ORACLE

Oracle Forms Standalone Launcher（FSAL）

FSALでのアプリケーションの実行

2023年2月、バージョン12.2.1.19 - Rev 0

公開

本書の目的

本書では、Oracle Forms Standalone Launcher（FSAL）で想定されている使用方法について概説しています。本書に記載されているセキュリティ情報と推奨事項は、例示のみを目的としています。説明されている概念を理解し、本書の内容を読んで行われるあらゆる実装を適切にテストし、セキュリティを確保するのは読者の責任です。オラクル製品やそこで使用されるテクノロジーを誤用すると、ご使用のシステムやアプリケーションのセキュリティが低下する恐れがあります。本書に記載されている資料を必ず参照して、製品および関連するすべてのテクノロジーを適切に使用してください。

目次

はじめに

Webブラウザ・ソフトウェアがプラグインのサポートから距離を置くようになるにつれて、Oracle Formsのようなテクノロジーを使用して開発されたアプリケーションを実行するための、ブラウザ以外の方法が必要になってきました。Oracle Formsのバージョン12.2.1.0から、そのような代替方法の1つが提供されました。

Forms Standalone Launcher（FSAL）は、エンドユーザーがOracle Forms 12.2.1以降のアプリケーションを実行するための代替方法です。FSALはブラウザ不要のクライアント/サーバーのようなエクスペリエンスを提供し、ユーザーはこれを使用してアプリケーションにアクセスし、やり取りすることができます。FSALではブラウザを使用しないため、Java Deploymentテクノロジー（Plug-in、Web Startなど）に依存せず、ブラウザから独立しています。
ただし、エンドユーザー側のマシンではJavaランタイムが必要です。

本書には、FSALの使用に関する情報と、アプリケーションをできる限り安全に実行するためのいくつかのヒントが掲載されています。

本書ではOracle Forms 12.2.1.19.0の使用を前提としています。本書で説明されている機能の一部は、以前のバージョンのForms 12cファミリーでは使用できない場合がありますが、多くのコンセプトは12.2.1のすべてのリリースに当てはまります。本書でJavaについて言及されている場合、特に記載のない限り、Javaバージョン8u341以降関連であることを前提としています。

Forms Standalone Launcherについて

Forms Standalone Launcher（FSAL）は、複数のクライアント/サーバーのコンセプトを、いくつかの最新技術を採用した上で模倣しています。以前のOracle Formsクライアント/サーバーでは、エンドユーザーのマシンに、そのマシンで実行する予定のアプリケーションだけではなく、Oracle Formsランタイム・ソフトウェアをインストールする必要がありました。これは複雑なプロセスであり、相当量のディスク領域を消費することが珍しくありませんでした。ユーザーのマシンにソフトウェアとアプリケーションをインストールすることは、セキュリティ上の問題とみなされることもありました。ユーザーがランタイム・ソフトウェア、付属するユーティリティ、コンパイル済みアプリケーション・モジュールに直接アクセスできるようになるからです。

以前は、エンドユーザーのマシンを管理しやすくするために、ソフトウェアをローカルではインストールせず、ネットワーク全体でリモート・インストールとリモート・アクセスを利用している企業もありました。リモート・アクセスやリモート・デスクトップ共有は多くの場合に有効ですが、問題もあり、オラクルによるサポート構成に含まれていませんでした（現在も含まれていません）。

FSALは、クライアント/サーバーとWebの両方のコンセプトから、ベストなものを採用しています。FSALを使用すると、見た目と機能はネイティブ・インストールされたアプリケーションのものとなり、クライアント/サーバーのFormsと非常によく似ています。アプリケーションはブラウザでフォームを実行する場合とは異なり、独立した親ウィンドウで実行されます。その結果、間違って「戻る」や「進む」ボタンを押したり、「ブックマーク」をクリックしたりして、実行中のフォームから離れてしまうというリスクがなくなります。FSALでは、Webにデプロイされたすべての Formsバージョンと同じように、アプリケーションは中央のアプリケーション・サーバー（WebLogic Serverなど）でホストされます。つまり、Formsのアプリケーション・モジュールは、リモート中間層のサーバーに安全に格納されます。ユーザーは、リモート・サーバー上のFormsのモジュール（FMX、MMX、PLXなど）には直接アクセスできませんが、以前にブラウザで実行していたときに使われていた一般的なURLを使用して、そのようなアプリケーションを実行できます。

FSALを使用すると、ブラウザでフォームを実行する場合と同じ、すべての機能がサポートされます（イベント駆動型のシングル・サインオフを除く）。シングル・サインオンは、12.2.1.4.0以降でサポートされています。FSALを使用して起動したアプリケーションでのJavaScript統合については、12.2.1.3.0以上のアドオン（WJSI）とサード・パーティ・ライブラリ（Eclipse/Jetty）を使用してサポートを追加/有効化できます。12.2.1.19.0以降のFSALは、カスタム・プロトコル・ハンドラを使用してWebページから起動できます。

要件

Forms Standalone Launcher

前述のように、FSALの機能は、Forms 12cファミリー以降のみで使用できます。構成の要件として、小さなJava JARファイル（frmsal.jar）をエンドユーザーのマシンに転送して格納する必要があります。このファイル/ユーティリティは、何らかの都合の良い方法（Webからのダウンロード、電子メール、FTP、カスタム・インストーラなど）を使用して、エンドユーザーのマシンに転送できます。該当のディレクトリとユーティリティにユーザーがアクセスできるのであれば、ユーザーのマシンのどこにユーティリティを格納してもかまいません。ユーザーのホーム・ディレクトリに格納することが推奨されていますが、必須ではありません。

このfrmsal.jarファイルが、“Forms Standalone Launcher”（FSAL）です。frmsal.jarはバージョン固有であるため、あるインストールのfrmsal.jarを別のインストールで使うことはできません。このファイルがサーバーおよびそのFormsのバージョンと正しく合致しているかを確かめるために、チェックサムも使用されます。このチェックサムは、ランチャ（frmsal.jar）が意図せずに、もしくは悪意のある目的で置き換えられていないかどうかを確かめるのにも役立ちます。そのため、Microsoft Windowsサーバーからダウンロードしたfrmsal.jarを、UNIX/Linuxサーバーでのアプリケーション実行に使用することはできず、その逆もまた同様です。このランチャはクライアントのプラットフォーム固有ではありません。Microsoft Windows、Unix/Linux、Apple Macのクライアント・マシンのいずれでも使用可能であり、基本的に、Formsで動作保証されているOracle Javaバージョンの実行をサポートする任意のプラットフォームでFSALを使用できます。

すべてのインストールに、使用法/構文のWebページが含まれています。ここではランチャの使い方が説明されており、frmsal.jarファイルのダウンロード・リンクも掲載されています。このページには、次のようなURLでアクセスできます。

https://yourserver:port/forms/html/fsal.htm

Java

FSALはJavaアプリケーションであり、ユーザーのマシンにはOracle Javaが必要です。ただし、特定の種類のJava（JRE - Plug-inなど）が必要となるブラウザに埋め込まれたFormsアプリケーションの実行とは異なり、FSALでは、Javaアプリケーションの実行をサポートし、ご使用のバージョンのOracle Formsで動作保証されたOracle Javaディストリビューションであれば、どれでも使用できます。ユーザーのプラットフォームと使用するOracle Javaバージョンに応じて、使用できるディストリビューションの種類は複数あります。ほとんどの一般的なエンドユーザー・プラットフォームで、以下のディストリビューションを使用して、FSALを実行できます。Java 8ファミリーでは、32ビットと64ビットの両方のディストリビューションで**JRE**、**JDK**、**Server JRE**が提供されています。Java 11以降で提供されているのは、64ビットの**JDK**ディストリビューションのみですが、これをFSALで使用できます。アプリケーションの必要性に応じて、どのディストリビューションやバージョンを選択するかは異なります。選択したOracle Javaインストールの内容を慎重に調査し、本番環境に移行する前に、ご使用のアプリケーションを徹底的にテストすることをお勧めします。各Javaディストリビューションの概要は以下のとおりです。

Java Runtime Environment - JRE

Java 8のみ。JREでは、標準的なエンドユーザーが必要とするコンポーネントのほとんどがインストールされます。ローカルのJavaアプリケーションを実行するのに必要なすべてのものと、Javaデプロイメントのコンポーネント（Java Web StartとJava Plug-in）が含まれます。Javaデプロイメントのコンポーネントは、Java 8以降のメジャー・リリースには含まれていません。

Java Development Kit - JDK

JDKはすべてのバージョンで提供されていますが、たいていJava開発者向けであり、一般的なユーザーが必要とするよりもずっと多くの機能を備えています。JDKにはJREのすべてのコンポーネントとJavaデプロイメント・コンポーネントに加えて、Javaアプリケーションの開発、デバッグ、監視、保守用のツールが含まれます。

Server JRE

Java 8のみ。Server JREの主な目的は、サーバー上でJavaアプリケーションをデプロイすることです。JVM監視ツールと、サーバー・アプリケーションで通常必要となるツールが含まれていますが、ブラウザ統合に必要なJavaデプロイメント・コンポーネント（Java Plug-inなど）、Java Web Start、自動更新、インストーラは含まれていません。このディストリビューションはzipファイルで配信されるため、手動で解凍する必要があります。このディストリビューションではソフトウェアがインストールされず、JREやJDKディストリビューションよりもかなり軽量であるため、カスタマイズされた起動スクリプトなどでFSALを使用する場合に理想的な選択肢です。

カスタムJRE

Oracle Javaバージョン11以降では、カスタムのJREディストリビューションを作成できます。このようなディストリビューションの作成方法については、本書では説明しません。カスタムJREを作成するには、**jdeps**ユーティリティ（必要なJavaモジュールを特定するために使用）と**jlink**ユーティリティ（カスタムJREを作成するために使用）が必要になります。必要な場合は、**jpackage**ユーティリティを使用すると、カスタムJREをインストール可能なディストリビューションとしてパッケージ化することができます。これらの新しいユーティリティの使用法について詳しくは、Javaのドキュメントを参照してください。

https://docs.oracle.com/en/java/javase/17/docs/specs/man/index.html

Fusion Middleware Certification Guideを参照して、対象のJavaバージョンが、ご使用のFormsバージョンで動作保証されていることを確認してください。

FSALでのアプリケーションの実行

基本事項

前述のとおり、FSALの目的は、Webブラウザを使用せずにFormsアプリケーションを起動し、それらのアプリケーションを実行および操作することです。技術的には可能ですが、このユーティリティをユーザーとのやり取り以外の目的（無人の負荷テストなど）で使用することは想定も推奨もされていません。ユーザーのマシン上で、多すぎる（1人のユーザーが合理的に必要とする数を超える）同時セッションまたは連続セッションを起動しようとすると、予測不可能な望ましくない動作の原因となる可能性があります。

FSALはブラウザと同じように、実行するアプリケーションの場所を把握していなければなりません。ブラウザでは、他のWebページ、さらにはFormsアプリケーションを起動するのに、ハイパーリンクが使われることがあります。FSALはブラウザからは完全に独立しているため、ユーザーのマシンでいくつかの構成を変更しない限り、Webページを使用してFSALがホストするFormsアプリケーションを起動することはできません。Forms 12.2.1.19以降では、カスタム・プロトコル・ハンドラ（fsal://およびfsals://）がサポートされているため、これを使用してWebページからFSALを立ち上げることができます。その際、リクエストを処理するために、完全なFormsアプリケーションURLを把握している必要があります。現時点では、URLリダイレクトまたはリライトの使用はサポートされていませんが、サーバーの構成によっては技術的に可能である場合があります。FSALは、Forms環境を示す完全修飾URLを受け取る必要があります。アプリケーションを簡単に、そしてエラーをできるだけ避けて起動するために、ハイパーリンクの代わりにデスクトップ・ショートカットやスクリプト/バッチ・ファイルを使用することができます。

上記の要件が満たされれば、FSALでのアプリケーションの実行は容易です。FSALでアプリケーションを起動するには、以下の手順に従ってください。

1. シェル（Microsoft WindowsのDOSなど）を開き、動作保証された必要なOracle Javaのバージョンがあることを確認します。

   ```
   java –version
   ```

   適切なOracle Javaのバージョンが出力されるはずです。出力されない場合は、システムのPATHが適切に設定されていない可能性があります。必要な修正を行ってから処理を続行します。

2. ディレクトリをユーザーのホーム・ディレクトリに変更して（frmsal.jarがユーザーのホーム・ディレクトリに格納されている場合）、次のコマンドを実行します。下の例のサーバー名およびポート番号を、適切な値で置き換えます。サーバーでSSLが有効化されておらず、代わりにHTTPを使用している場合は、URLエントリからプロトコル（“http://”）を省略できます。

   ```
   java -jar frmsal.jar -url "https://server:port/forms/frmservlet?config=standaloneapp"
   ```

   この例では、構成セクションに関連付けられている、[standaloneapp]というアプリケーションが起動します。FormsのWeb構成ページで[standaloneapp]の例に示されているエントリが含まれていれば、どの構成セクションを使用してもかまいません。

3. 通常はJava Consoleに表示されるOracle Formsの出力が、アプリケーションの起動に使用されるシェルに表示されます。“java”ではなく、“javaw”や同様のコマンドが使用される場合、コンソールは表示されない可能性があります。“java”コマンドを使用している場合、アプリケーションを起動したシェルを閉じるとアプリケーションが終了します。そのため、セッションが終わるまではシェルを開いておく必要があります。この動作は、さまざまなシェル・コマンドと関連するスイッチを使用して、アプリケーションのニーズに合うように変更できます。ユーザーのプラットフォームでコマンド・シェルを使用するための情報については、オペレーティング・システムのドキュメントを参照してください。

上記の基本的な情報を使用して、デスクトップ・ショートカット、バッチ・スクリプト・ファイル、またはカスタム実行可能ファイルを作成すると、アプリケーションをより簡単かつシームレスに起動できます。

コマンドライン引数の一覧については、本書のFSALリファレンスのセクションを参照してください。この情報は、インストールに含まれるFSAL使用法ページにも表示されます。使用法ページにアクセスするには、以下のURLをブラウザに入力します。

http://server:port/forms/html/fsal.htm

ファイル・キャッシュ

Java Plug-inやブラウザと同様に、FSALは、次回アプリケーションが起動されたときに再利用できるように、ファイルをキャッシュ（ローカルに保存）しようとします。これにより、起動時のパフォーマンスが向上します。Unix/LinuxシェルまたはWindows DOSシェルからFSALを起動すると、キャッシュされたファイルが保管されるディレクトリが、ロード・プロセス中のシェル出力に表示されます。また、ファイルがサーバーからダウンロードされているのか、既存のキャッシュから再利用されているのかが表示されます。ファイルがサーバーからダウンロードされる場合、以下のようなテキストが表示されます。

Inspecting archive files in cache directory
C:\Users\jdoe\AppData\Local\Temp\frmsal\<*server name*>\12.2.1.19

Downloading archive file frmall.jar to cache subdirectory 8ymuqdqvdfe13a0d5vvema0dh

以前にキャッシュされたファイルが使用される場合、以下のようなテキストが表示されます。

Inspecting archive files in cache directory
C:\Users\jdoe\AppData\Local\Temp\frmsal\<*server name*>\12.2.1.19

Using cached archive file frmall.jar from cache subdirectory 8ymuqdqvdfe13a0d5vvema0dh

キャッシュ・ファイルの場所を指定するには、アプリケーションの起動時にtmpdirの場所を変更します。次のMicrosoft Windowsユーザーの例を考えてみてください。ここでは、最初のレベルのディレクトリとして、Windowsシステム変数%USERNAME%の値が使用されています。

java -Djava.io.tmpdir=C:/%USERNAME%/fsal -jar frmsal.jar -url
“https://server:port/forms/frmservlet?config=standaloneapp”

この方法は、ユーザーが共有のUnix/Linuxプラットフォームを使用する場合に役立つ場合があります。ダウンロードされるキャッシュ・ファイルは、アプリケーションを実行した最初のユーザーに所有されるため、その後のユーザーには、古いキャッシュ・ファイルを新しいファイルで上書きするための十分な権限がない場合があります。ユーザーごとに個別の場所を作成することで、この問題を回避できます。

java -Djava.io.tmpdir=/u01/$user/fsal -jar frmsal.jar -url
"https://server:port/forms/frmservlet?config=standaloneapp"

アプリケーションによっては、キャッシュ・ファイルの使用は望ましくなく、常にサーバーからダウンロードするケースがあります。これは、技術的な問題をトラブルシューティングする際にも当てはまります。ファイルのキャッシュを無効にするには、Fusion Middleware Control（FMC）を使用してFSALのテンプレート・ファイル（アプリケーションによってbasesaa.txtまたはwebutilsaa.txt）を編集します。次の行を追加します。

ignoreSaaCache=%ignoreSaaCache%

FormsのWeb構成ファイル（formsweb.cfg）の**[standaloneapp]**構成か、アプリケーション実行に使用している構成に、新規パラメータ**ignoreSaaCache**を追加します。値は**TRUE**に設定します。これにより、キャッシュ・ファイルが無視され、アプリケーションが起動されるたびにサーバーからファイルがダウンロードされるようになります。

Java 11+を使用したオーディオ対応アプリケーションの実行

Oracle Java 11以降の"Long Term Support"（LTS）リリースを使用してFSALを実行できます。ただし、Java 11以降にはJavaFXが含まれていません。JavaFXは、Formsのオーディオ機能を正しく使用するためには不可欠です。アプリケーションでFormsのオーディオ機能を使用しない場合、以下の手順は省略できます。JavaFXを有効にする場合は、ユーザーのマシンで以下の手順を実行する必要があります。

1. ユーザーのマシンで、*Gluon*（非オラクル関連会社）からJavaFX SDKをダウンロードし、ユーザーがランタイム（読取り、実行）権限を持つディレクトリに解凍します。FSAL実行に使用するJavaバージョンともっとも近いバージョンをダウンロードします。現在、JavaFXはオープンソース・プロジェクトになっていますが、ダウンロードして使用する前に、利用規約とサポート・オプションをよく読んでください。

   https://gluonhq.com/products/javafx

   Oracle Supportは、8よりも新しいバージョンのJavaFXをサポートしていません。8よりも新しいバージョンは、Gluonまたは以下のオープンソースコミュニティ（あるいはその両方）によってサポートされています。

   https://openjfx.io

   https://github.com/openjdk/jfx

2. 標準的なFSAL起動コマンドにJFXを含めるように変更します。次に例を示します。

   ```
   java --module-path C:/javafx-sdk-17.0.4/lib --add-modules=javafx.media,javafx.swing -jar
   frmsal.jar -url "https://server:port/forms/frmservlet?config=standaloneapp"
   ```

Java Scriptの統合とFSALアプリケーションの使用

Forms Standalone Launcherとそれによって実行されるアプリケーションは、Webブラウザまたはブラウザ・アプリケーションとは無関係ですが、FSALアプリケーションからWebページと通信することは可能です。詳しくは、Forms 12cの新機能に関するガイドまたはOracle Formsの使用に関するガイドを参照してください。

プロキシ・サーバー経由の実行

多くの場合、ユーザーは企業ネットワークの内部でFormsアプリケーションにアクセスします。場合によっては、ユーザーのマシンで社内のコンテンツと社外のコンテンツの両方にアクセスするために、適切なプロキシ構成が必要になります。FSALを使用する場合、ブラウザまたはシステム・レベルの設定は、アプリケーションを起動するシェルには把握できない可能性があります。そのため、FSALの実行時にそのような設定を含めることが必要になる場合があります。FSALはJavaアプリケーションなので、必要なプロキシの設定を認識するのはJava（java.exeなど）の役割です。これらの設定をJavaに通知する方法は複数あります。

システムのプロキシ設定を使用するには、以下の手順を実行します。

```
java -Djava.net.useSystemProxies=true –jar frmsal.jar -url
"https://server:port/forms/frmservlet?config=standaloneapp"
```

このJavaオプション（-Djava.net.useSystemProxies=true）を使用すると、システム・レベルで指定されているプロキシ設定を使用して、Javaが呼び出しを実行します。自動構成スクリプト（wpad.datなど）を使用するようシステムが設定されている場合、この方法はサポートされない可能性があることに注意してください。詳しくは、Javaの公式ドキュメントを参照してください。

もう1つの方法として、プロキシ設定を明示的に含めることができます。以下に2つの例を示します。

java -Dhttps.proxyHost=<*proxyserver*> -Dhttps.proxyPort=<*proxyserver port number*> -Dhttps.nonProxyHosts="localhost|example.com" –jar frmsal.jar -url "https://*server:port*/forms/frmservlet?config=standaloneapp"

java -Dhttp.proxyHost=<*proxyserver*> -Dhttp.proxyPort=<*proxyserver port number*> -Dhttp.nonProxyHosts="localhost|example.com" –jar frmsal.jar -url http://*server:port*/forms/frmservlet?config=standaloneapp

参考： https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html

カスタム・プロトコル・ハンドラによるFSALの起動

カスタム・プロトコルは、その誕生以来ほぼずっとWebブラウザで使用されてきました。ブラウザのアドレス・バーに入力するものとして、通常、使用または表示が想定されるのは、http、https、ftp、mailtoです。しかし、Zoom会議を開くためのzoommtgや、特定のユーザー/チャンネル/メッセージに対してSlackクライアントアプリを開くためのslackといった特別なプロトコルも珍しくはありません。

Forms 12.2.1.19以降では、同じようなことをForms Standalone Launcher（FSAL）で実行できます。FSALの拡張により、**fsal**（非SSLリクエスト用）と**fsals**（SSLリクエスト用）という2つの特殊プロトコルが認識されるようになりました。カスタム・プロトコル・ハンドラを使用すると、Webページからハイパーリンクを使用して、FSALを使用したFormsアプリを起動できます。ただし、これには、カスタム・プロトコルがユーザーのマシンに登録されており、目的のWebページが正しい形式のハイパーリンクを含んでいる必要があります。

Webページでハイパーリンクを作成する場合、その他の目的で使用されるのと同じ方法でハイパーリンクを作成します。次のようなHTMLを追加することで、名前を付けたForms構成（standaloneapp）でFSALを起動するために使用できるハイパーリンクができます。言うまでもなく、適切なサーバーとポートを指定する必要があります。

```
<a href="fsal://<SERVER>:<PORT>/forms/frmservlet?config=standaloneapp">Run FSAL</a>
```

この新しいプロトコルをユーザーのマシンに登録する手順は、ユーザーが使用するオペレーティング・システムによって異なります。どのオペレーティング・システムを使用している場合でも、いくつかの方法を使用できます。以下に、ユーザーのマシンでMicrosoft Windowsが実行されている場合の一例を示します。

前提条件は次のとおりです。

- ユーザーのホーム・ディレクトリに最新のFormsのfrmsal.jarが格納されている。
- 最新のOracle Java 8（32ビット）JREがインストールされている。

以下に示すコマンドは、WindowsのDOSコマンド・プロンプトで個別実行することも、スクリプトによって実行することもできます。どの方法であっても、Windowsレジストリに書込みを行うため、ユーザーに管理者権限が付与されている必要があります。

警告：

***Windows*レジストリの変更はリスクを伴う処理であり、正しく実行しないと回復不能な破損を引き起こすおそれがあります。コマンドを実行する前に、必ずレジストリのバックアップとシステムのリストア・ポイントを作成してください。*Windows*レジストリの編集に不慣れな場合はコマンドを実行しないでください。**

以下に示すコマンドでは、必ず各セクションの最後のエントリを入念に確認してください。"javaw.exe"のパスと"frmsal.jar"のパスが（ユーザーのマシンでの）状況に一致していることを確認します。Javaパスは、JREのインストール時のみに作成される特別なパスです。JDKを使用している場合は、このパスを変更する必要があります。

非SSLリクエスト用のコマンド

```
reg add HKEY_CLASSES_ROOT\fsal /t REG_SZ /d "Oracle Forms Standalone Launcher" /f
reg add HKEY_CLASSES_ROOT\fsal /v "URL Protocol" /t REG_SZ /d "" /f
reg add HKEY_CLASSES_ROOT\fsal /v "UseOriginalUrlEncoding" /t REG_DWORD /d "00000001"
reg add HKEY_CLASSES_ROOT\fsal\shell /f
reg add HKEY_CLASSES_ROOT\fsal\shell\open /f
reg add HKEY_CLASSES_ROOT\fsal\shell\open\command /t REG_SZ /d "\"C:\Program Files
(x86)\Common Files\Oracle\Java\javapath\javaw.exe\" -jar %USERPROFILE%\frmsal.jar -url
\"%1\"" /f
```

SSLリクエスト用のコマンド

```
reg add HKEY_CLASSES_ROOT\fsals /t REG_SZ /d "Oracle Forms Standalone Launcher with
SSL/TLS" /f
reg add HKEY_CLASSES_ROOT\fsals /v "URL Protocol" /t REG_SZ /d "" /f
reg add HKEY_CLASSES_ROOT\fsal /v "UseOriginalUrlEncoding" /t REG_DWORD /d "00000001"
reg add HKEY_CLASSES_ROOT\fsals\shell /f
reg add HKEY_CLASSES_ROOT\fsals\shell\open /f
reg add HKEY_CLASSES_ROOT\fsals\shell\open\command /t REG_SZ /d "\"C:\Program Files
(x86)\Common Files\Oracle\Java\javapath\javaw.exe\" -jar %USERPROFILE%\frmsal.jar -url
\"%1\"" /f
```

上のコマンドの実行が正常に完了したら、http://やhttps://の代わりに、fsal://またはfsals://プロトコルをブラウザで使用できます。その他のオペレーティング・システムにカスタム・プロトコル・ハンドラを登録する方法については、オペレーティング・システムのドキュメントを参照するか、ベンダーにサポートを依頼してください。

セキュリティのヒント

アプリケーション、アプリケーションで送受信されるデータ、アプリケーションがホストされているネットワークで最高水準のセキュリティを確保することは、いかなるアプリケーションのデプロイメントにおいても、最重要事項であると考えるべきです。セキュリティのレイヤーが弱くなっていると、機密データの漏洩や、システムへの悪意ある攻撃の原因となる可能性があります。アプリケーション開発者、管理者、その他のすべての*IT*担当者には、アプリケーション、データ、ホスティング・システムを保護するために、適切なセキュリティ対策を施す責任があります。

このセクションでは、*FSAL*の使用に関連したセキュリティの強化に役立つ、いくつかの方法を掲載しています。これらの提案は、例としての使用にとどめることが重要です。コンセプトを理解し、十分に理解していない点について調査するのは読者の責任です。どのようなセキュリティ構成であっても、実装が不適切だと、システムに危険が及ぶ可能性があります。そのため、実施しようとする変更が適切であると仮定する前に、慎重に確認し、テストしてください。ここに掲載されている提案は、*FSAL*または*Oracle Forms*全般のみに当てはまるものではなく、その多くは使用されているテクノロジーに関連する標準的な提案事項です。そのため、ほかにもさまざまな追加情報を入手できます。ご使用のアプリケーション、そのデータ、その環境を保護するにあたり、本書のみを情報源として使用することはしないでください。

本書の最後にある参考資料のセクションも確認してください。

Secure Socket Layer

本書では、“キーストア”という用語と“トラストストア”という用語を区別せずに使用しています。

Secure Socket Layer（SSL）とTransport Layer Security（TLS）は、ネットワーク・トラフィックの送信元と送信先との間で、通信を暗号化するためのプロトコルです。これらのプロトコルは、信頼済みの接続を作成することで機能します。この接続はほとんどの場合、公開鍵と秘密鍵の交換によって確立されます。SSL/TLSの動作の詳細については、本書では説明しません。

ネットワーク上で通信するアプリケーションを実行する際にSSL/TLSを使用することは、データ保護の観点から非常に重要です。あらゆるアプリケーションの実行時にSSL/TLSを使用することは、任意ではなく必須事項であると考えるべきです。SSL/TLSを使用するアプリケーションをFSALで実行するには、ユーザーのJavaトラストストアにSSL/TLS証明書の公開鍵が含まれている必要があります。公開鍵がトラストストアに含まれていないか、既知の認証局（DigiCert、Entrust、Comodoなど）から提供されていない場合は、手動での証明書のインポートが必要になる可能性があります。具体的には、この鍵は、FSALによってフォームの実行に使用されているJavaトラストストアにインポートしなければなりません。必要なのはたいてい“ルート”証明書だけですが、証明書のチェーンが存在する場合、複数の証明書のインポートが必要になることもあります（ルート、中間者、ユーザーなど）。SSL/TLSを使用してアプリケーションを実行しようとしており、必要な証明書が見つからない、または適切にインポートされていない場合は、以下のいずれかのようなJavaエラーが表示される可能性があります。

```
java.security.cert.CertificateException:No subject alternative names present …
```

または

```
java.security.cert.CertificateException:No name matching <server:port> found …
```

または

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: …
```

上記のようなエラーのいずれかの後に、FSALのエラーが表示されます。これは、セキュアな接続が確立できず、アプリケーションの開始と実行に必要なパラメータをダウンロードできなかったからです。

```
FRM-92490:Unable to fetch applet parameters from server.
```

FSALでのSSL/TLSの構成

Forms 12.2.1.19以降で、FSALはSSL/TLS証明書をインポートして保存するためのオプションを2つ提供しています。

デフォルト構成では、FSAL証明書インポータが使用されます。このダイアログは、サーバーから提供された証明書がユーザーのシステムで認識されなかったときに表示されます。ユーザーが証明書を受け入れてインポート権限を付与すると、新しく作成され、Forms/FSALによって使用されるJavaトラストストアに新しい証明書が挿入されます。FSALは、作成したこのトラストストアを、すべての後続リクエストに対して使用しますが、ユーザーがランタイム・スイッチcert_truststoreを“java”に設定すると、これが無効になります。

もう1つの構成オプションは、Java提供ツールを使用して、不足している証明書を手動でインポートする方法です。これは以前のバージョンで使用されていたのと同じ方法です。以前の動作に戻す方法については、本書の終わり近くに記載されたコマンドライン・オプションを参照してください。

FSAL証明書インポータの使用

FSAL証明書インポータは、ほぼ自動的に使用できます。本書で前述したとおりにFSALが使用されます。サーバーから最初のフィードバックを受け取ると、FSALはトラストストアをチェックして、提供された証明書が既知のものかどうかを判断します。証明書が既知のもので信頼できると見なされる場合、中断なくアプリケーションが実行されます。サーバーから送信された証明書が既知の証明書でない場合、それを示すダイアログが表示され、ユーザーが続行方法（証明書をインポートするか、処理を取り消す）を決定できます。ユーザーが「OK」をクリックすると次に進み、「Cancel」をクリックするとインポートなしで処理が終了します。必要に応じて、次に進む前にダイアログで「More information」リンクをクリックすると、提供された証明書に関する情報を確認できます。

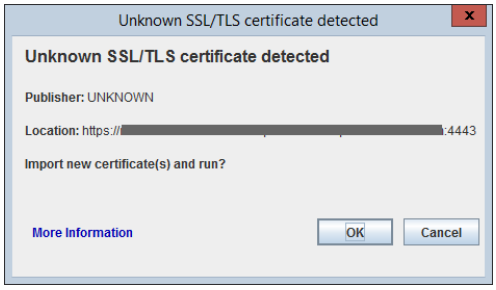

図1 - FSAL証明書インポータの例

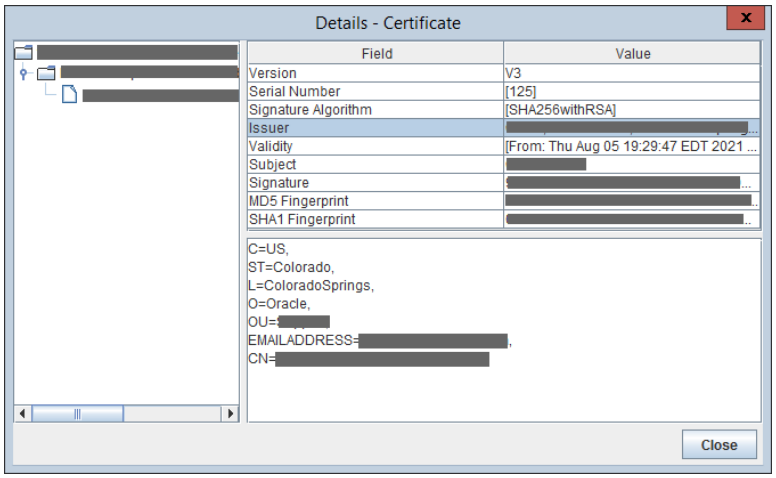

図2 - FSAL証明書インポータの詳細ビューアの例

証明書のインポートを許可してアプリケーションを実行すると、ユーザーのホーム・ディレクトリ（以下）でJavaトラストストアが作成/更新されます。

C:\Users\<USER NAME>\Oracle\forms

トラストストアのファイル名は、formstruststoreになります。生成されるトラストストアの名前変更はサポートされていません。ファイルが見つからない場合は、新しい空のファイルが作成されます。これはJavaで生成された標準トラストストア・ファイルなので、必要に応じて、Oracle Javaインストールに含まれるツール（keytool.exeなど）を使用してこのファイルを管理できます。

autoImportCertスイッチを使用すると、ユーザーによる介入なしで自動的に未知の証明書をインポートできます。この方法は推奨されておらず、おもにテストを目的に提供されています。以下に、使用例を示します。

```
java -jar frmsal.jar -url "https://server:port/forms/frmservlet?config=standaloneapp" -
autoImportCert true
```

詳細については、前述の使用法Webページに表示されるFSALコマンドライン引数一覧を参照してください。コマンドライン引数のリストは、本書の終わり近くにも記載されています。

キーストアの手動構成

FSALのSSL/TLS証明書インポータの使用が望ましくない場合は、FSALを使用する前に以下の手順を実行します。使用中の証明書とFSAL証明書インポータの間に互換性の問題が見つかった場合も、以下の手順が必要になることがあります。標準以外の証明書や自分で生成した証明書には、FSAL証明書インポータと互換性がないものがあります。

1. 既知の認証局からSSL/TLS証明書を取得し、Oracle HTTP Server（またはWebLogic Server）を構成します。これらのコンポーネントのドキュメントにある指示に従ってください。

2. 証明書の公開鍵の部分が、ユーザーのマシンで必要になります。公開鍵がない場合は、いくつかの方法で取得できます。サーバーにアクセスできるものであれば、どのマシンからでもキー・チェーンを取得できます。次に、openssl コマンドを使用して鍵を取得する方法の例を示します。これはほとんどのUnix/Linuxプラットフォームにプリインストールされていますが、Microsoft Windowsでも取得できます。

   openssl s_client -showcerts -connect *server:port* > output.txt

   上の例で、**server:port**をサーバー名とSSL/TLSのポート番号に置き換えてください。このコマンドの実行結果は、"output.txt"というファイル内に保存されます。出力には1つ以上の証明書が含まれます（証明書タイプと作成方法によって異なります）。証明書は、BEGINヘッダーとENDフッターの間に書かれているものです（ヘッダーとフッターも含みます）。各証明書を、それぞれ別のテキスト・ファイルにコピーします。以下に示されているように、BEGINおよびENDテキストとその間にある内容をそのまま含めてください。ファイルを保存する際には、**-----BEGIN CERTIFICATE-----** の上、または**-----END CERTIFICATE-----** の下に余計な行を入れないでください。ただし、これらのヘッダー/フッターは必ず含めます。

   次に例を示します。

   -----BEGIN CERTIFICATE-----

   MIIFKDCCBBCgAwIBAgIBPTANBgkqhkiG9w0BAQsFADCBrjEpMCcGA1UEAxMgTmlj

   Y2subWFuc0BvcmFjbGUuY29tMRAwDgYDVQQLEwdTdXBwb3J0MQ8wDQYDVQQKEwZP

   cmFjbGUxGTAXBgNVBAcTEENvbG9yYWRvIFNwcmluZ3MxETAPBgNVBAgTCENvbG9y

   . . .

   -----END CERTIFICATE-----

   以下のようにして、必要な証明書ファイルを直接生成することもできます。

   openssl s_client -showcerts -servername <*server name*> -connect <*server:port*> | openssl

   x509 -outform PEM > cert.cer

3. ユーザーのマシンのキーストアに、公開鍵をインポートします（cacertsなど）。証明書チェーンの場合は、チェーン内のすべての鍵（署名者、中間者、ルートなど）をインポートしてください。つまり、keytoolユーティリティを複数回実行しなければならない場合があるということです。証明書の公開鍵をインポートするときには、ほとんどのOracle Javaディストリビューションに付属しているJavaのkeytoolユーティリティを使用できます。

   以下に、"keytool"ユーティリティを使用して各証明書をインポートする方法の例を示します。keytoolユーティリティの使用方法について、詳しくはOracle Javaのドキュメントを参照してください。

   keytool -import -alias <server_name> -TrustStore <JAVA_HOME>/jre/lib/security/cacerts - file cert.cer

   "alias"は一意でなければなりません。つまり、チェーン内に複数の証明書がある場合、それぞれに一意のエイリアスを付ける必要があります。ただし、最初にルートをインポートし、そのエイリアスとしてサーバー名を使用することを推奨します。すでに使用されているエイリアスを使用しようとすると、エラーが発生します。エイリアスにサーバー名を使用することを推奨します。たとえば、ルート証明書を最初にインポートしてmyserverという名前を付け、2番目にはmyserver2を、3番目にはmyserver3を指定します。

4. 本書のFSALでのアプリケーションの実行のセクションで例示されているように、SSLを使用してFSALを実行します。ただし、証明書インポータの自動実行を回避するため、必ずスイッチ-cert_truststoreをjavaに設定します。

java -jar frmsal.jar -url "https://*server:port*/forms/frmservlet?config=standaloneapp" –cert_truststore java

SSL/TLSのトラブルシューティング

ヒント1：

Javaキーストアに証明書を手動で挿入した場合は、正しいキーストアが更新されたことを確認します。デフォルトでは、キーストアは、JRE_HOME/lib/securityまたはユーザーのホーム・ディレクトリにあります。デフォルトのファイル名（FSAL証明書インポータを使用しない場合）は、**cacerts**です。ファイルの更新日時が、importコマンドを実行した日時と同じであることを確認してください。FSAL証明書インポータを使用する場合、トラストストア名は**formstruststore**になります。

ヒント2：

チェーン内のすべての証明書がインポートされていることを確認します。多くの場合、必要なのはルート証明書のみです。証明書チェーンがあると思われる場合、次のJavaコマンドを使用するとすべての証明書をリスト表示して確認できます。

keytool -list -keystore "C:\java\jdk1.8.0_341\jre\lib\security\cacerts"

ヒント3：

更新したキーストアまたはトラストストアがJRE_HOMEまたはユーザーのホーム・ディレクトリにない場合は、JRE_HOMEにコピーします。上書きする前に、すべてのファイルのバックアップを必ず作成してください。

ヒント4：

Java SSL/TLSデバッグを有効化します。デバッグ・モードを有効にするには、次のようにFSALと目的のフォームを実行します。

java **-Djavax.net.debug=all** –jar frmsal.jar -url

https://yourserver:port/forms/frmservlet?config=standaloneapp

以下のように、出力をテキスト・ファイルにリダイレクトすることもできます（Microsoft Windowsの場合）。

java -Djavax.net.debug=all –jar frmsal.jar -url

"https://yourserver:port/forms/frmservlet?config=standaloneapp" >

C:/existing_directory/output.txt

ヒント5：

必要な証明書を含むキーストアまたはトラストストアが、（デフォルト・ロケーションにある）デフォルトではない場合、スイッチ“ keystore”または“truststore”を使用すると、目的のファイルを明示的に参照できます。

java **-Djavax.net.ssl.keystore**=C:/somewhere/mycacerts -jar frmsal.jar -url

“https://server:port/forms/frmservlet?config=standaloneapp”

または

java **-Djavax.net.ssl.truststore**=C:/somewhere/mycacerts -jar frmsal.jar -url

“https://server:port/forms/frmservlet?config=standaloneapp”

署名済みコード

アプリケーションの起動が要求されたとき、アプリケーションの所有者がわかっていると参考になります。ユーザーは、アプリケーション所有者が信頼できる既知の存在であるかどうかを確認し、信頼できない場合は処理の取消しを選択できます。アプリケーション所有者のIDを表示するには、アプリケーションにデジタル署名（証明書）を含める必要があります。Oracle Formsの場合、デジタル署名証明書はオラクルが提供するForms Java JARファイルに追加されます。

JARファイルに証明書を追加する方法については、"jarsigner"ユーティリティとそのドキュメントを参照してください。

Forms、Java Plug-in、Java Web Start

Java Plug-inまたはJava Web Startを使用してブラウザからFormsアプリケーションを実行するときには、ほとんどのユーザーがハイパーリンクやブックマークをクリックするか、ブラウザにURLを手動で入力します。Oracle Formsの使用とは関係なく、ユーザーがブラウザを使用するということは、そのシステムが潜在的なリスクにさらされることになるというのが純然たる事実です。インターネットからは多数の悪意あるWebサイトにアクセスできるため、意図的であってもそうではなくても、悪意あるサイトに遭遇してしまう事態はほぼ避けられません。Java Plug-inとJava Web Startには特別なセキュリティ機能が組み込まれており、ユーザーのマシンで悪意あるJavaアプリケーションが実行される危険性が軽減されています。そうした重要なセキュリティ機能の1つは、署名済み（信頼済み）アプリケーションのみの実行を許可するという要件です。署名されていないアプリケーションは許可されず、ブロックされます。信頼済みのコードで署名される証明書（デジタル署名）の取得にはコストがかかり、組織的な検証が必要になるため、この方法で悪意あるコンテンツを配信するというのは非常にまれですが、可能性はゼロではありません。

Oracle Formsで提供され、ユーザーのマシンで実行されるすべてのJava JARファイルは、前述のJavaの要件に準拠するために署名されています。そのため、ブラウザまたはJava Web Startを使用して、Formsアプリケーションを安全に実行することができます。カスタムJARファイル（オラクル以外の提供ファイル）についても、信頼済み証明書を使用して、適切に署名されている必要があります。オラクル提供以外のJARへの署名追加は、お客様の責任になります。自己生成した証明書の使用は推奨されていないので、使用しないでください。自己生成した証明書を使用しても機能するよう見えるかもしれませんが、将来的に、この機能はJavaでサポートされなくなる可能性があります。

FSALと署名済みJAR

FSALを使用する場合、すべてのデプロイメントでJARに署名することが推奨されます。ブラウザ使用に関連する固有のリスクは、FSALの使用によってほぼ回避されますが、このクライアント構成の選択は急速に広がっています。そのため、アプリケーション・デプロイメントのあらゆるセキュリティ要素を慎重に考慮することが、引き続き重要になります。

FSALなどのネイティブJavaアプリケーションを実行する場合は、Java Plug-inやJava Web Startと同レベルのセキュリティ・チェックが組み込まれているわけではありませんが、一定レベルの署名済みコードの検証を有効にすることは可能です。

以下の手順は、バージョン11より新しいJavaバージョンでは**非推奨**になっているため、Java 17以降でこの機能を有効にすると警告が表示される場合があります。ここでは、以前のソリューションのためだけに手順を示します。

FSALとユーザーのアプリケーションで署名済みコードが認識されるようにするには、以下の手順を実行してください。

1. アプリケーションで使用されているすべてのJARファイルに関連付けられた公開鍵を取得します。Oracleが提供するJARファイルは配信時に署名されており、この証明書はデフォルトでキーストアに含まれています。使用するJARファイル用の署名者の公開鍵がない場合は、以下のJavaコマンドを使用して、各JARに必要な証明書を取得できます。署名者の証明書はその後の手順で必要になるため、最初に取得します。

   ```
   keytool -printcert -rfc -jarfile yourJarFile.jar
   ```

2. ユーザーのマシンで、各公開鍵をJavaキーストアにインポートします。これは実行時に使用されます。

   ```
   keytool -importcert -alias foo -file C:\foo.pem
   ```

   上記のエイリアスは、英数字の文字列であれば何でもかまいませんが、この後の手順で必要になるため、覚えておいてください。また、-keystoreスイッチが含まれていないため、ユーザーのホーム・ディレクトリのキーストアが使用されます（ない場合は作成されます）。ファイル名は".keystore"です。他のキーストアやキーストア名を使用するには、必要に応じて-keystoreスイッチを追加してください。すべての証明書で、上記の手順を繰り返します。

"keytool"ユーティリティの使用に役立つ情報へのリンクが、本書の参考資料のセクションに記載されています。

3. .java.policyファイルがない場合や、完全にカスタマイズしたファイルが必要な場合は、ユーザーのホーム・ディレクトリに作成します。その他の場合は、デフォルトのファイル<JAVA_HOME>\jre\lib\security\java.policyを使用します。ファイルに以下を追加します。

```
keystore ".keystore";
grant signedBy "foo" {
    permission java.security.AllPermission;
};
```

"foo"とは、手順2で使用したエイリアスを示しています。手順2でインポートした鍵のそれぞれについて、"grant signedBy"セクションを追加します。keystoreの値は、使用する予定のキーストア・ファイルを示すものになります。

4. 少し修正を加えたこのコマンド・エントリを使用して、FSALでFormsアプリケーションを実行します。

```
java -Djava.security.manager -jar frmsal.jar -url
"https://yourserver:port/forms/frmservlet?config=standaloneapp"
```

アプリケーションの実行時には、JAR証明書はキーストア内の証明書と照合されます。一致しない場合、アプリケーションは実行されません。上記の例では、デフォルトのjava.policyが使用されている、またはユーザーのホーム・ディレクトリにjava.policyが作成されたと仮定しています。ユーザーのマシンの別のディレクトリまたはリモートの場所（URL）にある、カスタム・ポリシー・ファイルにアクセスすることもできます。

```
java -Djava.security.manager -Djava.security.policy=${user.home}/foo.policy
-jar frmsal.jar -url "https://yourserver:port/forms/frmservlet?config=standaloneapp"
java -Djava.security.manager -Djava.security.policy=https://yourserver:port/foo.policy
-jar frmsal.jar -url "https://yourserver:port/forms/frmservlet?config=standaloneapp"
```

この検証からユーザーが恩恵を受けるためには、起動スクリプトやそれに類するものを使用する必要があります。スクリプトを使用すると、入力ミスのエラーの可能性や、セキュリティ・マネージャの使用を忘れる可能性を回避するのに役立ちます。

シングル・サインオン

シングル・サインオンは、Webアプリケーションに数多くのメリットをもたらす認証概念です。たいていの場合、Formsとシングル・サインオン（SSO）を統合するという考えには、Formsデータベースのログイン機能使用時に少なくとも2つの大きな利点があると見なされます。まず、ユーザーがブラウザを開いている間に必要なアプリケーション認証が1回だけになります。この認証は、その他のブラウザ・アプリケーションやFormsアプリケーションと共有されるので、それぞれのアプリケーションからログインが要求されることはなく、最初の1回で完了します。これは、同じSSOサーバーで保護されるすべてのアプリケーションと、同じブラウザ・セッションから実行または起動されるアプリケーションに適用されます。次に、Formsアプリケーションを実行するためのデータベースの資格証明をユーザーから隠せるというメリットがあります。

FSALでSSOを使用するには、その他のケースでFormsとSSOを使用するのに必要な設定手順と同じ手順を実行する必要があります。FormsでのSSOの設定方法とその仕組みについて、詳しくはOracle Formsの操作を参照してください。この共通の設定が完了したら、Fusion Middleware Controlを使用して、Formsテンプレート・ファイルであるbasesaa.txtとwebutilsaa.txtに**ssoMode=%ssoMode%**を追加します。特定のアプリケーションでSSOを有効にするには、保護対象アプリケーションに関連付けられたFormsのWeb構成設定で、目的のアプリケーション設定に**ssoMode=true**を設定します。

FSALリファレンス

このセクションには、Web構成パラメータ、コマンドライン引数、関連するエラーメッセージが含まれています。詳細については、Working with Oracle Forms Guideを参照してください。

FSAL Web構成パラメータ

FormsのWeb構成パラメータについて、詳しくはOracle Formsの操作を参照してください。

サーブレット/アプレットのパラメータ	説明
ignoreSaaCache	既存のキャッシュを無視して、新しいファイルをサーバーからダウンロードするかどうかを指定します。 有効な値：TRUE、FALSE デフォルト：FALSE
ignoreMissingSaaArchives	指定されているが見つからないJARファイルがある場合、起動時にこれを無視するかどうかを指定します。このパラメータの使用は、一般に、テストおよびデバッグ時のみとしてください。TRUEに設定されているときにアーカイブが見つからないと、エラーが発生しますが、セッションは続行を試みます。これはデフォルトの動作です。FALSEに設定されている場合、FSALは可能であれば続行を試み、そうでない場合は失敗します。 有効な値：TRUE、FALSE デフォルト：FALSE
fsalJavaVersion	アプリケーション実行に必要なJavaバージョンを指定します。取りうる値の組合せについては、Oracle Formsの操作で、巻末近くにあるWeb構成パラメータ表を参照してください。
ssoSaaBrowserLaunchTimeout	Formsサーブレットが、FSALによってSSO認証用に起動されたブラウザからの初回リクエストを待つ時間を秒数で指定します。指定された時間が経過すると、致命的エラーFRM-93249が返されます。 有効な値：1～300の整数 デフォルト：15
ssoSaaBrowserPageTimeout	Formsサーブレットが、FSALアプリケーションのSSO認証中にユーザーによるブラウザ・ページへのデータ入力を待つ時間を秒数で指定します。指定された時間が経過すると、致命的エラーFRM-93382またはFRM-93383が返されます。 有効な値：15以上の整数または0（無期限に待機） デフォルト：0
ssoSaaWaitInterval	起動されたブラウザでSSO認証が処理されている間に、FSALがFormsサーブレットへのリクエストを再発行する間隔を秒数で指定します。値が大きいとネットワーク・トラフィックが減りますが、タイムアウトして致命的エラーFRM-93248が発生する可能性が高くなります。 有効な値：5以上の整数または0（リクエストを再発行しない） デフォルト：25
ssoSuccessLogonURL	FSALアプリケーションのSSO認証が成功した場合のリダイレクト先URL。

FSALコマンドライン引数

Formsコマンドライン引数について、詳しくはOracle Formsの操作を参照してください。

コマンドライン引数	説明
-url	Formsアプリケーションの実行に必要な完全修飾URLを指定します。URLには、目的のFSAL設定を含む構成参照を含める必要があります（例：config=standaloneapp）。 URLを引用符で囲みます。 SSL/TLSを使用しない場合、URLエントリでプロトコル（http://）を省略できます。
-t	タイムアウトするまでにランチャがサーバーからの最初の応答を待つ時間をミリ秒で指定します。 有効な値：1以上の正の整数 デフォルト：60000（ms）
-showConfig	アプリケーションのロード時に、FormsのWeb構成パラメータをコマンドラインに表示するかどうかを指定します。 有効な値：TRUE、FALSE デフォルト：FALSE
-showDetails	アプリケーションのロードと起動に関する追加情報を表示するかどうかを指定します。 有効な値：0、1、2、99 値の説明： 0 追加情報を何も表示しない。 1 リソースのロード元ロケーションを表示する。 2 証明書情報がFSALトラストストアに含まれない場合、SSL/TLS関連情報を表示する。 99 すべての情報を表示する。 その他の値は将来的な使用のために確保されています。 デフォルト：0
-changeFSALStorePass	FSALカスタム・トラストストアのパスワードを変更するかどうかを指定します。作成時にデフォルト・パスワードが設定されています。デフォルト・パスワードは、changeitです。 ほとんどの場合、トラストストアに秘密鍵/証明書情報が含まれていない限り、ユーザーのマシン上のトラストストアにパスワードを設定する理由はありません。 有効な値：TRUE、FALSE デフォルト：FALSE
-autoImportCert	ユーザー操作なしでSSL/TLS証明書をインポートするかどうかを指定します。cert_truststoreが設定されている場合、この引数は無視されます。 有効な値：TRUE、FALSE デフォルト：FALSE

-cert_truststore	SSL/TLS証明書を保管するために、Formsトラストストアまたはカスタム・キーストアのどちらを使用するかを指定します。"forms"という値を指定するか、何も指定しない場合、Formsが生成したJavaトラストストアが使用されます。"java"という値を指定すると、FSALはJavaのデフォルト・キーストアを使用してSSL/TLS証明書を保存および検証します。Javaのデフォルト・ストアを使用する場合は、SSL/TLSアプリケーションを実行する前に、必要な証明書をここに手動でインポートしておく必要があります。 有効な値：forms、java デフォルト：forms
-clearcache	すべてのFSALキャッシュを消去するかどうかを指定します。-clearcacheを一緒に使用した場合、-url引数は無視されます。 有効な値：TRUE、FALSE デフォルト：FALSE

参考資料

https://www.oracle.com/jp/application-development/technologies/forms/forms.html

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html

https://docs.oracle.com/javase/tutorial/security/tour2/step2.html

https://docs.oracle.com/javase/tutorial/security/toolsign/rstep1.html

https://docs.oracle.com/javase/tutorial/security/toolsign/rstep2.html

https://docs.oracle.com/javase/tutorial/security/toolsign/rstep3.html

https://docs.oracle.com/javase/8/docs/technotes/guides/security/PolicyFiles.html

https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/JSSERefGuide.html

https://docs.oracle.com/javase/8/docs/technotes/guides/security/jsse/ReadDebug.html

https://openjdk.java.net/jeps/411

https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.html

https://ja.wikipedia.org/wiki/Transport_Layer_Security

https://www.digicert.com/jp/what-is-an-ssl-certificate

https://en.wikipedia.org/wiki/Chain_of_trust

https://www.openssl.org

https://www.geeksforgeeks.org/difference-between-truststore-and-keystore-in-java

Connect with us

+1.800.ORACLE1までご連絡いただくか、**oracle.com**をご覧ください。北米以外の地域では、**oracle.com/contact**で最寄りの営業所をご確認いただけます。

blogs.oracle.com facebook.com/oracle twitter.com/oracle