DevTest Solutions

インストール

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: インストール前の注意事項

このセクションには、以下のトピックが含まれています。

システム要件

このセクションでは、さまざまな DevTest コンポーネントの要件を示します。 これらの一般的な要件は、プロジェクトの範囲に応じて変わる可能性があります。

オペレーティング システム要件

以下のオペレーティング システムがサポートされています。

- Microsoft Windows：
  - Windows Server 2012
  - Windows 7
  - Windows 8（最新のサービス パックとすべての重大な更新が適用されている）
- Linux および UNIX
  - Fedora 19
  - Red Hat Enterprise Linux 6.3
  - SUSE Linux 10 SP2、11.x
  - Ubuntu 11.04、12.04、13.x
  - Oracle Solaris 10、11
  - IBM AIX 6.1、7.0
- Mac OS X 10.9、10.10

特に DevTest サーバ には 64 ビットのオペレーティング システムおよび JRE をお勧めします。

DevTest サーバ JVM システム要件

2 GB を超えるヒープ サイズを必要とする場合、DevTest サーバ には 64 ビット Java 7 をお勧めします。

Java JDK

DevTest Solutions を構成する製品は、Java アプリケーションです。 Oracle JRE は、汎用 UNIX インストーラ以外の各オペレーティング システム固有インストーラ（JDK の **tools.jar** を含む 1.7.0_17 JRE）に含まれています。ユーザが独自の JVM を提供する (P. 13)場合、IBM JRE Version 7 Release 1 もサポートされます。

DevTest ワークステーション、DevTest サーバ、および VSE でサポートされる Java の最小バージョンは、Java 1.7 update 6 です。

この要件は DevTest 側のみの要件です。 Java 1.7 仮想マシン（VM）で実行される DevTest は、新旧の JRE を実行するアプリケーション サーバ上のアプリケーションをテストするために使用できます。

以下の表に、さまざまな JDK に対して提供されるサポートのリストを示します。

	DevTest ワークステーション	DevTest サーバ - コーディネータ、シミュレータ、レジストリ	DevTest サーバ - VSE	DevTest Java エージェント（CAI）
JDK 1.5	サポートなし	サポートなし	サポートなし	サポート対象
JDK 1.6	サポートなし	サポートなし	サポートなし	サポート対象
JDK 1.7	ランタイムに必要	ランタイムに必要	ランタイムに必要	サポート対象

注: DevTest は、IBM JRE Version 8、Oracle JRE 1.8、および OpenJDK をサポートしていません。

Kerberos 認証は IBM JRE ではサポートされていません。

DevTest ワークステーション は、Java の異なるバージョンで Web サービスおよびメッセージング バックボーンをテストできません。

以下に、DevTest Java エージェント（CAI）の JDK サポートに関する一般的なガイダンスを示します。

- CAI は、JDK 1.5、JDK 1.6、および JDK 1.7 ベースのシステムをサポートします。
- Oracle の JDK のみが完全にサポートされています。IBM JDK は、限定的にのみサポートされます。

DevTest によるフォルダのロック

LISA_HOME ディレクトリの以下のフォルダには、読み取り/書き込み権限が必要です。

- locks

一般的に、LISA_HOME ディレクトリのその他のフォルダは読み取り専用権限で制限できます。

lisatmp フォルダ

ユーザ ホーム ディレクトリ（UNIX、Linux、および OS X）または Documents and Settings （Windows）内の以下のフォルダは、読み取り/書き込み権限を必要とします。

- lisatmp_x.x （存在する場合）

DevTest CICS エージェント

DevTest CICS エージェントは、CICS バージョン 3.2、4.1、および 4.2 をサポートします。

独自の JVM の提供

汎用 UNIX インストーラには JRE が含まれていません。 汎用 UNIX インストーラを使用する場合、独自の JVM を提供する*必要があります*。

必要に応じて、この手順を別のプラットフォーム用のインストーラで使用して、含まれている JRE を無効にできます。 「Java 環境での DevTest ワークステーション の使用 (P. 80)」を参照してください。

注: DevTest は、IBM JRE Version 7 Release 1 をサポートしています。DevTest は、IBM JRE Version 8、Oracle JRE 1.8、および OpenJDK をサポートしていません。

次の手順に従ってください:

1. ご使用のプラットフォーム用の Java Development Kit（JDK）7 パッケージを Oracle の Web サイトからダウンロードします。
2. Java Cryptography Extension （JCE） Unlimited Strength Jurisdiction Policy Files 7 を Oracle の Web サイトからダウンロードします。
3. DevTest をインストールするコンピュータに、お使いのオペレーティング システム用の JDK 7 をインストールします。 **java** ディレクトリがない場合は作成します（mkdir java）。

   たとえば、JDK 7 を ¥usr¥java にインストールします。これにより、¥usr¥java¥jdk1.7.0_67 ディレクトリ（JDK_HOME）が作成されます。
4. jdk-7u67-*platform*. tar.gz ファイルをコピーした後に、次のコマンドを入力します。
   ```
   tar zxvf jdk-7u67-platform-x64.tar.gz
   ```
5. 環境変数を設定します。
   - **JDK_HOME** 環境変数を、JDK 7 をインストールしたディレクトリに設定します。
   - **JAVA_HOME** 環境変数を **JDK_HOME** ディレクトリを指すように設定します。

   以下に例を示します。

   ```
   cd ¥usr¥java¥jdk1.7.0_67
   pwd
   export JAVA_HOME=$PWD
   export JDK_HOME=$PWD
   ```

6. UnlimitedJCEPolicyJDK7.zip ファイルから UnlimitedJCEPolicy フォルダを抽出します。このフォルダから **JDK_HOME¥jre¥lib¥security** ディレクトリに以下の JAR ファイルを移動させます。

   - local_policy.jar
   - US_export_policy.jar

   この操作により、既存の JAR ファイルが同じ名前のファイルに置き換えられます。

7. **JDK_HOME¥lib** ディレクトリから **JDK_HOME¥jre¥lib¥ext** ディレクトリに **tools.jar** ファイルをコピーします。

   ```
   cd $JDK_HOME¥jre¥lib¥ext
   cp $JDK_HOME¥lib¥tools.jar
   ```

8. 含まれている JRE を無効にする場合は、DevTest をインストールした後に、以下の手順を実行します。

   a. **LISA_JAVA_HOME** 環境変数を設定します。 以下に例を示します。

      ```
      cd ¥usr¥java¥jdk1.7.0_67
      pwd
      export LISA_JAVA_HOME=$PWD
      ```

   b. 「Java 環境での DevTest ワークステーションの使用 (P. 80)」の説明に従って、DevTest インストール ディレクトリにある **jre** ディレクトリの名前を「jre_default」に変更します。

DevTest サーバ システム要件

CA Application Test、CA Service Virtualization、および CA Continuous Application Insight には、DevTest サーバ のレジストリが必要です。

DevTest サーバ の最小要件は以下のとおりです。

- **CPU：** 2 GHz 以上、4 コア以上
- **RAM：** 4 GB
- **ディスク容量：** 50 GB
- **データベース：** 「データベース システム要件 (P. 18)」を参照してください。データベースは別のシステムに配置でき、少なくとも 200 GB のストレージが必要です。

DevTest サーバ の推奨要件は以下のとおりです。

- **CPU：** 2 GHz 以上、8 コア以上
- **RAM：** 8 GB
- **ディスク容量：** 50 GB
- **データベース：** 「データベース システム要件 (P. 18)」を参照してください。データベースは別のシステムに配置でき、少なくとも 500 GB のストレージが必要です。

負荷およびパフォーマンス テストについては、以下のリソースを推奨します。

- シミュレータあたり 250 の仮想ユーザ
- シミュレータあたり 1 つのプロセッサ コアおよび 2 GB の RAM

4000 の同時仮想ユーザの例：16 個のシミュレータ、16 個のプロセッサ コア、32 GB の RAM （DevTest に対して）

各データ センターに対して

- 1 つのテスト レジストリおよび 1 つのコーディネータ
- 1 つのプロセッサ コア/プロセス = 2 つのプロセッサ コア
- 各 2 GB の RAM = 4 GB （DevTest に対して）

基本的な DevTest サーバ 構成には、1 つのエンタープライズ ダッシュボード、1 つのレジストリ、および 1 つのポータルがあり、これらはすべての製品に対して必要です。 CA Application Test には、1 つのコーディネータ サーバと 1 つのシミュレータ サーバが必要です。 CA Service Virtualization には、1 つの仮想サーバ環境が必要です。 CA Continuous Application Insight には、1 つのブローカが必要です。 どの特定の構成についても、必要なエンタープライズ ダッシュボード サーバは 1 つのみです。

注: ここに示されている要件は、ガイドラインとしての使用を意図しています。 負荷の大きい環境の場合、ニーズに合わせて負荷生成環境の開発を支援する専門サービスに問い合わせることをお勧めします。

CA Application Test 用の DevTest ワークステーション のシステム要件

DevTest ワークステーション の最小要件は以下のとおりです。

- **CPU：** 2 GHz 以上、2 コア以上
- **RAM：** 4 GB
- **ディスク空き容量：** 2 GB

CA Service Virtualization のシステム要件

CA Service Virtualization のコンポーネントである VSE は仮想化環境の管理に必要です。VSE はサーバ レベルのサービスで、コーディネータおよびシミュレータが接続されてい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 VSE の実行に必須ではありません。

以下の要件は、基準としての要件です。

- **CPU：** 2 GHz 以上、2 コア以上
- **RAM：** DevTest ワークステーション、DevTest サーバ、CAI の RAM 要件に加えて、VSE 用の 2 GB
- **ディスク空き容量：** 50 GB
- **データベース：** 「データベース システム要件 (P. 18)」を参照してください。 データベースは別のシステムに配置でき、少なくとも 10 GB のストレージが必要です。

CAI システム要件

CA Continuous Application Insight の最小要件は以下のとおりです。

- **CPU：** 2 GHz 以上、8 コア以上
- **RAM：** 8 GB
- **ディスク容量：** 50 GB のローカル ディスク ストレージ
- **データベース：** 「データベース システム要件 (P. 18)」を参照してください。データベースは別のシステムに配置でき、少なくとも 500 GB のストレージが必要です。

通信要件

DevTest Solutions によるネットワーク転送の送受信をファイアウォールが許可することを確認します。 DevTest Solutions が提供する機能には、ネットワーク リソースへのアクセスが必要であり、ファイアウォールによってブロックされる場合は正しく動作しません。 DevTest Solutions アプリケーションを認可します。

注: 安全な通信を実装するには、「*管理*」の「通信を保護するための SSL の使用」および「DevTest コンソールとの HTTPS 通信の使用」を参照してください。

DevTest ポートとの通信は、関連するファイアウォールで開かれている必要があります。 「*管理*」で以下のトピックを参照してください。

- DevTest サーバ のデフォルト ポート番号
- DevTest ワークステーション のデフォルト ポート番号
- デモ サーバのデフォルト ポート番号

データベース システム要件

以下のコンポーネントがデータベースに情報を格納します。

- **DevTest サーバ：** データベースはレポート結果に使用されます。レポート結果は、必要に応じてその他の形式にエクスポートできます。 データベースはアクセス制御（ACL）にも使用されます。
- **VSE：** データベースは使用数およびレガシー仮想サービス イメージに使用されます。
- **CAI：** データベースは、要求/応答データ、SQL ステートメント、およびアプリケーション ログなどのパスに使用されます。データベースはチケットにも使用されます。
- **エンタープライズ ダッシュボード：** データベースは、DevTest Solutions 使用率監査レポート データ、その他のレジストリ情報、履歴イベント ログ、およびメトリックに使用されます。

  重要: エンタープライズ ダッシュボードには、それ専用の一意の大規模なデータベースが必要です。データベースは別のシステムに配置でき、少なくとも 50 GB のストレージが必要です。IBM DB2 は、エンタープライズ ダッシュボードのデータベースとしてはサポートされません。

デフォルトでは、これらのコンポーネントは、DevTest に含まれている Apache Derby データベースを使用します。このデータベースは、負荷およびパフォーマンス テストを必要としない小規模な展開にのみ適しており、サポートされません。 その他のすべてのシナリオについては、外部データベースを使用するように DevTest を設定します。

外部データベースを使用する場合に適切なパフォーマンスを確保するために、、データベース サーバおよび DevTest サーバは、ネットワーク帯域幅が高く、遅延が低い必要があります。

DevTest は、分散設定で実行される場合、管理の行き届いたよくメンテナンスされたエンタープライズ クラス データベースに高帯域幅、低遅延で接続しているサーバ コンポーネントに強く依存します。

すべての DevTest サーバ コンポーネントは、そのデータベースと直接通信して、それらのアクションを記録します。このデータ フローに制限があると、悪影響が生まれます。

DevTest の正常な機能を確保するには、どの DevTest サーバ コンポーネントも、データベース ホストに対して 20 ミリ秒を超えるラウンド トリップ時間（RTT）を持たないようにする必要があります。 ネットワーク遅延がこの 20 ミリ秒の値を超えると、パフォーマンスに関する問題が発生する可能性があります。

重要: 新規インストールの場合は、必ず、クリーンなデータベース スキーマを使用してください。 同じ DevTest バージョンからのデータは、インストールの前に、クリーンなスキーマへリストアできます。 他のバージョンからのデータを使用しないでください。

以下の外部データベースがサポートされています。

- **IBM DB2 10.1 または 10.5**（エンタープライズ ダッシュボードではサポートされない）- データベースのコード ページは 1208 である必要があります。 また、ページ サイズは少なくとも 8 KB である必要があります。
- **MySQL 5.5 または 5.6** - MySQL データベースは、UTF-8 をサポートする照合および文字セットを提供する必要があります。2 バイト文字は、ACL およびレポート テーブルに格納されます。 データベースのデフォルト コード ページは UTF-8 である必要があります。 データベースを UTF-8 として定義することのみでは十分ではありません。
- **Oracle 11g リリース 2 または 12c** - 文字セットは Unicode セットである必要があります。
- **Microsoft SQL Server 2008 R2 または 2012**

レジストリが初めて起動されると、スキーマは外部データベースに自動的に作成されます。 スキーマが作成される前に、DevTest ユーザに DBA 権限があることを確認します。 スキーマが作成された後、ユーザから DBA 権限を削除できます。

セキュリティ ポリシーがこの方法を許可しない場合、データベース管理者はスキーマを手動で作成できます。 **LISA_HOME¥database** ディレクトリ内の DDL ファイルには、レポート テーブルおよびインデックスを作成するための SQL ステートメントと、CA Continuous Application Insight が使用するエージェント データベース スキーマを作成するための SQL ステートメントが含まれます。 この情報をデータベース管理者に提供します。

注: 外部データベースの設定の詳細については、「*管理*」を参照してください。 エージェント データベース スキーマ用の SQL ステートメントの取得の詳細については、「エージェント」を参照してください。

重要: レジストリとエンタープライズ ダッシュボードには、それぞれ一意のスキーマが必要です。 複数のレジストリで同じデータベース スキーマを指定しないでください。

負荷およびパフォーマンス テストについては、外部データベースを調整して DevTest が必要とする量のデータ ストレージをサポートできることを確認します。

レジストリ、コーディネータ、シミュレータ、およびすべての仮想サービス環境には、高パフォーマンス データベース アクセスが必要です。 パフォーマンス データは、これらのコンポーネントによってデータベースに直接記録されます。 このデータベースは同じデータ センター内に配置することをお勧めします。 仮想マシン内でホストされるデータベースを一般提供に使用することは推奨されません。

サポートされるブラウザ

DevTest には、以下の Web ベースのポータル、コンソール、およびダッシュボードが含まれます。

- エンタープライズ ダッシュボード（http://*hostname*:1506/）
- DevTest ポータル（http://*hostname*:1507/devtest）
- DevTest コンソール（http://*hostname*:1505/）
  - レポート ポータル
  - 継続的検証サービス
  - サーバ コンソール
  - VSEasy
  - CAI コンソール（DevTest ポータル にリダイレクト）
- デモ サーバ（http://*hostname*:8080/lisabank/）

DevTest ポータルには HTML 5 が必要です。 そのため、新しい Web ベース UI を使用するには、最新のインターネット ブラウザ テクノロジが必要です。 以下のインターネット ブラウザ バージョンは HTML 5 をサポートしています。

- Google Chrome 36
- Mozilla Firefox 30
- Apple Safari 7.0
- Microsoft Internet Explorer 11

注: Selenium 統合テストのレコーディングをサポートするブラウザは、Mozilla Firefox に制限されています。これらのテストを DevTest にインポートした後に、テスト ケースの実行が、Google Chrome、Mozilla Firefox 24、または Internet Explorer 10 および 11 で確認されています。

注: VSEasy をサポートする Internet Explorer ブラウザは、バージョン 10 に制限されています。 リストに示されているその他のすべてのブラウザがサポートされています。

DevTest システムの計画

正常に DevTest Solutions をインストールするため、インストール前に以下の決定ポイントを考慮します。

- どのサポートされているオペレーティング システムを使用するよう計画しますか。
- 共有インストールまたはローカル インストールを実行していますか。
- インストールの対象はワークステーションまたはサーバでの使用ですか。
- どのエンタープライズ データベースを使用するよう計画しますか。

  インストールが常設か、または通常使用可能な環境で使用する予定の場合、付属の Derby データベースはサポートされません。
- デモ サーバのインストールを計画しますか。

  デモ サーバは、製品インストーラとは別に提供されます。

インストールを開始する前に、ライセンス ファイル (devtestlic.xml) がターゲット システムに存在し、利用可能である必要があります。

ターゲット システム内のさまざまなサービスおよびプロセスの理想的なレイアウトに関する追加の考慮事項

- エンタープライズ ダッシュボードは、レジストリによってアクセス可能とする必要があります。しかし、レジストリに対してローカルである必要はありません。エンタープライズ ダッシュボードには、それ自身のデータベース インスタンスが必要です。
- レジストリとコーディネータは、可能な限り同じマシン上に存在する必要があります。
- 実行可能な場合、シミュレータはコーディネータと同じネットワーク セグメント上に存在する必要があります。
- レジストリ、コーディネータ、シミュレータ、およびすべての仮想サービス環境には、高パフォーマンス データベース アクセスが必要です。パフォーマンス データは、これらのコンポーネントによってデータベースに直接記録されます。このデータベースは同じデータ センター内に配置することをお勧めします。仮想マシン内でホストされるデータベースを一般提供に使用することは推奨されません。

データベースの要件の詳細については、「データベース システム要件 (P. 18)」を参照してください。

DevTest システムの計画を始める前に、インストールするリリースのリリース ノートを確認します。 システムをマイグレートしている場合は、「*マイグレーション ガイド*」を確認します。

DevTest システムの計画の一部は、DevTest サーバ をインストールする数と、DevTest サーバ が含まれる各ホスト コンピュータ上で実行されるプロセスまたはサービスを決定することです。

DevTest Solutions インストーラをダウンロードすると、3 つの製品のうち 1 つを選択できます。これらの製品はすべて同じインストーラを指定します。「DevTest コンポーネント (P. 24)」では、これら 3 つの製品をグラフィカルに示し、製品固有の UI およびプロセスと、すべての製品に共通の UI およびプロセスについて説明します。「DevTest サーバ コンポーネントについて (P. 26)」では、各プロセスの目的と、詳細を確認できる場所について説明します。

「分散システムでの関係の例 (P. 28)」では、分散システムの各ホスト上で実行できるプロセスに関する大概念について説明します。 「インストール後 (P. 69)」では、分散システムの調整に役立つトピック、具体的には以下について説明します。

- 別のシステム上にあるコンポーネントの実行 (P. 77)
- シミュレータ インスタンスの算出 (P. 78)
- 負荷およびパフォーマンス サーバのサイジング (P. 79)

計画の別の側面は、プロジェクト ディレクトリ構造 (P. 81)の命名規則の設計です。

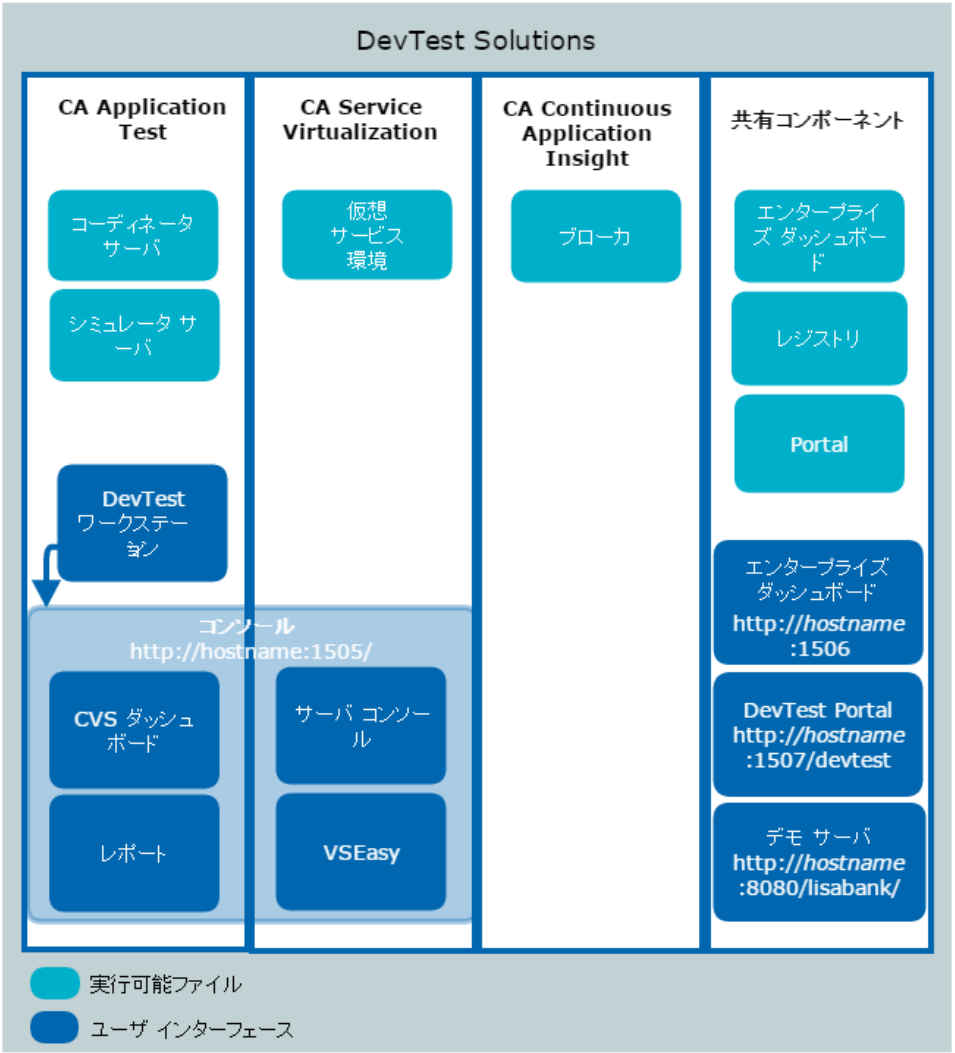
DevTest に付属する JRE を使用するか、または独自のものをインストールすることができます。 詳細については、「Java 環境での DevTest ワークステーション の使用 (P. 80)」を参照してください。

DevTest コンポーネント

DevTest サーバ（セットアップ ウィザードでは「サーバ」）をインストールすると、DevTest Solutions インストーラによって以下の製品がインストールされます。

- CA Application Test
- CA Service Virtualization
- CA Continuous Application Insight

これらの 3 つの製品にはそれぞれ独自の実行ファイル（および対応するサービス）があります。 ただし、それらは、同じレジストリ実行ファイルを必要とするため、独立した製品ではありません。 以前には、各製品に独自の UI がありました。 現在は、DevTest ポータルが、すべての製品の共有ユーザ インターフェースになっています （一部の機能には DevTest ワークステーション または DevTest コンソールからのみアクセスできます）。


DevTest Solutions
CA Application Test
CA Service Virtualization
CA Continuous Application Insight
共有コンポーネント
コーディネータ サーバ
シミュレータ サーバ
仮想サービス環境
ブローカ
エンタープライズ ダッシュボード
レジストリ
Portal
DevTest ワークステーション
コンソール
http://hostname:1505/
CVS ダッシュボード
サーバ コンソール
レポート
VSEasy
エンタープライズ ダッシュボード
http://hostname:1506
DevTest Portal
http://hostname:1507/devtest
デモ サーバ
http://hostname:8080/lisabank/
実行可能ファイル
ユーザ インターフェース

「DevTest サーバ コンポーネントについて (P. 26)」では、各実行ファイル コンポーネントについて説明しており、詳細情報の入手先も示しています。

「サーバ コンポーネントの起動 (P. 104)」には、実行ファイルを起動する順序が示されています。

「ユーザ インターフェースへのログイン (P. 112)」には、各 UI にアクセスする手順が含まれています。

DevTest Solutions のセットアップおよび保守の詳細については、「*管理*」を参照してください。

製品固有の詳細については、以下を参照してください。

- *CA Application Test の使用*
- *CA Service Virtualization の使用*
- *CA Continuous Application Insight の使用*

DevTest サーバ コンポーネントについて

一連のプロセスまたはサービスを起動することにより、DevTest サーバ を起動します。 最初に起動する 3 つのコンポーネント（エンタープライズ ダッシュボード、レジストリ、およびポータル）は、DevTest Solutions をサポートしています。 その他のコンポーネントは製品固有です。 スタートアップの詳細については、「サーバ コンポーネントの起動 (P. 104)」を参照してください。

name	コンポーネントを必要とする製品	コンポーネントを必要とする操作
エンタープライズ ダッシュボード	レジストリ（すべての DevTest Solutions 製品）	製品ライセンスで DevTest Solutions を起動します。 エンタープライズ アクティビティをモニタします。 使用状況レポートを生成します。

name	コンポーネントを必要とする製品	コンポーネントを必要とする操作
Registry	すべての DevTest Solutions 製品	DevTest サーバ および DevTest ワークステーション コンポーネントを登録します。 レジストリは、すべてのプロセスのセントラル ハブまたはエンジンです。 使用状況レポート用の使用状況データを収集します。
Portal	すべての DevTest Solutions 製品	DevTest ポータル UI を開始します。
ブローカ	CA Continuous Application Insight （CAI）	Java エージェントおよび CA Continuous Application Insight コンソールを調整します。
コーディネータ	CA Application Test	シミュレータ上で実行されたテストを調整します。
シミュレータ	CA Application Test	テストを実行します。
仮想サービス環境（VSE）	CA Service Virtualization	仮想サービスを作成して展開します。

参照先のトピックは、DevTest サーバ をサポートする各プロセスに関する詳細な情報を提供します。

プロセス	リファレンス
エンタープライズ ダッシュボード	「*管理*」の「エンタープライズ ダッシュボード メイン ウィンドウ」
Registry	「*CA Application Test の使用*」の「レジストリ」
Portal	「*はじめに*」の「DevTest ポータルの使用」
コーディネータ	「*CA Application Test の使用*」の「コーディネータ サーバ」
シミュレータ	「*CA Application Test の使用*」の「シミュレータ サーバ」
ワークステーション	「*CA Application Test の使用*」の「DevTest ワークステーション」
仮想サービス環境	「*CA Service Virtualization の使用*」の「仮想化の準備」
ブローカ	「エージェント」の「ブローカの起動」

DevTest プロセスの関係

レジストリは、すべての DevTest システムの中心です。 通常、インストーラは、複数のコンピュータ上で、異なる選択内容によって実行されます。以下の例について考えてみます。

- コンピュータ 1 : エンタープライズ ダッシュボードおよびサーバ コンポーネント（組み込み型のワークステーションは使用されない可能性がある）。
- コンピュータ 2 : サーバ（ここにあるレジストリ コンポーネントがコンピュータ 1 上のエンタープライズ ダッシュボードにリンクされている）。 通常、分散システムには 1 つのレジストリのみが必要です。この 2 つ目のレジストリは、ここでは単に複数のレジストリがサポートされることを示すために追加されています。
- コンピュータ 3 : ワークステーションおよびデモ サーバ（コンピュータ 1 上のレジストリへのリンクを持つ）。 このコンピュータは、ユーザ マシンを表します。
- コンピュータ 4（これもユーザ マシン、ポータル アクセスのみを必要とする CA Continuous Application Insight ユーザを表すことができる）。
- また、DevTest サーバを、1 つの製品固有サービスのみ（シミュレータ実行ファイルまたはサービスなど）を実行するコンピュータにインストールできます。

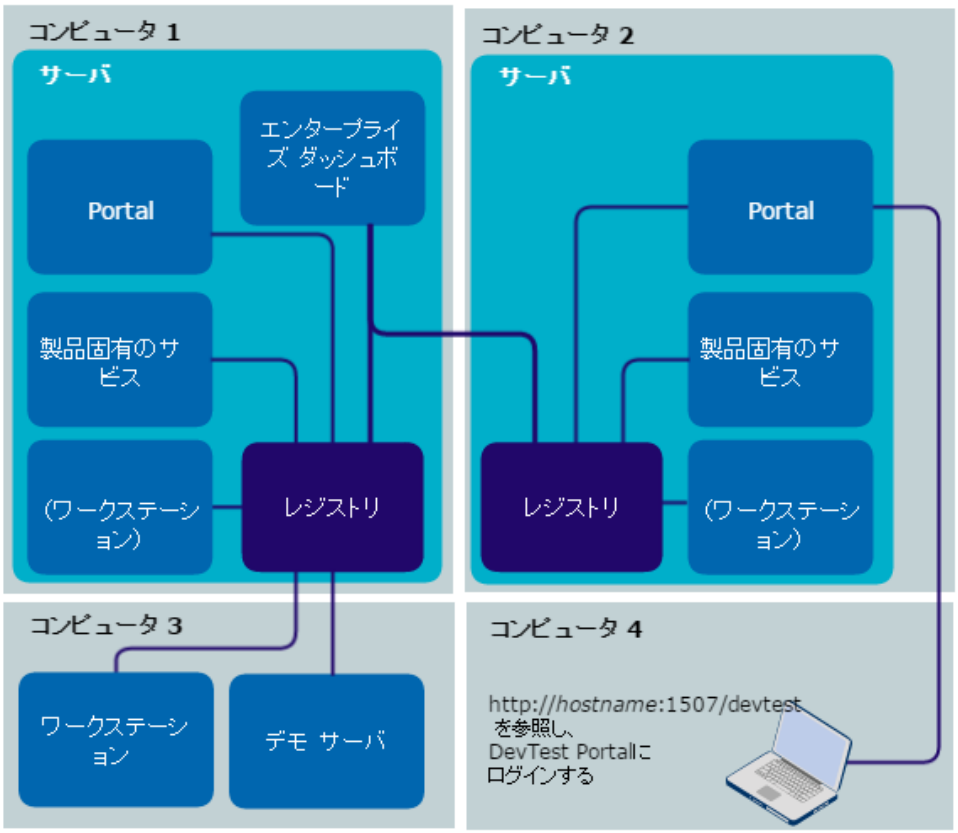
この例のシステムは、任意の DevTest システムと以下の特性を共有します。

- 各 DevTest Solutions システムには、エンタープライズ ダッシュボードが 1 つのみあります。

  注: アクセス可能なネットワークごとに 1 つのエンタープライズ ダッシュボードが必要です。クローズド ネットワーク（相互にアクセスできないネットワーク）を使用している場合、レジストリが実行されるネットワークごとにエンタープライズ ダッシュボードが必要です。

- 各エンタープライズ ダッシュボードは、1 つ以上のレジストリに接続されます。 通常、1 つのレジストリで十分です。
- 各 DevTest サーバ インストールでは、1 つのレジストリ（すべてのユーザ アクティビティを監査するコンポーネント）がインストールされます。
- 各 DevTest サーバ インストールでは、1 つのローカル ワークステーションがインストールされます。 この組み込み型のワークステーションは、スタンドアロン インストールで使用されます。 DevTest サーバが分散環境にインストールされる場合、ローカル ワークステーションが使用されない可能性があります。
- 分散システムの各 DevTest サーバ レジストリは、1 つ以上のリモート ワークステーションに接続できます。
- Web ブラウザは、ポート 1507 上の DevTest サーバ（URL の最後がすべて小文字の devtest）を参照することにより、DevTest ポータルにアクセスできます。 このポータルには任意の Web ブラウザからアクセスでき、他の DevTest Solutions ソフトウェアをそのコンピュータにインストールする必要はありません。

以下の図は、分散システム内のエンタープライズ ダッシュボード、レジストリ、ワークステーション、およびデモ サーバの関係を強調表示しています。 また、ローカル DevTest コンポーネントを持たないユーザが DevTest ポータルにアクセスできることを示しています。



DevTest Solutions アーキテクチャ

詳細:

DevTest サーバ コンポーネント (P. 38)
CA Application Test および CA Service Virtualization 用の DevTest サーバ のデータ フロー (P. 41)

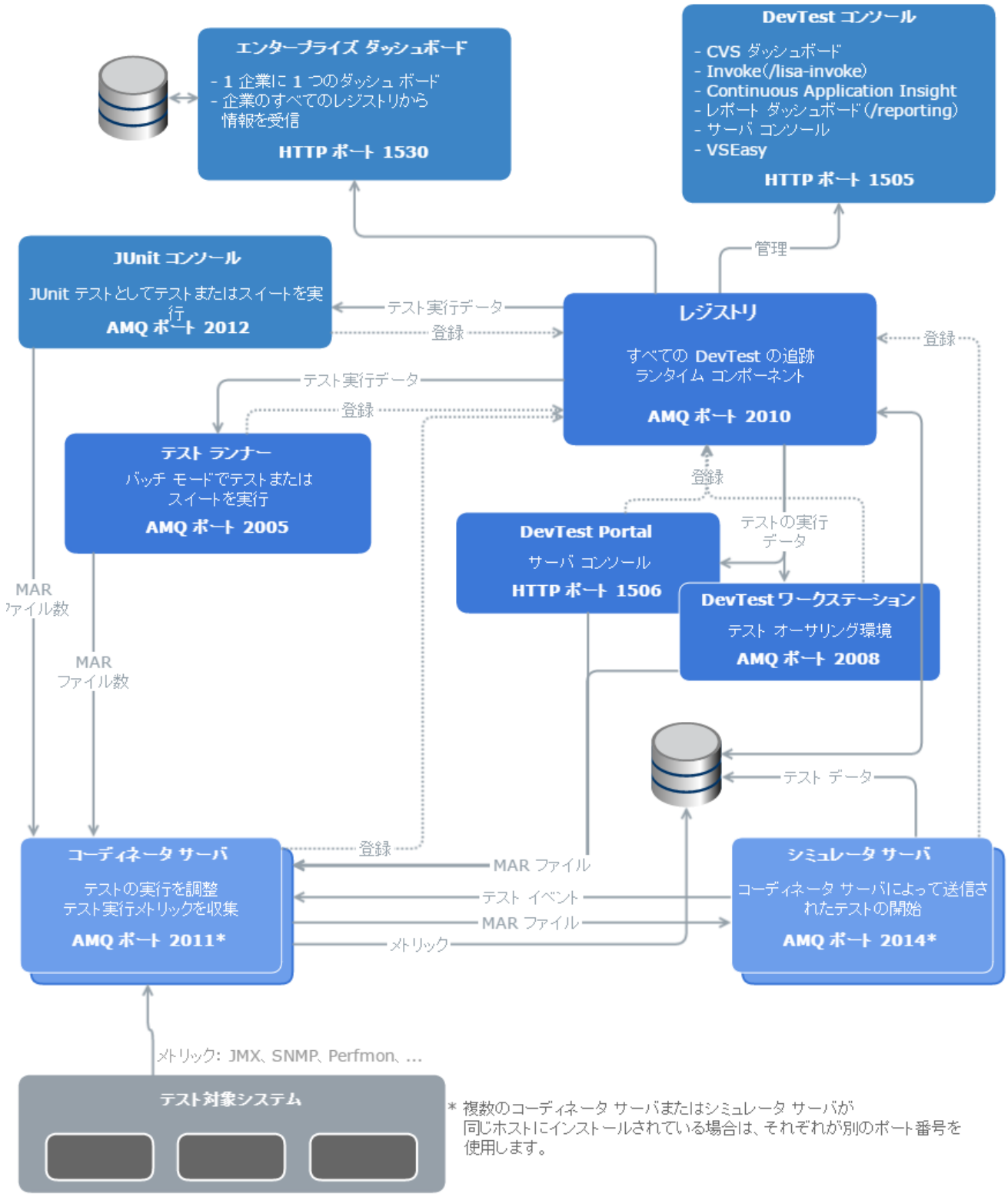
CA Application Test アーキテクチャ

CA Application Test は、以下の DevTest サーバ コンポーネントを使用します。

- Registry
- DevTest ワークステーション
- DevTest ポータル
- コーディネータ サーバ
- シミュレータ サーバ

DevTest ポータルおよび DevTest ワークステーション はテストを作成およびモニタするために使用されますが、テスト ケースは DevTest サーバ 環境で実行されます。 コーディネータ サーバおよびシミュレータ サーバは DevTest ワークステーション に組み込まれています。

以下の図は、CA Application Test アーキテクチャを示しています。


エンタープライズ ダッシュボード
- 1 企業に 1 つのダッシュ ボード
- 企業のすべてのレジストリから
情報を受信
HTTP ポート 1530
DevTest コンソール
- CVS ダッシュボード
- Invoke(/lisa-invoke)
- Continuous Application Insight
- レポート ダッシュボード(/reporting)
- サーバ コンソール
- VSEasy
HTTP ポート 1505
JUnit コンソール
JUnit テストとしてテストまたはスイートを実行
AMQ ポート 2012
管理
テスト実行データ
登録
レジストリ
すべての DevTest の追跡
ランタイム コンポーネント
AMQ ポート 2010
テスト ランナー
バッチ モードでテストまたは
スイートを実行
AMQ ポート 2005
DevTest Portal
サーバ コンソール
HTTP ポート 1506
テストの実行
データ
DevTest ワークステーション
テスト オーサリング環境
AMQ ポート 2008
MAR
ファイル数
テスト データ
コーディネータ サーバ
テストの実行を調整
テスト実行メトリックを収集
AMQ ポート 2011*
MAR ファイル
テスト イベント
メトリック
シミュレータ サーバ
コーディネータ サーバによって送信さ
れたテストの開始
AMQ ポート 2014*
メトリック: JMX、SNMP、Perfmon、...
テスト対象システム
* 複数のコーディネータ サーバまたはシミュレータ サーバが
同じホストにインストールされている場合は、それぞれが別のポート番号を
使用します。

テストはすべて、コーディネータ サーバのコーディネータの管理下でシミュレータ サーバによって生成された仮想ユーザ（シミュレータ）によって実行されます。

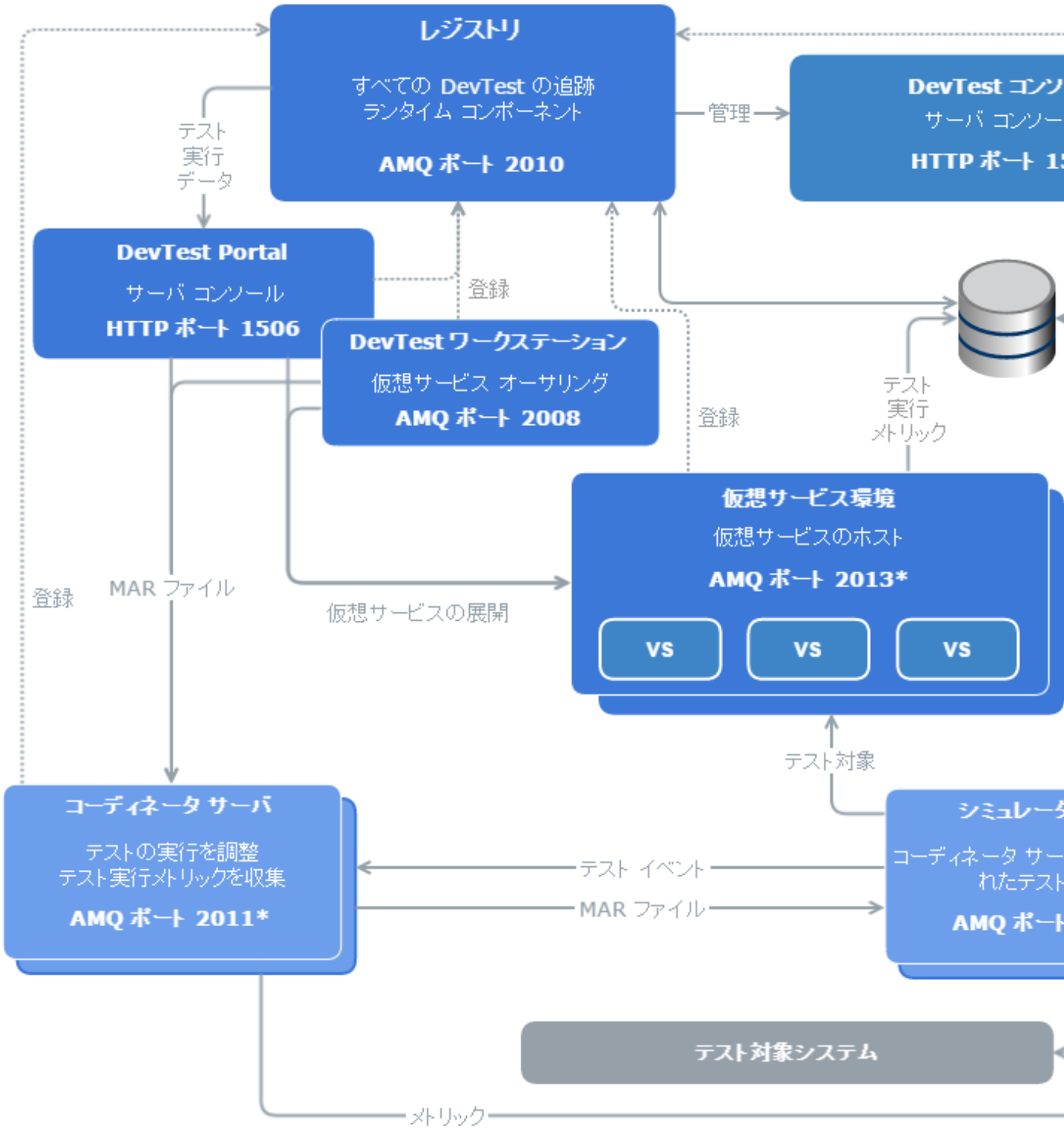
各シミュレータはテスト中のシステムに接続し、アクションを呼び出します。 仮想ユーザが並列モードで実行されていると、負荷テストが行われます。

DevTest ワークステーション の埋め込みインスタンスは、DevTest サーバが実行されているのと同じコンピュータ上で実行されます。 また、スタンドアロン DevTest ワークステーション を設定して、大規模な分散環境内の個別のコンピュータ上で実行することができます。 テスト要件には、使用されるサーバ アーキテクチャが指定されています。 DevTest Solutions は、さまざまなハードウェア/オペレーティング システム上に各種コンポーネントを分散させることにより、大規模なテスト環境に合わせて拡張できます。

DevTest 製品には CA Application Test、CA Service Virtualization、および CA Continuous Application Insight が含まれます。それぞれがセントラル DevTest レジストリに接続されます。 CA Application Test ユーザは、組み込み DevTest ワークステーション と共に、オプションのコーディネータ サーバおよびシミュレータ サーバと、必要に応じて、リモートでインストールされたすべての DevTest ワークステーション を使用します。 すべての DevTest 製品のコンポーネントは、セントラル レジストリに接続されます。

CA Service Virtualization アーキテクチャ

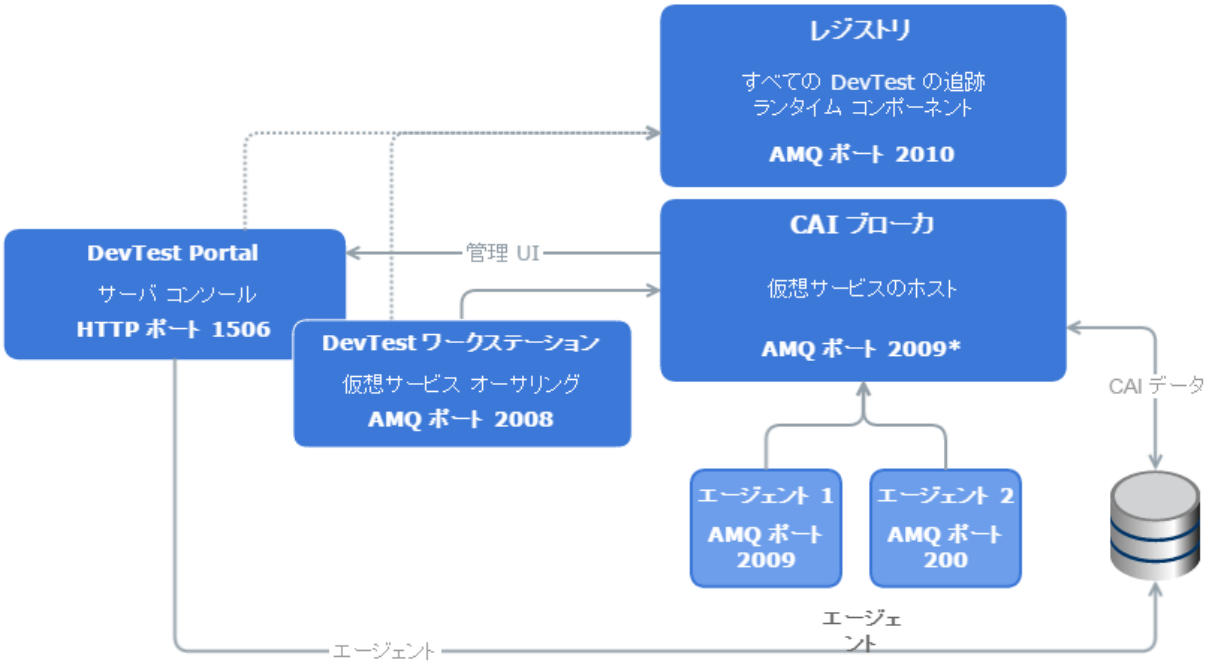
以下の図は、CA Service Virtualization アーキテクチャを示しています。



＊同じマシン上にある複数のコーディネータ、シミュレータ、および VSE は、別々のポート番号を使用します。

CAI アーキテクチャ

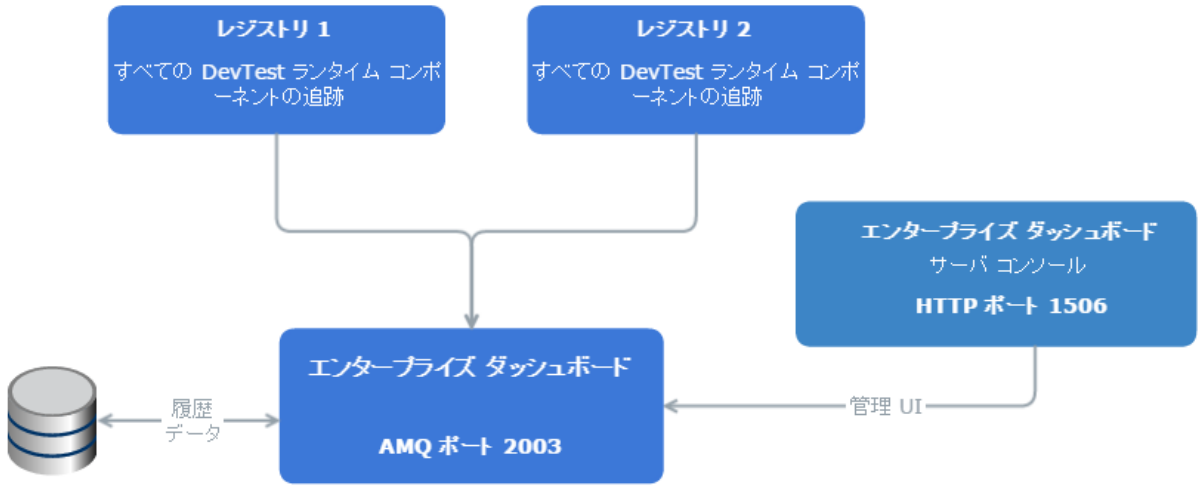
以下の図は、CAI アーキテクチャを示しています。



＊同じマシン上にある複数の VSE は、別々のポート番号を使用します。

エンタープライズ ダッシュボード アーキテクチャ

以下の図は、エンタープライズ ダッシュボード アーキテクチャを示しています。

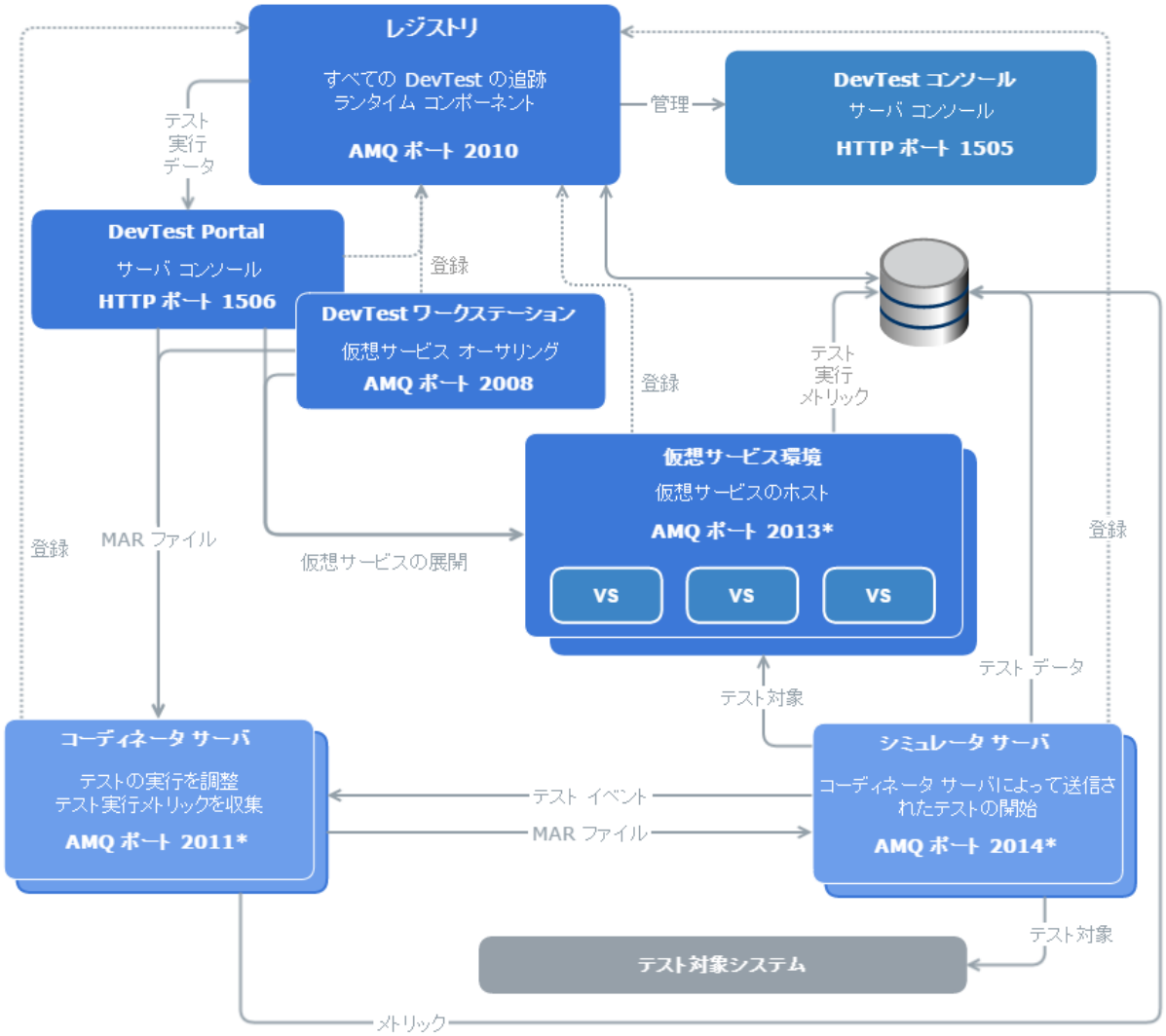


DevTest サーバ コンポーネント

CA Application Test では、テストは DevTest サーバ 環境で実行されます。DevTest ワークステーション は DevTest サーバ に接続して、DevTest ワークステーション で開発されたテストを展開およびモニタします。

CA Application Test および CA Service Virtualization は、以下の DevTest サーバ コンポーネントを使用します。

- **DevTest ワークステーション：** テスト ケース アセットおよび仮想サービス モデルが作成および編集される統合開発環境 (IDE)。テスト ケースおよびモデルをワークステーションでローカルに実行したり、ステージングしてリモートで実行したりできます。 DevTest ワークステーション は、テストおよび仮想モデル アセットを作成するユーザのデスクトップ コンピュータにインストールする必要があります。ワークステーションはいくつでもレジストリに接続でき、サーバ環境を共有できます。詳細については、「*CA Application Test の使用*」の「DevTest ワークステーション」を参照してください。
- **レジストリ：** 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用の一元的な場所。 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。
- **DevTest コンソール**： このコンソールには、サーバ コンソール (VSE など) 、CVS ダッシュボード、レポート、および VSEasy へのリンクが含まれています。
- **コーディネータ サーバ：** コーディネータは MAR ファイルとしてテスト ラン情報を受信し、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。
- **シミュレータ サーバ：** 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。
- **VSE：** 仮想サービス モデルを展開および実行するために使用します。詳細については、「*CA Service Virtualization の使用*」を参照してください。



* 同じマシン上にある複数のコーディネータ、シミュレータ、および VSE は、別々のポート番号を使用します。

レジストリ、コーディネータ サーバ、シミュレータ サーバ、および VSE は、個別の仮想マシンで実行される「ヘッドレス」Java アプリケーションです。

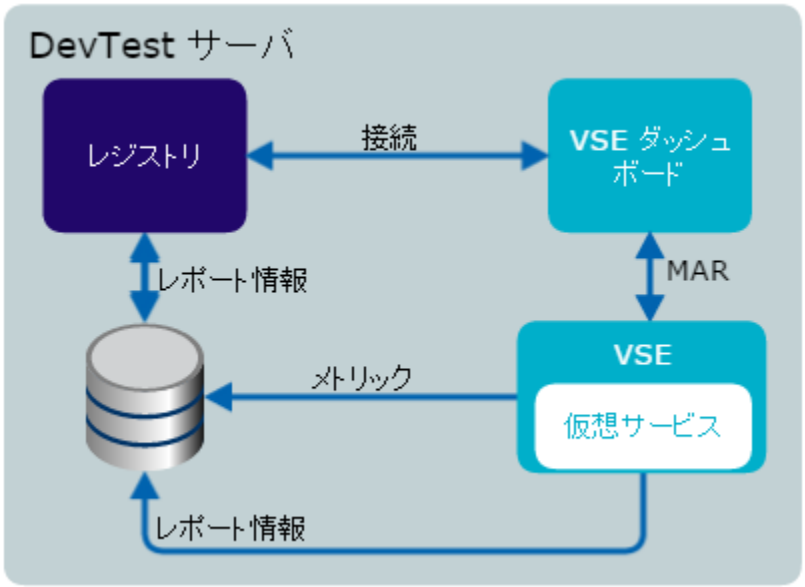
CA Application Test および CA Service Virtualization 用の DevTest サーバ の最小構成には、これらのコンポーネントをそれぞれ 1 つ以上含める必要があります。 特定のテスト環境に必要とされるのと同数のインスタンスを各タイプに設定できます。

通常、CA Application Test 用の DevTest サーバ 構成には 1 つのレジストリ、1 つのコーディネータ サーバ、および複数のシミュレータ サーバがあります。

VSE はサーバ レベルのサービスです。サービスは、コーディネータおよびシミュレータが接続されてい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 VSE の実行に必須ではありません。

CA Application Test および CA Service Virtualization 用の DevTest サーバ のデータ フロー

以下の図は、レジストリ、DevTest ワークステーション、コーディネータ サーバ、シミュレータ サーバ、およびデータベース間のデータ フローを示しています。



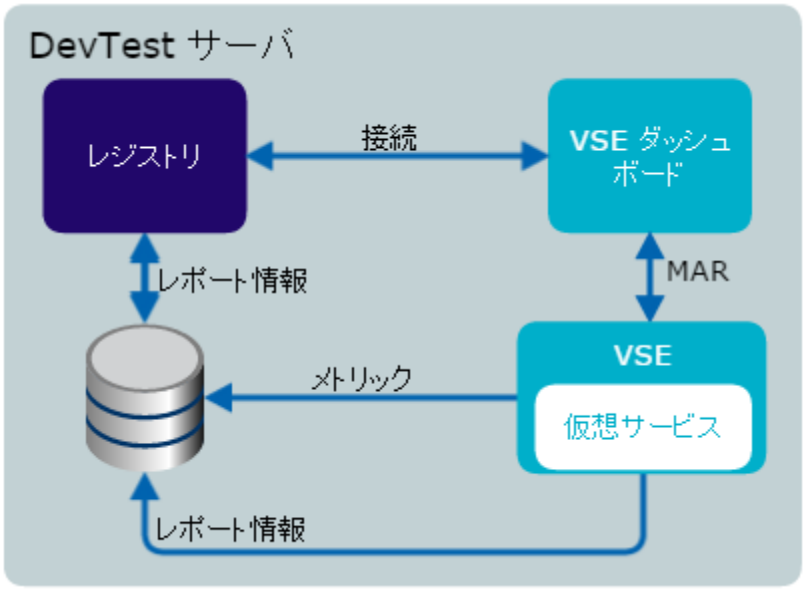
コーディネータ サーバは、1 つ以上のシミュレータ サーバにテスト ケースをモデル アーカイブ（MAR）形式で送信します。

シミュレータ サーバは、テスト中のシステムと対話します。 これらのコンポーネント間で交換されるデータのタイプは大きく異なる可能性があります。データは、簡単な HTTP 要求と HTML 応答、Web サービス コール、データベース コールなどです。

各種コンポーネントはデータベースにメトリックおよびレポート情報を送信します。

レジストリはレポート ポータルをホストします。そのため、データベースと通信してレポート データを取得します。

以下の図は、レジストリ、VSE、VSE ダッシュボード、およびデータベース間のデータ フローを示しています。


DevTest サーバ
レジストリ
接続
VSE ダッシュボード
レポート情報
MAR
メトリック
VSE
仮想サービス
レポート情報

DevTest Solutions インストーラのダウンロード

CA Support Online の Download Center では、DevTest Solutions のこのリリース用の 1 つ以上のプラットフォーム固有インストーラおよびオプションの関連ファイルをダウンロードできます。

まず、プラットフォーム固有インストーラをダウンロードしてください。ダウンロードが完了したら、サンプル付きのデモ サーバをダウンロードするプロセスを繰り返します。 デモ サーバを使用して、サンプルで練習したり、CA Application Test ドキュメントに含まれているチュートリアルで作業したりすることができます。

次の手順に従ってください:

1. support.ca.com にアクセスします。
2. ［Login］をクリックし、CA Support Online ユーザ名およびパスワードを使用してログインします。
3. ［Download Center］の上にマウス ポインタを移動させ、［Download Products］を選択します。

   ［Download Center］ページが、［All Products］オプションが選択された状態で表示されます。
4. 以下のいずれかの DevTest 製品名を入力すると、ドロップダウン リストに関連オプションが表示されます。
   - CA Service Virtualization for Performance – MULTI-PLATFORM
   - CA Service Virtualization Power User – MULTI-PLATFORM
   - CA Service Virtualization Runtime User – MULTI-PLATFORM
   - CA Service Virtualization Virtual User – MULTI-PLATFORM
   - CA Continuous Application Insight Power User – MULTI-PLATFORM
   - CA Continuous Application Insight Runtime User – MULTI-PLATFORM
   - CA Application Test – MULTI-PLATFORM
5. ダウンロードする製品を選択します。 これらの選択肢のそれぞれで、お使いのオペレーティング システム用の DevTest Solutions インストール ウィザードをダウンロードできます。 このインストーラにより、DevTest サーバ （3 つの製品がすべて含まれる）および DevTest ワークステーション をインストールできます。
6. ダウンロードする特定のリリースを選択します。

7. その他のフィールドはデフォルト値のままにしておきます。
8. ［Go］をクリックします。

   製品コンポーネントのリストが表示されます。
9. ダウンロードする各コンポーネントの［Add to Cart］列のチェック ボックスをオンにします。
   - （必須）お使いの各オペレーティング システム用のインストーラ
   - （オプション） DevTest デモ サーバ
10. ウィンドウの上部にある［My Download Cart］をクリックします。

    ［My Download Cart］ページが表示されます。
11. ［Checkout］セクションに電子メール アドレスを入力し、［Checkout］をクリックします。

    ［Download Method］ページが表示されます。
12. ［Download］をクリックし、ファイルを保存します。

第 2 章: DevTest インストールの概要

このインストール ドキュメントでは、すべてを含む単一の DevTest Solutions インストールについて説明します。製品ごとのセクションに分かれてはいません。 CA Application Test、CA Service Virtualization、および CA Continuous Application Insight は、1 つのソリューション セットとして一緒にインストールされます。

このセクションには、以下のトピックが含まれています。

インストール オプション

システム管理者権限のないユーザでも、サーバ（DevTest サーバ）、DevTest ワークステーション、およびデモ サーバをインストールできます。 ただし、サーバ コンポーネントを Windows サービスとしてインストールできるのはシステム管理者のみです。

インストールするコンポーネントのオプション

DevTest Solutions セットアップ ウィザードは、ローカル ホストにインストールされるオプションを制御する 3 つのダイアログ ボックスを提供します。各ダイアログ ボックスから 1 つのオプションのみを選択できます。デフォルト オプションは斜体で表示されます。

- **コンポーネントの選択**
  - *サーバ(ワークステーション付属）（DevTest サーバ とも呼ばれる）*
  - ワークステーション（単体）
- **エンタープライズ ダッシュボード**
  - *新しいエンタープライズ ダッシュボード*
  - 既存のエンタープライズ ダッシュボード サービス
    このオプションは、devtestlic.xml ライセンス ファイルが使用可能である必要があります。
- **デモ サーバ**
  - インストール
    このオプションは、デモ サーバ インストール ファイルがダウンロードされており、存在している必要があります。
  - *インストールしない*

インストーラは、設定タイプの選択を求めるダイアログ ボックスを表示します。設定タイプは、「共有」と「ローカル」の 2 つです。共有設定は、単一のシステム上の複数のユーザに適しています。ローカル設定は、ユーザが他のシステムからインストールにアクセスする可能性があるインストールや、コンポーネントが複数のシステム上で制御され、実行されるインストール（分散設定）に適しています。

一般的な設定

スタンドアロン（DevTest コンポーネントをホストするために使用される単一のコンピュータ）

新しいスタンドアロン DevTest Solutions を 1 つのホストにインストールする場合は、3 つのコンポーネントをすべてインストールします。

- サーバ（ワークステーション付属）
- 新しいエンタープライズ ダッシュボード
- デモ サーバ

複数のユーザが単一のコンピュータ上の組み込み型 DevTest ワークステーション を使用する場合は、インストール タイプの共有オプションを選択します。 詳細については、「共有インストール タイプ」を参照してください。

分散サーバ（DevTest コンポーネントをホストするために使用される複数のコンピュータ）

分散 DevTest Solutions システムを複数のホストにインストールする場合は、以下の方法を検討します。

1. *最初のインストール*の場合は、システム要件を満たすサーバから、以下のオプションを選択します。
   - サーバ
   - 新しいエンタープライズ ダッシュボード（ここからライセンス ファイルの **devtestlic.xml** にアクセスする）。 インストーラは、シグネチャ、バージョン、および有効期限について、ライセンス ファイルおよびそのコンテンツを検証します。
2. *サーバの追加インストール*の場合は、以下のオプションを選択します。
   - サーバ
   - 既存のエンタープライズ ダッシュボード

   注: サーバの 1 つのコンポーネントのみを使用する予定であっても、サーバをインストールしてください。 たとえば、多くのメトリックを収集していたり、多数のレポートを要求していたりする場合は、他のサービスを実行していないコンピュータ上でコーディネータ サーバを実行できます。
3. CA Application Test および CA Service Virtualization ユーザ用に個別のホストにコンポーネントをインストールするには、以下のオプションを選択します。
   - ワークステーション

     ユーザがワークステーションにログインすると、それらのユーザは、いずれかのサーバと共にインストールされたレジストリにリンクします。
   - デモ サーバ - インストール（チュートリアルが役立つ可能性のある新しいユーザ用）
4. CA Continuous Application Insight ユーザの場合は、サポートされているブラウザ (P. 21)がインストールされていることを確認します。すべての作業は、Web ベース ポータルから実行されます。

必要なプロセスをインストールするためのオプション（Windows）

DevTest Solutions を使用するには、最大 7 つのプロセスを起動する必要があります。 インストール ウィザードでは、このタスクを実行する方法を選択できます。

- ［スタート］メニュー フォルダの作成
- インストール サービス

  注: これらのオプションのいずれかを選択するか、両方のオプションを選択するか、どのオプションも選択しないことが可能です。 どのオプションも選択しない場合は、DevTest サーバ インストール ディレクトリ（LISA_HOME¥bin）の bin フォルダから、またはコマンド プロンプトから、プロセス実行ファイルを起動できます。

DevTest Solutions のインストールおよびセットアップ方法

このインストール ガイドには、システム要件、概念的背景、および特定の状況でのみ必要な手順が含まれています。初めて DevTest Solutions をインストールする場合は、以下のガイドラインに従ってください。

1. DevTest Solutions インストーラをダウンロードします (P. 43)。
2. エンタープライズ ダッシュボードをインストールする場合は、**devtestlic.xml** ライセンス ファイルをターゲット システムに格納します。

   注: 将来、CA 製品サポートによって使用できるように、ライセンス ファイルの場所を記録しておいてください。
3. 汎用 UNIX インストーラを使用する場合は、独自の JVM を提供します (P. 13)。
4. 必要な数のシステムに DevTest サーバ をインストールします。 最初の DevTest サーバ をインストールする場合、［New Enterprise Dashboard］を選択し、**devtestlic.xml** ライセンス ファイルを参照します。
   - Windows (P. 53)
   - UNIX (P. 59)
   - Mac (P. 63)

Post-Installation ステップ

1. 外部データベースを設定します。
   - レジストリに関して外部データベースを設定します。（DB2、MySQL、Oracle、または SQL Server）。
   - エンタープライズ ダッシュボードに関して外部データベースを設定します。 （MySQL、Oracle、または SQL Server）

   注: DevTest Solutions と共にインストールされた Apache Derby データベースからエンタープライズ クラスのデータベースへのマイグレーション パスはありません。 Derby を使用する場合は、エンタープライズ クラスのデータベースが必要とする場合、システムを一般に使用可能にするときに、必須ユーザ認証（ロール）データを再入力する必要があります。
2. エンタープライズ ダッシュボードおよびすべてのレジストリをアクティブにします。

a. アップグレードする場合は、既存のレジストリを設定します (P. 70)。レジストリ設定は、リリース 8.0.0 以降では自動化されています。

b. レジストリをアクティブにします (P. 67)。

注: このプロセスでは、製品キーによりエンタープライズ ダッシュボードもアクティブ化されます。

c. レジストリのアクティブ化を確認します (P. 68)。(新規またはアップグレード)

3. インストール後 (P. 69) のタスクを実行します。
4. (オプション) 追加の DevTest ワークステーション をデモ サーバと共に、各レジストリで使用できるようにインストールします。
   - Windows (P. 86)
   - UNIX (P. 89)
   - Mac (P. 92)
5. DevTest Solutions のインストールを確認します (P. 103)。
6. 不正使用を防ぐために、標準ユーザのパスワードを変更します。
7. 必要な統合ツールをインストールします (P. 115)。
8. モバイル テスト環境を設定します (P. 157)。

インストールが完了したら、ACL 適用セキュリティの設定および使用許諾契約の遵守の詳細について「管理」を参照してください。

ライセンス アクティブ化の仕組み

企業ごとに 1 つの DevTest Solutions ライセンス（**devtestlic.xml**）が発行されます。 このファイルにより、DevTest Solutions のすべての機能のロックが解除されます。 DevTest Solutions セットアップ ウィザードで最初の DevTest サーバ をインストールするときに、ライセンス ファイルに移動します。 その後、セットアップ プロセスによって、エンタープライズ ダッシュボードがインストールされ、そのファイルが DevTest サーバ インストール ディレクトリ（LISA_HOME）に配置されます。 追加の DevTest レジストリ サーバをインストールする場合は、エンタープライズ ダッシュボード サーバへの URL を提供します。

注: エンタープライズ ダッシュボード データベースの場所を変更するには、現在のデータベース スキーマを保持しながら、インストーラを再実行する必要があります。 再インストールの前に、カスタマイズされたプロパティ ファイルをバックアップしてください。 これらには、local.properties、site.properties、および各種 vmoptions ファイルが含まれます。

重要: エンタープライズ ダッシュボードへの URL が変更される場合は、エンタープライズ ダッシュボードにレポートする各レジストリ サーバの LISA_HOME ディレクトリにある **local.properties** ファイルを更新する必要があります（新しい URL で **lisa.enterprisedashboard.service.url** プロパティを設定します）。

エンタープライズ ダッシュボード プロセスおよび各レジストリ プロセスを起動すると、エンタープライズ ダッシュボード プロセスがレジストリ設定を読み取り、レジストリをアクティブにします。 アクティブにされたレジストリは、ユーザが確認できるように、エンタープライズ ダッシュボード UI に表示されます。

重要: レジストリのホスト名またはポートが変更される場合は、エンタープライズ ダッシュボードがレジストリを再アクティブ化できるように、レジストリを再起動する必要があります。

注: 「*管理*」の「レジストリまたはエンタープライズ ダッシュボードの再アクティブ化」を参照してください。

第 3 章: DevTest サーバ インストールおよびインストール後の作業

このセクションでは、DevTest サーバ をインストールおよび設定する方法を説明します。

このセクションには、以下のトピックが含まれています。

Windows での DevTest サーバ のインストール

このトピックでは、DevTest Solutions サーバ コンポーネントを Windows 環境にインストールする方法について説明します。

インストールの選択肢を事前に確認するには、「インストーラの実行時にインストールできるもの (P. 46)」を参照してください。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

  注: 通常、デモ サーバは、スタンドアロン システムでのみ、サーバ コンポーネントと共にインストールされます。分散システムでは、通常、デモ サーバは、DevTest Solutions の新規ユーザ用に DevTest ワークステーション と共にインストールされます。 デモ サーバは、チュートリアルのために、またサンプル プロジェクトの多数のアーティファクトのために使用されます。

次の手順に従ってください:

1. インストーラ ファイル（devtest_win_x64.exe など）を実行します。

   ［Welcome to the DevTest Solutions Setup Wizard］手順が表示されます。

2. ［次へ］をクリックします。

   ［CA End User License Agreement］手順が表示されます。

3. ライセンス契約書を読み、最後までスクロールして、［I accept the terms of the License Agreement］オプションを選択し、［Next］をクリックします。

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

4. インストール ディレクトリ（LISA_HOME）のパスおよびフォルダ名を入力します。現在のリリースを古いリリースと分離しておきたい場合は、リリース識別子を含む名前を検討してください。 たとえば、「C:¥DevTestServer_8.0」と入力します。または、デフォルト（C:¥Program Files¥CA¥DevTestSolutions）をそのまま使用します。 パスを参照し、新しいフォルダの名前を入力すると、そのフォルダがインストール ウィザードによって作成されます。

5. ［次へ］をクリックします。

［Installation Type］手順が表示されます。

6. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

   複数のユーザがログインして DevTest ワークステーション を使用できる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。 このインストール タイプは、スタンドアロン インストールに適したオプションです。

   ［Select Components］手順が表示されます。

7. ［Server］チェック ボックスがオンになっていることを確認します。これにより、組み込み型のワークステーションがサーバと共にインストールされます。 ［次へ］をクリックします。

［Enterprise Dashboard］手順が表示されます。

8. 以下のいずれかのオプションを指定し、［Next］をクリックします。

   New Enterprise Dashboard（**ライセンス ファイルの場所を指定する**）

   これが初めてインストールするサーバである場合は、［Browse］をクリックして、ライセンス ファイルの場所に移動し、**devtestlic.xml** を選択して、［Open］をクリックします。インストーラは、**devtestlic.xml** ファイルを、ローカル ホスト上の指定されたインストール ディレクトリ（LISA_HOME）にコピーします。 新しいエンタープライズ ダッシュボード プロセスが LISA_HOME¥bin ディレクトリにインストールされます。 このオプションは、すべてのレジストリが新しいエンタープライズ ダッシュボードに接続されることを指定します。

   Existing Enterprise Dashboard Service

   これがこのネットワークで初めてインストールするサーバでない場合は、既存のエンタープライズ ダッシュボードの URL を入力します。 このオプションは、このサーバと共にインストールされるレジストリが既存のエンタープライズ ダッシュボードに接続されることを指定します。「*localhost*」を適切なホスト名に置き換えます。

   ```
   tcp://localhost:2003/EnterpriseDashboard
   ```

   ［Demo Server］手順が表示されます。

9. インストーラで DevTestDemoServer.zip を LISA_HOME ディレクトリに解凍する場合は、［Install demo server］オプションをオンにします。次に、デフォルト パス（［ダウンロード］ディレクトリ）をそのまま使用するか、別の完全修飾パスを指定します。

   注: ユーザのコンピュータにワークステーションがインストールされる分散システムでは、デモ サーバは、通常、サーバではなく新規ユーザ用のワークステーションと共にインストールされます。

10. ［次へ］をクリックします。

11. ［Local］インストール タイプを選択した場合は、次の手順（［Shared］インストール タイプに適用される）をスキップします。

12. 各手順の後に［Next］をクリックして、データ ディレクトリを指定します。

    ［Select Start Menu Folder］手順が表示されます。

13. （オプション） DevTest Solutions プロセスが［スタート］メニューから起動できるかどうかを指定します （各プロセスの実行ファイルは、インストール ディレクトリの bin ディレクトリから手動で起動できます）。実行ファイルを起動する利点は、関連サービスとは対照的に、コマンド ライン インターフェース（CLI）に表示されるメッセージを監視できることです。
    - すべてのユーザについてショートカットを持つ［スタート］メニュー フォルダを作成するには、すべてデフォルト設定のままにします。 必要に応じて、新しいフォルダ名を入力します。
    - スタート メニュー フォルダを作成しない場合は、「Create a Start Menu folder］チェック ボックスをオフにします。
    - ［スタート］メニュー フォルダを作成し、ショートカットの表示をお使いの［スタート］メニューに制限する方法
        - ［Create a Start Menu folder］をオンのままにします。
        - デフォルトの名前を使用するか、別の名前を入力します。
        - ［Create shortcuts for all users］チェック ボックスをオフにします。
14. ［次へ］をクリックします。

    ［Desktop Icons］手順が表示されます。
15. （オプション）DevTest エンタープライズ ダッシュボード UI、DevTest ポータル UI、および DevTest ワークステーションのデスクトップ アイコンを作成しない場合は、［Create a desktop icon］チェック ボックスをオフにします。 ［次へ］をクリックします。
    - ［Local］インストール タイプを選択した場合、管理者であれば、［Windows Services］手順が表示されます。
    - ［Local］インストール タイプを選択した場合、管理者でなければ、次の手順をスキップします。
    - ［Shared］インストール タイプを選択した場合は、［information］手順に進みます。

16. （オプション）［Install Services］を選択して、以下の Windows サービスを作成します。

    - DevTest ブローカ サービス（CAI 用）
    - DevTest コーディネータ サービス
    - DevTest エンタープライズ ダッシュボード サービス
    - DevTest ポータル サービス
    - DevTest レジストリ サービス
    - DevTest シミュレータ サービス
    - DevTest VSE サービス

    この選択内容により、サービスが［管理ツール］-［コンポーネント サービス］-［サービス］に追加されます。 サービスを起動する利点は、関連する実行ファイルとは対照的に、システム トレイに表示されるアイコンの数が少なくなることです。［スタートアップの種類］を［自動］として定義して、ホスト コンピュータの再起動時にサービスが起動するようにするには、［Start on bootup］チェック ボックスをオンにします。ここでこのチェック ボックスをオンにしなくても、［サービス］内で自動スタートアップを設定できます。

17. ［次へ］をクリックします。

    ［Select File Associations］手順が表示されます。デフォルトでは、すべての関連付けがオンになっています。

18. すべての関連付けをオンのままにしておくか、DevTest Solutions に関連付けないファイル拡張子をオフにします。 DevTest Solutions と関連付けることができるファイル拡張子には、以下のものが含まれます。

    - *.tst -- CA Application Test でテスト ケースを作成するには、この拡張子を選択します。
    - *.vsm および *.vsi -- CA Service Virtualization で仮想サービスを作成するには、これらの拡張子を選択します。
    - *.ste -- CA Application Test のテスト ランナーでスイートを実行するには、この拡張子を選択します。
    - *.stg -- CA Application Test でテスト ケースをステージング ドキュメントとして実行するには、この拡張子を選択します。

19. ［Install］をクリックしてインストールを開始します。

［Installing］手順が表示されます。 インストールが完了すると、［Information］手順が表示されます。

20. 情報を読み、［Next］をクリックします。

［Completing the DevTest Solutions Setup Wizard］手順が表示されます。

21. ［終了］をクリックします。

「DevTest Solutions をインストールする方法 (P. 50)」の説明に従って作業を続行します。

UNIX での DevTest サーバ のインストール

このトピックでは、DevTest サーバ を UNIX または Linux 環境にインストールする方法について説明します。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

  注: 通常、デモ サーバは、スタンドアロン システムでのみ、サーバ コンポーネントと共にインストールされます。分散システムでは、通常、デモ サーバは、DevTest Solutions の新規ユーザ用に DevTest ワークステーション と共にインストールされます。

以下の手順は、GUI 版のインストーラに基づいています。 コマンドライン版のインストーラを使用するには、**-c** オプションを追加します。以下に例を示します。

```
./devtest_platform_x64.sh -c
```

注: 汎用 UNIX インストーラを使用する場合は、Java 仮想マシン（JVM） (P. 13)が同じコンピュータにあることを確認してください。 JVM のバージョンは 1.7 である必要があります。 **JAVA_HOME** 環境変数を設定することにより、特定の JVM を指定できます。 インストーラが JVM を検出できない場合、インストーラはメッセージを表示して終了します。

次の手順に従ってください:

1. ターミナル ウィンドウで、インストーラ ファイルがあるディレクトリに移動します。
2. インストーラ ファイルに実行権限があることを確認します。

   `chmod 777 devtest_platform_x64.sh`

   このコマンドにより、rwxrwxrwx 権限がファイルに付与されます。
3. インストーラ ファイルを実行します。 以下に例を示します。

   `./devtest_platform_x64.sh`

   ［Welcome to the DevTest Solutions Setup Wizard］が表示されます。
4. ［次へ］をクリックします。

   ［CA End User License Agreement］手順が表示されます。
5. ライセンス契約書を読み、「I accept the terms of the License Agreement」チェック ボックスをオンにし、［Next］をクリックします。

   ［Select Destination Directory］手順が表示されます。
6. DevTest Solutions をインストールするディレクトリ（/opt/CA/DevTest など）を指定します。 スペースが含まれるディレクトリ名によるパスを使用しないでください。
7. ［次へ］をクリックします。

［Installation Type］手順が表示されます。

8. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

   複数のコンピュータから複数のユーザがアクセスできる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。 共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。

   ［Select Components］手順が表示されます。

9. ［Server］チェック ボックスがオンになっていることを確認し、［Next］をクリックします。

   ［Enterprise Dashboard］手順が表示されます。

10. 以下のいずれかのオプションを指定し、［Next］をクリックします。

    New Enterprise Dashboard（**ライセンス ファイルの場所を指定する**）

    これが初めてインストールする DevTest サーバ である場合は、［Browse］をクリックして、ライセンス ファイルの場所に移動し、**devtestlic.xml** を選択して、［Open］をクリックします。 ［Next］をクリックすると、インストーラは、devtestlic.xml ファイルを、ローカル ホスト上のインストール ディレクトリ（LISA_HOME）にコピーします。 エンタープライズ ダッシュボード プロセスが LISA_HOME/bin ディレクトリにインストールされます。

    Existing Enterprise Dashboard Service

    この DevTest サーバ のレジストリが、最初にインストールした DevTest サーバ によってインストールされたエンタープライズ ダッシュボードに接続する場合は、既存のエンタープライズ ダッシュボード（ED）の URL を入力します。

［Specify Demo Server］手順が表示されます。

11. インストーラに、DevTest サーバ をインストールしているディレクトリにデモ サーバを解凍させる場合は、［Install demo server］チェック ボックスをオンにして、DevTestDemoServer.zip ファイルを選択します。
12. ［次へ］をクリックします。

    インストール タイプに［Shared］を選択した場合は、以下の手順でデータ ディレクトリおよび一時ファイル ディレクトリを指定するように促されます。
13. 各手順の後に［Next］をクリックして、ディレクトリを指定します。

    ［Select Directory for Symlinks］手順が表示されます。
14. ［Browse］をクリックし、DevTest が実行ファイルへのシンボリック リンクを作成するディレクトリに移動します。 デフォルトは /usr/local/bin です。 ディレクトリに書き込むための権限が必要です。 シンボリック リンクを作成しない場合は、［Create symlinks］チェック ボックスをオフにします。
15. ［次へ］をクリックします。

    ［Desktop Icons］手順が表示されます。
16. （オプション）DevTest エンタープライズ ダッシュボード、DevTest ポータル UI、および DevTest ワークステーションのデスクトップ アイコンを作成しない場合は、チェック ボックスをオフにします。
17. ［Install］をクリックしてインストールを開始します。

    インストールが完了すると、［Information］手順が表示されます。
18. 情報を読み、［Next］をクリックします。

    ［Completing the DevTest Solutions Setup Wizard］手順が表示されます。
19. ［終了］をクリックします。

「DevTest Solutions をインストールする方法 (P. 50)」の説明に従って作業を続行します。

Mac での DevTest サーバ のインストール

このトピックでは、DevTest Solutions サーバ コンポーネントを Mac にインストールする方法について説明します。

インストールの選択肢を事前に確認するには、「インストーラの実行時にインストールできるもの (P. 46)」を参照してください。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

  注: 通常、デモ サーバは、スタンドアロン システムでのみ、サーバ コンポーネントと共にインストールされます。分散システムでは、通常、デモ サーバは、DevTest Solutions の新規ユーザ用に DevTest ワークステーション と共にインストールされます。 デモ サーバは、チュートリアルに使用されます。

次の手順に従ってください:

1. インストーラ ファイル（たとえば、devtest_osx_x64.dmg）を実行します。

   ［Welcome to the DevTest Solutions Setup Wizard］手順が表示されます。
2. ［次へ］をクリックします。

   ［CA End User License Agreement］手順が表示されます。
3. ライセンス契約書を読み、最後までスクロールして、［I accept the terms of the License Agreement］オプションを選択し、［Next］をクリックします。

   ［Select Destination Directory］手順が表示されます。
4. DevTest Solutions の 1 つ以上のコンポーネントをインストールするフォルダを指定します。 存在しないフォルダを指定すると、そのフォルダがセットアップ ウィザードによって作成されます。
5. ［次へ］をクリックします。

［Installation Type］手順が表示されます。

6. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

   複数のユーザがログインして DevTest ワークステーション を使用できる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。 このインストール タイプは、スタンドアロン インストールに適したオプションです。

   ［Select Components］手順が表示されます。

7. ［Server］チェック ボックスがオンになっていることを確認します。これにより、組み込み型のワークステーションがサーバと共にインストールされます。 ［次へ］をクリックします。

［Enterprise Dashboard］手順が表示されます。

8. 以下のいずれかのオプションを指定し、［Next］をクリックします。

   New Enterprise Dashboard（**ライセンス ファイルの場所を指定する**）

   これが初めてインストールするサーバである場合は、［Browse］をクリックして、ライセンス ファイルの場所に移動し、**devtestlic.xml** を選択して、［Open］をクリックします。インストーラは、**devtestlic.xml** ファイルを、ローカル ホスト上の指定されたインストール ディレクトリ（LISA_HOME）にコピーします。 新しい Enterprise Dashboard プロセスが、LISA_HOME/bin ディレクトリにインストールされます。 このオプションでは、このサーバと共にインストールされたレジストリが、新しいエンタープライズ ダッシュボードに接続されることを指定します。

   Existing Enterprise Dashboard Service

   これがこのネットワークで初めてインストールするサーバでない場合は、既存のエンタープライズ ダッシュボードの URL を入力します。 このオプションは、このサーバと共にインストールされるレジストリが既存のエンタープライズ ダッシュボードに接続されることを指定します。「*localhost*」を適切なホスト名に置き換えます。

   ```
   tcp://localhost:2003/EnterpriseDashboard
   ```

   ［Demo Server］手順が表示されます。

9. インストーラでデモ サーバを LISA_HOME ディレクトリに解凍する場合、［Install demo server］オプションを選択します。 次に、デフォルト パス（［ダウンロード］ディレクトリ）をそのまま使用するか、別の完全修飾パスを指定します。

   注: ユーザのコンピュータにワークステーションがインストールされる分散システムでは、デモ サーバは、通常、サーバではなく新規ユーザ用のワークステーションと共にインストールされます。

10. ［次へ］をクリックします。

11. ［Local］インストール タイプを選択した場合は、次の手順（［Shared］インストール タイプに適用される）をスキップします。

12. 各手順の後に［Next］をクリックして、データ ディレクトリを指定します。

13. ［次へ］をクリックします。

［Desktop Icons］手順が表示されます。

14. （オプション）DevTest エンタープライズ ダッシュボード、DevTest ポータル UI、および DevTest ワークステーションのデスクトップ アイコンを作成しない場合は、チェック ボックスをオフにします。［次へ］をクリックします。

15. ［次へ］をクリックします。

    ［Select File Associations］手順が表示されます。デフォルトでは、すべての関連付けがオンになっています。

16. すべての関連付けをオンのままにしておくか、DevTest Solutions に関連付けないファイル拡張子をオフにします。 DevTest Solutions と関連付けることができるファイル拡張子には、以下のものが含まれます。

    - *.tst -- CA Application Test でテスト ケースを作成するには、この拡張子を選択します。
    - *.vsm および *.vsi -- CA Service Virtualization で仮想サービスを作成するには、この拡張子を選択します。
    - *.ste -- CA Application Test のテスト ランナーでスイートを実行するには、この拡張子を選択します。
    - *.stg -- CA Application Test でテスト ケースをステージング ドキュメントとして実行するには、この拡張子を選択します。

17. ［Install］をクリックしてインストールを開始します。

    ［Installing］手順が表示されます。 インストールが完了すると、［Information］手順が表示されます。

18. 情報を読み、［Next］をクリックします。

    ［Completing the DevTest Solutions Setup Wizard］手順が表示されます。

19. ［終了］をクリックします。

「DevTest Solutions をインストールする方法 (P. 50)」の説明に従って作業を続行します。

レジストリのアクティブ化

DevTest サーバ のインスタンスをすべてインストールした後、インストールした最初の DevTest サーバ 上で エンタープライズ ダッシュボード プロセスを起動します。 その後、その DevTest サーバ およびその他すべての DevTest サーバ 上のレジストリ プロセスを起動します。 レジストリは、すぐには有効化されません。 たとえば、10:15 と 10:30 の間にレジストリを再起動すると、有効化されるのは 11:00 です。

新規および既存の各 DevTest サーバ のレジストリがエンタープライズ ダッシュボードに表示されることを確認します。

次の手順に従ってください:

1. エンタープライズ ダッシュボード サーバを起動します。
   a. エンタープライズ ダッシュボードがインストールされているホストにログオンします。
   b. エンタープライズ ダッシュボード サーバを起動します。

      Windows ユーザは、［スタート］メニューの［エンタープライズ ダッシュボード］オプションから［エンタープライズ ダッシュボード サーバ］を選択できます。 あるいは、LISA_HOME¥bin ディレクトリに移動し、EnterpriseDashboard.exe を起動します。
2. 既存のレジストリがある場合は、既存のレジストリを設定します (P. 70)。
3. 各レジストリを起動します。
   a. DevTest サーバ がインストールされているホストにログオンします。
   b. レジストリを起動します。

      Windows ユーザは、DevTest Solutions の［スタート］メニュー、または DevTest サーバ インストール ディレクトリの bin ディレクトリにある Registry.exe からレジストリを起動します。
   c. 各レジストリでこれらの手順を繰り返します。

      注: このプロパティが設定された後、次の時間が始まるまでは、レジストリはエンタープライズ ダッシュボードにデータをプッシュしません。たとえば、午後 1:35 にこのプロパティを設定した場合、午後 2:00 になるまでレジストリ データは表示されません。

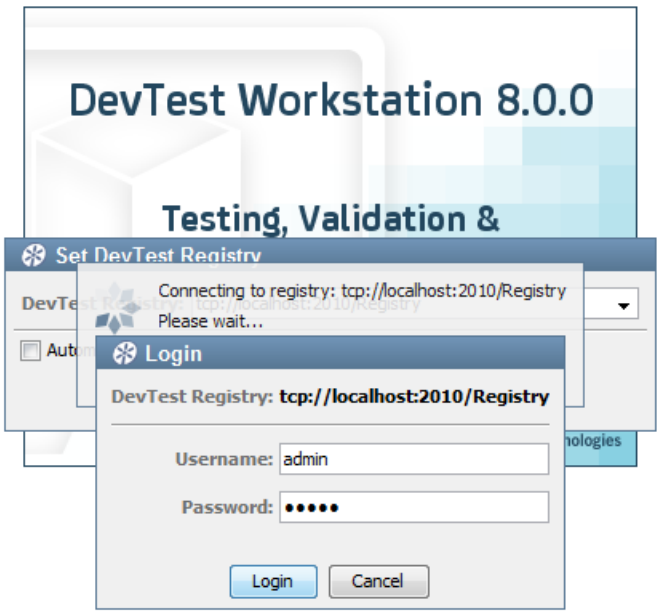
レジストリのアクティブ化の確認

各レジストリを起動した後、次時の 00 分になるまで待ってから、レジストリのアクティブ化を確認します。

次の手順に従ってください:

1. 以下のいずれかの方法でエンタープライズ ダッシュボード UI を開きます。
   - ブラウザでエンタープライズ ダッシュボードにアクセスします。IP アドレスまたはホスト名を指定する（リモートでインストールした場合）か、**localhost** を指定します（ローカルでインストールした場合）。

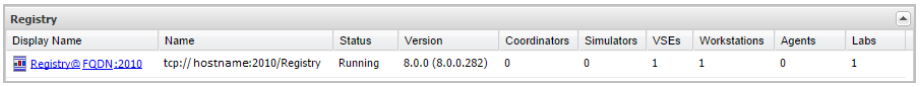
     `http://hostname:1506`
   - エンタープライズ ダッシュボードがインストールされているコンピュータから、Windows ユーザが、［スタート］-［Enterprise Dashboard］-［Enterprise Dashboard UI］を選択します。
2. ログインします。



ログインするには、［ユーザ名］フィールドに「*admin*」と入力し、［パスワード］フィールドに「*admin*」と入力して、［ログイン］をクリックします。 エンタープライズ ダッシュボードが表示されます。

3. エンタープライズ ダッシュボードでレジストリ設定を確認します。

   レジストリのリストは、以下の例と似た形式で表示されます。



4. 新規および既存の各 DevTest サーバ のレジストリがエンタープライズ ダッシュボードに表示されることを確認します。

注: 既存のレジストリが表示されない場合は、既存のレジストリを設定します (P. 70)。

インストール後

このセクションには、以下のトピックが含まれます。

既存のレジストリの設定

インストールする DevTest サーバ はそれぞれ 1 つのレジストリを持ちます。製品ライセンスは、レジストリが設定されることを必要とします。設定プロセスは、リリースによって異なります。

- **バージョン 8.0 以降：** インストール プロセスにより、新しいレジストリが自動的に設定されます。
- **リリース 7.5.x：** 「レジストリを設定する方法（DevTest 7.5.x）」の手順に従います。
- **7.5 より前のリリース：** 「DevTest 7.5 より前のリリースでレジストリを追加する方法」の手順に従います。

レジストリを設定する方法（DevTest 7.5.x）

1. DevTest サーバ がインストールされているコンピュータにログオンし、インストール ディレクトリに移動します。
2. local.properties を開き、「Section 1 - Enterprise Dashboard」を見つけます （local.properties ファイルがない場合は、_local.properties をコピーし、その名前を local.properties に変更します）。
3. 以下の行のコメントを外して、「*somehost*」をエンタープライズ ダッシュボードがインストールされているホスト名に置き換えます。

   ```
   lisa.enterprisedashboard.service.url=tcp://somehost:2003/EnterpriseDashboard
   ```

4. local.properties ファイルを保存し、ファイルを閉じます。
5. （オプション）リモート コンピュータからの DevTest ワークステーションを使用することを予定している場合は、特定のレジストリに自動的に接続することを可能にするプロパティを指定できます。
   a. ワークステーション がインストールされているコンピュータにログオンします。
   b. LISA_HOME に移動します。
   c. local.properties を開き、「Section 2 - Autoconnection」を見つけます
   d. 以下の行のコメントを外して、「*somehost*」を、ログオンしているコンピュータのホスト名に置き換えます。

      ```
      lisaAutoConnect=tcp://somehost:2010/Registry
      ```

   e. local.properties ファイルを保存し、ファイルを閉じます。
6. ターゲット レジストリのあるコンピュータにログオンします
7. レジストリを起動または再起動します。

a. LISA_HOME¥bin ディレクトリに移動します。

b. registry.exe を実行します。

8. レジストリごとに手順 1 ～ 7 を繰り返します。

9. レジストリの設定を確認するには、エンタープライズ ダッシュボードを参照します。

   `http://hostname:1506`

重要: DevTest Solutions をインストールするシステムのホスト名が変更された場合は、この手順を繰り返します。

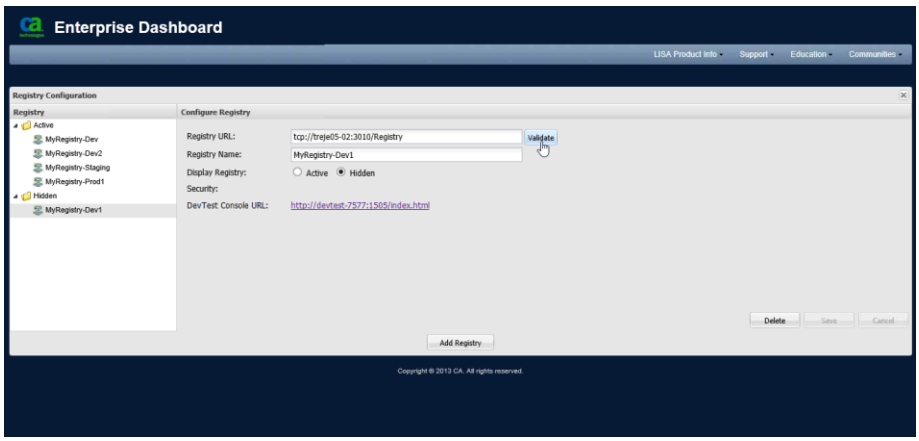
エンタープライズ ダッシュボードの［レジストリ設定］ウィンドウでは、古いレジストリを追加および設定したり、ダッシュボードに表示される使用されていないレジストリを削除したりすることができます。

DevTest 7.5 より前のリリースでレジストリを追加する方法

1. エンタープライズ ダッシュボードを開きます。
2. エンタープライズ ダッシュボードのメイン ウィンドウの右上隅にある［オプション］をクリックします。
3. ［設定］を選択します。

   ［レジストリ設定］ウィンドウが表示されます（［レジストリの設定］フィールドは空白の状態）。

   注: メイン ウィンドウに戻るには、［レジストリ ビューに戻る］ をクリックするか、またはページの左上隅にあるパンくずリスト内の［概要］をクリックします。



4. ［レジストリ設定］ウィンドウ内の［レジストリの追加］をクリックします。 確認のために［はい］をクリックします。
5. 以下のフィールドに入力します。

   レジストリ URL

   レジストリの URL。 このフィールドは必須です。

   例：

   `tcp://hostname:2010/Registry`

   URL を検証するには、［検証］をクリックします。

   レジストリ名

   このレジストリの共通名またはニックネーム。

   例：

   `MyRegistry@hostname:2010`

   レジストリの表示

   このレジストリをダッシュボードのメイン パネルに表示するには、［アクティブ］を選択します。このレジストリを非表示にするには、［非表示］を選択します。 このパラメータにより、このレジストリが左側のどのフォルダに含まれるかも制御されます。

   ［レジストリ設定］ウィンドウには、以下のフィールドもあります。

   セキュリティ

   ACL セキュリティが有効になっているか、無効になっているかを示します。

   DevTest コンソール URL

   このレジストリの DevTest コンソールの URL 例：

   `http://hostname:1505/index.html`
6. ［保存］をクリックすると、レジストリがダッシュボードに追加されます。

レジストリを検証する方法

1. ［オプション］メニューで、［設定］を選択します。
2. ［レジストリ］列のレジストリのリストから、検証するレジストリをクリックします。
3. ［検証］をクリックします。

   ダッシュボード サーバは、選択したレジストリのクエリを実行し、ステータス（実行中など）を表示します。
4. ［OK］をクリックします。

HTTP/S プロキシ サーバの使用 - DevTest サーバ

プレーン HTTP プロキシ サーバまたは SSL で保護された HTTP プロキシ サーバを使用している場合は、LISA_HOME ディレクトリにある **local.properties** ファイルで、そのプロキシ サーバと除外するホストを定義します。

次の手順に従ってください:

1. DevTest サーバ がインストールされているホストにログオンします。
2. LISA_HOME に移動します。
3. local.properties が存在しない場合は、**_local.properties** をコピーし、そのコピーを local.properties（アンダースコアなし）として保存します。
4. local.properties を編集するために開き、HTTP プロキシ サーバまたは HTTPS プロキシ サーバのいずれか（プロキシ サーバがプレーン HTTP を使用しているか SSL で保護された HTTP を使用しているかによって異なる）のセクション ヘッダを見つけます。
5. FQDN または IP アドレスおよびポートによってプロキシ サーバを識別します。
   - HTTP サーバの場合は、lisa.http.webProxy.host および lisa.http.webProxy.port プロパティを使用します
   - HTTPS サーバの場合は、lisa.http.webProxy.ssl.host および lisa.http.webProxy.ssl.port プロパティを使用します
6. プロキシ サーバを通過することから除外するホストを識別します。
   - HTTP サーバの場合は、lisa.http.webProxy.nonProxyHosts プロパティを使用します
   - HTTPS サーバの場合は、lisa.http.webProxy.ssl.nonProxyHosts プロパティを使用します
7. プロパティの前のコメント記号を削除したことを確認してください。
8. ファイルを保存して終了します。

例:

以下の例の最初の 2 行は、HTTP プロキシ サーバの URL が **http://192.168.24.242:49185** であることを指定します。

3 行目は、このプロキシを通過しないホストに、localhost（127.0.0.1）のループバック アドレスと 192.168.32.255 ～ 192.168.32.0 の範囲の IP アドレスが含まれることを指定します。除外する IP アドレスの区切り文字としてパイプ記号（|）が使用されることに注意してください。また、ワイルドカード（*）が任意の有効な値（IP アドレス ノードの有効な値は 0 ～ 255）を表すことにも注意してください。 除外するホストが標準命名規則を共有している場合は、ワイルドカード文字を FQDN およびホスト名と共に使用することもできます。

```
lisa.http.webProxy.host=192.168.24.242
lisa.http.webProxy.port=49185
lisa.http.webProxy.nonProxyHosts=127.0.0.1|192.168.32.*
```

local.properties での HTTP/S プロキシ サーバ設定

```
## ============================================
## HTTP Proxy Server
## ============================================
#lisa.http.webProxy.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.port=

## ============================================
## HTTPS Proxy Server
## ============================================
#lisa.http.webProxy.ssl.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.ssl.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.ssl.port=

## === Leave blank to use integrated NTLM authentication
#lisa.http.webProxy.host.domain= used for NTLM authentication
#lisa.http.webProxy.host.account=
#lisa.http.webProxy.host.credential=

## === Exclude simple host names from proxy use - default value is true
#lisa.http.webProxy.nonProxyHosts.excludeSimple=false

## === Preemptively send authorization information rather than waiting for a
challenge
## ===== valid values are basic or ntlm
#lisa.http.webProxy.preemptiveAuthenticationType=ntlm
```

異なるシステムで動作するコンポーネント

サーバ コンポーネントが別のシステム上にある場合は、以下のプロパティを正しく使用します。

- レジストリは、**lisa.registryName** プロパティを使用して自身にデフォルト以外の名前を付けます。
- 非レジストリ サーバ コンポーネントはロケータとして **lisa.registry.url** プロパティを使用します。

非レジストリ サーバ コンポーネントの **local.properties** ファイルでは、**lisa.registry.url** プロパティでレジストリを指定します。

```
lisa.registry.url=tcp://registry-hostname-or-ip:port/registry-name
```

以下に例を示します。

```
lisa.registry.url=tcp://myserver.example.com:2010/Registry
```

この目的のために **lisa.registryName** プロパティを使用しないでください。

レジストリを指定する別のオプションは、非レジストリ サーバ コンポーネントを起動するときに引数として **-m** を渡すことです。

```
./CoordinatorServer -m tcp://registry-hostname-or-ip:port/registry-name
```

以下に例を示します。

```
./CoordinatorServer -m tcp://myserver.example.com:2010/Registry
```

シミュレータ インスタンスの算出

特定のシミュレータのインスタンス数を算出するには、以下の分析を実行します。

1. DevTest ワークステーション を起動し、レジストリを選択して、ログインします。［ヘルプ］-［DevTest ランタイム情報］のメモリ使用量を記録します。
2. テスト スイートをローカルで実行し、［ヘルプ］-［DevTest ランタイム情報］のメモリ使用量を記録します。
3. 手順 2 と手順 1 のメモリ使用量の差を確認します。
4. 使用可能な RAM に 60 パーセントを掛けます。
5. 手順 4 の使用可能な RAM を手順 3 のメモリ使用量で割ります。

手順 5 の結果は、シミュレータ サーバで設定する必要のある仮想ユーザ（インスタンス）の数の良い初期推定値となります。

コーディネータ サーバとレジストリの両方がシミュレータ サーバと同じサーバ上で実行されている場合は、使用可能な RAM に 40 パーセントを掛けます。 コーディネータ サーバがすべてのレポートおよびメトリックを収集し、そのために RAM を消費するので、60 パーセントではなく 40 パーセントを使用します。

この方法により、出発点が得られます。 各シミュレータについてインスタンスの適切な数に到達するには、何回かの反復とその他の直観的な方法を使用します。

注: コマンド ライン オプションを使用して、シミュレータの同時インスタンスの数を設定できます。詳細については、コマンド プロンプトを開き、LISA_HOME¥bin ディレクトリに移動して、「**simulator --help**」と入力してください。

負荷およびパフォーマンス サーバのサイジング

特定の負荷テストに必要なシミュレーション サーバの数を算出するのは簡単ではありません。 必要なサーバの数は、以下のような多くの要因に左右されます。

- サーバ ホスト設定（CPU の数、RAM の容量）
- テスト ケース フットプリント（テスト ステップの数、テスト ステップのタイプ）
- その他のテスト要件（レポートの数、データ セットのサイズ）

パフォーマンス テストのテスト ランを数回実行することをお勧めします。これらのテスト ランによって、DevTest サーバ 環境の構成の決定に役立つデータを収集できます。 メトリックの収集とメモリおよび CPU 使用率のモニタは、特定のシミュレータ サーバで使用できる仮想ユーザ数の算出に非常に重要です。

レジストリは軽量で、コンピューティング リソースをほとんど必要としません。 レジストリは、ネットワーク内のほとんどすべてのコンピュータから実行できます。

コーディネータ サーバはリソースを必要とします。 コーディネータ サーバは専用のコンピュータを必要としませんが、一般に個別のコンピュータにインストールされます。 多くのメトリックを収集する場合、多くのレポートを要求する場合、またはその両方の場合は、この方法に従います。

シミュレータ サーバは、何千もの仮想ユーザをシミュレートするために使用されます。 物理サーバごとに 1 つのシミュレータ サーバを実行することをお勧めします。 技術的には、任意の数のインスタンスで単一のシミュレータ サーバを起動できます。 ただし、通常、サーバ メモリのサイズおよび速度により、各シミュレータのインスタンスの数が制限されます。適切な上限は約 250 の仮想ユーザです。

サーバのサイジングには、縦方向の拡張と横方向の拡張を使用できます。縦方向の拡張では、通常制限されている CPU 速度を上げ、使用可能なメモリを増加させます。 横方向の拡張では、サーバを追加します。 仮想ユーザの数を増加させるには、横方向の拡張をお勧めします。

シミュレータ当たりのインスタンスの数は、多くの要因に依存します。インスタンスの最大数の算出に単純なルールは使用できません。

ネットワーク遅延は、負荷とパフォーマンスに影響を与えます。 データベースは、DevTest の主要コンポーネントとして、同じデータ センター内のサーバ上に配置することをお勧めします。

Java 環境での DevTest ワークステーション の使用

DevTest インストールによって使用されたデフォルトの JRE を独自の Java 環境に変更できます。 それには、DevTest ワークステーション がどのように使用する JRE を選択するかを理解することが重要です。

DevTest ワークステーション の起動時には、以下の優先度で使用する Java VM が選択されます。

1. DevTest ワークステーション によって LISA_HOME¥jre ディレクトリにインストールされている JRE
2. LISA_JAVA_HOME 環境変数
3. JAVA_HOME 環境変数
4. JDK_HOME 環境変数

次の手順に従ってください:

1. LISA_HOME¥jre ディレクトリの名前を変更します（たとえば、jre を jre_default に変更する）。
2. LISA_JAVA_HOME 環境変数が Java インストール ディレクトリを指すようにします。

「独自の JVM の提供 (P. 13)」も参照してください。

デフォルト プロジェクト ホームの変更

デフォルトでは、プロジェクトは **LISA_HOME¥Projects** ディレクトリに保存されます。

この手順では、DevTest ポータルのデフォルトの場所を変更する方法について説明します。DevTest ワークステーション のデフォルトの場所は変更できません。

次の手順に従ってください:

1. **$LISA_HOME** ディレクトリに移動します。
2. **res-hub-config.properties** という名前のテキスト ファイルを作成します。
3. **resHub.projects.dir** プロパティをファイルに追加します。 Windows プラットフォーム上であっても、ディレクトリ パスには必ずスラッシュを使用します。 以下に例を示します。

   ```
   resHub.projects.dir=C:/MyNewProjectHome
   ```

4. ファイルを保存して閉じます。
5. ポータル サーバ コンポーネントが実行中の場合は、再起動します。

プロジェクト ディレクトリ構造

テスト アセット（たとえばプロジェクト）を使用するサーバ コンポーネントがテスト アセットを確実に使用できるようにすることをお勧めします。

テスト アセットへのアクセスを管理するための要件は以下のとおりです。

- 命名規則を使用します。 複数のチームが同じサーバ環境を使用できます。 所有者および目的を区別し、秩序を維持するために、命名規則を使用します。
- プロジェクト名は一意である必要があります。 サーバ環境では、2 つの展開されたプロジェクトが同じ名前である場合、予期しない状況が発生する可能性があります。

DevTest サーバ のアンインストール

LISA_HOME ディレクトリの最上位にはアンインストール アプリケーションが含まれます。DevTest サーバ コンポーネントをアンインストールするには、DevTest Solutions アンインストール ウィザードを使用します。

次の手順に従ってください:

1. DevTest Solutions を停止します。
   a. すべてのユーザが DevTest ワークステーション、ポータル、DevTest コンソール、およびエンタープライズ ダッシュボードからログオフしたことを確認します。
   b. 実行中の DevTest コマンド ライン ユーティリティがないことを確認します。
   c. シミュレータ、コーディネータ、VSE、ブローカ、ポータル、レジストリ、エンタープライズ ダッシュボードの順に、実行ファイルを閉じるかサービスを停止します。
2. （オプション）主なログ ファイルが置かれているディレクトリを削除します。
   a. ログがカスタム ディレクトリに格納されている場合は、DevTest ワークステーション を開き、［ヘルプ］メニューから［DevTest ランタイム情報］を選択して、lisa.tmpdir までスクロールします。 パスは lisa.tmpdir の値です。
   b. lisatmp_*リリース番号*ディレクトリに移動します。各リリース固有のログ ディレクトリ（*lisatmp_リリース番号*）のデフォルトの場所は USER_HOME ディレクトリです。
   c. ディレクトリを右クリックし、［Delete］を選択します。確認プロンプトに対して［Yes］をクリックします。
3. ［Shared］インストール タイプをアンインストールしている場合は、手動で lisa.user.properties ファイルを削除します。デフォルトの場所は、DevTest をインストールしたユーザの USE_HOME ディレクトリです。

   重要: このファイルを削除しない場合、［Local］インストール タイプでの将来のインストールは正しくインストールされません。
4. アンインストール プロセスを開始します。
   - Windows： 以下からこのアプリケーションを起動できます。
     - Windows の［スタート］メニュー オプション

- DevTest Solutions アンインストーラ
- ［コントロール パネル］-［プログラム］-［プログラムと機能］-［プログラムのアンインストールまたは変更］ウィンドウ

- UNIX または Linux： DevTest を開き、［uninstall］をクリックして、［Run］を選択します。

［DevTest Solutions Uninstall］手順が表示されます。

5. ［次へ］をクリックします。
6. データベース、hotDeploy、lib/core、および関連ユーザ設定のフォルダを削除するには、［Delete all files］チェック ボックスをオンにします。
7. ［次へ］をクリックします。

   すべてのファイルを削除することを選択しなかった場合は、［Results of Uninstaller］手順が表示され、削除できなかったファイルのリストが示されます。
8. ［終了］をクリックします。

第 4 章: DevTest ワークステーション のインストール

DevTest ワークステーション が組み込まれた DevTest サーバ をインストールすることも、DevTest ワークステーション をスタンドアロン アプリケーションとしてインストールこともできます。

このセクションでは、DevTest ワークステーション をスタンドアロン アプリケーションとしてインストールし、設定する方法について説明します。

DevTest ワークステーション をインストールおよび設定したら、「*使用*」の「DevTest ワークステーション を開く」の手順に従ってログインできます。コンピュータに DevTest サーバ なしで DevTest ワークステーション がインストールされている場合は、リモート コンピュータで実行されているレジストリを指定します。DevTest ワークステーション インストールにはローカル レジストリが含まれません。

詳細については、「Java 環境での DevTest ワークステーション の使用 (P. 80)」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。

Windows での DevTest ワークステーション のインストール

このトピックでは、DevTest ワークステーション をデモ サーバと共に、またはデモ サーバなしで Windows 環境にインストールする方法について説明します。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

次の手順に従ってください:

1. インストーラ ファイル（devtest_win_x64.exe など）を実行します。

   ［Welcome to the DevTest Solutions Setup Wizard］が表示されます。

2. ［次へ］をクリックします。

   ［CA End User License Agreement］手順が表示されます。

3. ライセンス契約書を読み、［I accept the terms of the License Agreement］オプションを選択し、［Next］をクリックします。

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

4. DevTest Solutions の 1 つ以上のコンポーネントをインストールするフォルダを指定します。 **C:¥Program Files¥CA¥DevTestSolutions** などのスペースが含まれるディレクトリも使用できます。存在しないフォルダを指定すると、そのフォルダがセットアップ ウィザードによって作成されます。

5. ［次へ］をクリックします。

   ［Installation Type］手順が表示されます。

6. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

複数のコンピュータから複数のユーザがアクセスできる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。 共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。

［Select Components］手順が表示されます。

7. ［Server］チェック ボックスをオフにし、［Workstation］チェック ボックスがオンになっていることを確認し、［Next］をクリックします。

   インストール タイプに［Shared］を選択した場合は、以下の手順でデータ ディレクトリおよび一時ファイル ディレクトリを指定するように促されます。

8. 各手順の後に［Next］をクリックして、ディレクトリを指定します。

   ［Demo Server］手順が表示されます。

9. インストーラでデモ サーバを LISA_HOME ディレクトリに解凍する場合、［Install demo server］オプションを選択し、デモ サーバ zip ファイルの完全修飾パスを指定します。

10. ［次へ］をクリックします。

    ［Select Start Menu Folder］手順が表示されます。

11. DevTest ワークステーション を［スタート］メニューに追加するかどうか、また追加する場合は DevTest ワークステーション をその他のユーザの［スタート］メニューにも追加するかどうかを指定します。

    - すべてのユーザについてショートカットを持つ［スタート］メニュー フォルダを作成するには、すべてデフォルト設定のままにします。 必要に応じて、新しいフォルダ名を入力します。
    - スタート メニュー フォルダを作成しない場合は、「Create a Start Menu folder］チェック ボックスをオフにします。
    - ［スタート］メニュー フォルダを作成し、ショートカットの表示をお使いの［スタート］メニューに制限する方法
      - ［Create a Start Menu folder］をオンのままにします。
      - デフォルトの名前を使用するか、別の名前を入力します。
      - ［Create shortcuts for all users］チェック ボックスをオフにします。

12. ［次へ］をクリックします。

    ［Select File Associations］手順が表示されます。

13. 「*CA Application Test の使用*」で提供されるサンプル プロジェクト チュートリアルを使用する場合は、すべての拡張子を選択します。DevTest Solutions と関連付けることができるファイル拡張子には、以下のものが含まれます。

    - *.tst -- CA Application Test でテスト ケースを作成するには、この拡張子を選択します。
    - *.vsm および *.vsi -- CA Service Virtualization で仮想サービスを作成するには、この拡張子を選択します。
    - *.ste -- CA Application Test のテスト ランナーでスイートを実行するには、この拡張子を選択します。
    - *.stg -- CA Application Test でテスト ケースをステージング ドキュメントとして実行するには、この拡張子を選択します。

14. ［Install］をクリックしてインストールを開始します。

    インストールが完了すると、［Information］手順が表示されます。

15. 情報を読み、［Next］をクリックします。

    ［Completing the DevTest Solutions Setup Wizard］手順が表示されます。

16. ［終了］をクリックします。

UNIX での DevTest ワークステーション のインストール

このトピックでは、DevTest ワークステーション を UNIX または Linux 環境にインストールする方法について説明します。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

以下の手順は、GUI 版のインストーラに基づいています。 コマンドライン版のインストーラを使用するには、**-c** オプションを追加します。以下に例を示します。

```
./devtest_linux_x64.sh -c
```

注: 汎用 UNIX インストーラを使用する場合は、Java 仮想マシン（JVM） (P. 13)が同じコンピュータにあることを確認してください。 JVM のバージョンは 1.7 である必要があります。 **JAVA_HOME** 環境変数を設定することにより、特定の JVM を指定できます。 インストーラが JVM を検出できない場合、インストーラはメッセージを表示して終了します。

次の手順に従ってください:

1. ターミナル ウィンドウで、インストーラ ファイルがあるディレクトリに移動します。
2. インストーラ ファイルに実行権限があることを確認します。

   ```
   chmod 777 devtest_platform_x64.sh
   ```

   これにより、rwxrwxrwx 権限がファイルに付与されます。
3. インストーラ ファイルを実行します。アイコンをダブルクリックするか、ターミナル ウィンドウから以下のコマンドに類似したコマンドを入力します。

   ```
   ./devtest_platform_x64.sh
   ```

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

［CA End User License Agreement］手順が表示されます。

5. ライセンス契約書を読み、「I accept the terms of the License Agreement」チェック ボックスをオンにし、［Next］をクリックします。

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

6. DevTest ワークステーション のインストール先ディレクトリを指定します。スペースが含まれるディレクトリを使用しないでください。(デフォルトのパスは /opt/CA/DevTest です。)

7. ［次へ］をクリックします。

   ［Installation Type］手順が表示されます。

8. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

   複数のコンピュータから複数のユーザがアクセスできる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。 共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。

   ［Select Components］手順が表示されます。

9. ［Server］チェック ボックスをオフにし、［Workstation］チェック ボックスがオンになっていることを確認し、［Next］をクリックします。

   インストール タイプに［Shared］を選択した場合は、以下の手順でデータ ディレクトリおよび一時ファイル ディレクトリを指定するように促されます。

10. 各手順の後に［Next］をクリックして、ディレクトリを指定します。

    ［Specify Demo Server］手順が表示されます。

11. インストーラでデモ サーバを **LISA_HOME** ディレクトリに解凍する場合、［Install demo server］チェック ボックスをオンにし、デモ サーバ zip ファイルの完全修飾パスを指定します。

12. ［次へ］をクリックします。

    ［Select Additional Tasks］手順が表示されます。

13. DevTest のデスクトップ アイコンを作成しない場合は、チェック ボックスをオフにします。

14. ［Install］をクリックします。

    インストールが完了すると、［Information］手順が表示されます。

15. 情報を読み、［Next］をクリックします。

    ［Completing the DevTest Solutions Setup Wizard］手順が表示されます。

16. ［終了］をクリックします。

Mac での DevTest ワークステーション のインストール

このトピックでは、デモ サーバと共に、またはデモ サーバなしで DevTest ワークステーション を Mac にインストールする方法について説明します。

インストール手順を開始する前に、Download Center (P. 43) から以下のファイルをダウンロードします。

- ご使用のプラットフォーム用のインストーラ
- （オプション）デモ サーバ zip ファイル

次の手順に従ってください:

1. インストーラ ファイル（たとえば、devtest_osx_x64.dmg）を実行します。

   ［Welcome to the DevTest Solutions Setup Wizard］が表示されます。

2. ［次へ］をクリックします。

   ［CA End User License Agreement］手順が表示されます。

3. ライセンス契約書を読み、［I accept the terms of the License Agreement］オプションを選択し、［Next］をクリックします。

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

4. DevTest Solutions の 1 つ以上のコンポーネントをインストールするフォルダを指定します。 存在しないフォルダを指定すると、そのフォルダがセットアップ ウィザードによって作成されます。

5. ［次へ］をクリックします。

   ［Installation Type］手順が表示されます。

6. 以下のいずれかのオプションを選択し、［Next］をクリックします。

   Local

   すべての DevTest Solutions コンポーネントをローカル コンピュータの単一のディレクトリにインストールします。デフォルトでは、すべてのデータはこのディレクトリに格納されます。また、ユーザごとの個人の一時ディレクトリがあります。 ローカル インストールは、ほとんどの環境で使用される最も一般的なインストール タイプです。

   Shared

複数のコンピュータから複数のユーザがアクセスできる共有の場所にすべての DevTest Solutions コンポーネントをインストールするように、管理者によって使用されます。 データおよび一時ファイルはすべてユーザ指定のディレクトリに格納されます。 ユーザごとに個人データを持ちますが、ユーザは共通の DevTest Solutions インストールを共有します。 共有インストールでは、ユーザは DevTest Solutions プログラム ディレクトリに対する読み取りアクセス権のみが必要です。

［Select Components］手順が表示されます。

7. ［Server］チェック ボックスをオフにし、［Workstation］チェック ボックスがオンになっていることを確認し、［Next］をクリックします。

   ［Demo Server］手順が表示されます。

8. インストーラでデモ サーバを解凍する場合、［Install demo server］オプションを選択し、デモ サーバ zip ファイルの完全修飾パスを指定します。

9. ［次へ］をクリックします。

   ［Select Additional Tasks］手順が表示されます。

10. （オプション）DevTest ワークステーション デスクトップ アイコンを作成しない場合は、チェック ボックスをオフにします。

11. ［次へ］をクリックします。

    ［Select File Associations］手順が表示されます。

12. 「*CA Application Test の使用*」で提供されるサンプル プロジェクト チュートリアルを使用する場合は、すべての拡張子を選択します。 DevTest Solutions と関連付けることができるファイル拡張子には、以下のものが含まれます。

    - *.tst -- CA Application Test でテスト ケースを作成するには、この拡張子を選択します。
    - *.vsm および *.vsi -- CA Service Virtualization で仮想サービスを作成するには、この拡張子を選択します。
    - *.ste -- CA Application Test のテスト ランナーでスイートを実行するには、この拡張子を選択します。
    - *.stg -- CA Application Test でテスト ケースをステージング ドキュメントとして実行するには、この拡張子を選択します。

13. ［Install］をクリックしてインストールを開始します。

    インストールが完了すると、［Information］手順が表示されます。

14. 情報を読み、［Next］をクリックします。

    ［Completing the DevTest Solutions Setup Wizard］手順が表示されます。

15. ［終了］をクリックします。

HTTP/S プロキシ サーバの使用 - DevTest ワークステーション

プレーン HTTP プロキシ サーバまたは SSL で保護された HTTP プロキシ サーバを使用している場合は、LISA_HOME にある **local.properties** ファイルで、そのプロキシ サーバと除外するホストを定義します。

次の手順に従ってください:

1. DevTest ワークステーション がインストールされているホストにログオンします。
2. LISA_HOME に移動します。
3. local.properties が存在しない場合は、**_local.properties** をコピーし、そのコピーを local.properties（アンダースコアなし）として保存します。
4. local.properties を編集するために開き、HTTP プロキシ サーバまたは HTTPS プロキシ サーバのいずれか（プロキシ サーバがプレーン HTTP を使用しているか SSL で保護された HTTP を使用しているかによって異なる）のセクション ヘッダを見つけます。
5. FQDN または IP アドレスおよびポートによってプロキシ サーバを識別します。
   - HTTP サーバの場合は、lisa.http.webProxy.host および lisa.http.webProxy.port プロパティを使用します
   - HTTPS サーバの場合は、lisa.http.webProxy.ssl.host および lisa.http.webProxy.ssl.port プロパティを使用します
6. プロキシ サーバを通過することから除外するホストを識別します。
   - HTTP サーバの場合は、lisa.http.webProxy.nonProxyHosts プロパティを使用します
   - HTTPS サーバの場合は、lisa.http.webProxy.ssl.nonProxyHosts プロパティを使用します
7. プロパティの前のコメント記号を削除したことを確認してください。
8. ファイルを保存して終了します。

例:

以下の例の最初の 2 行は、HTTP プロキシ サーバの URL が **http://192.168.24.242:49185** であることを指定します。

3 行目は、このプロキシを通過しないホストに、localhost（127.0.0.1）のループバック アドレスと 192.168.32.255 ～ 192.168.32.0 の範囲の IP アドレスが含まれることを指定します。 除外する IP アドレスの区切り文字としてパイプ記号（|）が使用されることに注意してください。また、ワイルドカード（*）が任意の有効な値（IP アドレス ノードの有効な値は 0 ～ 255）を表すことにも注意してください。 除外するホストが標準命名規則を共有している場合は、ワイルドカード文字を FQDN およびホスト名と共に使用することもできます。

```
lisa.http.webProxy.host=192.168.24.242
lisa.http.webProxy.port=49185
lisa.http.webProxy.nonProxyHosts=127.0.0.1|192.168.32.*
```

local.properties での HTTP/S プロキシ サーバ設定

```
## ============================================
## HTTP Proxy Server
## ============================================
#lisa.http.webProxy.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.port=

## ============================================
## HTTPS Proxy Server
## ============================================
#lisa.http.webProxy.ssl.host=<machine name or ip>
##list of excluded machine names or ip addresses delimited by pipes, * wildcard
accepted <machine name or ip>[|<machine name or ip>]*
lisa.http.webProxy.ssl.nonProxyHosts=127.0.0.1
#lisa.http.webProxy.ssl.port=

## === Leave blank to use integrated NTLM authentication
#lisa.http.webProxy.host.domain= used for NTLM authentication
#lisa.http.webProxy.host.account=
#lisa.http.webProxy.host.credential=

## === Exclude simple host names from proxy use - default value is true
#lisa.http.webProxy.nonProxyHosts.excludeSimple=false

## === Preemptively send authorization information rather than waiting for a
challenge
## ===== valid values are basic or ntlm
#lisa.http.webProxy.preemptiveAuthenticationType=ntlm
```

環境設定

DevTest ドキュメントでは、**%LISA_HOME%**（Windows 用）または **$LISA_HOME**（OSX または UNIX 用）という名前のトークンが説明されています。 このトークンは、DevTest Solutions がインストールされた場所を示します。

サポート対象のすべてのオペレーティング システムで、環境変数は起動スクリプトまたはプログラムからこの名前で自動的に設定されます。

たとえば、DevTest サーバ を **C:¥DevTest_***release_number* へインストールした場合は、**%LISA_HOME%** の値になります。DevTest ワークステーション は、LISA_HOME という名前のプロパティ内のこの変数の値にもアクセスできます。

DevTest クラスパスに JAR、zip、またはディレクトリを追加するには、以下の 2 つのオプションがあります。

- 環境変数 **LISA_POST_CLASSPATH** を定義し、目的のリソースを設定します。
- **%LISA_HOME%/hotDeploy** ディレクトリに追加します。

注: 環境設定の詳細については、「*CA Application Test の使用*」の「共通のプロパティと環境変数」を参照してください。

第 5 章: デモ サーバのインストール

オプションのデモ サーバは、DevTest の機能を示すためのいくつかのアプリケーションがある JBoss 4.2.3 アプリケーション サーバです。

- **examples** プロジェクトには、デモ サーバを使用するテスト ケースが含まれます。
- 「*CA Application Test の使用*」のチュートリアルの一部はデモ サーバを使用します。

注: デモ サーバはポート 1529 を使用します。その他のアプリケーションでは、そのポートを使用できません。このポートが使用可能でない場合、デモ サーバは正常に起動しません。

次の手順に従ってください:

1. 「DevTest のインストーラのダウンロード (P. 43)」の説明に従って、デモ サーバをダウンロードします。
2. 以下のいずれかのインストール方法を選択します。
   - （推奨）DevTest Solutions （サーバまたはワークステーション）をインストールし、セットアップ ウィザードで同じインストール ディレクトリ（**LISA_HOME**）にファイルを解凍します。セットアップ ウィザードは、デモ サーバのデスクトップ アイコンも作成します。
   - コンピュータ上で **DevTestDemoServer.zip** ファイルを解凍します（Windows では、このファイルは ダウンロード フォルダにダウンロードされます）。**lisa-demo-server** フォルダに移動し、README ファイルに記載されている手順に従います。 README ファイルには、デモ サーバを起動するためのプラットフォームに固有の手順が含まれます。
3. 2 つ目の方法を選択する場合は、以下の情報に注意してください。
   - システムには、個別に Java 7 をインストールしておく必要があります。
   - **JAVA_HOME** 環境変数を設定します。この変数は、JBoss が JSP ファイルをコンパイルおよび実行するために必要です。
   - デスクトップまたはスペースが含まれるパスに JBoss サーバ ディレクトリを配置しないでください。 ディレクトリへのパスにスペースがある場合、JBoss は JSP をコンパイルできません。

注:

- コマンド ラインからデモ サーバを起動するには、**LISA_HOME¥DemoServer¥lisa-demo-server** ディレクトリに移動し、お使いのオペレーティング システム用のスクリプトを起動します。
  - （Windows）**start-windows.bat**
  - （UNIX または Linux）**start-unix-linux.sh**
  - （OS/X）**start-osx.command**
- UNIX または Linux でデモ サーバを実行するには、**/bin/bash** シェルを使用します。
- デモ サーバはデフォルトで DevTest Java エージェントを実行し、CA Continuous Application Insight にできるだけ多くの情報をレポートします。 このレポートをオフにするには、**-noagent** フラグを使用します。ヒープ/スタック情報のレポートのみをオフにするには、**-noheapss** フラグを使用します。
- ネイティブ エージェント バイナリが、広く使用される UNIX および Linux ディストリビューションに存在する場合、デモ サーバはネイティブ エージェントを代わりに起動します。 この場合、デモ サーバに Pure Java エージェントを使用させるには、起動コマンドに **--javaagent** パラメータを渡します。 ネイティブ エージェント バイナリは、Java 1.4 以前での使用のみを意図しています。
- デモ サーバが初めて起動されると、デモ サーバ データベースが作成されます。 このデータベースは、**LISA_HOME¥DemoServer¥lisa-demo-server¥jboss¥server¥default¥data¥lisa-demo-server.db** ディレクトリにあります。
- デモ サーバが起動したら、ブラウザを使用してポート 8080 でサーバにアクセスできます。

第 6 章: DevTest Solution のインストールの確認

DevTest Solution のインストールが正常終了したことを確認するには、DevTest サーバを起動し、各ユーザ インターフェースを開いてログインします。

このセクションには、以下のトピックが含まれています。

DevTest の起動および UI へのログイン

次の手順に従ってください:

1. サーバ コンポーネントの起動 (P. 104)。
2. ユーザ インターフェースにログインする準備をします。以下を行うことができます。
   - 標準 DevTest スーパー ユーザとしてのログイン (P. 108)
   - 自分用にスーパー ユーザ ロールが付与された DevTest ユーザを作成 (P. 110) し、自分の認証情報でログインします。
3. 各ユーザ インターフェースにアクセス (P. 112) してログインします。UI を確認します。
4. （オプション） DevTest サーバ ディレクトリを確認します。

DevTest プロセスまたはサービスの起動

このセクションでは、使用可能なすべてのコンポーネントを持つ DevTest サーバ を起動するプロセスについて説明します。 すべてのコンポーネントが確実に起動するように、示されているシーケンスを使用してください。起動プロセスの後に、DevTest プロセスまたはサービスを開始できるいくつかの方法を説明します。

注: このプロセスは、「DevTest サーバ コンポーネントについて (P. 26)」で説明します。

起動順序シーケンス

以下のシーケンスで DevTest サーバ プロセス（またはサービス）を起動します （［スタート］メニュー ショートカットが示されます）。

次の手順に従ってください：

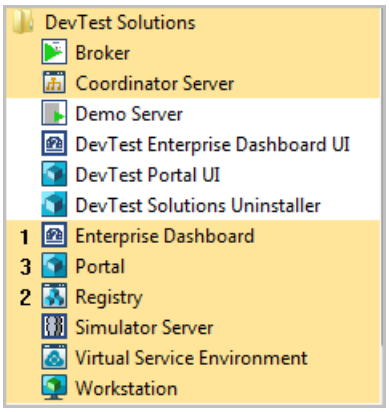
1. エンタープライズ ダッシュボード サーバを起動します。
2. 各レジストリを起動します。
3. ポータルを起動します。
4. 以下のコンポーネントを任意の順番で起動します。
   - ブローカ
   - コーディネータ サーバ
   - 各コーディネータに関連付けられたシミュレータ サーバ
   - 仮想サービス環境
5. DevTest サーバ で使用している場合
   - ワークステーション
   - デモ サーバ

注: サーバ コンポーネントをシャットダウンするには、逆の順番を使用します。

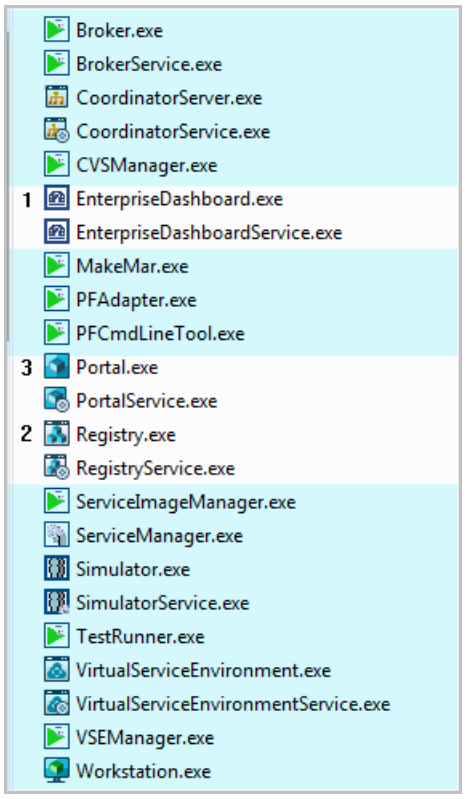
DevTest サーバ を起動する方法

以下のいずれかの方法で DevTest サーバ プロセスにアクセスします。

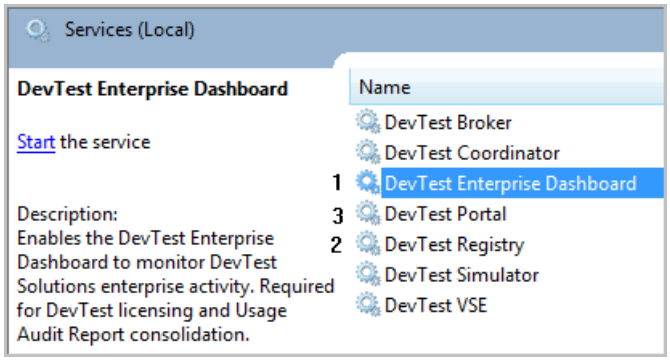
- （Windows）［スタート］メニューをクリックし、［DevTest Solutions］を展開します。 起動順序シーケンスでプロセスを起動します。


DevTest Solutions
Broker
Coordinator Server
Demo Server
DevTest Enterprise Dashboard UI
DevTest Portal UI
DevTest Solutions Uninstaller
1 Enterprise Dashboard
3 Portal
2 Registry
Simulator Server
Virtual Service Environment
Workstation

- LISA_HOME¥bin フォルダに移動します。 起動順序シーケンスで実行ファイルを起動します。

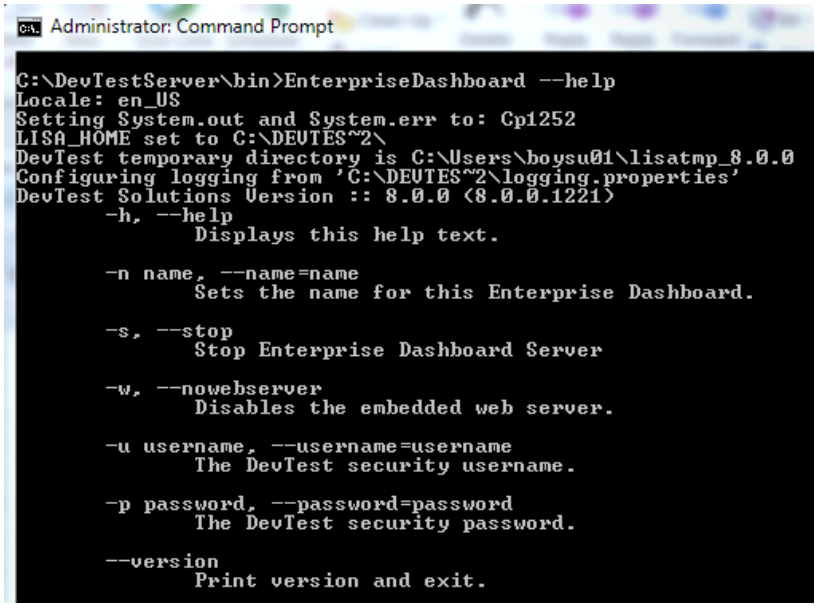


- （Windows） ［サービス］に移動し、起動順序シーケンスでサービスを起動します。



- 管理者としてコマンド プロンプトを開くか、ターミナル ウィンドウを開きます。 LISA_HOME¥bin に移動し、起動順序シーケンスで各プロセスを起動するコマンドを入力します。 それぞれのサービスを起動したり、関連する vmoptions ファイルをきどうしたりすることもできます。

  コマンド名の後に「--help」を続けて入力することにより、ヘルプを表示します。 以下に例を示します。

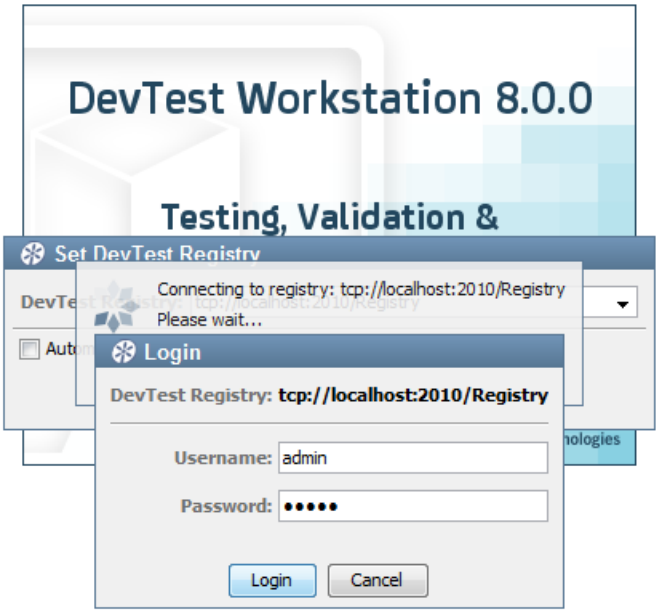

Administrator: Command Prompt
C:\DevTestServer\bin>EnterpriseDashboard --help
Locale: en_US
Setting System.out and System.err to: Cp1252
LISA_HOME set to C:\DEVTES~2\
DevTest temporary directory is C:\Users\boysu01\lisatmp_8.0.0
Configuring logging from 'C:\DEVTES~2\logging.properties'
DevTest Solutions Version :: 8.0.0 (8.0.0.1221)
-h, --help
Displays this help text.
-n name, --name=name
Sets the name for this Enterprise Dashboard.
-s, --stop
Stop Enterprise Dashboard Server
-w, --nowebserver
Disables the embedded web server.
-u username, --username=username
The DevTest security username.
-p password, --password=password
The DevTest security password.
--version
Print version and exit.

標準スーパー ユーザとしてのログイン

重要：ACL（Access Control List）は必須です。 DevTest Solutions は、最初に有効な名前およびパスワードを使用してログインしないと使用できません。

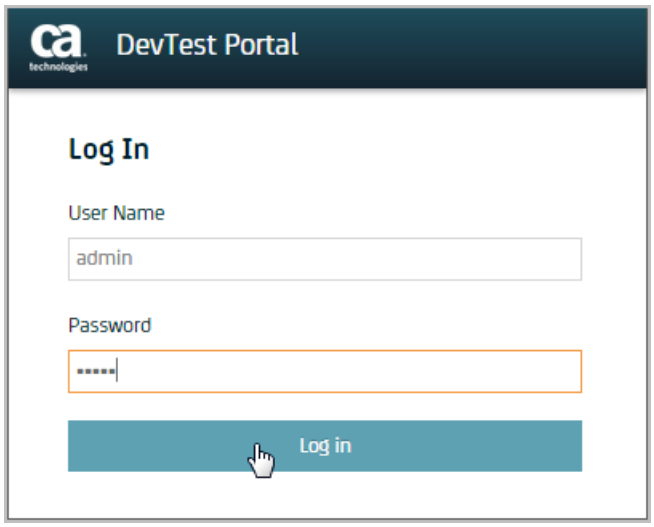
セキュリティの設定には、ロール ベースのユーザ アカウントを作成と、LDAP または定義した認証情報による ACL の指定が含まれます。LDAP を認証に使用することを計画している場合、「スーパー ユーザ ロールが付与されたユーザの作成 (P. 110)」で説明されているとおり、自分のユーザ名およびパスワードは追加しません。LDAP の設定前に UI にアクセスするには、スーパー ユーザ ロールで定義される標準ユーザを使用します。

DevTest ワークステーション にアクセスすると、以下の図のようなログイン ダイアログ ボックスが表示されます。



ログインするには、［ユーザ名］フィールドに「*admin*」と入力し、［パスワード］フィールドに「*admin*」と入力して、［ログイン］をクリックします。 DevTest ワークステーション が開きます。

DevTest ポータルを参照すると、ログイン領域は以下のようになります。



ログインするには、［ユーザ名］フィールドに「**admin**」と入力し、［パスワード］フィールドに「**admin**」と入力して、［ログイン］をクリックします。

DevTest コンソールでは、同様のログイン ダイアログ ボックスが表示されます。 同じ標準ユーザでそのブラウザにログインできます。

スーパー ユーザ ロールが付与されたユーザの作成

ユーザ自身に DevTest Solutions へのフル アクセスを付与します。

次の手順に従ってください:

1. サーバ コンソールを参照します。

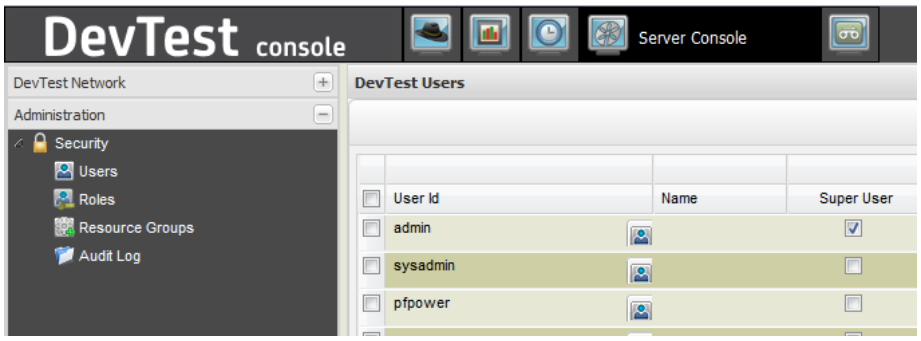
   `http://localhost:1505`

2. 標準管理者ユーザとしてログインします。

   a. ユーザ名に「*admin*」と入力します。

   b. パスワードに「*admin*」と入力します。

3. 左ナビゲーション バーの［管理］ペインを展開します。スーパー ユーザは、［セキュリティ］エリアで、ユーザを認証するための認証情報を入力し、ロールの割り当てを介して機能にアクセスする権限を付与します。



4. 忘れないパスワードを使用して、自分のユーザ アカウントを作成します。 DevTest Solutions にフル アクセスするための、スーパー ユーザ ロールを割り当てます。

   a. 右側のパネルの下部にある［ユーザの追加］をクリックします。

   b. ［ユーザ ID］フィールドに、英数字、ハイフン（-）、アンダースコア（_）、ピリオド（.）、およびアンパサンド（@）文字の任意の組み合わせからなる一意の ID を入力します。

   c. ［パスワード］フィールドに、英数字、ハイフン（-）、アンダースコア（_）、およびアンパサンド（@）文字の任意の組み合わせからなるパスワードを入力します。

   d. ［パスワードの再入力］フィールドに、もう一度パスワードを入力します。

   e. ［名前］フィールドに、ユーザ名（英数字、ハイフン（-）、アンダースコア（_）およびスペース文字）を入力します。

   f. ［ユーザのロール］領域で、［スーパー ユーザ］を選択します。

   g. ［ユーザの追加］をクリックします。

   h. ［OK］をクリックします。

任意の DevTest ユーザ インターフェースまたはコマンド ライン インターフェースにアクセスし、定義したユーザ ID およびパスワードでログインできるようになりました。

DevTest ユーザ インターフェースへのアクセス

使用するエンタープライズ ダッシュボード、レジストリ、およびポータルは、すべての UI に対して実行される必要があります。「Start the Server Components (P. 104)」を参照してください。

ここでの説明に従って UI にアクセスします。 その後、標準のスーパーユーザとしてログインする (P. 108)か、スーパー ユーザ ロールを使用したユーザがユーザ自身を定義した (P. 110)ユーザ独自の認証情報でログインします。

デモ サーバ

DevTest Solutions に関して学習するには、ブラウザでローカル ホスト上のデモ サーバにアクセスし、「*CA Application Test の使用*」のチュートリアルに従います。

- ブラウザでデモ サーバにアクセスして、ログインします。

  `http://localhost:8080/lisabank`

- （Windows）［スタート］メニューの［DevTest Solutions］を展開して、［Demo Server］を選択します。

エンタープライズ ダッシュボード

エンタープライズ ダッシュボードに接続されているレジストリのリストを表示する方法

- ブラウザでエンタープライズ ダッシュボード UI にアクセスして、ログインします。

  `http://hostname:1506`

- （Windows）エンタープライズ ダッシュボードがインストールされているサーバにログインします。 ［スタート］メニューの［DevTest Solutions］を展開して、［DevTest Enterprise Dashboard UI］を選択します。

Portal

CA Continuous Application Insight を使用する前に、DevTest Java エージェントをインストールします。 ブラウザで以下の URL にアクセスし、DevTest ポータルにログインして、CAI にアクセスします。 実行中のレジストリを持つコンピュータのホスト名を指定します。 「*CAI の使用*」を参照してください。

CAI は、ポータルが実行中である必要があります。 UI は、ブローカが実行中である必要があります。 CA Application Test および CA Service Virtualization は、一部の機能のためにポータルを使用します。 「*CA Application Test の使用*」および「*CA Service Virtualization の使用*」を参照してください。

- ブラウザでポータル UI にアクセスして、ログインします。

  `http://hostname:1507/devtest`

- （Windows）［スタート］メニューの［DevTest Solutions］を展開して、［DevTest Portal UI］を選択します。

ワークステーション

ローカル ホスト上の DevTest ワークステーション を起動します。ワークステーションでは、テストを作成および編集したり、テストのステージングも行います。 代替方法には、TestRunner、junitlisa Ant タスクなどがあります。 「*CA Application Test の使用*」ドキュメントを参照してください。 UI は、コーディネータ サーバおよびシミュレータ サーバが実行中である必要があります。

- （Windows）［スタート］メニューの［DevTest Solutions］を展開して、［ワークステーション］を選択し、レジストリを指定して、ログインします。
- LISA_HOME¥bin に移動し、Workstation.exe を実行します。

DevTest コンソール

このコンソールには、サーバ コンソール（VSE など）、CVS ダッシュボード、レポート、および VSEasy へのリンクが含まれています。 実行中のレジストリを持つコンピュータのホスト名を指定します。 サーバ コンソールの［Administrative］タブの使用については、「*DevTest Solutions の管理*」を参照してください。その他の情報については、「*CA Service Virtualization の使用*」を参照してください。 CA Service Virtualization の UI は、VSE が実行中である必要があります。

- DevTest コンソールにアクセスして、ログインします。

  `http://hostname:1505`

- ワークステーションを開き、［サーバ コンソール］ツールバー ボタンをクリックします。

第 7 章: 統合ツールのインストール

このセクションでは、DevTest と一緒に使用できるサードパーティ製ツールをインストールおよび設定する方法について説明します。

このセクションには、以下のトピックが含まれています。

パフォーマンス モニタ（Perfmon）のインストール

パフォーマンス モニタ（Perfmon）は、ローカルまたはリモート システムのパフォーマンスのモニタを行うユーティリティです。 Perfmon は、パフォーマンス カウンタを使用してシステム パフォーマンスをモニタする方法を示します。

Perfmon を使用して Windows システムのパフォーマンスをモニタする方法

- .NET Framework 2.0 との互換性が必要です。
- コマンド プロンプトから、**LISA_HOME¥bin** ディレクトリにある **setup-wperfmon.bat** ファイルを実行します。
- Windows では、コマンド プロンプトは、「管理者として実行」する必要があります。

注: Windows 2012 を実行している場合、.NET Framework 3.5 のインストールにより .NET Framework 2.0 との互換性を実現できます。

さらに、以下を確認します。

- ユーザ ID が両方のコンピュータで同じである。
- ユーザ ID が両方のコンピュータで管理者権限を持つ。
- ファイルおよびプリンタの共有がオンになっている。
- 簡易ファイルの共有がオフになっている。
- デフォルトの C$ 共有、ADMIN$ 共有、または両方が有効である。

モニタ対象のコンピュータ上のファイアウォールを停止する必要がある場合があります。

リモート モニタが動作していることを確認する方法

1. ［スタート］-［コントロール パネル］-［管理ツール］-［パフォーマンス］を選択します。
2. 監視するコンピュータにモニタを追加します。

DevTest と Windows は、リモート モニタを実行するのに同じテクノロジを使用します。 Windows のモニタが動作する場合、通常 DevTest のモニタは動作します。

Perfmon を使用して DevTest のメトリックを収集するには、「*CA Application Test の使用*」の「Windows Perfmon メトリック」を参照してください。

SNMP のインストールおよび設定

SNMP （Simple Network Management Protocol）の Microsoft Windows への実装は、以下のタスクに使用されます。

- リモート デバイスの設定
- ネットワーク パフォーマンスのモニタ
- ネットワーク使用状況の監査
- ネットワーク障害または不適切なアクセスの検出

Windows での SNMP のサポート

Windows 7 は、SNMP 要求に応答し、トラップを送信できるエージェントを提供します。

- Microsoft SNMP エージェントのインストール（P. 118）
- Microsoft SNMP エージェントの設定（P. 120）

UNIX での SNMP のサポート

SNMP のサポートはオペレーティング システム ベンダーから入手可能です。または、Net-SNMP オープン ソース SNMP パッケージを利用できます。インストールおよび設定の詳細については、付属のドキュメントを参照してください。

Microsoft SNMP エージェントのインストール

Windows 7 での Microsoft SNMP エージェントのインストール

Windows 7 オペレーティング システムで Microsoft SNMP エージェントをインストールできます。

次の手順に従ってください:

1. ［スタート］-［コントロール パネル］をクリックします。
2. ［プログラム］をクリックします。
3. ［プログラムと機能］で［Windows の機能の有効化または無効化］を選択します。

   ［Windows の機能］ウィンドウが表示されます。
4. ［簡易ネットワーク管理プロトコル（SNMP）］チェック ボックスをオンにして、［OK］をクリックします。
5. SNMP がインストールされるまで待機します。

以前の Windows バージョンでの Microsoft SNMP エージェントのインストール

次の手順に従ってください:

1. Windows の［コントロール パネル］を開きます。
2. ［プログラムの追加と削除］アイコンをダブルクリックします。

   ［プログラムの追加と削除］ウィンドウが表示されます。
3. ウィンドウの左側で［Windows コンポーネントの 追加と削除］をクリックします。

   Windows コンポーネント ウィザードが表示されます。
4. ［コンポーネント］リスト内の［管理とモニタ ツール］を選択し、［詳細］をクリックします。

   ［管理とモニタ ツール］ウィンドウが表示されます。
5. ［管理とモニタ ツールのサブコンポーネント］リストから［簡易ネットワーク管理プロトコル（SNMP）］を選択し、［OK］をクリックします。
6. ［次へ］をクリックします。

   Windows コンポーネント ウィザードが Microsoft SNMP エージェントをインストールします。
7. 完了したら、［完了］をクリックします。

Microsoft SNMP エージェントの設定

この手順では、SNMP エージェントを設定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. Windows の［コントロール パネル］を開きます。
2. ［管理ツール］をダブルクリックします。

   ［管理ツール］ウィンドウが表示されます。
3. ［サービス］をダブルクリックします。

   ［サービス］ウィンドウが表示されます。
4. SNMP サービスをダブルクリックします。

   ［SNMP サービスのプロパティ］ダイアログ ボックスが表示されます。
5. ［全般］タブの［スタートアップの種類］を［**自動**］に変更します。

   この操作により、システムの起動時に Microsoft SNMP エージェントを起動するように SNMP サービスが設定されます。
6. ［トラップ］タブをクリックします。
7. ［コミュニティ名］フィールドに、コンピュータがトラップ メッセージを送信するコミュニティ名を入力します。
8. ［一覧に追加］をクリックします。
9. ［適用］をクリックし、［OK］をクリックします。
10. ［OK］をクリックします。

Windows SNMP を使用してメトリックを収集するには、「*CA Application Test の使用*」を参照してください。

TCPMon の実行

TCPMon は、TCP ベースの会話で渡されるメッセージをモニタできるユーティリティです。

TCPMon は以下の要素で構成されています。

- Windows の場合： .jar ファイル、.bat ファイル
- UNIX の場合： シェル スクリプト

TCPMon を実行する方法

Windows 上で .bat ファイルをダブルクリックします。または UNIX でシェル スクリプトを実行します。

tcpmon.bat ファイルは **LISA_HOME¥bin** ディレクトリにあります。TCPMon の最新のバージョンは、http://ws.apache.org/commons/tcpmon/ から取得できます。

注: このセクションでは、Apache の TCPMon バージョンについて説明します。この TCPMon バージョンには、DevTest で配布される TCPMon バージョンで使用できない［**Sender**］タブが含まれます。

詳細:

明示的な中継としての TCPMon の使用 (P. 122)
要求送信者としての TCPMon の使用 (P. 123)

明示的な中継としての TCPMon の使用

TCPMon の最も一般的な使用パターンは、「中継」としての使用することです。 クライアントは、メッセージを監視するために元のエンドポイントではなく中継を指す必要があるため、この使用は明示的なものとして指定されます。

この設定で TCPMon を起動する方法

1. TCPMonitor の［Admin］タブで、リスン ポート、ホスト名、およびリスナのポートを入力します。
2. メッセージを参照できる新しいタブ（ポート 8000）を開くには、［Add］をクリックします。

これで、要求は、元のエンドポイントではなく TCPMon のリスナ ポートを指すようになります。

localhost:8080 で送受信されるすべてのメッセージがモニタされます。

- リスナをポート 8000 に設定しました。これには、ローカル コンピュータの任意の未使用ポートを使用できます。
- ホストが **localhost**、ポートが 8080 のリスナを追加しました。

- ブラウザが localhost:8080 の代わりに **localhost:8000** を指すようにします。

要求送信者としての TCPMon の使用

TCPMon は Web サービスの要求送信者として使用することもできます。

- SOAP 要求メッセージは［Sender］ウィンドウへ貼り付けて、サーバに直接送信できます。
- Web サービス エンドポイントは［connection Endpoint］テキスト ボックスに入力されます。

HP ALM - Quality Center プラグインのインストール

HP ALM - Quality Center プラグインを使用すると、HP ALM - Quality Center スイートから Quality Center テストとして DevTest テスト ケースをロードおよび実行できます。 DevTest テストを Quality Center にインポートして、実行できます。 この統合によって、DevTest テストを活用しながら、すべての Quality Center 機能を利用することができます。Quality Center に DevTest テスト ケースをロードすることによって、DevTest テストをリアルタイムで実行できます。また、テスト中のシステムからテスト結果の完全なキャプチャおよび DevTest コールバックも取得します。 DevTest テストは Quality Center のワークフローで実行可能で、テスト プロセスのコンテキストおよびステータスを保持するために結果をレポートします。

以下のソフトウェアをインストールする必要があります。

- DevTest 7.0 以降
- DevTest HP ALM - Quality Center プラグイン
- HP Quality Center 10 または 11
- HP Quality Center クライアント コンポーネント
- .NET 2.0 ランタイム

DevTest がインストールされているコンピュータにこのプラグインをインストールする場合、**LisaQCRunner.exe** ファイルは **LISA_HOME¥bin** ディレクトリに追加されます。 **LisaQCRunner.exe** はインストール プロセスの一部として登録される COM サーバです。DevTest は Quality Center VAPI-XP インターフェースを使用して呼び出されます。 VAPI-XP スクリプトは、COM オブジェクトのインスタンスを作成します。 これは実行されるテストをステージングし、テスト結果をリスンします。 最後に、COM オブジェクトは結果を取得し、DevTest からの結果で Quality Center インスタンスを更新します。

以下のファイルもインストールされます。

- Script_Template_js.txt
- Script_Template_vbs.txt

これらのファイルには、Quality Center で DevTest テストを設定するプロセス中に使用されるスクリプトが含まれます。

注: HP ALM - Quality Center の使用方法については、「*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. クライアント コンピュータに .NET ランタイムがインストールされていることを確認します。
2. HP Quality Center クライアント コンポーネントをインストールします。
3. DevTest をインストールします。
4. **LISA_HOME¥addons¥qc** ディレクトリに移動し、**td_plugin.exe** ファイルを実行します。
5. ウィザードの手順を完了します。

IBM Rational Quality Manager のインストール

IBM Rational Quality Manager は、テスト計画、ワークフロー制御、トラッキング、メトリック レポート用の Web ベースの一元化されたテスト管理環境です。 Rational Quality Manager は多くの方法で拡張可能です。これには、「コネクタ」（プラグイン）を使用して RQM と外部システムをブリッジする機能も含まれます。DevTest RQM プラグインを使用すると、DevTest テスト ケースを再利用または作成して RQM に関連付けることができます。 その後、DevTest テスト ケースを RQM インターフェースから実行できます。テスト ランの結果は RQM のテストの実行およびレポート履歴に集約されます。

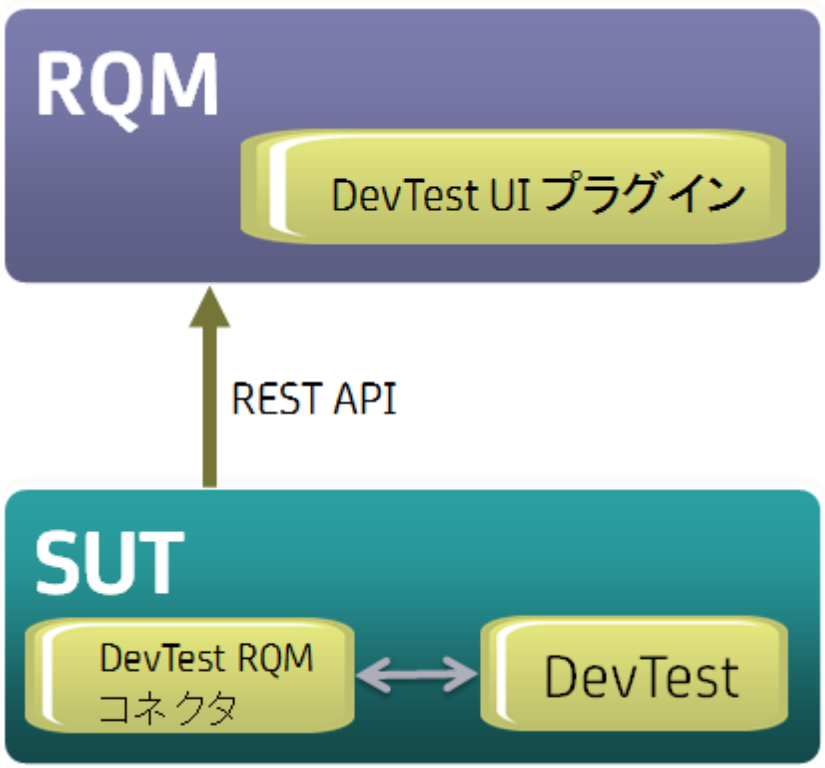
このセクションには、以下のトピックが含まれます。

実装

DevTest RQM ソリューション全体は、以下の 2 つのコンポーネントとして実装されます。

- DevTest RQM コネクタ
- Web UI 拡張

Web UI は、標準の UI の拡張を可能にします。 この UI を使用すると、必要なパラメータを DevTest と RQM の間で渡したり、表示したりすることができます。 DevTest RQM コネクタは、RQM がスケジュールする作業タスクに応答するために使用されます。 UI が収集するタスク情報はコネクタに渡され、コネクタは必要なパラメータを使用してテスト ランナーを呼び出します。



コネクタは TestRunner -html スイッチを利用し、後ほど RQM にアップロードされる HTML 出力を生成するために使用します。 テストが完了した後、テスト結果は RQM にアップロードされ、テスト ランと関連付けられます。その後、ユーザは RQM インターフェースを使用して結果を確認できます。

アダプタ UI のインストール

次の手順に従ってください:

1. **LISA_HOME¥addons** ディレクトリに移動し、**rqm-adapter.zip** という名前の zip ファイルを見つけます。
2. ファイルを解凍し、**lisa-adapter-ui** フォルダ内の **com.itko.lisa.integration.ibm.rqm.update.site.zip** ファイルを見つけます。
3. RQM サーバで、**com.itko.lisa.integration.ibm.rqm.update.site.zip** ファイルを解凍します。
4. **com.itko.lisa.integration.ibm.rqm.adapter.web.update.ini** ファイルを開き、参照を更新して更新サイトの場所を指すようにします。
5. 変更した .ini ファイルを **<RQM install root>¥server¥server¥conf¥jazz¥provision_profiles¥** にコピーします。
6. Rational Quality Manager Server Reset サービスを使用してサーバ設定を更新します。 リセット ユーティリティは、以下の URL にあります。

   [https://<hostname>:<portnumber>/jazz/admin?internal#action=com.ibm.team.repository.admin.serverReset|https://%3chostname%3e%3cportnumber%3e]

   このサービスは、IBM Rational Quality Manager 管理コンソールにログインするようにユーザに促します。ユーザ ID には管理者権限が必要です。
7. RQM サーバを停止します。
8. Rational Quality Manager サーバを再起動します。

コマンドライン アダプタの実行

次の手順に従ってください:

1. **LISA_HOME¥addons** ディレクトリに移動し、**rqm-adapter.zip** という名前の zip ファイルを見つけます。
2. ファイルを解凍し、フォルダ内の子フォルダ **lisa-adapter2.0** を見つけます。
3. コマンドラインから、アダプタを起動します。 以下のパラメータが必要です。
   - -repository [https://*rqmserver*:*port*/jazz|https://%3crqmserver%3e%3cport%3e]
   - [https://%3crqmserver%3e%3cport%3e]-user *userid*
   - -password *password*
   - -adapter *adapter*(com.itko.lisa.integration.ibm.rqm)
   - -adapterName *adapterName* (LISA RQM Adapter)## -LISA_HOME *lisaHomePath* (d¥:/lisa)
   - [-projectArea *project area* (Quality%20Manager)]
   - [-sleepTime *sleep time*(5)]

一般的な使用ワークフロー

RQM でテスト スクリプトを作成します。

1. テスト スクリプトと DevTest テスト ケース、ステージング ドキュメント、およびオプションの設定ファイルを関連付けます。
2. テスト ケースと 1 つ以上のテスト スクリプトを関連付けます。これで、テスト ケースを実行する準備が整います。
3. テスト ケースの実行を選択し、実行中のアダプタを選択して、［OK］をクリックします。
4. サーバは、RQM コネクタ プラグインを介してテスト中のシステムと通信し、このシステムは DevTest を呼び出します。
5. テストが実行されるまで待機し、実行結果を RQM で確認します。

CA APM との統合の設定

DevTest Solutions には、ワークステーションとサーバの 2 つのバージョンがあります。 ワークステーション バージョンは、サーバ コンポーネントの DevTest テスト ケースと仮想シミュレーションを作成および確認するように設計されています。 ワークステーションは単一の Java プロセス（DevTest ワークステーション）で実行されるため、テストとシミュレーションのサイズには制限があります。 サーバ バージョンは、必要に応じて、多くの機能を複数のシステムで実行できる 1 つ以上の個別の Java プロセスに分割します。

DevTest の Java プロセスはすべて、CA Application Performance Management（APM） Introscope エージェントでインスツルメントできます。 このインスツルメンテーションによって、プロセスの状態を説明する一般的なメトリックを Introscope Enterprise Manager にレポートできます。 これらのメトリックには以下のものが含まれます。

- CPU とメモリの使用率
- ガベージ コレクション操作
- ソケットおよびバックエンド データベースの使用

また、DevTest TestEvent トレーサは、DevTest コーディネータをインスツルメントして DevTest テスト ケースの実行を説明する一連のメトリックを生成できます。 このインスツルメンテーションは、コーディネータが個別のプロセスとして実行されるか、または DevTest ワークステーションで実行されるかにかかわらず使用可能です。 DevTest ワークステーションは Windows ベース UI による対話型プロセスです。 DevTest ワークステーション 以外のすべての DevTest プロセスは、通常の Windows プロセスまたは Windows サービスとして実行できます。 両方の種類のプロセスを Introscope エージェントでインスツルメントできます。

このセクションには、以下のトピックが含まれます。

DevTest プロセスのインスツルメント

LISA_HOME¥bin ディレクトリには、各 DevTest プロセスの Windows 実行ファイルが含まれています。 このフォルダには、通常の Windows プロセスおよび Windows サービスとして実行できるプロセスの個別のバージョンが含まれます。 以下に、プロセスおよびその実行可能ファイル名のリストを示します。

DevTest プロセス名	プロセス タイプ	実行可能ファイル名
コーディネータ	通常	CoordinatorServer.exe
コーディネータ	サービス	CoordinatorService.exe
エンタープライズ ダッシュボード	通常	EnterpriseDashboard.exe
エンタープライズ ダッシュボード	サービス	EnterpriseDashboardService.exe
Portal	通常	Portal.exe
Portal	サービス	PortalService.exe
Registry	通常	Registry.exe
Registry	サービス	RegistryService.exe
シミュレータ	通常	Simulator.exe
シミュレータ	サービス	SimulatorService.exe

仮想サービス環境	通常	VirtualServiceEnvironment.exe
仮想サービス環境	サービス	VirtualServiceEnvironmentService.exe
ワークステーション	通常	Workstation.exe

注: このリストには、Introscope インスツルメンテーションに関係しない bin ディレクトリ内の実行ファイルは含まれていません。

Introscope エージェントで DevTest プロセスをインスツルメントするには、実行可能ファイルと同じ名前と拡張子 **.vmoptions** でファイルを作成します。たとえば、DevTest ワークステーション をインスツルメントするには、DevTest サーバ インストール ディレクトリの **bin** ディレクトリに **Workstation.vmoptions** ファイルを作成します。.vmoptions ファイルには以下の 2 行が含まれます。

```
-javaagent:<AGENT_HOME>/Agent.jar
-Dcom.wily.introscope.agentProfile=<AGENT_HOME>/core/config/IntroscopeAgent.profile
```

<AGENT_HOME> は DevTest 固有の Introscope エージェント インストールへのパスです。 通常、このパスは絶対パスですが、DevTest プロセスが実行されるカレント ディレクトリを基準にした相対パスも可能です。

Introscope エージェントがデフォルトの DevTest エージェント以外の名前で レポートするようにするには、**Java com.wily.introscope.agent.agentName** システム プロパティを定義する行を追加します。たとえば、エージェント DevTest ワークステーション 6.0.5.87 Agent を呼び出すには、以下の行を追加します。

```
-Dcom.wily.introscope.agent.agentName=DevTest Workstation 6.0.5.87 Agent
```

必要に応じてその他の JVM コマンドライン オプションまたはシステム プロパティを設定するために、行を追加します。 オプションはそれぞれ個別の行で指定する必要があります。

DevTest インストール、または DevTest プロセスに対して、固有の Introscope エージェント インストールを使用することができます。それらの個別のエージェント インストールで **IntroscopeAgent.profile** の内容を変えることにより、さまざまなエージェント設定を作成できます。または、単一のエージェント インストールで複数のエージェント プロファイルを作成するには、各 **.vmoptions** ファイルで定義されている **com.wily.introscope.agentProfile** システム プロパティ値を変更します。 この方法により、異なるレベルの追跡またはログ記録（さらには異なる Introscope Enterprise Manager）を使用して異なる DevTest プロセスを監視することが可能になります。

たとえば、サイトに、Windows サービスとして実行されるコーディネータノードを使用する DevTest の実稼働使用を含めることができます。このサイトには、通常の Windows プロセスとして実行されるコーディネータを使用するテスト/試験使用を含めることもできます。 このコーディネータは DevTest ワークステーション に組み込むこともできます。 実稼働メトリックは実稼働 EM にレポートされ、テスト/試験メトリックはテスト EM にレポートされます。 これは、複数のプロファイルを使用し、関連する .vmoptions ファイル内の正しいプロファイルを指定することにより行うことができます。 また、プロファイルで定義されている **introscope.agent.enterprisemanager.connectionorder** および **introscope.agent.enterprisemanager.transport.*** プロパティの値を上書きすることにより行うこともできます。これには .vmoptions ファイルに同じ名前の Java システム プロパティを設定します。Introscope エージェント設定オプションの詳細については、「Introscope Java Agent ガイド」を参照してください。

Introscope エージェント ファイル

以下の表に、DevTest 固有の Introscope エージェント ファイルを示します。これらのファイルの内容、および DevTest のインスツルメンテーションをカスタマイズするためにそれらを変更する方法の詳細については、「トレーサ設定 (P. 138)」を参照してください。

ファイル	<AGENT_HOME> に相対的な位置	内容
Lisa.jar	core¥ext	DevTest 固有のトレーサ、名前フォーマッタ。
IntroscopeAgent.profile	core¥config	DevTest 固有のエージェント プロファイル。デフォルトでは、**lisa-typical.pbl** を参照して DevTest プロセスをインスツルメントするために使用されるトレーサのセットを定義します。
lisa.pbd	core¥config	トレーサおよびインスツルメンテーション ポイントの定義。
lisa-full.pbl	core¥config	「完全」な DevTest インスツルメンテーションに使用される PBD ファイルのリスト。**lisa-toggles-full.pbd** を参照します。

lisa-toggles-full.pbd	core¥config	「完全」モードでインスツルメントする場合、特定のインスツルメンテーション機能をオンするために使用されるトグルのリスト。
lisa-typical.pbl	core¥config	「標準」の DevTest インスツルメンテーションに使用される PBD ファイルのリスト。**lisa-toggles-typical.pbd** を参照します。
lisa-toggles-typical.pbd	core¥config	「標準」モードでインスツルメントする場合、特定のインスツルメンテーション機能をオンするために使用されるトグルのリスト。

トレーサによってレポートされるメトリック

DevTest トレーサは、以下の 4 つのレベルの 1 つ以上でメトリックを生成します。

- DevTest
- テスト ケース
- シミュレータ
- テスト ステップ

シミュレータおよびテスト ステップ レベルで作成されるメトリックも、ユーザ設定可能な最大レベル（デフォルトではテスト ケース）に集約されます。テスト ケースが 2 つのシミュレータで実行され、2 つのテスト ステップが含まれる場合、「Responses Per Interval」メトリックは各シミュレータの各テスト ステップに対して生成されます。 その後、（設定に応じて）値が集約され、各シミュレータの「Responses Per Interval」メトリックが生成されます。 最後に、（設定に応じて）値が再度集約され、テスト ケースの「Responses Per Interval」メトリックが生成されます。 どのメトリックもロールアップできる最高レベルはテスト ケース レベルです。 そのため、以下の表には、DevTest レベルで生成されるメトリックのみを DevTest のレベルと共に示します。

各エージェント レポート間隔（15 秒）における関連するタイプのすべてのイベントからのデータは、以下のように組み合わせられます。

メトリック	レベル	Source
平均応答時間（ミリ秒）	テスト ステップ	すべての「ステップ応答時間」イベント（id 18）の「詳細な説明」値の平均（整数に変換）
間隔ごとの応答数	テスト ステップ	すべての「ステップ応答時間」イベント（id 18）の数
間隔ごとのエラー数	テスト ステップ	すべての「ステップ エラー」イベント（id 20）の数
間隔ごとの失敗数	テスト ステップ	すべての「中止」（id 50）および「サイクルの失敗」（id 13）イベントの数。
テスト実行	テスト ケース	エージェント レポート間隔の最後に実行されているテスト ケースの数。 カウントは「テストの開始」イベント（id 4）ごとに増加し、「テストの終了」イベント（id 5）ごとに減少します。

実行中仮想ユーザ	シミュレータ	エージェント レポート間隔の最後に実行されている仮想ユーザの数。 カウントは「サイクルの開始」イベント（id 11）ごとに増加し、「サイクル終了」イベント（id 24）ごとに減少します。
Test Runner Errors Per Interval	DevTest	すべての「サイクル ランタイム エラー」イベント（id 25）の数
Staging Errors Per Interval	DevTest	すべての「モデル定義エラー」イベント（id 23）の数

最初のエラー イベントが検出された場合にのみ、テスト ステップ、シミュレータ、またはテスト ケースの Errors Per Interval メトリックがレポートされます。 Failures Per Interval の場合も同様です。

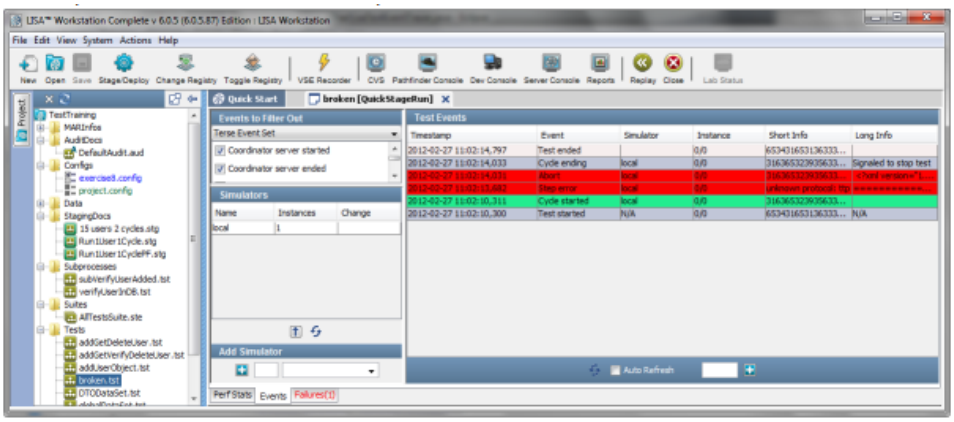
少数の短いテスト ケースのみが実行された場合、Tests Running および Virtual Users Running メトリックは各エージェント レポート サイクルに対して 0 をレポートし続ける場合があります。これは開始イベントと終了イベントの両方が同じ 15 秒間に発生し、その間隔の終了時の数が 0 の場合です。

メトリックが集約される場合、低いレベルのメトリックは 1 つのエージェント レポート間隔に作成されます。 集約されたメトリックは、その後の間隔に作成される場合があります。 間隔内で低いレベルとロールアップされたメトリックの値が一致しないときがあります。 この状況は回避不能です。

DevTest メトリックの派生

DevTest 固有のテスト イベント トレーサは「メソッド」トレーサです。メソッド トレーサは、特定のメソッド コールの直前または直後に挿入されるコードが含まれていることを意味します。テスト イベント トレーサは、BaseCoordinator クラスのコンストラクタであるメソッドをインスツルメントするために設計されています。コンストラクタが完了した後、トレーサはテスト イベント リスナを作成し、コーディネータにこのリスナを登録します。 その後、リスナは、事前に示されたタイプの各 TestEvent オブジェクトを受信します。 リスナはこれらの TestEvent オブジェクトから関連情報を抽出し、Introscope メトリック値の形式で Introscope Enterprise Manager にその情報を送信します。

以下の DevTest ワークステーション のスクリーン ショットは、「broken」という名前のテスト ケースの 1 回の実行で生成されたテスト イベントを示しています。これにはエラーが含まれています。 どの Introscope メトリックが作成されるかを理解するには、［イベント］列の各値を参照し、トレーサによってレポートされるメトリック (P. 135)を確認します。



トレーサ設定

DevTest トレースはすべて、エージェント プロファイル（デフォルトでは IntroscopeAgent.profile）内の DevTest 固有の .pbl ファイル（**lisa-full.pbl** または **lisa-typical.pbl**）の削除により無効にすることができます。 **Lisa.jar** を削除することもできます。 ただし、**lisa.pbd** で定義されたトレーサへの参照が残っている場合、エージェントは正しく起動できません。 これは、エージェント拡張内に見つからないトレーサを使用しようとしたためです。

DevTest テスト イベント トレーサは、有効な DevTest トグル ファイル（**lisa-toggles-full.pbd** または **lisa-toggles-typical.pbd**）内の「TurnOn: LisaTestEventTracing」ディレクティブをコメント アウトすることにより無効にできます。

その他の多くのパラメータを使用して、レポートされるメトリックの数を制限できます。これらのパラメータは **lisa.pbd** ファイルで定義されています。 以下のパラメータを制御できます。

- **minMetricLevel：** メトリックがレポートされる最低レベルを制御するパラメータ。 このパラメータは、以下のいずれかの値をとることができます。
  - Lisa
  - TestCase （デフォルト値）
  - シミュレータ
  - TestStep
- **maxRollupLevel：** メトリックを集約できる最大レベルを制御するパラメータ。 このパラメータは、以下のいずれかの値をとることができます。
  - TestCase （デフォルト値）
  - シミュレータ
  - TestStep

minMetricLevel の現在の設定より下のレベルで生成されたメトリックは、**minMetricLevel** に到達するまで集約されます。 これらのメトリックは、**minMetricLevel** が **maxRollupLevel** の上にない限り、**maxRollupLevel** のレベルまでレポートされます。**minMetricLevel** が **maxRollupLevel** の上にある場合、メトリックはレポートされません。 **maxRollupLevel** より上のレベルで生成されたメトリックは、集約されたレベルではなく、生成されたレベルでのみレポートされます。

また、その他の 6 つのパラメータを使用すると、テスト イベント トレーサが収集するメトリックをテスト ケース、シミュレータ、テスト ステップの選択した組み合わせに制限することができます。 トレーサを設定して、テスト ケース、シミュレータ、またはテスト ステップの名前に基づいて包含したり除外したりすることができます。 名前は、正規表現または正規表現のペアと照合できます。 包含する正規表現に一致し、除外する正規表現に一致しないすべての名前が、メトリックをレポートするために選択されます。 包含する正規表現が定義されていない場合、すべての名前が含まれます。 除外する正規表現が定義されていない場合、名前は除外されません。

デフォルトでは、すべての使用可能なテスト ケース、シミュレータ、およびテスト ステップが含まれます。「abort」および「end」という名前の内部テスト ステップのみが除外されます。正規表現に特定の特殊文字（選択パターンを指定するために使用される | 記号など）が含まれる場合、正規表現全体を二重引用符で囲む必要があります。 それ以外の場合、引用符はオプションです。 以下に、**lisa.pbd** で定義されているデフォルトの包含および除外パターンを示します。

```
SetTracerParameter: LisaCoordinatorTracer includeTestCasesRegExp
""
SetTracerParameter: LisaCoordinatorTracer excludeTestCasesRegExp
""
SetTracerParameter: LisaCoordinatorTracer
includeSimulatorsRegExp ""
SetTracerParameter: LisaCoordinatorTracer
excludeSimulatorsRegExp ""
SetTracerParameter: LisaCoordinatorTracer includeTestStepsRegExp
""
SetTracerParameter: LisaCoordinatorTracer excludeTestStepsRegExp
"abort|end"
```

メトリックがレポートされるメトリック パスを設定できます。 メトリック パスの DevTest レベルの部分は、[set the init variable for your book] メソッドの **TraceOneMethodWithParametersIfFlagged** ディレクティブで指定される最後の値です。 これはデフォルトで「DevTest」です。 その他のレベルについては、以下のパラメータが、メトリック パスのテスト ケース、シミュレータ、およびテスト ステップのレベルの部分を定義します。

- **pathComponentForTestCase**
- **pathComponentForSimulator**
- **pathComponentForTestStep**

これらのパラメータの値には、以下の文字を使用しないでください。

- コロン（:）はメトリック パス内で有効ではありません。
- メトリック パスの先頭または末尾には、メトリック パス ノード区切り文字（|）を使用できません。
- メトリック パスには、2 つの隣接したメトリック パス ノード区切り文字（|）を含めることができません。

コロンまたは縦棒は、それぞれアンダースコア（_）と置換されます。 先頭および末尾のスペースはこれらの名前から削除されます。null 値または空の文字列を使用した名前は、値「Unknown」と置換されます。

パラメータがコメント アウトされているか、または空の文字列である場合、これらのパラメータのデフォルト値が適用されます。 各値には、関連するレベルの名前がメトリック パスへ挿入される場所を示すために関連するプレースホルダを含める必要があります。 以下に、**lisa.pbd** で定義されているデフォルト値を示します。

```
SetTracerParameter: LisaCoordinatorTracer
pathComponentForTestCase "Test Case|{TestCase}"
SetTracerParameter: LisaCoordinatorTracer
pathComponentForSimulator "Simulator|{Simulator}"
SetTracerParameter: LisaCoordinatorTracer
pathComponentForTestStep "Step|{TestStep}"
```

Introscope エージェントのインストール

DevTest 用の Introscope エージェントは、配布ファイル **CALISAIntegrationNoInstaller9.1.1.0.zip** （Windows）または **CALISAIntegrationNoInstaller9.1.1.0.tar** （UNIX/Linux）の内容を抽出することによりインストールできます。

プロファイル ファイル名の競合を回避するには、その他の Introscope Java エージェントの配布が含まれていないディレクトリへこのファイルを抽出します。 次に、DevTest プロセスのセットをインスツルメントする 1 つ以上の vmoptions ファイルを作成します。 DevTest 固有のメトリックをレポートする DevTest プロセスは、DevTest ワークステーション およびコーディネータのみです。その他のプロセスはインスツルメントできます。ただし、その他のプロセスでは、メトリックの最小セットがレポートされます。 これらのメトリックは通常、CPU 使用率、メモリ使用率、エージェントおよび JVM アイデンティティに制限されています。

トレーサ ログの設定

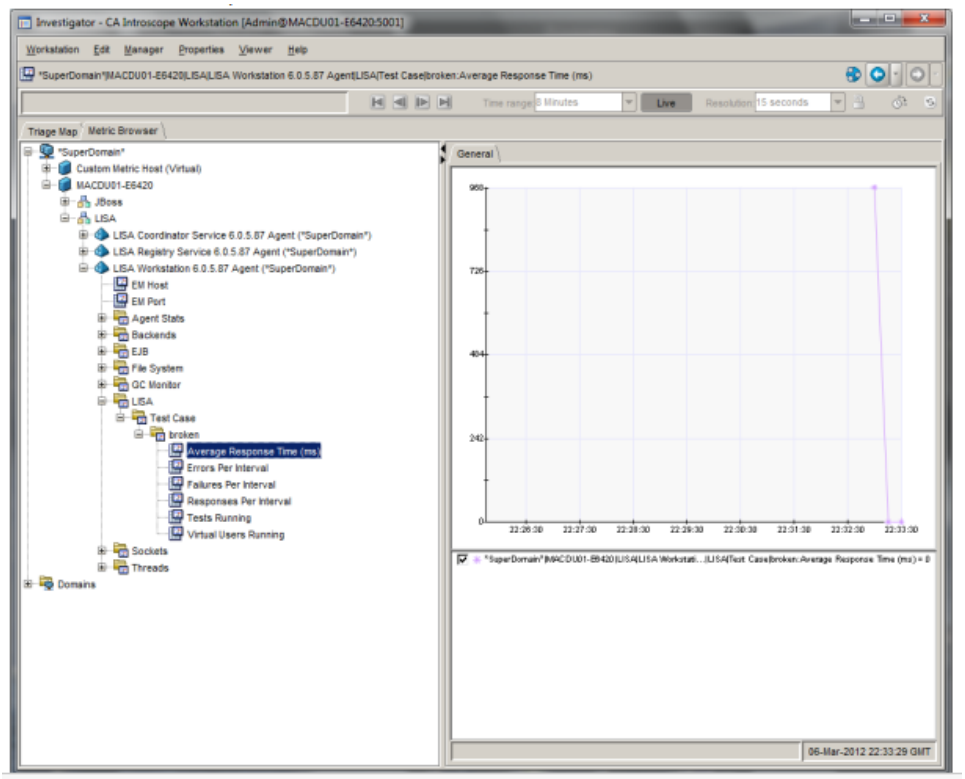
DevTest 固有のトレーサの動作に関する詳細をエージェント ログ ファイルに書き込むには、エージェント プロファイルに以下の行を追加します。

```
log4j.additivity.IntroscopeAgent.LisaCoordinatorTracer=false
log4j.logger.IntroscopeAgent.LisaCoordinatorTracer=DEBUG, logfile
```

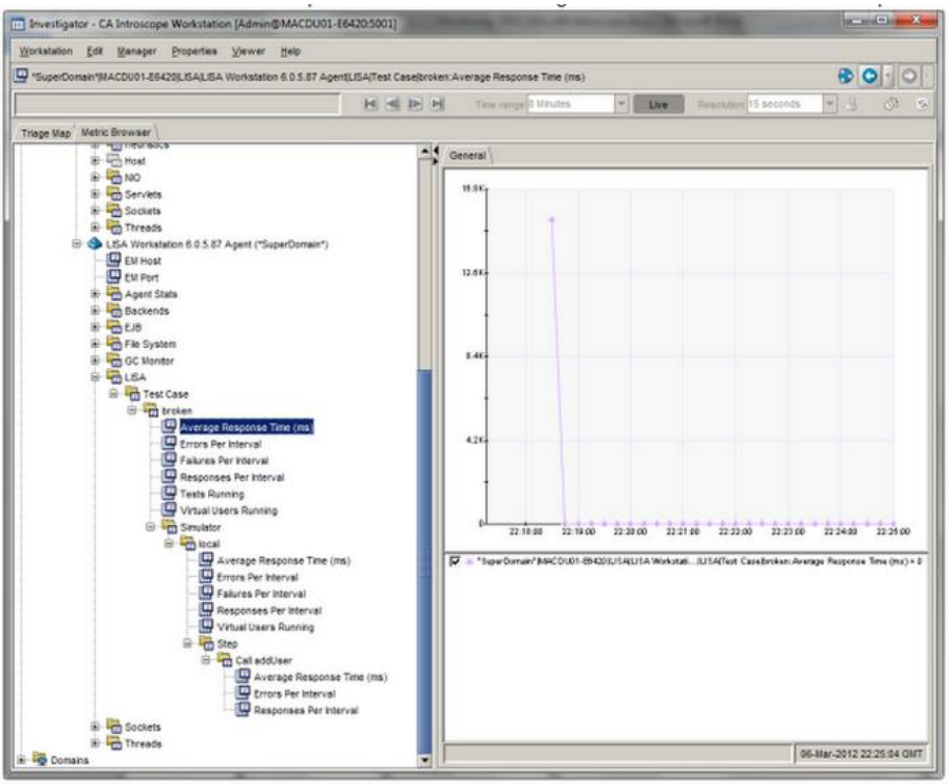
エージェント ログ レベルが DEBUG に設定されている場合、これらの行は必要ありません。 ただし、そのレベルのログ記録では、エージェント ログに大量の情報が書き込まれます。 この情報量により、DevTest トレーサから行を検索することが困難になる場合があります。

レポートされる標準メトリック

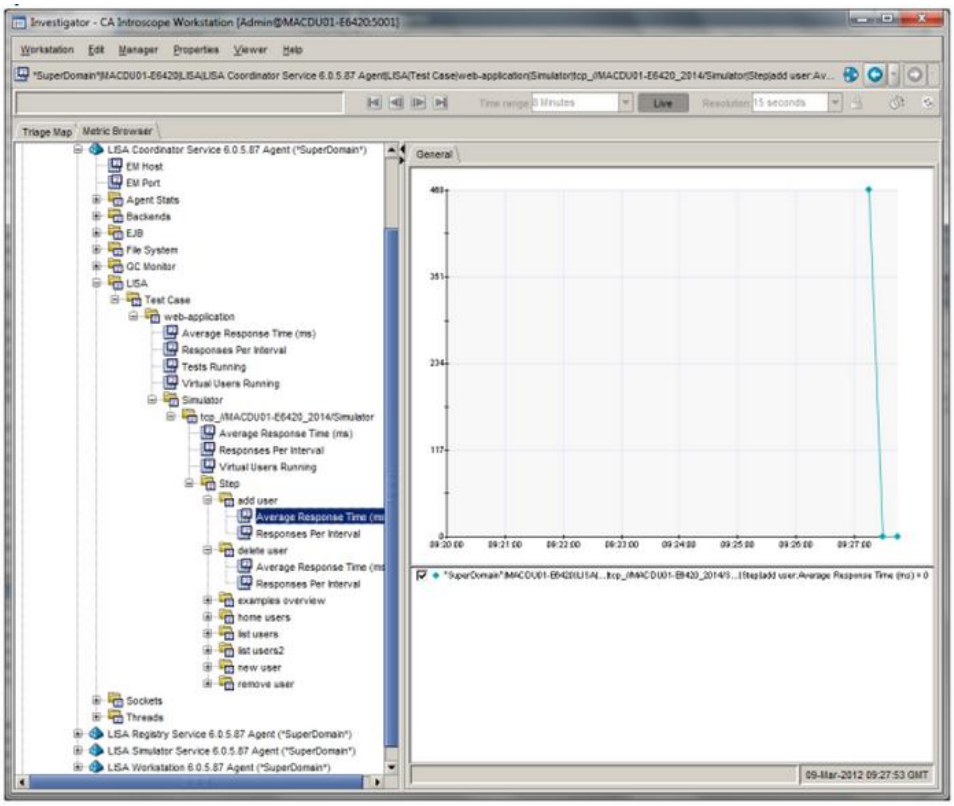
以下の図は、テスト イベント トレーサがレポートするメトリックの標準セットを示しています。以下の例では、「broken」という名前の簡単なテスト ケースが実行されています。 このテスト ケースには、設定エラーが含まれるテスト ステップが 1 つだけ含まれています。 このエラーが原因でテスト ケースは失敗します。 このエラーにより、「Errors Per Interval」および「Failures Per Interval」メトリックは以下のようにレポートされます。



以下の図は、**minMetricLevel** パラメータがデフォルトの TestCase から TestStep に変更された後に実行された同じテスト ケースを示しています。メトリックはシミュレータおよびテスト ステップ レベルで生成されます。



以下の図は、8 つのテスト ステップを含み、実行時にエラーを生成しないより複雑なテスト ケースを示しています。 また、このテストは、DevTest ワークステーション に組み込まれたプロセスではなく、個別のコーディネータおよびシミュレータ プロセスを使用してステージングされています。 このウィンドウには、追加のテスト ステップ ノードが表示されていますが、「Errors Per Interval」や「Failures Per Interval」メトリックが表示されていません。 メトリックは、Coordinator Service Agent ノード（名前には DevTest バージョン番号を含む）下にレポートされています。 また、シミュレータ名にメトリック パス名で無効なコロンが含まれていたため、コロンがアンダースコアに置換されていることにも注意してください。



メトリックの表示

Introscope Enterprise Manager には、DevTest 環境のモニタに関連するメトリック グループ、アラート、ダッシュボード、およびレポートが含まれる DevTest 固有の管理モジュールが付属しています。 以下の 2 つのダッシュボードが提供されます。

- 概要ダッシュボード (P. 146)
- テスト中のシステムの概要ダッシュボード (P. 152)

概要ダッシュボード

概要ダッシュボードは、インスツルメントされたすべてのプロセスおよび DevTest インストール内で実行されているテストの概要を示します。

概要ダッシュボードの [テスト] セクションには 4 つの簡易アラート アイコンが含まれます。

失敗

テストで最後のレポート間隔に失敗がレポートされた場合、警告に設定されます。 テストで最後のレポート間隔に 2 つ以上の失敗がレポートされた場合、危険に設定されます。

エラー

テストで最後のレポート間隔にエラーがレポートされた場合、警告に設定されます。 テストで最後のレポート間隔に 2 つ以上のエラーがレポートされた場合、危険に設定されます。

ランナー エラー

任意のレポート間隔の Test Runner Errors Per Interval メトリックの値が 1 の場合、警告に設定されます。任意のレポート間隔の値が 2 以上の場合、危険に設定されます。

ステージング エラー

任意のレポート間隔の Staging Errors Per Interval メトリックの値が 1 の場合、警告に設定されます。任意のレポート間隔の値が 2 以上の場合、危険に設定されます。

このセクションには以下のエレメントも含まれます。

全体アラート

上記の 4 つのアラートのいずれかが警告に設定されると、警告に設定されるサマリ アラート。 上記の 4 つのアラートのいずれかが危険に設定されると、このアラートは危険に設定されます。

テスト平均応答時間（ミリ秒）グラフ

テスト ケース、シミュレータ、およびテスト ステップについて、最大 10 個の Average Response Time (ms) メトリック グラフを表示します。選択された 10 個のグラフには、表示された期間の上位 10 件のメトリック値が表示されます。

テスト間隔ごとの応答数グラフ

テスト ケース、シミュレータ、およびテスト ステップについて、最大 10 個の Responses Per Interval メトリック グラフを表示します。選択された 10 個のグラフには、表示された期間の上位 10 件のメトリック値が表示されます。

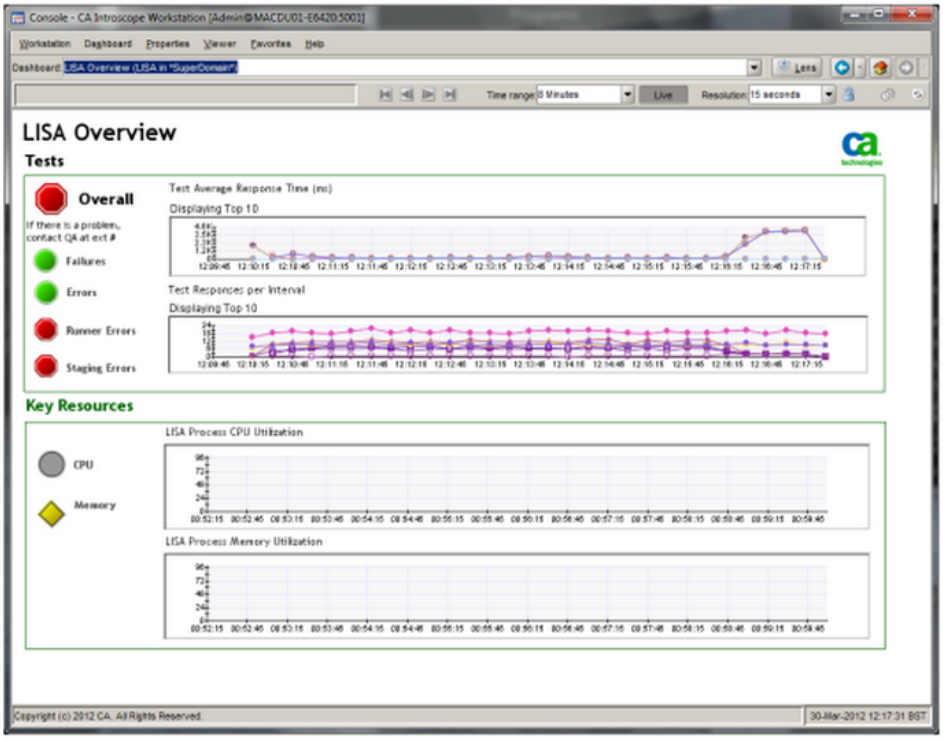
LISA プロセスの CPU 使用率グラフ

インスツルメントされているすべてのプロセスについて、**CPU:Utilization Percent** (process) メトリックのグラフを表示します。

LISA プロセスのメモリ使用率グラフ

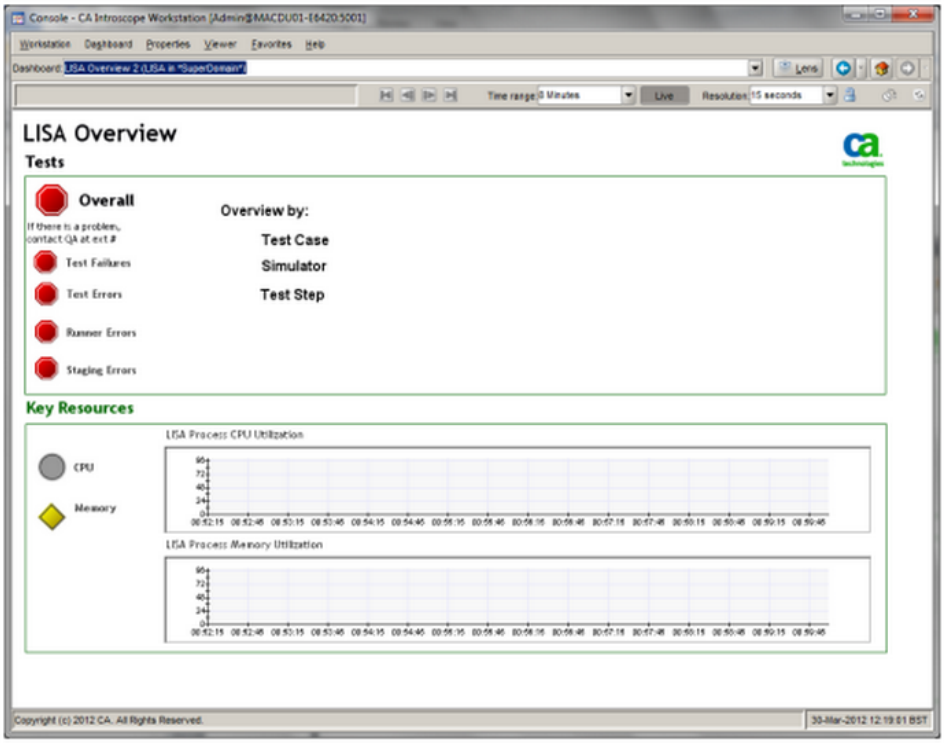
インスツルメントされているすべてのプロセスについて、**GC Heap:Bytes In Use** メトリックのグラフを表示します。

以下の図は、LISA の概要ダッシュボードを示しています。



概要ダッシュボードの代替バージョンでは、左上のパネル内のグラフが 3 つの新しいダッシュボードへのリンクに置き換わっています。 これらのダッシュボードは、以下のメトリックの概要を提供します。

- テスト ケース メトリック
- シミュレータ メトリック
- テスト ステップ メトリック

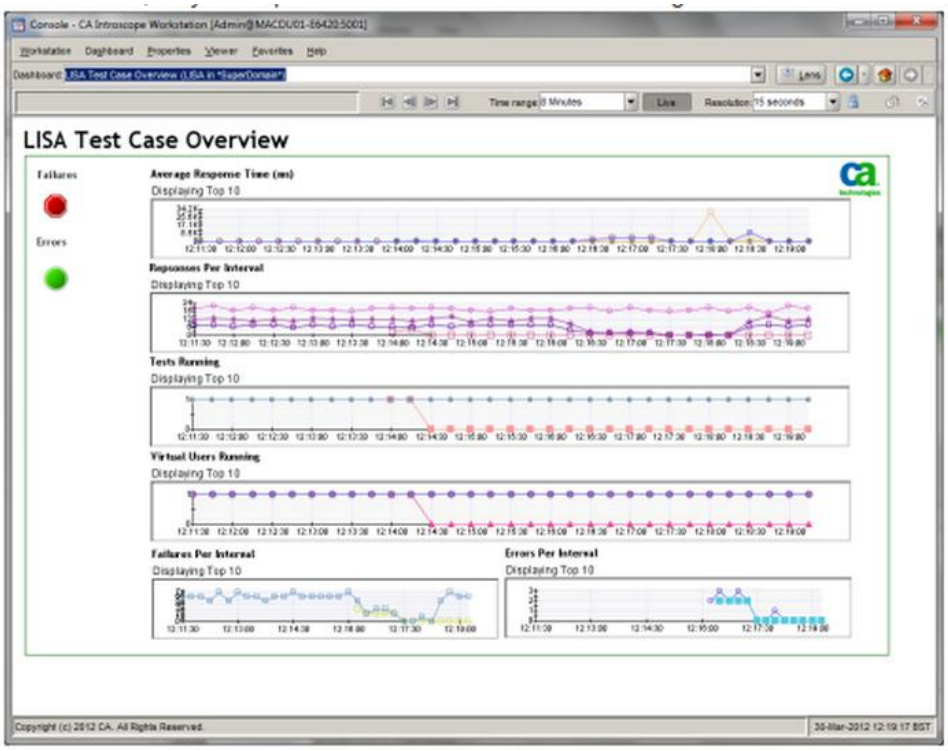


テスト ケースの概要**ダッシュボード**

テスト ケースの概要ダッシュボードは、以下のメトリックのグラフを表示します。

- 平均応答時間（ミリ秒）
- 間隔ごとの応答数
- テスト実行
- 実行中仮想ユーザ
- 間隔ごとの失敗数
- 間隔ごとのエラー数

各ケースで、上位 10 件のメトリックのみが表示されます。

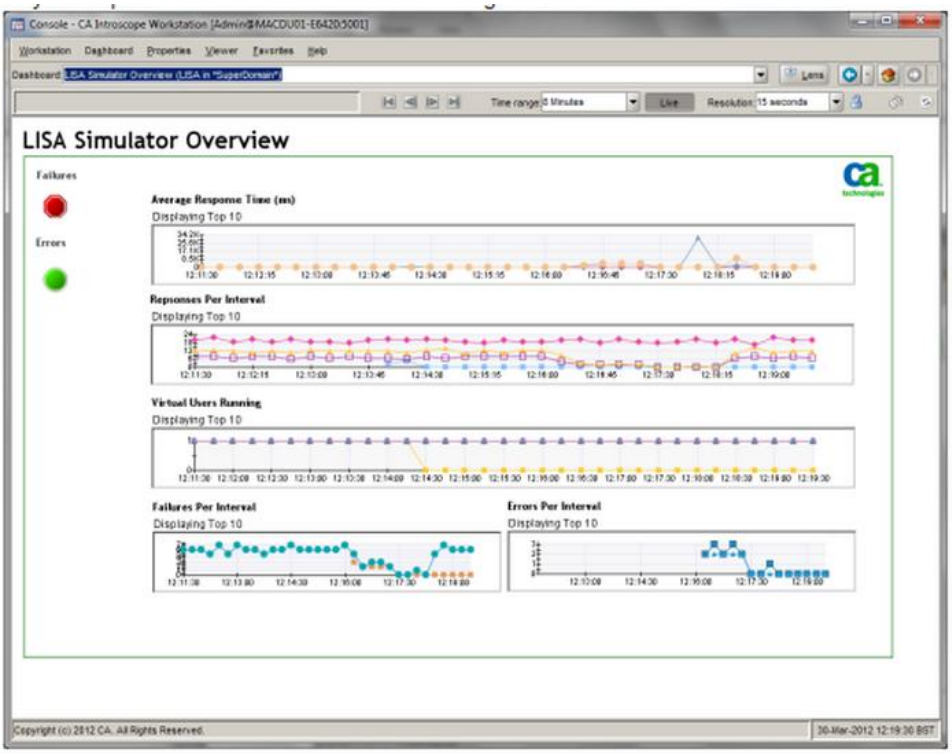


シミュレータの概要**ダッシュボード**

シミュレータの概要ダッシュボードは、以下のメトリックのグラフを表示します。

- 平均応答時間（ミリ秒）
- 間隔ごとの応答数
- 実行中仮想ユーザ
- 間隔ごとの失敗数
- 間隔ごとのエラー数

各ケースで、上位 10 件のメトリックのみが表示されます。

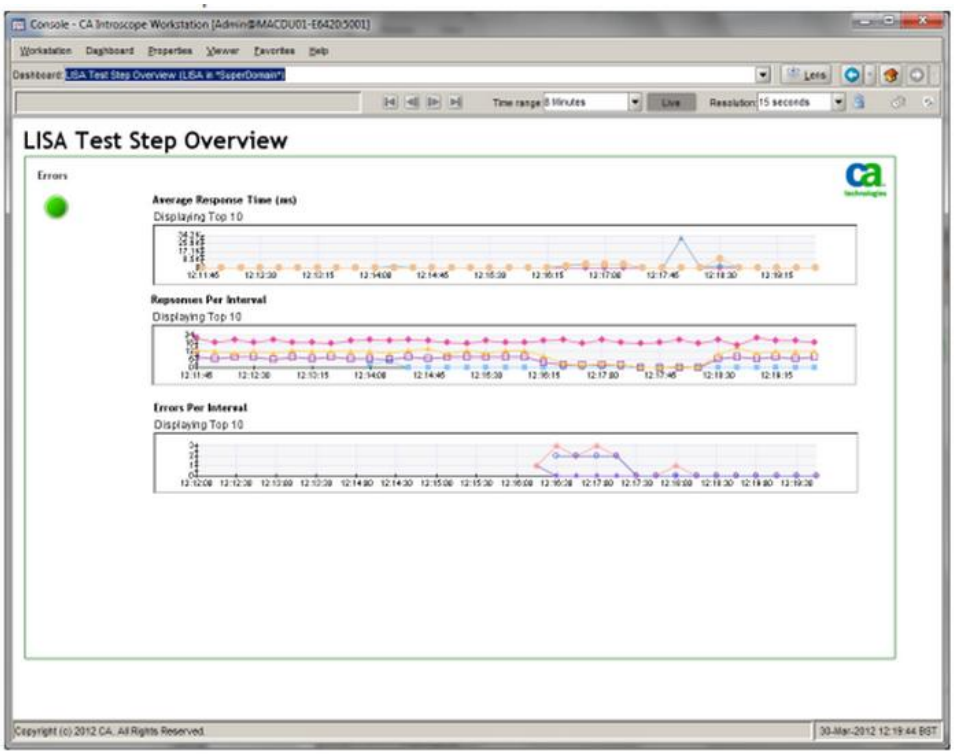


テスト ステップの概要**ダッシュボード**

テスト ステップの概要ダッシュボードは、以下のメトリックのグラフを表示します。

- 平均応答時間（ミリ秒）
- 間隔ごとの応答数
- 間隔ごとのエラー数

各ケースで、上位 10 件のメトリックのみが表示されます。


Console - CA Introscope Workstation [Admin@MACDU01-E64205001]
Workstation Dashboard Properties Viewer Favorites Help
Dashboard: LISA Test Step Overview (LISA in "SuperDomain")
Lens
Time range: 5 Minutes
Live
Resolution: 15 seconds
LISA Test Step Overview
Errors
Average Response Time (ms)
Displaying Top 10
Reponses Per Interval
Displaying Top 10
Errors Per Interval
Displaying Top 10
Copyright (c) 2012 CA. All Rights Reserved.

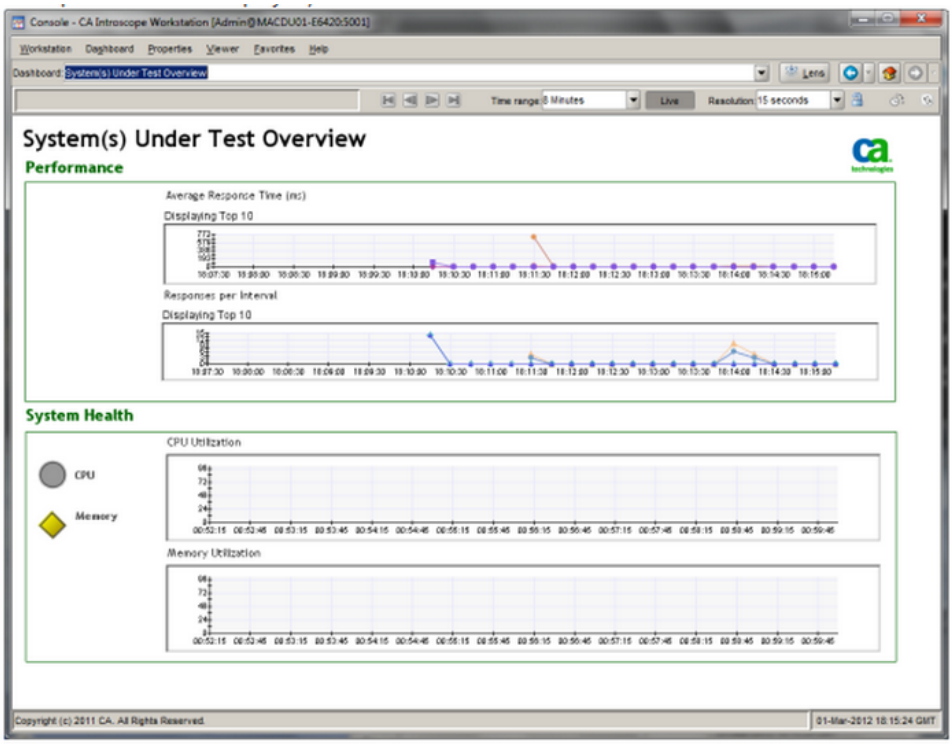
テスト中のシステムの概要ダッシュボード

テスト中のシステムの概要ダッシュボードは、DevTest テストから送信されるトランザクションを実行しているシステムの概要を示します。 DevTest がテストしているシステムを識別するのは困難です。 メトリック グループを編集して、DevTest がテストしているシステムでもある、Introscope がインスツルメントするシステムを定義できます。 デフォルトでは、グループにはデモ サーバが含まれます。

［テスト中のシステム］ダッシュボードは、DevTest によってテスト中のシステムのメトリックを表示します。 テスト中のシステムをインスツルメントするエージェントを定義するには、「Systems Under Test」で始まる名前で管理モジュール で定義されているすべてのメトリック グループに対してメトリック グループ エージェントの式を編集します。 デフォルトでは、この式は「**(.)|(.)|JBoss LISA Demo Server(.*)**」に設定されています。 これはデモ サーバを選択します。

ダッシュボードは 2 つのセクションで構成されています。上部のセクションには、テスト中のすべてのアプリケーション サーバの上位 10 件のフロントエンドの Average Response Time (ms) および Responses Per Interval メトリックが表示されます。

下部のセクションには、テスト中のすべてのアプリケーション サーバの CPU 使用率とメモリ使用率が表示されます。 また、このセクションには、CPU 使用率とメモリ使用率に関連するアラートを示す 2 つのアラート アイコンも含まれます。



レポート

管理モジュールには、DevTest レポートという名前のレポートが含まれます。 このレポートには、DevTest によってテスト中のシステムの状態を示す一連のグラフが含まれます。 また、このレポートには、DevTest がそれらのテストの実行から収集するテスト メトリックも含まれます。 レポートを複製およびカスタマイズして、特定のテストに焦点を当てることができます。 特定のテストに焦点を当てるには、複製したメトリック グループ式を変更して、特定のレポートに関連するメトリックのみを選択します。また、複製したレポートから個別のグラフを削除することもできます。

SAP システム ランドスケープ ディレクトリのセットアップ

このトピックでは、CA Service Virtualization に対して SAP システム ランドスケープ ディレクトリ（SLD）をセットアップ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. SLD サーバ用の以下の情報を取得します。
   - ホスト
   - ポート
   - ユーザ
   - パスワード
2. システム ランドスケープ ディレクトリにログオンします。

   注: このタスクを実行するには、LcrInstanceWriterLD 役割が必要です。
3. 以下のいずれかの操作を実行します。
   - SLD への DevTest の手動登録 (P. 155)
   - DevTest SLD XML ファイルのインポート (P. 156)

SLD への DevTest の手動登録

このトピックでは、SAP システム ランドスケープ ディレクトリ（SLD）に手動で DevTest を登録する方法について説明します。 この手順では、製品の名前/バージョンおよびソフトウェア コンポーネントの名前/バージョンを SLD ソフトウェア カタログ内に作成します。

次の手順に従ってください:

1. SLD ソフトウェア カタログで、［Products］-［New Product Version］を選択します。
2. ［Product Name］フィールドに、「**CA LISA SV**」と入力します。
3. ［Vendor Name］フィールドに、「**CA Technologies**」と入力します。
4. ［Product Version］フィールドに、バージョン番号（「**V7.5**」など）を入力します。
5. ［Create］をクリックします。
6. ［Technical Name］フィールドに、「**CA LISA SV**」と入力します。
7. ［Software Component Name］フィールドに、「**CA LISA**」と入力します。
8. ［Software Component Version］フィールドに、バージョン番号（「**V7.5**」など）を入力します。
9. ［Production State］フィールドに、「**released**」と入力します。
10. ［Create］をクリックします。

DevTest SLD XML ファイルのインポート

このトピックでは、DevTest SLD XML ファイルのインポートにより、SAP システム ランドスケープ ディレクトリ（SLD）をセットアップ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. SLD ソフトウェア カタログで、［Administration］-［Content］-［Import］を選択します。
2. ［Selected File］フィールドに、提供された SLD データ ZIP ファイルのパスを入力します。

   このファイルは、**LISA_HOME¥addons¥sap¥CALISA.xml** にあります
3. ［Import Selected File］をクリックします。

DevTest をインストールした後に、以下のパスを参照します。

- デフォルト インストール パス： **C:¥lisa¥bin¥**
- デフォルト ログ ファイル パス： **C:¥Users¥<userID>¥lisatmp_<バージョン番号>**

  各ログ ファイルおよびその意味の詳細については、「*管理*」の「ログ ファイルの概要」を参照してください。

  ログ ファイルはすべて人間が理解できます。
- デフォルト設定ファイル パス： **C:¥LISA_HOME¥**
- デフォルト設定ファイル： **lisa.properties** および **local.properties**

  ログ ファイル内のプロパティ ファイルおよび設定値の順序の詳細については、「*CA Application Test の使用*」の「プロパティ ファイル」を参照してください。

  プロパティ ファイルはすべて人間が理解できる、「java 名 = 値」形式のエントリです。

第 8 章: モバイル テスト環境の設定

このセクションには、以下のトピックが含まれています。

モバイル アプリケーション テストのシステム要件

このセクションでは、DevTest Solutions のモバイル アプリケーション テストのシステム要件について説明します。

モバイル テストでサポートされているオペレーティング システム

以下のオペレーティング システムがサポートされています。

Mac OS X 10.8 - Mountain Lion

VM イメージがサポートされています。

Mac OS X 10.9 - Mavericks

VM イメージがサポートされています。

Windows 7 以降

リモート iOS テストのみがサポートされています。 ローカル iOS テストはサポートされていません。

サポートされているモバイル オペレーティング システム

以下のモバイル オペレーティング システムがサポートされています。

- iOS 6.1 以降
- Android 4.2 以降
- ネイティブ、ハイブリッド、および Web

注: SauceLabs Android のサポートは、バージョン 4.2 に制限されています。

モバイル テストのハードウェア要件

以下のハードウェアがモバイル テストをサポートするために必要です。

CPU

- Intel Core i5 2.0 GHz または同等の AMD

メモリ

- クラウドまたはリモート モバイル テスト： 4 GB
- ローカル シミュレータ： 8 GB

モバイル テスト用のインストール前の手順

モバイル テスト用の DevTest Solutions をインストールする前に、以下のタスクを完了します。

モバイル テスト用のインストール前の手順（Macintosh）

次の手順に従ってください:

1. Xcode 6.1 （P. 159）をインストールします。

   Xcode は、Mac App Store から無償で入手可能です。

2. 以下の URL から最新の Android SDK （ADT Bundle）をダウンロードします。

   http://developer.android.com/sdk/index.html#download

   注: モバイル テストは SDK Platform 19 rev 3 以降のバージョンをサポートしています。 ADT Bundle にはビルド ツールが含まれており、モバイル テストにはバージョン 19.0.1、19.1.0、または 20.0.0 以降が必要です。これらのバージョンが ADT Bundle によってインストールされず、DevTest ワークステーション で zipalign または aapt に関するエラー メッセージが表示される場合は、「Android SDK ビルド ツール (P. 164)」を参照してください。

3. ANDROID_HOME プロパティを定義します （P. 162）。

4. Android SDK をセットアップします （P. 163）。

注: 使用可能なモバイル デバイス シミュレータは、Xcode または SDK のインストールされているバージョンによって異なります。

Xcode バージョン 6.1 のインストール

モバイル テストには、Xcode バージョン 6.1 が必要です（その後のバージョンはサポートされていません）。

新規バージョンのダウンロード

次の手順に従ってください：

1. https://developer.apple.com/support/xcode に移動します。
2. Xcode 6.1 をダウンロードしてインストールします。

注: Xcode 6 からの iOS 7 テストの実行はサポートされていません。 iOS 7 テストを実行するには Xcode 5 をインストールする必要があります。

IPA 署名

デバイス上で iOS アプリケーションをテストするには、SSL 証明書を取得し、テストするデバイスをプロビジョニングする必要があります。これは、Apple Developer Member Center で行うことができます。

その後、証明書で署名されているアプリケーションを識別する、モバイル プロビジョニング ファイルを使用してデバイスをプロビジョニングできます。モバイル プロビジョニング ファイルを（IPA 署名によって）.ipa ファイルに追加して、デバイス上でアプリケーションを実行できるユーザの署名と証明書の署名が一致することを確認します。

モバイル テストでは、テスト デバイスに iOS.ipa ファイルを展開する前に、ファイルに署名できます。 正しく機能させるには、多少の設定が必要です。

次の手順に従ってください:

1. Apple Developer Member Center にログインします。
2. ［Certificates, Identifiers, & Profiles］をクリックします。
3. USB ケーブルを使用して iOS デバイスを接続します。
4. ［iOS Apps］（左側）で、［Devices］をクリックします。
5. デバイスを追加するには、［+］アイコンをクリックします。

   注: iTunes を使用してデバイスを追加することもできます。
6. デバイスが iOS デバイスのリストに含まれることを確認します。
7. ［Certificates］で、［Development］をクリックします。
8. リスト内の証明書をクリックします。
9. ［Download］をクリックして、証明書をハード ドライブに保存します。
10. この証明書をキーチェーン アクセスにインストールします。
11. ［Provisioning Profiles］で、［Development］をクリックします。
12. リストで、［iOS Team Provisioning Profile］を選択します。
13. このファイルをダウンロードし、ハード ドライブに保存します。

    ファイル名は次の例のようになります。
    iOS_Team_Provisioning_Profile_.mobileprovision。
14. DevTest を開きます。

15. lisa.properties プロパティ ファイルで、以下のプロジェクト プロパティを設定します。

    `MOBILE_PROVISION = path to iOS_Team_Provisioning_Profile_.mobileprovision`

    `IOS_CERTIFICATE = Keychain Access に表示される証明書の名前`（証明書ファイル名そのものではありません）

    注: codesign ユーティリティで一致させることができるため、正確な文字列を入力する必要はありません。「iPhone Developer」のような名前を使用できます。一意の名前であれば十分です。ただし、その他の iOS 証明書がある場合、証明書の名前全体を使用することをお勧めします。

UIAutomation の有効化

iOS 8 デバイスでは、UIAutomation を有効にする必要があります。このモードを有効にすると、Appium でインスツルメントを検出できます。

UIAutomation オプションは、［Settings］-［Developer］-［Enable UI Automation］にあります。

モバイル テスト用のインストール前の手順（Windows）

次の手順に従ってください:

1. 最新の Android SDK をダウンロードしてインストールします。

   http://developer.android.com/sdk/index.html#download

   注: DevTest は SDK Platform 19 rev 3 以降のバージョンをサポートしています。 ADT Bundle にはビルド ツールが含まれており、モバイル テストにはバージョン 19.0.1、19.1.0、または 20.0.0 以降が必要です。 これらのバージョンが ADT Bundle によってインストールされず、DevTest ワークステーション で zipalign または aapt に関するエラー メッセージが表示される場合は、「Android SDK ビルド ツール (P. 164)」を参照してください。

2. ANDROID_HOME プロパティを定義します (P. 162)。
3. Android SDK をセットアップします (P. 163)。

ANDROID_HOME の定義

お使いの Android 仮想デバイス（AVD）が正しく機能するには、ANDROID_HOME プロパティを定義する必要があります。

次の手順に従ってください:

1. DevTest ワークステーション で、［システム］-［プロパティの編集］をクリックします。

   ［システム プロパティ］ウィンドウが表示されます。

2. 以下のプロパティを挿入します。

   ```
   ANDROID_HOME= <お使いの ADT Bundle の完全修飾パス>
   ```

   以下に例を示します。

   ```
   ANDROID_HOME= C:¥ADT
   Bundle¥adt-bundle-windows-x86_64-20131030¥sdk
   ```

3. ［保存］をクリックします。
4. ［システム プロパティ］ウィンドウを閉じます。

Android SDK のセットアップ

DevTest Solutions インストーラには、Android アプリケーションを開発するために使用されるツールを提供する Android Developer Tools（ADT）Bundle が含まれています。 ADT Bundle には、Eclipse IDE とビルトイン ADT のバージョンが含まれています。 1 つ以上の Android 仮想デバイス（AVD）を作成するには、ADT Bundle を使用します。

次の手順に従ってください:

1. 最新の Android SDK （ADT Bundle）をダウンロードします。

   http://developer.android.com/sdk/index.html#download

2. ADT Bundle を解凍します。
3. ADT Bundle の eclipse フォルダにある **eclipse.exe** を実行します。

   Workspace Launcher が開きます。
4. プロジェクトを保存するワークスペースを選択して［OK］をクリックします。

   ［Java ADT］ウィンドウが表示されます。
5. ［Window］-［Android Virtual Device Manager］をクリックします。

   Android Virtual Device Manager が開きます。
6. ［New］をクリックします。

   ［Create New Android Virtual Device (AVD)］ページが表示されます。
7. 以下のフィールドに入力します。

   AVD Name

   作成する Android 仮想デバイスの名前。 エミュレータ セッション アセットの作成時に、［Android AVD］フィールドでこの名前を使用します。

   デバイス

   テストするデバイスのタイプを選択します。

   Target

   AVD のターゲットを選択します。

   Memory Options

   ［RAM］値を 512、［VM Heap］値を 32 に設定します。

8. 残りのフィールドはデフォルト値のままにするか、またはお使いの AVD 用にカスタマイズします。
9. ［OK］をクリックします。

   新しいデバイスが、Android Virtual Device Manager のデバイスのリストに追加されます。

Android SDK ビルド ツール

Android SDK Bundle には、ビルド ツールと呼ばれるコンポーネントが含まれています。モバイル テストにはバージョン 19.0.1、19.1.0、または 20.0.0 以降が必要です。

これらのバージョンが ADT Bundle によってインストールされていないと、DevTest ワークステーション でモバイル アセットを作成するときに次のエラー メッセージが表示される場合があります。

`'aapt'' コマンドが見つかりません。 Android SDK Build-tools Rev. 20.`をインストールしてください。

または

`'zipalign'' コマンドが見つかりません。 Android SDK Build-tools Rev. 20.` をインストールしてください。

このエラー メッセージが表示される場合は、Android SDK ビルド ツールのバージョンを更新します。

次の手順に従ってください：

1. 解凍された ADT Bundle が含まれているフォルダから **android sdk** を実行します。

   Android SDK Manager が開きます。
2. Tools フォルダで、19.0.1、19.1.0、または 20.0.0 がインストールされ、選択されていることを確認します。 バージョン 20.0.0 は、使用に最適なバージョンです。

適切なバージョンのビルド ツールをインストールしたら、モバイル アセットの作成を続行できます。

USB デバッグの有効化

デバイスを USB ケーブルでコンピュータに直接接続した後、Android で USB デバッグ モードを有効化できます。 このモードを有効にすると、Android デバイスと Android SDK 間の接続が可能になります。

USB デバッグ モードは、お使いのデバイスの設定メニューの開発者向けオプションにあります。

Genymotion の使用

Genymotion は、Android エミュレータとして VirtualBox を使用する別のエミュレータです。Genymotion は、Android ソフトウェア開発キット（ASDK）で提供されている Android エミュレータより、高速で全体的に高いパフォーマンスを提供します。 このツールを使用するには、追加のセットアップが必要です。

次の手順に従ってください:

1. www.genymotion.com から VirtualBox および Genymotion の両方をダウンロードしてインストールします。

   注: Genymotion をインストールするには、Genymotion のユーザ アカウントを作成する必要がありますが、基本的なダウンロードは無償です。Windows プラットフォームには、統合 Virtualbox/Genymotion インストーラがあります。

2. お使いの仮想デバイスを追加します。

   a. Genymotion を起動します。

      はじめて Genymotion を起動するときに、仮想デバイスを追加するように促されます。

   b. ［はい］をクリックします。

      ［Create a New Virtual Device］ウィンドウが表示されます。

   c. ［Connect］をクリックし、Genymotion ユーザ ID およびパスワードを入力すると、使用可能なデバイスが表示されます。

      使用可能な仮想デバイスのリストが表示されます。 ページの上部で、Android バージョンまたはデバイス モデルによってデバイスのリストをフィルタできます。

   d. 追加するデバイスを選択し、［Next］をクリックします。

   e. 選択したデバイスの詳細を確認します。

      注: デバイスの名前は、DevTest Solutions のアセット ダイアログ ボックスに入力した名前です。 Genymotion のデフォルト名はより簡単な名前に変更できます。

   f. ［次へ］をクリックします。

      仮想デバイスのダウンロードが開始されます。 ダウンロード イメージのサイズは、通常約 200 MB です。

   g. ダウンロードが完了したら、［Finish］をクリックします。

h. 別の仮想デバイスを追加するには、Genymotion のメイン ウィンドウの［Add］をクリックし、上記の手順を繰り返します。

3. DevTest で、新しい仮想デバイス用の設定ファイルおよびアセットを作成します。

   a. Android 仮想デバイス用の標準的なシミュレータ セッション アセットを作成します。 詳細については、「*CA Application Test の使用*」の「モバイル アセット」を参照してください。

      詳細については、「*はじめに*」の「モバイル チュートリアル 2 - Android テスト ケースの記録」を参照してください。

   b. ［Android AVD］フィールドに、「**genymotion:**」に続けて Genymotion デバイスの名前を入力します。 以下に例を示します。

      ```
      genymotion:Galaxy Nexus - 4.3 - API 18 - 720x1280
      ```

Genymotion デバイスの管理

Genymotion デバイスは、VirtualBox でインストールされる VBoxManage ツールで管理できます。

役立つコマンド ライン コマンドのリストを以下に示します。

vboxmanage list avd

使用可能な仮想デバイスをすべてリスト表示します。 このコマンドの出力は以下のようになります。

```
Nexus 7 - 4.3 - API 18 - 1280x800"
{144161dd-750e-4fff-8d46-0da8bc0c226b}
GalaxyNexus4.2.2-API17" {d740a4fa-df15-4768-aee1-ffebfb883dc1}
```

各行は識別子のペアです。 最初の識別子は、そのデバイスの作成時にユーザが指定した名前です。 2 番目の識別子は UID です。 コマンド ラインでターゲットとしてデバイスを渡す場合、どちらの識別子も使用できます。 コマンド ラインで UID を使用する場合、中かっこで囲む必要はありません。

vboxmanage list runningvms

実行中の仮想デバイスをリスト表示します。

ADK コマンド **$ANDROID_HOME/platform-tools/adb devices** で同じ機能を実行できます。 ADK コマンドは、Genymotion および ADK でエミュレートされたデバイスの両方をリスト表示します。

このコマンドの出力は以下のようになります。

```
emulator-5554   device
192.168.0.56:5555   device
```

ADK デバイスは「emulator」で始まり、Genymotion デバイスは IP アドレスで始まります。 その後の数値は、デバイスが使用しているポートです。

player --vm-name <UID または名前>

デバイスがインストールされているパスに移動し、このコマンドを使用して仮想デバイスを起動します。Genymotion インターフェースからデバイスを起動することもできます。

VBOXMANAGE_CMD の定義

VirtualBox をデフォルトではない場所にインストールする場合は、VBOXMANAGE_CMD プロパティを定義します。 このプロパティにより DevTest が vboxmanage ツールを検索できるため、Genymotion シミュレータが効率よく機能できます。

このプロパティは 3 つの方法で定義できます。

- Windows の環境変数を定義する。
- lisa.properties ファイルを編集する（すべてのプロジェクトに影響します）。
- プロパティ エディタでプロジェクト プロパティ ファイルにプロパティを追加する（単一のプロジェクトに影響します）。

Windows の環境変数の定義

マシンの［システム プロパティ］で［環境変数］ダイアログ ボックスを開き、以下のコマンドを追加します。

Windows

```
VBOXMANAGE_CMD=c:¥program
files¥oracle¥virtualbox¥vboxmanage.exe
```

Mac

```
export VBOXMANAGE_CMD=/usr/bin/vboxmanage
```

LISA プロパティ ファイルの編集

lisa.properties ファイルの編集は、DevTest ワークステーション で定義されているプロジェクトのすべてに影響します。

次の手順に従ってください：

1. メイン メニューからの［システム］-［LISA プロパティの編集］を選択します。
2. システム ファイルで VBOXMANAGE_CMD を定義するには、以下のテキストを追加します。

   ```
   VBOXMANAGE_CMD=c:¥program
   files¥oracle¥virtualbox¥vboxmanage.exe
   ```

3. ファイルを保存します。

プロジェクト プロパティ ファイルへのプロパティの追加

プロパティ エディタでのプロジェクト プロパティ ファイルへのプロパティの追加は、そのプロジェクトのみに影響します。

次の手順に従ってください：

1. プロジェクト パネルで、Configs フォルダにある **project** をダブルクリックします。

   プロパティ エディタが開きます。

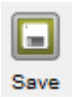
2. 行を追加するには、プロパティ エディタの下部にある  ［追加］をクリックします。

3. ［キー］フィールドに以下のように入力します。

   ```
   VBOXMANAGE_CMD
   ```

4. ［値］フィールドに以下のように入力します。

   ```
   c:¥program files¥oracle¥virtualbox¥vboxmanage.exe
   ```

5. メイン ツールバーの 

［保存］をクリックします。

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」を参照してください。

イベント

*イベント*は、発生したアクションに関するメッセージです。 テスト ケースまたは仮想サービス モデル レベルでイベントを設定できます。 詳細については、「*CA Application Test の使用*」の「イベントについて」を参照してください。

会話ツリー

*会話ツリー*は、仮想サービス イメージにおいてステートフル トランザクションの会話パスを表すリンクされたノードのセットです。 各ノードは、withdrawMoney などの操作名でラベル付けされます。 getNewToken、getAccount、withdrawMoney、deleteToken は、金融機関システムの会話パスの一例です。 詳細については、「*CA Service Virtualization の使用」を参照してください。*

仮想サービス モデル（VSM）

*仮想サービス モデル*は、実際のサービス プロバイダなしでサービス要求を受信および応答します。 詳細については、「*CA Service Virtualization の使用*」の「仮想サービス モデル（VSM）」を参照してください。

監査ドキュメント

*監査ドキュメント*では、1 つのテスト、またはスイート内の 1 つのテスト セットに対する成功条件を設定できます。 詳細については、「*CA Application Test の使用*」の「監査ドキュメントの作成」を参照してください。

クイック テスト

*クイック テスト*機能を使用すると、最小のセットアップでテスト ケースを実行できます。 詳細については、「*CA Application Test の使用*」の「クイック テストのステージング」を参照してください。

グループ

グループ、または仮想サービス グループは、VSE コンソールでまとめてモニタできるように、同じグループ タグでタグ付けされている仮想サービスのコレクションです。

継続的検証サービス（CVS）ダッシュボード

*継続的検証サービス（CVS）ダッシュボード*では、長期間にわたって定期的に実行するテスト ケースおよびテスト スイートをスケジュールできます。 詳細については、「*CA Application Test の使用*」の「継続的検証サービス（CVS）」を参照してください。

コーディネータ

*コーディネータ*はテスト ランの情報をドキュメントとして受け取り、1 つ以上のシミュレータ サーバで実行されるテストをコーディネートします。詳細については、「*CA Application Test の使用*」の「コーディネータ サーバ」を参照してください。

コンパニオン

*コンパニオン*は、すべてのテスト ケースの実行の前後に実行されるエレメントです。 コンパニオンは、単一のテスト ステップではなく、テスト ケース全体に適用されるフィルタとして理解できます。 コンパニオンはテスト ケース内で（テスト ケースに対して）グローバルな動作を設定するために使用されます。 詳細については、「*CA Application Test の使用*」の「コンパニオン」を参照してください。

サービス イメージ（SI）

*サービス イメージ*は、CA Service Virtualization で記録されたトランザクションの正規化バージョンです。 各トランザクションは、ステートフル（会話型）またはステートレスです。 サービス イメージを作成する方法の 1 つは、仮想サービス イメージ レコーダを使用することです。 サービス イメージは、プロジェクトに格納されます。 サービス イメージは、*仮想サービス イメージ*（VSI）とも呼ばれます。詳細については、「*CA Service Virtualization の使用*」の「サービス イメージ」を参照してください。

サブプロセス

*サブプロセス*は、別のテスト ケースによってコールされるテスト ケースです。 詳細については、「*CA Application Test の使用*」の「サブプロセスの作成」を参照してください。

シミュレータ

*シミュレータ*は、コーディネータ サーバの管理下でテストを実行します。詳細については、「*CA Application Test の使用*」の「シミュレータ サーバ」を参照してください。

ステージング ドキュメント

*ステージング ドキュメント*には、テスト ケースを実行する方法に関する情報が含まれます。詳細については、「*CA Application Test* の使用」の「ステージング ドキュメントの作成」を参照してください。

設定

*設定*は、プロパティの名前付きのコレクションであり、通常はテスト中のシステムの環境に固有の値を指定します。 ハードコードされた環境データをなくすことにより、設定を変更するだけで、異なる環境内のテスト ケースまたは仮想サービス モデルを実行できます。 プロジェクトのデフォルト設定の名前は project.config です。 プロジェクトは多数の設定を持つことができますが、一度にアクティブになるのは 1 つの設定のみです。詳細については、「*CA Application Test の使用*」の「設定」を参照してください。

対話型テスト ラン（ITR）

*対話型テスト ラン（ITR）*ユーティリティを使用すると、テスト ケースまたは仮想サービス モデルをステップごとに実行できます。 テスト ケースまたは仮想サービス モデルを実行時に変更し、結果を確認できます。 詳細については、「*CA Application Test の使用*」の「対話型テスト ラン（ITR）ユーティリティの使用」を参照してください。

ディセンシタイズ

*ディセンシタイズ*は、機密データをユーザ定義の代替データに変換するために使用されます。 クレジット カード番号や社会保障番号は機密データの例です。 詳細については、「*CA Service Virtualization の使用*」の「データのディセンシタイズ」を参照してください。

データ セット

*データ セット*は、実行時にテスト ケースまたは仮想サービス モデルにプロパティを設定するために使用できる値のコレクションです。 データ セットによって、テスト ケースまたは仮想サービス モデルに外部のテスト データを使用することができます。データ セットは、DevTest の内部または外部（たとえば、ファイルやデータベース テーブル）に作成できます。 詳細については、「*CA Application Test の使用*」の「データ セット」を参照してください。

データ プロトコル

*データ プロトコル*は、データ ハンドラとも呼ばれます。 CA Service Virtualization では、データ プロトコルは、要求の解析処理を行います。一部のトランスポート プロトコルは、要求を作成するジョブの委任先のデータ プロトコルを許可（または要求）します。 結果として、プロトコルは要求ペイロードを認識する必要が生じます。 詳細については、「*CA Service Virtualization の使用*」の「データ プロトコルの使用」を参照してください。

テスト ケース

*テスト ケース*は、テスト中のシステムのビジネス コンポーネントをテストする方法の仕様です。 各テスト ケースには、1 つ以上のテスト ステップが含まれます。 詳細については、「*CA Application Test の使用*」の「テスト ケースの作成」を参照してください。

テスト スイート

*テスト スイート*は、順番に実行されるようにスケジュールされたテスト ケース、その他のテスト スイート、またはその両方のグループです。 スイート ドキュメントは、スイートのコンテンツ、生成するレポート、および収集するメトリックを指定します。詳細については、「*CA Application Test の使用*」の「テスト スイートの作成」を参照してください。

テスト ステップ

*テスト ステップ*は、実行される単一のテスト アクションを表すテスト ケース ワークフローのエレメントです。 テスト ステップの例としては、Web サービス、Java Bean、JDBC、JMS メッセージングなどがあります。 テスト ステップには、フィルタ、アサーション、データ セットなどの DevTest エレメントを含めることができます。 詳細については、「*CA Application Test の使用*」の「テスト ステップの作成」を参照してください。

トランザクション フレーム

*トランザクション フレーム*は、DevTest Java エージェントまたは CAI Agent Light がインターセプトしたメソッド コールに関するデータをカプセル化します。詳細については、「*CA Continuous Application Insight の使用*」の「ビジネス トランザクションおよびトランザクション フレーム」を参照してください。

ナビゲーション許容差

*ナビゲーション許容差*は、CA Service Virtualization が会話ツリーを検索して次のトランザクションを見つける方法を制御する設定です。 オプションは、CLOSE、WIDE、および LOOSE です。 詳細については、「*CA Service Virtualization の使用*」の「ナビゲーション許容差」を参照してください。

ネットワーク グラフ

ネットワーク グラフは、DevTest クラウド マネージャおよび関連するラボをグラフで表示するサーバ コンソールの領域です。詳細については、「*CA Application Test の使用*」の「ラボの開始」を参照してください。

ノード

DevTest の内部では、テスト ステップは*ノード*とも呼ばれます。これが、一部のイベントがイベント ID 内にノードを持つ理由です。

パス

*パス*には、Java エージェント がキャプチャしたトランザクションに関する情報が含まれます。詳細については、「*CA Continuous Application Insight の使用*」を参照してください。

パス グラフ

*パス グラフ*には、パスおよびそのフレームのグラフ表示が含まれています。 詳細については、「*CA Continuous Application Insight の使用*」の「パス グラフ」を参照してください。

反応時間

*反応時間*は、テスト ステップを実行する前にテスト ケースが待機する時間です。詳細については、「*CA Application Test の使用*」の「テスト ステップの追加 - 例」および「ステージング ドキュメント エディタ - ［ベース］タブ」を参照してください。

フィルタ

*フィルタ*は、ステップの前後に実行されるエレメントです。フィルタは、結果のデータを処理、またはプロパティに値を格納する機会を提供します。グローバル フィルタは、テスト ケースまたは仮想サービス モデルの各ステップに適用されます。 詳細については、「*CA Application Test の使用*」の「フィルタ」を参照してください。

プロジェクト

*プロジェクト*は、関連する DevTest ファイルのコレクションです。 ファイルには、テスト ケース、スイート、仮想サービス モデル、サービス イメージ、設定、監査ドキュメント、ステージング ドキュメント、データ セット、モニタ、および MAR 情報ファイルなどが含まれます。 詳細については、「*CA Application Test の使用*」の「プロジェクト パネル」を参照してください。

プロパティ

*プロパティ*は、ランタイム変数として使用できるキー/値ペアです。 プロパティには、さまざまなタイプのデータを格納できます。 一般的なプロパティには、LISA_HOME、LISA_PROJ_ROOT、LISA_PROJ_NAME などがあります。 設定は、プロパティの名前付きのコレクションです。 詳細については、「*CA Application Test の使用*」の「プロパティ」を参照してください。

マジック ストリング

*マジック ストリング*は、サービス イメージの作成中に生成される文字列です。 マジック ストリングは、仮想サービス モデルによって応答内で意味のある文字列値が提供されることを確認するために使用されます。 {{=request_fname;/chris/}} は、マジック ストリングの一例です。 詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

マジック デート

レコーディング中、日付パーサは要求および応答をスキャンします。 日付表示形式の広範な定義に一致する値は、*マジック デート*に変換されます。 マジック デートは、仮想サービス モデルによって応答内で意味のある日付値が提供されることを確認するために使用されます。 {{=doDateDeltaFromCurrent("yyyy-MM-dd","10");/*2012-08-14*/} は、マジック デートの一例です。 詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

メトリック

*メトリック*により、テストおよびテスト中のシステムのパフォーマンス/機能面に定量的手法および測定単位を適用できます。 詳細については、「*CA Application Test の使用*」の「メトリックの生成」を参照してください。

モデル アーカイブ（MAR）

*モデル アーカイブ（MAR）*は、DevTest Solutions における主要な展開アーティファクトです。 MAR ファイルには、プライマリ アセット、プライマリ アセットを実行するために必要なすべてのセカンダリ ファイル、情報ファイル、および監査ファイルが含まれます。 詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

モデル アーカイブ（MAR）情報

*モデル アーカイブ（MAR）情報*ファイルは、MAR を作成するために必要な情報が含まれるファイルです。詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

ラボ

*ラボ*は、1 つ以上のラボ メンバの論理コンテナです。 詳細については、「*CA Application Test の使用*」の「ラボとラボ メンバ」を参照してください。

レジストリ

*レジストリ*は、すべての DevTest サーバおよび DevTest ワークステーション コンポーネントの登録を一元的に行うための場所です。 詳細については、「*CA Application Test* の使用」の「レジストリ」を参照してください。

仮想サービス環境（VSE）

*仮想サービス環境（VSE）*は、仮想サービス モデルを展開して実行するために使用する DevTest サーバ アプリケーションです。 VSE は CA Service Virtualization とも呼ばれます。 詳細については、「*CA Service Virtualization の使用*」を参照してください。