

テキストボディエディタが[テキスト]に設定されている場合、応答ボディ XML を右クリックして、応答テキストを整形します。



応答ペイロードが EDI データであることが検出されると、［XML］タブで XML に変換されます。



[トランザクション]タブ

サービス イメージ エディタから［トランザクション］タブにアクセスできます。このタブには、それぞれ少しずつ異なるコンポーネントと共に、ステートレスおよびステートフル（会話）トランザクションに関する情報が表示されます。 このセクションでは、両タイプのトランザクションで同じである［トランザクション］タブの一般的なコンポーネントの表示について説明します。

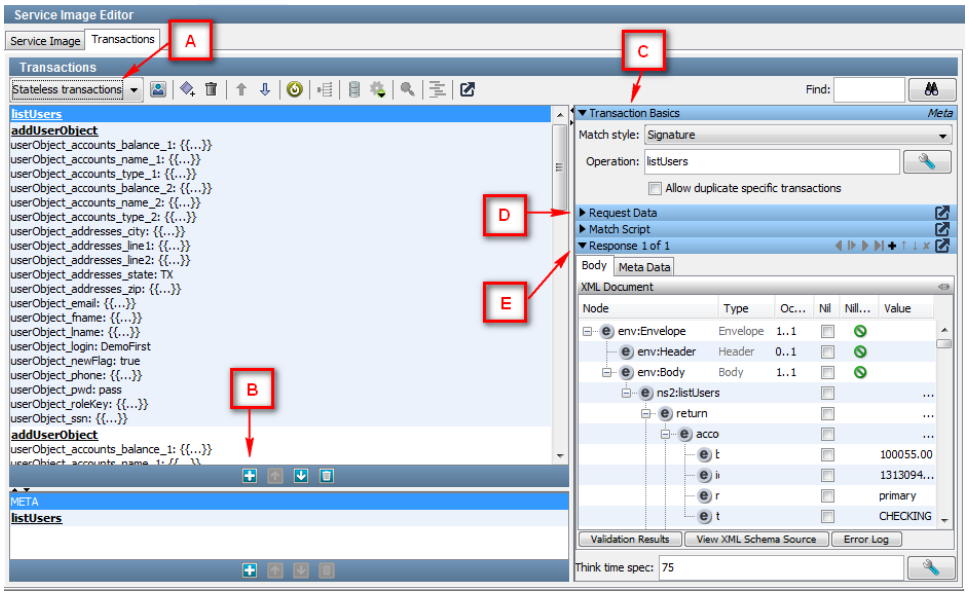
会話とステートレス トランザクションの表示を切り替えるには、ドロップダウン リストから[会話 <番号>] または[ステートレス トランザクション］のいずれかを選択します。

詳細情報

ステートレス トランザクション用の［トランザクション］タブ (P. 322)
会話用の［トランザクション］タブ (P. 335)
会話エディタ (P. 338)

ステートレス トランザクション用の[トランザクション]タブ

ステートレス トランザクションを表示する場合、以下の図に示すコンポーネントを参照できます。



- **A：** ステートレス トランザクションを表示および編集するには、［ステートレス トランザクション］リストを使用します。ステートレス トランザクションを追加、移動、または削除するには、ペインの下部のツールバーを使用します。
- **B：** 1 つの論理トランザクション（ステートレス トランザクション リスト内）には、1 つのメタ トランザクションおよび任意の数の特定のトランザクションが含まれます。［トランザクション］リストでは、論理トランザクションの下にこれらのトランザクションが表示されます。ステートレス トランザクションを追加、移動、または削除するには、ペインの下部のツールバーを使用します。
- **C：** 特定のトランザクションまたはメタ トランザクションのいずれか（［トランザクション］領域から選択）のトランザクション要求/応答データを表示および編集するには、［トランザクション基本情報］領域を使用します。ここで、特定のトランザクションの一致タイプを選択できます。
- **D：** ［要求データ］パネルには、ステートレス要求が表示されます。
- **E：** ［応答］パネルには、ステートレス 要求に対する応答が表示されます。

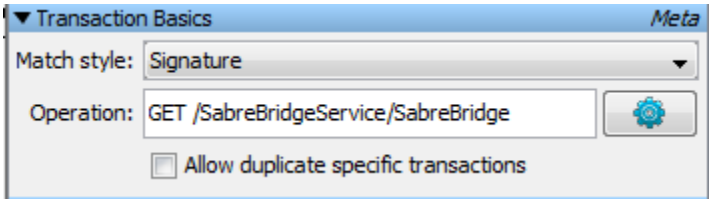
注: 複数のトランザクションを追加または変更する場合は、マジック ストリングおよびデータ変数の再生成を行うボタンをクリックします。 マジック ストリングと日付変数が作成されます。 既存のマジック ストリングと変数は変更されません。

このウィンドウの特定エレメントの詳細については、以下を参照してください。

- トランザクション基本情報エディタ (P. 324)
- 要求データ エディタ (P. 324)
- 一致スクリプト エディタ (P. 329)
- 一致スクリプト エディタ ツールバー (P. 332)
- 応答データ エディタ (P. 333)

トランザクション基本情報エディタ

特定のトランザクションまたはメタ トランザクションのトランザクション データを表示および編集するには、トランザクション基本情報エディタを使用します。 ［トランザクション］リストから、特定のトランザクションまたはメタ トランザクションを選択します。



トランザクション基本情報エディタでは、以下の情報を指定できます。

一致スタイル

- 有効なオプションは［**シグネチャ**］または［**操作**］です。

操作

選択されている操作。

特定トランザクションの重複を許可

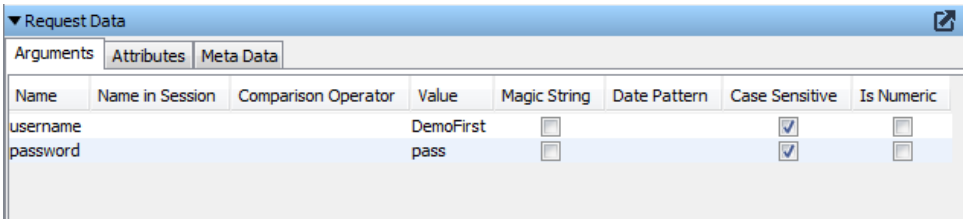
DevTest が別の応答を選択して同じコールに 2 回以上応答できるかどうかを指定します。

値

- **オン**： DevTest は、別の応答を使用して同じコールに複数回応答できます。 このチェック ボックスをオンにすると、ラウンド ロビン一致のみが発生します。
- **オフ**： DevTest は特定のコールに 1 回のみ応答できます。

要求データ エディタ

要求データ エディタでは、要求に関連付けられたデータを更新することができます。



引数

VSE は、受信トランザクションに一致する応答を検索するために操作名と引数を使用します。

引数の追加と削除

メタ トランザクションは、特定のトランザクション用のテンプレートです。 詳細については、「論理トランザクション (P. 67)」を参照してください。 そのため、メタ トランザクションに対して引数が追加または削除される可能性があり、これらの変更はその論理グループ内のすべての特定のトランザクションに適用されます。 引数は、特定のトランザクションに直接追加または削除されない場合があります。

引数の変更

name

引数の名前を定義します。これは要求から解析されます。 ほとんどの場合、このフィールドを変更する必要はありません。これは、メタ トランザクション レベルでのみ変更できます。また、これらの変更は特定のトランザクションに伝達されます。

セッション内の名前

マジック ストリングが識別されたときにアプリケーションによって自動的に生成される値を定義します。値は、{{ }} 表記を使用して現在（または後）の応答で参照できます。 このフィールドが空でない場合、受信した値は指定された名前を使用してセッションに格納されます。 この値は通常は変更する必要はありません。

比較演算子

一致ロジックで使用する演算子を定義します。 デフォルトでは、特定のトランザクションに対して、すべての引数が完全一致すると想定されています。これを変更してより柔軟な一致ロジックを作成するには、比較演算子を変更します。 比較演算子の定義については、「引数一致演算子 (P. 69)」を参照してください。

マジック ストリング

指定した値をマジック ストリングの候補として含めるかどうかを指定します。

値

- **オン**： このチェック ボックスをオンにして、［マジック ストリングの再生成］ボタンをクリックすると、指定した値はマジック ストリングの候補になります。 このチェック ボックスをオンにしても、マジック ストリングを識別するためのルールはオーバーライドされません。 何かを強制的にマジック ストリングにするために使用することはできません。これは、**lisa.properties** ファイル内の VSE magic string properties によって引き続き制御されます。
- **オフ**： このチェック ボックスをオフにして、［マジック ストリングの再生成］をクリックすると、指定した値はマジック ストリングの候補から除外されます。何かがマジック ストリングとして使用されたときに、マジック ストリングが置換されることを望まない場合は、このチェック ボックスをオフにして、［マジック ストリングの再生成］を選択します。

日付パターン

アプリケーションが受信および指定した値を日付として解釈するパターンを定義します。 この値は自動的に生成されます。通常、変更する必要はありません。

このパターンは、Java 日時パターンです。 これは厳密に解釈されるため、値はパターンに正確に一致する必要があります。両方の値またはいずれかの値を日付として解析できない場合、引数は一致しなかったと見なされます。 両方の値を日付として解析できる場合、それらは以下の演算子を使用して日付として比較できます。

- =
- !=
- <
- <=
- >
- >=

今までどおり、「すべて」、「正規表現」、および「プロパティ式」を使用できます。「正規表現」を使用する場合、受信した値は文字列として扱われます。

大文字と小文字を区別

一致で大文字と小文字を区別するかどうかを指定します。

値

- **オン**： すべての一致で大文字と小文字を区別します。
- **オフ**： 比較演算子は大文字と小文字を区別しません。

デフォルト： 選択済み。

数値

引数値を文字列または数値として処理するかどうかを指定します。

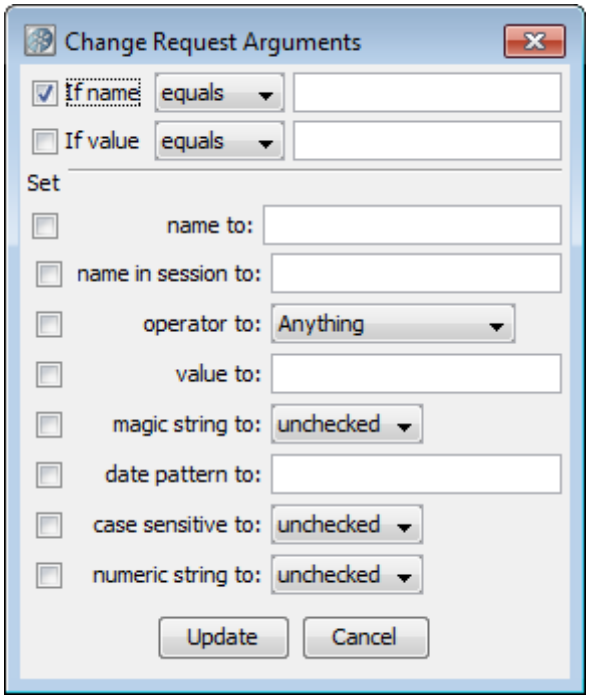
値

- **オン**： アプリケーションは引数値を数値として処理します。
- **オフ**： アプリケーションは引数値を文字列として処理します。つまり、アルファベット順で「1」は「9」の前に来るため、「10000」は「9」より小さいと見なされます。

デフォルト： オフ。

一括変更

要求引数の一括変更を実行するには、一括変更  をクリックします。［要求引数の変更］ダイアログ ボックスが表示されます。



一括変更を指定するには、［要求引数の変更］ダイアログ ボックスのフィールドに適切に入力し、［更新］をクリックします。

属性

キー/値ペアを追加、編集、移動、および削除するには、［属性］タブを使用します。

メタ データ

メタ データのキー/値ペアを追加、編集、移動、および削除するには、［メタ データ］タブを使用します。

一致スクリプト エディタ

情報用にサンプル一致スクリプトを挿入するには、［一致スクリプト］パネルを右クリックします。 また、［スクリプトを使用しない］チェックボックスをオン/オフにすることにより、一致スクリプトをオン/オフにすることができます。

スクリプト言語を指定するには、ペインの右下にある言語ドロップダウンを使用します。

言語

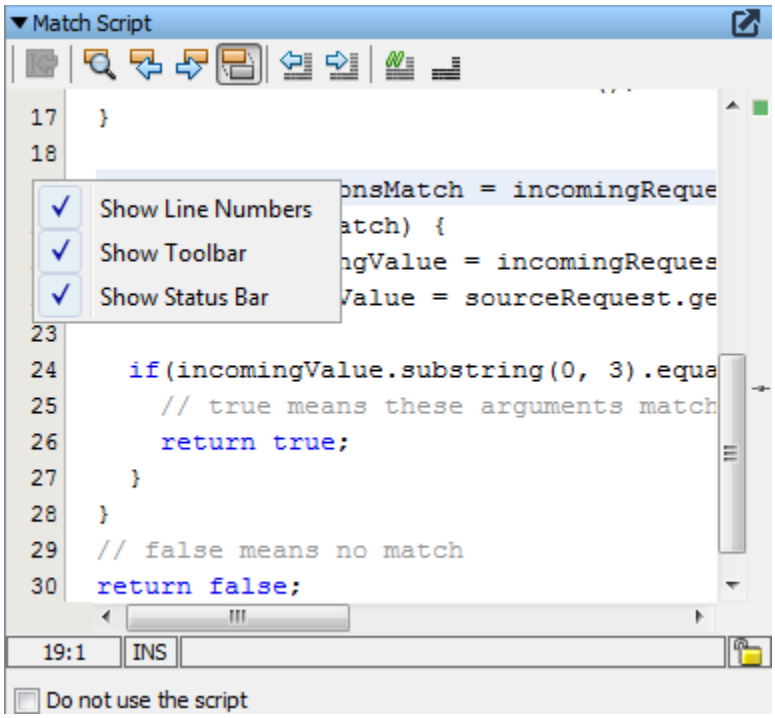
使用するスクリプト言語を指定します。

値

- Applescript （OS X 用）
- Beanshell
- Freemarker
- Groovy
- JavaScript
- 速度

デフォルト： Beanshell

行番号、エディタのツールバー、およびエディタのステータス バーを表示または非表示にするには、［一致スクリプト］パネルの左側を右クリックし、ショートカット メニューから適切なオプションを選択します。 以下の図は、表示されるすべてのオプションを示しています。



一致スクリプトは、特定のトランザクションが受信トランザクションに一致するかどうかを VSE が判断する方法を定義します。特定の条件に基づいて一致を受信するには、適切なアクションを実行する BeanShell スクリプトを記述します。

以下に例を示します。

```
/* joe という名前に常に一致 */

ParameterList args = incomingRequest.getArguments();

if ("joe".equals(args.get("name"))) return true else return

defaultMatcher.matches();
```

一致スクリプトが動作するために、一致許容差のレベルまたは一致演算子を指定する必要はありません。 一致は、一致スクリプト内の条件に基づいて検出されます。

デフォルト（一致スクリプトなし）では、インバウンド要求は、操作、引数、またはその両方を比較することにより、「一致したか?」の答えが true/false になることで、サービス イメージ要求と照合されます。 一致スクリプトは、意味があり、かつ引き続き「一致したか?」の答えが true/false になる任意のロジックでこのロジックを単に置き換えます。

スクリプトでは、デフォルトの一致ロジックを使用できます。 スクリプト内で、「defaultMatcher.matches()」式を使用します。 この式は、VSE のデフォルトの一致ロジックを使用して、true または false を返します。

一致スクリプトは、スクリプト化されたアサーションに似ています。 基本的には標準の BeanShell スクリプトですが、以下に示す変数（および通常のプロパティと testExec 変数）がプリロードされています。

- com.itko.lisa.vse.stateful.model.Request sourceRequest（記録された要求）
- com.itko.lisa.vse.stateful.model.Request incomingRequest （受信ライブ要求）
- com.itko.lisa.vse.RequestMatcher defaultMatcher（この変数をデフォルトで使用可能）

スクリプトからブール値を返します。true は、一致が見つかったことを意味します。

スクリプトの評価がエラーの場合、VSE は意図的にエラーを無視し、デフォルトで標準の一致ロジックを使用します。 スクリプトが実行されていないと思われる場合は、VSE ログ ファイルを確認します。

一致スクリプトにログ記録および追跡を追加する良い方法は、VSE 一致ロガーにコールを埋め込むことです。 VSE 一致ロガーは、**vse_xxx.log** ファイル内にメッセージを生成します。ここで、*xxx* はサービス イメージ名です。以下に例を示します。

```
import com.itko.lisa.VSE;

VSE.info(testExec, "short msg", "a longer message");

VSE.debug(testExec, "", "I got here¥!¥!");

VSE.error(testExec, "Error¥!", "Some unexpected condition");

return defaultMatcher.matches();
```

INFO でメッセージをログに記録する場合、後で実稼働設定が logging.properties ファイルに適用されると、ログ レベルが WARN となり、メッセージが DevTest テスト イベント（「ログ メッセージ」イベント）として表示されます。

logging.properties のヒント

- デバッグを簡単にするために、VSE トランザクション一致/不一致イベントに対してログを個別に保持します。

- 実稼働システムの場合は、INFO を WARN に変更するか、以下の行をコメント アウトします。

  ```
  log4j.logger.VSE=INFO, VSEAPP
  ```

- INFO 値では、通常は一致の失敗がすべてレポートされます。

一致スクリプト エディタ ツールバー

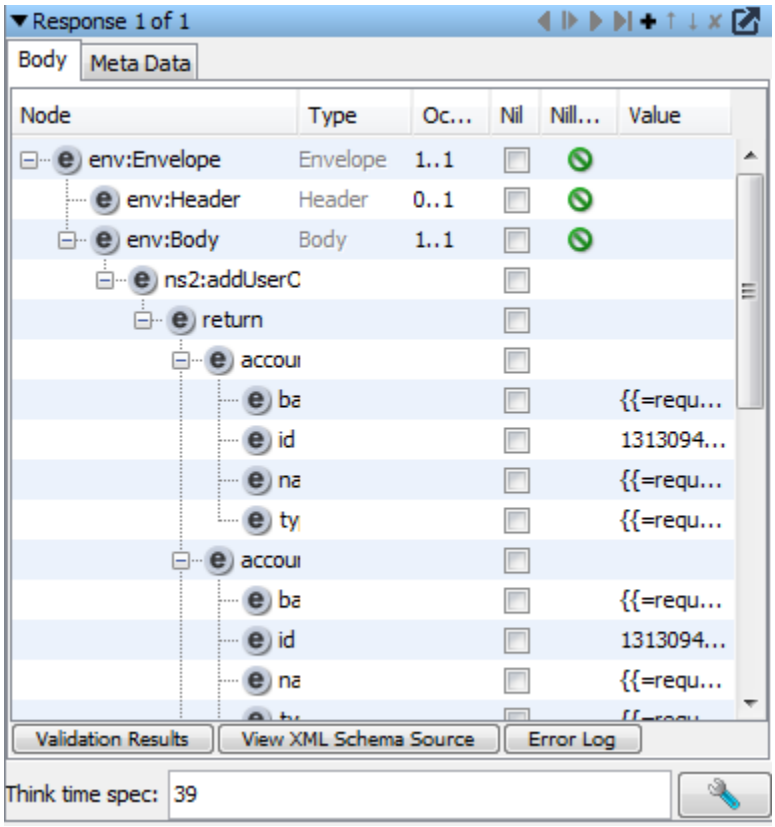
一致スクリプト エディタ ツールバーでは、以下の機能を実行できます。

	実行した最後の編集に戻ります
	次に出現する強調表示されたテキストを検索します
	前の出現を検索します
	次の出現を検索します
	強調表示の検索を切り替えます
	現在の行を左へ空白文字 4 つ分移動します
	現在の行を右へ空白文字 4 つ分移動します
	カーソル位置にコメント スラッシュ（//）を挿入します
	コメント スラッシュを削除します（//）

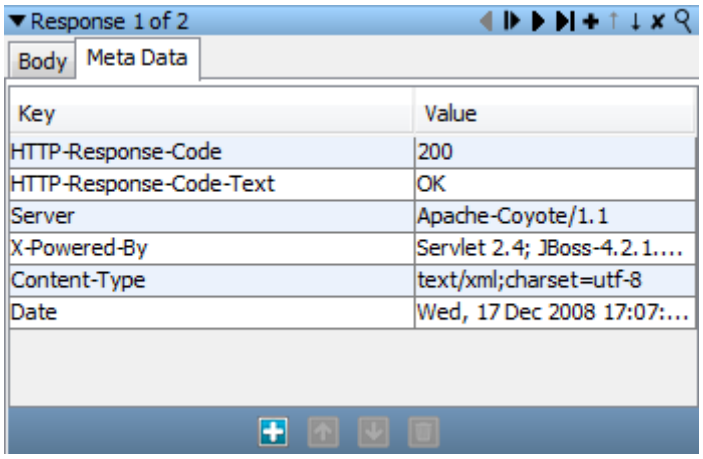
応答データ エディタ

応答情報を表示および編集するには、応答データ エディタを使用します。

- トランザクション対して予期される応答を編集するには、［応答ボディ］領域を使用します。
- 応答を追加、並び替え、削除、およびチェーンするには、「一致スクリプト エディタ ツールバー (P. 332)」の説明に従ってツールバーを使用します。
- このパネルを拡大するには、矢印アイコン を使用します。
- 応答エディタをカスタマイズするには、「応答エディタのカスタマイズ (P. 318)」の説明に従ってレンチ アイコン を使用します。
- 検証結果の確認、XML スキーマ ソースの表示、およびエラー ログの参照を行うには、パネルの下部のボタンを使用します。



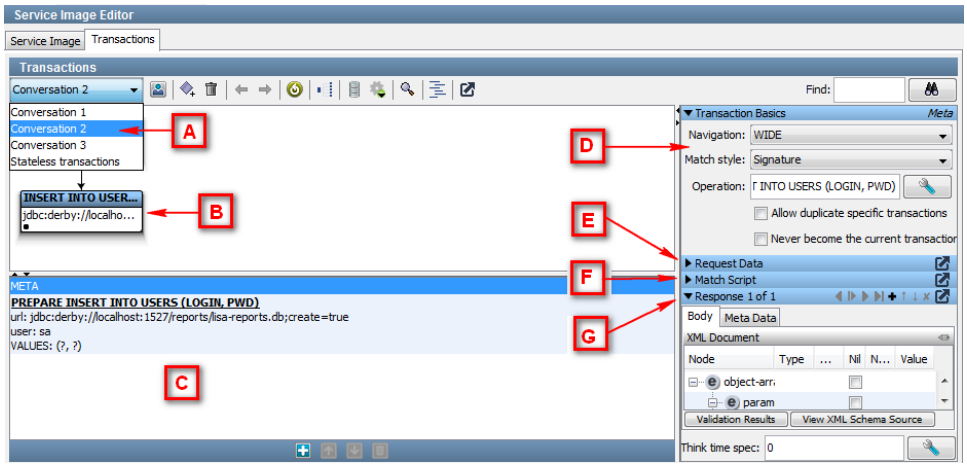
必要に応じて、［反応時間］を編集します。



キー値ペアを追加、編集、移動、および削除するには、［メタ データ］タブを使用します。

会話用の[トランザクション]タブ

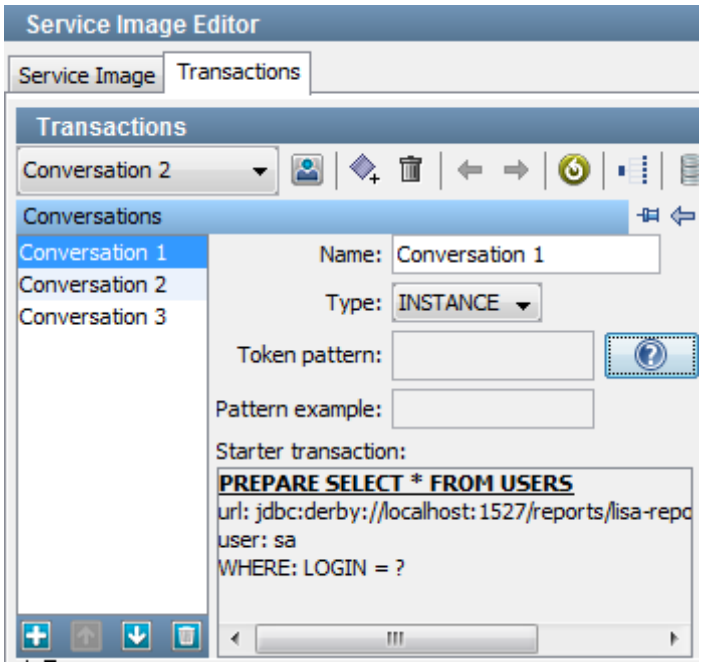
ステートフル トランザクション（会話を持つトランザクション）は、以下のコンポーネントで構成されます。



- **A：** ［会話］リストには、サービス イメージ内のすべての会話が表示されます。 会話を表示および編集するには、それを選択します。 会話は、複数の論理的なトランザクションから構成されます。
- **B：** 会話ツリー エディタ (P. 342)には、［会話］リストで選択されている論理的な会話が表示されます。 会話は、グラフ ノード ツリー ビューまたは標準ツリー ビューのいずれかで表示されます。
- **C：** 特定のトランザクション、または選択された論理的なトランザクション内のトランザクションのメタ データを表示および編集するには、［トランザクション］リストを使用します。 ステートレス トランザクションを追加、移動、または削除するには、ペインの下部のツールバーを使用します。
- **D：** 特定のトランザクションまたはメタ トランザクション（［トランザクション］領域から選択）のトランザクション要求/応答データを表示および編集するには、［トランザクション基本情報 (P. 324)］領域を使用します。 フィールドは、選択されたトランスポート プロトコルおよびデータ プロトコルによって異なります。［トランザクション基本情報］領域およびそのタブの詳細については、サービス イメージ エディタの「ステートレス トランザクション用の［トランザクション］タブ」 (P. 322)を参照してください。
- **E：** 再生中に会話型要求のデータを入力するには、トランザクションの［要求データ］ (P. 324)ペインを使用します。

- **F：** 指定された一致条件に基づいてアクションを返すスクリプトを入力および編集するには、一致スクリプト エディタ（P. 329）を使用します。
- **G：** 特定のトランザクションまたはメタ トランザクションの応答コンテンツ、反応時間、およびキー/値ペアを表示および編集するには、［トランザクション応答データ（P. 322）］ペインを使用します。

表示ペインの切り替え



［トランザクション］タブのツールバーの表示切り替えアイコン  をクリックすることにより、会話の詳細を参照できます。 このペインでは、以下の値を表示および編集できます。

- タイプ： タイプは［インスタンス］または［トークン］のいずれかです。
- トークン パターン： トークン ベースの会話に必要です。 クエスチョン マーク アイコンをクリックすると、文字列ジェネレータ パターンの例が表示されます。
- パターンの例： 指定されたトークン パターンの例。
- スタータ トランザクション： 会話のスタータ トランザクション。

注: 複数のトランザクションを追加または変更する場合は、［基本情報］タブに戻ってマジック ストリングおよびデータ変数の再生成を行うボタンをクリックします。マジック ストリングと日付変数が作成されます。既存のマジック ストリングと変数は変更されません。

会話エディタ

記録されたトランザクションを表示および編集するか、またはトランザクションを手動で作成するには、会話エディタを使用します。以下の 2 つの表示モードでナビゲーション ツリーを表示できます。

- グラフ ビュー (P. 340)
- ツリー ビュー (P. 342)

ビューを切り替える場合、選択したノードが選択されたままになります。グラフ ビューとツリービューの両方で、エディタ ツールバー、右クリック メニュー、またはビュー自体から、以下のアクションを実行できます。

会話エディタ ツールバー

会話ツリー エディタ ツールバーは、表示によって多少異なります。

グラフ ビュー ツール

ツリー ビュー ツール

注: ツールが淡色表示されている場合、選択されたトランザクションでは使用できません。

会話エディタ ツールバーには、以下のコンポーネントが含まれます。

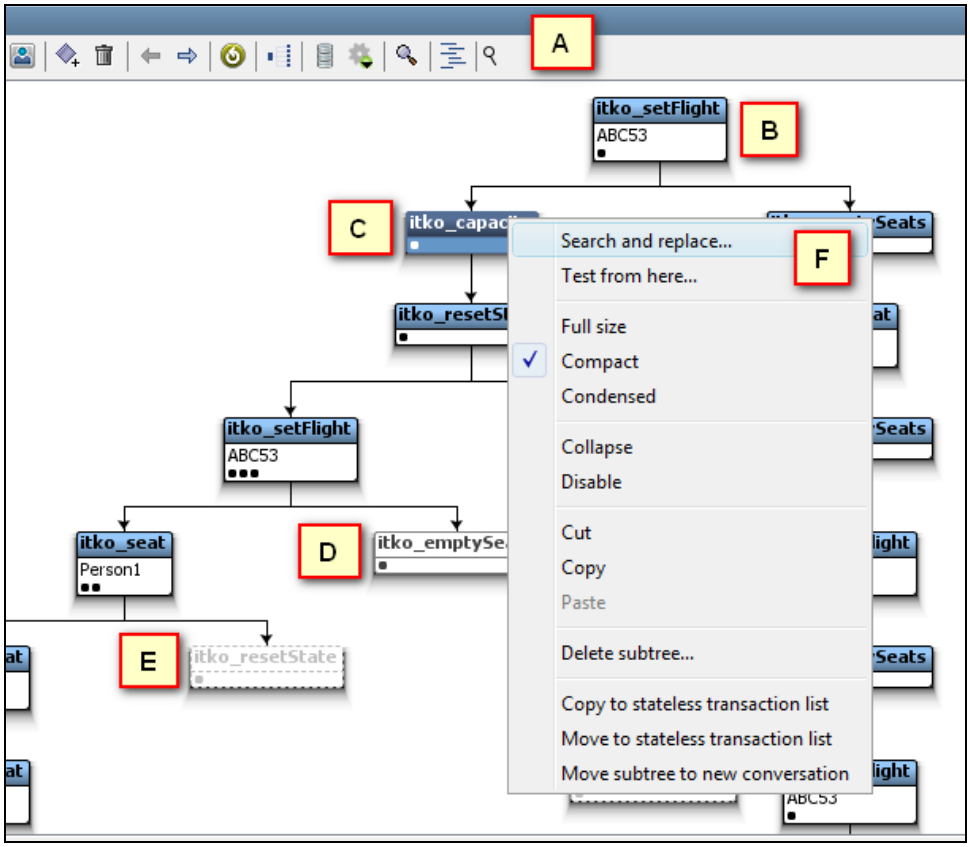
ツール	アイコン	説明
表示切り替え		会話のリストを管理するパネルの表示を切り替えます。 名前、タイプ、トークン パターン、パターンの例、またはスタータ トランザクションを指定できます。
新規トランザクションの作成		新しいトランザクションを追加します。
選択したトランザクションの削除		選択したトランザクションを削除します。

上矢印		ツリー ビュー：選択したノードを、兄弟リスト内で時間的に前の方向に移動します。
下矢印		ツリー ビュー：選択したノードを、兄弟リスト内で時間的に後の方向に移動します。
右矢印		グラフ ビュー：選択したノードを、兄弟リスト内で時間的に前の方向に移動します。
左矢印		グラフ ビュー：選択したノードを、兄弟リスト内で時間的に前の方向に移動します。
再生成		すべてのトランザクションに対して、マジック ストリングと日付変数を再生成します。
ナビゲーション表示		ステートフル トランザクション（会話）に関して、メニューナビゲーション強調表示を選択するドロップダウン メニューを表示します。 以下のオプションを選択できます。 ■ ナビゲーション強調表示なし ■ トランザクションの許容差を強調表示 ■ CLOSE を強調表示 ■ WIDE を強調表示 ■ LOOSE を強調表示
切り替え		デバッグ用のトランザクション ID の表示を切り替えます。
一致		クリップボードから一致の説明を取得し、サービス イメージ内の関連情報を強調表示します。
ズーム		ズームのドロップダウン メニューを表示します。
表示		会話表示からツリー表示に切り替えます。
表示		会話表示からグラフ表示に切り替えます。

パネルのズーム		パネルをその最大サイズに拡大します。

会話エディタのグラフビュー

会話エディタでは、ノードはステータスに従って表示されます。



会話グラフ エディタ コンポーネントおよびノード テーマについて、以下の図の引き出し線文字ごとに説明します。

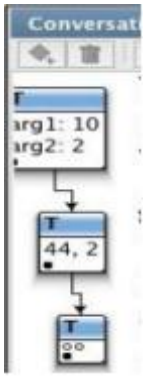
- **A:** ツールバー。詳細については、「会話エディタ ツールバー (P. 338)」を参照してください。
- **B :** 標準ノード。
- **C :** 選択されたノード。
- **D :** 折りたたまれたノード。 子ノードは表示されていません。 子ノードを表示するには、ノードを展開します。
- **E :** 無効なノード。 このノードおよび子ノード（表示されていない）は、実行時に無視されます。
- **F :** 右クリック メニュー。

ノード表示ステータス

ノードを 3 つの形式で表示できます。

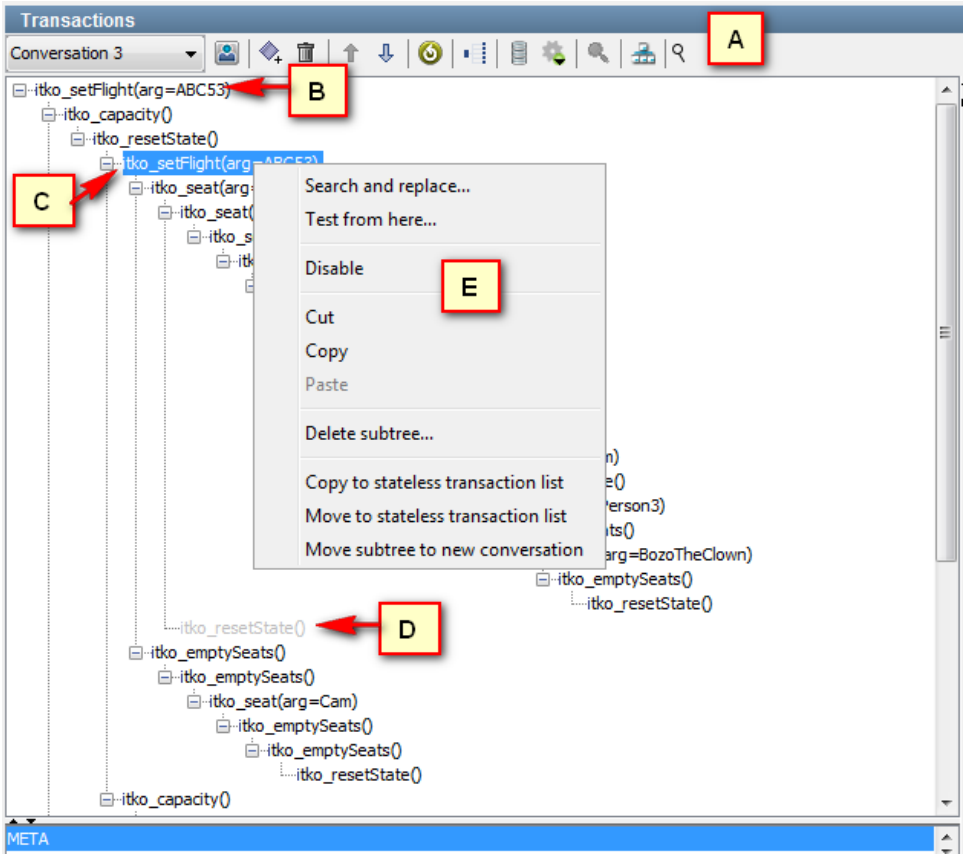
- フル サイズ
- コンパクト（デフォルト）|
- 要約

3 つすべての形式で、各ノードの下部ある黒いドットの行は、そのノードに属する特定のトランザクションの数を示しています。 要約形式では、中の白いドットが要求に対する引数の数を示しています。 以下の図では、最初のノードはフル サイズ、2 番目はコンパクト、3 番目は要約で表示されています。



会話エディタ ツリー ビュー

ツリー ビューには、グラフ ビューと同じ情報がよりコンパクトに表示され、ノードがステータスに従って表示されます。ツリー ビューには、以下のコンポーネントがあります。



- **A:** ツールバー。詳細については、「会話エディタ ツールバー (P. 338)」を参照してください。
- **B：** 標準ノード。
- **C：** 選択されたノード。
- **D：** 無効なノード。 このノードおよび関連する子ノード（表示されていない）は、実行時に無視されます。
- **E：** ショートカット メニュー。

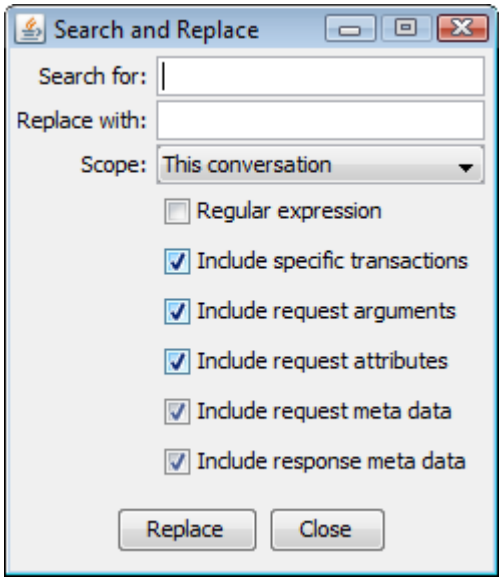
このウィンドウの特定エレメントの詳細については、以下を参照してください。

- ［検索と置換］アクション (P. 343)

- ［ここからテスト］アクション（P. 344）
- ナビゲーションの可能性の強調表示（P. 346）
- 会話の再構成（P. 351）

［検索と置換］アクション

右クリック メニューから［検索と置換］アクションを選択して、会話またはトランザクションの値を変更できます。



［スコープ］では、検索と置換の範囲を以下のように指定できます。

- この会話（デフォルト）
- このトランザクション
- このトランザクションと子
- 全イメージ

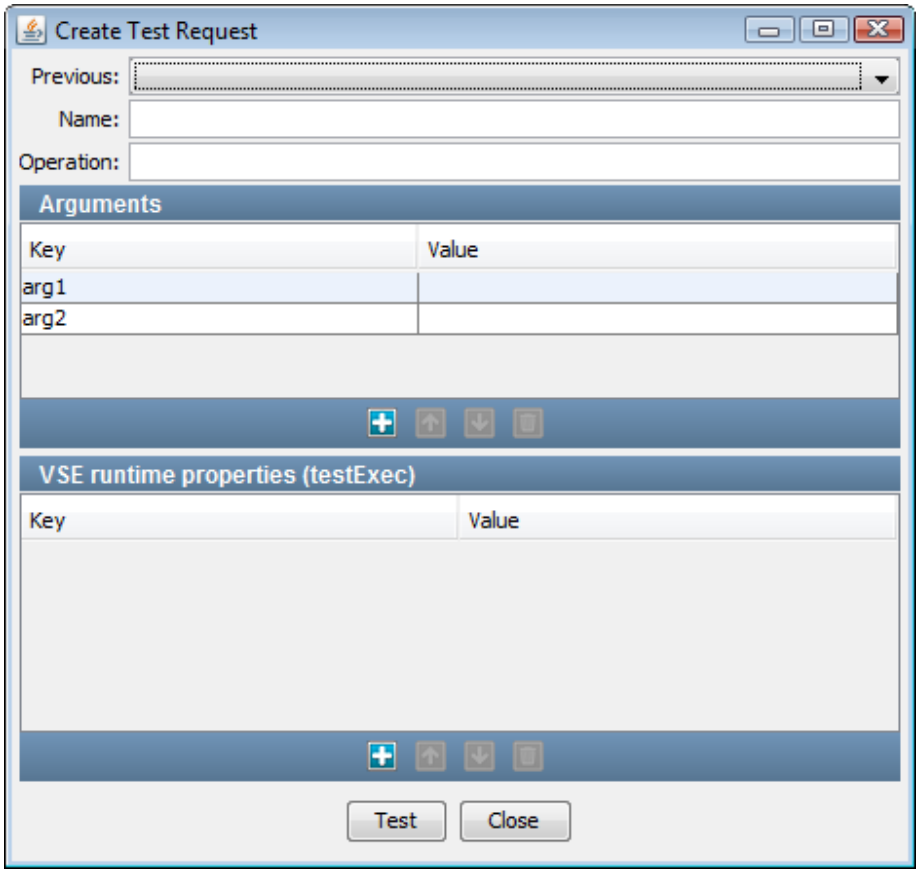
また、このチェック ボックスを使用して、包めるトランザクションを指定できます。

[ここからテスト]アクション

グラフ ビューとツリー ビューの両方で選択したトランザクションの予期される応答をテストするには、会話エディタを使用します。

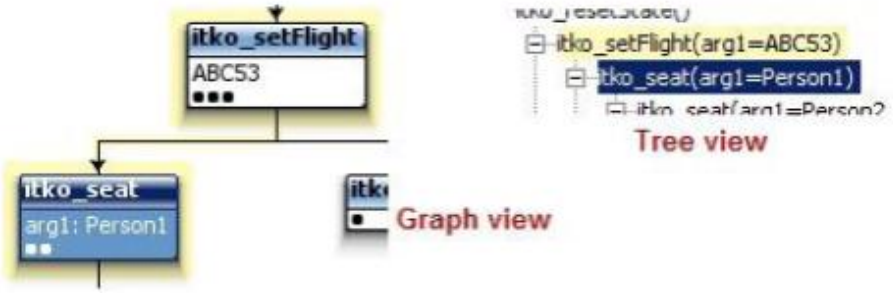
次の手順に従ってください:

1. 会話ツリー エディタで、トランザクションを右クリックします。
2. ショートカット メニューから［ここからテスト］を選択します。
3. ［テスト要求の作成］ウィンドウが表示されます。



4. 一意の操作名を入力し、必要に応じて、引数のキー/値ペアを追加します。
5. ［テスト］をクリックします。

   以下の図のように、結果は青で表示され、パスが黄色で表示されます。



注: テストが成功しない場合、アプリケーションには以下のエラーが表示されます。

「この会話内には、要求に一致し、選択したトランザクションに後続するトランザクションはありません。」

6. ［OK］をクリックして続行します。
7. ［閉じる］をクリックします。

ナビゲーションの強調表示

選択したナビゲーション許容差に基づく会話内のナビゲーションの可能性を表示するには、会話ツリーを使用します。

トランザクションを選択し、ナビゲーション表示ボタンをクリックします。

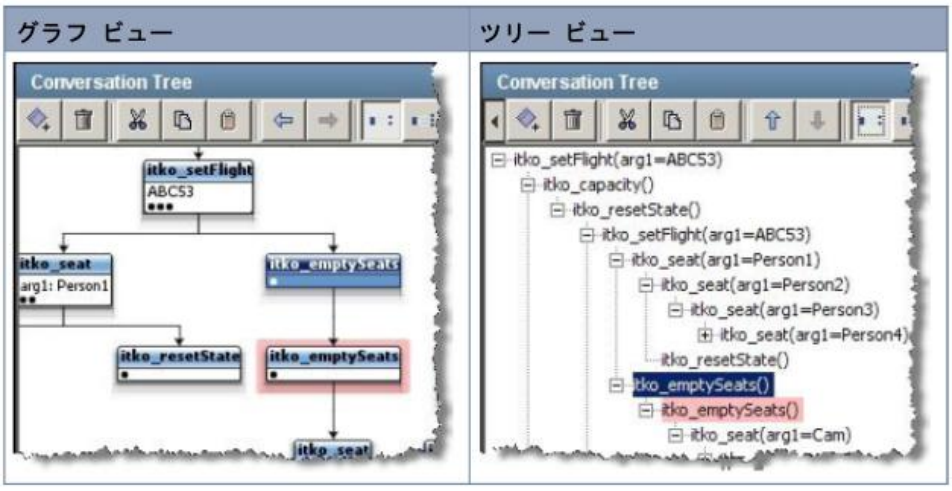
メニューではデフォルトで［ナビゲーション強調表示なし］が選択されています。これは、トランザクションが強調表示されないことを意味しています。

トランザクションを選択して［トランザクションの許容差を強調表示］をクリックすると、選択したトランザクションに対して定義されているナビゲーション許容差に従って、そのトランザクションおよびそれに関連するトランザクションが強調表示されます。 オプションには、CLOSE、WIDE、LOOSE があります。以下のセクションでは、各許容差レベルに対して予期される結果について説明します。

注: エディタの右上のドロップダウンを使用して現在のトランザクションのナビゲーション許容差を変更した場合は、このメニューよりも優先されます。 この場合、強調表示を「正しい」状態に戻すには、もう一度ナビゲーション強調表示オプションを選択します。

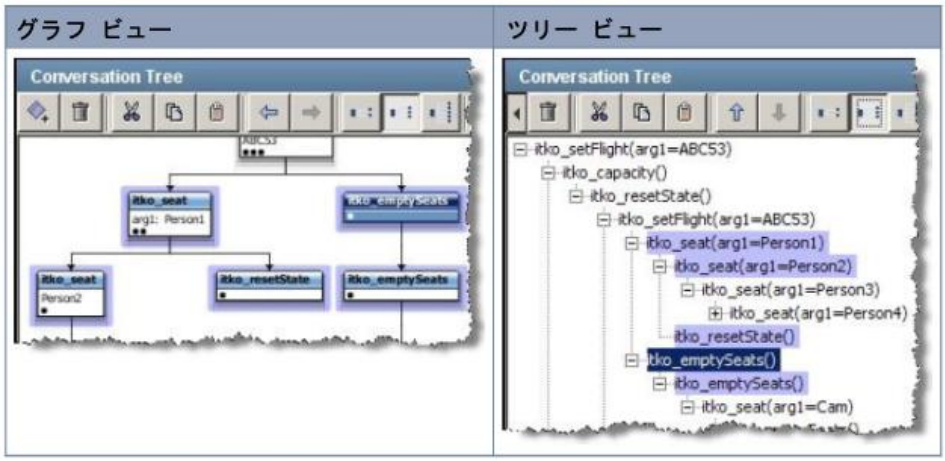
CLOSE ナビゲーションの表示

トランザクションを選択し、ナビゲーション表示ボタンをクリックします。選択したトランザクション サブツリーで検索されるトランザクションが、赤で強調表示されます。



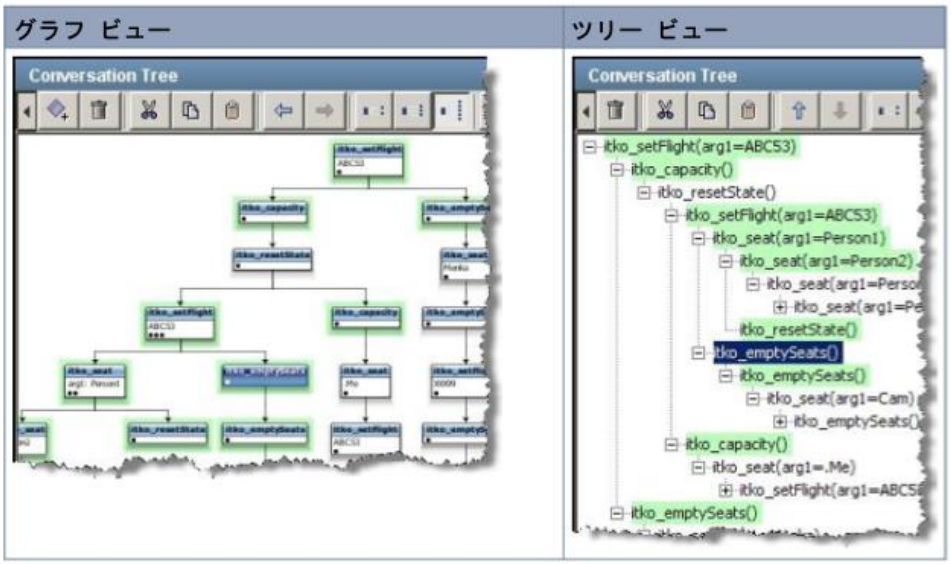
WIDE ナビゲーションの表示

トランザクションを選択し、ナビゲーション表示ボタンをクリックします。選択したトランザクション サブツリーおよび兄弟サブツリーで検索されるトランザクションが、青で強調表示されます。



LOOSE ナビゲーションの表示

トランザクションを選択し、ナビゲーション表示ボタンをクリックします。選択したトランザクション サブツリー、兄弟サブツリー、および親で検索されるトランザクションが、緑で強調表示されます。 また、会話全体の再起動が可能です。

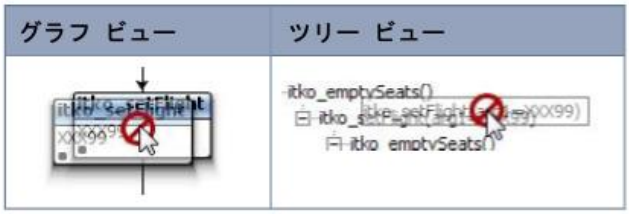

グラフ ビュー
ツリー ビュー
Conversation Tree
Conversation Tree
itko_setFlight(arg1=ABC53)
itko_capacity()
itko_resetState()
itko_setFlight(arg1=ABC53)
itko_seat(arg1=Person1)
itko_seat(arg1=Person2)
itko_resetState()
itko_emptySeats()
itko_emptySeats()
itko_seat(arg1=Cam)
itko_capacity()
itko_seat(arg1=.Me)
itko_emptySeats()

会話の再構成

場合によっては、トランザクションおよびそのサブツリー (存在する場合) をドラッグ アンド ドロップすることにより、会話を再構成します。 グラフ ビューとツリー ビューの両方で再構成を実行できます。

次の手順に従ってください:

1. トランザクションをクリックします。


グラフ ビュー
ツリー ビュー
itko_emptySeats()
itko_emptySeats()

トランザクションをドラッグすると、「不可」を示す記号が表示されます。

2. トランザクションを、新しく親トランザクションになるトランザクションにドラッグします。

   トランザクションをドロップできる場合、「不可」記号が消えます。


グラフ ビュー
ツリー ビュー
itko_setFlight(arg1=ABC53)
XXX99

3. トランザクションをドロップします。

   トランザクションおよびそのサブツリー内のすべてのトランザクションが、新しい親の下に移動されます。

JMS トランスポート プロトコルのサービス イメージ

このトピックでは、JMS トランスポート プロトコル (P. 165)を使用して作成されるサービス イメージに固有の特徴について説明します。

以下の項目が要求側に適用されます。

- デフォルトでは、トランザクションの操作名は、仮想サービス モデルの要求ステップで定義されている要求チャネル名に設定されます。
- プレフィックスでグループ化されないメタデータ プロパティには、キューとプロキシ要求キューの接続情報が含まれます。
- **liveRequest** で始まるメタデータ プロパティには、キューとライブ要求キューの接続プロパティが含まれます。
- **channel.name** メタデータ プロパティは、仮想サービス モデルの要求ステップおよびライブ呼び出しステップで定義されている要求チャネル名に一致する必要があります。 このプロパティは、要求側で唯一の必要なメタデータ プロパティです。要求チャネル名が同じであるかぎり、必要に応じて仮想サービス モデルの要求キューを変更できます。

以下の項目が応答側に適用されます。

- プレフィックスでグループ化されないメタデータ プロパティには、キューとプロキシ応答キューの接続情報が含まれます。
- **msg** で始まるメタデータ プロパティは、JMS 応答メッセージおよびそのプロパティを再構築するために使用されます。 必須プロパティは **msg.type** のみです。これは、送信するメッセージのタイプを示します。
- **channel.name** メタデータ プロパティは、仮想サービス モデルの応答ステップおよびライブ呼び出しステップで定義されている応答チャネル名に一致する必要があります。 この値は、クライアントに応答を送信するために使用される応答チャネル名を指定します。 複数の応答がある場合、応答は異なるチャネル名を指定できます。
- **liveResponse** で始まるメタデータ プロパティには、キューとライブ応答キューの接続プロパティが含まれます。

第 10 章: VSM の編集

仮想サービス イメージのレコーディングでは、選択したオプション（［フレキシブル］または［効率重視］）に応じて、6 ステップまたは 8 ステップで仮想サービス モデル（VSM）が作成されます。 生成されたステップを編集したり、ステップを追加したりして、VSM を編集したい場合があります。

VSM を編集する必要がある場合、またはレコーダなしで VSM を作成する必要がある場合を除いて、このセクションはスキップできます。

VSM は特殊なテスト ケースであるため、VSM の編集または作成はテスト ケースの編集または作成に似ています。 VSM には、さまざまなタイプのステップを追加できます。 このメニューに表示されていないステップを追加できますが、ステップのタイプによっては、VSM を展開できなくなる場合があります。

メニューから［ステップの追加］-［仮想サービス環境］を選択することにより、VSM エディタからステップ タイプのメニューにアクセスできます。

このセクションには、以下のトピックが含まれています。

仮想サービス ルータ ステップ

このステップは、仮想サービス リスン ステップからの要求を応答セレクタ ステップおよびプロトコル固有のライブ呼び出しステップ、またはその両方にルーティングします。 この決定は、実行中のモデルの現在の実行モードに基づいて行われます。

次の手順に従ってください:

1. 説明に従って、以下のフィールドに入力します。

   ライブ呼び出しステップ

   リストからライブ呼び出しのステップを選択します。

   環境エラーの場合

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップを指定します。

   デフォルト： テストを中止する。

   動的モードの場合は、次を使用して実際のモードを決定

   ［サブプロセス］または［スクリプト］を選択します。

2. ステップのパラメータをテストするには、［テスト］をクリックします。

仮想サービストラッカ ステップ

このステップは、実行中の仮想サービス内の応答を追跡し、必要に応じてライブ システムを検証するために使用します。 この検証により、サービス モデルのデバッグおよびサービス モデルの修正が簡単になります。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   イメージ応答

   イメージ応答ファイルを選択します。

   ライブ応答

   ライブ応答ファイルを選択します。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

仮想会話型/ステートレス応答セレクタ ステップ

仮想会話型/ステートレス応答セレクタ ステップは、すべての VSM の主要ステップと考えることができます。 このステップでは、指定されたサービス イメージが確認され、特定の要求に対する適切な仮想応答が選択されます。 1 つの要求に対して複数の応答が存在する場合があるため、応答は常にリストとして表示されます。 このステップは、通常、何らかの形式のサービス トラフィックを記録および仮想化するときに作成します。

説明に従って、以下のフィールドに入力します。

サービス イメージの場所

このステップで関連付けに使用できるサービス イメージをドロップダウン リストから選択します。ここでサービス イメージを選択した場合、［開く］ボタンをクリックして表示または編集用の新しいタブを開きます。

要求プロパティ名

インバウンド要求を検索するプロパティを定義するために、プロパティ名を設定します。 プロパティ名は、通常、前のステップの応答です。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

仮想 HTTP/S リスナ ステップ

仮想 HTTP/S リスナ ステップは、HTTP サーバをシミュレートするために使用します（SSL サポートを含む）。 このステップは、受信 HTTP 要求をリスンし、それらを標準の仮想要求形式に変換します。

仮想 HTTP/S リスナ ステップのデフォルトの名前は、「**仮想 HTTPS リスナ <***ポート番号***>**」です。 ステップ名は、いつでも変更できます。

説明に従って、以下のフィールドに入力します。

リスン ポート

DevTest が HTTP/S トラフィックをリスンするポートを入力します。

バインド アドレス

接続を受信できるローカル IP アドレスを入力します。バインド アドレスを指定しない場合、リスン ステップは、接続を受信する NIC（または IP アドレス）に関係なく、指定されたポートで接続を受け入れます。

バインドのみ

ネットワーク リソースを取得して次のステップに移動するには、このチェック ボックスをオンにします。[バインドのみ] オプションを使用しない 2 番目のリスン ステップが必要です。 このオプションを使用すると、モデルがポートをリスンできます（要求は、リスン ステップで処理されるまでキューに格納されます）。 モデルは、待機/処理/応答ループに入る前にセットアップ タスクを実行します。 たとえば、モデルのステップ 1 でリスニング ポートを取得し（［バインドのみ］を使用）、ステップ 2 で要求を送信する外部ソフトウェアをトリガします。

クライアントに SSL を使用

セキュアな HTTP/S Web サイトをシミュレートする場合は、このチェック ボックスをオンにします。 次に、SSL キーストア情報を提供します。

SSL キーストア ファイル

SSL キーストア ファイルを参照するには、［選択...］をクリックします。 VS モデルを展開する VSE サーバに対して、同じキーストア ファイルが使用可能である必要があります。

キーストア パスワード

キーストア パスワードを入力し、［検証］をクリックします。

ベース パス

リスン ステップが処理する、HTTP が要求したリソース URI を識別します。 要求を受信すると、キュー名のリストがスキャンされ、その要求に対して URI を起動する名前（ベース パス）が検索されます。 一致するキュー名は、要求が格納されるキュー名です。（ベース パスによって）キューに関連付けられているリスン ステップが要求を処理します。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

仮想 HTTP/S ライブ呼び出しステップ

仮想 HTTP/S ライブ呼び出しステップは、実際のサーバに対する実際の HTTP コールを、仮想化された HTTP サービスのコンテキストで行うために使用します。 このステップは、通常、何らかの形式の HTTP トラフィックを記録および仮想化することによって作成します。 このステップは、現在の VSE 要求に基づいて実際の要求を実行します。

仮想 HTTP/S ライブ呼び出しステップのデフォルトの名前は、「**仮想 HTTPS ライブ呼び出し<***ポート番号***>**」です。 ステップ名は、いつでも変更できます。

説明に従って、以下のフィールドに入力します。

ターゲット サーバ

要求が行われるサーバの名前を入力します。

ターゲット ポート

要求が行われるポートの名前を入力します。

置換 URI

GET/POST 要求で URI 全体を置換するには、新しいターゲット パス フィールドを入力します。 URI を DevTest プロパティとして提供できます。 このフィールドは空白にできます。その場合、ライブ要求からの URI が使用されます。

クライアントから受信したホスト ヘッダ パラメータは変更しないでください

これは、クライアント アプリケーションから受信したホスト ヘッダをターゲット サーバに転送するようにライブ呼び出しに指示します。 選択されていない場合、ライブ呼び出しは、ホスト ヘッダ パラメータを再生成して、「host: <ターゲット ホスト>:<ターゲット ポート>」にします。

サーバに SSL を使用

オンにすると、HTTPS （セキュア レイヤ）要求をサーバに送信します。

SSL キーストア ファイル

SSL キーストア ファイルを参照するには、［選択...］をクリックします。 VS モデルを展開する VSE サーバに対して、同じキーストア ファイルが使用可能である必要があります。

クライアント認証が必要な場合、キーストア内の最初の証明書がターゲット サーバに提示されます。この証明書は、**local.properties** 内の **ssl.client.*** プロパティがライブ呼び出しに対して指定する証明書をオーバーライドします。

キーストア パスワード

キーストア パスワードを入力し、［検証］をクリックします。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

不正な応答コード

システムからの失敗応答を定義します。

形式： 3 文字コードのカンマ区切りのリスト。各文字は数値または文字 x （ワイルドカード）です。

例： 4xx、5xx

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

注: 仮想 HTTP/S ライブ呼び出しステップは、使用されているクライアント ソケットを制御するために、**lisa.http.timeout.socket** プロパティと **lisa.http.timeout.connection** プロパティをサポートしています。 **lisa.vse.http.live.invocation.max.idle.socket** プロパティは、長時間アイドル状態のクライアント ソケットを使用しなくなるまでに待機する時間を制御します。 これは、デフォルトでは 2 分です。

仮想 HTTP/S レスポンダ ステップ

このステップは、仮想 HTTP/S リスナ ステップと一緒に使用し、リスナによって生成された HTTP 要求に対する応答を送信します。 このステップは、仮想応答を、対応する要求への応答として使用します（HTTP/S プロトコルを使用）。

このステップは、HTTP トラフィックを記録および仮想化することによって作成できます。

説明に従って、以下のフィールドに入力します。

応答リスト プロパティ名

応答を送信するために検索するプロパティの名前。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

会話型モデル プロパティ

プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。 ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。 これにより、その値がダウンストリームの会話型要求で使用可能になります。

仮想 JDBC リスナ ステップ

仮想 JDBC リスナ ステップは、JDBC データベース トラフィックのシミュレーションを制御するために使用します。 このステップは、データベース クライアントに組み込まれているシミュレーション ドライバとの通信を管理します。

仮想 JDBC リスナ ステップのデフォルトの名前は、「**仮想 JDBC リスナ<** *ポート番号*>」です。 ステップ名は、いつでも変更できます。

説明に従って、以下のフィールドに入力します。

エンドポイント情報

必要に応じて、シミュレーション ホストとポートの範囲（デフォルトは 2999）を設定します。 JDBC VSE は、レコーディングおよび再生時に複数のエンドポイントをサポートします。 エンドポイント情報は、［ドライバ ホスト］、［ベース ポート］、および［最大ポート］のテーブルです。 ［ベース ポート］と［最大ポート］が異なる場合、各ポート間で包含的に一意のエンドポイントが作成されます。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

接続/切断

JDBC シミュレータに接続するには、［接続］をクリックします。接続している場合、接続を終了するには、［切断］をクリックします。 このボタンを使用して接続情報を検証できます。

インストールおよび初期化済み JDBC ドライバ

データベース クライアントにインストールされ、初期化される JDBC ドライバ。

現在の SQL アクティビティ

データベース クライアント内の現在の SQL アクティビティ。

仮想 JDBC レスポンダ ステップ

このステップは、会話型応答選択ステップによって選択された JDBC データ コールの結果を取得します。その後、このステップは、データベース クライアントに組み込まれているシミュレーション ドライバにその結果を送信します。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   応答リスト プロパティ名

   応答を送信するために検索するプロパティの名前。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

   会話型モデル プロパティ

   プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。これにより、その値がダウンストリームの会話型要求で使用可能になります。

ソケット サーバ エミュレータ ステップ

このステップは、テキスト ベースのサーバ ソケット（通常 HTTP）をシミュレートするために使用します。 ソケット サーバ エミュレータ ステップは、リスン、応答、およびバインディングをサポートしています。 ソケット サーバ エミュレータ ステップは下位レベルです。 このステップを使用する場合、応答ステップで送信されるテキストのブロックが完全に HTTP に準拠していることを確認する必要があります。

注: 応答モードでソケット サーバ エミュレータ ステップを使用する場合、送信されるテキストは結果として有効な HTTP 応答メッセージである**必要があります**。

説明に従って、以下のフィールドに入力します。

プロセス モード

リストから、プロセス モードを選択します。 有効なオプションは以下のとおりです。

- フル プロセス： デフォルト。
- 非同期セットアップ： ネットワーク リソースを取得し、次のステップに移動します。
- リスンのみ
- 応答のみ

リスン ポート

DevTest が HTTP/S トラフィックをリスンするポートを入力します。

バインド アドレス

接続を受信できるローカル IP アドレスを入力します。バインド アドレスを指定しない場合、リスン ステップは、接続を受信する NIC（または IP アドレス）に関係なく、指定されたポートで接続を受け入れます。

すぐに閉じる

このステップの設計時テストを使用する場合に選択します。 このオプションは、設定された作業を実行し、その後すぐにネットワーク リソースをクリーンアップするようステップに指示します。

SSL を使用

セキュアな HTTPS Web サイトをシミュレートする場合は、このチェック ボックスをオンにします。 次に、SSL キーストア情報を提供します。

SSL キーストア ファイル

SSL キーストア ファイルを参照するには、［選択...］をクリックします。 VS モデルを展開する VSE サーバに対して、同じキーストア ファイルが使用可能である必要があります。

キーストア パスワード

キーストア パスワードを入力し、［検証］をクリックします。

ベース パス

リスン ステップが処理する、HTTP が要求したリソース URI を識別します。 要求を受信すると、キュー名のリストがスキャンされ、その要求に対して URI を起動する名前（ベース パス）が検索されます。 一致するキュー名は、要求が格納されるキュー名です。（ベース パスによって）キューに関連付けられているリスン ステップが要求を処理します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

レコード終了文字

レコード ベースのサービスをシミュレートするソケット エミュレータの場合は、レコードの終了をマークする文字を入力します。このフィールドを空白のままにすると、行指向のレコードまたは HTTP プロトコルのいずれかがシミュレートされます。

HTTP 応答の形式を確認

デフォルトでは、このオプションはオンになっています。 応答を送信するプロセス モードで応答が有効な HTTP 応答の場合、このオプションは、応答テキスト内の HTTP ヘッダが正しく整形されていること、および必要に応じて、**Content-Length:** HTTP 応答ヘッダが存在し、正しいことを確認します。 このチェック ボックスは、行区切り記号が HTTP に準拠しており、Content-Length ヘッダが存在し、正しいことのみを確認します。ただし、完全に動作するには、メッセージは、すでに正しく整形された HTTP メッセージである必要があります。

リスナ ステータス

リスナが実行されているかどうかを示すフィールド。

テスト

クリックすると、リスナ セットアップをテストします。

リスナのクリア

クリックすると、ステップのテストを停止します。

[応答]タブ

［送信する応答］には、応答のテキストが含まれます。

ファイルから応答を読み取り

クリックすると、応答のファイル システムを参照します。

[要求]タブ

設計時にのみ使用される最終/最初の要求を表示します。このタブは、ステップが受信した最終要求を表示します。

ソケット サーバ エミュレータ ステップのデフォルトの名前は、「**ソケット サーバ エミュレータ<***ポート番号***>**」です。 ステップ名は、いつでも変更できます。

メッセージング仮想化マーカ ステップ

このステップは、メッセージ ベースのテスト ケースが仮想サービス環境で使用されるように指定するために使用します。 テスト ケースが JMX を介してリスンまたは応答を行う場合は、このステップを仮想サービス モデルに追加して、そのモデルを VSE に展開できることを検証する必要があります。

注: JMS トランスポート プロトコルではこのステップは不要です。

応答用の文字列比較ルックアップ ステップ

このステップは、仮想サービスへの受信要求を監視します。 その後、このステップは、いずれのサービス イメージも参照せずに、ステートレス形式で適切な応答を決定します。VSE のステートフルな部分はサポートされていません。 部分テキスト一致、正規表現などを使用して、受信要求に一致させることができます。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

一致対象テキスト

条件に一致させるテキストを入力します。 この値は通常、LASTRESPONSE などのプロパティ参照です。

一致対象範囲

範囲の開始および終了を入力します。

一致が見つからない場合

リストから、一致が見つからない場合に移動するステップを選択します。

環境エラーの場合

デフォルト： テストを中止する。

環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

応答をテスト ケース ファイルに圧縮形式で格納します

デフォルトで選択されます。 このオプションは、テスト ケース ファイルに応答を圧縮します。

ケース応答エントリ

エントリを追加、移動、削除します。

有効

エントリを追加する場合にデフォルトでオンになります。 エントリを無視するにはオフにします。

名前

ケース応答エントリの一意の名前を入力します。

遅延仕様

遅延仕様の範囲を入力します。 デフォルトは **1000-10000** です。これは、1,000 から 10,000 ミリ秒のランダムに選択された遅延時間が使用されることを示します。 構文は、反応時間仕様と同じ形式です。

条件

この領域は、［一致対象テキスト］フィールドと比較する文字列を提供します。 条件を編集するには、［ケース応答エントリ］領域で適切な行を選択し、［条件］リストから別の設定を選択します。

比較タイプ

以下のリストからオプションを選択します。

- 文字列内の検索（デフォルト）
- 正規表現
- 前方一致
- 後方一致
- 完全一致

応答

この領域には、エントリが［一致対象テキスト］フィールドに一致する場合のこのステップの応答が表示されます。 応答を編集するには、［ケース応答エントリ］領域で適切な行を選択し、［応答］リストから別の設定を選択します。

条件

エントリの条件文字列を更新できます。

応答

エントリのステップ応答を更新できます。

次のステップ用の文字列比較ルックアップ ステップ

このステップは、受信要求を確認し、適切な次のステップを決定するために使用します。 部分テキスト一致、正規表現などを使用して、受信要求に一致させることができます。

各一致条件では、一致が検出された場合の転送先ステップ名を指定します。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   一致対象テキスト

   条件に一致させるテキストを入力します。 この値は通常、LASTRESPONSE などのプロパティ参照です。

   一致対象範囲

   範囲の開始および終了を入力します。

   一致が見つからない場合

   リストから、一致が見つからない場合に移動するステップを選択します。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

   次のステップのエントリ

   エントリを追加、移動、削除します。

   有効

   エントリを追加する場合にデフォルトでオンになります。 エントリを無視するにはオフにします。

   名前

   次のステップのエントリの一意の名前を入力します。

   遅延仕様

   遅延仕様の範囲を入力します。 デフォルトは **1000-10000** です。これは、1,000 から 10,000 ミリ秒のランダムに選択された遅延時間が使用されることを示します。 構文は、反応時間仕様と同じ形式です。

条件

この領域は、［一致対象テキスト］フィールドと比較する文字列を提供します。 条件を編集するには、［ケース応答エントリ］領域で適切な行を選択し、［条件］リストから別の設定を選択します。

比較タイプ

以下のリストからオプションを選択します。

- 文字列内の検索（デフォルト）
- 正規表現
- 前方一致
- 後方一致
- 完全一致

次のステップ

リストから、一致が見つかった場合に移動するステップを選択します。

条件

エントリの条件文字列を更新できます。

仮想 Java リスナ ステップ

このステップは、EJB またはその他のリモート システムへのコールなど、仮想化された JVM コールを処理するために使用します。このステップは、DevTest Java エージェントがインターセプトするメソッド コールをリスンし、標準の VSE 要求に変換します。

利用可能なオンライン エージェント

接続可能なすべてのオンライン エージェントをリスト表示します。

接続されたエージェント

リストには、仮想サービス モデルに対して接続するオンライン エージェントまたはオフライン エージェントがすべて含まれます。オフライン エージェントは、灰色の斜体フォントで表示されます。

エージェントの接続

［利用可能なオンライン エージェント］リストからエージェントを選択できます。 1 つまたは複数のエージェントを選択し、右矢印ボタンをクリックします。 エージェントが接続したエージェントのリストに移動されます。

［利用可能なオンライン エージェント］リストまたは接続したエージェントのリストのいずれかからエージェントを選択すると、リストの下の情報バーに、エージェントのホストおよびメイン クラス情報が表示されます。

接続したエージェントのリストにエージェントを手動で追加するには、［接続されたエージェント］の上の［エージェントの追加］フィールドを使用します。

エージェント名は空にすることはできません。または、接続したエージェントのリストにすでに存在します。 入力したエージェント名がオンライン エージェント リストに存在する場合、そのエージェントは、接続したエージェントのリストから接続したエージェントのリストに移動されます。

［利用可能なオンライン エージェント］リストに既存のエージェントが存在しない場合、ダイアログ ボックスに入力し、［追加］  をクリックします。

このメカニズムは主に、以前は接続リストに存在しなかったオフラインエージェントを追加するために提供されています。

エージェントの切断

エージェントを切断するには、接続したエージェントのリストからエージェントを選択し、左矢印ボタンをクリックします。

クラスおよびプロトコルの選択

クラスを検索するには、［クラスの検索］矢印を選択してクラス名を入力します。 これらのクラスは、正規表現を使用して、完全修飾名（パッケージを含む）として入力します。 クラスを選択するには、リストでクラス名を選択し、右矢印を選択して右ペインにクラスを移動します。 いくつかのクラスが 2 回以上表示されます。仮想化するためには、クラスは 1 回のみ選択する必要があります。

クラスを手動で入力するには、［クラス名の手動入力］矢印を選択します。 クラスの名前を入力します。 クラスを右ペインに移動するには、選択して右矢印をクリックします。 DevTest エージェントが仮想化を提案したクラスのリストを取得するには、［エージェント提案］矢印を選択し、［取得］  をクリックします。

レコーディングにプロトコルを追加するには、［プロトコル］矢印を選択します。 使用可能なプロトコルのリストから記録するプロトコル選択し、右矢印をクリックして右ペインに移動させます。

プロトコルの設定情報を表示または変更するには、プロトコル名の右に 3 つのドット（...）がある任意の行をダブルクリックします。 ［プロトコル設定］ウィンドウが表示され、パラメータを更新できます。

仮想 Java ライブ呼び出しステップ

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   ステップ応答を XML としてフォーマット

   VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

   - 応答オブジェクト
   - 応答オブジェクトのリスト
   - いずれかを表す XML ドキュメント

   注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

   デフォルト： ステップ応答は XML として整形されます。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

仮想 Java レスポンダ ステップ

このステップは、仮想 Java リスナ ステップと一緒に使用し、仮想化された JVM コールに対する応答を提供します。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   応答リスト プロパティ名

   応答を送信するために検索するプロパティの名前。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

   会話型モデル プロパティ

   プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。 ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。 これにより、その値がダウンストリームの会話型要求で使用可能になります。

仮想 TCP/IP リスナ ステップ

このステップは、サーバ アプリケーションへの TCP/IP 接続をシミュレートするために使用します。 このステップは、受信 TCP/IP トラフィックをリスンし、標準の VSE 要求に変換します。

仮想 TCP/IP リスナ ステップのデフォルトの名前は、「**仮想 TCP/IP リスナ<*ポート番号*>**」です。 ステップ名は、いつでも変更できます。

説明に従って、以下のフィールドに入力します。

リスン ポート

DevTest が TCP/IP トラフィックをリスンするポートを入力します。

バインド アドレス

接続を受信できるローカル IP アドレスを入力します。バインド アドレスを指定しない場合、リスン ステップは、接続を受信する NIC（または IP アドレス）に関係なく、指定されたポートで接続を受け入れます。

サーバに SSL を使用

オンにすると、SSL（セキュア レイヤ）要求をサーバに送信します。このチェック ボックスはオンにできますが、［クライアントに SSL を使用］は選択できません。 その場合、レコーディングに対してプレーン TCP 接続が使用されますが、それらの要求は SSL を使用してサーバに送信されます。

クライアントに SSL を使用

このオプションは、［サーバに SSL を使用］が選択されている場合のみ有効です。 カスタム クライアント キーストアを使用して、クライアントからの SSL 要求を再生できるかどうかを確認します。［クライアントに SSL を使用］を指定すると、カスタム キーストアとパスフレーズを指定できます。 これらの値を入力した場合、ハードコードされたデフォルトの代わりに使用されます。

SSL キーストア ファイル

SSL キーストア ファイルを参照するには、［選択...］をクリックします。 VS モデルを展開する VSE サーバに対して、同じキーストア ファイルが使用可能である必要があります。

キーストア パスワード

キーストア パスワードを入力し、［検証］をクリックします。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

仮想 TCP/IP ライブ呼び出しステップ

このステップは、実際のサーバに対する実際の TCP/IP コールを、仮想化された TCP/IP サービスのコンテキストで行うために使用します。 このステップは、通常、何らかの形式の TCP/IP トラフィックを記録および仮想化することによって作成します。このステップは、現在の VSE 要求に基づいて実際の要求を実行します。

仮想 TCP/IP ライブ呼び出しステップのデフォルトの名前は、「**TCP プロトコル ライブ呼び出し<***ポート番号***>**」です。 ステップ名は、いつでも変更できます。

説明に従って、以下のフィールドに入力します。

ターゲット ポート

要求が行われるポートの名前を入力します。

ターゲット サーバ

要求が行われるサーバの名前を入力します。

サーバに SSL を使用

オンにすると、SSL (セキュア レイヤ) 要求をサーバに送信します。このチェック ボックスはオンにできますが、［クライアントに SSL を使用］は選択できません。 その場合、レコーディングに対してプレーン TCP 接続が使用されますが、それらの要求は SSL を使用してサーバに送信されます。

クライアントに SSL を使用

このオプションは、［サーバに SSL を使用］を選択する場合のみ有効です。 ［クライアントに SSL を使用］は、カスタム クライアント キーストアを使用してクライアントからの HTTPS 要求を再生できるかどうかを指定します。

値

- **オン：** カスタム キーストアおよびパスフレーズを指定できます。 これらの値を入力する場合、アプリケーションは、それらをハードコードされたデフォルトの代わりに使用します。
- **オフ：** アプリケーションは、カスタム クライアント キーストアを使用してクライアントからの SSL 要求を再生できません。

SSL キーストア ファイル

SSL キーストア ファイルを参照するには、［選択...］をクリックします。 VS モデルを展開する VSE サーバに対して、同じキーストア ファイルが使用可能である必要があります。

キーストア パスワード

キーストア パスワードを入力し、［検証］をクリックします。

応答をテキストとして処理

選択した場合、応答をテキストとして処理するよう指定します。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

仮想 TCP/IP レスポンダ ステップ

説明に従って、以下のフィールドに入力します。

応答リスト プロパティ名

応答を送信するために検索するプロパティの名前。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

会話型モデル プロパティ

プロパティを入力し、［追加］をクリックします。プロパティを削除するには、リストから選択し、［削除］をクリックします。ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。これにより、その値がダウンストリームの会話型要求で使用可能になります。

仮想 CICS リスナ ステップ

仮想 CICS リスナ ステップは、CICS LINK サーバをシミュレートするために使用します。

このパネルのフィールドの詳細については、「パネルの仮想化 (P. 211)」を参照してください。

仮想 CICS レスポンダ ステップ

仮想 CICS レスポンダ ステップは、仮想 CICS リスナ ステップと一緒に使用し、リスナによって作成された要求に対する応答を送信します。 このステップは、仮想応答を取得し、対応する要求への応答として使用します。

次の手順に従ってください:

1. 説明に従って、フィールドにデータを入力します。

   応答リスト プロパティ名

   応答を送信するために検索するプロパティの名前。

   環境エラーの場合

   デフォルト： テストを中止する。

   環境エラーのためにテストが失敗した場合に実行するアクションまたは移動するステップをリストから選択します。

2. ［会話型モデル プロパティ］リストに入力するには、［追加］ボタンと［削除］ボタンを使用します。

CICS トランザクション ゲートウェイ リスナ ステップ

CICS トランザクション ゲートウェイ リスナ ステップは、CTG サーバをシミュレートするために使用します（SSL サポートを含む）。 このステップは、受信 CTG 要求をリスンし、それらを標準の仮想要求形式に変換します。

説明に従って、以下のフィールドに入力します。

リスン ポート

DevTest が CTG トラフィックをリスンするポートを入力します。

バインド アドレス

接続を受信できるローカル IP アドレスを入力します。バインド アドレスを指定しない場合、リスン ステップは、接続を受信する NIC（または IP アドレス）に関係なく、指定されたポートで接続を受け入れます。

バインドのみ

ネットワーク リソースを取得して次のステップに移動するには、このチェック ボックスをオンにします。[バインドのみ] オプションを使用しない 2 番目のリスン ステップが必要です。 このオプションを使用すると、モデルがポートをリスンできます（アプリケーションは要求をリスン ステップで処理されるまでキューに格納します）。 モデルは、待機/処理/応答ループに入る前にセットアップ タスクを実行します。 たとえば、モデルのステップ 1 でリスニング ポートを取得し（［バインドのみ］を使用）、ステップ 2 で要求を送信する外部ソフトウェアをトリガします。

クライアントから SSL で受信

このチェック ボックスをオンにすると、レコーダは、クライアントが SSL を使用して接続すると想定します。 関連するキーストアとパスワードを提供した場合、セキュリティ情報（証明書など）の取得に使用されます。

SSL キーストア ファイル

キーストア ファイルの名前。

キーストア パスワード

キーストア ファイルのパスワード。

CTG プロトコル バージョン

使用する CTG プロトコルのバージョンを指定します。

ロケール

初期プロトコル ハンドシェイク中に CTG クライアントにレポートされる言語と国コードを表すロケールを定義します。

JVM テキスト

メインフレーム上の JVM を説明します。

サーバ クラス

アプリケーションがクライアントにレポートする文字列を定義します。 このフィールドは、通常、使用されません。

クライアント アプリケーション ID

アプリケーションがクライアントにレポートする文字列を定義します。 このフィールドは、通常、使用されません。

セキュリティあり

ユーザ認証が必要かどうかを CTG クライアントに対して指定します。

サーバ ping の有効化

CTG クライアントがサーバに接続すると、サーバはクライアントへの「ping」メッセージの定期送信を開始して、接続を検証します。これは、テスト環境で VSE が使用される場合、厳密には必要ありませんが、このオプションを選択すると、VSE CTG サーバがこれらの「ping」メッセージをエミュレートします。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

CICS トランザクション ゲートウェイ ライブ呼び出しステップ

このステップは、実際のサーバに対する実際の CTG コールを、仮想化された CTG サービスのコンテキストで行うために使用します。 このステップは、通常、何らかの形式の CTG トラフィックを記録および仮想化することによって作成します。現在の VSE 要求に基づいて実際の要求を実行します。

説明に従って、以下のフィールドに入力します。

ターゲット サーバ

サーバが実行されているターゲット ホストの名前または IP アドレスを入力します。

ターゲット ポート

CTG サーバがリスンするポートの番号を入力します。

ターゲット サーバに対して SSL を開始

選択すると、SSL（Secure Socket Layer）要求をサーバに送信します。

SSL キーストア ファイル

キーストア ファイルの名前。

キーストア パスワード

キーストア ファイルのパスワード。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

CICS トランザクション ゲートウェイ レスポンダ ステップ

このステップは、CICS トランザクション ゲートウェイ リスナ ステップと一緒に使用し、リスナによって生成された CTG 要求に対する応答を送信します。 このステップは、仮想応答を、対応する要求への応答として使用します（CTG プロトコルを使用）。

このステップは、CICS トランザクション ゲートウェイ トラフィックを記録および仮想化することによって作成できます。

説明に従って、以下のフィールドに入力します。

応答リスト プロパティ名

応答を送信するために検索するプロパティの名前。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

会話型モデル プロパティ

プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。これにより、その値がダウンストリームの会話型要求で使用可能になります。

仮想 DRDA リスナ ステップ

仮想 DRDA リスナ ステップは、TCP/IP を介した DRDA トラフィックを仮想化するために使用します。 このステップは、受信 DRDA データをインターセプトし、それを標準の仮想要求形式に変換します。

説明に従って、以下のフィールドに入力します。

リスン/記録ポート

ターゲット ホスト

ターゲット ポート

DB2 IP アドレス

LISA IP アドレス

ストアド プロシージャ パラメータ区切り文字

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

仮想 DRDA 応答ビルダ ステップ

このステップは、DRDA 要求を個々の DRDA コマンドに分解して、異なるグループ化で DRDA コマンドを受信する VSE の複雑さを抑えます。たとえば、DRDA は同じコマンド セットを要求あたり 1、2、3、または 4 セットで送信します。 個々のコマンドを一致させ、複合応答を構築することによって、再生のためにモデルの柔軟性を可能な限り高めます。

仮想 DRDA ライブ呼び出しステップ

仮想 DRDA ライブ呼び出しステップは、実際のサーバに対する実際の DRDA コールを、仮想化された DRDA サービスのコンテキストで行うために使用します。 このステップは、通常、DRDA トラフィックを記録および仮想化することによって作成します。このステップは、現在の VSE 要求に基づいて実際の要求を実行します。

説明に従って、以下のフィールドに入力します。

リスン/記録ポート

ターゲット ホスト

ターゲット ポート

要求が行われるポートの名前を入力します。

DB2 IP アドレス

LISA IP アドレス

ストアド プロシージャ パラメータ区切り文字

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

IMS Connect リスン ステップ

IMS Connect リスン ステップは、IMS Connect 仮想サービスによって、IMS Connect 要求に応答するために使用します。

説明に従って、以下のフィールドに入力します。

リスン/記録ポート

クライアントが DevTest と通信するポートを定義します。

ターゲット ホスト

リスン ステップでは無効です。

ターゲット ポート

リスン ステップでは無効です。

IMS 形式ファイル

リスン ステップでは無効です。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

IMS Connect ライブ呼び出しステップ

以下のパラメータを入力します。

リスン/記録ポート

ライブ呼び出しステップには適用されません。

ターゲット ホスト

サーバが実行されているターゲット ホストの名前または IP アドレスを指定します。

ターゲット ポート

IMS サーバがリスンするターゲット ポート番号を指定します。

IMS 形式ファイル

デフォルトで DevTest に含まれている IMS Connect サポートを使用するには、このフィールドを空白のままにします。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

仮想 IMS Connect レスポンダ ステップ

このステップは、IMS Connect リスナ ステップと一緒に使用し、IMS Connect 応答を送信します。

このステップは、IMS Connect トラフィックを記録および仮想化することによって作成できます。

説明に従って、以下のフィールドに入力します。

応答リスト プロパティ名

応答を送信するために検索するプロパティの名前。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

会話型モデル プロパティ

プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。これにより、その値がダウンストリームの会話型要求で使用可能になります。

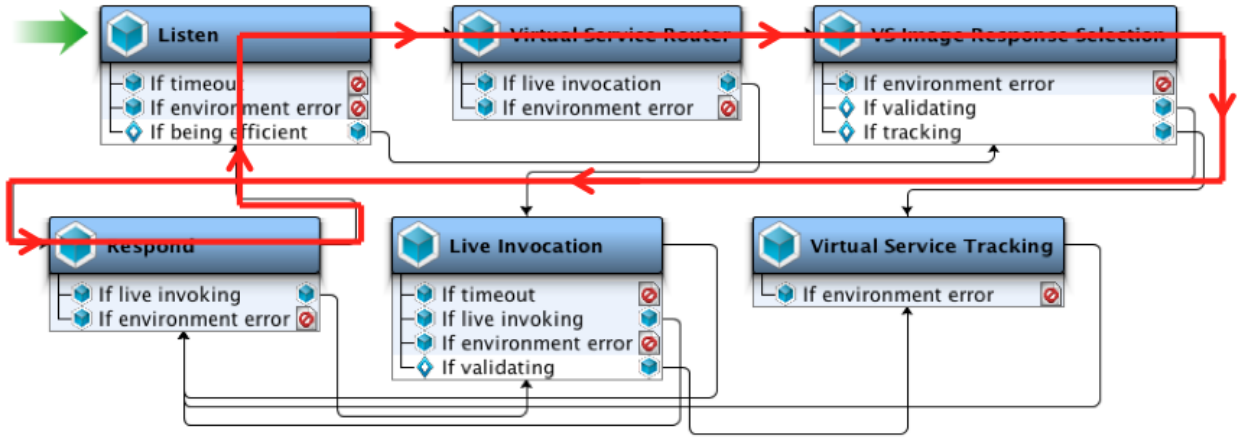
JMS VSE ステップ

以下のステップは、JMS トランスポート プロトコル (P. 165)を使用して作成される仮想サービス モデルに固有です。

- JMS VSE リスン
- JMS VSE 応答
- JMS VSE ライブ呼び出し

通常動作中、仮想サービス モデルの実行フローはその他の VSE サービスに似ています。

以下の図では、フローを示すためにオーバーレイを使用しています。



名前 **Listen** を持ったステップは JMS VSE リスン ステップです。 名前 **Respond** を持ったステップは JMS VSE 応答ステップです。 JMS VSE ライブ呼び出しステップは使用されません。

1. **リスン** ステップは、要求メッセージを受信し、それを VSE 要求に変換します。
2. **仮想サービス ルータ** ステップは、応答選択ステップにフローをルーティングします。
3. **VS イメージ応答選択**ステップは、サービス イメージから一致するトランザクションを選択し、VSE 応答を生成します。
4. **応答**ステップは、VSE 応答で 1 つ以上の応答メッセージを送信します。
5. 手順 1 に戻ります。

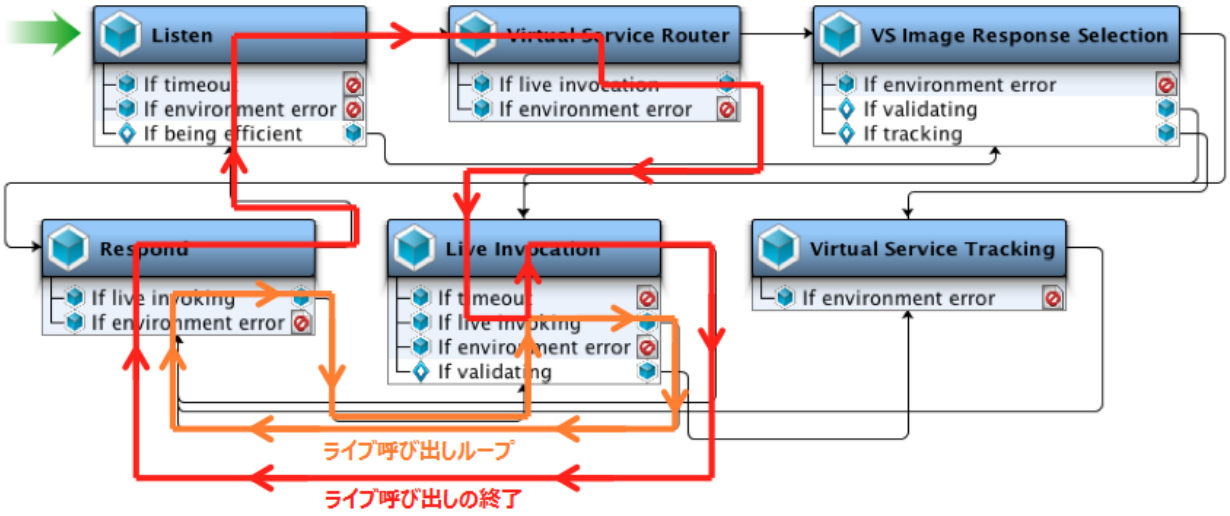
ライブ呼び出し中、仮想サービス モデルの実行フローはより複雑になります。

非同期メッセージングの以下の側面がライブ呼び出しを難しくします。

- 単一の要求に複数の応答が存在する場合があります。
- 応答は、返されるまでに任意の時間をかけることができます。
- 一般的に、すべての応答がいつ受信されたかを断定できません。

その結果、**ライブ呼び出し**ステップおよび**応答**ステップはループで実行されます。

以下の図では、フローを示すためにオーバーレイを使用しています。



名前 **Listen** を持ったステップは JMS VSE リスン ステップです。 名前 **Live Invocation** を持ったステップは JMS VSE ライブ呼び出しステップです。 名前 **Respond** を持ったステップは JMS VSE 応答ステップです。

1. **リスン** ステップは、要求メッセージを受信し、それを VSE 要求に変換します。
2. **仮想サービス ルータ ステップ**は、ライブ呼び出しステップにフローをルーティングします。
3. **ライブ呼び出し**ステップは、ライブ サービスに要求を転送します。
4. **ライブ呼び出し**ステップは、すべてのライブ応答キューでリスンを開始します。

5. **ライブ呼び出し**ステップは、任意のライブ応答キューからの単一の応答を受信し、それを VSE 応答に変換します。
6. **応答**ステップは、クライアントに 1 つの応答メッセージを返します。
7. 手順 5 に戻り、トランザクションが完了していると**ライブ呼び出し**ステップが判定するまで繰り返します。以下のいずれかの条件によって、**ライブ呼び出し**ステップはこの判定を行います。
   - タイムアウトが設定されていて、どのライブ応答キューからも別の応答を受信せずに経過した場合。デフォルト値は 30 秒です。タイムアウトは、VSE レコーダまたは**ライブ呼び出し**ステップで設定できる詳細パラメータです。
   - 応答の最大数が設定されていて、その数に達した場合。 デフォルト値は 1 です。これは、ループを 1 回のみ繰り返すことができることを示します。 応答の最大数は、VSE レコーダまたは**ライブ呼び出し**ステップで設定できる詳細パラメータです。
   - 仮想サービス モデルが、現在のモデル実行スレッドによってそのトランザクションを終了し、新しい要求を処理する必要がある容量に十分に近づいている場合。 タイムアウトおよび応答の最大数が設定されていない場合、このアクションは、モデルがライブ応答の待機を終了して、**リスン** ステップに戻るただ 1 つの方法です。
8. **ライブ呼び出し**ステップは、ループでのすべての応答メッセージを含んだ最終の VSE 応答を作成します。 ステップは、**応答**ステップに対して最後の 1 回ループします。
9. **応答**ステップは、最後のループではメッセージを送信しません。 代わりに、**応答**ステップは、一般的な VSE 状態クリーンアップ タスクを完了します。
10. 手順 1 に戻ります。

JMS VSE リスン ステップ

JMS VSE リスン ステップは、受信 JMS 要求をリスンし、それらを標準的な VSE 要求に変換します。

ステップ エディタには、基本パラメータおよび詳細パラメータがあります。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。

［要求］タブには、受信操作のリストが含まれます。

［チャネル名］フィールドは、要求チャネル名を定義します。 値は、サービス イメージで定義されている操作名に一致する必要があります。

個別の要求チャネルを無効にして再度有効にするには、［有効］チェックボックスを使用します。少なくとも 1 つの要求チャネルが有効である必要があります。

［送信先のマッピング］タブは、分離されたメッセージング シナリオに適用可能です。 このタイプのシナリオは、以下の状況で発生します。

- 1 セットのキューに DevTest がアクセス可能です。
- 別のセットのキューには、DevTest はアクセス可能ではありません。
- アプリケーションはアクセス可能でないキュー上で実行されます。
- メッセージは、DevTest が自動的にアクセス可能な語句に関して自動的に転送される語句です。

マッピングはそれぞれ 2 つの項目（応答チャネル名およびクライアントによって使用されるクライアント返信先）から構成されます。

JMS VSE 応答ステップ

JMS VSE 応答ステップは、VSE 応答で 1 つ以上の応答メッセージを送信します。

ステップ エディタには、基本パラメータおよび詳細パラメータがあります。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。

［チャネル名］フィールドは、応答チャネル名を定義します。 値は、サービス イメージで定義されている **operation.name** メタデータ プロパティに一致する必要があります。

個別の応答チャネルを無効にして再度有効にするには、［有効］チェックボックスを使用します。

JMS VSE ライブ呼び出しステップ

JMS VSE ライブ呼び出しステップは、ライブ サービスに要求を送信します。

ステップ エディタには、基本パラメータおよび詳細パラメータがあります。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。

［ライブ要求の送信］タブには、ライブ要求の送信に使用する操作のリストが含まれます。

［チャネル名］フィールドは、要求チャネル名を定義します。値は JMS VSE リスン ステップのチャネル名に一致する必要があります。

個別の要求チャネルを無効にして再度有効にするには、［有効］チェックボックスを使用します。少なくとも 1 つの要求チャネルが有効である必要があります。

［ライブ応答の受信］タブには、ライブ応答の受信に使用する操作のリストが含まれます。

［チャネル名］フィールドは、応答チャネル名を定義します。値は JMS VSE 応答ステップのチャネル名に一致する必要があります。

個別の応答チャネルを無効にして再度有効にするには、［有効］チェックボックスを使用します。少なくとも 1 つの応答チャネルが有効である必要があります。

［タイムアウト］パラメータおよび［最大応答数］パラメータは、ライブ呼び出しのループ (P. 396)を抜けるかどうかを判定するためにステップが使用できる条件です。

［送信先のマッピング］タブは、分離されたメッセージング シナリオに適用可能です。 このタイプのシナリオは、以下の状況で発生します。

- 1 セットのキューに DevTest がアクセス可能です。
- 別のセットのキューには、DevTest はアクセス可能ではありません。
- アプリケーションはアクセス可能でないキュー上で実行されます。
- メッセージは、DevTest が自動的にアクセス可能な語句に関して自動的に転送される語句です。

マッピングはそれぞれ 2 つの項目（応答チャネル名、および応答が応答チャネル上のライブ応答キューを通過するためにサービスに送信する必要があるサービス返信先）から構成されます。

JCo IDoc リスナ ステップ

このステップは、JCo IDoc 仮想サービスから JCo IDoc 要求に応答するために使用します。 このステップを作成するには、JCo IDoc トラフィックを記録および仮想化します。

以下のパラメータを入力します。

クライアント RFC 接続プロパティ

SAP ゲートウェイに対してプログラム ID で登録して IDoc を受信するために VSE が 使用する接続プロパティが含まれるクライアント RFC 接続プロパティ ファイルを定義します。 プロパティは、.jcoServer ファイルで指定されたものと同じである必要があります。

クライアント RFC 送信先名

RFC 送信先を識別する一意の名前を指定します。

クライアント システム接続プロパティ

IDoc をクライアント SAP システムに返すための接続プロパティが含まれるクライアント システム接続プロパティ ファイルを指定します。 これらのプロパティは、クライアント SAP システムに接続するために使用できる .jcoDestination ファイルで指定されたものと同じである必要があります。

クライアント システム名

クライアント SAP システムを識別する一意の名前を指定します。

要求識別子の XPath 式

識別子を生成するために要求 IDoc XML と一緒にプロトコルが使用する XPath 式を指定します。 要求識別子の XPath 式は、単一の XPath 式にすることができます。 この識別子は、要求 IDoc と応答 IDoc を関連付けるために使用されます。また、XPath 式は、XPath 式のカンマ区切りリストにすることもできます。その場合、複数の式からの結果の値は連結され（ダッシュで区切られ）、単一の識別子として使用されます。

応答識別子の XPath 式

識別子を生成するために応答 IDoc XML と一緒にプロトコルが使用する XPath 式を指定します。
応答識別子の XPath 式は、単一の XPath 式にすることができます。この識別子は、要求 IDoc と以前に受信した応答 IDoc を関連付けるために使用されます。 また、XPath 式は、XPath 式のカンマ区切りリストにすることもできます。その場合、複数の式からの結果の値は連結され（ダッシュで区切られ）、単一の識別子として使用されます。

ステップ応答を XML としてフォーマット

VSE フレームワークは、応答ステップで以下のいずれかを受理することを予期します。

- 応答オブジェクト
- 応答オブジェクトのリスト
- いずれかを表す XML ドキュメント

注: このチェック ボックスがオフの場合、このステップでは応答オブジェクトのリストが作成されます。 リストに応答が 1 つしか含まれていない場合でも、リストが作成されます。

デフォルト： ステップ応答は XML として整形されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

JCo IDoc ライブ呼び出しステップ

以下のパラメータを入力します。

クライアント RFC 送信先名

RFC 送信先を識別する一意の名前を指定します。

サーバ RFC 接続プロパティ

SAP ゲートウェイに対してプログラム ID で登録して IDoc を受信するために VSE が 使用する接続プロパティが含まれるプロパティ ファイルを指定します。 プロパティは、サーバ SAP システムから IDoc を受信する JCo サーバ プログラムを起動するために .jcoServer ファイルで指定されたものと同じである必要があります。

サーバ RFC 接続先名

サーバ RFC 接続先を識別する一意の名前を指定します。

サーバ システム接続プロパティ

サーバ システム接続プロパティ ファイルには、クライアント SAP システムに IDoc を送信するための接続プロパティが含まれます。これらのプロパティは、SAP サーバ システムに接続するために使用できる .jcoDestination ファイルで指定されたものと同じである必要があります。

サーバ システム名

サーバ SAP システムを識別する一意の名前。

JCo IDoc レスポンダ ステップ

このステップは、JCo IDoc リスナ ステップと一緒に使用し、JCo IDoc 応答を送信します。 このステップは、JCo IDoc トラフィックを記録および仮想化することによって作成できます。

このステップのパラメータはありません。

JCo RFC リスナ ステップ

このステップは、JCo RFC 仮想サービスから JCo RFC 要求に応答するために使用します。 このステップを作成するには、JCo RFC トラフィックを記録および仮想化します。

以下のパラメータを入力します。

クライアント システム名

クライアント SAP システムを識別する一意の名前を指定します。

クライアント システム接続プロパティ

クライアント システム接続プロパティ ファイルには、クライアント システム上の送信先に接続するための接続プロパティが含まれます。 これはプロジェクトの Data ディレクトリの .properties ファイルである必要があり、.jcoServer ファイルで通常見つかるプロパティが含まれます。 このファイルでは、**jco.server.repository_destination** を指定しないでください。 サポートされているプロパティの詳細については、インストール ディレクトリの **doc** フォルダにある **com.sap.conn.jco.ext.ServerDataProvider** の JavaDoc を参照してください。

ステップ応答を XML としてフォーマット

このチェック ボックスをオンにした場合、ステップ応答（受信要求）は XML としてシリアライズされ、テキストとして操作できます。 これは、応答ルックアップ ステップでデシリアライズされます。 このオプションを選択すると、仮想サービスの速度が大きく低下します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

JCo RFC ライブ呼び出しステップ

以下のパラメータを入力します。

送信先システム名

RFC が実行される SAP システムを識別する一意の名前を定義します。 これは多くの場合、リポジトリ名と同じです。

送信先システム接続プロパティ

リポジトリがあるシステムに接続するための接続プロパティが含まれる送信先システム接続プロパティ ファイルを指定します。 これは、プロジェクトの Data ディレクトリの .properties ファイルである必要があり、.jcoDestination ファイルで通常見つかるプロパティが含まれます。 サポートされているプロパティの詳細については、インストール ディレクトリの **doc** フォルダにある **com.sap.conn.jco.ext.DestinationDataProvider** の JavaDoc を参照してください。これは、リポジトリ接続プロパティと同じファイルにすることができます。

ステップ応答を XML としてフォーマット

このチェック ボックスをオンにした場合、ステップ応答（ライブ システムからの応答）は XML としてシリアライズされ、テキストとして操作できます。 これは、応答ステップでデシリアライズされます。 このオプションを選択すると、仮想サービスの速度が大きく低下します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

VSE Lookup Step

ライブ呼び出しステップがフェールオーバ実行モードをサポートするには、VSE 応答の検索に使用されるステップを把握し、必要に応じて VS モデルを正しいステップにリダイレクトできるようにする必要があります。 このフィールドには、VS モデルのステップのリストが含まれます。 標準的な VSE 応答ルックアップ ステップを選択します。これにより、必要に応じて、ライブ呼び出しステップから VSE 応答ルックアップ ステップに移行できるようになります。

JCo RFC レスポンダ ステップ

このステップは、JCo RFC リスナ ステップと一緒に使用し、JCo RFC 応答を送信します。 このステップは、JCo RFC トラフィックを記録および仮想化することによって作成できます。

以下のパラメータを入力します。

応答リスト プロパティ名

応答を送信するために検索するプロパティの名前。

会話型モデル プロパティ

プロパティを入力し、［追加］をクリックします。 プロパティを削除するには、リストから選択し、［削除］をクリックします。ここでリスト表示されるプロパティは、現在の会話セッションと関連付けられます。これにより、その値がダウンストリームの会話型要求で使用可能になります。

第 11 章: データのディセンシタイズ

VSE では、ディセンシタイズは、レコーディング中に機密データを認識し、そのデータに対して、ランダムな値でありながら正しくフォーマットされた値を代入することを意味しています。 テスト データとして実際の顧客データを使用したくない場合に、データ ディセンシタイズを使用します。

VSE のデータ ディセンシタイズは、以下の 3 つの方法でアクティブにできます。

- 動的なディセンシタイズ (P. 410)
- 静的なディセンシタイズ (P. 411)
- データ ディセンシタイザ データ プロトコル (P. 270)

動的なディセンシタイズ

動的なディセンシタイズは、トランスポート層で発生します。 ディセンシタイズは、仮想サービス レコーダの［基本］タブで［ディセンシタイズ（トランスポート層）］チェック ボックスを有効にすることにより呼び出されます。

このディセンシタイズ プログラムは、レコーディング段階で機密情報がディスクに書き込まれないようにする揮発性フィルタを使用します。 DevTest ホーム ディレクトリの **desensitize.xml** ファイルにより、クレジット カード番号などの既知のパターンを認識するようにデータ ディセンシタイザが設定されます。 このファイルでは、ライブ データが、現実的ではあるが使用不可能な内容で置換されます。 このファイルでは、正規表現のパターン一致を使用して機密データを認識および検索します。 このファイルは、レコーダが開始されるたびに解析されます。

置換データ オプションとして、組み込みの TestData 文字列生成パターンを使用できます。 TestData は、40,000 行のテスト データを提供します。 これには、名前、住所、電話番号、クレジット カード番号などのいくつかの一般的なデータ タイプの置換データが含まれます。

これらのプリセット パターンをカスタマイズして、独自のものを作成できます。 RegexBuddy などの正規表現ツールキットを推奨します。 RegexBuddy では、記録されたペイロードに貼り付け、正規表現を調整して、正規表現の一致を対話形式で強調表示できます。

一致は、ファイル内に存在する順番で処理されます。したがって、より詳細な一致を最初に配置します。

テキストをエスケープする手間を省くには（特に正規表現の場合）、CDATA エレメントで <regex> と <replacement> 子テキストを囲む**必要があります**。

静的なディセンシタイズ

静的なデータ ディセンシタイズには、既存のサービス イメージ内のデータを手動で検索および置換することが含まれます。［検索と置換］メニューにアクセスするには、サービス イメージ内のノードを右クリックして［検索と置換］を選択します。

別の文字列と置換する特定の文字列を指定し、変更のスコープを指定します。［置換］をクリックすると、選択した領域に対して検索/置換機能が実行されます。

データ ディセンシタイザ データ プロトコル ハンドラ

データ ディセンシタイザ データ プロトコル ハンドラでは、別のデータ プロトコルで要求ボディまたは応答ボディを「明確」にする必要がある場合、ディセンシタイズ ルールを適用できます。たとえば、HTTP メッセージ ボディが gzip されている場合があります。

この機能の詳細については、「*CA Service Virtualization の使用*」の「データ ディセンシタイザ (P. 270)」を参照してください。

第 12 章: 仮想化の実行

仮想化は、サーバが存在しない状態で、VSE がクライアントに応答するプロセスです。 仮想化では、VSM とサービス イメージを使用します。

このセクションには、以下のトピックが含まれています。

仮想化の準備

仮想サービス環境の起動

仮想化を実行するには、仮想サービス環境（VSE）を起動する必要があります。 VSE は、DevTest レジストリに登録する必要があります。

VSE を起動するには、［スタート］-［プログラム］-［DevTest］-［仮想サービス環境］を選択します。

lisa.VSEServer という名前の VSE を作成し、レジストリ インスタンスに接続するためのウィンドウが表示されます。

注: 仮想サービス環境ウィンドウは最小化できますが、このウィンドウを閉じないでください。

DevTest Windows システム サービスをインストール済みの場合は、システム サービスを使用して VSE を起動できます。

また、コマンド プロンプトから以下のコマンドを使用して、名前の付いた VSE を起動することもできます。

[LISA_HOME]¥bin¥VirtualServiceEnvironment.exe -n *VSEName* **-m** *RegistryName*

VSEName

VSE の名前を指定します。

RegistryName

既存のレジストリの名前を指定します。

複数の仮想サービス環境の実行

1 台のコンピュータで複数の VSE サーバを実行するには、**VirtualServiceEnvironment.exe** の実行時に **-p** *port* コマンドライン オプションを追加します。ここで、**port** は、ポート番号を指定します。 ポート番号は、VSE インスタンスによって異なります。

ライセンスの **maxvirtualservices** プロパティによって、実行できる仮想サービスの数が制限されます。

VSE マネージャを使用した VSE の設定

VSE マネージャでは、仮想サービス環境を変更できます。

VSE マネージャの詳細については、「*CA Service Virtualization の使用*」の「VSE マネージャ コマンド (P. 454)」を参照してください。

仮想サービスの展開および実行

次の手順に従ってください:

1. VSE コンソールで、[新しい仮想サービスを環境に展開します]  をクリックします。

   [仮想サービスの展開] ウィンドウが表示されます。

2. アップロードするモデル アーカイブ (MAR) の名前を入力し、[展開] をクリックします。

   - MAR は、仮想サービスを含む必要があります。 [展開] をクリックすると、サービスが VSE コンソールにロードされ、サービスが実行可能になります。
   - または、DevTest ワークステーション のプロジェクト パネルに移動し、仮想サービス モデル (VSM) を右クリックして [仮想サービスの展開] ウィンドウを開きます。

   [仮想サービスの展開] ウィンドウが表示されます。

3. 必要に応じてフィールドを変更します。

   name

   選択した VSM が参照する仮想サービスの名前が表示されます。

   VS モデル

   選択した仮想サービス モデルが表示されます。

   設定

   (オプション) 代替設定ファイルを選択できます。

   グループ タグ

   この仮想サービスの仮想サービス グループ (P. 468)の名前を指定します。 展開した仮想サービスにグループ タグがある場合、それらのタグがドロップダウン リストに表示されます。グループ タグは英数字で始まる必要があり、英数字と以下の特殊文字を含めることができます。

   - ピリオド (.)
   - ダッシュ (-)
   - アンダースコア (_)
   - ドル記号 ($)

同時実行数

負荷容量を示す数値を指定します。 *容量*は、VSM で同時に何人の仮想ユーザ（インスタンス）が実行できるかを示します。 ここでの容量は、このサービス モデルに対する要求を処理するスレッドの数を表します。

VSE は、同時実行数の合計と同数のスレッドを割り当てます。 休止している場合でも、各スレッドはシステム リソースを多少消費します。 したがって、システム全体のパフォーマンスを最適化するため、この設定は可能な限り小さな値にします。必要なパフォーマンスを達成するか、または設定値を大きくしてもさらなるパフォーマンス向上が見られなくなるまで設定を調整することにより、適切な設定値を経験的に決定します。

標準装備のプロトコルは、スレッド使用を最小化するため、フレームワーク レベル タスク実行サービスを使用します。このようなプロトコルの場合、VSM が高度にカスタマイズされていない限り、同時実行数が 1 コア当たり 2 ～ 3 を超えていても、それが役立つことはほとんどありません。

拡張、および標準装備のプロトコルを使用しない VSM では、反応時間を長く設定すると、その期間中にスレッドを消費する可能性があります。この場合、同時実行数を大きくすることをお勧めします。

このような場合のおおよその初期設定値は、以下の数式で与えられます。

同時実行数 = （必要な 1 秒あたりのトランザクション数/1000） * 平均反応時間（ミリ秒） * （反応時間スケール/100）

例：

反応時間を処理するためにフレームワーク タスク実行サービスを使用しない、カスタム プロトコルを使用していると仮定します。全体的なスループットとして必要な、1 秒あたりのトランザクション数は 100 です。 サービス イメージ全体にわたっての平均反応時間は、200 ミリ秒です。また、仮想サービスは、反応時間スケール 100 （%）で展開されます。

（1 秒あたりのトランザクション数 100/1000） * 200 ミリ秒 * （100/100） = 20

この場合、各スレッドは応答前に平均約 200 ミリ秒ブロックされ、その間は新しい要求を処理できません。そのため、100 トランザクション/秒に対応するには、同時実行数 20 が必要です。スレッドは平均で 10 ミリ秒ごとに利用可能になり、100 トランザクション/秒を実現するのに十分となります。

デフォルト： 1

反応時間スケール

記録された反応時間に対する反応時間の割合（%）を指定します。

注: テストの実行でのペースの整合性を保つため、ステップ自体の処理時間は反応時間から引かれます。

デフォルト： 100

例：

- 反応時間を 2 倍にするには、「200」と入力します。
- 反応時間を半分にするには、「50」と入力します。

展開時にサービスを開始する

サービスをすぐに展開および開始するかどうかを指定します。

値

- **オン：** サービスをすぐに展開および開始します。
- **オフ：** サービスを展開し、その後 VSE コンソールから手動で開始します。

サービスが終了した場合、自動的に再起動する

エミュレーション セッションがそのエンド ポイントに到達した後も、サービスの実行を継続するかどうかを指定します。

値

- **オン：** エミュレーション サービスが終了した後もサービスの実行を継続します。
- **オフ：** エミュレーション サービスが終了したときにサービスを停止します。

デフォルト： 選択済み

4. ［展開］をクリックします。

VSE コンソールには、ロードされた仮想サービスのステータスが表示されます。

注: 仮想サービスのステータスには、以下のものがあります。

展開

入力した名前のサービスはまだ展開されていません。 サービスは展開されています。

再展開

入力した名前のサービスは、入力したサービスと同じ .vsm ファイルで展開されています。 サービスは再展開されています。

オーバーライド

入力した名前のサービスは、入力したサービスに関連付けられたものとは異なる .vsm ファイルで展開されています。 展開されているサービスのオーバーライドを促すメッセージが表示されます。

ライブ要求の実行

仮想サービスを展開した後、VSE に対してライブ要求を実行します。 可能な場合は、ライブ サービスを停止し、クライアントが VSE と通信できるようにゲートウェイ/プロキシ設定を設定します。

詳細については、以下を参照してください。

- VSE コンソールへのアクセス (P. 420)
- ツールバー (P. 421)
- ［サービス］タブ (P. 423)
- ［一致］タブ (P. 426)
- ［イベント詳細の要求］タブ

VSE コンソールへのアクセス

展開したサービス イメージを管理およびモニタするには、VSE コンソールを使用します。 このコンソールから、仮想サービスを展開、開始、表示、停止、再展開、および削除することができます。

次の手順に従ってください:

1. DevTest ワークステーション から、［サーバ コンソール］をクリックして、DevTest サーバ コンソールを開きます。
2. VSE サービスを選択し、ダブルクリックして VSE コンソールを開きます。
3. VSE コンソールで、仮想サービスが展開されていることを確認します（ステータスは［実行中］）。
4. VSE コンソールで、トランザクション数（［トランザクション数］）を表示して、サービスが要求を受信したことを確認します。

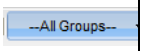
   注: 展開した VS モデルにトランザクションが表示されていない場合、クライアントは正しく設定されていません。 実際のシステムの代わりに仮想モデルを参照するようにクライアントを再設定します。 別のサービスがそのポートを使用している場合は、そのサービスを停止するかポート設定を変更して競合を解消します。

VSE コンソール ツールバー

VSE コンソールには固有のツールバーが存在し、以下のボタンで構成されています。 これらの機能には、サービスにマウス ポインタを重ねて右クリックすることによってもアクセスできます。

VSE コンソールのツールバーには、以下のコマンドが含まれています。

		仮想サービスを環境に展開または再展開します。
		選択した仮想サービスを開始します。
表示		選択された仮想サービスの検査ビューを表示します。
表示		選択された仮想サービスのセッション/トラッキング情報を表示します
設定		選択された仮想サービスの動作を指定します。
リセット		選択された仮想サービスのトランザクション数およびエラー数をリセットします。
停止		選択された仮想サービスを停止します。

削除		選択された仮想サービスを環境から削除します。
設定		トラッキング データのクリーンアップを設定します。
シャットダウン		仮想サービス環境全体をシャットダウンします。
グループ		すべてのグループまたは 1 つのグループに関連付けられたサービス、または関連付けられたグループがないサービスを表示するかどうかを選択します。

環境に新しい仮想サービスを展開します

モデル アーカイブ (MAR) から仮想サービスを展開するには、このオプションを選択します。

アップロードするモデル アーカイブ (MAR) の名前を入力します。MAR は、仮想サービスを含む必要があります。［展開/再展開］をクリックすると、サービスが VSE コンソールにロードされ、サービスが実行可能になります。

選択された仮想サービスを開始します

選択した仮想サービスを開始します。

確認ウィンドウが開き、サービスが開始したことが示されます。 ［OK］をクリックします。

選択された仮想サービスの検査ビューを表示します

検査ビューの表示  ボタンは、仮想サービス用のインスペクタ パネルのタブを開きます。 このパネルには、 ［一致 (P. 426)］と［イベント詳細の要求］の 2 つのタブがあります。

VSE コンソール - [サービス]タブ

[サービス]タブ

［サービス］タブでは、列名の右の下矢印をクリックすることにより、昇順または降順に列の値を並べ替えることができます。 また、この矢印を使用して、このウィンドウに表示する列を選択します。

このタブには、以下のフィールドが含まれます。

名前

現在展開されている仮想サービス モデル

リソース/タイプ

ポート、およびサービスのタイプまたはプロトコル。

ステータス

仮想サービスの現在の状態。

稼働時間

サービスの開始後に経過した時間。

トランザクション数

サービスの開始後に記録されたトランザクション数。

実行モード

仮想サービスの実行モード。

グループ

仮想サービスの仮想サービス グループ (P. 468)。

エラー

赤いドットは、サービスの実行中にエラーが発生したことを示します。

ウィンドウの下部の［仮想サービス詳細］パネルには、サービスの詳細が表示されます。

このパネルには、以下のフィールドが含まれます。

モデル名

現在展開されている仮想サービス モデルの名前。

実行モード

この仮想サービスの実行モードを表示します。 「*CA Service Virtualization の使用*」の「選択したモデルの動作を指定する」を参照してください。

最後の開始日

このサービスが最後に開始された日時。

トランザクション数

サービスの開始後に記録されたトランザクション数。

現在のトランザクション/秒

現在実行中のトランザクション数。

容量

仮想サービス モデルで一度に実行できる仮想ユーザ（インスタンス）数。［容量］は、このサービス モデルで要求を処理するスレッドの数を示します。 このフィールドは、サービスの実行中に更新できます。

グループ タグ

この仮想サービスの仮想サービス グループ (P. 468)の名前。 展開した仮想サービスにグループ タグがある場合、このフィールドに文字を入力するときにそれらのタグを使用できます。グループ タグは英数字で始まる必要があり、英数字およびピリオド（.）、ダッシュ（-）、アンダースコア（_）、ドル記号（$）の 4 つのその他の文字を含めることができます。

グループ タグを入力した後で **Tab** キーまたは **Enter** キーを使用してから、［更新］をクリックしてフィールドを更新する必要があります。

仮想サービスからグループ タグを削除するには、グループ タグ名の隣にある［**X**］をクリックし、次に［更新］をクリックします。

設定名

このサービスが使用する設定ファイルの名前。

自動再起動

このサービスに対して自動再起動オプションが選択されているかどうか。 サービスの実行中に、この値を［はい］から［いいえ］、または［いいえ］から［はい］に切り替えるには、このフィールドをクリックします。

最後の終了日

このサービスが最後に停止した日時。

エラー数

受信したエラーの数。

ピークトランザクション/秒

同時に実行されるトランザクションの最大数。

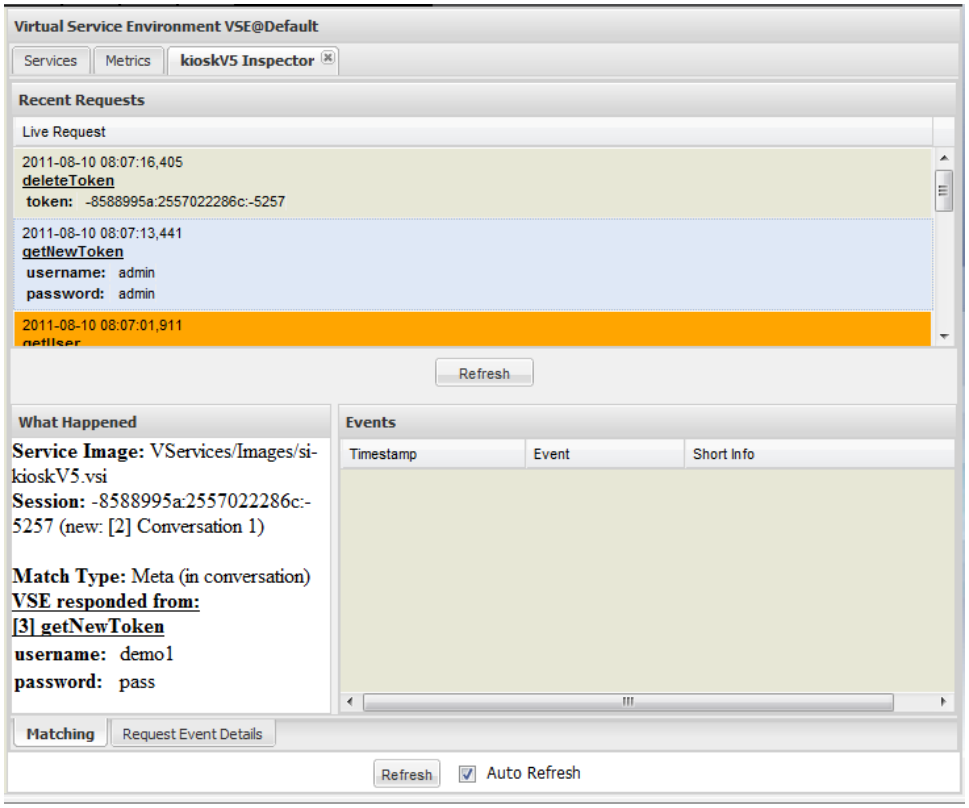
反応時間スケール

記録された反応時間のパーセント値。

VSE コンソールで仮想サービスの名前をクリックすると、仮想サービスのバックアップ アーカイブをダウンロードできます。 この仮想サービスに関連付けられた MAR ファイルを保存するように要求されます。

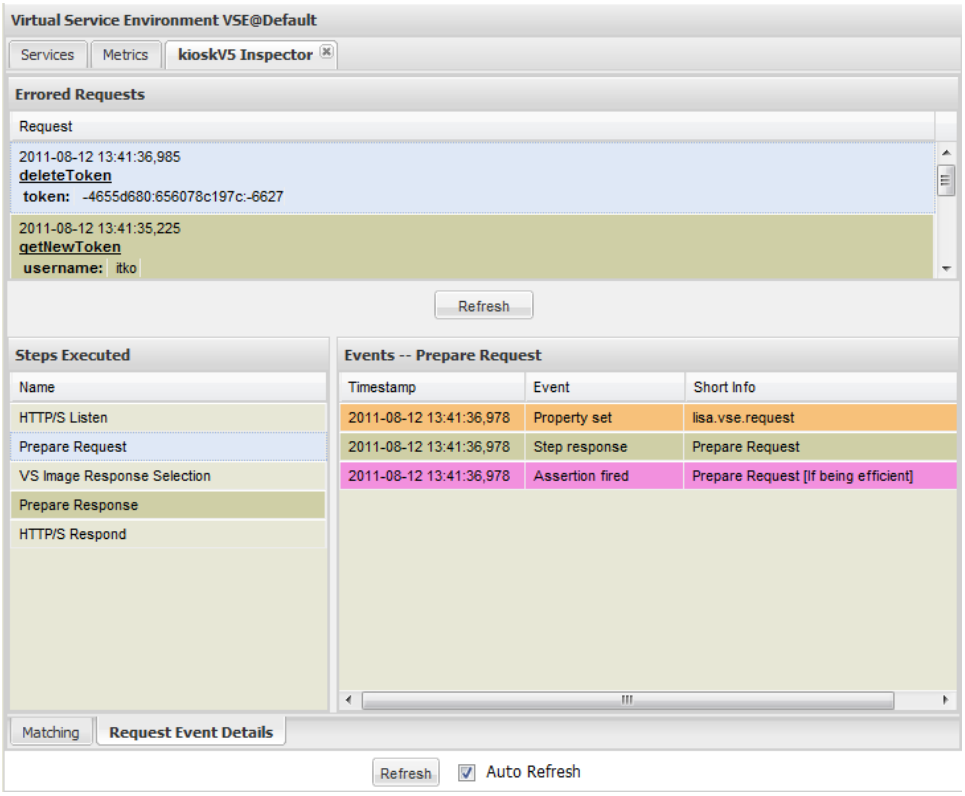
VSE コンソール - [一致]タブ

［一致］タブには、仮想サービスが処理した最近の要求がリスト表示されます。 要求を選択すると、要求がどのように一致した（または、しなかった）かの説明が表示されます。 同じ情報が **vse_matches.log** ファイルに記録されます。 選択した要求にイベントが関連付けられている場合、それらはパネルの右側に表示されます。



VSE コンソールの［イベント詳細の要求］タブ

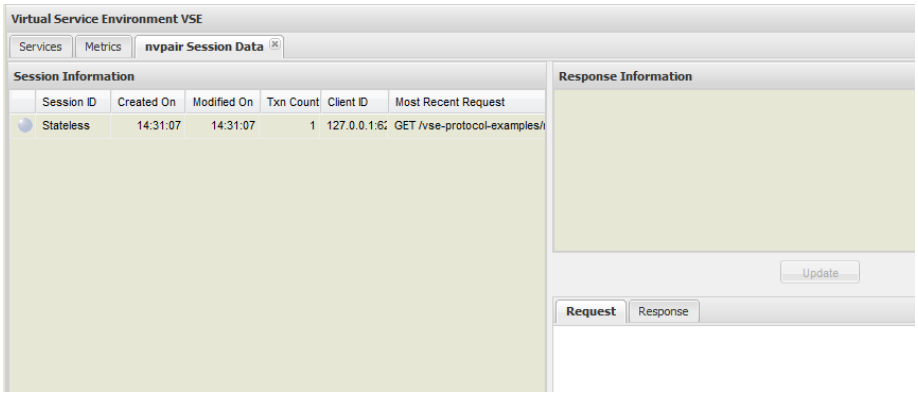
［イベント詳細の要求］タブには、仮想サービスでエラーを発生させたインバウンド要求のリストが表示されます。 要求を選択すると、実行された VSM ステップのリストが表示され、選択したエラー イベントが含まれるステップが示されます。 そのステップの処理中に発生したイベントを参照するには（ITR と同様）、ステップを選択します。



注: 仮想サービスの毎秒のトランザクション数が 100 を超えている場合、プロパティの設定およびプロパティの削除イベントは、全体のパフォーマンスを向上させるために無効になります。

選択された仮想サービスのセッション/トラッキング情報を表示します

このオプションでは、仮想サービスのセッション トラッキング情報を表示できます。 詳細については、「セッションの表示およびモデルの修正 (P. 433)」を参照してください。

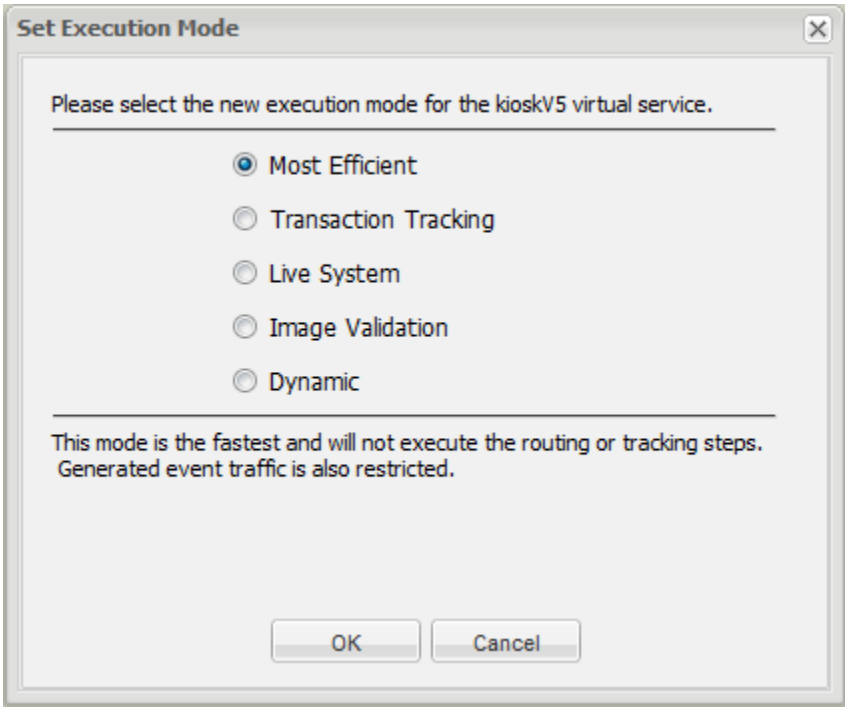


選択された仮想サービスを環境に再展開します

サービス イメージまたは VSM を編集する場合は、変更を保存し、［展開］/［再展開］をクリックして VSE コンソールで変更されたサービス イメージを再展開します。

選択したモデルの動作を指定する

このオプションでは、仮想サービスの実行モードを設定できます。 仮想サービスに使用可能な実行モードの数は、テスト ステップのタイプによって異なります。 たとえば、モデルにライブ呼び出しステップがない場合、そのモデルでは［Learning］または［ライブ呼び出し］オプションがサポートされません。



使用可能な実行モードは以下のとおりです。

最も効率的

最も高速のモード。ステップのルーティング とトラッキングは実行されません。 また、このモードは、生成されたイベントのトラッキングを制限します。

代役

代役モードは、最初に仮想サービスに要求をルーティングします ([最も効率的] モードと同様) 。ただし、仮想サービスの応答がない場合、要求はライブ システムに自動的にルーティングされます。ライブ呼び出しステップを持つ仮想サービスについてのみ、代役モードを有効にできます。 代役モードでは、特別なトラッキングは実行されません。単に、仮想サービスがライブ サービスにフォールバックすることを可能にします。

トランザクション トラッキング

このモードでは［最も効率的］より多くのイベントが起動され、セッションのトランザクション フローを記憶します。このトランザクション情報は、特定の要求に対して特定の応答が選択された理由を判断するために役立ちます。 このモードは、［最も効率的］ほど効率的に実行されません。トランザクション トラッキング モードでは、ライブ システム応答は表示されません。サービス イメージからの応答のみ表示されます。

ライブ システム

このモードでは、モデルのライブ呼び出しステップを使用して、現在の要求に対する応答を決定します。 仮想サービスからの応答を使用する代わりに、ライブ サービスにアクセスして応答を取得します。 ライブ呼び出しのターゲット システムは、パフォーマンスを制御します。このモードは、パススルーとも呼ばれます。

フェールオーバ

フェールオーバ モードは、最初にライブ システムに要求をルーティングします（［ライブ システム］モードと同様）。 ただし、ライブ システムの応答がない場合、要求は仮想サービスに自動的にルーティングされます。 このモードは、代役の反対です。 ライブ呼び出しステップを持つ仮想サービスについてのみ、フェールオーバ モードを有効にできます。 フェールオーバ モードでは、ライブ呼び出しステップが実際に失敗すると、サービス イメージを使用して応答が決定されます（ライブ システムが使用可能できなかった場合と同様）。

学習

学習モードは、イメージの検証モードと似ていますが、ライブ システムからの新しい応答または更新された応答を持つように仮想サービスを自動的に「修復」または修正します。 仮想サービスに自動的に渡される次の要求は、「学習された」新しい応答を参照します。 学習のために 1 つのシステムのみが確認されるわけではなく、両方のシステムが確認され、現在はライブ システムが優位性を持っています。

仮想サービスが学習モードで実行されており、新しい知識を取得した場合、VSE コンソールの仮想サービス名の左に  アイコンが表示されます。 このアイコンは、仮想サービスを再展開するまで表示されたままになります。

イメージの検証

このモードでは、VSE とライブ システムの両方を使用して、現在の要求に対する応答を取得します。応答は、ログに記録され、後で、［View Session and Tracking］パネルを使用してサービス イメージに適用できます。 このモードでは、VSE および対応するライブ システムによって提供される応答の間でライブ比較が可能となり、違いが存在する場合、VSE サービス イメージに対してパッチまたは修正を行うことで、ライブ システムとの同期を保つことができます。 このモードは、「ライブ修正モード」とも呼ばれます。イメージの検証モードは、すべてのモードの中で最も非効率的なモードです。

動的

このモードでは、モデルがその他のモードのどれを使用するかを要求ごとに決定できます。 そのため、パフォーマンスは予測できません。唯一の要件は、プロトコルの 1 つまたは複数のリスン ステップの後にモデルに VS ルーティングが存在することです。

選択された仮想サービスのトランザクション数およびエラー数をリセットします

選択した仮想サービスのトランザクション数とエラー数の両方をゼロにします。

選択された仮想サービスを停止します

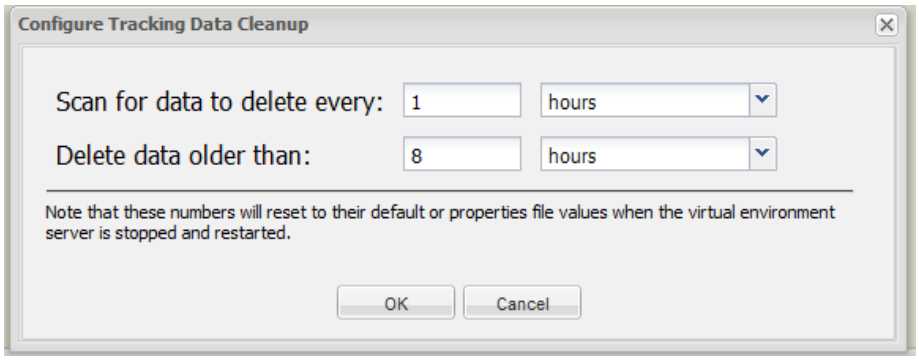
選択した仮想サービスを停止するには、このボタンをクリックします。サービスが停止される前に、確認メッセージが表示されます。

選択された仮想サービスを環境から削除します

選択した仮想サービスをコンソール表示から削除するには、このボタンをクリックします。 サービスが削除される前に、確認メッセージが表示されます。

トラッキング データのクリーンアップを設定します

［トラッキング データのクリーンアップの設定］ウィンドウを開きます。



ここで、サービスが停止および再起動されるまで有効となるデータ クリーンアップ値を入力します。 サービスが再起動されると、この値はデフォルト値またはプロパティ ファイルに設定された値に従ってリセットされます。

削除対象データのスキャン間隔

削除するトラッキング データをスキャンする間隔（ミリ秒、秒、分、時、日、週）を定義します。

データを削除するまでの保持期間

削除するまでのデータ保持期間（ミリ秒、秒、分、時、日、週）を定義します。

仮想サービス環境全体をシャットダウンします

VSE 環境全体をシャットダウンするには、このボタンをクリックします。シャットダウンの確認メッセージに対してはいと回答すると、VSE がシャットダウンされ、対応するウィンドウが閉じられます。

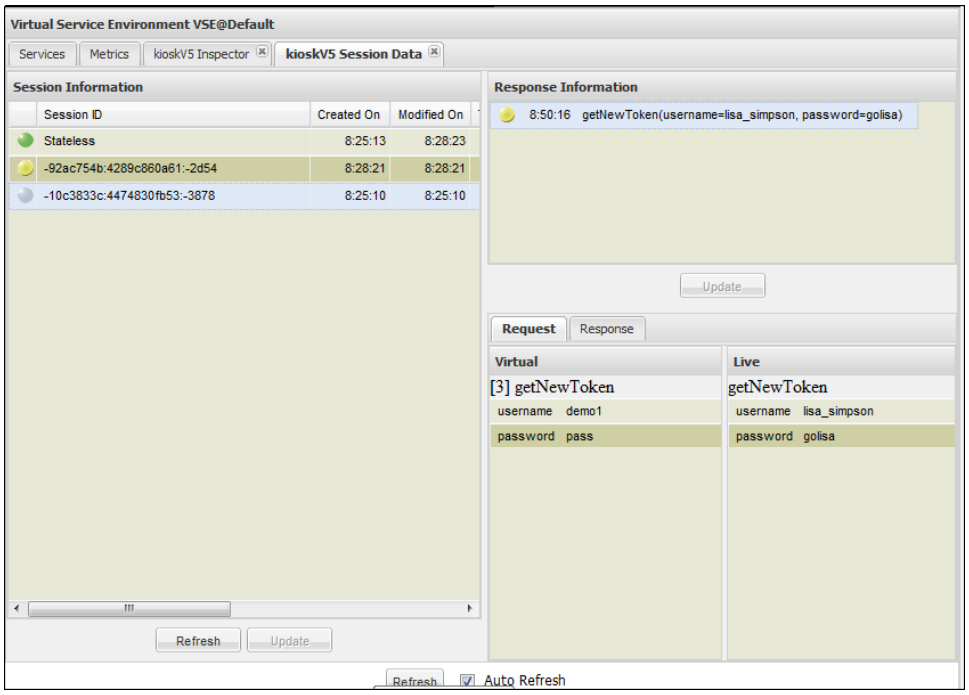
セッションの表示およびモデルの修正

*セッションの表示*では、VSE ユーザは、VSE サーバ上の現在（または最近の）セッションの動作を実際に確認できます。 ユーザは、特定の要求に対して応答が返された理由を判断できます。 また、セッションの表示では、VSE および対応するライブ システムによって提供される応答の間で比較が可能となります。 違いが存在する場合、アプリケーションは、*モデルの修正*を使用して VSE サービス イメージに対してパッチまたは修正を行うことで、ライブ システムとの同期を保つことができます。

セッションの表示は、トランザクション トラッキングまたはイメージの検証実行モードで実行される仮想サービスに対してのみ使用できます。 モデルの修正は、イメージの検証モードで実行される仮想サービスに対してのみ使用できます。 実行モードの詳細については、「選択したモデルの動作を指定する」を参照してください。

学習では、CA Service Virtualization およびライブ応答の違いが検出されるとただちにサービス イメージが変更されるため、モデルの修正と学習は異なります。 修正では、後で［View Session and Tracking］情報パネルによって確認し、サービス イメージに適用するために、それらの違いがログに記録されます。

セッションの表示とモデルの修正は、VSE ダッシュボードからアクセス可能なパネルを使用することによって、または仮想サービス モデルの編集時に実行できます。



選択した仮想サービスに関するセッション トラッキングを表示するには、VSE コンソールの［サービス］タブからサービスを選択し、セッション/トラッキング情報  をクリックします。

このアイコンをクリックすると、セッション パネルが開き、記録されたセッションとトランザクションが表示されます。 展開した仮想サービスに記録されたトランザクションが存在しない場合、セッション パネルがセッション情報なしで表示されます。

セッション パネルは、［セッション情報］と［応答情報］の 2 つのペインで構成されています。

［セッション情報］ペイン

［セッション情報］ペインには、選択した仮想サービスのすべてのセッションが表形式でリスト表示されます。 表を並べ替えるか、または表示されている列を選択またはクリアするには、各列見出しの右の矢印をクリックします。

セッション ステータス

現在のセッション ステータスを示すアイコンを表示します。

灰色のボールは、トランザクション トラッキング モードで記録されたセッション、またはモデルの修正を使用して修正されたセッションを示します。

赤のボールは、イメージの検証モードで記録された修正の候補であるセッションを示します。

緑のボールは、イメージの検証モードで記録された、ライブ応答と VSE 応答が一致するセッションを示します。

セッション ID

各セッションの一意の ID を示します。

作成日

そのセッションの最初のトランザクションのタイムスタンプを表示します。

変更日

最新のトランザクションのタイム スタンプを表示します。

トランザクション数

サービスの開始後に記録されたトランザクションの数を示します。

クライアント ID

（プロトコルに固有）HTTP では、トランザクションをサブミットしたクライアントのエンドポイントを表示します。

最新の要求

特定のセッションでの最新の要求を示します。

［応答情報］ペイン

［応答情報］ペインには、選択したセッションのトランザクションのリストが表示されます。 このペインの最初の列にある色付きのボールにマウス ポインタを重ねると、ツールヒントによって、そのトランザクションの一致が示されます。 特定のトランザクションをクリックすると、ペインの下部に要求と応答のタブが表示されます。 これらのタブは、VSE システムとライブ システムの間で要求/応答を比較します。

［応答情報］ペインでは、以下のアイコンがトランザクションを表します。

- 緑色のボール: メタ トランザクションでのシグネチャの一致を示します。

- 黄色のボール: メタ トランザクションでのシグネチャの一致を示します。イメージ ナビゲーションは成功しますが、VSE とライブ システムとの間で応答ボディが異なります。
- 赤のボール: ライブ システムと VSE イメージとの間で異なる会話型トランザクションを示します。

ライブ セッション/ステートレス トランザクションのサービス イメージを更新するには、［更新］ボタンを使用します。 このプロセスは、モデルの修正と呼ばれています。モデルの修正を使用して VSE イメージとライブ システムの間のディスパリティを除去し、VSM が正しく動作するようにします。 ［更新］をクリックすると、赤のボールでマークされたセッションが灰色のボールに変わります。 このセッションは、現在追跡されています。

- ［応答情報］の［更新］ボタンは、選択したトランザクションのサービス イメージを更新します。
- ［セッション情報］の［更新］ボタンでは、［セッション情報］ペインに表示されている複数のセッションを選択して、それらを同時に更新できます。

VSE メトリック

VSE メトリックの表示

DevTest コンソールの［メトリック］タブでは、VSE メトリックを表示できます。

次の手順に従ってください:

1. アクティブなサービスのリストを表示するには、［VSE サービスの選択］をクリックします。
2. 表示するサービスを選択します。
3. 選択したサービスで使用可能なチャートを表示するには、［チャートの選択］をクリックします。
4. チャートを選択し、［チャートの生成］をクリックします。

   表示をリフレッシュするには、［Refresh］をクリックします。データをより見やすくするには、ズーム スライダおよび スクロール スライダを使用してチャートを移動します。

注: メトリック チャートを表示するには、サーバ コンソールへのアクセスに使用するブラウザに、Adobe Flash プラグインをインストールする必要があります。

メトリック収集の定義

VSE メトリック収集のパラメータを定義するには、**lisa.properties** ファイル内の以下のプロパティを更新します。

- lisa.vse.metrics.collect
- lisa.vse.metrics.txn.counts.level
- lisa.vse.metrics.sample.interval
- lisa.vse.metrics.delete.cycle
- lisa.vse.metrics.delete.age

注: 詳細については、「*使用*」の「付録 A - LISA プロパティ ファイル」を参照してください。

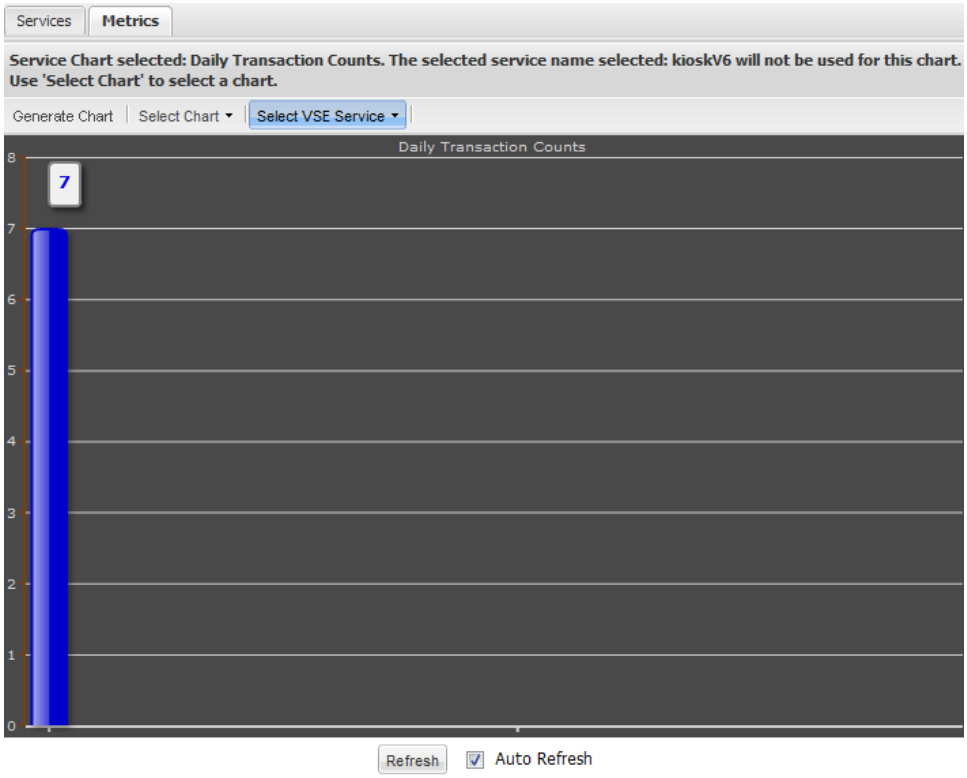
サーバ チャート メトリック

サーバ チャート メトリックは、この仮想サーバ（VSE）上のすべてのアクティビティに適用されるメトリックです。

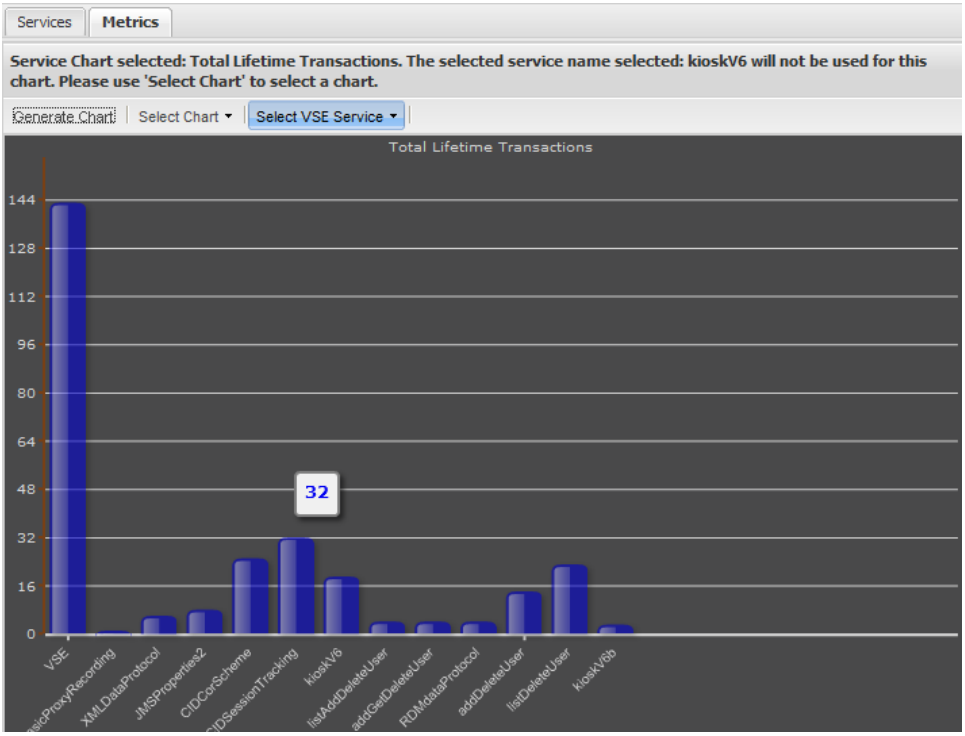
注: サーバ チャートを選択すると、パネル見出しに、選択された仮想サービスがこのチャートに使用されないことが示されます。 これらのチャートは、選択したサービスだけでなくサーバ全体に適用されます。

サーバの可用性

日単位トランザクション数



ライフタイム トランザクション合計



サービス チャート メトリック

サービス チャート メトリックは、各仮想サービスに適用されるメトリックです。 右ペインには、この環境に展開された仮想サービスのリストが表示されます。それらの 1 つを選択して、リストからチャートを選択できます。

応答時間


Services
Metrics
Service Chart selected: Response Time. Service name selected: kioskV6. Use 'Select Chart' to select a chart.
Generate Chart
Select Chart
Select VSE Service
Response Time : kioskV6
Average
Min
Max
1,276
1,136
996
856
716
576
436
296
156
16
02/08/12
08:36

強制された遅延

強制された遅延

VSE が結果を送信する前のミリ秒数を示します。

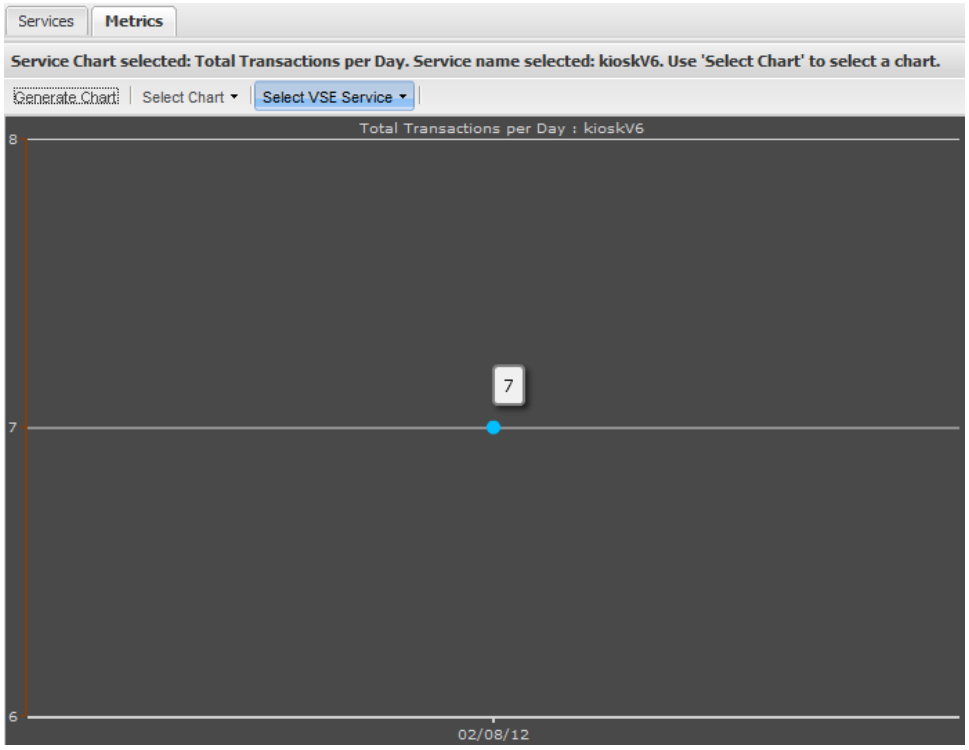
正の数は、指定された反応時間のため、VSE が結果を送信するのを意図的に待機したことを意味します。 たとえば、反応時間が 10 ミリ秒で、リスンして応答する時間が 8 ミリ秒だった場合、VSE は 2 ミリ秒待機します。

負の数は、VSE が結果を生成するのに、反応時間よりも長い時間を要したことを意味します。 たとえば、反応時間が 10 ミリ秒である一方、リスンして応答するのに 12 ミリ秒かかりました。 そのレポートでは、-2 が表示されます。

1 秒あたりのトランザクション数

トランザクションのヒット数およびミス数

1 日あたりの合計トランザクション数



トランザクション スループット

第 13 章: VSE マネージャ - 仮想サービスの管理および展開

VSE マネージャは、DevTest 仮想サービスを展開および管理できる Eclipse プラグインです。

VSE マネージャは Eclipse リリース 4.3 以降でサポートされています。Eclipse は必ず Java 7 JVM を使用して実行します。

このセクションには、以下のトピックが含まれています。

VSE マネージャのインストール

VSE マネージャをインストールするには、新しいソフトウェアを追加するための Eclipse の標準的な手順に従います。

次の手順に従ってください:

1. Eclipse で、［Help］-［Install New Software］を選択します。

   インストールのダイアログ ボックスが表示されます。

2. ［Add］を選択し、以下の VSE Provisioning Platform（p2）リポジトリ URL を追加します。

   `http://www.itko.com/downloads/eclipse/8.0/updates/`

3. ［VSE Manager UI］機能を選択し、［Next］をクリックします。

   ［Install Details］ウィンドウが表示されます。

4. インストールの詳細を確認し、［Next］をクリックします。

   ［Review Licenses］ウィンドウが表示されます。

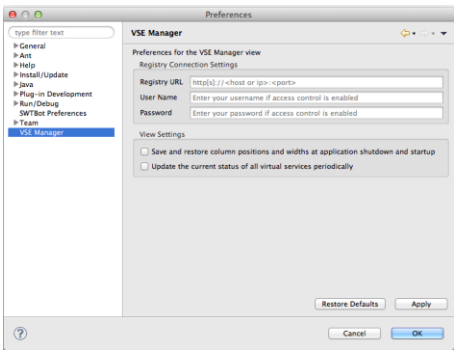
5. ライセンスを確認して同意し、［Finish］をクリックします。

VSE マネージャの使用

VSE マネージャの設定

VSE マネージャを使用する前に、Eclipse の［Preferences］ダイアログ ボックスにある［VSE Manager］ページを設定する必要があります。

以下の図は、［VSE Manager］ページが選択されている［Preferences］ダイアログ ボックスを示しています。



注:

- ［Registry URL］フィールドを **https** URL に設定してください。「*管理*」で説明されているように、このタイプの URL を指定するには、DevTest コンソールとの HTTPS 通信が有効になっていることが必要です。
- 必ず［User Name］フィールドと［Password］フィールドに値を入力します。これらのフィールドを空白のままにすると、Eclipse の［Password Required］ダイアログ ボックスが表示されます。これは VSE マネージャの正しいダイアログ ボックスではありません。［Password Required］ダイアログ ボックスが表示された場合は、ユーザ名およびパスワードを入力せずに、［Cancel］をクリックします。

次の手順に従ってください:

1. レジストリ接続を設定するには、VSE Web サーバの URL を入力します。
2. 必要に応じて、ユーザ名とパスワードを入力します。
3. （オプション）ビューを閉じて再び開いたときに列の順序およびそれらの関連する幅を保存するには、［Save and restore...］チェック ボックスをオンにします。
4. （オプション）ビューが約 10 秒ごとに自動的にコンテンツをリフレッシュするようにするには、［Update the current status...］チェック ボックスをオンにします。

5. ダイアログ ボックスを完了して閉じるには、［OK］をクリックします。

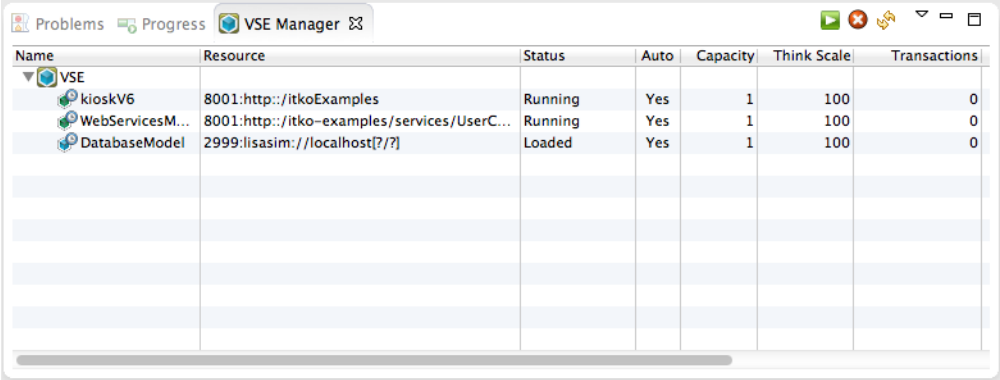
LISA プロジェクトのインポート

次の手順に従ってください:

1. メイン メニューから、Eclipse のインポート ウィザードを開きます。
2. CA LISA カテゴリから、［LISA Project］を選択します。
3. LISA プロジェクトを選択し、［Project Name］フィールドにその名前を入力します。
4. ［終了］をクリックします。

   プロジェクトがインポートされます。

［VSE Manager］ビューの使用

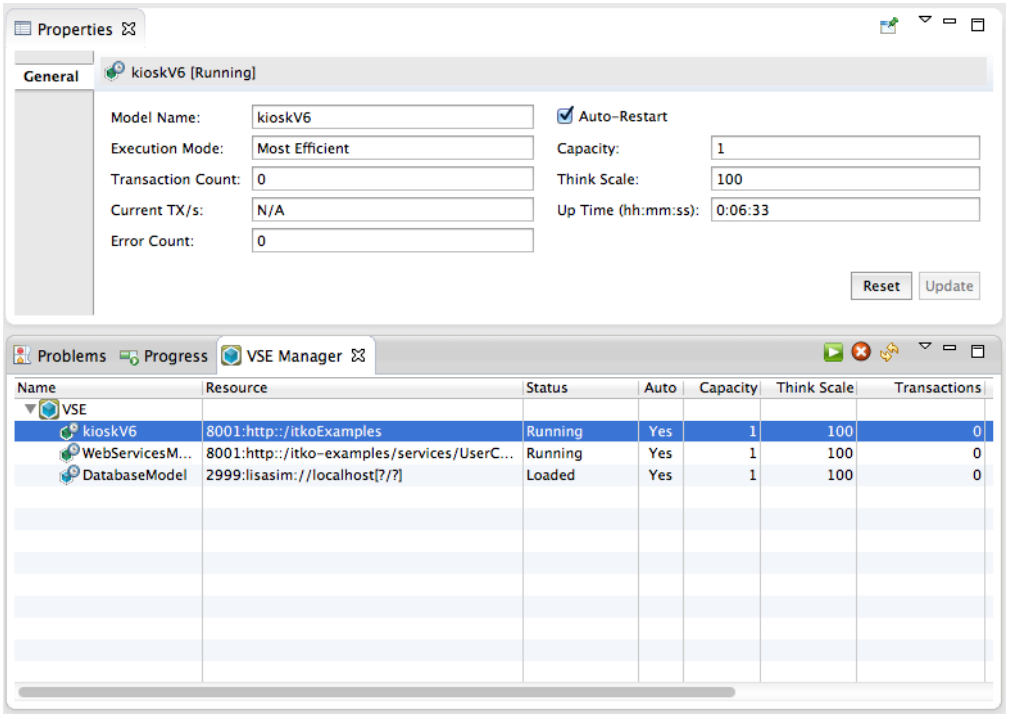


［VSE Manager］ビューは、以下のアクションをサポートしています。

- ドラッグ アンド ドロップによるビューの列の並び替え。
- ドロップダウン リストでの選択による列の表示または非表示。
- 選択された仮想サービスの開始、停止、または展開解除。
- Eclipse ベースの製品およびネイティブ ファイル システムからのドラッグ アンド ドロップによる VSE への新しい仮想サービスの展開（MAR ファイル形式）。

［Properties］ビューの統合

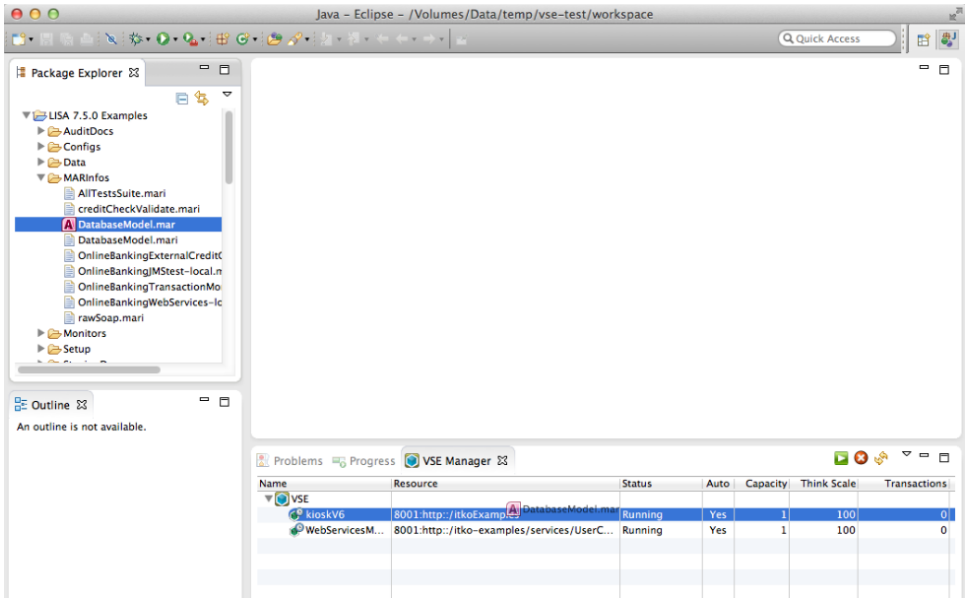
仮想サービスが［VSE Manager］ビューで選択されており、［Properties］ビューが開いている場合、選択したサービスに関してより多くの情報を表示および更新できます。



このビューでは、以下のことが可能です。

- VSE サーバの起動時のサービスの開始を自動的に制御できます。
- 展開した仮想サービスの容量を変更できます。
- 仮想サービスに対する要求の反応時間スケールを変更できます。

VSE マネージャに MAR ファイルをドラッグして展開できます。


Java – Eclipse – /Volumes/Data/temp/vse-test/workspace
Quick Access
Package Explorer
LISA 7.5.0 Examples
AuditDocs
Configs
Data
MARInfos
AllTestsSuite.mari
creditCheckValidate.mari
DatabaseModel.mar
DatabaseModel.mari
Monitors
Setup
Outline
An outline is not available.
Problems
Progress
VSE Manager
Name
Resource
Status
Auto
Capacity
Think Scale
Transactions
VSE
kioskV6
Running
Yes
1
100
0
WebServicesM...
8001:http://itko-examples/services/UserC...
Running
Yes
1
100
0

第 14 章: VSE コマンド

このセクションには、以下のトピックが含まれています。

VSE マネージャ コマンド - 仮想サービス環境の管理

仮想サービス環境を管理するには、VSE マネージャのコマンド ライン ツールを使用します。LISA 6.0 以降の実装では、コマンド ライン ツールを使用するようお勧めします。 詳細については、「サービス マネージャ コマンド (P. 455)」を参照してください。

VSE マネージャは、<lsrv> の bin ディレクトリにツールとして含まれています。

VSE マネージャのコマンドの形式は以下のとおりです。

```
VSEManager [--registry 名 | --vse 名 | --username=名 |--password=名 ]

[--status [VS 名 | ALL ] | --deploy アーカイブ ファイル | --redeploy アーカイブ ファイル | --update VS 名 [capacity 容量] [thinkscale スケール] [auto-restart [ON|OFF] [grouptag グループ タグ] | --remove VS 名 | --start VS 名 | --set-exec-mode VS 名 --mode 実行モード | --stop VS 名 | --help | --version
```

--registry *名前*

処理する DevTest レジストリの名前を設定します。

注: コマンドが実行される前に、--registry パラメータを定義する必要があります。 定義しないと、コマンドは、デフォルトではローカル レジストリで実行されます。ローカル レジストリがない場合、コマンドは実行されません。

--vse *名前*

処理する仮想サービス環境の名前を設定します。

--username=*名前*

ACL 対する認証に使用する ID を設定します。

--password=*名前*

ACL を使用する場合、認証するユーザのパスワードを設定します。

--status [*VS 名* | ALL]

現在の環境の 1 つまたはすべての仮想サービスのステータスを出力します。

--deploy *アーカイブ ファイル*

新しい仮想サービスを現在の環境に展開します。アーカイブ ファイルには、そのプライマリ アセットとして仮想サービス モデルが存在する必要があります。

--redeploy *アーカイブ ファイル*

既存の仮想サービスを現在の環境に再展開します。

--update *VS 名*[**c**apacity *容量* [--thinkscale *スケール*] [--auto-restart [ON|OFF] [--grouptag *グループ タグ*]

指定した仮想サービスの容量、反応時間スケール、グループ タグ、自動再起動の設定、またはこれらのパラメータの任意の組み合わせを更新します。

--remove *VS 名*

指定した仮想サービスを現在の環境から削除するために使用します。

--start *VS 名*

指定した仮想サービスを現在の環境で開始するために使用します。

--set-exec-mode *vs-name* --mode [DYNAMIC | VALIDATION | LIVE | TRACK | EFFICIENT | LEARNING | STAND_IN | FAILOVER]

現在の環境内の指定した仮想サービスの実行モードを設定します。

--stop *VS 名*

現在の環境内の指定した仮想サービスを停止するために使用します

--help

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。

ServiceManager コマンド -- サービスの管理

ServiceManager コマンド ライン ツールを使用して、DevTest サービスをモニタ、リセット、停止します。 このコマンドは、任意の DevTest レジストリ、シミュレータ、コーディネータ、および VSE サーバに適用されます。サービス マネージャは、DevTest サーバ の bin/ ディレクトリに含まれています。

詳細については、「*管理*」の「ServiceManager」を参照してください。

ServiceImageManager コマンド -- サービス イメージの管理

新しいまたは既存のサービス イメージにトランザクション（RAW またはセッション）をインポートするには、VSE ServiceImageManager コマンド ライン ツールを使用します。 このツールでは、2 つ以上のサービス イメージを組み合わせることもできます。 ServiceImageManager は、LISA_HOME¥bin ディレクトリにあります。

レコーディングは、対話型、時間制限、切断の 3 つの方法のいずれかで行うことができます。3 つすべてのスタイルについて、**--vrs=** および **--si-file=** 引数が必要です。 必要に応じて、対話型および切断スタイルでは、**--go** を指定して開始シグナルの待機をスキップし、レコーディングをすぐに開始できます。

- **対話型：** 対話型スタイルは、**--record** 引数のみを指定した場合に使用されます。 レコーダの準備が完了すると、コンソールで Enter キーが押されるまで、レコーディング プロセスの開始を待機します。 次に、レコーダは、プロセスを停止するために Enter キーが再度押されるのを待機します。
- **時間制限：** 時間制限スタイルは、**--record** 引数と **--stop=** 引数（および必要に応じて **--start=** 引数）を指定した場合に使用されます。

切断： 切断スタイルは、**--record** 引数と **--port=** 引数を指定した場合に使用されます。 これにより、**--signal** 引数が制御シグナルをレコーダへ渡すために使用するリスナが設定されます。 レコーダが切断スタイルで開始されたら、別のプロセスでこのツールを使用して、開始および停止シグナルをポート経由で送信して制御できます。 この制御には、**--signal=** および **--port=** 引数のみ指定する必要があります。

インポートを行うには、インポートする RAW またはセッション トラフィック トランザクション ファイルを **--import** 引数の後に指定します。 インポートの実行方法を制御するには、トランスポート、要求、および応答データ プロトコルと、ナビゲーション許容差を使用します。 **-vrs=** 引数をプロトコルまたは許容差引数と共に使用する場合、ほかの引数の前に配置します。 **-vrs=** 引数は、初期状態へのリセットを行います。 インポートするサービス イメージが含まれるファイルの名前を指定するには、**--si-file** 引数を使用します。 少なくとも、以下の引数を指定します。

--import=、**--si-file=**、および **--vrs=** または **--transport=** のいずれか

さらに、以下の引数を追加で指定します。

--request-data=、**--response-data=**、**--non-leaf=**、または **--leaf=**

組み合わせるには、**--combine** 引数の後にターゲット サービス イメージを指定します。 **--combine=** が定義するファイル名はターゲット VSI であり、リストされるその他のファイルはソースです。 類似トランザクションの組み合わせ方法を制御し、ターゲット イメージに組み合わせるソース イメージのファイル名をリストするには、**--favor** を使用します。 少なくとも、以下の引数を指定します。

--combine=

さらに、以下の引数を追加で指定できます。

--favor=

組み合わせ操作用のソース ファイルがリストされます。

注: 値にスペースが含まれる引数（プロトコル名など）は、引用符で囲む必要があります。 例： **"--import=my file.xml"** および **-i"my file.xml"**

ServiceImageManager コマンドの形式は以下のとおりです。

--help -h

ヘルプ テキストを表示します。

--record -d

サービス イメージ ファイルにトランザクションを記録します。

--vrs=*レコーディング セッション ファイル* **-v *レコーディング セッション ファイル***

レコーディングまたはインポートのすべての設定情報を含むレコーディング セッション ファイルを指定します。

--go -G

対話型および切断スタイルで指定します。開始シグナルの待機がバイパスされます。

--start=***時間の指定*** -S ***時間の指定***

指定時間後に、レコーダがレコーディングを開始するように指定します。

--stop=***時間の指定*** -E ***時間の指定***

指定時間後に、レコーダがレコーディングを停止するように指定します。 この時間は、レコーダが記録を開始した時点からの時間で、ミリ秒単位で指定します。

--port=*ポート番号* -P *ポート番号*

レコーダ制御に使用するポートを指定します。 **--record** と共に使用した場合、レコーダが制御をリスンするポートが指定されます。 **--signal** と共に使用した場合、開始/終了シグナルが送信されるポートが指定されます。

--signal=start|stop -g start|stop

別のプロセスでレコーダに送信するシグナルを指定します。 このコマンドには、**--port=** 引数も必要です。

--import=*RAW/トラフィック ファイル* -i *RAW/トラフィック ファイル*

指定された RAW またはセッション トラフィック XML ドキュメントをサービス イメージ ファイルにインポートします。

--transport=*プロトコル* -t *プロトコル*

インポート中に使用するトランスポート プロトコルを指定します。

値

- HTTP/S
- IBM MQ シリーズ
- JMS
- 標準 JMS (非推奨)
- Java
- TCP
- JDBC （ドライバ ベース）
- CICS LINK <DPL>、DTP <MRO、LU6.1、LU6.2>
- CICS トランザクション ゲートウェイ <ECI>
- DRDA
- IMS 接続
- JCo 経由の SAP RFC
- JCo IDoc プロトコル

不透明データ処理

--request-data=*プロトコル* -r *プロトコル*

インポート中に使用する要求側のデータ プロトコルを指定します。

オプション：

Web サービス（SOAP）

Web サービス（SOAP ヘッダ）

Web サービス ブリッジ

WS-Security 要求

要求データ マネージャ

要求データ コピー

自動ハッシュ トランザクション ディスカバリ

ジェネリック XML ペイロード パーサ

データ ディセンシタイザ

Copybook データ プロトコル

区切りテキスト データ プロトコル

スクリプタブル データ プロトコル

DRDA データ プロトコル

XML データ プロトコル

CICS Copybook データ プロトコル

--response-data=*プロトコル* -R *プロトコル*

インポート中に使用する応答の側データ プロトコルを指定します。

オプション：

WS-Security のレスポンス

データ ディセンシタイザ

Copybook データ プロトコル

CTG データ プロトコル

区切りテキスト データ プロトコル

スクリプタブル データ プロトコル

DRDA データ プロトコル

CICS Copybook データ プロトコル

--non-leaf=CLOSE | WIDE | LOOSE -n CLOSE | WIDE | LOOSE -n *許可差*

作成された任意の非リーフ会話ノードに適用されるデフォルトのナビゲーション許容差を指定します。

デフォルト： WIDE

--leaf=CLOSE | WIDE | LOOSE -l CLOSE | WIDE | LOOSE -l *許可差*

作成された任意のリーフ通信ノードに適用されるデフォルトのナビゲーション許容差を指定します。

デフォルト： LOOSE

--config=*設定ファイル* -C *設定ファイル*

レコーディング モードで、使用する設定ファイルを指定します。

--si-file=*VSI ファイル* -s *VSI ファイル*

トランザクションをインポートするサービス イメージ ファイルの名前を指定します。 このファイルが存在しない場合は、作成されます。存在する場合には、インポートされたトランザクションは、既存のサービス イメージにマージされます。

--vsm_file=*VSM ファイル* -m *VSM ファイル*

レコーディング中に作成する仮想サービス モデル ファイルの名前を指定します。

--combine=*ターゲット VSI ファイル* -c *ターゲット VSI ファイル*

1 つ以上のサービス イメージ ファイルを、指定されたサービス イメージ ファイルへ組み合わせます。 **favor** 引数が指定されていない場合、ソース サービス イメージが優先されます。

--favor=source | target -f source | target

サービス イメージを組み合わせる際、類似トランザクションを組み合わせる方法を指定します。 **source** を指定すると、ターゲット側の類似トランザクション（およびその他のデータ）がソース側のトランザクションで更新されます。 **target** を指定すると、ターゲット側の類似トランザクション（およびその他のデータ）は変更されません。

--version

バージョン番号を出力します。

ServiceImageManager コマンドには、以下の 2 つのエラー コードがあります。

1 - 引数の例外が発生した場合（終了ステータスは不良なパラメータ）

2 - その他の一般的なエラーが発生した場合

VirtualServiceEnvironment コマンド

VirtualServiceEnvironment コマンドは VSE アプリケーションの管理に使用されます。 この実行可能ファイルは、[LISA_HOME]¥bin ディレクトリにあります。

コマンドの形式は以下のとおりです。

`VirtualServiceEnvironment [--help|-h] [--name=`*名前*`|-n=`*名前*`] [--registry=`*レジストリの指定*`|-m=`*レジストリの指定*`] [--labName=`*ラボ名*`|-l=`*ラボ名*`] [--force|-f] [--port=`*ポート番号*`|-P=`*ポート番号*`] [--version]`

--help -h

ヘルプ テキストを表示します。

--name=***名前*** -n ***名前***

環境サーバ用のサービス名を定義します。

デフォルト： システムのプロパティ **lisa.vseName**。 システム プロパティのデフォルト値は、**VSEServer** です。

--registry=***レジストリの指定*** -m ***レジストリの指定***

接続先のレジストリ。

例：

-m=tcp://localhost:2010/registry1

--labName=***ラボ名*** -l ***ラボ名***

使用するラボの名前を指定します。

デフォルト: Default

--force -f

このサーバを強制的に DevTest レジストリに登録し、環境のサービス名ですでに登録されているすべてのオブジェクトを置換します。

--port=***ポート番号*** -P ***ポート番号***

VSE が発行するサービス ポートを指定します。 これは、-name 引数の一部としてポートを指定するのと同じです。

--username=**ユーザ名** -u username

DevTest セキュリティのユーザ名を指定します。

--password=**パスワード** -p password

DevTest セキュリティのパスワードを指定します。

--version

バージョン番号を出力します。

第 15 章: VSE Java エージェント プロパティ

DevTest Java エージェントの設定プロパティには、VSE に適用されるいくつかのプロパティが含まれます。

これらのプロパティは、DevTest ポータルの ［エージェント］ ウィンドウから設定できます プロパティは、［設定］ タブに表示されます。

起動モード

起動時の仮想化モードを指定します。

値

- パススルー
- レコーディング
- 再生

浅いレコーディング

トップ レベル（スタック フレーム内）の仮想化されたメソッドでのみコールバックを実行することを指定します。

最大グラフ サイズ

XML シリアル化オブジェクト グラフの最大サイズ（バイト）を定義します。

返信タイムアウト

再生時に VSE の返信に許可される最大間隔を指定します。 この値は、テストの反応時間より大きい必要があります。

参照により省略

参照および void メソッドが変更する引数を記録または再生するかどうかを指定します。

値

- **オン：** 参照および void メソッドが変更する引数を記録または再生しません。 引数の状態を変更する void メソッドを仮想化しない場合は、このチェック ボックスをオンにします。void メソッドが VSE に送信されず、すぐに返されるため、パフォーマンスが向上します。
- **オフ：** 参照および void メソッドが変更する引数を記録または再生します。

キャッシュ応答

応答オブジェクト キーをその文字列表現でキャッシュして、アンマーシャリングを回避するかどうかを指定します。

値

- **オン：** 応答オブジェクト キーをその文字列表現でキャッシュします。 オンにすると、このプロパティは引き続き VSE に要求を行います。 エージェントに XML ペイロードが返されると、それをデシリアライズする代わりに、キャッシュでオブジェクトを検索します。 キーは、基本的には応答の XML 表現です。 応答が変更可能な状態でない場合、この設定はパフォーマンスを向上させるために役立ちます。 このプロパティをオンにすると、メモリ使用量が増加する場合があります。
- **オフ：** 応答オブジェクト キーをその文字列表現でキャッシュしません。

JIT の有効化

VSE JIT は、モデル/イメージ ロジックがエージェントで部分的に（または完全に）キャッシュされることを示します。

JIT しきい値

VSE JIT を発生させるメソッド呼び出しの数を定義します。

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 (P. 68)」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） (P. 49)」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 (P. 50)」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 (P. 409)」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」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 (P. 66)」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 (P. 55)」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 (P. 55)」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。