DevTest Solutions

管理

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: 一般的な管理

このセクションには、以下のトピックが含まれています。

デフォルト ポート番号

このトピックでは、以下の DevTest コンポーネントが使用する各デフォルト ポート番号を定義します。

- DevTest サーバ (P. 9)
- DevTest ワークステーション (P. 11)
- デモ サーバ (P. 12)

DevTest サーバ のデフォルト ポート番号

以下の表に、DevTest サーバ のさまざまなコンポーネントが使用するデフォルト ポート番号のリストを示します。 この表には、各ポート番号を設定するプロパティも含まれます。

ポート	DevTest コンポーネント	プロパティ
1505	組み込み Web サーバ	lisa.webserver.port
1506	エンタープライズ ダッシュボード（組み込み Web サーバ）	dradis.webserver.port
1507	DevTest ポータル	devtest.port
1528	Derby データベース	lisadb.internal.port、lisadb.pool.common.url

1529	デモ サーバ	該当なし
1530	エンタープライズ ダッシュボード Derby データベース ポート	dradisdb.internal.port
2003	エンタープライズ ダッシュボード ネットワーク ポート	lisa.net.15.port
2004	VSE マネージャ ネットワーク ポート	lisa.net.9.port
2005	テスト ランナー ネットワーク ポート	lisa.net.5.port
2006	ServiceManager ネットワーク ポート	lisa.net.11.port
2008	ワークステーション ネットワーク ポート	lisa.net.0.port
2009	ブローカ ネットワーク ポート	lisa.pathfinder.broker.port
2010	レジストリ ネットワーク ポート	lisa.net.3.port
2011	コーディネータ ネットワーク ポート	lisa.net.2.port
2012	JUnit 実行ネットワーク ポート	lisa.net.4.port
2013	VSE ネットワーク ポート	lisa.net.8.port
2014	シミュレータ ネットワーク ポート	lisa.net.1.port
2999	JDBC シミュレーション ドライバ	該当なし
3128	HTTP プロキシ	laf.httpproxy.port
3997	LPAR エージェント	lisa.mainframe.bridge.server.port
8443	組み込み Web サーバ（SSL が有効になっている場合）	lisa.webserver.ssl.port
61617	LPAR エージェント	lisa.mainframe.bridge.port

同じコンピュータで複数のシミュレータを作成すると、新しいシミュレータのポート番号は 2014 から増加していきます。 たとえば、3 台のシミュレータを追加すると、ポート番号は 2015、2016、および 2017 となります。

通常、各ラボに 1 つのコーディネータが存在します。 ただし、より多くのコーディネータを作成する場合は、コマンドラインでポートを定義します。以下に例を示します。

```
CoordinatorServer --name=tcp://hostname:2468/Coordinator2
```

lisa.properties ファイルは、ほとんどのデフォルト ポート番号を定義します。

インストール後に 1 つ以上のポート番号を変更する場合は、**site.properties** ファイルにプロパティを追加し、新しい値を設定します。 **lisa.properties** ファイルは更新しないでください。

JDBC シミュレーション ドライバのポート番号は、DevTest ワークステーション の仮想 JDBC リスナ ステップを編集することによって変更できます。

DevTest ワークステーション のデフォルト ポート番号

DevTest ワークステーション インストールのデフォルト ポート番号は、DevTest サーバ のデフォルト ポート番号のサブセットです。

ポート	DevTest コンポーネント	プロパティ
1505	組み込み Web サーバ	lisa.webserver.port
2004	VSE マネージャ	lisa.net.9.port
2005	テスト ランナー	lisa.net.5.port
2006	サービス マネージャ	lisa.net.11.port
2009	ブローカ	lisa.pathfinder.broker.port
2999	JDBC シミュレーション ドライバ	該当なし
3128	HTTP プロキシ	laf.httpproxy.port
8443	組み込み Web サーバ （SSL が有効になっている場合）	lisa.webserver.ssl.port

デモ サーバのデフォルト ポート番号

以下の表に、デモ サーバが使用するデフォルト ポート番号のリストを示します。

ポート	DevTest コンポーネント
1098	JNDI[JNDI]
1099	JNDI[JNDI]
1528	Derby データベース
8080	HTTP

デフォルト ポート番号は、以下のファイルで定義されています。

- lisa-demo-server/jboss/server/default/conf/jboss-service.xml
- lisa-demo-server/jboss/server/default/deploy/jboss-web.deployer/server.xml

インストール後に 1 つ以上のポート番号を変更する場合は、これらのファイルを直接更新できます。

共有インストール タイプ

共有インストール タイプは、以下の特徴がある環境用に設計されています。

- 複数のユーザが複数のコンピュータから 1 つの DevTest インストールを共有する。
- 各ユーザが個別のデータを持つ。

共有インストールでは、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保存されます。 ユーザはそれぞれ自分のデータを持ちますが、共通の DevTest 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 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。

インストーラで共有インストール タイプを選択すると、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。 各ディレクトリのデフォルトの場所は **USER_HOME** ディレクトリです。

共有インストールでは、**lisa.user.properties** ファイルは、**LISA_HOME** ディレクトリ、および インストールを実行しているユーザの **USER_HOME** ディレクトリに追加されます。 このファイルには、**lisa.data.dir** および **lisa.tmpdir** プロパティが含まれます。 ファイルの場所は、**lisa.user.properties** システム プロパティまたは **LISA_USER_PROPERTIES** 環境変数を設定することによって指定できます。 これらのプロパティのいずれかがシステム プロパティとして定義されている場合は、そのシステム プロパティ定義が優先されます。

注: **lisa.tmpdir** は、一時ファイルを格納できる場所の変更を可能にしますが、このプロパティを変更して、外部のマウント ポイントまたは外部共有に一時ファイルを保存することはお勧めしません。 一時ファイルの保存に外部共有を使用しているケースでご使用の製品が不安定な場合、環境を引き続きサポートするために、サポートが以前のように一時ファイルを、ローカルディスクを使用して保存するように指示することがあります。

別のコンピュータ上の新しいユーザは、共有インストールにアクセスしようとする前に、**USER_HOME** ディレクトリに自分用の **lisa.user.properties** ファイルを手動でセットアップする必要があります。 このファイルは、**LISA_HOME** ディレクトリからコピーできます。また、別のディレクトリを使用する場合は、手動で作成できます。

このプロパティ ファイルを設定するまでは、どの DevTest コンポーネントを起動しようとしても、そのコンポーネントは起動しません。**lisa.user.properties** ファイルの読み取りエラーを示すエラー メッセージが、画面およびログ ファイルに出力されます。

ディレクトリ構造

このトピックでは、DevTest ワークステーション (P. 15) および DevTest サーバ (P. 17) のディレクトリ構造について説明します。

このトピックでは、DevTest Solutions のインストール中にローカル インストール タイプを選択したと仮定します。 共有インストール タイプ (P. 13) を選択した場合、以下のディレクトリは **LISA_HOME** 以外の場所に存在する可能性があります。

- cvsMonitors
- データベース
- hotDeploy
- locks
- tmp

DevTest ワークステーション ディレクトリ

DevTest ワークステーション の **LISA_HOME** ディレクトリには、以下のディレクトリがあります。

注: locks ディレクトリには、読み取り/書き込み権限が必要です。

.install4j

インストーラが含まれます。

addons

DevTest アドオンが含まれます。

エージェント

DevTest エージェント用のファイルが含まれます。

bin

DevTest ワークステーション およびその他のコンポーネントの実行可能ファイルが含まれます。

database

さまざまなデータベース用の DDL ファイルが含まれます。

defaults

すべてのプロジェクトに共通する監査ドキュメントおよびステージング ドキュメントが含まれます。 プロジェクトに固有のステージング ドキュメントは Projects ディレクトリにあります。

doc

ライセンス契約書が含まれます。

examples

デモ サーバを使用するサンプルが含まれるデフォルト プロジェクト。

examples_src

サンプルのソース ファイルおよびキオスク関連のファイル。

hotDeploy

DevTest がモニタするディレクトリ。このファイルには Java クラスおよび JAR ファイルが含まれます。 このディレクトリの Java クラスおよび JAR ファイルは、DevTest クラスパスにあります。 このディレクトリに追加された新しいファイルまたはディレクトリは、すべて DevTest クラスパスに動的に追加されます。

incontainer

コンテナ内テスト手順が含まれます。

jre

必要な JRE が含まれます。

lib

必要な JAR ファイルが含まれます。

ライセンス

DevTest Solutions の実行に必要なライセンス ファイルが含まれます。

locks

プロセスの並列処理に使用されるロック ファイルが含まれます。

レポート

DevTest によって作成される XML ベースのテスト レポートが含まれます。

snmp

SNMP 関連ファイルが含まれます。

tmp

DevTest によって作成されるログ ファイルが含まれます。 問題についてサポートに連絡する場合、このディレクトリから 1 つ以上のファイルを送信するように求められることがあります。

umetrics

メトリックの収集に関連するファイルが含まれます。

DevTest ワークステーション でプロジェクトを作成すると、**Projects** ディレクトリが追加されます。

DevTest サーバ ディレクトリ

DevTest サーバの **LISA_HOME** ディレクトリには、以下のディレクトリがあります。

注: locks ディレクトリには、読み取り/書き込み権限が必要です。

.install4j

インストーラが含まれます。

addons

DevTest アドオンが含まれます。

エージェント

DevTest エージェント用のファイルが含まれます。

bin

レジストリ、コーディネータ、シミュレータ、VSE、DevTest ワークステーション、およびその他のコンポーネントの実行可能ファイルが含まれます。 このディレクトリには、以下のバッチ ファイルも含まれます。

- startdefservers.bat - デフォルト サーバを起動します。
- stopdefservers.bat - デフォルト サーバを停止します。

cvsMonitors

展開された CVS モニタが含まれます。

database

さまざまなデータベース用の DDL ファイルが含まれます。

defaults

すべてのプロジェクトに共通する監査ドキュメントおよびステージング ドキュメントが含まれます。 プロジェクトに固有のステージング ドキュメントは Projects ディレクトリにあります。

doc

ライセンス契約書が含まれます。

examples

デモ サーバを使用するサンプルが含まれるデフォルト プロジェクト。

examples_src

サンプルのソース ファイルおよびキオスク関連のファイル。

hotDeploy

DevTest がモニタするディレクトリ。このファイルには Java クラスおよび JAR ファイルが含まれます。 このディレクトリの Java クラスおよび JAR ファイルは、DevTest クラスパスにあります。 このディレクトリに追加された新しいファイルまたはディレクトリは、すべて DevTest クラスパスに動的に追加されます。

incontainer

コンテナ内テスト手順が含まれます。

jre

必要な JRE が含まれます。

lib

必要な JAR ファイルが含まれます。

ライセンス

DevTest Solutions の実行に必要なライセンス ファイルが含まれます。

locks

プロセスの並列処理に使用されるロック ファイルが含まれます。

レポート

DevTest によって作成される XML ベースのテスト レポートが含まれます。

snmp

SNMP 関連ファイルが含まれます。

tmp

DevTest によって作成されるログ ファイルが含まれます。 問題についてサポートに連絡する場合、このディレクトリから 1 つ以上のファイルを送信するように求められることがあります。

umetrics

メトリックの収集に関連するファイルが含まれます。

webserver

Web サーバ ファイルが含まれます。

DevTest ワークステーション でプロジェクトを作成すると、**Projects** ディレクトリが追加されます。

サービスとしてのサーバ コンポーネントの実行

サーバ コンポーネントを常に実行したままにしておく場合は、**LISA_HOME¥bin** ディレクトリ内のサービス実行可能ファイルを使用できます。

特定の名前なしでコマンド ラインから開始した場合、サーバ コンポーネントのデフォルトの名前が使用されます。 lisa.properties ファイルは、以下のプロパティおよびデフォルト値でインストールされます。

- lisa.registryName=Registry
- lisa.coordName=Coordinator
- lisa.simulatorName=Simulator
- lisa.vseName=VSE

これらのプロパティのいずれかのデフォルト値を上書きする場合は、**local.properties** ファイルで新しいプロパティ値を指定します。

以下に示された順番でコンポーネントを起動および停止することをお勧めします。

サービスとしてのサーバ コンポーネントを起動する方法

以下のコマンドを入力します。

```
EnterpriseDashboardService start
RegistryService start
PortalService start
BrokerService start
CoordinatorService start
SimulatorService start
VirtualServiceEnvironmentService start
```

サービスとしてのサーバ コンポーネントを停止する方法

以下のコマンドを入力します。

```
SimulatorService stop
CoordinatorService stop
VirtualServiceEnvironmentService stop
```

```
BrokerService stop
PortalService stop
RegistryService stop
EnterpriseDashboardService stop
```

UNIX で自動的に開始するサービスを設定するには、システム管理者に問い合わせてください。

DevTest ワークステーション がレポート データベース（Derby）を起動した場合は、DevTest ワークステーション がシャットダウンするとレポート データベースもシャットダウンします。

コーディネータがデータベース（Derby）を起動した場合は、コーディネータがシャットダウンしてもデータベースはシャットダウンしません。

Ant および JUnit による DevTest Solutions の実行

Ant ビルドの実行パス内で JUnit テストとして DevTest テストを実行できます。

この機能は、現実的な自動ビルドおよびテストの統合機会を提供します。Ant の柔軟性と JUnit の簡潔性を DevTest Solutions の機能と共に活用することにより、以下の自動化ニーズを満たすことができます。

- 統合
- ロード
- ストレス
- 実稼働環境のモニタ

Ant は、Java ベースのオープン ソース自動ソフトウェア ビルド ツールです。 Ant は、JUnit などの多くのサードパーティ ツールをサポートするように拡張されています。 詳細については、http://ant.apache.org/ を参照してください。

JUnit は、Java ベースのオープン ソース ユニットテスト フレームワークです。JUnit は、DevTest Solutions などのサードパーティ テスト ツールによって広くサポートされています。 詳細については、http://www.junit.org を参照してください。

- JUnit テスト ケース/スイートの実行ステップにより、JUnit テストを実行できます。 DevTest Ant タスク (P. 24)は、まるで JUnit テストであるかのように、すべての DevTest テストまたはスイートを実行できます。
- JUnit レポートおよびログ ファイルは、テストを実行するために **junitlisa** Ant タスクが使用される場合にのみ生成されます。
- JUnit ステップは、既存の JUnit テスト ケースのテスト ワークフローへの統合を可能にします。DevTest を使用して、JUnit テストをラップし、DevTest を介してそれらのテストを実行できます。
- **junitlisa** タスクは、DevTest ユーザ インターフェースを介さないテストの自動化を可能にします。結果の送信先となるユーザ インターフェースが存在しないため、**junitlisa** タスクは HTML レポートを生成します。

JUnit テストとしての DevTest テストの実行

JUnit ステップは、JUnit3 および JUnit4 のテスト ケース/テスト スイートをサポートします。ここでは、JUnit テストとして DevTest テストを実行し、ネイティブ JUnit テストとしてレポートを出力するために必要な手順について説明します。

次の手順に従ってください:

1. PC で Ant と JUnit の両方を使用でき、PATH に ANT_HOME¥bin が設定されていることを確認します。
2. JUnit インストールから ANT_HOME¥lib に **junit.jar** をコピーします。
3. システム プロパティ **LISA_HOME** を定義し、その値を DevTest インストール ディレクトリに設定します。
4. **junitlisa** タスクを使用して、JUnit テストとして DevTest テストを実行します。
5. （オプション） **junitlisareport** タスクを使用して、JUnit の XML 出力から HTML ベースのレポートを作成します。

ログ出力は、**user.home¥lisatmp** ディレクトリの **junitlisa_log.log** ファイルに書き込まれます。

使用されるログ レベルは、**LISA_HOME¥logging.properties** で設定されるレベルと同じです。

ログ レベルを変更する方法

このファイルの 1 行目（以下の行）を編集します。

```
log4j.rootCategory=INFO,A1
```

以下のように変更してください。

```
log4j.rootCategory=DEBUG,A1
```

JUnit の標準出力は、**user.home¥lisatmp¥junit¥index.html** にあります。

DevTest Solutions Ant タスク

DevTest Solutions には、以下の Ant タスクが含まれます。

- junitlisa（P. 25）
- junitlisareport（P. 28）

junitlisa Ant タスク

junitlisa タスクは、Ant で使用可能な JUnit タスクに置き換わる「簡易」バージョンですが、JUnit テストの代わりに DevTest テストを実行します。 ほとんどの継続的なビルド システムは、XML 出力ファイルを認識して、ビルド ダッシュボードをテスト結果と統合します。

このタスクは、JUnit タスクの直接のサブクラスです。そのため、junitlisa タスクは、JUnit タスクと同じ属性およびネストされたエレメントを持ちます。 ただし、以下の動作の違いに注意してください。

- **fork** 属性を false に設定できません。
- ネストされた **test** エレメントを追加できません。 代わりに、**test** 属性を使用します。
- ネストされた **batchtest** エレメントを追加できません。代わりに、**suite** 属性を使用します。
- **LISA_HOME¥bin¥*.jar**、**LISA_HOME¥lib¥*.jar, *.zip**、および **LISA_HOME¥lib¥endorsed¥*.jar** から構成されるクラスパスが暗黙的に追加されます。
- **LISA_HOME¥lib¥endorsed** を指している **java.endorsed.dirs** システム プロパティが暗黙的に追加されます。
- フォーマッタが指定されない場合は、**xml** タイプのデフォルトのフォーマッタが追加されます。
- **printsummary** 属性は、デフォルトで true になります。
- **maxmemory** 属性は、デフォルトで 1024m になります。
- **showoutput** 属性は、デフォルトで true になります。

JUnit タスクから継承された属性に加えて、**junitlisa** タスクには以下の属性があります。

suite

スイート ドキュメントのファイル名。

例： suite="AllTestsSuite.ste"

test

テスト ケースのファイル名。

例： test="multi-tier-combo.tst"

stagingDoc

ステージング ドキュメントのファイル名。

例： stagingDoc="Run1User1Cycle.stg"

config

名前付きの内部設定セットまたはファイル名。

例： config="project.config"

outfile

レポート データを書き込むために使用されるファイル名。 値が junitlisareport の標準の命名規則に準拠しない場合は、完全に設定された junitreport タスクを代わりに指定します。

例： outfile="report"

registry

テスト ケースをリモートでステージングするときに使用するレジストリへのポインタ。

例： registry="tcp://testbox:2010/Registry"

preview

テスト ケースを実行せずに、各テスト ケースの名前および説明を書き出せるようにします。

例： preview="true"

user

ACL のユーザ名を指定します。

例： user="admin"

password

ACL のパスワードを指定します。

例： password="admin"

mar

MAR ドキュメントのファイル名。

例： mar=”example.mar”

mari

MAR 情報ドキュメントのファイル名。

例： mari=”example.mari”

junitlisa タスクには、**lisatest** という名前のネストされたエレメントが含まれます。 このエレメントには、以下の属性があります。

suite

スイート ドキュメントのファイル名。

例： suite="AllTestsSuite.ste"

test

テスト ケースのファイル名。

例： test="multi-tier-combo.tst"

stagingDoc

ステージング ドキュメントのファイル名。

例： stagingDoc="Run1User1Cycle.stg"

mar

MAR ドキュメントのファイル名。

例： mar=”example.mar”

mari

MAR 情報ドキュメントのファイル名。

例： mari=”example.mari”

属性値は中かっこを使用でき、通常の方法で解決されます。

少なくとも 1 つのテストまたはスイートを指定する必要があります。テストまたはスイートは、ネストされた **lisatest** エレメント、あるいは **test** または **suite** 属性に指定できます。 **lisatest** エレメントを追加することにより、複数のテストおよびスイートを指定できます。 テストおよびスイートは、XML に記述されている順に実行されます。

test 属性を持つ単一のテストを実行する場合、デフォルトの動作は以下のとおりです。

- 単一の仮想ユーザを使用したステージング
- 1 回実行
- 100 パーセントの反応時間

このデフォルトの動作を変更するには、スイート内でテストをラップし、別のステージング ドキュメントを指定します。

junitlisareport Ant タスク

junitlisareport タスクまたは標準の **junitreport** タスクを使用して、HTML ベースのレポートを作成できます。

junitlisareport タスクは、適切なデフォルトが指定されていることを除き、標準の **junitreport** エレメントのサブクラスです。 これは以下のコードと同等です。

```
<junitreport todir="${testReportDir}">

   <fileset dir="<todir specified in the junitreport tag>">
    <include name="TEST-*.xml"/>
  </fileset>

   <report format="frames" todir="<todir specified in the junitreport tag>"/>

</junitreport>
```

独自のファイル セットおよびレポートを指定できます。 タスクが **junitreport** の直接のサブクラスであるため、**junitreport** が持つ属性およびネストされたエレメントがすべてサポートされます。

ほとんどの場合、継承された **toDir** 属性を指定することをお勧めします。デフォルトでは、現在の作業ディレクトリになります。

Ant および JUnit の使用例

例: 完全な Ant ビルド ファイル

LISA_HOME¥examples ディレクトリ内の **build.xml** ファイルは完全な Ant ビルド ファイルです。

このビルド ファイルには 2 つのターゲットが含まれます。

- **lisaTests** ターゲットは、JUnit テストとしてスイートを実行します。 スイートは、ネストされた **lisatest** エレメントで指定されます。
- **oneTest** ターゲットはテスト ケースを JUnit テストとして実行します。 テスト ケースは、**test** 属性で指定されます。

ビルド ファイルは、アクセス制御（ACL）が有効になっていると仮定します。 そのため、**user** 属性および **password** 属性が含まれています。

例: テスト ケースの junitlisa タスク

以下の **junitlisa** タスクは、リモート レジストリで単一のテスト ケースを実行するように設定されています。

```
<junitlisa test="MyTest.tst"
    config="dev"
    registry="tcp://testbox:2010/Registry"
    toDir="${testReportDir}"
    haltOnError="no"
    errorProperty="test.failure">
    <jvmarg value="-DmySystemProp=someValue"/>
  </junitlisa>
```

JUnit の使用例

JUnit3 テスト ケース (P. 30)

JUnit3 テスト スイート (P. 30)

JUnit4 テスト ケース (P. 31)

JUnit4 テスト スイート (P. 31)

JUnit3 テスト ケース

JUnit3 テスト ケースについては、テスト ケースの作成に関する JUnit の規則に従ってください。 たとえば以下を実行します。

- テスト クラスは junit.framework.TestCase を拡張する必要がある。
- メソッド名は「test」で始まる。

たとえば、以下のようになります。

```
import junit.framework.TestCase;

public class JUnit3TestCase extends TestCase {

   public void testOneIsOne() {
      assertEquals (1, 1);
   }

   public void testTwoIsThree() {
      assertEquals (2, 3);
   }

}
```

JUnit3 テスト スイート

JUnit3 テスト スイートの場合、スイートは junit.framework.TestSuite を拡張する必要はありません。ただし、テスト ケースが JUnit4TestAdapter でラップされた suite() メソッドを実装する必要があります。

たとえば、以下のようになります。

```
import junit.framework.Junit4TestAdapter;
import junit.framework.TestSuite;

public class JUnit3VanillaTestSuite {

   public static TestSuite suite() {
      TestSuite suite = new TestSuite();
      suite.addTest ( new JUnit4TestAdapter ( MyJUnit3TestCase.class) ) ;
      return suite;
   }

}
```

JUnit4 テスト ケース

JUnit4 テスト ケースの場合、テスト メソッドには、JUnit4 で必要なアノテーション「@org.junit.Test」が必要です。

たとえば、以下のようになります。

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class JUnit4TestCase {

    @Test
    public void oneIsOne() { assertEquals (1, 1); }

    @Test
    public void twoIsThree() { assertEquals (2, 3); }

}
```

JUnit4 テスト スイート

JUnit4 テスト スイートを実装するには、クラスにテスト スイートのフラグを立てるために @RunWith および @Suite.SuiteClasses アノテーションを追加します。

たとえば、以下のようになります。

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses ( { JUnit4TestCase.class } )

public class JUnit4VanillaTestSuite { // empty }
```

注: JUnit テストをロードすることによって JUnit ステップに関する IllegalArgumentException が返される場合は、クラス名の末尾に **.class** が追加されていることを確認します。 クラス名のスペルを手動で確認するか、またはクラスパス ブラウザを使用して class を見つけることができます。

CruiseControl との統合

継続的な統合を提供するために CruiseControl を使用している場合、そのコントロール パネル内にテストの失敗を含めるように設定できます。

「Ant および JUnit の使用例 (P. 29)」で説明されている JUnit Ant タスクでは、**${testReportDir}** は **${LISA_HOME}/bin/lisa-core.jar** として定義されます。

以下のサンプルは、標準的な **CruiseControl config.xml** を示しています。

```
<project name="myproj " buildafterfailed="false">
  <log>
    <merge dir="/home/cruise/build"/>
    <merge dir="/path/to/lisajunit/output"/>
  </log>
.
.
.
</project>
```

その他のサンプル ビルド ファイル

「Ant および JUnit の使用例 (P. 29)」で説明されている **build.xml** ファイルに加えて、**LISA_HOME¥examples** ディレクトリには DevTest プロジェクトの自動化に関する以下のサンプル ファイルが含まれています。

- **automated-build.xml**：プロジェクト階層の最上位に配置するマスタ ビルド ファイル
- **lisa-project-build.xml**：各プロジェクトに配置するビルド ファイル。ファイル名を **build.xml** に変更する必要があります。
- **common.xml**：ルートおよびプロジェクトの間の各サブディレクトリに配置するファイル
- **common-macros.xml**：このファイルには、DevTest サーバ コンポーネントを使用してさまざまなタスクを実行するためのマクロが含まれています。たとえば、レジストリを起動するか、スイートを実行するか、または仮想サービスを展開できます。 このファイルは、スタンドアロンのビルド ファイルではなく、既存の Ant ベース フレームワークに組み込まれます。

詳細については、各ファイル内のコメントを参照してください。

メモリ設定

Windows オペレーティング システムの場合、DevTest ワークステーションのデフォルトのメモリ制限は 512 MB （-Xmx512m）です。

Windows システムでは、必要に応じてより多くのメモリを利用できるように、DevTest サーバ を Windows 64 ビット上で使用することをお勧めします。

メモリ割り当ての変更

vmoptions ファイルは、より多くのパラメータを Java プロセスへ渡して JVM で使用されるデフォルト設定を変更するために使用されます。これらのファイルは、サーバで使用される各 DevTest プロセスのメモリ割り当て設定をカスタマイズするために使用されます。 DevTest サーバ をインストールすると、LISA_HOME¥bin フォルダに、各実行可能ファイルとして同じ名前の .vmoption ファイルが配置されます。 JVM パラメータの全リストは、Oracle Web サイトの Configuring the Default JVM and Java Arguments で詳しく説明されています。

Windows および UNIX 上で Java ヒープ サイズを変更する方法

1. 拡張子が **.vmoptions** のターゲット ファイルを開きます。

   たとえば、以下の内容を含む **RegistryService.vmoptions** を開きます。

   ```
   # Enter one VM parameter per line
   # For example, to adjust the maximum memory usage to 512 MB, uncomment the following line:
   # -Xmx512m
   # To include another file, uncomment the following line:
   # -include-options [path to other .vmoption file]
   ```

2. 割り当てられる最大メモリ容量（Xmx）を変更するには、以下の行のコメントを外します。

   ```
   # -Xmx512m
   ```

3. 割り当てられる最小メモリ容量（Xms）を変更するには、別の行に VM 引数を追加します。

   ```
   -Xms128M
   ```

   これらの操作により、ファイルはメモリの割り当て範囲を以下の例に示すように指定します。

   ```
   -Xms128M
   -Xmx512M
   ```

4. このテキスト ファイルを **LISA_HOME¥bin** フォルダに保存します。

Mac OS X 上で Java ヒープ サイズを変更する方法

LISAWorkstation.app を参照し、**Info.plist** ファイルを編集します。

パスは **LisaHome/bin/LISAWorkstation.app/Contents/Info.plist** です。 **Info.plist** ファイルを変更できます。ただし、変更は DevTest ワークステーション にのみ適用されます。

個々のプロセス用のメモリを調整する方法

以下のプログラム用のメモリを調整できます。

- ブローカ
- BrokerService
- CoordinatorServer
- CoordinatorService
- Registry
- RegistryService
- シミュレータ
- SimulatorService
- VirtualServiceEnvironment
- VirtualServiceEnvironmentService
- ワークステーション

すべてのプロセスで、各プロセスのメモリを調整するために、個々のスクリプト ファイルを編集できます。たとえば、レジストリ用に 128M、シミュレータ用に 1024M を割り当てることができます。

DevTest Solutions を再インストールすると、編集内容がインストーラによって上書きされます。

これらのファイルをコピーして、コピーを編集することもできます。 たとえば、MyRegistry に Registry をコピーし、MyRegistry に若干の変更を加えることができます。

Third-Party ファイル要件

さまざまなサードパーティ アプリケーションと共に DevTest Solutions を使用するには、DevTest に使用可能なサードパーティ アプリケーションから JAR ファイルを作成する必要があります。以下のセクションでは、特に指定のない限り、これらの方法のどれを使用してもかまいません。

- **LISA_HOME¥hotDeploy** ディレクトリにファイルを配置する
- **LISA_HOME¥lib** ディレクトリにファイルを配置する
- **LISA_POST_CLASSPATH** 変数を定義する

以下の例は、Windows コンピュータ上の **LISA_POST_CLASSPATH** 変数を示します。 この例のファイルは WebSphere MQ ファイルです。

```
LISA_POST_CLASSPATH="C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mq.commonservices.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mq.headers.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mq.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mq.jmqi.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mq.pcf.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥com.ibm.mqjms.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥connector.jar;C:¥Program
Files¥IBM¥MQSeries¥Java¥lib¥dhbcore.jar"
```

CA は、このトピックで示されているサードパーティ ファイルを提供しません。

注: このトピックでは、DevTest Solutions のインストール中にローカル インストール タイプを選択したと仮定します。共有インストール タイプ (P. 13)を選択した場合、**hotDeploy** ディレクトリは **LISA_HOME** 以外の場所に存在する可能性があります。

JCAPS ファイル要件

JCAPS メッセージング（ネイティブ）ステップを使用する場合、必要になる可能性がある JAR ファイルについては、JCAPS のドキュメントを参照してください。

JCAPS メッセージング（JNDI）ステップを使用する場合は、**com.stc.jms.stcjms.jar** ファイルが必要です。 必要になる可能性があるその他の JAR ファイルについては、JCAPS のドキュメントを参照してください。

JAR ファイルは、JCAPS インストールの **lib** ディレクトリにあります。

JMS メッセージング ファイル要件

JMS メッセージング（JNDI）ステップを使用する場合、必要になる可能性がある JAR ファイルについては、JMS プロバイダのドキュメントを参照してください。

Oracle OC4J ファイル要件

以下の JAR ファイルが必要です。

- dms.jar
- oc4j.jar
- oc4jclient.jar

必要になる可能性がある JAR ファイルの詳細については、OC4J のドキュメントを参照してください。 JAR ファイルは、OC4J インストールの **lib** ディレクトリにあります。

SAP ファイル要件

以下の JAR ファイルが必要です。

- sapjidoc3.0.8/sapidoc3.jar - SAP IDoc ステップおよび仮想化に必要
- sapjco3.0.9/sapjco3.jar
- sapjco3.0.9/*platform*/ プラットフォームのネイティブ ライブラリ ファイル（*platform* は使用するコンピュータ システム（osx_64、windows_x86 など））- SAP RFC ステップ、SAP IDoc ステップおよび仮想化で必要

<SAP> IDoc 仮想化の要件

1. クライアントおよびサーバの両方の SAP システムに、タイプ T の RFC 接続先を作成します。 DevTest は、この RFC 接続先を使用して、両方のシステムから IDoc を受信します。

2. クライアントおよびサーバ SAP システム上の送信パートナー プロファイルを更新します。 手順 1 で作成した RFC 接続先に関連付けられているポート番号に送信するように、それぞれの IDoc タイプ用のパートナー プロファイル内の送信パラメータを更新します。これによって、DevTest がクライアントおよびサーバ SAP システムから IDoc を受信できるようになります。
3. Data ディレクトリのプロジェクト内に、接続プロパティ ファイルを作成してインポートします。
    a. 手順 1 で作成したクライアント RFC 接続先に対して、RFC 接続（.jcoServer 内のプロパティを使用）を作成します。
    b. クライアント SAP システムに対して、システム接続プロパティ ファイル（.jcoDestination 内のプロパティを使用）を作成します。
    c. 手順 1 で作成したサーバ RFC 接続先に対して、RFC 接続（.jcoServer 内のプロパティを使用）を作成します。
    d. サーバ SAP システムに対して、システム接続プロパティ ファイル（.jcoDestination 内のプロパティを使用）を作成します。

レコーディングを開始する前に、**Data** ディレクトリのプロジェクト内に、接続プロパティ ファイルを作成してインポートする必要があります。 これらのファイルは、JCo サーバを起動してクライアントおよびサーバ SAP システムに対して IDoc の受信および IDoc の転送を行うために必要です。 RFC 接続（.jcoServer 内のプロパティを使用）およびシステム接続（.jcoDestination 内のプロパティを使用）プロパティ ファイルが必要です。 これらのファイルは、JCo サーバを起動してクライアントおよびサーバ SAP システムに対して IDoc の受信および IDoc の転送を行うために必要です。 これらのプロパティ ファイルの作成については、SAP 管理者に問い合わせてください。

<SAP> RFC 仮想化の要件

1. クライアント SAP システムに、タイプ T の RFC 接続先を作成します。 DevTest は、この RFC 接続先を使用して、クライアント システムが行うリモート ファンクション コールをインターセプトします。
2. リモート ファンクション コールが新しい接続先を使用するように、ABAP コードを更新します。
3. Data ディレクトリのプロジェクト内に、接続プロパティ ファイルを作成してインポートします。これらのプロパティ ファイルの作成については、SAP 管理者に問い合わせてください。

a. 手順 1 で作成したクライアント RFC 接続先に対して、RFC 接続（.jcoServer 内のプロパティを使用）を作成します。 このファイルでは、jco.server.repository_destination を指定しないでください。

b. RFC リポジトリとして機能するシステムに対して、リポジトリ接続プロパティ ファイル（.jcoDestination 内のプロパティを使用）を作成します。 これは通常、RFC が実行されたシステムと同じです。

c. RFC が実行されるシステムに対して、システム接続プロパティ ファイル（.jcoDestination 内のプロパティを使用）を作成します。これがリポジトリ接続と同じである場合、1 つのプロパティ ファイルのみ必要です。

SonicMQ ファイル要件

以下の JAR ファイルが必要です。

- mfcontext.jar
- sonic_Client.jar
- sonic_XA.jar

JAR ファイルは、SonicMQ インストールの **lib** ディレクトリにあります。

TIBCO ファイル要件

TIBCO の要件は、アプリケーションによって異なります。

TIBCO JAR ファイルを **LISA_HOME¥lib** ディレクトリにコピーするか、または **LISA_POST_CLASSPATH** 変数を定義します。 このファイルを **LISA_HOME¥hotDeploy** ディレクトリにコピーしないでください。

TIBCO Rendezvous メッセージング

以下の JAR ファイルが必要です。

- tibrvj.jar
- tibrvjms.jar
- tibrvjsd.jar
- tibrvjweb.jar

- tibrvnative.jar
- tibrvnativesd.jar

TIBCO Rendezvous.dll ファイルも必要です。 TIBCO Rendezvous の **bin** ディレクトリから **LISA_HOME¥bin** ディレクトリにすべての .dll ファイルをコピーします。 PATH 環境変数で **LISA_HOME¥bin** の場所を参照します。

また、**PATH** 環境変数に TIBCO Rendezvous の **bin** ディレクトリを追加します。

TIBCO EMS メッセージングまたは TIBCO ダイレクト JMS

以下の JAR ファイルが必要です。

- tibjms.jar
- tibjmsadmin.jar
- tibjmsapps.jar
- tibrvjms.jar

TIBCO Hawk メトリック

以下の JAR ファイルが必要です。

- console.jar
- talon.jar
- util.jar

どのトランスポートを TIBCO Hawk が使用しているかによって、TIBCO Rendezvous JAR ファイルまたは TIBCO EMS JAR ファイル（あるいはその両方）を **LISA_HOME¥lib** ディレクトリにコピーする必要もあります。

WebLogic ファイル要件

weblogic.jar ファイルが必要です。 セキュリティまたは JMX を使用している場合は、その他の JAR ファイルが必要になる可能性があります。

必要になる可能性がある JAR ファイルの詳細については、WebLogic のドキュメントを参照してください。JAR ファイルは、WebLogic インストールの **lib** ディレクトリにあります。

webMethods ファイル要件

以下の JAR ファイルが必要です。

- （webMethods Integration Server 8.2）webMethods インストール ディレクトリから DevTest installation_directory¥lib¥shared ディレクトリに wm-isclient.jar をコピーします。
- （webMethods Integration Server 7.1 以降）installation_directory¥lib¥shared¥wm-isclient.jar
- （webMethods Integration Server 7.0 以前）installation_directory¥lib¥client.jar
- wm-enttoolkit.jar
- wmbrokerclient.jar
- wmjmsadmin.jar
- wmjmsclient.jar
- wmjmsnaming.jar

JAR ファイルは、webMethods インストールの **lib** ディレクトリにあります。

WebSphere MQ ファイル要件

WebSphere MQ JAR ファイルを **LISA_HOME¥lib** ディレクトリにコピーするか、または **LISA_POST_CLASSPATH** 変数を定義します。 このファイルを **LISA_HOME¥hotDeploy** ディレクトリにコピーしないでください。

注: オペレーティング システムが日本語バージョンである場合は、WebSphere MQ 7 用に記載されているファイルを使用します。

WebSphere MQ 5.2 の場合は、以下の JAR ファイルが必要です。

- com.ibm.mqjms.jar
- com.ibm.mqbind.jar
- com.ibm.mq.pcf.jar
- com.ibm.mq.jar
- connector.jar

WebSphere MQ 6 の場合は、以下の JAR ファイルが必要です。

- com.ibm.mq.jar

- com.ibm.mq.pcf.jar
- com.ibm.mqjms.jar
- connector.jar
- dhbcore.jar

WebSphere MQ 7 の場合は、以下の JAR ファイルが必要です。

- com.ibm.mq.commonservices.jar
- com.ibm.mq.headers.jar
- com.ibm.mq.jar
- com.ibm.mq.jmqi.jar
- com.ibm.mq.pcf.jar
- com.ibm.mqjms.jar
- connector.jar
- dhbcore.jar

JAR ファイルは、**MQ_HOME¥java¥lib** ディレクトリにあります。

追加スクリプト言語の有効化

ユーザが実行スクリプト テスト ステップ、スクリプタブル アサーション、一致スクリプト エディタ、およびスクリプタブル データ プロトコルでより多くの JSR-223 言語を使用できるようにするために、追加の言語を有効にできます。

追加のスクリプト言語を有効にするには、リモート コーディネータ、サーバ、または CA Service Virtualization の **hotdeploy** ディレクトリにその言語の jar ファイルを配置します。 追加した言語は、次のスタートアップで DevTest ワークステーション の［言語］ドロップダウン リストに表示されます。

第 2 章: データベース管理

このセクションには、以下のトピックが含まれています。

内部データベースの設定

組織でエンタープライズ データベース ソリューションへの移行の準備を行う際に、機能するシステムを持てるように、標準装備のデータベースとして Apache Derby が提供されます。 Derby はエンタープライズ データベース ソリューションとしてはサポートされていません。 標準装備のこのデータベースは、**LISA_HOME¥database¥lisa.db** ディレクトリにあります。手動でのデータ移行に伴う問題を防止するために、インストール後のタスクとしてエンタープライズ データベースを設定することを推奨します。

以下のコンポーネントがデータベースと対話します。

- レポート
- アクセス制御（ACL）
- Java エージェント ブローカ
- VSE
- エンタープライズ ダッシュボード

DevTest Solutions で使用するエンタープライズ データベースをバックアップするための、サイト固有の手順。

注: データベース システムの要件の詳細については、「*Installing*」 *を参照してください。*

外部レジストリ データベースの設定

LISA_HOME ディレクトリの **site.properties** ファイルを編集することにより、外部データベースを使用するように DevTest を設定できます。

注:

- データベース システムの要件の詳細については、「*Installing*」を参照してください。
- エンタープライズ ダッシュボード は、IBM DB2 *をサポートしません。*

LISA_HOME¥database ディレクトリには、サポートされている各データベースのテンプレート **site.properties** ファイルが含まれています。

- **db2-site.properties：** IBM DB2
- **derby-site.properties：** Derby
- **mysql-site.properties：** MySQL
- **oracle-site.properties：** Oracle
- **sqlserver-site.properties：** Microsoft SQL Server

DevTest サーバ を実行している場合、各 DevTest ワークステーション インストールを再設定する必要はありません。**site.properties** の設定は、各ワークステーション、VSE、コーディネータ、シミュレータ サーバ、およびレジストリに接続するその他の DevTest コンポーネントに伝達されます。

一般的な手順は以下のとおりです。

1. 適切なテンプレート **_site.properties** ファイルを **LISA_HOME** ディレクトリにコピーし、名前を **site.properties** に変更します。
2. 新しい **site.properties** ファイルを変更します。ターゲット データベースと一致するように、**Properties to configure** セクションのプロパティを変更してください。
3. Oracle、DB2、MySQL、SQL Server など、DevTest がサポートする外部データベースに設定する場合は、以下の手順に従います。
   a. 該当するセクションで、設定する外部データベースに対応する、以下のプロパティのコメントを外します。

      以下に例を示します。

      lisadb.pool.common.driverClass=oracle.jdbc.driver.OracleDriver
      lisadb.pool.common.url

lisadb.pool.common.url=jdbc:oracle:thin:@HOST:1521:SID

b. 適切なホスト名、ポートおよび SID 値で、**lisadb.pool.common.url** プロパティを更新します。

c. データベースにアクセスするための正しい値で、以下のユーザおよびパスワード プロパティを更新します。

lisadb.pool.common.user

lisadb.pool.common.password

d. Derby データベースでは、以下の 2 つのプロパティのコメントを外します。

#lisadb.pool.common.driverClass=org.apache.derby.jdbc.ClientDriver

#lisadb.pool.common.url=jdbc:derby://localhost:1528/database/lisa.db;create=true

e. 以下のプロパティを false に設定します。

lisadb.internal.enabled=false

4. レジストリまたは DevTest ワークステーション を起動します。

ACL テーブル

アクセス制御（ACL）機能は、以下のテーブルを使用します。

- ACL_ACTIVITIES
- ACL_ACTIVITIES_ITEMS
- ACL_ACTIVITY_LIST_ITEMS
- ACL_ACTIVITY_NUMERIC_LIMITS
- ACL_AUDIT_LOG
- ACL_CUSTOM_ACTIVITY_INFO
- ACL_NUMERIC_LIMITS
- ACL_RESOURCES
- ACL_RESOURCE_GROUPS
- ACL_RESOURCE_GROUPS_RESOURCES
- ACL_RESOURCE_GROUPS_ROLES
- ACL_ROLE_ACTIVITY_ITEM
- ACL_ROLE_CUSTOM_ACT_INFO

- ACL_ROLE_NUMERIC_LIMITS
- ACL_ROLES
- ACL_ROLES_ACTIVITIES
- ACL_USERS
- ACL_USERS_ROLES

DB2 を使用するための DevTest の設定

このトピックでは、IBM DB2 データベースを使用するために DevTest を設定する方法について説明します。

以下の手順で、DB2 バージョンの **site.properties** ファイルを設定します。このファイル内の以下のプロパティがデータベース設定に関連しています。

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.poolName=common

lisadb.pool.common.driverClass=com.ibm.db2.jcc.DB2Driver
lisadb.pool.common.url=jdbc:db2://HOSTNAME:PORT/DATABASENAME
lisadb.pool.common.user=database_username
lisadb.pool.common.password=database_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

データベースへの接続数を最小限に抑えるために、接続プールが使用されます。プール機能の基盤となる実装は c3p0 です。設定の詳細については、http://www.mchange.com/projects/c3p0/index.html#configuration_properties を参照してください。 デフォルトでは、すべてのコンポーネントが共通の接続プールを使用します。 ただし、コンポーネントごとに個別のプールを定義したり、組み合わせたりできます。

レポート データベースの場合は、最適なパフォーマンスを得るために少なくとも 10 GB を確保しておくことをお勧めします。 VSE 用のデータベースは、6.0 より前のリリースで作成されたレガシー イメージを操作する場合にのみ必要です。

password で終わるすべてのプロパティの値は、起動時に自動的に暗号化されます。

注: DevTest ワークステーション、コーディネータ、シミュレータ サーバ、VSE、またはその他のリモート DevTest コンポーネントのリモート インストールは、これ以上の設定を必要としません。 これらはすべて、データベース アクセスを接続および設定するときに、レジストリから **site.properties** を受信します。

次の手順に従ってください:

1. DB2 データベースのコード ページが 1208 であることを確認します。
2. DB2 データベースのページ サイズが少なくとも 8 KB であることを確認します。
3. **LISA_HOME¥database** ディレクトリから **LISA_HOME** ディレクトリに **db2-site.properties** ファイルをコピーします。
4. ファイル名を **site.properties** に変更します。
5. **site.properties** ファイルを開きます。
6. データベース設定プロパティを設定します。

   通常は、以下のプロパティを更新します。

   - lisadb.pool.common.url
   - lisadb.pool.common.user
   - lisadb.pool.common.password
7. レジストリが初めて起動されると、スキーマはデータベースに自動的に作成されます。 ただし、DevTest ユーザが DBA 権限を持つことを望まない場合は、前もってスキーマを手動で作成できます。 **LISA_HOME¥database** ディレクトリの **db2.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ベースとして利用できる SQL ステートメントが含まれています。
8. **lisadb.internal.enabled** プロパティを false に設定します。
9. DB2 JDBC ドライバを、**LISA_HOME¥lib¥shared** ディレクトリと **LISA_HOME¥webserver¥phoenix¥phoenix-1.0.0¥WEB-INF¥lib** ディレクトリに追加します。

   重要: JDBC ドライバは、両方のディレクトリに追加する必要があります。
10. レジストリを起動します。

MySQL を使用するための DevTest の設定

このトピックでは、MySQL データベースを使用するために DevTest を設定する方法について説明します。

以下の手順で、MySQL バージョンの **site.properties** ファイルを設定します。このファイル内の以下のプロパティがデータベース設定に関連しています。

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.poolName=common

lisadb.pool.common.driverClass=com.mysql.jdbc.Driver
lisadb.pool.common.url=jdbc:mysql://DBHOST:DBPORT/DBNAME
lisadb.pool.common.user=database_username
lisadb.pool.common.password=database_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

データベースへの接続数を最小限に抑えるために、接続プールが使用されます。プール機能の基盤となる実装は c3p0 です。設定の詳細については、http://www.mchange.com/projects/c3p0/index.html#configuration_properties を参照してください。 デフォルトでは、すべてのコンポーネントが共通の接続プールを使用します。 ただし、コンポーネントごとに個別のプールを定義したり、組み合わせたりできます。

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。 データベース用のデフォルト コード ページは UTF-8 である必要があります。ご使用のデータベースを UTF-8 として定義するだけでは不十分です。MySQL データベースが UTF-8 でない場合は、ランタイム エラーが発生します。

レポート データベースの場合は、最適なパフォーマンスを得るために少なくとも 10 GB を確保しておくことをお勧めします。 VSE 用のデータベースは、6.0 より前のリリースで作成されたレガシー イメージを操作する場合にのみ必要です。

password で終わるすべてのプロパティの値は、起動時に自動的に暗号化されます。

注: DevTest ワークステーション、コーディネータ、シミュレータ サーバ、VSE、またはその他のリモート DevTest コンポーネントのリモート インストールは、これ以上の設定を必要としません。 これらはすべて、データベース アクセスを接続および設定するときに、レジストリから **site.properties** を受信します。

次の手順に従ってください:

1. **LISA_HOME¥database** ディレクトリから **LISA_HOME** ディレクトリに **mysql-site.properties** ファイルをコピーします。
2. ファイル名を **site.properties** に変更します。
3. **site.properties** ファイルを開きます。
4. データベース設定プロパティを設定します。

   通常は、以下のプロパティを更新します。

   - lisadb.pool.common.url
   - lisadb.pool.common.user
   - lisadb.pool.common.password
5. レジストリが初めて起動されると、スキーマはデータベースに自動的に作成されます。 ただし、DevTest ユーザが DBA 権限を持つことを望まない場合は、前もってスキーマを手動で作成できます。 **LISA_HOME¥database** ディレクトリの **mysql.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ベースとして利用できる SQL ステートメントが含まれています。
6. **lisadb.internal.enabled** プロパティを false に設定します。
7. MySQL JDBC ドライバを、**LISA_HOME¥lib¥shared** ディレクトリと **LISA_HOME¥webserver¥phoenix¥phoenix-1.0.0¥WEB-INF¥lib** ディレクトリに追加します。MySQL JDBC ドライバの最小バージョンは 5.1.25 です。

   重要: JDBC ドライバは、両方のディレクトリに追加する必要があります。
8. レジストリを起動します。

Oracle を使用するための DevTest の設定

このトピックでは、Oracle データベースを使用するために DevTest を設定する方法について説明します。

以下の手順で、Oracle バージョンの **site.properties** ファイルを設定します。このファイル内の以下のプロパティがデータベース設定に関連しています。

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.poolName=common

lisadb.pool.common.driverClass=oracle.jdbc.driver.OracleDriver
lisadb.pool.common.url=jdbc:oracle:thin:@HOST:1521:SID
lisadb.pool.common.user=oracle_username
lisadb.pool.common.password=oracle_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

データベースへの接続数を最小限に抑えるために、接続プールが使用されます。プール機能の基盤となる実装は c3p0 です。設定の詳細については、http://www.mchange.com/projects/c3p0/index.html#configuration_properties を参照してください。 デフォルトでは、すべてのコンポーネントが共通の接続プールを使用します。 ただし、コンポーネントごとに個別のプールを定義したり、組み合わせたりできます。

レポート データベースの場合は、最適なパフォーマンスを得るために少なくとも 10 GB を確保しておくことをお勧めします。 VSE 用のデータベースは、6.0 より前のリリースで作成されたレガシー イメージを操作する場合にのみ必要です。

password で終わるすべてのプロパティの値は、起動時に自動的に暗号化されます。

注: DevTest ワークステーション、コーディネータ、シミュレータ サーバ、VSE、またはその他のリモート DevTest コンポーネントのリモート インストールは、これ以上の設定を必要としません。 これらはすべて、データベース アクセスを接続および設定するときに、レジストリから **site.properties** を受信します。

次の手順に従ってください:

1. Oracle データベースの文字セットが Unicode をサポートしていることを確認します。
2. **LISA_HOME¥database** ディレクトリから **LISA_HOME** ディレクトリに **oracle-site.properties** ファイルをコピーします。
3. ファイル名を **site.properties** に変更します。
4. **site.properties** ファイルを開きます。
5. データベース設定プロパティを設定します。

   通常は、以下のプロパティを更新します。

   - lisadb.pool.common.url
   - lisadb.pool.common.user
   - lisadb.pool.common.password

   lisadb.pool.common.url プロパティに対しては、サービス名を使用できます。
6. レジストリが初めて起動されると、スキーマはデータベースに自動的に作成されます。 ただし、DevTest ユーザが DBA 権限を持つことを望まない場合は、前もってスキーマを手動で作成できます。 **LISA_HOME¥database** ディレクトリの **oracle.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ベースとして利用できる SQL ステートメントが含まれています。
7. **lisadb.internal.enabled** プロパティを false に設定します。
8. Oracle JDBC ドライバを、LISA_HOME¥lib¥shared ディレクトリと **LISA_HOME¥webserver¥phoenix¥phoenix-x.0.0¥WEB-INF¥lib** ディレクトリに追加します。

このドライバは、http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html からダウンロードできます。より古いデータベース サーバを使用していても、最新のドライバをダウンロードすることをお勧めします。 ほとんどの場合、**ojdbc7.jar** は Java 1.7 用（DevTest に付属）の正しいドライバです。 Java 1.6 を使用している場合は、**ojdbc6.jar** を使用します。 Java 1.5 を使用している場合は、**ojdbc5.jar** を使用します。

重要: JDBC ドライバは、両方のディレクトリに追加する必要があります。

9. レジストリを起動します。

SQL Server を使用するための DevTest の設定

このトピックでは、Microsoft SQL Server データベースを使用するために DevTest を設定する方法について説明します。

以下の手順で、SQL Server バージョンの **site.properties** ファイルを設定します。 このファイル内の以下のプロパティがデータベース設定に関連しています。

```
lisadb.reporting.poolName=common
lisadb.vse.poolName=common
lisadb.acl.poolName=common
lisadb.broker.poolName=common

lisadb.pool.common.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
lisadb.pool.common.url=jdbc:sqlserver://SERVER:PORT;databaseName=DATABASENAME
lisadb.pool.common.user=database_username
lisadb.pool.common.password=database_password

lisadb.pool.common.minPoolSize=0
lisadb.pool.common.initialPoolSize=0
lisadb.pool.common.maxPoolSize=10
lisadb.pool.common.acquireIncrement=1
lisadb.pool.common.maxIdleTime=45
lisadb.pool.common.idleConnectionTestPeriod=5
```

データベースへの接続数を最小限に抑えるために、接続プールが使用されます。プール機能の基盤となる実装は c3p0 です。設定の詳細については、http://www.mchange.com/projects/c3p0/index.html#configuration_properties を参照してください。 デフォルトでは、すべてのコンポーネントが共通の接続プールを使用します。 ただし、コンポーネントごとに個別のプールを定義したり、組み合わせたりできます。

レポート データベースの場合は、最適なパフォーマンスを得るために少なくとも 10 GB を確保しておくことをお勧めします。 VSE 用のデータベースは、6.0 より前のリリースで作成されたレガシー イメージを操作する場合にのみ必要です。

password で終わるすべてのプロパティの値は、起動時に自動的に暗号化されます。

注: DevTest ワークステーション、コーディネータ、シミュレータ サーバ、VSE、またはその他のリモート DevTest コンポーネントのリモート インストールは、追加の設定を必要としません。 これらはすべて、データベース アクセスを接続および設定するときに、レジストリから **site.properties** を受信します。

次の手順に従ってください:

1. **LISA_HOME¥database** ディレクトリから **LISA_HOME** ディレクトリに **sqlserver-site.properties** ファイルをコピーします。
2. ファイル名を **site.properties** に変更します。
3. **site.properties** ファイルを開きます。
4. データベース設定プロパティを設定します。

   通常は、以下のプロパティを更新します。

   - lisadb.pool.common.url
   - lisadb.pool.common.user
   - lisadb.pool.common.password
5. レジストリが初めて起動されると、スキーマはデータベースに自動的に作成されます。 ただし、DevTest ユーザが DBA 権限を持つことを望まない場合は、前もってスキーマを手動で作成できます。 **LISA_HOME¥database** ディレクトリの **sqlserver.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ベースとして利用できる SQL ステートメントが含まれています。
6. **lisadb.internal.enabled** プロパティを false に設定します。
7. SQL Server 用の JDBC ドライバを、**LISA_HOME¥lib¥shared** ディレクトリと **LISA_HOME¥webserver¥phoenix¥phoenix-x.0.0¥WEB-INF¥lib** ディレクトリに追加します。

   重要: JDBC ドライバは、両方のディレクトリに追加する必要があります。
8. レジストリを起動します。

外部エンタープライズ ダッシュボード データベースの設定

エンタープライズ ダッシュボードは、デフォルトでは内部 Derby データベースを使用します。 このデータベースではなく、外部のデータベースを使用することをお勧めします。

次の手順に従ってください:

1. エンタープライズ ダッシュボードがインストールされているサーバにログオンします。
2. LISA_HOME に移動し、lisa.properties.ファイルを開きます。
3. ［Enterprise Dashboard Options］セクション（エンタープライズ ダッシュボードで内部 Derby データベースを使用するかどうかを指定するセクション）で、以下の行を見つけます。
   ```
   # Should we start the internal Derby DB instance in the Enterprise Dashboard?
   dradisdb.internal.enabled=true
   ```
4. 値を false に変更します。
   ```
   dradisdb.internal.enabled=false
   ```
5. 以下のプロパティのコメントを外します。
   ```
   # Internal Derby DB network interface to use (0.0.0.0 == all network interfaces)
   #dradisdb.internal.host=0.0.0.0
   # Internal Derby DB port number to use
   #dradisdb.internal.port=1530
   ```
6. **Oracle** データベースを使用するには、以下の例で示すように、次のプロパティを編集して値を外部データベースに設定します。
   ```
   lisadb.pool.dradis.driverClass=oracle.jdbc.driver.OracleDriver
   lisadb.pool.dradis.url=jdbc:oracle:thin:10.130.150.36:1521:ORCL
   lisadb.pool.dradis.user=dradis21
   lisadb.pool.dradis.password=dradis21
   ```
7. **SQL Server** データベースを使用するには、以下の例で示すように、次のプロパティを編集して値を外部データベースに設定します。
   ```
   lisadb.pool.dradis.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
   lisadb.pool.dradis.url=jdbc:sqlserver://SERVER:PORT;databaseName=DATABASENAME
   lisadb.pool.dradis.user=dradis21
   lisadb.pool.dradis.password=dradis21
   ```
8. **MySQL** データベースを使用するには、以下の例で示すように、次のプロパティを編集して値を外部データベースに設定します。
   ```
   lisadb.pool.dradis.driverClass=com.mysql.jdbc.Driver
   lisadb.pool.dradis.url=jdbc:mysql://DBHOST:DBPORT/DBNAME
   lisadb.pool.dradis.user=dradis21
   lisadb.pool.dradis.password=dradis21
   ```

9. 手順 6、7、または 8 で、使用する外部データベース タイプ用に編集したプロパティをコピーします。
10. lisa.properties ファイルを保存します。
11. local.properties を開き、ファイルの末尾にコピーした行を貼り付けます。
12. local.properties ファイルを保存します。

データベースの保守

さまざまなプロパティを使用して、データベース内のデータの量を管理できます。

- レポート プロパティについては、「自動レポート保守 (P. 58)」を参照してください。
- アクセス制御（ACL）プロパティの詳細については、「監査ログ エントリの自動削除 (P. 60)」を参照してください。
- CA Continuous Application Insight プロパティについては、「トランザクションの自動削除 (P. 61)」および「ケースの自動削除 (P. 62)」を参照してください。

自動レポート保守

バックグラウンドで実行される 2 つのプロセスがレポートに影響します。

- パフォーマンス サマリ計算機
- レポート クリーナ

パフォーマンス サマリ計算機

このプロセスは、テストまたはスイートの実行に関する統計を計算します。レポート ポータル内の多くのグラフがこのプロセスに依存しているため、これを無効にすることはできません。 パフォーマンス サマリ計算機のスケジュールは、以下のプロパティを変更することによって変更できます。

rpt.summary.initDelayMin=**7**

レジストリを起動した後、このプロセスを開始する前に何分待機するかを決定します。

rpt.summary.pulseMin=1

このプロセスの各実行間にどれだけの時間待機するかを決定します。

レポート クリーナ

このプロセスは、「期限切れになった」レポート実行をデータベースから削除します。

perfmgr.rvwiz.whatrpt.autoExpire=true

このプロパティを設定することにより、レポート クリーナを有効化または無効化できます。 無効にすると、レポート クリーナ プロセスは実行されず、残りのプロパティは無効になります。

perfmgr.rvwiz.whatrpt.expireTimer=30d

スイートまたはテストの有効期限の値を設定します。 この値より古くなったスイートまたはテストは削除されます。 このプロパティ値は、サフィックスが付いた数字です。 以下のサフィックスが有効です。デフォルトは **t** です。

- t = ミリ秒
- s = 秒

- m = 分
- h = 時間
- d = 日
- w = 週

テストおよびスイートは、作成された日の時刻とほぼ同じ時刻に削除されることに注意してください。 数時間を追加することにより、その日の間に実行される削除プロセスを最小限に抑えることができます。 ただし、削除プロセスが実行される時刻を確定する絶対的な方法はありません。

perfmgr.rvwiz.whatrpt.forceCompleteTimer=24h

テストを強制的に終了させます（レポート データベース内）。 正常に完了していない一部のスイートまたはテストは、データベース内で「終了」としてマークされません。 「終了」としてマークされていないテストはレポート クリーナによって削除されず、パフォーマンス サマリ計算は実行されません。 この値よりも古いすべてのテストは、データベース内で完了としてマークされます。完了したテストは、expireTimer 値に達した後、初めてデータベースから削除されます。デフォルトで 24 時間より長い時間実行されるテストがある場合は、このプロセスの値を増やしてください。

レポート クリーナのスケジュールは、以下のプロパティを変更することによって変更できます。

rpt.cleaner.initDelayMin=10

レジストリを起動した後、このプロセスを開始する前に何分待機するかを決定します。

rpt.cleaner.pulseMin=60

このプロセスの各実行間にどれだけの時間待機するかを決定します。

監査ログ エントリの自動削除

アクセス制御（ACL）機能は、監査ログ (P. 144)を DevTest データベースに格納します。 レジストリは、データベースから古い監査ログ エントリを削除するプロセスを定期的に実行します。

この動作は、以下のプロパティで制御します。

lisa.acl.audit.logs.delete.frequency

自動削除プロセスの実行頻度を指定します。デフォルト値は **1d** です。これは、プロセスが 1 日に 1 回実行されることを意味します。有効な時間単位は、d、h、m、s （日、時間、分、秒）です。

lisa.acl.audit.logs.delete.age

自動削除プロセスによって削除される監査ログ エントリの最小経過期間を指定します。 デフォルト値は 30d です。これは、エントリが 30 日経過後に古いと見なされることを意味します。有効な時間単位は、d、h、m、s （日、時間、分、秒）です。

各プロパティのデフォルト値は **lisa.properties** ファイルにあります。 デフォルト値を変更する場合は、**local.properties** ファイルにプロパティを追加します。

トランザクションの自動削除

DevTest データベース内の古いトランザクションを自動的に削除するように CA Continuous Application Insight を設定できます。

デフォルトでは、トランザクションの自動削除は有効です。

以下のプロパティが使用できます。

クリーナの有効化

トランザクションの自動削除が有効かどうかを制御します。

クリーンアップ頻度

クリーナ プロセスを実行する頻度を指定します。 値の単位は分です。

最長経過期間

クリーナ プロセスによって削除されるトランザクションの経過期間を指定します。 値の単位は分です。

これらのプロパティは、DevTest ポータルの ［エージェント］ ウィンドウから設定できます プロパティは、［設定］ タブに表示されます。

ケースの自動削除

DevTest データベース内の古いケースを自動的に削除するように CA Continuous Application Insight を設定できます。

デフォルトでは、ケースの自動削除は有効です。

以下のプロパティが使用できます。

クリーナの有効化

ケースの自動削除が有効かどうかを制御します。

クリーンアップ頻度

クリーナ プロセスを実行する頻度を指定します。 値の単位は分です。

最長経過期間

クリーナ プロセスによって削除されるケースの経過期間を指定します。 値の単位は分です。

これらのプロパティは、DevTest ポータルの ［エージェント］ ウィンドウから設定できます プロパティは、［設定］ タブに表示されます。

第 3 章: ライセンス管理

このセクションには、以下のトピックが含まれています。

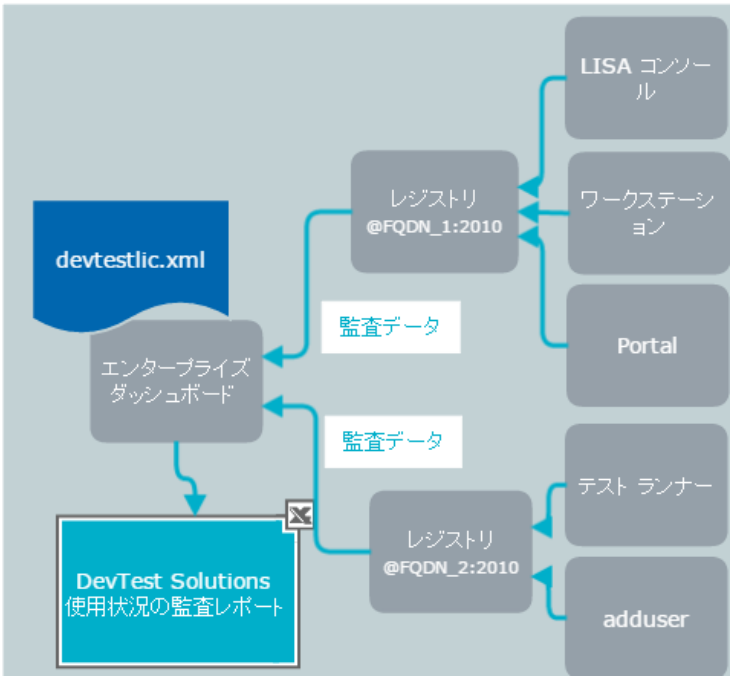
信用ベースのライセンス

信用ベースのライセンスの概要。

- ライセンス契約書は、ユーザ タイプごとに許可される同時ユーザ セッションの最大数をベースにしています。 お客様固有のライセンス契約書に関して質問がある場合は、担当のアカウント チームにお問い合わせください。
- ライセンスはファイルベースで提供され、企業ごとに 1 つのファイルが用意されます。 DevTest Solutions は、インストール後の初期スタートアップ時に、このファイルによりアクティブになります。
- DevTest Solutions 8.0 では、ローカルのライセンス サーバ（LLS）およびインターネットベースのライセンス サーバはサポートされなくなりました。
- ユーザ タイプごとの同時使用状況データは自動的に収集されます。
- エンタープライズ ダッシュボードは、ライセンス レポートを作成します。
- 使用状況監査レポート（ユーザ タイプごとの最大同時使用状況を報告するレポート）は、ライセンス契約書が遵守されているかどうかを評価するのに役立つツールです。 管理者権限を持つユーザは、このレポートにアクセスできます。

詳細については、「ライセンス、ACL、監査レポートの連携の方法 (P. 65)」を参照してください。

以下の図は、エンタープライズ ダッシュボードを使用した、保存されているライセンス ファイルによる DevTest Solutions の有効化を示します。レジストリは、DevTest コンソール、ワークステーション、ポータル、ならびに TestRunner、adduser などのコマンドライン ユーティリティを含む、ユーザがログインする UI および CLI から監査データを収集します。 レジストリは、エンタープライズ ダッシュボードに監査データを転送します。管理者は、DevTest Solutions 使用状況監査レポート を生成して、ライセンス契約の遵守を検証できます。



ライセンス、ACL、監査レポートの連携の方法

トピックは、以下の項目で構成されます。

- Usage-Based ライセンス契約書
- ファイル ベースのライセンス有効化
- ACL の有効化
- 信用ベースの遵守
- ユーザ タイプごとの使用状況データの継続的な収集
- 使用状況監査レポート

Usage-Based ライセンス契約書

お客様の CA Technologies とのライセンス契約は、ユーザ タイプごとの同時ユーザ セッションの最大数をベースにしています。 以下のユーザ タイプがあります。

- ランタイム ユーザ
- テスト パワー ユーザ
- SV パワー ユーザ
- CAI パワー ユーザ

ユーザが DevTest Solutions で実行できるアクティビティにはそれぞれ対応する権限があります。 権限はそれぞれロールに関連付けられます。ロールはそれぞれユーザ タイプの 1 つと関連付けられます。 管理者は、ユーザに権限を割り当てます。

ライセンスの有効化

DevTest Solutions のライセンスは、ファイルベースで有効化します。 CA Technologies は、お客様にライセンス ファイル（devtestlic.xml）を提供します。 DevTest Solutions のインストールまたはアップグレード中に、セットアップ ウィザードが、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上の LISA_HOME ディレクトリにファイルを配置します。 サーバ コンポーネント（DevTest サーバ）の以降のインストールは、エンタープライズ ダッシュボードの URL を参照します。 お客様が最初にエンタープライズ ダッシュボードおよびレジストリを起動するとき、ライセンスが有効化されます。 その他のコンポーネントは、ライセンスの有効化を必要としません。

アップグレードする場合は、新しい devtestlic.xml ライセンスを CA Technologies に請求します。 現在のライセンス キー生成プロセスは devtestlic.xml ファイルを介して配布されるプロダクト キーを生成するために利用されます。 このキーは、エンタープライズ ダッシュボードのアクティブ化またはロック解除に使用されます。

ACL の有効化

アクセス制御（ACL）は、デフォルトで有効です。 DevTest Solutions にアクセスするには、事前に、ユーザが有効な認証情報で認証され、ユーザのロールと関連付けられたタスクを実行する権限を付与される必要があります。 サーバ コンソールの［管理］タブを使用して、各ユーザのユーザ名、パスワード、および一連の権限で構成されるロールを定義します。 ロールはユーザ タイプに結び付けられます。

UI（ユーザ インターフェース）または CLI（コマンド ライン インターフェース）にアクセスするには、有効な認証情報でログインする必要があります。 ユーザがワークステーションを開くか、ポータルにアクセスするか、DevTest コンソールにアクセスすると、ログオン ダイアログ ボックスが表示されます。 DevTest を LDAP 認証情報を使用するように設定している場合、ユーザは DevTest または LDAP で定義された認証情報でログインします。 入力された認証情報が権限のあるユーザの認証情報と一致する場合、UI が開きます。 ユーザに提示される機能は、そのユーザに与える権限に基づきます。

ただし、下位互換性のために、1 つの例外が存在します。 ACL は有効ですが、権限のあるユーザは、有効なログイン ユーザ名およびパスワードを指定せずに、テストを実行し、仮想サービスを開始することができます。

注: 認証されたユーザのみが DevTest Solutions にアクセスできるように、標準ユーザのパスワードを変更する (P. 135)ことをお勧めします。

信用ベースの遵守

お客様は、契約条件によるライセンス遵守の責任を負います。 発行されるライセンスは、ユーザ タイプごとの最大同時使用数を強制的に適用するものではありません。 「使用状況監査レポート」の項で説明するように、要求があれば、DevTest Solutions 使用状況監査レポートが CA Technologies に送信されます。 監査レポートが、あるユーザ タイプで同時使用状況が契約の条件を超えていることを示す場合、購買契約の拡大に対するお客様の関心を判断するために、CA のアカウント マネージャから問い合わせがある可能性があります。

購入していない機能の評価を望むお客様は、お客様担当のアカウント チームに連絡して、概念実証トライアルについて話し合うか、または新しい機能に関して質問する必要があります。 購入されていない製品機能について提起されるサポート ケースは、すべて「範囲外」として処理されます。

ユーザ タイプごとの使用状況データの継続的な収集

ユーザが DevTest Solutions UI または CLI にログインすると、各レジストリがユーザ タイプごとのセッション データをキャプチャし、毎時、00 分にエンタープライズ ダッシュボードに転送します。 レジストリとエンタープライズ ダッシュボード間の接続が失われると、接続が再確立されるまで、使用状況の監査データはレジストリに蓄積されます。 エンタープライズ ダッシュボードは、ユーザ タイプごとの同時セッション数を数えるために、企業全体から使用状況データを収集します。

使用状況監査レポート

管理者は、定期的にエンタープライズ ダッシュボードにログオンし、指定した期間の 使用状況監査レポートを生成します。 特定の UI または CLI ユーザ ログインからのデータがレポートされます。 レポートの計算には、特定のユーザ タイプの最大ユーザ数を含む同時セッションが使用されます。 レポートには、ユーザ タイプごとの、同時使用数の詳細が、複数のレジストリにまたがって記録されます。 レポート ジェネレータは、データを伴うユーザ タイプごとに、ドリルダウン詳細を含むタブを作成します。 監査レポートで詳述されるアクセスを、ライセンス契約書と比較して、遵守のレベルを判定できます。 ライセンス契約書の観点からこのレポートを評価して、遵守の確認や契約のアップグレードを希望するかどうかの確認を行えます。

DevTest Solutions 使用状況監査レポート

DevTest Solutions 使用状況監査レポートは以下のタブを持った Excel ワークブックとして生成されます。

- 最初のタブ： ［概要］
- ユーザ タイプ タブ： ［Admin User］、［PF Power User］、［SV Power User］、［Test User］、［Runtime User］

  期間中に、同時使用または単一使用で、何らかのアクティビティがあったユーザ タイプごとにタブが用意されます。管理者ユーザはレポートに含まれますが、ライセンス契約書の遵守を評価する際に、考慮する必要はありません。
- 最後のタブ： ［Component By User］。この期間中にアクセスがあった各 UI または CLI の列が用意されます。 このタブ上のエントリは、同時アクセスを意味しない場合があります。

概要

［概要］タブは、企業が使用している範囲、つまり、ライセンスが許可する各ユーザ タイプの現在のユーザの数の詳細を提供します。 レポートは、指定した日付範囲のデータを、すべてのレジストリから収集して提供します。 カウント データはユーザ タイプごとに表示されます。

Report Information

［Report Information］セクションでは、ライセンス情報およびこの監査レポートの日付範囲を示します。

- Company： このレポートを生成した企業の名前。
- Key ID： ライセンス キー識別子
- Expiration： ライセンスが期限切れになる日付
- Reporting Period Start： このレポートの開始日。
- Reporting Period End： このレポートの終了日。

User Type and Count

レポートされるユーザ タイプには、PF パワー ユーザ、SV パワー ユーザ、テスト パワー ユーザ、およびランタイム ユーザがあります。 管理者ユーザもレポートされますが、ライセンスのユーザ タイプとしてはカウントされません。

ユーザが複数のユーザ タイプと関連付けられる権限を付与されている場合、上位のユーザ タイプが使用されます。ユーザ タイプは、以下に示す順序で、上位から下位に階層化されています。

a. PF パワー ユーザまたは SV パワー ユーザ（同レベルのユーザ タイプ）

b. テスト パワー ユーザ

c. ランタイム ユーザ

d. 管理者ユーザ

「ユーザ タイプ (P. 74)」を参照してください。

すべてのレジストリが、UI および CLI をサポートするサービスから使用状況データを継続的に収集します。 同時ログイン セッションがレジストリによって記録されている間、企業全体を対象に同じユーザ タイプによる同時使用状況の生データが保管されます。 同時使用状況は、同じユーザ タイプの 2 人以上のユーザが DevTest UI または DevTest CLI にアクセスし、そのセッションが正しいテンポで重複する場合に発生します。

レポートのカウント計算は、最大同時使用数に到達しているグループ カウントに基づきます。 最小、最大、および標準偏差は、複数のレジストリにまたがる分布を伝えます。 「カウントの計算例 (P. 72)」を参照してください。

ユーザ タイプ タブ（[Admin User]、[PF Power User]、[SV Power User]、[Test User]、[Runtime User]）

各ユーザ タイプ タブは、このレポート期間に、最大同時使用数に到達した、同時使用グループのメトリックおよび統計をレポートします。

- 平均：
  - 平均は、同時使用グループの一部を構成するレジストリごとに、計算されます。
  - 合計は、ユーザ タイプを対象に計算されます。この値は、［概要］ページにも表示されます。
- STDEV：レジストリ サンプル セットごとに、計算される標準偏差。
- 最小：各レジストリ サンプル セットの最低値。
- 最大：各レジストリ サンプル セットの最高値。

Component By User

［Component By User］タブは、エンタープライズ ダッシュボードに接続されている各レジストリからのユーザ セッション データをレポートします。

Registry

レジストリ名は、

registry@FQDN:2010 という形式です。

ユーザ名

1 つ以上のセッションでユーザ 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ログインしたユーザの名前。

ユーザ タイプ

関連するユーザに割り当てられるユーザ タイプ。ユーザ タイプは、次のように表示されます。

- ADMIN_USER
- PF_POWER_USER
- SV_POWER_USER
- TEST_POWER_USER
- SVT_RUNTIME_USER

合計セッション数

関連するユーザについてレポートされた列からの合計数。ここでは、列が UI セッションまたは CLI を使用するセッションのいずれかを表します。ユーザ インターフェースには DevTest コンソール、DevTest ワークステーション、ポータルなどがあります。CLI には、AddUser、Test Runner などがあります。列は、ユーザのログイン位置に左右され、レポートごとに異なる可能性があります。以下に、いくつかの例を示します。

- ユーザの追加： このユーザが adduser コマンド ライン ユーティリティを開始した回数。
- コンソール： このレジストリの DevTest コンソールにこのユーザがログインした回数。
- ワークステーション： このユーザが DevTest ワークステーションを開き、関連付けられたレジストリに接続し、ログインした回数。
- ポータル： このユーザがこのレジストリの http://*hostname*:1507/devtest にアクセスしてログインした回数。

注: このレポートの生成の詳細については、「使用状況の監査データのエクスポート (P. 188)」を参照してください。

カウントの計算例

DevTest Solutions 使用状況監査レポート (P. 69)のカウント計算は、3 段階で構成されるプロセスです。

1. 企業全体の中で同時使用が検出されるたびに、関係するすべてのレジストリについて、グループ カウントが記録されます。 テスト パワー ユーザ ユーザ グループの以下の例に注目してください。
2. レポート作成のために、同時使用数が最大のグループ カウントが使用されます。 このサンプルの例では、網掛けされた行（1、3、および 4）が使用されます。

テスト パワー ユーザ				
タイムスタンプ	レジストリ 1	レジストリ 2	レジストリ 3	グループ カウント
1	1	0	3	4

テスト パワー ユーザ				
タイムスタンプ	レジストリ 1	レジストリ 2	レジストリ 3	グループ カウント
2	0	0	2	2
3	1	2	1	4
4	0	0	4	4
5	1	1	1	3

3. レジストリごとの最大グループ カウント データを識別した後、レジストリごとのカウントの平均が計算されます。 最小カウントおよび最大カウント値が格納されます。 これらの 3 つの値（平均、最小、および最大）がレポートに表示されます。

テスト パワー ユーザ						
レジストリ	カウント 1	カウント 2	カウント 3	平均	最小	最大
レジストリ 1	1	1	0	0.67	0	1
レジストリ 2	0	2	0	0.67	0	2
レジストリ 3	3	1	4	2.67	1	4
合計				**4**	1	7

このデータは、［User Type and Count］として、レポートの［概要］ページでまとめられます。 表示されるカウント データは、ユーザ タイプごとに合算されます。 データは、ユーザ タイプごとに、複数のレジストリにまたがって、サンプル内の合計最小値および最大値の平均を求めることで、取得されます。 カウントは、そのユーザ タイプの最大同時使用数を意味します。これは、サンプルを構成するのが、同時使用数が最大のグループ カウントだけだからです。

ユーザ タイプ	カウント
テスト パワー ユーザ	**4**

ユーザ タイプ

ACL には以下のユーザ タイプがあります。

- PF パワー ユーザ
- SV パワー ユーザ
- テスト パワー ユーザ
- ランタイム ユーザ

注: ライセンスでは、管理者ユーザ タイプ（［ロール］ページには、PF パワー ユーザと SV パワー ユーザの結合として表示されます）は使用されません。 ただし、DevTest Solutions 使用状況監査レポートには、参考のために管理者ユーザが含まれます。

ユーザ タイプは 1 つ以上のロールと関連付けることができます。 各ロールは、1 つのユーザ タイプだけに関連付けられます。 たとえば、ランタイム ユーザ ユーザ タイプは 3 つのロール（システム管理者、ランタイム、およびゲスト）を持ちますが、ランタイム ロールは ランタイム ユーザ ユーザ タイプだけに関連付けられます。

ユーザには 1 つ以上のロールを付与できます。複数のロールを付与される場合、使用状況の監査には、最上位のユーザ タイプと関連付けられているロールが使用されます。 ユーザ タイプ階層は、以下のユーザ タイプで構成されます。

- PF パワー ユーザ と SV パワー ユーザ は階層的には等しく、最上位のユーザ タイプです。
- テスト パワー ユーザは PF パワー ユーザ および SV パワー ユーザより低く、ランタイム ユーザより高い地位にあります。
- ランタイム ユーザは最下位のユーザ タイプです。

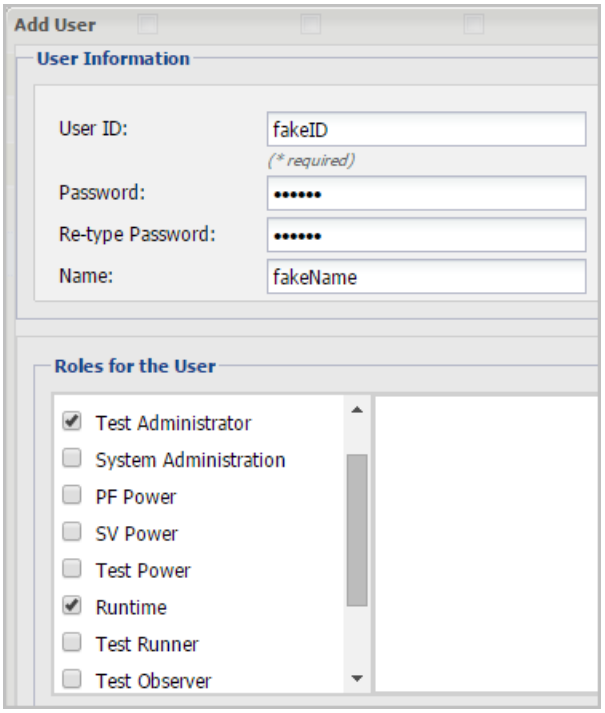
監査目的での、ユーザ タイプの判定方法の例

ユーザに割り当てられるロールに関連付けられる権限は、ユーザが実行を許可されるタスクを決定します。管理者はユーザに 1 つ以上のロールを割り当てることができます。 通常、ビルトイン ロールの権限は、ユーザ タイプ階層を念頭において割り当てられるので、管理者は各ユーザに単一のロールを割り当てます。 たとえば、テスト パワーおよび Runtime に関連付けられる権限は、PF パワー ロールにも割り当てられます。

ユーザに上位のロールの一部でない権限を与えるために、ユーザに複数の権限を割り当てることができます。

次の例を検討してみましょう。ここでは、ユーザが以下のロールを割り当てられています。

- テスト管理者
- ランタイム

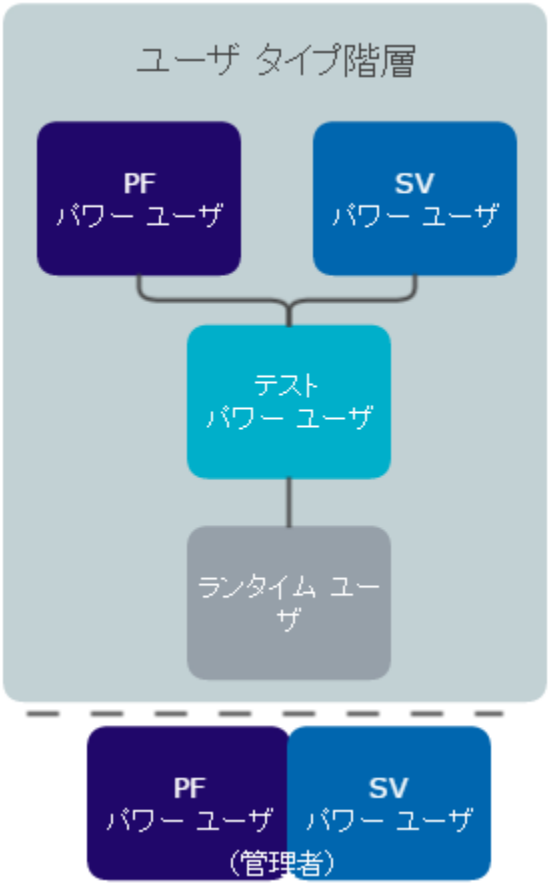


割り当てられているロールには、別々のユーザ タイプが関連付けられます。

- テスト管理者ロールは、SV パワー ユーザ ユーザ タイプに関連付けられます。
- Runtime ロールは、Runtime User ユーザ タイプに関連付けられます。

  注: 詳細については、「標準ユーザ タイプおよび標準ロール (P. 106)」を参照してください。

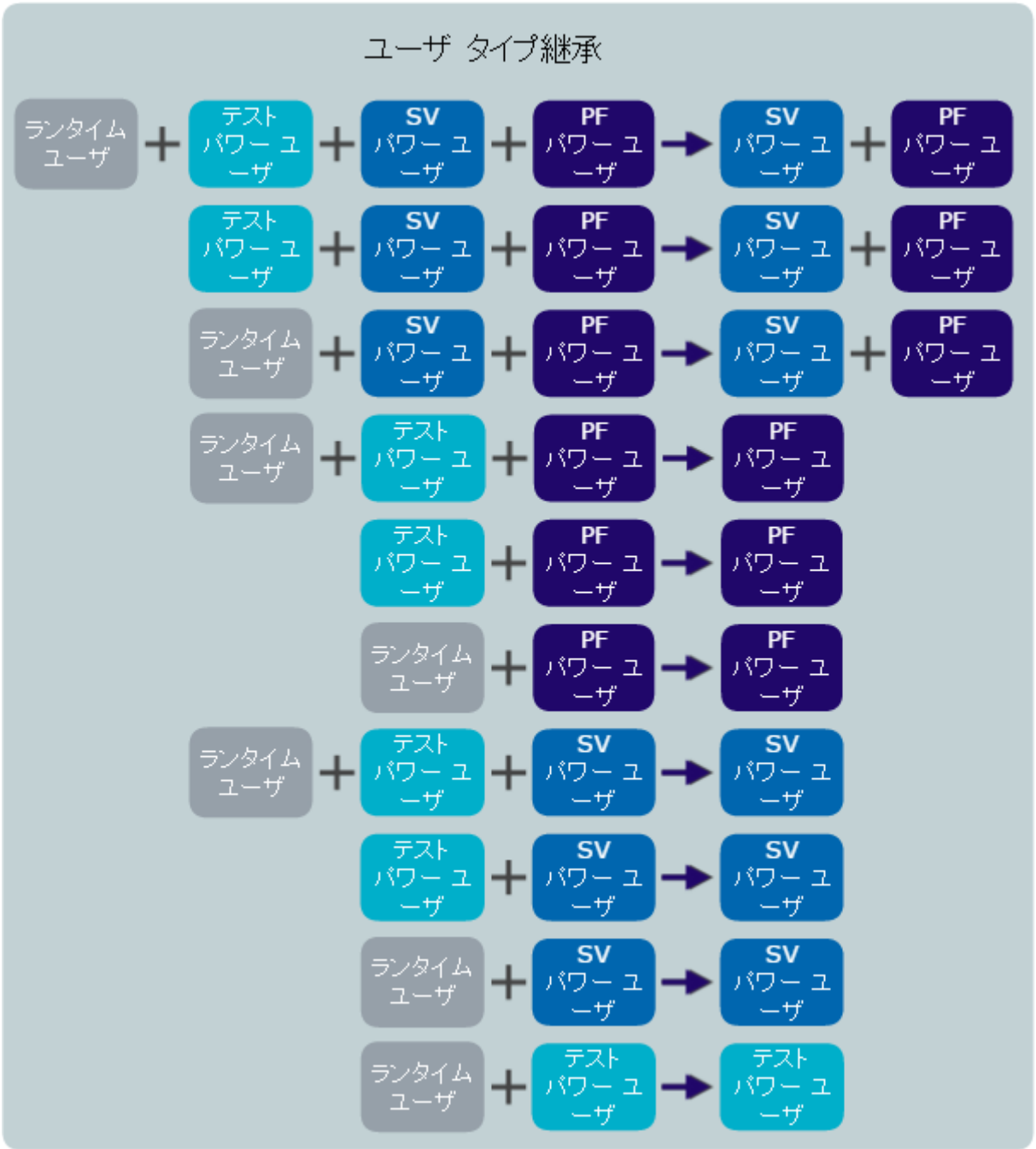
ユーザ タイプは階層化されています。



DevTest Solutions 使用状況監査レポートでは、レポートの目的に合わせて、ユーザのカウントに使用されるユーザ タイプは、割り当てられたロールと関連付けられるユーザ タイプの中で最上位のユーザ タイプです。 この例では、*fakeName* ユーザが DevTest UI または CLI にログインすると、ユーザ セッションは、SV パワー ユーザ ユーザ タイプに属するものとして監査されます。これは、*fakeName* がレポート管理など、Runtime ロールと関連付けられるタスクのみを実行する場合も同じです。

ユーザ タイプ継承チャート

以下の図では、ユーザが異なるユーザ タイプからの複数のロールに割り当てられている場合に、ログイン ユーザのユーザ タイプを判定する方法を、すべてのケースを対象に示します。 「最上位の」ユーザ タイプが、下位のどのユーザ タイプに対しても優先されます。



第 4 章: セキュリティ

このセクションには、以下のトピックが含まれています。

通信を保護するための SSL の使用

デフォルトでは、コンポーネント間の通信は暗号化されていないプロトコルを使用します。 必要に応じて、SSL （Secure Sockets Layer）を使用してネットワーク トラフィックを暗号化できます。 たとえば、パブリック クラウドでラボを実行し、ワークステーションから送信されたトラフィックが暗号化されるようにするとします。

SSL を有効にする最も簡単な方法は、DevTest プロパティを設定することです。

```
lisa.net.default.protocol=ssl
```

このプロパティは、**site.properties** では指定できません。これは、ブートストラップ段階では遅すぎるためです。 このプロパティは、**local.properties** （またはコマンド ライン）で指定する必要があります。

追加のパラメータなしでレジストリを起動する場合（たとえば、レジストリがポート 2010 （通常のポート）でリスンし、クライアントが SSL プロトコルを使用すると想定する場合）、レジストリのサービス名は *ssl://hostname:2010/Registry* です。

DevTest ワークステーション からのそのレジストリに接続する場合、通常の *tcp://hostname:2010/Registry* ではなく、*ssl://hostname:2010/Registry* を使用します。 同じコンピュータ上のシミュレータを起動する場合、シミュレータは *ssl://hostname:2014/Simulator* で使用可能であり、自動的に *ssl://hostname:2010/Registry* でレジストリに接続します（プロパティの変更は不要）。

また、SSL と通常の TCP プロトコルを混在させることもできます。 **lisa.net.default.protocol** プロパティをデフォルト設定（tcp）のままにする場合、デフォルトの「tcp:」プレフィックスではなく、「ssl:」プロトコル プレフィクスを使用して個別のサービスの名前を指定することにより、SSL 用の特定のサービスを有効にできます。 たとえば、SSL モードでレジストリを起動するには、以下のように指定します。

```
Registry --name=ssl://reghost.company.com:2010/Registry
```

SSL を有効にするには、サービス名で「tcp」の代わりに「ssl」を使用します。 以下に例を示します。

```
Registry --name=ssl://reghost.company.com:2010/Registry
```

これは、SSL を有効にした状態でレジストリを起動します。

このレジストリにシミュレータを接続するには、完全修飾レジストリ アドレスを使用してシミュレータを起動します。

```
Simulator --name=ssl://simhost.company.com:2014/Simulator
--registry=ssl://reghost.company.com:2010/Registry
```

このコマンドは、SSL を使用してシミュレータを保護しながらレジストリと通信するようにシミュレータに指示します。 シミュレータ自体を保護する必要がない場合は、以下のようにします。

```
Simulator --registry=ssl://reghost.company.com:2010/Registry
```

保護されたサーバと保護されていないサーバを混在させることは一般的ではありません。 ただし、保護されていないサーバをファイアウォールの内部、保護されたサーバをパブリック クラウドに配置する場合があります。 SSL 暗号化の使用には多少のオーバーヘッドを伴いますが、それはハードウェアによって大きく異なります。

lisa.net.default.protocol プロパティは、ActiveMQ 接続用のデフォルト プロトコルを定義します。 このプロパティは、DevTest コンポーネントの起動時に使用されるプロトコルには影響しません。

SSL 証明書

デフォルトでは、自己署名証明書は、コンポーネント間のメッセージを暗号化および復号化します。 VSE も、https:// スタイルの Web トラフィックを記録するときに、この証明書を使用します。 この証明書は、**LISA_HOME¥webreckeys.ks** ファイル内にあります。

デフォルト プロトコルを SSL に設定し、それ以外は何も変更していない場合、「内部 DevTest」証明書が使用されます。 （ユーザの組織のみではなく）すべての DevTest ユーザがこの内部証明書を共有します。 この証明書を使用することによってネットワーク トラフィックは暗号化されますが、権限のないユーザがパブリック クラウド上のシミュレータに接続することは防止できません。 このタイプの不正アクセスを防ぐには、独自の証明書を使用します。 「既知の」 DevTest 証明書を使用し続けることもできますが、アクセス制御を有効にしてください。

独自の証明書を使用する場合、以下のプロパティを指定することによって証明書キーストアよりも優先できます。

```
lisa.net.keyStore=/path/to/keystore.ks
```

```
lisa.net.keyStore.password=plaintextPassword
```

DevTest では、プレーン テキスト パスワードが初めて読み取られると、そのパスワードは暗号化されたプロパティに変換されます。

```
lisa.net.keyStore.password_enc=33aa310aa4e18c114dacf86a33cee898
```

独自の自己署名証明書の作成

この例では、JRE （Java Runtime Environment）に含まれている keytool ユーティリティを使用します。

独自の自己署名証明書を作成する方法

1. プロンプトに適切な応答を入力します。

```
prompt>keytool -genkey -alias serverA -keyalg RSA -validity 365
-keystore keystore.ks
```

キーストアのパスワードを入力してください: <実際のプレーン テキストは表示されません>

新規パスワードを再入力してください: MyNewSecretPassword

姓名を入力してください。

```
  [Unknown]: serverA
```

組織単位名を入力してください。

```
  [Unknown]: dev
```

組織名を入力してください。

```
  [Unknown]: ITKO
```

都市名または地域名を入力してください。

```
  [Unknown]: Dallas
```

都道府県名を入力してください。

```
  [Unknown]: TX
```

この単位に該当する 2 文字の国番号を入力してください。

```
  [Unknown]: US
```

CN=serverA, OU=dev, O=ITKO, L=Dallas, ST=TX, C=US でよろしいですか?

```
  [no]: yes
```

<serverA> の鍵パスワードを入力してください。

(キーストアのパスワードと同じ場合は RETURN を押してください)

このユーティリティは、365 日間有効な証明書が含まれるファイルを作成します。

2. このファイルを LISA_HOME にコピーして、**local.properties** を更新します。

```
lisa.net.keyStore={{LISA_HOME}}keystore.ks
lisa.net.keyStore.password=MyNewSecretPassword
```

3. DevTest では、プレーン テキスト パスワードが初めて読み取られると、そのパスワードは暗号化されたプロパティに変換されます。

   ```
   lisa.net.keyStore.password_enc=33aa310aa4e18c114dacf86a33cee898
   ```

   サーバ側の接続設定はこれで完了です。

4. クライアントを設定します。

   この証明書は自己署名されているため、証明書を信頼するように明示的にクライアントに指示します。 通常、SSL サービスに接続（たとえば、ブラウザを使用して https://www.MyBank.com に接続）すると、信頼された認証局によって証明書が認証されます。 信頼されたサードパーティは自己署名証明書を認証しないため、証明書をトラスト ストアに追加する必要があります。

   ```
   lisa.net.trustStore={{LISA_HOME}}trustStore.ts
   ```

   ```
   lisa.net.trustStore.password=MyNewSecretPassword
   ```

   同じ keytool ユーティリティがトラスト ストアを操作します。 一般的に、キーストアには 1 つの証明書が含まれ、トラスト ストアには 1 つ以上の証明書が含まれます。

5. サーバ キーストアの証明書をエクスポートします。

   ```
   keytool -exportcert -rfc -alias serverA -keystore keyStore.ks
   -file serverA.cer
   ```

   -rfc は、コピーおよび貼り付けを容易に行えるように、バイナリではなく ASCII テキストとして証明書をエクスポートすることを意味します。 この例では、結果として生成される **serverA.cer** ファイルは以下のようになります。

   ```
   -----BEGIN CERTIFICATE-----
   ```

   ```
   MIICEzCCAXygAwIBAgIEThZnYzANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJ
   DQjELMAkGA1UE
   ```

   ```
   CBM420IxCzAJBgNVBAcTAkNCMQswCQYDVQQKEwJDQjELMAkGA1UECxMCQ0IxCzA
   JBgNVBAMTAkNC
   ```

   ```
   MB4XDTExMDcwODAyMTE0N1oXDTEyMDcwNzAyMTE0N1owTjELMAkGA1UEBhMCQ0I
   xCzAJBgNVBAgT
   ```

   ```
   AkNCMQswCQYDVQQHEwJDQjELMAkGA1UECDMCQ0IxCzAJBgNVBAsTAkNCMQswCQY
   DVQQDEwJDQjCB
   ```

   ```
   nzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAhYfaN+dCrKQwYZ+KeaaPUI8DeXN
   iqQ/mS+KGnXnh
   ```

   ```
   Pz08vdX/7HDLW4pzFhntjmkxxOi9dMwlO2thTD1cOxI571PotenMENo4nyiUAEn
   MK9MTiWEYr2cQ
   ```

```
b6/TUueBCjRJ9I0GPCI0WPS+0Na2Q/wq8gPCHmDRpw1Xgo4uZ1v6C/ECAwEAATA
NBgkqhkiG9w0B

AQUFAAOBgQByCsX9EoBFIGhcSwoRwEvapIrv8wTaqQPOKKyeIevSmbnERRu6+oi
+cJftbdEfw6GG

CBddJH+dGZ9VeqLU8zBGasbU+JPzG5ElOgOXcUGeQQEaM1YMv6XWrIwNSljQk/M
PZSt3ROtJOlae

JPKJXSQ610xof9+yLHH0ebUGhUjdlQ==

-----END CERTIFICATE-----
```

6. この証明書をクライアント トラスト ストアに追加します。

   トラスト ストア ファイルを作成しているため、パスワードを 2 回入力します。 このクライアント トラスト ストアにさらに証明書を追加する場合は、パスワードを 1 回入力します。

```
prompt> keytool -importcert -file serverA.cer -keystore
trustStore.ts

キーストアのパスワードを入力してください:

新規パスワードを再入力してください:

所有者: CN=serverA, OU=dev, O=itko, L=Dallas, ST=Texas, C=US

発行者: CN=serverA, OU=dev, O=itko, L=Dallas, ST=Texas, C=US

シリアル番号: 4e155338

有効期間の開始日: Thu Jul 07 16:33:28 EST 2011 until: Wed Oct 05
17:33:28 EST 2011

Certificate fingerprints:

     MD5:  5B:10:F6:C8:02:3E:36:F5:AA:6D:FC:10:EF:F5:7F:54

     SHA1:
09:DA:8E:71:7C:D5:BB:44:89:14:13:07:F4:A1:C7:06:35:CD:BE:B1

     署名アルゴリズム名: SHA1withRSA

     バージョン: 3

この証明書を信頼しますか? [no]:  yes

証明書がキーストアに追加されました
```

   これで、パブリック クラウドの DevTest サーバへの暗号で強化された通信手段が構築されました。相互に通信するためには、2 つの DevTest コンポーネントの両側に証明書が必要です。

7. クライアントが複数のリモート SSL サーバと通信する場合、同じ keytool コマンドを実行して証明書をトラスト ストアにインポートします。

注: トランスポート レベルのセキュリティ（SSL）に加えて、よりきめ細かなアクセス制御リスト（ACL）を有効にすることもできます。 アクセス制御リストを使用すると、ユーザはユーザ名およびパスワードによって認証する必要があります。 このタイプのセキュリティは、HTTPS を使用していてもユーザ ID の提示を必要とする金融機関の Web サイトに似ています。

複数の証明書による SSL の使用

設定した DevTest のローカル コピーが複数のサーバ証明書を使用して安全に通信するためには、各サーバ証明書をローカルのトラストストア ファイルに追加します。

この例では、serverA、serverB、および workstation を使用します。

serverA の管理者は、keytool を使用して証明書をエクスポートします。

```
serverA> keytool -exportcert -alias lisa -file serverA.cer -keystore
serverA.ks
```

同様に、serverB の管理者は serverB の証明書をエクスポートします。

```
serverB> keytool -exportcert -alias lisa -file serverB.cer -keystore serverB.ks
```

serverA.cer および **serverB.cer** のコピーを取得し、それらをクライアント トラスト ストアにインポートします。

```
workstation>keytool -importcert -alias serverA -file serverA.cer
-keystore trustStore.ts
```

```
workstation>keytool -importcert -alias serverB -file serverB.cer
-keystore trustStore.ts
```

トラスト ストアを変更するために、そのパスワードを入力します。

ワークステーションがこのトラストストア（この時点で serverA と serverB の両方の証明書が含まれます）を使用していることを確認します。

このファイルを **LISA_HOME** にコピーし、以下のように **local.properties** を更新します。

```
lisa.net.trustStore={{LISA_HOME}}trustStore.ts
```

```
lisa.net.trustStore.password_enc=33aa310aa4e18c114dacf86a33cee898
```

DevTest ワークステーション を実行するときに、レジストリを選択できます。

ssl://serverA:2010/Registry

および

ssl://serverB:2010/Registry

ssl://serverC:2010/Registry に接続しようとすると、必要な証明書がないため、DevTest は接続を拒否します。

相互（双方向）認証

サーバとクライアントの両方が相互に認証する必要があるように DevTest を設定することができます。 このタイプの認証では、サーバ側でプロパティを設定する必要があります。

```
lisa.net.clientAuth=true
```

各クライアントがクライアント トラスト ストア内にサーバ証明書を必要とするのに加えて、サーバ コンポーネントはサーバ トラスト ストア内に各クライアントのクライアント証明書を必要とします。

```
serverA>keytool -importcert -alias clientX -file clientX.cer
-keystore trustStore.ts
```

```
serverA>keytool -importcert -alias clientY -file clientY.cer
-keystore trustStore.ts
```

serverA は serverA トラスト ストア内に clientZ の証明書を持たないため、clientZ が serverA に接続しようとすると接続に失敗します。 clientZ が clientZ トラスト ストア内に serverA の証明書を持っていたとしても、この失敗は発生します。

DevTest コンソールとの HTTPS 通信の使用

DevTest コンソールとの HTTPS 通信を有効にするには、以下のタスクを実行します。

1. 新しいキー ペアおよび証明書の生成（P. 91）
2. LISA_HOME への新しいキーストアのコピー（P. 92）
3. lisa.webserver プロパティの更新（P. 93）

新しいキー ペアおよび証明書の生成

キーおよび証明書を生成する最も簡単な方法は、JDK に付属している keytool アプリケーションを使用することです。このアプリケーションは、キーおよび証明書をキーストア内に直接生成します。

詳細については、http://wiki.eclipse.org/Jetty/Howto/Configure_SSL を参照してください。

次の手順に従ってください:

1. コマンド プロンプト ウィンドウを開きます。
2. 以下のコマンドを入力します。
   ```
   cd JAVA_HOME¥bin
   ```
3. 以下のコマンドを入力します。
   ```
   keytool -keystore keystore -alias jetty -genkey -keyalg RSA
   ```
   注: エイリアスとして「jetty」を使用する必要があります。

   このコマンドは、証明書の情報、およびキーストアおよびその中のキーの両方を保護するためのパスワードの入力をユーザに求めます。
4. 以下のプロンプトに対して入力を行います。

   キーストアのパスワードを入力してください。

   パスワードでは大文字と小文字が区別されます。 入力したパスワードのテキストは表示されません。

   新規パスワードを再入力してください:

   パスワードでは大文字と小文字が区別されます。 入力したパスワードのテキストは表示されません。

   姓名を入力してください。
   [Unknown]:

   レジストリ名で使用されているものと同じマシン名を入力します。通常、これはサーバの非修飾ホスト名です。 たとえば、**jetty.eclipse.org** という名前のマシンでは、jetty.eclipse.org と入力します。

   ただし、-m コマンド ライン パラメータで IP アドレスまたは完全修飾ホスト名を使用して、レジストリを起動することは可能です。その場合、Web ブラウザでの証明書エラーを回避するために、SSL 証明書内のホスト名と一致させておく必要があります。

   注: この入力だけが必須です。

組織単位名を入力してください。
[Unknown]:

組織名を入力してください。
[Unknown]:

都市名または地域名を入力してください。
[Unknown]:

都道府県名を入力してください。
[Unknown]:

この単位に該当する 2 文字の国番号を入力してください。
[Unknown]:

入力内容の確認メッセージが表示されます。

5. 確認したら「**yes**」と入力します。

   以下のプロンプトが表示されます。

   <jetty> の鍵パスワードを入力してください。
   (キーストアのパスワードと同じ場合は RETURN を押してください):

6. Enter キーを押します。

   ユーティリティによって、「**keystore**」という名前の新しいファイルがカレント ディレクトリに作成されます。

LISA_HOME への新しいキーストアのコピー

次の手順に従ってください:

1. **LISA_HOME** ディレクトリに新しいキーストア ファイルをコピーします。
2. キーストア ファイルの名前を **webserver.ks** に変更します。

   注: webserver.ks は、lisa.properties ファイルで指定されているデフォルトのファイルです。 別のファイル名を使用する場合は、lisa.properties を開き、**lisa.webserver.ssl.keystore.location** プロパティを変更して、正しいパス名およびファイル名を反映させます。 詳細については、「Web サーバ プロパティの更新 (P. 93)」を参照してください。

Web サーバ プロパティの更新

次の手順に従ってください:

1. LISA_HOME ディレクトリにある local.properties ファイルを開きます。
2. このファイルに以下のプロパティを追加します。

```
# enable https and setup the webserver ssl keystore
 lisa.webserver.https.enabled=true
 lisa.webserver.ssl.keystore.location={{LISA_HOME}}webserver.ks
 lisa.webserver.ssl.keystore.password=yourpassword
 lisa.webserver.ssl.keymanager.password=yourpassword
 lisa.webserver.port=8443
# should lisa workstation use https when launching the portals?
 lisa.portal.use_https=true
 lisa.portal.url.prefix=http://
```

3. 各プロパティを変更して、正しい値を指定します。

   lisa.webserver.https.enabled

   DevTest コンソールとの通信に HTTPS を使用するように、このプロパティを **true** に設定します。

   lisa.webserver.ssl.keystore.location

   このプロパティのデフォルト値は **{{LISA_HOME}}webserver.ks** です。別のファイル名のキーストア ファイル、または別のディレクトリにあるキーストア ファイルを使用する場合は、この値を変更します。

   lisa.webserver.ssl.keystore.password

   キーストア ファイルを生成するときに定義したパスワードを、このプロパティに設定します。

   lisa.webserver.ssl.keymanager.password

   キーストア ファイルを生成するときに定義したキー マネージャのパスワードを、このプロパティに設定します。 別のパスワードを指定していない場合、このパスワードはキーストアのパスワードと同じです。

   lisa.webserver.port

   このプロパティの設定はオプションですが、HTTPS 用のデフォルト ポートは 8443 です。

   lisa.portal.url.prefix

   このプロパティの値を **http://** から **https://** に変更します。

注: システムが初めてこの local.properties 内のパスワードを読み取るときに、システムがパスワードを暗号化されたプロパティに変換します。

4. 変更を保存して、local.properties を閉じます。
5. レジストリを再起動します。

Kerberos 認証の使用

Kerberos のサポートは、DevTest における基本認証および NTLM のサポートに似ています。 DevTest は、その手順の一部でアクセスするアプリケーションまたはリソースが Kerberos 認証で保護されている場合に Kerberos のサポートを使用します。たとえば、HTTP/HTTPS、Web サービス XML は、NTLM および基本認証と同じ手順を使用します。 Kerberos のサポートは、local.properties ファイル内の以下のプロパティを使用します。

lisa.java.security.auth.login.config

ログイン設定ファイルの場所。

lisa.java.security.krb5.conf

事前設定された場所を上書きするために使用される Kerberos 設定ファイルの場所。

lisa.http.kerberos.principal

プリンシパル + パスワード認証の DevTest でのサポートを使用する場合にログインに使用されるプリンシパルの名前。 DevTest ワークステーション は、起動時にこのプリンシパルを暗号化します。

lisa.http.kerberos.pass

プリンシパル + パスワード認証の DevTest でのサポートを使用する場合にログインに使用されるパスワード。 DevTest ワークステーションは、起動時にこのプリンシパルを暗号化します。

lisa.java.security.auth.login.config および **lisa.java.security.krb5.conf** 設定のみを使用して認証できます。これらのファイルおよびその設定は、DevTest が動作するオペレーティング システムよって異なります。プリンシパル + パスワード認証の DevTest サポートを使用しない認証用にこの 2 つのファイルを設定する方法については、適切なドキュメントを参照してください。

プリンシパル + パスワード認証の DevTest サポート

DevTest に認証情報を提供することによってログ記録をサポートするには、DevTest のログイン設定ファイルを使用するようにユーザのログイン設定ファイルを設定する必要があります。 以下の例は、このファイルの内容を示しています。

```
com.sun.security.jgss.initiate {
```

```
  com.itko.lisa.http.LisaKrb5LoginModule required
doNotPrompt=false;

};
```

カスタムの **LisaKrb5LoginModule** は、1 つの変更を含む標準の **com.sun.security.auth.module.Krb5LoginModule** の拡張です。 この拡張は、ユーザに認証情報を要求する代わりに、**lisa.http.kerberos.principal** および **lisa.http.kerberos.pass** に指定された認証情報をサブミットします。

サンプル krb5.conf ファイル

```
[libdefaults]
        default_realm = EXAMPLE.COM
        allow_weak_crypto = true


[realms]
        EXAMPLE.COM = {
                kdc = kdc.fakedomain.com:60088
        }
[domain_realm]
        .example.com = EXAMPLE.COM
        example.com = EXAMPLE.COM
[login]
        krb4_convert = true
```

Active Directory を KDC として持つサンプル krb5.conf ファイル

```
[libdefaults]
        default_realm = FAKEDOMAIN.COM
        allow_weak_crypto = false
        default_tkt_enctypes = arcfour-hmac-md5
        default_tgs_enctypes = arcfour-hmac-md5
```

```
        permitted_enctypes = RC4-HMAC arcfour-hmac-md5
[realms]
        FAKEDOMAIN.COM = {
            kdc = kdc.fakedomain.com
            master_kdc = kdc.fakedomain.com
            admin_server = kdc.fakedomain.com
            default_domain = FAKEDOMAIN.COM
        }
[domain_realm]
        fakedomain.com = FAKEDOMAIN.COM
[login]
        krb4_convert = true
```

サンプル login.config ファイル

```
com.sun.security.jgss.initiate {
  com.itko.lisa.http.LisaKrb5LoginModule required
doNotPrompt=false;
};
```

アクセス制御（ACL）

アクセス制御（ACL）は、ユーザの認証とロールの適用のために DevTest が使用するメカニズムです。 アクセス制御（ACL）は、デフォルトで有効です。

このセクションには、以下のトピックが含まれます。

ACL の概要

アクセス制御リスト（ACL）は広く使用されるセキュリティ メカニズムです。ACL がユーザに付与するアクセス権は、そのユーザのロール ベースのアクティビティを実行するために必要なアプリケーション機能のみを対象にしています。 DevTest Solutions では、ライセンス契約書の遵守をモニタして維持するために、ACL が必要です。 ライセンス契約書は、同時ユーザ セッションの最大数を基準にしています。

この概要では、以下の ACL トピックについて説明します。

- ACL 展開のプランニング
- 認証
- 認証
- ユーザ セッション
- 自動化されたテスト ケース
- ACL データベース

ACL 展開のプランニング

DevTest Solutions 8.0 以降では、アクセス制御リスト（ACL）システムを介したアクセスの制限が要求されます。ACL システムに対する認証を行わずに、DevTest ワークステーション または DevTest ポータルにアクセスすることはできません。

ACL を設定する前に、ユーザがそれぞれ DevTest Solutions、および各タイプのユーザにとって必要な、対応するアクセスをどのように使用するか慎重に検討します。

デフォルトでは、スーパー ユーザおよびシステム管理者のみが、ACL システムへの管理アクセス権を提供するサーバ コンソールへのアクセス権を持ちます。

ACL 管理者

ACL 管理者は、以下のアクティビティを担当します。

- ACL システムでユーザを作成する。

- 各ユーザの責任および必要なアクセス権に合わせて、ユーザにロールを割り当てる。
- DevTest Solutions のさまざまな部分へのアクセスを制限する。リソースグループにコンポーネントを割り当てることにより実現します。
- 定義されたリソース グループにユーザ アクセスを割り当てる。

管理者は、さまざまな ACL ロールおよびそれらのロールに関連付けられた権限について十分に理解し、各ユーザに適切なロールを割り当てる必要があります。

重要: ACL システムの管理に、デフォルトのスーパー ユーザまたはシステム管理者ユーザを使用しないでください。 デフォルトのスーパー ユーザを使用して、デフォルト スーパー ユーザおよびシステム管理者と同じロールを持つ新しいユーザを作成します。ACL システムの管理には新しいユーザを利用し、不正アクセスを防ぐために、デフォルト スーパー ユーザおよびシステム管理者のパスワードを変更してください。

DevTest Solutions と組み合わせたライトウェイト ディレクトリ アクセス プロトコル（LDAP）の使用

また、LDAP による DevTest Solutions のパスワードの管理を選択することもできます。 この方法は、特に LDAP または Active Directory システムがすでに使用可能な場合に有用です。 LDAP を使用する場合、ユーザ パスワードの変更は LDAP 管理者によって行われます。 ACL 管理者は、パスワードを変更できなくなります。

重要: ACL システムの実装、および LDAP サービスとの統合の責任が、お客様の側にあることに変わりはありません。 これらの実装アクティビティについて支援が必要な場合は、CA Services にお問い合わせください。 ACL によるユーザ管理は、ACL または LDAP 管理者が担当します。 CA Support は、ロール テーブルへの閲覧アクセスを使用できないケースを前に進めることはできません。

たとえば、お客様がテストのステージングの際に権限の問題が理由で発生したトラブルを報告する場合、CA Support が関わる際に、お客様の側で、ACL/LDAP 管理者が対応できるようにしておく必要があります。 ACL 管理者は、そのユーザおよびグループにロールを割り当てる際に、これらの要素を考慮に入れる必要があります。

認証

DevTest がユーザを認証する方法を決定できます。 ACL データベースに ユーザを手動で追加 (P. 139)して、認証情報を指定できます。 認証情報とは、DevTest Solutions 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ユーザがログインする際に使用するユーザ ID およびパスワードのことです。 また、LDAP データベースの認証情報でユーザがすでに定義されている場合は、認証に LDAP サーバを使用することができます。この場合、「LDAP 認証を使用するための ACL の設定 (P. 146)」の手順を実行します。

認証

DevTest は、ユーザの業務上の役割に基づいて、各ユーザがアクセスできる DevTest 機能を制限します。 DevTest Solutions では、インストールの際に、12 個を超える標準ロールがセットされます。 標準ユーザとしてログオンすることにより、さまざまなロールを持ったユーザがどのように DevTest を経験するかを知ることができます。標準ユーザ (P. 132)は一意の標準的なロール (P. 106)を割り当てられます。

重要: ACL 管理者は、セキュリティを保証するために、できるだけ早く、デフォルト パスワード（admin、guest）を変更する必要があります。忘れにくいパスワードを設定してください。

ユーザを ACL データベースに手動で追加する際に、各ユーザにロールを割り当てます。 ロールは、1 つの権限セットを付与します。 複数のロールを割り当てることもできますが、責任が重いロールには関連する責任が軽いロールの権限が含まれるため、必要なことはほとんどありません。LDAP を使用する場合、ACL には、ユーザごとの行が自動的に埋め込まれます。 この場合、「authorize users authenticated by LDAP (P. 148)」の説明に従って、ロールのみを割り当てます。

以下のアクティビティは、権限を使用して制御できるアクティビティの例です。

- テスト ケースの作成
- ステージング ドキュメントの作成
- テスト ケースのステージング

ユーザ セッション

認定ユーザが DevTest UI または CLI にログインすると、ユーザ セッションが作成されます。 ユーザ セッションは監査され、使用状況監査レポートの基礎を形成します。 ユーザ タイプ (P. 74)が複数のロールを含むカテゴリである場合、これらのレポートにはユーザ タイプごとの最大同時ユーザ セッションについてのメトリックおよび統計が含まれます。 「ACL and User Sessions」を参照してください。

認証情報なしで実行された CLI および API の ACL と下位互換性

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するには、ユーザは有効なユーザ名およびパスワードでログインする必要があります。 認証情報の要件は、またテストの実行および仮想サービスの開始にも適用されます。 前のリリースで自動化したテスト ケースを認証情報なしで実行する場合、ACL を一時的にオーバーライドして、スケジュールどおりに実行を続けることができます。 「ACL and Command-Line Tools or APIs (P. 103)」を参照してください。

ACL データベース

ACL データはインストール後、デフォルトの内部 Derby データベースに保存されます。 「*Installing*」で説明するように、Derby データベースはエンタープライズ データベースと入れ替える必要があります。 詳細は、「Database Administration (P. 43)」を参照してください。

ACL およびコマンド ライン ツールまたは API

LISA リリース 7.5.2 以前では、ACL は適用されませんでした。 DevTest Solutions は、デフォルトで ACL を適用します。 CA は、指定したユーザ名およびパスワードなしで実行されてきた、既存のコマンド ライン ツールおよび API を更新するための移行時間の必要性を理解しています。自動化されたテスト、仮想サービス、および TestRunner などログイン認証情報を持つプログラムの更新中に、内部で導出された Runtime ユーザ名で動作するようにプログラムに指示するパラメータを設定できます。 この場合、OS にログインするユーザの ID がセッション オブジェクトに格納されます。

ACL を一時的に無効にするには、以下の手順に従います。

1. 使用中のレジストリが含まれる DevTest サーバにログオンします。
2. インストール ディレクトリに移動します。
3. local.properties ファイルを開いて、編集します。
4. lisa.acl.use.runtime パラメータを追加し、このパラメータを true に設定します。以下のように入力してください。
   `lisa.acl.use.runtime=true`
5. local.properties を保存します。

   ご使用のコマンドライン プログラムが、_runtime_<username> として実行されます。ここで、<username> は、user.name システム プロパティから抽出されます。

ACL を例外なく適用するには、以下の手順に従います。

1. 使用中のレジストリが含まれる DevTest サーバにログオンします。
2. インストール ディレクトリに移動します。
3. local.properties ファイルを開いて、編集します。
4. 次のパラメータのコメントを外すか、または false に設定します。
   `lisa.acl.use.runtime=`
5. local.properties を保存します

   すべての UI および CLI が、有効なログイン認証情報で実行されます。

権限のタイプ

権限は、以下のタイプに分類できます。

- ブール値
- 数値制限
- リスト
- カスタム

ブール値

ほとんどの権限はブール値です。 これらの権限は、アクティビティが許可されているか、許可されていないかを示します。

たとえば、「レジストリの停止」権限は、ユーザがレジストリを停止できるかどうかを示します。

数値制限

権限は、数値制限を表すことができます。

たとえば、「テストのステージング」権限は、指定された最大数の仮想ユーザを使用してユーザがテスト ケースをステージングすることを許可します。

数値制限権限の値は、-1、0、または正の整数である必要があります。

値が -1 である場合、権限が許可され、数値制限はありません。

値が 0 である場合、権限が拒否されます。

リスト

権限は、文字列のリストと関連付けることができます。 文字列には、正規表現を使用できます。

カスタム

開発およびテスト環境のクラウド ベースのプロビジョニングでは、カスタム プロパティを使用します。 DevTest は、VLM （Virtual Lab Manager）プロバイダにアクセスするために必要なユーザ名およびパスワードを指定するためのカスタム プロパティを作成します。

詳細については、「*CA Application Test の使用*」の「DCM のプロパティの設定」を参照してください。

標準ユーザ タイプおよび標準ロール

DevTest Solutions は、関連付けられたロールを持つ標準ユーザ タイプを作成します。 ロールはそれぞれ権限 (P. 120)の一意のセットと関連付けられています。DevTest が標準ロールに割り当てている権限は変更 (P. 141)できます。 また、標準ロールを削除 (P. 141)することもできます。

PF パワー ユーザおよび SV パワー ユーザ (P. 107)

PF パワー ユーザ/SV パワー ユーザ結合ユーザ タイプは、以下のロールを含みます。

- スーパー ユーザ
- DevTest 管理者

PF パワー ユーザ (P. 109)

PF パワー ユーザ ユーザ タイプは、以下のロールを含みます。

- PF パワー

SV パワー ユーザ (P. 111)

SV パワー ユーザ ユーザ タイプは、以下のロールを含みます。

- テスト管理者
- SV パワー
- テスト ランナー

テスト パワー ユーザ (P. 114)

テスト パワー ユーザ ユーザ タイプは、以下のロールを含みます。

- テスト パワー
- テスト オブザーバ
- 負荷テスター
- ユーザ

ランタイム ユーザ (P. 118)

ランタイム ユーザ ユーザ タイプは、以下のロールを含みます。

- システム管理者
- Runtime
- ゲスト

結合 PF パワー ユーザ/SV パワー ユーザ ユーザ タイプ

PF パワー ユーザ/SV パワー ユーザの結合は、以下のロールを含みます。

- スーパー ユーザ
- DevTest 管理者

スーパー ユーザおよびシステム管理者は、ロールに権限を再割り当てできます。 「ロールの追加と更新 (P. 141)」を参照してください。

スーパー ユーザ

スーパー ユーザ ロールには、すべての標準権限が含まれます。

- LISA コンソール管理 (P. 121)
- ユーザおよびロール管理 (P. 121)
- リソース管理 (P. 121)
- テスト/スイート管理 (P. 121)
- 仮想サービス管理 (P. 122)
- DevTest サーバ 管理 (P. 124)
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- メトリックおよびイベント管理 (P. 127)
- Pathfinder 管理 (P. 127)
- DevTest ワークステーション (P. 128)
- クラウド ラボ統合権限 (P. 130)
- ターゲットにアクセス可能

DevTest 管理者

DevTest 管理者ロールには、LISA コンソール管理、ユーザおよびロール管理、およびリソース管理を除く、完全な標準権限があります。 以下の権限があります。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理 (P. 121)
- 仮想サービス管理 (P. 122)
- DevTest サーバ管理 (P. 124)
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- メトリックおよびイベント管理 (P. 127)
- Pathfinder 管理 (P. 127)
- DevTest ワークステーション (P. 128)
- クラウドラボ統合権限 (P. 130)
- ターゲットにアクセス可能

PF パワー ユーザ ユーザ タイプ

PF パワー ユーザ ユーザ タイプは PF パワー ロールで構成されます。

スーパー ユーザおよびシステム管理者は、ロールに権限を再割り当てできます。 「ロールの追加と更新 (P. 141)」を参照してください。

PF パワー

PF パワー ロールには、以下の権限が含まれます。

- DevTest コンソール管理 (P. 121)の一部
  - Pathfinder コンソールへのアクセス（PFP）
  - VSEasy コンソールの表示
- テスト/スイート管理 (P. 121)
- 仮想サービス管理 (P. 122)の一部
  - VSE ダッシュボードの表示
  - VSE サービスの展開
  - VSE サービス アーカイブの取得
  - VSE サービスの開始
  - VSE サービスの停止
  - 仮想サービス容量の更新
  - 仮想サービス反応時間スケールの更新
  - 仮想サービス自動再起動の更新
  - 仮想サービス実行モードの設定
  - 仮想サービス トランザクション数のリセット
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- メトリックおよびイベント管理 (P. 127)
- CAI 管理 (P. 127)
- DevTest ワークステーション (P. 128) の一部
  - DevTest ワークステーション を開始
  - 設定を作成、編集、および表示

- ステージング ドキュメントを作成、編集、および表示
- スイートを作成、編集、および表示
- テスト ケースを作成、編集、および表示
- 監査ドキュメントを作成、編集、および表示
- 仮想サービス モデルの表示
- 仮想サービス イメージの表示
- ITR を実行
- すぐに再生を実行
- ITR プロパティの表示
- ITR テスト イベントの表示
- 利用可能なラボのリスト表示
- ラボを開始および停止
- ラボを拡張
- 他のユーザのラボを強制終了

- ターゲットにアクセス可能

SV パワー ユーザ ユーザ タイプ

SV パワー ユーザ ユーザ タイプには、以下のロールがあります。

- テスト管理者
- SV パワー
- テスト ランナー

スーパー ユーザおよびシステム管理者は、ロールに権限を再割り当てできます。 「ロールの追加と更新 (P. 141)」を参照してください。

テスト管理者

テスト管理者ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理権限 (P. 121)
- 仮想サービス管理権限 (P. 122)
- DevTest ワークステーション 権限 (P. 128)
- クラウド ラボ統合権限 (P. 130)
- ターゲットにアクセス可能

SV パワー

SV パワー ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理権限 (P. 121)
- 仮想サービス管理 (P. 122)の一部
  - VSE ダッシュボードの表示
  - VSE サービスの展開
  - VSE サービス アーカイブの取得
  - VSE サービスの開始
  - VSE サービスの停止
  - 仮想サービス容量の更新
  - 仮想サービス反応時間スケールの更新
  - 仮想サービス自動再起動の更新
  - 仮想サービス グループ タグの更新
  - 仮想サービス実行モードの設定
  - 仮想サービス トランザクション数のリセット
  - 仮想サービス イメージの修復
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- メトリックおよびイベント管理 (P. 127)
- LISA ワークステーション (P. 128)の一部
  - DevTest ワークステーション を開始
  - 設定を作成、編集、および表示
  - ステージング ドキュメントを作成、編集、および表示
  - スイートを作成、編集、および表示
  - テスト ケースを作成、編集、および表示
  - 監査ドキュメントを作成、編集、および表示
  - 仮想サービス モデルを作成、編集、および表示

- 仮想サービス イメージを作成、編集、および表示
- ITR を実行
- すぐに再生を実行
- ITR プロパティの表示
- ITR テスト イベントの表示
- 利用可能なラボのリスト表示
- ラボを開始および停止
- ラボを拡張
- 他のユーザのラボを強制終了

- ターゲットにアクセス可能

テスト ランナー

テスト ランナー ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理権限 (P. 121)
- DevTest ワークステーション 権限 (P. 128)
- クラウド ラボ統合権限 (P. 130)
- ターゲットにアクセス可能

テスト パワー ユーザ ユーザ タイプ

テスト パワー ユーザ ユーザ タイプは、以下のロールを含みます。

- テスト パワー
- テスト オブザーバ
- 負荷テスター
- ユーザ

スーパー ユーザおよびシステム管理者は、ロールに権限を再割り当てできます。 「ロールの追加と更新 (P. 141)」を参照してください。

テスト パワー

テスト パワー ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理 (P. 121)
- 仮想サービス管理 (P. 122)の一部
  - VSE ダッシュボードの表示
  - VSE サービスの展開
  - VSE サービス アーカイブの取得
  - VSE サービスの開始
  - VSE サービスの停止
  - 仮想サービス容量の更新
  - 仮想サービス反応時間スケールの更新
  - 仮想サービス自動再起動の更新
  - 仮想サービス グループ タグの更新
  - 仮想サービス実行モードの設定
  - 仮想サービス トランザクション数のリセット
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- メトリック/イベント管理 (P. 127)

- 以下を除く、DevTest ワークステーション 権限 (P. 128)からのすべて：
  - 仮想サービス モデルを作成または編集
  - 仮想サービス イメージを作成または編集
- クラウド ラボ統合権限 (P. 130)
- ターゲットにアクセス可能
- 指定されたターゲットのリストにのみアクセス可能。

テスト オブザーバ

テスト オブザーバ ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- Test/Suite 管理権限 (P. 121)
- LISA ワークステーション (P. 128)の一部
  - DevTest ワークステーション を開始
  - 設定の表示
  - ステージング ドキュメントの表示
  - スイートの表示
  - テスト ケースの表示
  - 監査ドキュメントの表示
  - 仮想サービス モデルの表示
  - 仮想サービス イメージの表示
  - ITR プロパティの表示
  - ITR テスト イベントの表示
- クラウド ラボ統合権限 (P. 130)
- 指定されたターゲットのリストにのみアクセス可能

負荷テスター

負荷テスター ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- Test/Suite 管理権限 (P. 121)
- CVS 管理権限 (P. 125)
- LISA ワークステーション (P. 128)の一部
  - DevTest ワークステーション を開始
  - 設定の表示
  - ステージング ドキュメントの表示
  - スイートの表示
  - テスト ケースの表示
  - 監査ドキュメントの表示
  - 仮想サービス モデルの表示
  - 仮想サービス イメージの表示
  - ITR プロパティの表示
  - ITR テスト イベントの表示
- クラウド ラボ統合権限 (P. 130)
- 指定されたターゲットのリストにのみアクセス可能

ユーザ

ユーザ ロールには、以下の権限が含まれます。

- LISA コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- Test/Suite 管理権限 (P. 121)
- LISA ワークステーション (P. 128)の一部
  - DevTest ワークステーション を開始
  - 設定の表示
  - ステージング ドキュメントの表示
  - スイートの表示
  - テスト ケースの表示
  - 監査ドキュメントの表示
  - 仮想サービス モデルの表示
  - 仮想サービス イメージの表示
  - ITR プロパティの表示
  - ITR テスト イベントの表示
  - 利用可能なラボのリスト表示
  - ラボの開始/停止
  - ラボを拡張
- ターゲットにアクセス可能

ランタイム ユーザ ユーザ タイプ

ランタイム ユーザ ユーザ タイプは、以下のロールを含みます。

- システム管理者
- Runtime
- ゲスト

スーパー ユーザおよびシステム管理者は、ロールに権限を再割り当てできます。 「ロールの追加と更新 (P. 141)」を参照してください。

システム管理者

システム管理者 ロールには、以下のデフォルト権限が含まれます。

- DevTest コンソール管理 (P. 121)の一部
  - DevTest コンソール アクセス
  - サーバ コンソール アクセス
- ユーザおよびロール管理 (P. 121)
- リソース管理 (P. 121)
- 仮想サービス管理 (P. 122)の一部
  - VSE トラッキング データ クリーンアップの設定
  - VSE サーバの停止
  - VSE サーバのリセット
  - VSE サーバのモニタ
- DevTest サーバ 管理 (P. 124)

Runtime

Runtime ロールには、以下のデフォルト権限が含まれます。

- DevTest コンソール管理 (P. 121)の一部
  - VSEasy コンソールの表示
- テスト/スイート管理 (P. 121)
- 仮想サービス管理 (P. 122)の一部
  - VSE ダッシュボードの表示
  - VSE サービスの展開（R）
  - VSE サービス アーカイブの取得（R）
  - VSE サービスの開始（R）
  - VSE サービスの停止（R）
  - 仮想サービス容量の更新
  - 仮想サービス反応時間スケールの更新
  - 仮想サービス自動再起動の更新
  - 仮想サービス グループ タグの更新
  - 仮想サービス実行モードの設定（R）
  - 仮想サービス トランザクション数のリセット（R）
- CVS 管理 (P. 125)
- レポート管理 (P. 125)
- DevTest ワークステーション (P. 128) の一部
  - DevTest ワークステーションを開始
  - 設定の表示
  - ステージング ドキュメントの表示
  - スイートの表示
  - テスト ケースの表示
  - 監査ドキュメントの表示
  - 仮想サービス モデルの表示
  - 仮想サービス イメージの表示
  - ITR プロパティの表示

- ITR テスト イベントの表示
- クラウド ラボ統合 (P. 130)の一部
  - 利用可能なラボのリスト表示
  - ラボを拡張
- 指定されたターゲットのリストにのみアクセス可能

ゲスト

ゲスト ロールには、以下のデフォルト権限が含まれます。

- DevTest ワークステーション (P. 128) の一部
  - DevTest ワークステーション を開始
  - スイートの表示
  - テスト ケースの表示

標準権限

権限は、ユーザが特定のアクティビティを実行できるかどうかを制御します。

親権限が許可されると、子権限はすべて許可されます。

以下のトップ レベル権限は自動的に作成されます。

権限

LISA コンソール管理権限 (P. 121)
ユーザおよびロール管理権限 (P. 121)
リソース管理権限 (P. 121)
テスト/スイート管理権限 (P. 121)
仮想サービス管理権限 (P. 122)
DevTest サーバ 管理権限 (P. 124)
CVS 管理権限 (P. 125)
レポート管理権限 (P. 125)
メトリック/イベント管理 (P. 127)
CA Continuous Application Insight 管理 (P. 127)
DevTest ワークステーション 権限 (P. 128)
クラウド ラボ統合権限 (P. 130)
DevTest サーバ のデバッグ (P. 131)

LISA コンソール管理権限

LISA コンソール管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
LISA コンソール アクセス	ブール値	ユーザが DevTest コンソールにログインすることを許可します。
サーバ コンソール アクセス	ブール値	ユーザがサーバ コンソールにログインすることを許可します。
Pathfinder コンソール アクセス	ブール値	ユーザが Pathfinder コンソールにログインすることを許可します。
VSEasy コンソールの表示	ブール値	ユーザが VSEasy コンソールを表示することを許可します。

ユーザおよびロール管理権限

ユーザおよびロール管理権限は、ユーザがサーバ コンソールからアクセス制御（ACL）を管理することを許可するブール値権限です。

リソース管理権限

リソース管理権限は、ユーザがリソース グループ (P. 150)を管理することを許可するブール値権限です。

テスト/スイート管理権限

テスト/スイート管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
テストのステージング	数値制限	指定された最大数の仮想ユーザを使用してユーザがテスト ケースをステージングすることを許可します。
スイートのステージング	数値制限	指定された最大数の仮想ユーザを使用してユーザがスイートをステージングすることを許可します。
テストを停止	ブール値	ユーザがレジストリ モニタでテストを停止することを許可します。

テストの強制終了	ブール値	ユーザがレジストリ モニタでテストを強制終了することを許可します。
テストの最適化	ブール値	ユーザがレジストリ モニタでテストを最適化することを許可します。
テストの表示	ブール値	ユーザがレジストリ モニタでテストを表示することを許可します。
テストのクイック ステージング	ブール値	ユーザがクイック テストをステージングすることを許可します。
ローカル スイートのステージング	ブール値	ユーザがスイートをローカルに実行することを許可します。

仮想サービス管理権限

仮想サービス管理権限には、以下の子権限が含まれます。

子権限レベル 1	子権限レベル 2	子権限レベル 3	タイプ	説明
VSE ダッシュボードの表示			ブール値	ユーザがサーバ コンソールで VSE ダッシュボードを表示することを許可します。
VSE サーバ管理			ブール値	
	VSE トラッキング データ クリーンアップの設定		ブール値	指定された時間よりも古いトラッキング データを削除するプロセスをユーザが設定することを許可します。
	VSE サーバの停止		ブール値	ユーザが仮想サービス環境をシャットダウンすることを許可します。
	VSE サーバのリセット		ブール値	ユーザが仮想サービス環境をリセットすることを許可します。
	VSE サーバのモニタ		ブール値	ユーザが仮想サービス環境をモニタすることを許可します。

VSE サービス管理			ブール値	
	VSE サービスの展開		ブール値	ユーザが仮想サービスを展開することを許可します。
	VSE サービスアーカイブの取得		ブール値	仮想サービスと関連付けられたモデル アーカイブ（MAR）をユーザがダウンロードすることを許可します。
	VSE サービスの実行		ブール値	
		VSE サービスの開始	ブール値	ユーザが仮想サービスを開始することを許可します。
		VSE サービスの停止	ブール値	ユーザが仮想サービスを停止することを許可します。
	展開された VSE サービスの更新		ブール値	
		仮想サービス容量の更新	ブール値	展開された仮想サービスの同時実行数をユーザが更新することを許可します。
		仮想サービス反応時間スケールの更新	ブール値	展開された仮想サービスの反応時間スケールをユーザが更新することを許可します。
		仮想サービス自動再起動の更新	ブール値	展開された仮想サービスの自動再起動オプションをユーザが更新することを許可します。
		仮想サービス グループ タグの更新	ブール値	展開された仮想サービスのグループ タグをユーザが更新することを許可します。
	仮想サービス実行モードの設定		ブール値	展開された仮想サービスの新しい実行モードをユーザが選択することを許可します。

	仮想サービス トランザクション数のリセット		ブール値	展開された仮想サービスのトランザクション数およびエラー数をユーザがリセットすることを許可します。
	仮想サービス イメージの修復		ブール値	ユーザがモデルの修正を実行することを許可します。

DevTest サーバ 管理権限

DevTest サーバ 管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
DevTest サーバ のデバッグ	ブール値	ユーザがサーバ コンポーネントのヒープ ダンプの作成、スレッド ダンプの作成、およびガベージ コレクションを実行することを許可します。
レジストリのモニタ	ブール値	ユーザがレジストリ モニタにアクセスすることを許可します。
レジストリのリセット	ブール値	ユーザがレジストリをリセットすることを許可します。
レジストリの停止	ブール値	ユーザがレジストリを停止することを許可します。
コーディネータのモニタ	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでコーディネータ ステータス メッセージを表示することを許可します。
コーディネータのリセット	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでコーディネータをリセットすることを許可します。
コーディネータの停止	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでコーディネータを停止することを許可します。
シミュレータのモニタ	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでシミュレータ ステータス メッセージを表示することを許可します。
シミュレータのリセット	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでシミュレータをリセットすることを許可します。

シミュレータの停止	ブール値	ユーザがサーバ コンソールまたはレジストリ モニタでシミュレータを停止することを許可します。

CVS 管理権限

CVS 管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
CVS ダッシュボードの表示	ブール値	ユーザが CVS ダッシュボードにアクセスすることを許可します。
モニタを CVS に展開/再展開	ブール値	ユーザが CVS モニタを展開/再展開することを許可します。
モニタを CVS から削除	ブール値	ユーザが CVS モニタを削除することを許可します。
モニタを CVS 上ですぐに実行	ブール値	いつ実行されるようにスケジュールされているかにかかわらず、ユーザが CVS モニタをすぐに実行することを許可します。
モニタを CVS 上でアクティブ化/非アクティブ化	ブール値	ユーザが CVS モニタをアクティブ化/非アクティブ化することを許可します。

レポート管理権限

レポート管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
レポート コンソールにアクセス	ブール値	ユーザがレポート コンソールにアクセスすることを許可します。
レポート データの表示	ブール値	ユーザがレポート コンソールで自分のレポート データを表示することを許可します。
他のユーザのレポート データの表示	ブール値	ユーザがレポート コンソールでほかのユーザのレポート データを表示することを許可します。
PDF レポート データの表示	ブール値	ユーザがレポート コンソールで自分のレポート データを PDF ファイルとして表示することを許可します。

他のユーザの PDF レポート データの表示	ブール値	ユーザがレポート コンソールでほかのユーザのレポート データを PDF ファイルとして表示することを許可します。
XML レポート データのインポート	ブール値	ユーザがレポート コンソールに XML データをインポートすることを許可します。
XML レポート データのエクスポート	ブール値	ユーザがレポート コンソールから自分のレポート データを XML ファイルとしてエクスポートすることを許可します。
他のユーザの XML レポート データのエクスポート	ブール値	ユーザがレポート コンソールからほかのユーザのレポート データを XML ファイルとしてエクスポートすることを許可します。
Excel レポート データのエクスポート	ブール値	ユーザがレポート コンソールから自分のレポート データを Excel ファイルとしてエクスポートすることを許可します。
他のユーザの Excel レポート データのエクスポート	ブール値	ユーザがレポート コンソールからほかのユーザのレポート データを Excel ファイルとしてエクスポートすることを許可します。
レポート データの削除	ブール値	ユーザがレポート コンソールから自分のレポート データを削除することを許可します。
他のユーザのレポート データの削除	ブール値	ユーザがレポート コンソールからほかのユーザのレポート データを削除することを許可します。
レポート フィルタの作成	ブール値	ユーザがレポート コンソールでフィルタを作成することを許可します。
レポート フィルタの削除	ブール値	ユーザがレポート コンソールから自分のフィルタを削除することを許可します。
他のユーザのレポート フィルタの削除	ブール値	ユーザがレポート コンソールからほかのユーザのフィルタを削除することを許可します。
すべてのフィルタを表示	ブール値	ユーザがレポート コンソールですべてのユーザのフィルタを表示することを許可します。

メトリック/イベント管理

メトリック/イベント管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
メトリックをランタイムに追加	ブール値	ユーザが実行時にメトリックを追加することを許可します。
メトリックをランタイムに削除	ブール値	ユーザが実行時にメトリックを削除することを許可します。
メトリック収集の一時停止	ブール値	ユーザがメトリック収集を一時停止することを許可します。
間隔データの保存	ブール値	ユーザが間隔データを保存することを許可します。
メトリック データの保存	ブール値	ユーザがメトリック データを保存することを許可します。
イベント データの保存	ブール値	ユーザがイベント データを保存することを許可します。

CA Continuous Application Insight 管理

CA Continuous Application Insight （CAI）管理権限には、以下の子権限が含まれます。

子権限	タイプ	説明
パスの表示	ブール値	ユーザが DevTest ポータルでパスを表示することを許可します。
ベースラインの作成	ブール値	ユーザが DevTest ポータルでベースライン テスト ケースおよびスイートを作成することを許可します。
仮想サービス モデルの作成	ブール値	ユーザが DevTest ポータルで仮想サービス モデルおよび RAW トラフィック ファイルを作成することを許可します。
抽出データ	ブール値	ユーザが DevTest ポータルで XML 要求または応答からテスト データを抽出することを許可します。
チケットの表示	ブール値	ユーザが DevTest ポータルでチケットを表示することを許可します。

チケットの編集	ブール値	ユーザが DevTest ポータルでチケット情報を編集することを許可します。
エージェントの表示	ブール値	ユーザが DevTest ポータルでエージェント情報を表示することを許可します。
エージェント管理	ブール値	ユーザが DevTest ポータルでエージェントに対するディスパッチを開始/停止することを許可します。
パスのインポート	ブール値	ユーザが DevTest ポータルに 1 つ以上のパスをインポートすることを許可します。
パスのエクスポート	ブール値	ユーザが DevTest ポータルから 1 つ以上のパスをエクスポートすることを許可します。
ドキュメントの作成	ブール値	ユーザが DevTest ポータルでトランザクションからドキュメントを作成することを許可します。
マニュアル ケースの表示	ブール値	ユーザが DevTest ポータルでマニュアル テスト ケースを表示することを許可します。
マニュアル ケースの作成	ブール値	ユーザが DevTest ポータルでマニュアル テスト ケースを作成することを許可します。
Edit Point of Interest（注目ポイントの編集）	ブール値	ユーザが DevTest ポータルで注目ポイントのトランザクションを編集することを許可します。
View Point of Interest（注目ポイントの表示）	ブール値	ユーザが DevTest ポータルで注目ポイントのトランザクションを表示することを許可します。
障害の表示	ブール値	ユーザが DevTest ポータルで障害のあるトランザクションを表示することを許可します。

DevTest ワークステーション 権限

DevTest ワークステーション 権限には、以下の子権限が含まれます。

子権限	タイプ	説明
DevTest ワークステーション を開始	ブール値	ユーザが DevTest ワークステーション を開始することを許可します。
設定の表示	ブール値	ユーザが設定を表示することを許可します。

設定の編集	ブール値	ユーザが設定を編集することを許可します。
新規設定の作成	ブール値	ユーザが設定を作成することを許可します。
ステージング ドキュメントの表示	ブール値	ユーザがステージング ドキュメントを表示することを許可します。
ステージング ドキュメントの編集	ブール値	ユーザがステージング ドキュメントを編集することを許可します。
新規ステージング ドキュメントの作成	ブール値	ユーザがステージング ドキュメントを作成することを許可します。
スイートの表示	ブール値	ユーザがスイートを表示することを許可します。
スイートの編集	ブール値	ユーザがスイートを編集することを許可します。
新規スイートの作成	ブール値	ユーザがスイートを作成することを許可します。
テスト ケースの表示	ブール値	ユーザがテスト ケースを表示することを許可します。
テスト ケースの編集	ブール値	ユーザがテスト ケースを編集することを許可します。
新規テスト ケースの作成	ブール値	ユーザがテスト ケースを作成することを許可します。
監査ドキュメントの表示	ブール値	ユーザが監査ドキュメントを表示することを許可します。
監査ドキュメントの編集	ブール値	ユーザが監査ドキュメントを編集することを許可します。
新規監査ドキュメントの作成	ブール値	ユーザが監査ドキュメントを作成することを許可します。
仮想サービス モデルの表示	ブール値	ユーザが仮想サービス モデルを表示することを許可します。
仮想サービス モデルの編集	ブール値	ユーザが仮想サービス モデルを編集することを許可します。

新規仮想サービス モデルの作成	ブール値	ユーザが仮想サービス モデルを作成することを許可します。
仮想サービス イメージの表示	ブール値	ユーザがサービス イメージを表示することを許可します。
仮想サービス イメージの編集	ブール値	ユーザがサービス イメージを編集することを許可します。
新規仮想サービス イメージの作成	ブール値	ユーザがサービス イメージを作成することを許可します。
ITR を実行	ブール値	ユーザが対話型テスト ラン（ITR）ユーティリティを開始することを許可します。
すぐに再生を実行	ブール値	ユーザが対話型テスト ラン（ITR）ユーティリティで特定の時点にテスト ケースを再生することを許可します。
ITR プロパティの表示	ブール値	ユーザが対話型テスト ラン（ITR）ユーティリティで［プロパティ］タブを表示することを許可します。
ITR テスト イベントの表示	ブール値	ユーザが対話型テスト ラン（ITR）ユーティリティで［テスト イベント］タブを表示することを許可します。

クラウド ラボ統合権限

クラウド ラボ統合権限には、以下の子権限が含まれます。

子権限	タイプ	説明
利用可能なラボのリスト表示	ブール値	ユーザが利用可能なラボのリストを表示することを許可します。
ラボの開始/停止	ブール値	ユーザがラボを開始/停止することを許可します。
ラボを拡張	ブール値	ユーザがテスト ラボを動的に拡張することを許可します。
他のユーザのラボを強制終了	ブール値	ほかのユーザが開始したラボをユーザが強制終了することを許可します。

DevTest サーバ のデバッグ

この権限は、ヒープ ダンプおよびスレッド ダンプの作成、およびガベージ コレクションの強制実行をユーザに許可します。この権限は、サーバ コンソールの右クリック、および ServiceManager (P. 163) コマンド ライン ユーティリティを介するコマンドの発行に適用されます。

標準ユーザ

レジストリを初めて起動すると、標準ユーザが作成されます。 標準ユーザはそれぞれ、ユーザ タイプ内のロールと、デフォルトのパスワードを割り当てられます。 各ロールと関連付けられる権限については、「標準ユーザ タイプおよび標準ロール (P. 106)」を参照してください。

重要: 不正アクセスを防ぐために、これらのユーザのデフォルト パスワードをできるだけ早く変更 (P. 135)することをお勧めします。

管理者

管理者ユーザは、PF パワー ユーザ/SV パワー ユーザ結合ユーザ タイプにあたる、スーパー ユーザ ロールを持ちます。デフォルトのパスワードは **admin** です。

pfpower

pfpower ユーザは PF パワー ロール（PF パワー ユーザ ユーザ タイプ）を持ちます。 デフォルトのパスワードは **pfpower** です。

svpower

svpower ユーザは SV パワー ロール（SV パワー ユーザ ユーザ タイプ）を持ちます。 デフォルトのパスワードは **svpower** です。

tpower

tpower ユーザはテスト パワー ロール（テスト パワー ユーザ ユーザ タイプ）を持ちます。 デフォルトのパスワードは **tpower** です。

devtest

devtest ユーザは ランタイム ロール（ランタイム ユーザ ユーザ タイプ）を持ちます。 デフォルトのパスワードは **devtest** です。

sysadmin

sysadmin ユーザはシステム管理者ロール（Runtime ユーザ タイプ）を持ちます。 デフォルトのパスワードは **sysadmin** です。

ゲスト

ゲスト ユーザは、ゲスト ロールを持ちます。 デフォルトのパスワードは **guest** です。

標準ユーザのロールの体験

DevTest Solutions は、それぞれが一意の認証情報および異なるロールを持つ、標準ユーザでインストールされます。 新しいユーザにロールを割り当てる準備として、ロールについての実体験に基づく知識を得るために標準ユーザを使用します。 各標準ユーザの認証情報でログインして、さまざまなロールに関連付けられる権限を使用するときに標準ユーザがどんな経験するのかを実際に知ることができます。

次の手順に従ってください:

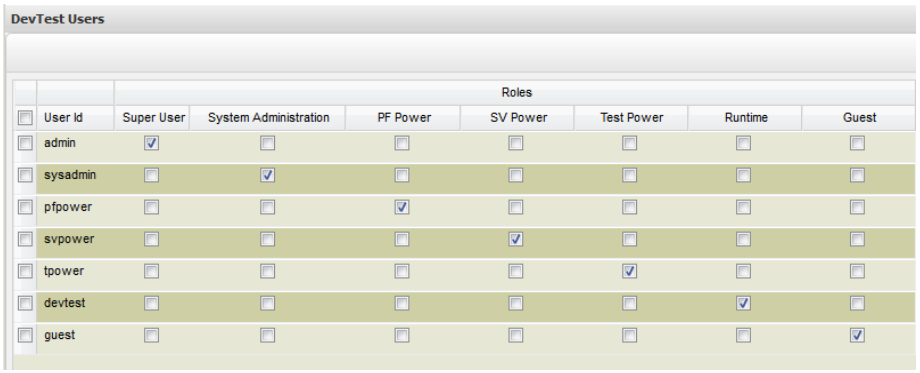
1. サーバ コンソールにアクセスして、スーパー ユーザ ロールを持つユーザとしてログインします。

   `http://hostname:1505`

2. 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。

3. ［ユーザ］をクリックします。

   標準ユーザが表示されます。



   デフォルトのパスワードは、各標準ユーザのユーザ ID と同じです。

   - スーパー ユーザ： admin、admin
   - PF パワー： pfpower、pfpower
   - SV パワー： svpower、svpower
   - テスト パワー： tpower、tpower
   - Runtime： devtest、devtest
   - システム管理： sysadmin、sysadmin
   - ゲスト： guest、guest

4. 各 UI および CLI に、スーパー ユーザの認証情報でログインします。
5. このロールが関連付けられたユーザがアクセスできる機能を検証します。
6. 他の標準ユーザで、最後の 2 つの手順を繰り返します。

このプロセスにより、アクセス制御（ACL）のセットアップに向けた準備が可能になります。 アクセス制御は、ユーザを追加し、そのユーザが実行するタスクに必要な権限のみを付与するロールを割り当てることによって、セットアップします。

重要: アクセス制御の一部として、標準ユーザのパスワードを変更する (P. 135)ことをお勧めします。パスワードの変更は、許可されていないユーザによるシステムへのアクセスを防止するための 1 つの方法です。

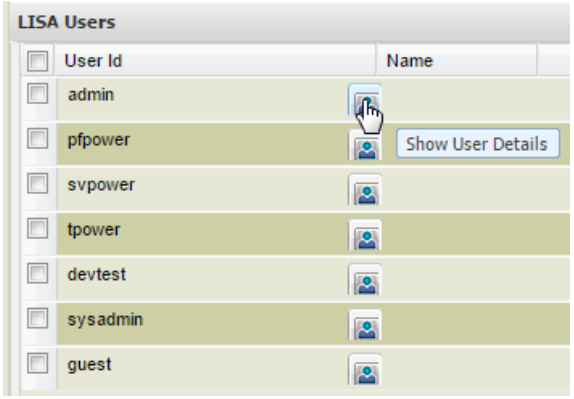
標準ユーザのパスワードの変更

初めてレジストリを開始するときに、7 つの標準ユーザが作成されます。各標準ユーザには、ロール、ユーザ タイプ、およびデフォルトのパスワードがを割り当てられます。 不正アクセスを防ぐために、これらのユーザのデフォルト パスワードをできるだけ早く変更することをお勧めします。

次の手順に従ってください:

1. DevTest Solutions が実行されていることを確認します。「Start the Server Components」を参照してください。
2. DevTest コンソールにアクセスします。
   `http://localhost:1505`
3. DevTest コンソールにログインします。 認証情報を持っていない場合は、以下のいずれかの方法をとります。
   - 認証に LDAP を使用することを計画している場合は、標準のスーパー ユーザでログインします。
   - ユーザの認証情報を定義することを計画しているが、まだ自身をユーザとして定義していない場合は、スーパー ユーザ ロールでユーザを作成します。
4. ［サーバ コンソール］をクリックします。
5. ［管理］ナビゲーション タブをクリックします。
6. ［ユーザ］をクリックします。

   標準ユーザが表示されます。
7. 各標準ユーザに対して、以下の手順に従います。
   a. いずれかの標準ユーザの［ユーザ詳細を表示］をクリックします。



b. ［パスワード］フィールドに、新しいパスワードを入力します。
c. ［パスワードの再入力］フィールドに、新しいパスワードを入力します。
d. ［保存］をクリックします。

8. ［ログアウト］をクリックします。

DevTest ワークステーション からのユーザ情報の表示

DevTest ワークステーション にログインしている場合は、以下の情報を提供するダイアログ ボックスを開くことができます。

- ユーザ名の一意の ID
- 割り当てられているロール
- 割り当てられている権限

次の手順に従ってください:

1. メイン メニューから［システム］-［セキュリティ権限の表示］を選択します。

   ［ユーザのセキュリティ権限］ダイアログ ボックスが表示されます。

2. 表示された情報を確認し終わったら、ダイアログ ボックスを閉じます。

ユーザとロールの管理

スーパー ユーザ アクセス権を持った管理者は、サーバ コンソール内の［管理］パネルからユーザおよびロールを管理します。 以下の方法について検討してください。

1. ロールおよびユーザ タイプの確認。
   a. ロールの選択。
   b. ［権限］タブに、使用可能なロールが関連付けられたユーザ タイプと共に表示されることに注目します。 たとえば、テスト管理者ロールは SV パワー ユーザ ユーザ タイプに関連付けられます。
   c. ライセンス契約書は、各ユーザ タイプで許可されている同時ユーザの最大数を指定します。 ユーザにロールを割り当てる際に、この図を思い出します。 多くのロールが同じユーザ タイプに対応することに注目します。
   d. 各ロールに関連付けられた権限を検討し、必要に応じて、権限をカスタマイズします。
   e. 必要に応じて、標準的なロールをベースにして新しいロールを追加します。 新しいロールは、そのロールがベースにするロールのユーザ タイプを継承します。
2. DevTest ユーザをすべて追加します。
   a. ユーザを選択します。
   b. DevTest は、指名されたユーザを認証するために、お客様が指定したユーザ ID およびパスワードを使用します。
   c. DevTest は、お客様が指定したロールを使用してユーザを認証し、ユーザが与えられた権限に基づいてさまざまなアクティビティを実行できるようにします。
3. 適切なロールを使用してすべてのユーザを設定し、その内容が満足のいくものであることを確認します。 既存のロールをカスタマイズした場合や、新しいロールを追加した場合は、設定を確認します。
4. 社内の保守ポリシーに基づき、戦略的に**データベースをバックアップ**します。 バックアップは、データベースが破損した場合に、ユーザおよびロールの設定の迅速な回復を可能にする事前措置です。 破損した ACL データベースの解決とはリストアの実行のことであり、これはデータベース管理者が管理する必要があります。

詳細:

ロールの優先度の変更 (P. 143)
監査ログの表示 (P. 144)
adduser コマンドライン ユーティリティ (P. 145)
ロールの追加、更新、および削除 (P. 141)
ユーザの追加、更新、および削除 (P. 139)

ユーザの追加、更新、および削除

サーバ コンソール (P. 168)を使用して、ユーザの追加、ユーザの詳細の変更、およびユーザの削除を行うことができます。 変更できる詳細にはパスワードが含まれます。

ユーザ ID とパスワードは、大文字と小文字の区別に関して異なります。

- ユーザ ID では、大文字と小文字が区別されません。 たとえば、ユーザ ID **AAAA1** と **aaaa1** は同じ値として扱われます。
- パスワードでは、大文字と小文字が区別されます。

現在ログインしているユーザ自身は削除できません。

［ユーザ］ウィンドウでは、任意の列の表示/非表示を切り替えることができます。 列のドロップダウン矢印をクリックします。 ［列を表示/非表示］を選択します。適切なチェック ボックスをオンまたはオフにします。

ユーザを追加する方法

1. サーバ コンソールで［管理］パネルを表示します。
2. ［ユーザ］ノードをクリックします。
3. 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。

   ［ユーザの追加］ダイアログ ボックスが表示されます。
4. ［ユーザ ID］フィールドで、ユーザの一意の ID を入力します。

   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（@）の任意の組み合わせを入力できます。 最大文字数は 100 です。
5. ［パスワード］フィールドに、ユーザのパスワードを入力します。

   英数字、ハイフン（-）、アンダースコア（_）、およびアンパサンド（@）の任意の組み合わせを入力できます。
6. ［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。
7. ［名前］フィールドに、ユーザ名を入力します。

   英数字、ハイフン（-）、アンダースコア（_）、および空白文字の任意の組み合わせを入力できます。 最大文字数は 100 です。
8. ［その他の情報］フィールドに、任意の追加情報を入力します。

   最大文字数は 600 です。

9. ［ユーザのロール］領域で、ユーザに割り当てるロールを 1 つ以上選択します。
10. ［ユーザの追加］をクリックします。

    「ユーザの追加」メッセージが表示されます。
11. ［OK］をクリックします。

ユーザを更新する方法

1. ユーザ ID の右側にある［ユーザ詳細を表示］アイコンをクリックします。

    ［ユーザ詳細］ダイアログ ボックスが表示されます。
2. 適切な変更を行います。

    ロール名をクリックすると、ロールの権限を表示できます。
3. ［保存］をクリックします。

ユーザを削除する方法

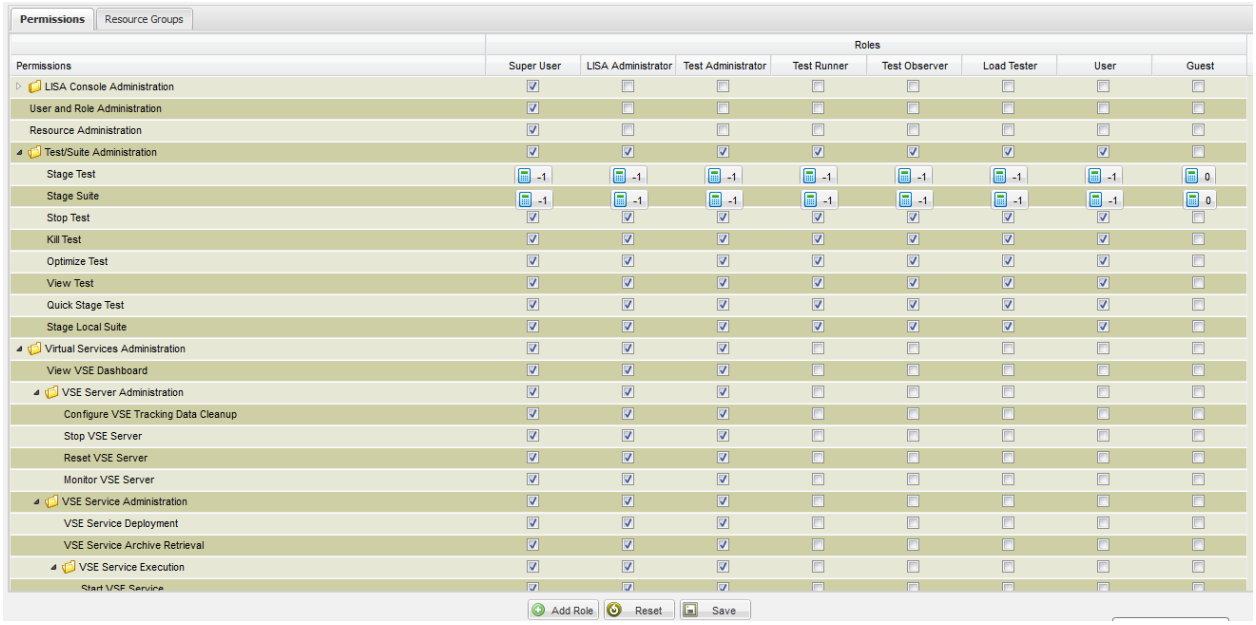
1. ユーザ ID の左側にあるチェック ボックスをオンにします。
2. ［ユーザの削除］をクリックします。
3. ［はい］をクリックします。

ロールの追加、更新、および削除

サーバ コンソール (P. 168)を使用して、ロールの追加、ロールの詳細の変更、およびロールの削除を行うことができます。

サーバ コンソールには、ロールおよび権限がグリッド形式で表示されます。

- 権限は、左側に行として表示されます。
- ロールは、上部に列として表示されます。 左端のロールには、最も高い優先度が与えられています。 右端のロールには、最も低い優先度が与えられています。 ユーザが複数のロールを持つ場合、その順序が重要になります。 サーバ コンソールでは、優先度を変更 (P. 143) できます。



親権限をオンにすると、子権限も自動的にオンになります。 親権限をオフにしても、子権限は自動的にオフに*なりません*。 子権限をオフにすると、親権限も自動的にオフになります。

［ロール］ウィンドウでは、任意の列の表示/非表示を切り替えることができます。 ［権限］列のドロップダウン矢印をクリックします。 列の上にマウス ポインタを置くと、［権限］列にドロップダウン矢印が表示されます。 ［列を表示/非表示］を選択します。 適切なチェック ボックスをオンまたはオフにします。

ロールを追加する方法

1. サーバ コンソールで［管理］パネルを表示します。
2. ［ロール］ノードをクリックします。
3. 右側のパネルの下部にある［ロールの追加］をクリックします。

   ［ロールの追加］ダイアログ ボックスが表示されます。
4. ［名前］フィールドに、ロール名を入力します。

   英数字、ハイフン（-）、アンダースコア（_）、および空白文字の任意の組み合わせを入力できます。 最大文字数は 50 です。
5. ［説明］フィールドに、ロールの説明を入力します。

   最大文字数は 200 です。
6. ［新しいロールの権限］領域で、ロールに権限を割り当てます。

   ドロップダウン ボックスを使用して、新しいロールのベースとなる既存のロールを選択することができます。
7. ［ロールの追加］をクリックします。

   新しいロールの列が追加されます。 この列は、既存のロールの右側に表示されます。

ロールを更新する方法

1. ロールと権限のグリッドでロールの列を見つけます。
2. ブール値 (P. 104)権限を更新するには、チェック ボックスをオンまたはオフにします。
3. 数値制限 (P. 104)権限を更新する方法
   a. ［数値制限］アイコンをクリックします。
   b. ［... の最大数］フィールドでは、［無制限］または［なし］を選択するか、数値制限の値を入力します。
   c. ［保存］をクリックします。
4. リスト項目 (P. 104)権限を更新する方法
   a. ［リスト項目］アイコンをクリックします。
   b. 必要に応じて、リスト項目を追加、編集、または削除します。
   c. ［保存］をクリックします。
5. ［保存］をクリックします。

ロールを削除する方法

1. ロールと権限のグリッドで、ロール名を選択し、ドロップダウン矢印をクリックします。
2. ［ロールの削除］をクリックします。
3. ［はい］をクリックします。

ロールの優先度の変更

サーバ コンソールには、ロールが優先度の順に表示されます。 ユーザが複数のロールを持つ場合、その順序が重要になります。

次の手順に従ってください:

1. サーバ コンソールで［管理］パネルを表示します。
2. 以下のいずれかの操作を実行します。
   - ［ロール］ノードをクリックします。
   - ［ロールの表示］をクリックします。
3. ［優先度の再設定］をクリックします。

   ［ロール優先度のリセット］ダイアログ ボックスが表示されます。
4. ロールの順序を変更するには、ロールを選択して目的の位置までドラッグします。
5. ［順序の保存］をクリックします。

監査ログの表示

監査ログには、ACL によって制御されるアクティビティの一連のエントリが含まれます。 各エントリには以下の情報が含まれます。

タイムスタンプ

エントリが作成された日時

ユーザ ID

ユーザの一意の ID

ロール

ユーザに割り当てられているロール

権限 ID

権限の数字の識別子

権限名

アクティビティを制御する権限

ステータス

アクティビティが許可されているか、拒否されているか

実行先

サーバ コンポーネント、テスト ケース、または仮想サービス モデルの名前（使用可能な場合）

その他詳細

詳細情報（使用可能な場合）

次の手順に従ってください:

1. サーバ コンソールで［管理］パネルを表示します。
2. 以下のいずれかの操作を実行します。
   - ［監査ログ］ノードをクリックします。
   - ［監査ログの表示］をクリックします。
3. ［リフレッシュ］をクリックします。

adduser コマンドライン ユーティリティ

adduser コマンドライン ユーティリティを使用してユーザを追加できます。 このユーティリティは、DevTest サーバ インストールの **LISA_HOME¥bin** ディレクトリにあります。このユーティリティを使用するには、ユーザおよびロール管理権限が必要です。

このユーティリティの形式は以下のとおりです。

```
adduser -d userid -w password [-r role] -u userid -p password [-m
registry_url]
```

以下のオプションにより、新しいユーザに基本情報を割り当てることができます。

- **-d** または **--adduser** オプションを使用して、新しいユーザにユーザ ID を割り当てます。
- **-w** または **--addpassword** オプションを使用して、新しいユーザにパスワードを割り当てます。
- （オプション） **-r** または **--rolename** オプションを使用して、新しいユーザにロールを割り当てます。 ロールを割り当てない場合、サーバ コンソールからロールが割り当てられるまで、そのユーザは権限を持ちません。

以下のオプションにより、認証情報を指定できます。

- **-u** または **--username** オプションを使用して、ユーザ ID を指定します。
- **-p** または **--password** オプションを使用して、ユーザのパスワードを指定します。

レジストリがリモート コンピュータ上にある場合、**-m** または **--registry** オプションを使用して、レジストリ URL を指定します。

例

以下の例では、**user1** という名前のユーザを追加します。 ロール名に複数の単語が含まれているため、引用符が使用されています。

```
adduser -d user1 -w password1 -r "Load Tester" -u admin -p
myadminpassword
```

LDAP 認証を使用するための ACL の設定

DevTest データベースではなく LDAP サーバ内の情報に基づいてユーザ認証が行われるように、アクセス制御（ACL）を設定できます。 認証プロセスは、引き続き DevTest データベースを使用します。 ACL 管理者は、以下で説明するプロパティ項目に基づく設定と実装について LDAP 管理者と協議する必要があります。

注: LDAP サーバを使用するように ACL を設定すると、ユーザはユーザの LDAP データベースに含まれるアカウントでのみログインできます。 DevTest UI または CLI へのログインに、標準ユーザ (P. 132)の認証情報は使用できません。

LDAP が正常にユーザを認証し、そのユーザが DevTest データベースに存在しない場合、そのユーザはデータベースに自動的に追加されます。

設定プロセス中に、以下のプロパティを追加します。

lisa.acl.ldap.ldapUrl

LDAP サーバの URL。

lisa.acl.ldap.securityPrincipal

セキュリティ プリンシパルの識別名。

lisa.acl.ldap.securityCredential

セキュリティ プリンシパルのパスワード。 レジストリは、起動時に **_enc** という文字列をプロパティ名に追加し、値を暗号化します。

lisa.acl.ldap.securityAuthentication

使用するセキュリティ レベル。 有効な値は **none** および **simple** です。 値を **none** に設定した場合は、ユーザが指定したパスワードが LDAP 認証コールで無視され、ユーザ名のみが検証されます。 また、**lisa.acl.ldap.securityPrincipal** プロパティや **lisa.acl.ldap.securityCredential** プロパティを含める必要もありません。 値を **simple** に設定した場合は、ユーザ名およびパスワード（クリア テキストとして渡されます）が検証されます。

lisa.acl.ldap.baseContext

ユーザ検索が開始されるノードの識別名。

lisa.acl.ldap.userSearchFilter

ユーザ エントリのオブジェクト クラスを指定する検索フィルタ。例：**(objectClass=user)**。

lisa.acl.ldap.usernameAttribute

ユーザ名を指定する属性。例： **sAMAccountName**。

lisa.acl.ldap.userSearchAllDepths

すべてのサブノードを検索するかどうかを示します。 有効な値は **true** および **false** です。

lisa.acl.ldap.lisaDefaultRole

正常に認証された後で DevTest データベースに追加されたユーザに割り当てられるデフォルト ロール。 このプロパティを含めない場合、デフォルト ロールはゲストです。

lisa.acl.ldap.referralSupport

LDAP サーバでリフェラルが使用されている場合、このプロパティを使用してリフェラルのタイプを指定できます。有効な値は、**follow**、**ignore**、および **throw** です。 このプロパティを含めない場合、デフォルト値は **false** になります。

注: プロパティ ファイルに **lisa.acl.auth.enabled** プロパティも含まれていて、その値が **true** である場合、LDAP 認証は正しく動作しません。 **lisa.acl.auth.enabled** プロパティを削除するか、コメント アウトしてください。

LDAP 認証では、デフォルトの ACL モジュールで提供されるものと同じログイン ダイアログ ボックスを使用します。

次の手順に従ってください:

1. レジストリが配置されているコンピュータで **local.properties** ファイルまたは **site.properties** ファイルを開きます。
2. 以下の行を追加します。

   ```
   lisa.acl.auth.module.impl=com.itko.lisa.acl.custom.BaseLDAPAuthenticationModule
   ```

3. このトピックですでに説明した **lisa.acl.ldap.*** プロパティを追加します。 以下に例を示します。

   ```
   lisa.acl.ldap.ldapUrl=ldap://172.24.255.255:389
   ```

   ```
   lisa.acl.ldap.securityPrincipal=CN=admin,OU=users,DC=example,DC=com
   ```

   ```
   lisa.acl.ldap.securityCredential=adminpwd
   ```

```
lisa.acl.ldap.securityAuthentication=simple
lisa.acl.ldap.baseContext=OU=users,DC=example,DC=com
lisa.acl.ldap.userSearchFilter=(objectClass=user)
lisa.acl.ldap.usernameAttribute=sAMAccountName
lisa.acl.ldap.userSearchAllDepths=true
lisa.acl.ldap.lisaDefaultRole=DevTest Administrator
```

4. **local.properties** ファイルをまたは **site.properties** ファイルを保存します。
5. レジストリを起動します。

LDAP で認証されたユーザへの権限付与

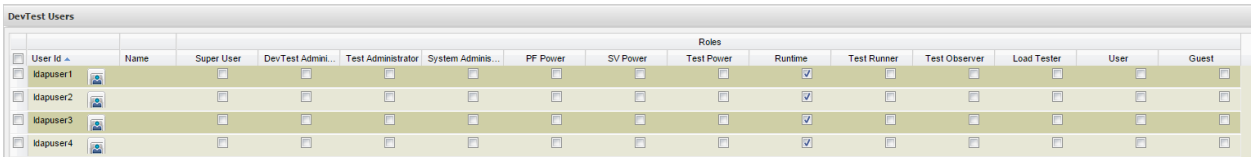
ACL をユーザの LDAP に設定する場合は、以下の設定でデフォルト ロールを指定します。

lisa.acl.ldap.lisaDefaultRole

正常に認証された後で DevTest データベースに追加されたユーザに割り当てられるデフォルト ロール。このプロパティを含めない場合、デフォルト ロールはゲストです。

ユーザが有効な LDAP 認証情報でログインすると、LDAP はユーザを認証し、そのユーザが DevTest データベースに存在しない場合は、そのユーザをデータベースに自動的に追加します。 行にはユーザ ID、および **lisa.acl.ldap.lisaDefaultRole** で現在指定されているロールが含まれます。

13 個のロールがあります。 たとえば、デフォルト ロールを Runtime に設定できます。 各ユーザが最初に DevTest UI または CLI にログオンしたときに、ユーザ ID とデフォルト ロールを含む行が、DevTest Users テーブルに追加されます。



個々のロール割り当てにより、DevTest Users テーブルを更新する方法

1. すべての DevTest ユーザに、DevTest Solutions へのログインとその後のログアウトを依頼します。
2. DevTest コンソールにアクセスして、ログインします。

   `http://hostname:1505`
3. ［サーバ コンソール］をクリックし、左のナビゲーション ペインで［管理］タブをクリックします。
4. ［ユーザ］をクリックします。

   DevTest Users テーブルが開きます。
5. ユーザごとに、デフォルト ロールをクリアして、適切なロールを選択します。
6. （オプション）［ユーザ詳細］ダイアログ ボックスの右ペインにロールの権限を表示するには、［ユーザ詳細を表示］アイコンをクリックしロール名をクリックします。
7. ［保存］をクリックします。

DevTest Users テーブルを自動的に更新する方法

1. デフォルトを特定のロールに設定します。
2. 該当するロールを割り当てたい、すべてのユーザに DevTest Solutions へのログインとその後のログアウトを依頼します。
3. デフォルトを別のロールに変更します。
4. 該当するロールを割り当てたい、すべてのユーザに DevTest Solutions へのログインとその後のログアウトを依頼します。
5. 各ロールで、手順 3 および 4 を繰り返します。

リソース グループ

リソース グループは、1 つ以上の DevTest サーバ または VSE です。ユーザまたはプロジェクトがアクセスできるリソースを決定するために、リソース グループを定義します。

ACL を使用する場合は、どのロールがどのリソースを使用できるかを決定するために、ロールとリソース グループを関連付けます。

このセクションには、以下のトピックが含まれます。

リソース グループの管理

サーバ コンソールを使用して、リソース グループの追加、リソース グループの詳細の変更、リソース グループの削除、およびリソース グループからのリソースの削除を行います。

［リソース グループ］ウィンドウでは、任意の列の表示/非表示を切り替えることができます。 列のドロップダウン矢印をクリックします。 ［列を表示/非表示］を選択します。 適切なチェック ボックスをオンまたはオフにします。

リソース グループを追加する方法

1. サーバ コンソールで［管理］パネルを表示します。
2. ［リソース グループ］ノードをクリックします。
3. 右側のパネルの下部にある［リソース グループの追加］をクリックします。 ［リソース グループの追加］ダイアログ ボックスが表示されます。
4. ［名前］フィールドに、リソース グループの一意の名前を入力します。名前には、英数字、スペース、ハイフン、またはアンダースコアのみを含める必要があります。
5. ［説明］フィールドに、リソース グループの説明を入力します。
6. グループの隣にあるチェック ボックスをオンにすることにより、リソース グループ（複数可）を選択します。
7. ［追加］をクリックします。

   ［リソース グループ］パネルにリソースが表示され、関連付けられているリソース グループが［リソース グループ］列に表示されます。

リソース グループを表示する方法

1. 表示をリフレッシュするには、ウィンドウの右上隅にある［リフレッシュ］ボタンをクリックします。
2. 非アクティブであるが、まだリソース グループに関連付けられているリソースは、「取り消し線」を使用したラベルで表示されます。

リソース グループを削除する方法

1. リソース グループの名前の隣にある黒い 3 角形を選択します。
2. ドロップダウン リストから、［リソース グループの削除］を選択します。

3. ［リソース グループの削除中］ダイアログ ボックスで、［はい］をクリックします。

リソース グループからリソースを削除する方法

注: リソースがアクティブである場合は、リソース グループからリソースを削除できません。

リソース グループの隣にあるチェック ボックスをオフにして、［保存］をクリックします。

リソース グループへのロールの付与

リソース グループにロールを付与するには、サーバ コンソールを使用します。

次の手順に従ってください:

1. ［管理］-［セキュリティ］-［リソース グループ］を選択します。
2. 各リソース グループに関連付けるロールのチェック ボックスをオンにします。
3. ［保存］をクリックします。

アクセスを制御するためのリソース グループの使用

リソースへのアクセスの制御は、アクセス制御（ACL）の他に、リソース グループを使用して行えます。

次の手順に従ってください:

1. 組織に適したリソース グループを追加します。手順については、「リソース グループの管理 (P. 151)」を参照してください。
2. DevTest ワークステーション で設定ファイルを開くには、プロジェクト パネルからダブルクリックします。
3. パネルの下部にある［追加］アイコンをクリックします。
4. プロパティ エディタで［キー］列をクリックします。
5. プロパティのリストから、［RESOURCE_GROUP］を選択します。
6. ［値］列をクリックします。
7. この設定に関連付けるリソース グループを選択します。

   ドロップダウン リストからは 1 つのリソース グループのみを選択できます。ただし、［値］列を編集すると、カンマ区切りでリソース グループのリストを入力できます。
8. 設定を保存するには、［保存］をクリックします。

この設定を使用してステージングするすべてのテスト ケース、テスト スイート、または VSM は、そのリソース グループ内のリソースを使用するように制限されます。

注: リソース グループの関連付けは、設定がアクティブな設定である場合にのみ有効です。

第 5 章: ログ記録

このセクションには、以下のトピックが含まれています。

ログ ファイルの概要

このセクションでは、以下のログ ファイル、およびそれらが格納されている場所について説明します。

- 主要ログ ファイル (P. 156)
- デモ サーバ ログ ファイル (P. 157)

主要ログ ファイル

主要ログ ファイルには、以下のものが含まれます。

- **coordinator.log**： コーディネータのログ出力
- **cvsmgr.log**： 継続的検証サービス（CVS）のログ出力
- **devtest_broker_pid.log**： DevTest Java エージェントのブローカ コンポーネントのログ出力
- **devtest_console_pid.log**： DevTest Java エージェントのコンソール コンポーネントのログ出力
- **marmaker.log**： Make Mar コマンドライン ユーティリティのログ出力
- **pfbroker.log**： DevTest Java エージェントのブローカ コンポーネントの累積ログ出力
- **portal.log**： DevTest ポータルのログ出力
- **registry.log**： レジストリのログ出力
- **simulator.log**： シミュレータのログ出力
- **svcimgmgr.log**： サービス イメージ マネージャ コマンドライン ユーティリティのログ出力
- **svcmgr.log**： サービス マネージャ コマンドライン ユーティリティのログ出力
- **trunner.log**： テスト ランナー コマンドライン ユーティリティのログ出力
- **vse.log**： VSE サーバのログ出力。
- **vsemgr.log**： VSE マネージャ コマンドライン ユーティリティのログ出力
- **vse_xxx.log**： VSE 会話およびサービス イメージ ナビゲーションのログ出力（*xxx* はサービス イメージ名）
- **workstation.log**： DevTest ワークステーション のログ出力

lisa.tmpdir プロパティは、これらのログ ファイルの場所を制御します。
DevTest ワークステーション から **lisa.tmpdir** プロパティの値を参照する方法

1. メイン メニューから［ヘルプ］-［DevTest ランタイム情報］をクリックします。
2. ［システムのプロパティ］タブで、**lisa.tmpdir** を見つけます。

レジストリ、コーディネータ、シミュレータ、または VSE が Windows サービスとして実行されている場合 (P. 20)、ログ ファイルは **LISA_HOME¥lisatmp** ディレクトリにあります。

注: **lisa.tmpdir** は、一時ファイルを格納できる場所の変更を可能にしますが、このプロパティを変更して、外部のマウント ポイントまたは外部共有に一時ファイルを保存することはお勧めしません。 一時ファイルの保存に外部共有を使用しているケースでご使用の製品が不安定な場合、環境を引き続きサポートするために、サポートが以前のように一時ファイルを、ローカルディスクを使用して保存するように指示することがあります。

デモ サーバ ログ ファイル

デモ サーバには固有のログファイルがあります。それらは、**lisa-demo-server/jboss/server/default/log** ディレクトリにあります。

この場所は、**lisa.tmpdir** プロパティで制御されません。

デモ サーバには、以下のログ ファイルがあります。

- boot.log
- server.log

ログ プロパティ ファイル

DevTest は、Apache log4j ログ記録フレームワークを使用します。 **LISA_HOME** ディレクトリの **logging.properties** ファイルでは、ログの動作を設定できます。

DevTest ワークステーション からより多くのログ情報を取得するには、**log4j.rootCategory** ファイルでログ レベルを変更します。

```
log4j.rootCategory=INFO,A1
```

このファイルには、DevTest に含まれるサードパーティ コンポーネントのための一連のロガーが含まれています。 これらのロガーのデフォルト ログ レベルは、これらのサードパーティ コンポーネントによる大量のメッセージでログ ファイルがいっぱいになることを防止するように意図されています。 通常は、ログ レベルを変更する必要はありません。

```
log4j.logger.com.teamdev=WARN
log4j.logger.EventLogger=WARN
log4j.logger.org.apache=ERROR
log4j.logger.com.smardec=ERROR
log4j.logger.org.apache.http=ERROR
log4j.logger.org.apache.http.header=ERROR
log4j.logger.org.apache.http.wire=ERROR
log4j.logger.com.mchange.v2=ERROR
log4j.logger.org.hibernate=WARN
log4j.logger.org.jfree=ERROR
log4j.logger.com.jniwrapper=ERROR
log4j.logger.sun.rmi=INFO
```

デフォルトのアペンダは **com.itko.util.log4j.TimedRollingFileAppender** です。 コンポーネントに対するログ ステートメントは、一定のサイズに達するとバックアップされるファイルに追加されます。 デフォルトの最大ファイル サイズは 10 MB です。 バックアップ ファイルのデフォルトの数は 5 つです。

```
log4j.appender.A1=com.itko.util.log4j.TimedRollingFileAppender
log4j.appender.A1.File=${lisa.tmpdir}/${LISA_LOG}
log4j.appender.A1.MaxFileSize=10MB
log4j.appender.A1.MaxBackupIndex=5
log4j.appender.A1.layout=org.apache.log4j.EnhancedPatternLayout
log4j.appender.A1.layout.ConversionPattern=%d{ISO8601}{UTC}Z (%d{HH:mm})
[%t] %-5p %-30c - %m%n
```

バックアップ ファイルは、ログ ファイルと同じディレクトリに配置されます。 たとえば、レジストリのログ ファイルが 3 回バックアップされている場合、ディレクトリには以下のファイルが含まれます。

- registry.log
- registry.log.1
- registry.log.2
- registry.log.3

レイアウトは、ログ ステートメントの形式を制御します。 デフォルトのレイアウトは **oorg.apache.log4j.EnhancedPatternLayout** です。 デフォルトの変換パターンは **%d{ISO8601}{UTC}Z (%d{HH:mm}) [%t] %-5p %-30c - %m%n** です。 この変換パターンは、ログ ステートメントに日付、スレッド、優先度、カテゴリ、およびメッセージが含まれることを指定します。 以下に例を示します。

```
2014-11-20 14:09:08,152Z (07:09) [main] INFO  com.itko.lisa.net.ActiveMQFactory
- Starting amq broker
```

日付には、協定世界時（UTC）を使用します。 この規則により、レジストリが DevTest ワークステーション とは異なるタイム ゾーンで実行されている場合に、ログ イベントを容易に追跡できます。

スレッド ダンプ プロパティの詳細については、「自動スレッド ダンプ (P. 160)」を参照してください。

サーバ コンポーネント用ステータス メッセージ

以下のサーバ コンポーネントは、指定された間隔でそれぞれのログ ファイルにステータス メッセージを書き込みます。

- レジストリ
- コーディネータ
- シミュレータ
- VSE

以下の例は、レジストリによってログ ファイルに書き込まれたものです。

```
2012-03-05 12:48:01,136 [Event Sink Thread Pool Thread 1] INFO
com.itko.lisa.coordinator.TestRegistryImpl -
Coordinator Servers: 0 Simulator Servers: 0 VSEs: 0 Running vusers: 0 Labs: 1
Memory used 83mb, allocated 158mb, max 227mb (36%) labSims: 0 labVSEs: 0 labCoords:
0
Our cpu usage 0%, system cpu used 16%
```

ステータス メッセージのデフォルトの間隔は 30 秒です。 **lisa.properties** ファイル内の以下のプロパティを編集して、間隔を変更できます。

- lisa.defaultRegistry.pulseInterval
- lisa.coordinator.pulseInterval
- lisa.simulator.pulseInterval
- lisa.vse.pulseInterval

自動スレッド ダンプ

logging.properties ファイルを使用して自動スレッド ダンプを有効にすることができます。これは、パフォーマンスの問題のデバッグに役立ちます。

以下のプロパティを見つけ、**WARN** を **INFO** に変更します。

```
log4j.logger.threadDumpLogger=WARN, THREAD_DUMPS
```

スレッド ダンプを無効にするには、**INFO** を **WARN** に戻します。

スレッド ダンプのデフォルトの間隔は 30 秒です。**lisa.properties** ファイル内の **lisa.threadDump.interval** プロパティを編集して、間隔を変更できます。

テスト ステップ ロガー

「*CA Application Test の使用*」の「テスト ステップのエレメント」で説明されているように、各テスト ステップには 1 つのログ メッセージ エレメントが含まれます。

実行時に特定のファイルにログ メッセージを送信するには、**LISA_HOME** ディレクトリの **logging.properties** ファイルに以下のプロパティを追加します。

```
log4j.logger.com.itko.lisa.test.StepLogger=DEBUG, A2
log4j.additivity.com.itko.lisa.test.StepLogger=false
log4j.appender.A2=org.apache.log4j.RollingFileAppender
log4j.appender.A2.File=${lisa.tmpdir}/log.log
log4j.appender.A2.MaxFileSize=10MB
log4j.appender.A2.MaxBackupIndex=5
log4j.appender.A2.layout=org.apache.log4j.PatternLayout
log4j.appender.A2.layout.ConversionPattern=%d [%t] %-5p %-30c - %m%n
```

File、**MaxFileSize**、および **MaxBackupIndex** プロパティの値は、上記の例で表示される値とは異なる可能性があります。

注: これらのプロパティを追加したら、DevTest ワークステーション を再起動します（現在実行されている場合）。

結果として生じるログ ファイル内のメッセージは、以下の例のようになります。

```
2012-07-11 17:32:59,390 [basic-test/basic-test [QuickStageRun]/0] DEBUG
com.itko.lisa.test.StepLogger  -
LOG basic-test,basic-test [QuickStageRun],local,0,3,my log message
```

LOG に続くメッセージの部分は、6 つのコンポーネントから構成されます。

- テスト ケースの名前。
- ステージング ドキュメントの名前。
- シミュレータの名前。
- インスタンス/仮想ユーザ番号。 10 人の仮想ユーザ テストでは、この番号は 1 ～ 10 の間で変化します。
- サイクル番号。 この番号は、ある状態が発生するまでテストを何度も実行するようにステージング ドキュメントが設定されている状況に適用されます。 値は、この特定の仮想ユーザがテストを実行した回数です。
- テスト ステップ内に設定されたログ メッセージ

第 6 章: 監視

サービス マネージャ コマンドライン ユーティリティ、Web ベースのサーバ コンソール、レジストリ モニタ、またはエンタープライズ ダッシュボードを使用することによって、DevTest Solutions をモニタできます。

このセクションには、以下のトピックが含まれています。

Service Manager

ServiceManager コマンドライン ユーティリティを使用して、レジストリ、コーディネータ、シミュレータ、または VSE サーバでさまざまなアクションを実行できます。

```
ServiceManager [--command]=service-name
```

service-name は、影響を与えるサービスの名前です。

このコマンドと名前のペアは、繰り返すことができます。

名前を検索するには、lisa.properties キーを二重中かっこで囲みます。 以下に例を示します。

```
{{lisa.registryName}}
```

サービス名の例

- tcp://localhost:2010/Registry
- Simulator （tcp://localhost:2014/Simulator に解決されます）

サービス マネージャ オプション

-h、--help

ヘルプ テキストを表示します。

-s service-name、--status=service-name

サービスに関するステータス メッセージを表示します。 service-name に *all* を入力すると、すべての登録済みサービスに関するステータス メッセージが返されます。

-r service-name、--reset=service-name

サービスをメモリに保持し、状態をリフレッシュします。

-o service-name、--stop=service-name

終了するようにサービスに指示します。

-i valid-remote-init-service-name、--initialize=valid-remote-init-service-name

サービスをリモートで初期化します。

-t service-name、--threaddump=service-name

診断用スレッド ダンプ（スタック トレース）を生成するようにサービスに指示します。

-b simulator-name、--attached=simulator-name

接続されているモバイル デバイスのリストを返すように simulator-name に指示します。

-e service-name、--heapdump=service-name

メモリ診断用 .hprof ファイルを作成するようにサービスに指示します。

-g service-name、--gc=service-name

Java ガベージ コレクションを強制するようにサービスに指示します。

-d service-name、--diagnostic=service-name

サービス用診断ファイルが含まれる zip ファイルを作成します。 サービスがレジストリである場合、この zip ファイルには、接続しているすべてのコーディネータ、シミュレータ、および VSE サーバ用診断ファイルも含まれます。 通常、このオプションは、サポートによって要求されたときに使用します。

loglevel

このオプションには、**loglevel** という名前のセカンダリ パラメータも含まれます。loglevel に続けて、キーワード（error、warn、info、debug、trace のいずれか）を指定します。

たとえば、**ServiceManager -d tcp://10.1.1.23:2010/Registry loglevel debug** は、エージェントを含めたすべてのコンポーネントのログ レベルをデバッグ レベルに設定します。その後、サービス マネージャは以下のメッセージを書き込みます。

> ログ レベルがデバッグに設定されました。 再現ステップを実行した後、リターン キーを押してログ レベルをリストアして診断をキャプチャしてください。

生成される ZIP ファイルには、接続しているすべてのコンポーネントおよびスレッド ダンプに対するデバッグ ログ レベルのログ、ライセンス情報、およびプロパティが含まれます。また、この zip ファイルには、レジストリ ブローカに接続しているエージェントのログも含まれます。

注: VSE、コーディネータ、またはシミュレータ サーバに -d コマンドが送信された場合は、そのコンポーネントに対するログおよび診断のみが zip に含められます。

エージェントにはトレース ログ レベルはありませんが、dev ログ レベルがよく似ており、同様に処理されます。

-u **ユーザ名**、--username=**ユーザ名**

ユーザ名を指定するには、このコマンドを使用します。

-p **パスワード**、--password=**パスワード**

パスワードを指定するには、このコマンドを使用します。

--version

バージョン番号を出力します。

-m **レジストリ名**、--registry-name=**レジストリ名**

接続するレジストリを指定するために、*initialize* と共に使用されます。

-n **コンポーネント名**、--component-name=**レジストリ名**

コンポーネント名を指定するために、*initialize* と共に使用されます。

-l **ラボ名**、--lab-name=**ラボ名**

作成するラボ名を指定するために、*initialize* と共に使用されます。

-a **アプリケーション名**、--app=**アプリケーション名**

サーバのアプリケーション ID を指定するために、*initialize* と共に使用されます。

例:

MySim と命名するシミュレータがあり、*MyRegistry* にそのシミュレータを接続して、*MyDevLab* と命名する新しいラボを開始する場合は、以下のように入力します。

```
./SimulatorService -n MySim -m tcp://1.2.3.4:2010/MyRegistry -l
MyDevLab
```

別のシミュレータをそのラボに追加する場合は、以下のように入力します。

```
./SimulatorService –component-name=MySecondSim
–registry-name=tc;"//1.2.3.4:2010/MyRegistry –lab-name=MyDevLab
```

そのレジストリに VSE を追加するが、ラボは別のものにする場合は、以下のように入力します。

```
./VirtualServiceEnvironment -n CoreServices -m
tcp://1.2.3.4:2010/MyRegistry -l QA
```

サービス マネージャの例

以下の例は、レジストリのステータスを確認します。

```
ServiceManager -s Registry
Coordinator Servers: 1 Simulator Servers: 2 VSEs: 1 Running vusers: 0
Labs: 1 Memory used 76mb, allocated 155mb, max 253mb (30%)
labSims: 2 labVSEs: 1 labCoords: 1
```

以下の例は、すべての登録済みサービスのステータスを確認します。

```
Coordinator Server: tcp://bdert-mbp.local:2011/Coordinator
OK: 1 Coordinators running. Memory used 223mb, allocated 461mb, max 910mb (24%)
Our cpu usage 0%, system cpu used 8%
Simulator Server: tcp://bdert-mbp.local:2014/Simulator
OK: 1 Simulators running. Memory used 301mb, allocated 437mb, max 910mb (33%) Our
cpu usage 0%, system cpu used 8%
```

以下の例は、レジストリを停止します。

```
ServiceManager -o Registry
Sending stop request to Registry.
```

以下の例は、VSE サーバのスレッド ダンプを生成します。

```
ServiceManager --threaddump=tcp://remote.host.com:2013/VSE
< a bunch of stack traces >
```

以下の例は、VSE サーバの Java ガベージ コレクションを強制します。

```
ServiceManager --gc=VSE
After GC: Memory used 55mb, allocated 225mb, max 246mb (22%)
```

以下の例は、レジストリに接続しているすべてのコンポーネントに対するトレース レベル ログが含まれる zip ファイルを生成します。

```
ServiceManager --diagnostic=Registry loglevel TRACE
```

以下の例は、シミュレータに対するデバッグ レベル ログが含まれる zip ファイルを生成します。

```
ServiceManager -d Simulator loglevel debug
```

サーバ コンソールを開く

DevTest ワークステーション または Web ブラウザからサーバ コンソールを開くことができます。

DevTest ワークステーション からサーバ コンソールを開く方法

メイン メニューから［表示］-［サーバ コンソール］を選択します。

Web ブラウザからサーバ コンソールを開く方法

1. レジストリが実行されていることを確認します。
2. Web ブラウザに「**http://localhost:1505/**」と入力します。

   レジストリがリモート コンピュータ上にある場合は、**localhost** をそのコンピュータの名前または IP アドレスに置き換えます。

   DevTest コンソールが表示されます。
3. ［サーバ コンソール］をクリックします。

コンポーネント稼働状況サマリの表示

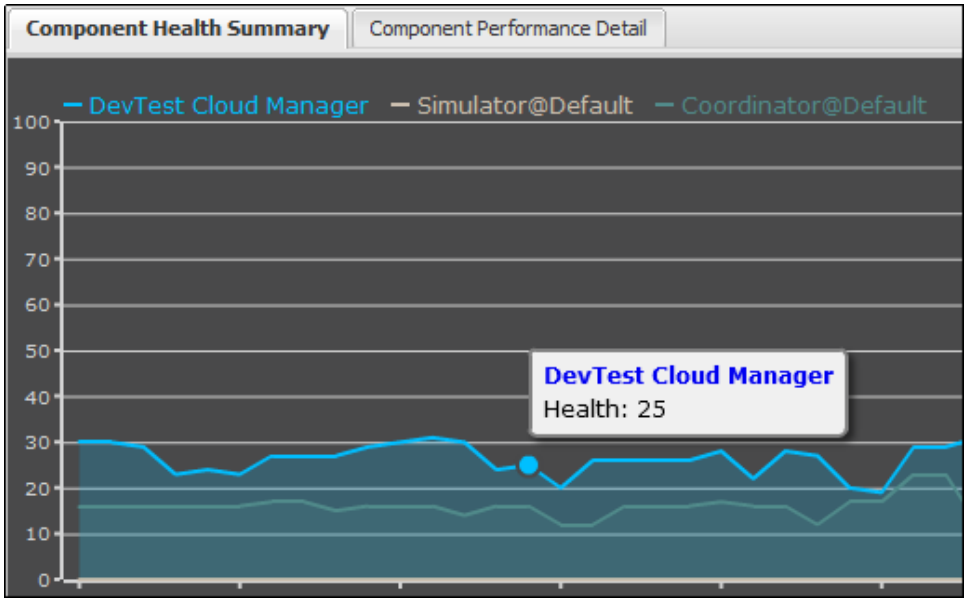
サーバ コンソールの［コンポーネント稼働状況サマリ］タブには、実行中のコンポーネントの全体的な稼働状況を示すインジケータが表示されます。

稼働状況インジケータは、0 ～ 100 のスケールで表示されます。 値は、以下の統計から派生します。

- 全体の CPU 負荷
- サーバが使用している CPU の量
- サーバが独自に使用しているヒープの量

値 0 は、アイドル状態のシステムを表します。 値 100 は、最大値に達しているシステムを表します。

以下の図は、［コンポーネント稼働状況サマリ］タブを示しています。 X 軸は、時間を表します。 Y 軸は、稼働状況インジケータの値を表します。



コンポーネント稼働状況サマリを表示する方法

1. サーバ コンソールに移動します。
2. DevTest ネットワークパネルで［DevTest クラウド マネージャ］ノードをクリックします。

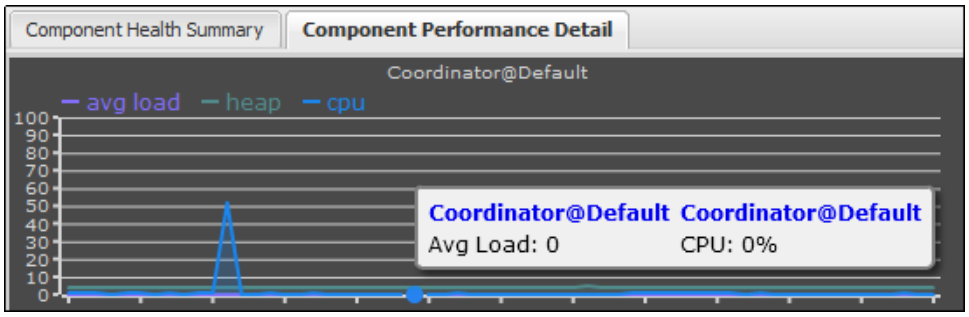
［コンポーネント稼働状況サマリ］タブにデータが表示されます。

コンポーネント パフォーマンス詳細の表示

サーバ コンソールの［コンポーネント パフォーマンス詳細］タブでは、実行中のラボ メンバに関する以下のパフォーマンス統計を表示できます。

- **平均負荷**： この値は、整数として表示されます。
- **ヒープ**： この値は、パーセンテージとして表示されます。
- **CPU**： この値は、パーセンテージとして表示されます。

以下の図は、Default ラボ内でのコーディネータのパフォーマンス データを示しています。 平均負荷および CPU メトリックのツールヒントが表示されています。



コンポーネント パフォーマンス詳細を表示する方法

1. サーバ コンソールに移動します。
2. DevTest ネットワーク パネルで［DevTest クラウド マネージャ］ノードをクリックします。
3. ［コンポーネント パフォーマンス詳細］タブをクリックします。
4. ネットワーク グラフ内のラボ メンバを右クリックし、［パフォーマンス データの表示］を選択します。

   ラボ メンバが［コンポーネント パフォーマンス詳細］タブに追加されます。メトリックは異なる色で表示されます。メトリックはそれぞれツールヒントを提供します。

ヒープ ダンプおよびスレッド ダンプの作成

サーバ コンソールでは、サーバ コンポーネントのヒープ ダンプおよびスレッド ダンプを作成できます。

通常、これらの手順は、テクニカル サポートによって要求されたときのみ実行します。

サービス マネージャ コマンドライン ユーティリティを使用してヒープ ダンプを作成することもできます。

ヒープ ダンプを作成する方法

1. サーバ コンソールに移動します。
2. ネットワーク グラフ内のサーバ コンポーネントを右クリックし、［ヒープをダンプ］を選択します。
3. メッセージが表示されたら、ファイルを保存します。
4. ファイルを開きます。

   このファイルには、.hprof ファイルの完全修飾パス名（例：**C:¥Users¥myusername¥lisatmp_6.0.7¥Coordinator_Server_HeapDump_2012-02-28_08-36-55.hprof**）が含まれます。

注: この機能は、Sun Java 6 Java ランタイムでのみ使用できます。 この診断ツールは、サポートがメモリ不足状態の原因を特定するうえで役立ちます。 メモリ不足状態が発生すると、ヒープが自動的にダンプされます。このボタンは、ヒープ ダンプを手動でトリガするためのものです。

スレッド ダンプを作成する方法

1. サーバ コンソールに移動します。
2. ネットワーク グラフ内のサーバ コンポーネントを右クリックし、［スレッドをダンプ］を選択します。
3. メッセージが表示されたら、スレッド ダンプ ファイルを保存します。
4. スレッド ダンプ ファイルを開き、内容を確認します。

ガベージ コレクションの強制

サーバ コンソールでは、ガベージ コレクションを実行するようにサーバ コンポーネントに強制できます。

この手順は、ガベージ コレクションを実行するのに加えて、メモリ統計が含まれるファイルを作成します。 たとえば、以下のようになります。

```
Before: Memory used 35mb, allocated 97mb, max 227mb (15%) ;
After: Memory used 19mb, allocated 95mb, max 227mb (8%)
```

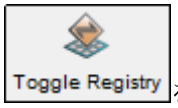
通常、この手順は、テクニカル サポートによって要求されたときのみ実行します。

ガベージ コレクションを強制する方法

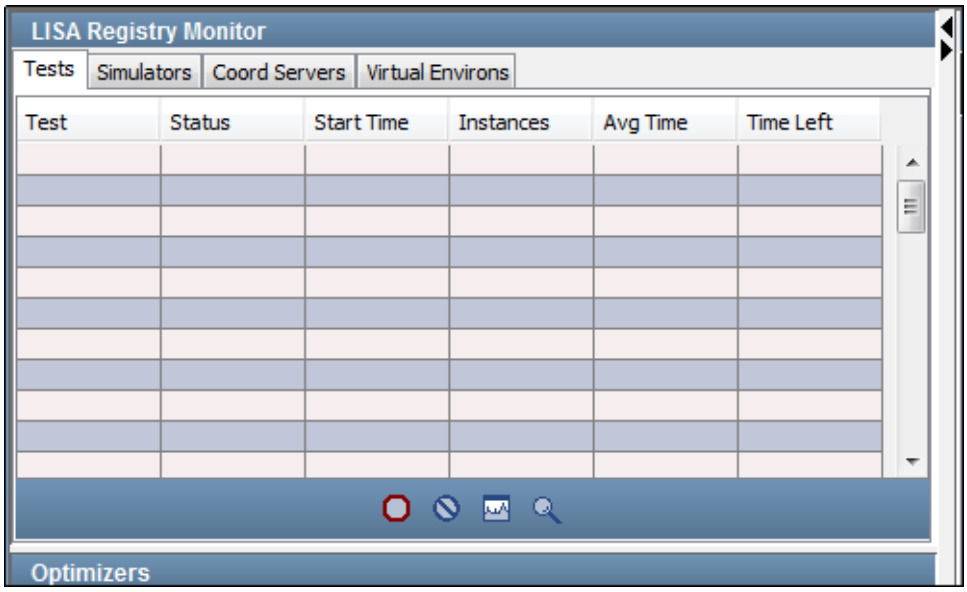
1. サーバ コンソールに移動します。
2. ネットワーク グラフ内のサーバ コンポーネントを右クリックし、［ガベージ コレクションを実行］を選択します。
3. メッセージが表示されたら、統計ファイルを保存します。
4. 統計ファイルを開き、内容を確認します。

レジストリ モニタの使用

レジストリ モニタを使用すると、テスト スイートのテスト ケース、シミュレータ、コーディネータ、および仮想環境をモニタできます。

レジストリ モニタを開くには、DevTest ワークステーション のメイン ツールバーの［レジストリ モニタの切替］ 

をクリックします。



レジストリ モニタには、以下のタブがあります。

- ［テスト］タブ (P. 174)
- ［シミュレータ］タブ (P. 174)
- ［コーディネータ サーバ］タブ (P. 175)
- ［仮想環境］タブ (P. 175)

レジストリ モニタ - [テスト]タブ

［テスト］タブには、このテスト スイートで現在実行されているすべてのテスト ケースに関する情報がリスト表示されます。

テスト ケースごとに、テスト名、ステータス、開始時刻、インスタンス、平均時間、および残り時間を表示できます。

選択したテストに対して、以下のアクションを実行できます。

- テストを停止するには、［停止］  をクリックします。
- テストを強制終了（即時停止）するには、［強制終了］  をクリックします。
- テストを最適化するには、［テストの最適化］  をクリックします。

  詳細については、「*CA Application Test の使用*」の「ロード テスト オプティマイザの使用」を参照してください。
- テストを表示するには、［テストの表示］  をクリックします。

  テスト情報は、テスト モニタに表示されます。

レジストリ モニタ - [シミュレータ]タブ

［シミュレータ］タブでは、選択したシミュレータ サーバにリアルタイムで仮想ユーザを追加できます。

このタブには、シミュレータ サーバおよび使用可能なインスタンスがリスト表示あれます。

レジストリ モニタ - [コーディネータ サーバ]タブ

［コーディネータ サーバ］タブには、実行されているコーディネータ サーバに関する情報がリスト表示されます。

選択したコーディネータ サーバに対して、以下のアクションを実行できます。

- このサービスをシャットダウンするには、［停止］  をクリックします。
- このサービスをリセット（現在のアクティビティを停止してクリア）するには、［リセット］  をクリックします。
- ステータス メッセージを表示するには、［ステータス メッセージの表示］  をクリックします。

レジストリ モニタ - [仮想環境]タブ

［仮想環境］タブには、実行されている仮想環境（ある場合）に関する情報がリスト表示されます。

エンタープライズ ダッシュボードの使用

このセクションには、以下のトピックが含まれます。

エンタープライズ ダッシュボードを開く

Web ブラウザから エンタープライズ ダッシュボードを開く方法

1. レジストリが実行されていることを確認します。
2. エンタープライズ ダッシュボードをインストールしたディレクトリに移動します。
3. **bin** ディレクトリに移動して **EnterpriseDashboard.exe** を実行するか、または [スタート] - [プログラム] メニューから [エンタープライズ ダッシュボード] を選択します。
4. Web ブラウザに「**http://localhost:1506/**」と入力します。

   エンタープライズ ダッシュボードがリモート コンピュータで実行されている場合は、**localhost** をそのコンピュータの名前または IP アドレスに置き換えます

   エンタープライズ ダッシュボードが表示されます。

エンタープライズ ダッシュボードのメイン ウィンドウ

エンタープライズ ダッシュボードのメイン ウィンドウには、実行中のレジストリに関する以下のパフォーマンス統計が表示されます。

表示名

［表示名］（設定ウィンドウでは「レジストリ名」とも呼ばれます）は、そのレジストリの設定時に割り当てた名前です。

name

レジストリの URL。

ステータス

値は［*実行中*］または［*停止*］です。

Version

そのレジストリの DevTest バージョン

コーディネータ、シミュレータ、VSE、ワークステーション、エージェント、およびラボ

レジストリに関連付けられている各リソースの数。

注: エージェントを表示するには、start broker.exe を起動する必要があります。

DevTest **コンソール** URL

DevTest コンソールのアドレス。

詳細な情報が含まれる［レジストリ サマリ］ウィンドウを表示するには、マウス カーソルをレジストリの［表示名］の上に移動します。

表示名

［表示名］（設定ウィンドウでは「レジストリ名」とも呼ばれます）は、そのレジストリの設定時に割り当てた名前です。

URL

レジストリの URL。

DevTest Solutions **のバージョン**

レジストリが動作しているコンピュータで実行されている DevTest のバージョン。

Java **バージョン**

レジストリが動作しているコンピュータで実行されている Java のバージョン。

オペレーティング システム

レジストリが動作しているコンピュータで実行されているオペレーティング システムのバージョン。

セキュリティ

［有効］または［無効］。レジストリが ACL を使用するかどうかによって異なります。

ステータス

値は［*実行中*］または［*停止*］です。

稼働時間

レジストリが起動してからの時間（日、時間、分、および秒）。

前回の開始時刻

レジストリが最後に起動した日時。

エンタープライズ ダッシュボードが使用しているデータベースに関する情報を表示するには、パネルの右上隅にあるデータベース アイコン  をクリックします。データベース URL、データベース タイプ、データベース バージョン、ドライバ名、ドライバ バージョン、およびユーザが、ポップアップ ウィンドウに表示されます。

注: 一部の情報は、7.1 よりも前のリリースでは表示されません。 以下の表は、以前のリリースで表示される機能の情報を示しています。

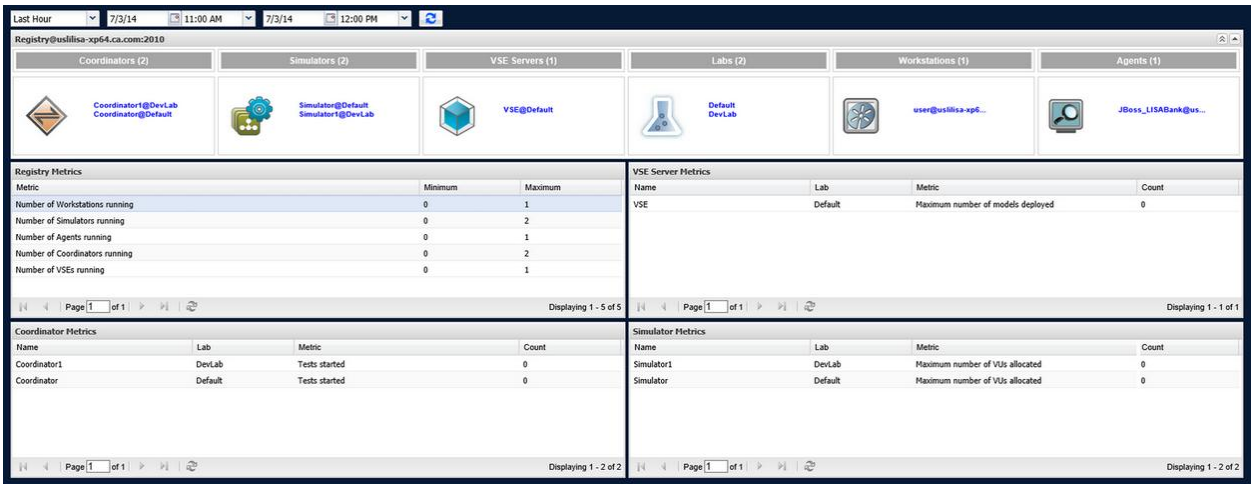
機能	7.1 よりも前のレジストリで使用可能	7.1 のレジストリで使用可能	7.5 のレジストリで新規追加
レジストリの表示名		x	
URL	x		
ステータス	x		
稼働時間		x	
セキュリティ	x		
DevTest バージョン		x	
Java バージョン		x	

機能	7.1 よりも前のレジストリで使用可能	7.1 のレジストリで使用可能	7.5 のレジストリで新規追加
稼働時間		x	
オペレーティング システム		x	
DevTest コンソール URL		x	
コーディネータ名	x		
シミュレータ名	x		
VSE サーバ名	x		
ラボ名	x		
メトリック			x
履歴データ			x

エンタープライズ ダッシュボードの[レジストリ詳細]ウィンドウ

エンタープライズ ダッシュボードのレジストリ詳細ウィンドウには、レジストリに関する詳細が表示されます。

注: このウィンドウには、バージョン 7.5 以降を実行するレジストリに対するデータだけが表示されます。



ウィンドウの上部には、そのレジストリ用のすべてのアクティブなコーディネータ、シミュレータ、VSE サーバ、ラボ、ワークステーション、およびエージェントがリスト表示されます。

メトリックの表示に使用するタイムフレームを指定するには、パネルの左上にあるドロップダウン フィールドを使用します。 最初のドロップダウン フィールドには、標準のタイムフレームのリストが表示されます。 より詳細なタイムフレームを使用するには、その他のドロップダウン フィールドで開始日時および終了日時を指定します。 ［リフレッシュ］アイコンをクリックして、表示をリフレッシュします。

エンタープライズ ダッシュボードがしばらくの間アイドル状態になっていた場合は、［リフレッシュ］アイコン  をクリックして各パネルのデータをリフレッシュしてください。

ウィンドウの下部には、レジストリ、VSE サーバ、コーディネータ、およびシミュレータのメトリックが表示されます。

パネルの［レジストリ メトリック］領域には、指定された期間中に実行中であったワークステーション、シミュレータ、コーディネータ、および VSE の最大数および最小数が表示されます。

パネルの［VSE サーバ メトリック］領域には、指定された期間中のトランザクションの数、およびアクティブな各 VSE に対して展開されたラボの数が表示されます。

パネルの［コーディネータ メトリック］領域には、指定された期間中に、各コーディネータに対して開始されたテストの数が表示されます。

テスト スイートを実行し、エンタープライズ ダッシュボードの［開始テスト］のカウントが想定どおりではない場合、DevTest ワークステーションの［スイート結果］セクションを参照します。失敗したテストのそれぞれの結果を確認します。テストがステージングに失敗した場合、そのテストは［開始テスト］のカウントには含まれません。

パネルの［シミュレータ メトリック］領域には、指定された期間中にアクティブであった仮想ユーザの最大数が表示されます。

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。

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

バージョン 8.0 以降の DevTest のインストール セットアップ ウィザードは新しいレジストリを設定します。 エンタープライズ ダッシュボードやレジストリを開始するとき、レジストリは自動的にアクティブ化されます。ただし、以降に、以下のいずれかを変更した場合、再アクティブ化が必要です。

- レジストリがインストールされているサーバのホスト名
- レジストリ名
- レジストリ ポート
- エンタープライズ ダッシュボードの URL

次の手順に従ってください：

1. エンタープライズ ダッシュボードのホスト名またはポートが変更された場合は、以下の手順に従います。
   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コンピュータにログオンします。
   b. LISA_HOME に移動し、local.properties を開きます。
   c. エンタープライズ ダッシュボードのホスト名またはポートが変更された場合は、以下の行を更新します。

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

   d. ファイルを保存します。
   e. エンタープライズ ダッシュボードを再起動します。
2. エンタープライズ ダッシュボードが実行されていることを確認します。
3. レジストリがインストールされているサーバのホスト名が変更された場合は、レジストリを再起動して再アクティブ化します。
4. レジストリのレジストリ名またはレジストリ ポートが変更された場合は、以下の手順に従います。
   a. レジストリの変更があったコンピュータにログオンします。
   b. コマンド プロンプトまたはターミナル ウィンドウを開いて、LISA_HOME に移動します。
   c. 新しい名前またはポートでレジストリを開始します。 以下に例を示します。

```
./bin/Registry.exe –n “tcp://localhost:2093/MyRegistry”
```

5. レジストリの再設定を確認します。

レジストリのメンテナンス

レジストリのメンテナンスには、以下の手順が含まれます。

- DevTest サーバ のレジストリを追加し、レジストリを検証します。

  注: DevTest 7.5 より前のリリースからのレジストリの追加の詳細については、「Configure Existing Registries」を参照してください。 8.0 以降のバージョンでは、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。
- 既存のレジストリを更新し、レジストリを検証します。
- サービス内に存在しなくなったレジストリを DevTest サーバ から削除します。
- エンタープライズ ダッシュボードで表示される各レジストリを、定期的に検証します。

レジストリを追加する方法（DevTest 7.5.x）

1. 新しくインストールされた DevTest サーバ の LISA_HOME ディレクトリに移動します。
2. _local.properties をコピーし、コピーの名前を local.properties に変更します。
3. local.properties を開いて編集します。Section 1 - Enterprise Dashboard を見つけます。
4. 以下の行のコメントを外し、somehost を localhost または有効なホスト名に書き換えます。

   ```
   lisa.enterprisedashboard.service.url=tcp://localhost:2003/EnterpriseDashboard
   ```
5. local.properties を保存して終了します。

レジストリを削除する方法

1. エンタープライズ ダッシュボードを開きます (P. 176)。
2. ［オプション］メニューで、［設定］を選択します。
3. ［レジストリ］列のレジストリのリストから、削除するレジストリをクリックします。
4. ［削除］をクリックします。

注: バージョン 7.5 以降で実行されているレジストリを削除する場合、そのレジストリ用の **local.properties** ファイル内の **lisa.enterprise.dashboard.url** プロパティも更新する必要があります。

注: ［レジストリ設定］ページでレジストリを削除すると、ダッシュボードからそのレジストリが削除されます。 ただし、そのレジストリの履歴データは、データベースから自動的にはパージされません。 履歴データのパージの詳細については、「ダッシュボード データのパージ (P. 189)」を参照してください。

レジストリを更新する方法

1. エンタープライズ ダッシュボードを開きます (P. 176)。
2. ［オプション］メニューで、［設定］を選択します。
3. ［レジストリ］列のレジストリのリストから、更新するレジストリをクリックします。
4. ［レジストリ名］および［レジストリの表示］フィールドを編集します。
5. ［保存］をクリックします。

レジストリを検証する方法

1. エンタープライズ ダッシュボードを開きます (P. 176)。
2. ［オプション］メニューで、［設定］を選択します。
3. ［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。
4. ［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。
5. ［OK］をクリックします。

ダッシュボード データのエクスポート

エンタープライズ ダッシュボードでは、現在および履歴のダッシュボード データを Excel 形式でエクスポートすることができます。メイン ウィンドウ (P. 177)または［レジストリ詳細］ウィンドウのいずれかからデータをエクスポートできます。

以下のメトリックがエクスポートされます。

- **レジストリ メトリック**
  - 実行中ワークステーションの数（最小/最大）
  - 実行中シミュレータの数（最小/最大）
  - 実行中エージェントの数（最小/最大）
  - 実行中コーディネータの数（最小/最大）
  - 実行中 VSE（仮想サービス環境）の数（最小/最大）
- **VSE メトリック**
  - 展開されるモデルの最大数
- **コーディネータ メトリック**
  - Number of Tests Started （開始テスト数）
- **シミュレータ メトリック**
  - 割り当てられる VU の最大数

次の手順に従ってください:

1. ウィンドウの右上隅の［オプション］をクリックし、［Excel にエクスポート］を選択します。

   ［レジストリ詳細］ウィンドウからエクスポートする場合は、手順 4 に進みます。

   メイン ウィンドウからエクスポートする場合は、［レジストリ データのエクスポート］ウィンドウが表示されます。

2. 以下のフィールドに入力します。

   ライブ データのみのエクスポート

   このチェック ボックスをオンにすると、ライブ データだけに限定してエクスポートされます。

このチェック ボックスをオフにすると、履歴データがエクスポートされます。

レジストリに ping

このチェック ボックスをオンにすると、各レジストリに ping を実行してレジストリのステータスを判定し、エクスポートされたファイルにそのステータスを出力します。

このチェック ボックスをオフにすると、エクスポートされたファイル内の各レジストリのステータスは「不明」になります。

注: このオプションは履歴データだけに適用されます。 ライブデータのエクスポートでは、各レジストリのステータスは自動的に判定されます。

3. ［OK］をクリックします。

エクスポートされた Excel ファイルを保存するか、または開くかのいずれかを選択するように求められます。

4. 以下の 1 つをクリックします。
   - エクスポートされたファイルを、指定したディレクトリに保存するには、［保存］をクリックします。
   - エクスポートされたファイルを表示するには、［開く］をクリックします。

使用状況監査データのエクスポート

DevTest Solutions 使用状況監査レポート は、お客様のライセンス契約書の遵守の詳細を提供します。この情報は、以下のユーザ タイプごとの最大同時使用状況に基づいています。

- PF パワー ユーザ
- SV パワー ユーザ
- テスト パワー ユーザ
- ランタイム ユーザ

レポートには管理者ユーザを含めることができます。管理者ユーザは、PF パワー ユーザと SV パワー ユーザの結合として サービス コンソールの［ロール］グリッドに表示されます。 ライセンスはユーザ タイプのこの結合を無視します。

使用状況監査レポートは、オンデマンドで生成できます。 デフォルトでは、3 か月ごとに生成されます。

次の手順に従ってください:

1. 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。

   `http://hostname:1506`

2. ［オプション］ドロップダウン リストをクリックして、［Export Usage Audit Data］を選択します。

   ［Export Usage Audit Data］ダイアログ ボックスが表示されます。

3. レポートの日付範囲の開始日と終了日を選択するために、カレンダ アイコンをクリックして、［OK］をクリックします。

   指定した日付範囲から生成されたレポートが、ウィンドウの左下にダウンロードされます。

4. 作成したレポートが含まれる Excel ブックを開くには、［DevTestSolutionsUsageAuditReport.xlsx］をクリックします。

   各タブの詳細については、「DevTest Solutions 使用状況監査レポート (P. 69)」を参照してください。

ダッシュボード データのパージ

エンタープライズ ダッシュボードでは、指定した日時より古い履歴イベント ログおよびメトリックをパージできます。また、［レジストリ設定］ページで削除したレジストリをデータベースからパージできます。 詳細については、「レジストリの設定」を参照してください。

次の手順に従ってください:

1. ウィンドウの右上隅にある［オプション］をクリックし、［データのパージ］を選択します。

   ［データのパージ］ダイアログ ボックスが表示されます。

2. パージするデータの日時を選択します。

   パージ機能は、指定された日時より古いすべてのイベント ログおよびメトリックをデータベースから永久に削除します。

   デフォルト値では、現在の日時の 100 日前より古いイベント ログおよびメトリックが削除されます。

3. ［レジストリ設定］ページで削除したすべてのレジストリをデータベースから永久に削除するには、［**削除対象としてマークされたレジストリをパージ**］チェック ボックスをオンにします。

   注: ［レジストリ設定］ページでレジストリを削除すると、ダッシュボードからそのレジストリが削除されます。 ただし、そのレジストリの履歴データは、データベースから自動的にはパージされません。 このチェック ボックスをオンにすると、削除されたレジストリがデータベースから永久に削除され、そのレジストリに関連付けられているイベント ログ、メトリック、およびコンポーネントもすべて永久に削除されます。

4. ［OK］をクリックします。

   確認ダイアログ ボックスが表示されます。

5. ［はい］をクリックします。

   選択したデータがエンタープライズ ダッシュボードからパージされます。

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。