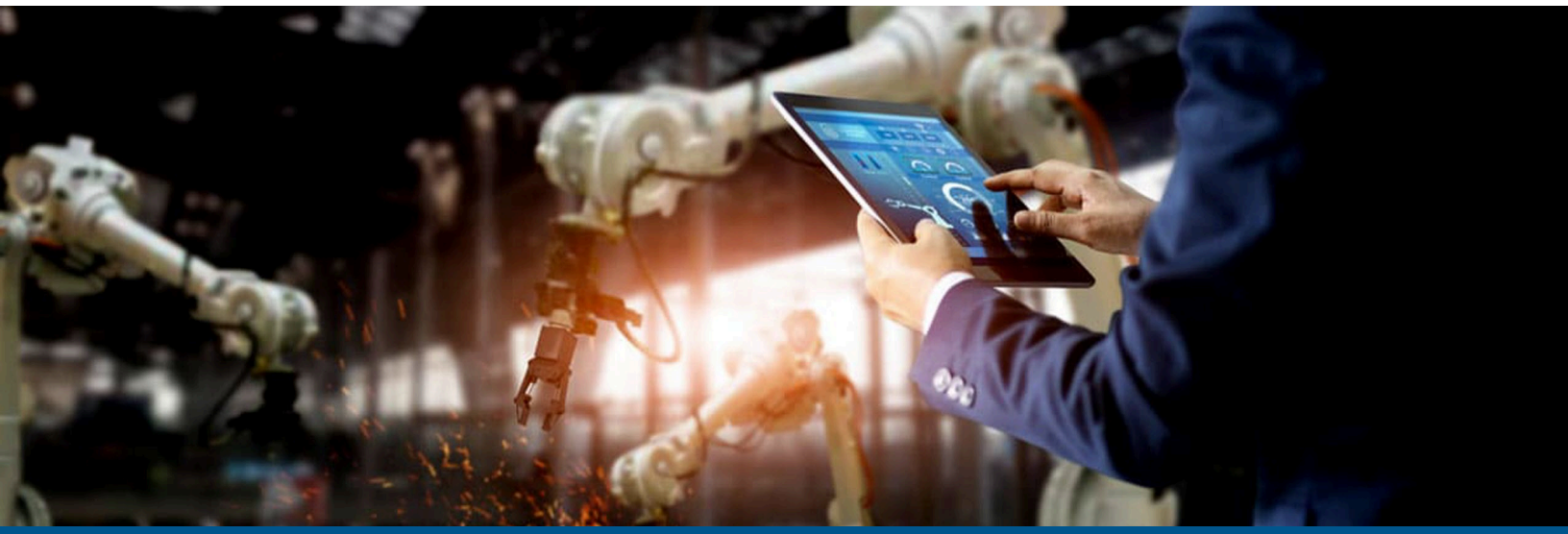


Kofax RPA
管理者ガイド

バージョン: 11.5.0

日付: 2023-10-02

KOFAX

目次

はじめに

本ガイドは、エンタープライズ環境で Kofax RPA を展開するシステム管理者を対象としています。

以前の Kofax RPA のバージョンのいずれかを実行している場合は、アップグレード手順が記載された『Kofax RPA アップグレード ガイド』を参照してください。

そのガイドには次に挙げる Kofax RPA の管理情報が含まれています:

- ランタイム
- Tomcat Management Console
- Management Console 監査ログ
- Kofax RPA テーブルの SQL スクリプト

関連ドキュメント

Kofax RPA のドキュメント セットには次の場所からアクセスできます。[1]

https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm

ドキュメント セットには、次のようなリソースがアルファベット順で含まれています。

Kofax RPA 管理者ガイド
Kofax RPA での管理タスクについて説明します。

Kofax RPA のベストプラクティス ガイド
Kofax RPA 環境でロボット ライフサイクル マネジメントを使用しながらパフォーマンスを最適化し、成功を確実にするために推奨される方法とテクニックを提供します。

Kofax RPA Desktop Automation サービス ガイド
リモート コンピューターで Desktop Automation を使用するために必要な Desktop Automation サービスを設定および管理する方法について説明します。

Kofax RPA 開発者ガイド
RoboServer でロボットを実行するために使用される Java および .NET API のプログラマー ユーザー ガイドが含まれています。また、製品で提供される Management Console REST サービスに関する情報が含まれています。

[1] オンラインのドキュメント セットにアクセスするにはインターネットに接続する必要があります。インターネットに接続せずにアクセスする方法については、『インストール ガイド』を参照してください。

Kofax RPA ロボット構築の開始ガイド
Kofax RPA を使用してロボットを構築するプロセスを実行するためのチュートリアルを提供します。

Kofax RPA Document Transformation スタート ガイド
OCR、抽出、フィールドの書式設定、検証などを含む Kofax RPA 環境の Document Transformation 機能を使用する方法について説明します。

Kofax RPA のヘルプ
Kofax RPA の使用方法について説明しています。ヘルプは、『Kofax RPA ユーザー ガイド』という PDF 形式のドキュメントとしても提供されています。

Kofax RPA インストール ガイド
Kofax RPA およびそのコンポーネントを開発環境にインストールする方法について説明します。

Kofax RPA Java API documentation (Kofax RPA Java API ドキュメント***)***
開発者が Kofax RPA で使用できる Kofax RPA Java API パッケージおよびクラスへのアクセスを提供します。

Kofax RPA API は、元の製品名である「RoboSuite」に対する詳細な参照を含んでいます。RoboSuite の名前は下位互換性を確保するために残されています。API ドキュメントの中では、RoboSuite という用語は Kofax RPA と同じ意味で使われています。

Kofax RPA リリース ノート
その他の Kofax RPA ドキュメントからは入手できない最新の詳細やその他の情報が含まれています。

Kofax RPA 技術仕様
サポートされるオペレーティング システムおよびその他のシステム要件に関する情報が含まれています。

Kofax RPA アップグレード ガイド
Kofax RPA やそのコンポーネントを新しいバージョンにアップグレードする手順が含まれています。

Kofax RPA ユーザー ガイド
Kofax RPA とそのコンポーネントの使用手順が記載されています。*Kofax RPA* のヘルプ トピックに加えて、ヘルプに記載されていない詳細な内容が含まれています。

システム要求

サポートされるオペレーティング システムおよびその他のシステム要件については、以下の Kofax RPA 製品のドキュメント サイトの『Kofax RPA 技術仕様』ドキュメントを参照してください: https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm。

トレーニング

Kofax は、Kofax RPA ソリューションを最大限に活用するために、教室でのトレーニングとコンピュータでのトレーニングを提供しています。利用可能なトレーニング オプションとスケジュールの詳細については、https://learn.kofax.com/ の Kofax 教育ポータルを参照してください。

また、https://smarthub.kofax.com/ の Kofax Intelligent Automation SmartHub にアクセスして、追加のソリューション、ロボット、コネクタなどを見つけることもできます。

Kofax 製品のヘルプの入手

[Kofax Knowledge Portal (Kofax ナレッジ ポータル)] リポジトリにある記事の内容は定期的に更新され、Kofax 製品の最新情報について参照できます。製品に関してご不明の点がある場合は、Knowledge Portal (ナレッジ ポータル) で情報を検索することをお勧めします。

[Kofax Knowledge Portal] にアクセスするには、https://knowledge.kofax.com にアクセスしてください。

[Kofax Knowledge Portal] は Google Chrome、Mozilla Firefox、または Microsoft Edge 向けに最適化されています。

[Kofax Knowledge Portal] は以下の内容を提供します。

- 強力な検索機能で必要な情報をすぐに見つけることができます。
  [Search (検索)] ボックスに目的の語句を入力し、検索アイコンをクリックしてください。
- 製品情報、設定の詳細、リリース情報などのドキュメント。
  記事を見つけるには、Knowledge Portal のホームページにアクセスし、製品に該当するソリューション ファミリを選択するか、[View All Products (すべての製品を表示)] ボタンをクリックします。

Knowledge Portal のホームページからは、次の操作を実行できます。

- Kofax Community (Kofax コミュニティ) へのアクセス (全カスタマー)。
  [Resources (リソース)] メニューで、**[Community (コミュニティ)]** リンクをクリックします。
- Kofax Customer Portal (Kofax カスタマー ポータル) へのアクセス (一部のカスタマーのみ)。
  [Support Portal Information (サポート ポータルの情報)] ページに移動し、**[Log in to the Customer Portal (カスタマー ポータルにログイン)]** をクリックします。
- Kofax Partner Portal (Kofax パートナー ポータル) へのアクセス (一部のパートナーのみ)。
  [Support Portal Information] ページに移動し、**[Log in to the Partner Portal (パートナー ポータルにログイン)]** をクリックします。
- Kofax サポート コミットメント、ライフサイクル ポリシー、電子フルフィルメントの詳細、セルフサービス ツールへのアクセス。
  [Support Details (サポートの詳細)] ページに移動し、適切な記事を選択します。

第 1 章

ランタイム

Kofax RPA は開発したロボットを実行するための多くのツールを提供しています。次のセクションでは、これらのツールについて説明します:

- RoboServer は、リモート クライアントがロボットを実行できるようにするサーバー アプリケーションです。これを設定するには、Management Console および RoboServer 設定アプリケーション (セキュリティや認証などの詳細設定用) を使用します。
- Management Console は、ロボットの実行をスケジュールし、ログと抽出されたデータを表示するために役立ちます。また、RoboServer のクラスタを設定するための一元的な場所も提供します。

⚠ Kofax RPA のいずれかのコンポーネントの設定を変更した後、変更を有効にするためにそれぞれのコンポーネントを再起動します。

プロパティを使用して追加された設定は、ユーザー インターフェイスで構成された設定よりも優先されます。

ℹ タイム ゾーン定義は、バンドルされた JRE に組み込まれています。リリース日以降に定義が変更された場合、Oracleが提供する *Timezone Updater Tool* を使用して JRE を更新できます。詳細については、Oracle の Web サイトを参照してください。

RoboServer

RoboServer は Design Studio で作成されたロボットを実行します。ロボットは次のようにさまざまな方法で起動できます: 特定の時間に実行するようにスケジュールされている Management Console、REST Web サービス、Java または .NET API、または Kapplet からの実行。

デフォルトのブラウザ エンジンで作成されたロボットを実行するには、Linux の最小インストールに次のパッケージが含まれている必要があります。

- `libX11.so.6`
- `libGL.so.1`
- `libXext.so.6`

`yum install` または `sudo apt-get` を使用して、Linux プラットフォームに必要なライブラリをインストールするよう要求します。

また、Webkit ロボットを実行するためのすべてのフォントがシステムにインストールされていることを確認してください。一部の Linux インストール パッケージにはフォントが含まれていないため、これはヘッドレス Linux インストールを使用する場合に必要になることがあります。

フォントをインストールするには、以下の手順を参照してください。

- Instructions for installing fonts for CentOS / RedHat
- Instructions for installing fonts for Ubuntu

RoboServer がロボットを実行できるようにするには、Management Console によって RoboServer が有効化されている必要があります。RoboServer は、有効なライセンスで Management Console のクラスタに属しており、十分な KCU がクラスタに割り当てられている場合にアクティブになります。Management Console はクラスタ上で設定された RoboServer から設定も受け取ります。

RoboServer と Management Console の間では、次のような 3 つの接続タイプを利用できます。

- クライアント接続: RoboServer は Management Console への HTTP(S) 接続を作成し、指定されたクラスタに登録されます。この接続タイプは、「クライアント」タイプのクラスタでのみ使用してください。通常、この接続タイプは、Kofax RPA がクラウド環境に展開されている場合に使用されます。これは、高可用性モードではサポートされません。
- ソケット サービス: RoboServer はサービスとして機能します。指定されたクラスタに RoboServer が登録された後に、Management Console はソケット接続を開始します。この接続タイプは、「サービス」タイプのクラスタでのみ使用してください。
- ソケット **SSL** サービス: RoboServer はサービスとして機能します。指定されたクラスタに RoboServer が登録された後に、Management Console は暗号化されたソケット接続を開始します。この接続タイプは、「Service SSL」タイプのクラスタでのみ使用してください。

次のオプションを設定します。

- RoboServer 設定アプリケーション内。RoboServer の設定を参照してください。
- コマンド ライン内。RoboServer の開始を参照してください。

RoboServer およびクラスタの管理の詳細については、『Kofax RPA のヘルプ』の「Management Console」の章を参照してください。

Kofax RPA バージョン 11.2.0 以降では、RoboServer は UTC 時間でログを書き込みます。デフォルトでは、11.2.0 より前のバージョンの RoboServer は、ローカル サーバー時間でログを書き込みます。そのため、11.2.0 以降とそれより前のバージョンの両方の RoboServer が同じログ データベースにログを記録すると、タイムスタンプに不整合が生じる可能性があります。11.2.0 より前のバージョンの RoboServer を Management Console バージョン 11.2.0 以降に接続する場合は、`RoboServer.conf` ファイル内の次のオプションをコメント解除することで、ローカル サーバー時間ではなく UTC 時間でログ メッセージを書き込むように設定できます。

```
wrapper_java_additional.41=-DwriteLogdbUtc=true
```

RoboServer は、このパラメータをサポートするフィックス パック バージョンに更新する必要があります。詳細については、対応するフィックス パックの ReadMe ファイルを参照してください。

RoboServer の開始

RoboServer は、次のように複数の異なる方法で開始することができます。

- コマンド ラインから呼び出す。
- サービスとして実行する。サーバーを自動的に起動を参照してください。
- Docker コンテナを実行する。Kofax RPA の展開のための Docker ツールを参照してください。
- RoboServer プログラム アイコン (または、Management Console と RoboServer を開始する、[スタート] メニューの [Start Management Console](Management Console を起動)プログラム アイコン) をクリックする。デモおよびテストのみを目的としています。

コマンド ラインから RoboServer を呼び出すには、コマンド プロンプト ウィンドウを開き、以下の `Kofax RPA` インストール フォルダにある `bin` フォルダに移動し、次のコマンドを実行します:

```
RoboServer
```

必要なパラメータがすべて設定ファイルで指定されている場合は (`C:Users\[ユーザー]\AppData\Local\[バージョン]\Kofax RPA\Configuration\roboserver.settings`)、RoboServer が開始されます。

必要なパラメータのいずれかが欠落している場合、RoboServer はエラーを生成し、使用法のヘルプと使用可能なパラメータを表示します。

RoboServer パラメータ

RoboServer を開始するためのコマンド ラインは、次のパラメータを含めることができます。

```
RoboServer [-client]  [-s <service:params>] [-mcUrl <url>] [-ss <MC Shared Secret> [-cl
 <Cluster Name>] [-b <url>] [-p <port number>] [-sslPort <port number>] [-v]
```

RoboServer は、次の表のパラメータを受け入れます。RoboServer 設定アプリケーションですべてのパラメータを編集できることに注意してください。詳細については、RoboServer の設定 を参照してください。Docker 環境では、`docker-compose` ファイルに環境変数を設定する必要があります。

パラメータ	説明
`-mcUrl <arg>`	この必須パラメータでは、次の形式で登録する Management Console を指定します。 `http[s]://[ホスト名]:[ポート番号]` 例：`-mcUrl http://myserver:8080/ManagementConsole` ロボットでコールバック オプションとともに Document Transformation ステップを使用する場合は、`-mcUrl` パラメータで RoboServer ホスト名または IP アドレスを使用します。Document Transformation サービスが Management Console にアクセスできなくなり、コールバック ロボットがキューに格納されなくなるため、「localhost」は使用しないでください。
`-ss` `--mcSharedSecret [MC 共有シークレット]`	この必須パラメータにより、RoboServer を Management Console で認証するために使用される共有シークレットを指定します。共有シークレットは、Management Console の [サービス認証] セクションからコピーする必要があります。詳細については、『Kofax RPA のヘルプ』の「サービス認証」を参照してください。
`-cl` `--cluster <arg>`	この必須パラメータは Management Console で指定されたクラスタに RoboServer を自動的に登録します。以下の例では、RoboServer が *Production* クラスタにそれ自体を登録します。 例：`-cl Production` 例：`-mcUrl http://myserver:8080/ManagementConsole -ss [MC 共有シークレット] -cl Production`
`-eh` `--externalHost <ポート番号>`	RoboServer ホストの名前または IP アドレスを明示的に指定します。 このパラメータは、クラウド内で NAT を使用して実行しているとき、または Docker コンテナ内で RoboServer を実行しているときのように、RoboServer がローカルマシンで見つけたものとホスト アドレスが異なる場合に指定する必要があります。 例：`-eh 10.10.0.123`
`-ep` `--externalPort <ポート番号>`	RoboServer ホストのポート番号を明示的に指定します。 クラウド内で NAT を使用して実行しているとき、または Docker コンテナ内で RoboServerを実行しているときのように、RoboServer がローカル マシンで見つけたものとホスト ポートが異なる場合、このパラメータを指定する必要があります。
`-jmxPass`	JMX アプリケーションを使用して RoboServer を監視し、パスワードが必要な場合、JMX パスワードを設定します。
`-v` `--verbose`	このオプション パラメータにより、RoboServer がステータスおよびランタイム イベントを出力します。
`-V` `--version`	このオプション パラメータによって、RoboServer でバージョン番号が出力され、終了します。

パラメータ	説明
-h --help	ヘルプを表示します。
-pauseAfterStartupError	起動時にエラーが発生した場合に一時停止します。
-s --service <service-name:service-parameter>	このパラメータによって、RoboServer で開始する必要がある RQL または JMX サービスが指定されます。このパラメータは少なくとも一度指定する必要があり、同じ RoboServer で複数のサービスを開始するには複数回指定する場合があります。利用可能なサービスは、インストールによって異なります。 例：--service socket:50000 例：--service jmx:50100 詳細については、下の表で「利用可能なサービス」を確認してください。
-p --port <ポート番号>	これは、-s socket:<ポート番号> を呼び出すための省略形です。 例：--port 50000
-sslPort <ポート番号>	これは、-s ssl:<port number> を書き込むための省略形です。
-nd --NoDoc	このオプション パラメータにより、この RoboServer に対するロボットのドキュメント リクエストを禁止します。
-sn --serverName	このオプションのパラメーターは、後で Kofax Analytics for RPA に表示される、RoboServer 統計を記録するためのサーバー名を設定します。サーバー名を指定しない場合、統計はサーバーの IP アドレスに基づいて収集されます。
-ll --licenseLimit <arg>	このパラメータは、RoboServer が受け取ることができるライセンス ユニットの最大数を指定します。
-client	RoboServer が Management Console クラスタ向けのクライアントとして実行されるように指定します。高可用性モードではサポートされません。
利用可能なサービス	
--service socket:<portNumber>	RoboServer が Management Console クラスタ向けのサービスとして実行されるように指定します。 <portNumber>: 監視対象のソケット サービスのポート番号。
--service ssl:<portNumber>	RoboServer を Management Console クラスタ向けのサービスとして実行し、セキュアな接続を作成することを指定します。 <portNumber>: 監視対象のソケット サービスのポート番号。
--service jmx:<jmx_port_Number>,<jmx_rmi_url>	<jmx_port_Number>: 監視対象の JMX サービスのポート番号。 <jmx_rmi_url>: JMX サービス向けのオプションの RMI ホストとポート。ファイアウォールを接続する必要がある場合に使用します。 例：--service jmx:example.com:51001

RoboServer と Management Console の間の接続タイプを設定するには、次のパラメータのいずれかを選択します。

- `-client`
- `--service socket:<portNumber>`
- `--service ssl:<portNumber>`

RoboServer に接続するコマンド ラインの例:

- Management Console クラスタ向けのクライアントとして:

  ```
  -client -mcUrl http://localhost:8080/ManagementConsole -cluster <clientCluster> -ss
   <MC Shared Secret>
  ```

  ここで、`clientCluster` は、Management Console で作成されたクライアント接続タイプのクラスタの名前です。
- Management Console クラスタ向けのサービスとして:

  ```
  -service socket:50000 -mcUrl http://localhost:8080/ManagementConsole -cluster
   <serviceCluster> -ss <MC Shared Secret>
  ```

  ここで、`<serviceCluster>` は、Management Console で作成されたサービス接続タイプのクラスタの名前です。

共有シークレットを設定するには、RoboServer 設定アプリケーションを使用します。詳細については、「RoboServer の設定」を参照してください。

⚠ Kofax RPA バージョン 10 以降、すべての RoboServer は Management Console に自動登録する必要があります。したがって、Management Console を起動するときに、RoboServer の URL と共有シークレット、およびクラスタ名を指定する必要があります (次の例のようにコマンド ラインで指定するか、[Management Console に登録] オプションで RoboServer 設定アプリケーションを使用します)。

`RoboServer.exe -mcUrl http://myserver:8080/ManagementConsole -ss [MC 共有シークレット] -cluster Production -service socket:50000`。

サーバーを自動的に起動

インストールに結合された RoboServer と Management Console サーバー機能が含まれている場合は、サーバーが自動的に開始するように設定できます。

これら 2 つの機能は、開始時に指定された引数に応じて、同じ RoboServer サーバー プログラムによって提供されます。

「RoboServer パラメータ」セクションには、RoboServer プログラム用のコマンドライン引数の詳細な説明が含まれています。RoboServer プログラムを有効にするには口ボットを実行し、 `-service` 引数を指定します。同様に、`-MC` 引数によって Management Console 機能を有効にします。

RoboServer およびその他の RPA コンポーネントをサービスとして開始する方法については、「サービスとしての RPA コンポーネントの実行」を参照してください。

RoboServer のシャットダウン

RoboServer は、次のコマンド ライン ツールを使用してシャットダウンできます。引数なしで `ShutDownRoboServer` を実行して、サーバーをシャットダウンする方法、特に現在サーバーで実行中の ロボットを処理する方法のさまざまなオプションを確認します。

本番構成

RoboServer は Design Studio で作成された ロボットを実行します。ロボットは次のようにさまざまな方法で起動できます: 特定の時間に実行するようにスケジュールされている Management Console、REST Web サービス、Java または .NET API、または Kapplet からの実行。

安定したパフォーマンスの実稼働環境を得るために、デフォルトの RoboServer パラメーターの一部を調整が必要となる場合があります。次の設定オプションを見ていきます。

- RoboServer インスタンスの数
- 同時に実行される ロボットの数
- メモリ割り当て

RoboServer インスタンスの数

RoboServer は Oracle の Java 仮想マシン (JVM) で実行されます。JVMは、ハードウェア上で実行されるオペレーティング システム (OS) で実行されます。JVM および OS にはパッチが適用され、ハードウェア アーキテクチャが変更され、新しい反復のたびにパフォーマンスの向上が図られます。パフォーマンスに関しては一般的な複数のガイドラインが提供されていますが、最適な設定を確認する唯一の方法は、テストすることです。

一般的なルールとして、RoboServer の 2 つのインスタンスを起動することで、わずかにパフォーマンスが向上します。JVM は、ガベージ コレクション (GC) と呼ばれるメモリ管理を使用します。ほとんどのハードウェアでは、GC 中にアクティブになる CPU コアが 1 つだけであるため、クアッドコア CPU では CPU 待機率は 75% のままになります。RoboServer の 2 つのインスタンスを起動した場合、一方のインスタンスは引き続きフル CPU を使用し、もう一方のインスタンスは GC を実行します。ただし、ガベージ コレクタの CPU 使用率は使用環境が動作する JDK 仕様に依存するため、GC プロセス中に複数の CPU コアを使用できることに注意してください。

同時に実行される ロボットの数

RoboServer が同時に実行される ロボットの量は、CPU の量、さらに RoboServer がプロセスに必要なデータを取得する速度によって異なります。同時に実行される ロボットの数は、Management Console [クラスタ設定] で設定されます。遅い Web サイトに対して実行する ロボットは、速い応答時間で Web サイトに対して実行する ロボットよりもはるかに少ない CPU を使用します。次の説明がその理由です。プログラムが使用する CPU の量は、次の式で説明できます。

`CPU ( コア ) %= 1-WaitTime / TotalTime`

ロボットの実行には 20 秒かかりますが、Web サイトの待機に 15 秒かかっている場合、ロボットは 5 秒間しか実行していないため、20 秒間で (CPUコアの) 平均 25% を使用しています。ロボットのステップは順番に実行されます。つまり、単一の実行 ロボットは、一度に 1 つの CPU コアのみを使用します。最新の CPU の多くは複数のコアを備えているため、ロボットは 20 秒で実行されますが、15 秒待機し、実際にはクアッドコア CPU の約 6% しか使用しません。

デフォルトでは、RoboServer は 20 台の ロボットを同時に実行するように設定されています。同時に実行される ロボットの数は Management Console [クラスタ設定] で設定されています。すべての ロボットが 6% のCPU を使用している場合、16 ~ 17 個の ロボットを同時に実行しているときに CPU が完全に使

用されます。これらの 6% のロボットのうち 33 個を同時に起動すると、RoboServer が過負荷になります。使用可能な CPU の量は一定であるため、結果として、各ロボットの終了には 2 倍の時間がかかります。現実の世界では、ロボットの CPU 使用率は、ロボット ロジックとそれが対話する Web サイトに応じて、CPU コアの 5 ~ 95% の間です。結果として、最大同時ロボット数の正しい値を推測または計算することは困難です。正しい値を確保する唯一の方法は、負荷テストを実行して、RoboServer CPU 使用率、および負荷が増加したときのロボット ランタイムを監視することです。

メモリ割り当て

各 RoboServer が処理できる同時に実行されるロボットの数に影響を与える可能性がある別のパラメーターはメモリの量です。ロボットが使用するメモリの量は、数メガバイト (MB) から数百 MB までさまざまです。デフォルトでは、RoboServer は、64 ビット システムに 2048 MB を使用するよう設定されています。RAM 割り当てを変更する を確認してメモリ割り当ての制御方法を確認します。メモリ不足エラーを回避するには、RoboServer に十分なメモリを指定します。適切なメモリ割り当てを確保するには、負荷テスト中にメモリ使用率を監視します。JVM は、使用可能なメモリをすべて専有することはありませんが、メモリは OS からも予約されています。JVM がメモリの使用を開始すると、OS に戻されません。最適なメモリ割り当てを見つけるには、CPU を 100% にプッシュする一連の負荷テストを実行します。各テストが完了したら、JVM (java.exeプロセス) が実際に使用した予約メモリの量を確認します。2048Mb (デフォルト) をすべて使用した場合は、メモリを増やし (通常は 2 倍)、テストを再度実行します。ある時点で、JVM は予約されたメモリのすべてを使用せず、使用されたメモリの数は実際のメモリ要件を反映し、RoboServer に指定する必要があります。

RoboServer のスケーリングの使用

デフォルトの方法は負荷分散スケーリングです。スケーリング方法を定義する RoboServer の特性は、CRE の量ではなく、同時に実行されるロボットの空きスロットの量です。その量が等しい場合、スケーリング方法によりキュー サイズがチェックされます。

オンサイト展開の場合、ロボットは負荷分散方式を使用して RoboServer ごとにスケジュールされます。これらの環境では、ロボットは、空き実行スロットが最も多い RoboServer のキューに入れられます。作業はそれぞれの RoboServer に均等に分散されます。

マルチテナントのクラウド展開の場合、RoboServer を別の方法でスケーリングする必要があることがあります。クラウド環境でのみ使用できるスケーラブルな方法として、空の実行スロットを持つ任意の RoboServer でロボットをキューに入れ、利用可能なスロットが最も少ないものを優先します。RoboServer のアイドル状態がより速くなった場合、この方法によって、未使用の RoboServer がスケールダウンされます。この方法を使用することで、最大数に達しておらず、RoboServer がシャットダウン モードでない場合に限り、スケーラビリティが最適化されます。

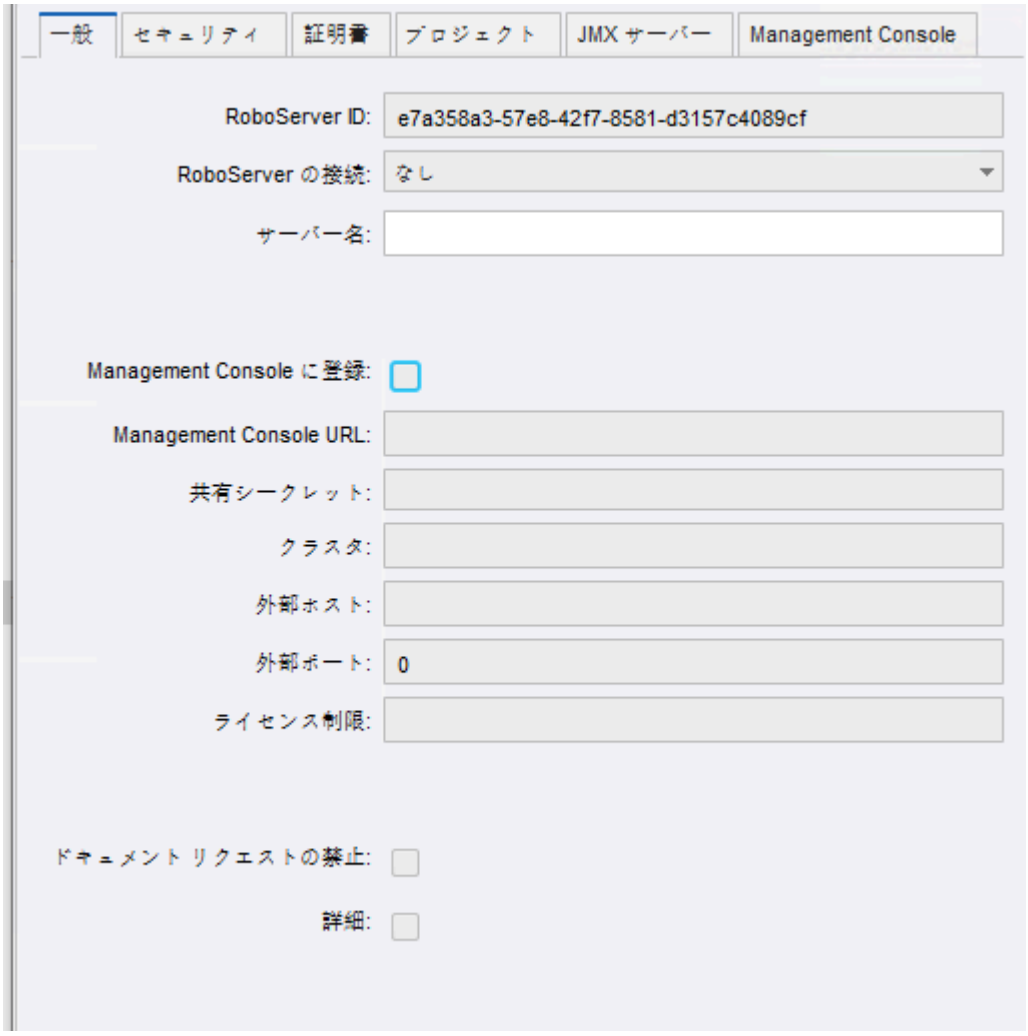
起動時のスケーリング方法を変更します。Management Console の実行中にメソッドを変更した場合、変更を有効にするには再起動が必要になります。

次のいずれかのファイルでスケーラブルな方法を設定します。

- Docker 設定ファイルで、`SETTINGS_CLUSTERMANAGER_DISTRIBUTIONSTRATEGY` 設定を指定します。
- `common.xml` ファイルで、`<property name="distributionStrategy"="SCALABLE"/>` 設定を指定します。

RoboServer の設定

RoboServer は、Windows のスタート メニューから起動する RoboServer 設定アプリケーションで設定できます。



RoboServer 設定メイン ウィンドウ

このアプリケーションを使用して以下の設定を行います。

- 一般: RoboServer 接続オプション、共有シークレットを含む Management Console 接続オプション、RoboServer ホスト設定、ライセンス ユニットの数、および詳細オプション。
- セキュリティ: 認証や権限などの セキュリティ設定。
- 証明書: 証明書 の使用。
- プロジェクト: デフォルト プロジェクト の場所。
- **JMX** サーバー: JMX サーバーの設定 。
- [**Management Console**]: 組み込み Management Console の設定。

設定を変更した後に、[**OK**] をクリックして新しい設定を保存し、実行中のいずれかの RoboServer を再起動して変更を有効にします。

Kofax RPA バージョン 10 以降、すべての RoboServer は Management Console に自動登録する必要があります。そのため、クラスタ名と Management Console の URL および共有シークレット

は、RoboServer の開始時に指定する必要があります (コマンド ライン、または **RoboServer** 設定アプリケーションを使用)。

[RoboServer ID] は、**Management Console** > **[管理]** > **[RoboServer]** > **[サーバー]** タブで RoboServer を識別するために使用される一意の識別子です。`roboserver-id` 変数値を編集することで、`roboserver.settings` 設定ファイルでこの ID を変更できます。

クラウド内で NAT を使用している場合、または Docker コンテナ内で RoboServer を実行している場合など、RoboServer がローカル コンピューターで見つけたものと RoboServer ホストの名前、IP アドレスとポート番号が異なっていれば後者を指定する必要があります。

RoboServer が使用する RAM の最大量を変更する必要がある場合は、RAM 割り当てを変更する を参照してください。

組み込み Management Console でのライセンスの開始と入力

Management Console にライセンス情報を入力する前に、Management Console を開始する必要があります。組み込まれた Management Console を使用する場合、次のように開始します。Tomcat Management Console についての情報は、Tomcat の展開 を参照してください。

Windows

スタート メニューの **[Start Management Console]** (Management Console を起動) の項目を使用します。

コマンド ラインから Management Console を開始するには、インストール フォルダの `bin` サブフォルダで次のコマンドを実行します。

```
RoboServer.exe -p 50000 -MC -mcUrl http://localhost:50080
```

コマンド ラインを使用して RoboServer を開始し、Management Console に登録することもできます。

`RoboServer.exe -p 50000 -MC -mcUrl http://localhost:50080 -cl "Production"` コマンドで、ポート `50000` で RoboServer を開始し、`Production` クラスタの下の `ServerName:port` に Management Console を登録します。

Linux

コマンド ラインから Management Console を開始します。これは RoboServer プログラムの一部で、インストール ディレクトリの下の `bin` ディレクトリにあります。

```
$./RoboServer -p 50000 -MC -mcUrl http://localhost:50080
```

自動起動

別の方法として、RoboServer の開始 で説明されているように、Management Console の自動起動を後で設定する場合、手動で Management Console を起動する代わりに、今すぐ起動を選択することができます。

Management Console の起動後に、ブラウザで開きます。Windows では、スタート メニューにある Management Console の項目をクリックします。すべてのプラットフォームで、ブラウザーを開いて `http://localhost:50080/` に進みます。デフォルトの `admin` ユーザー クレデンシャルを使用して Management Console にログインし、ライセンス条項に同意して、ライセンス キーを含むライセンス情報を入力します。後でライセンス情報を変更する必要がある場合は、[管理] > [ライセンス] で変更できます。

組み込み Management Console の設定

この設定は、RoboServer 設定アプリケーションの [**Management Console**] タブで使用できます。

RoboServerには、Management Console を実行する組み込み Web サーバーが含まれています。Web サーバーは RoboServer の一部ですが、RoboServer が `-MC` オプションで開始される時のみ有効になります。デフォルトでは、Web サーバーはポート 50080 を使用して通信を行うため、Management Console Web インターフェイスは以下のポートで利用できます:

`http://host:50080/`

プロトコルとポート

別のポートで HTTP および HTTPS を介してアクセスできるように Web サーバーを設定することができます。プロトコルが有効になっている場合、ポート番号を選択する必要があります。デフォルトはポート 50080 (HTTP) およびポート 50443 (HTTPS) です。

HTTPS を有効にするには、JKS 形式のサーバー証明書を `Certificates/Web` フォルダの `webserver.keystore` というファイルに保存する必要があります。このファイルは、`C:Users\[User]\AppData\Local\Kofax RPA\[version]` にあります。デフォルト (*changeit*) 以外の証明書パスワードを使用する必要がある場合は、証明書パスワードのフィールドに入力します。

組み込み Management Console に JDBC ドライバーをアップロードできるユーザーを制限することもできます (詳細については、『Kofax RPA のヘルプ』の「データベース ドライバー」を参照してください)。選択肢のうち「許可されていません」では、いずれのユーザーも JDBC ドライバーをアップロードすることはできません。「localhost の管理者」 では、管理ユーザーはローカル コンピューターから Management Consoleにアクセスするとき、ドライバーをアップロードできます。また、「すべてのホストの管理者」では、管理ユーザーが必要に応じて JDBC ドライバーをアップロードできます。

ユーザー管理

Management Console は同じコンピューター (localhost) からだけでなく、他のコンピューターからもアクセスできます。Management Console を使用するポイントの 1 つは、ロボットの実行を調整することです。したがって、通常は多くのクライアントがアクセスできる必要があります。

他のコンピューターから Management Console にアクセスすることによる潜在的なセキュリティ リスクを軽減するために、組み込みモードではユーザー管理がデフォルトで有効になっており、デフォルトの `admin` スーパーユーザー パスワードを使用できます (ユーザー名: `admin`、パスワード: `admin`)。ブラウザから Web インターフェイスにアクセスする場合は、これらのクレデンシャルを使用する必要があります。詳細については、定義済みのユーザー ロールを参照してください。

セキュリティ

RoboServer 設定の [セキュリティ] タブでは、RoboServer にアクセスに認証が必要かどうかに関わらず、RoboServer TLS 構成、一般的なセキュリティ制限、ロギング設定の監査を指定します。

ファイル システムとコマンド ラインのアクセスを許可

RoboServer を有効にして RoboServer が実行するコンピューターでファイルを作成および編集できるようにします。

組み込みの Derby データベースの使用時に、このオプションが選択されていない場合、ロボットはコンピューター上でファイルを作成および編集できます。お使いのネットワーク環境では、MySQL または別のエンタープライズクラス データベースを使用することをお勧めします。

Connector の使用を許可

RoboServer で実行している場合、この設定により、RoboServer が実行されているコンピュータ上のロボットでカスタムの Connector を使用できるようになります。ロボットのカスタム アクション ステップでカスタムの Connector を使用します。詳細については、『Kofax RPAのヘルプ』を参照してください。

Management Console から **JDBC** ドライバーを受け入れる

JDBC ドライバーを Management Console から RoboServer へ分配します。

コマンドのタイムアウト

RoboServer がリモート デバイスのコマンドからの応答をどれくらい待つ必要があるかを指定します。このオプションは、ロボットでのターミナルの自動化および Web サイトのブラウジングにのみ適用されます。

コマンドとは、マウス ボタンをクリックする、アプリケーションを開く、「該当するロケーション」ガードを追加するといったオートメーション デバイスに送信される命令のことです。コマンドが指定した時間内に完了できない場合、サービスによって通知が送信され、ロボットの実行が停止します。

ガード チョイス ステップの場合、この設定はワークフローでのガードの呼び出しに適用されますが、ガードが満たされるまでの待機はこのタイムアウトとは無関係のため、無制限に待機し続ける可能性があります。マウス移動 ステップと抽出ステップの使用時に、同様の状況が発生します。コマンドはフィールドで指定されたタイムアウト以内にデバイスで呼び出される必要がありますが、ロボットはコマンドの完了を最大 240 秒間待機します。

TLS

RPA コンポーネントを正しく動作させるには、以下の TLS 証明書を設定する必要があります。

1. RoboServer は、RoboServer 設定アプリケーションの [証明書] タブの 4 つのプロパティに対応した 4 種類の方法で証明書を適用します。これらは以下の間の通信に関連しています。
   - Management Console と RoboServer
   - RoboServer とオートメーション デバイス
   - RoboServer と API
   - RoboServer と Web サイト

   RoboServer と API および RoboServer と Web サイトの間の通信プロパティは、ロボットが実行の一部として Web サーバーにアクセスする方法に関係しています。
2. Management Console。Management Console では、Tomcat の設定、Management Console 内の設定、およびその他の関連フォルダ内の設定を必要とする API との通信に証明書が必要です。
3. Kapplets サービス

4. Robot File System
5. Desktop Automation サービス

証明書の準備
HTTPS 接続の準備をするには、自己署名証明書を作成する必要があります。JDK パッケージと一緒に提供されている Java keytool ユーティリティを使用すると、次のようにコマンドライン ツールで自己署名証明書を生成できます。コマンドラインで、パス `[JAVA_HOME]\bin` を使用し、`keytool` を見つけます。次に、以下の例を実行します。

```
keytool -genkeypair -alias mc -keyalg rsa -validity 3650 -keystore mc.p12 -storetype
 pkcs12 -ext san=dns:<host machine name>,ip:127.0.0.1,ip:::1,ip:<IP>
```

証明書には、RPA コンポーネントへの接続時に使用する予定の Tomcat Management Console コンピュータのすべてのホスト名/IP アドレスが含まれている必要があります。共通名 (`cn`) パラメータは、Tomcat Management Console コンピュータのホスト名と一致させる必要があります。

異なる名前を持つ 2 つのキーストアを作成する必要があります。1 つは Tomcat に Management Console を展開するためのキーストアで、もう 1 つは Management Console を使用して RPA の他の部分と通信するためのキーストアです (`communication.p12`)。同じオプションを両方の証明書に使用できます。

次のコマンドを使用して、キーストアから証明書を抽出します。

```
keytool -exportcert -alias mc -keystore mc.p12 -file mc.cer
```

```
keytool -exportcert -alias communication -keystore communication.p12 -file
 communication.cer
```

Management Console TLS の設定

RPA の各部分で SSL を安定して使用するために、以下の手順を実行してください。

1. **[Tomcat]** > **[Conf]** にある server.xml ファイルを次のように編集します。

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
 maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="<path to the certificate/
mc.p12>" certificateKeystorePassword="<your password>" type="RSA"
 certificateKeyAlias="mc"/>
    </SSLHostConfig>
</Connector>
```

2. 接続を常に TLS にリダイレクトする必要がある場合 (つまり、HTTP 接続を許可しない場合) は、**[Tomcat]** > **[Conf]** にある web.xml ファイルに次のコマンドを追加します。

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Protected Context</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

SSL Tomcat の設定の詳細については、Apache Tomcat の Web サイトに掲載されているこちらの記事を参照してください。

3. Management Console の開始: `https://localhost:8443/ManagementConsole/`。
4. Management Console を起動し、**[SSL を使用]** を選択して新しいクラスタを作成します。続いて TLS の設定に進み、通信の残りの半分をセットアップします。
5. 必要に応じて、RoboServer との RQL 暗号化通信のために、`[Tomcat_home]/webapps/[アプリケーション名 (ManagementConsole)]/WEB_INF` フォルダにある certs.xml ファイルで次のように指定します。

```
<property name="privateCertificateLocation" value="/WEB-INF/communication.p12"/>
<property name="privateCertPassword" value="changeit"/>
```

RoboServer TLS の設定

Kofax RPA は、オートメーション デバイスと RoboServer、Kapplets、ロボット ファイル システム、または Design Studio の間に TLS 通信を設定する手段を提供します。通信には、通信を暗号化するための証明書が使用されます。暗号化では、接続の保護にはパブリック - プライベート キー構造が使用されます。

RoboServer 設定アプリケーションで、 [セキュリティ] タブの [TLS 構成の設定] で、組み込みの証明書セットを使用するか、他の証明書を指定するかを指定できます。

- Kofax RPA 証明書を使用するには、[デフォルトを使用] を選択します。
- 他の証明書を使用するには、[デフォルトを使用] をクリアにして、対応するフィールドで秘密鍵と公開鍵、および信頼済み証明書フォルダへのパスを指定します。

詳細については、『*Kofax RPA* のヘルプ』の「「TLS コミュニケーションを使用」」を参照してください。

RoboServer と **Management Console** のクラスタとの間の **SSL** 接続の確立

1. Management Console を起動し、**[SSL を使用]** を選択して新しいクラスタを作成します。
2. [**RoboServer** 設定] アプリケーションで、**[証明書]** > **[API]** の順に移動し、[**API** クライアント証明書を検証] を選択して、Management Console からの接続のみを受け入れます。
3. [API サーバー証明書] に communication.p12 証明書を追加します。
4. 指定された Management Console certs.xml のキーストア (communication.p12) の communication.cer 証明書を次の場所に配置します。
   - Linux の場合: `[USER_HOME]/.Kofax RPA/[version]/Certificates/API/TrustedClients`
   - Windows の場合: `[User]\AppData\Local\Kofax RPA\[バージョン]\Certificates\API\TrustedClients`

   証明書の `cn` 属性は、Management Console の解決されたホスト名と一致している必要があります。
5. RoboServer で使用される JRE キーストアに communication.cer を追加します。

```
keytool -import -alias communication -keystore "C:\Program Files\[JAVA_HOME]\lib
\security\cacerts"       -trustcacerts -file C:\<path>\communication.cer
```

6. コマンド プロンプト ウィンドウを使用して RoboServer を起動します。

```
RoboServer -service ssl:50001 -mcUrl https:\\<hostname or IP> -ss <MC Shared
 Secret> -cluster SSL
```

証明書の検証を有効にするか、Management Console で証明書を使用して RoboServer がその有効性を確認できるようにするには、設定を変更し、`ManagementConsole\WEB-INF` にある certs.xml ファイルでキーストア (communication.p12) へのパスを指定します。

認証のリクエスト

このオプションは非推奨となりました。

RoboServer を不正アクセスから守るため、認証をオンにできます。この認証は、サービスとして開始された RoboServer、またはコマンド ラインから開始された RoboServer を含む、Kofax RPA インストールから実行されたすべての RoboServer に影響を与えます。

認証をオンにするには、RoboServer 設定の [RoboServer 認証が必要] 項目のチェック ボックスを選択します。認証をオンにした状態で RoboServer でロボットを実行するには、追加ボタンをクリックしてユーザーを追加する必要があります。その後、ユーザーのリストに表示されるユーザー名などのユーザーに関する情報を入力できます。

ユーザーは次の表のプロパティを使用して設定します。

ユーザーのプロパティ

プロパティ	説明
ユーザー名	ユーザーが RoboServer にアクセスするときに使用するユーザー名です。
パスワード (パスワード ハッシュ)	ユーザーが RoboServer にアクセスするときに使用するパスワードです。
コメント	ここで、ユーザーに関するコメントを書くことができます。
ロボットを開始	ユーザーは RoboServer でロボットを起動できます。
ロボットを停止	ユーザーは RoboServer でロボットを停止できます。
RoboServer をシャットダウン	ユーザーが Management Console から RoboServer をシャットダウンできるようにします。

Kapplets サービスの TLS 構成

Kapplets から Management Console への接続の TLS 構成を確立するには、Kapplets の展開と同じ設定を使用しますが、次の違いがあります。

1. kapplets.xml の Management Console への SSL パスを使用します。

```
<Environment name="kapplets.services.mc.connection.url" value="https://<
 hostname>8443/ManagementConsole/" type="java.lang.String" override="false"/>
```

2. Management Console の証明書を、Kapplet Service Tomcat が使用する JRE キーストア (`[JAVA_HOME]\...\lib\security\cacerts`) に追加します。これを行うには、次のコマンドを使用します。

```
keytool -import -alias mc -keystore "C:\Program Files\[JAVA_HOME]\lib\security
\cacerts" -trustcacerts -file C:\<path>\mc.cer
```

ロボット ファイル システムの設定

次の手順を使用して、Robot File System (RFS) の TLS 接続を設定します。

1. Management Console についての説明と同じ方法で、SSL Tomcat の設定を使用します。
2. `web.xml` ファイルで、次のパラメータを設定します。
   - Management Console の TLS アドレスを設定します。`https://<hostmachine>:8443/ManagementConsole`。
   - Management Console を使用して認証するように RFS 共有シークレットを設定します。共有シークレットをファイルに保存し、このファイルへのパスを `shared-secret-file` パラメータに設定するか、プレーン テキストの共有シークレットを `shared-secret` パラメータに貼り付けます。
3. Tomcat が使用する Java 環境の `cacerts` キーストアに `mc.cer` を追加します。たとえば、`[JAVA_HOME]\lib\security\cacerts` のようになります。
4. Management Console の [一般] タブの RFS サーバー設定で、RFS サーバーの TLS アドレスを設定します。`https://[ホストマシン]:8443/rfs`
5. Management Console および RoboServer で使用されている JRE キーストア (`C:\Program Files\Kofax RPA 11.5.0.0\jre\lib\security\cacerts`) に `mc.cer` を追加します。これを行うには、次のコマンドを使用します。

```
keytool -import -alias mc -keystore "Kofax RPA 11.5.0.0\jre\lib\security\cacerts"
 -trustcacerts -file C:\<path>\mc.cer
```

Management Console と RoboServer が異なる Tomcat サーバーを使用している場合は、Robot File System 用のキーペアを生成します。それ以外の場合は、Management Console 証明書が Java の `cacerts` にインポートされていることを確認します。

Robot File System で自己署名証明書またはプライベート ルート CA で署名された証明書を使用する場合は、RoboServer、Design Studio、または Desktop Automation サービスを実行するすべてのシステムで CA 証明書を設定します。その結果、Robot File System が使用可能になります。これを行うには、各システムで次の手順を完了させます。

1. システムで RoboServer、Design Studio、または Desktop Automation サービスを実行するために使用されるアカウントを特定します。
2. `certificate.cer` ファイルをシステムにコピーします。
3. ステップ 1 で特定したアカウントに環境変数 `NODE_EXTRA_CA_CERTS` を追加するようにシステムを設定します。
   - `NODE_EXTRA_CA_CERTS` 環境変数がアカウントに対してすでに定義されている場合は、参照するファイルを見つけて、`certificate.cer` ファイルの内容をこのファイルに追加します。
   - `NODE_EXTRA_CA_CERTS` 環境変数が定義されていない場合は、定義を追加し、その値を `certificate.cer` ファイルのフル パスに設定します。
4. ステップ 1 で特定されたアカウントに、`NODE_EXTRA_CA_CERTS` によって参照されるファイルへの読み取りアクセス権があることを確認してください。

Desktop Automation サービス設定

Desktop Automation サービスをすべての Kofax RPA コンポーネントで利用できるようにするには、次の手順を完了させます。

1. **[Desktop Automation** サービス**]** > **[Management Console** の設定**]** の Management Console への SSL パスを使用します。 `https://[ホスト名]:8443/ManagementConsole/`.
2. Management Console の証明書 `communication.cer` をダウンロードし、Desktop Automation サービスの [**CA** ファイル] フィールドに追加します。

ルート証明書を Google Chrome ブラウザから Management Console にファイルとして保存する別の方法もあります。これを行うには、次の手順を完了させます。

1. アドレス バーのロック アイコンを右クリックし、[証明書] をクリックします。
2. [証明のパス] タブで、一番上のルート証明書を選択し、[証明書の表示] をクリックします。
3. [詳細] タブで [ファイルにコピー] をクリックし、ウィザードに従いルート証明書を base-64 エンコード X.509 証明書としてエクスポートします。

JMX サーバーの設定

組み込み JMX サーバーを使用して、JConsole などのツールを通じて実行中の RoboServer を監視します。RoboServer コマンド ラインで引数を指定して、JConsole を有効にします。

センシティブ情報のロボット入力を隠す

[入力を表示] オプションにより、ロボットの入力パラメーターを管理インターフェイスに表示するかどうかを制御します。これにより、パスワードなどのセキュリティ上の機密情報を隠すことができます。

JMX サーバー アクセス

デフォルトでは、サーバー上の正しいポートにアクセスできるすべてのクライアントが JMX サーバーにアクセスできます。[パスワード使用] オプションを選択すると、選択したユーザー名とパスワードが接続時に必要になります。

ハートビート通知

0 より大きい間隔 (秒単位) が指定されている場合、RoboServer がクエリを実行して応答している限り、JMX サーバーは、指定された間隔でハートビート通知を送信します。

デフォルトの RoboServer プロジェクト

[RoboServer 設定] の [プロジェクト] タブにある、デフォルトの RoboServer プロジェクト フォルダの場所を設定できます。デフォルトでは、このフォルダはインストール プロセス中に作成したデフォルトのロボット プロジェクトに設定されています。ロボット プロジェクトの詳細については、『Kofax RPA のヘルプ』の「Design Studio」の章を参照してください。

RoboServer デフォルトのプロジェクトは API によってのみ使用されます。API を使用してロボットを実行する場合、タイプ、スニペット、またはその他のリソースへの参照は、デフォルトのプロジェクトを調べることで解決できます。

RAM 割り当てを変更する

インストール時に、それぞれ Kofax RPA アプリケーションは、使用可能な最大量の RAM で設定されます。多くの場合、この量は通常の作業には十分ですが、多くのロボットを RoboServer で並行して実行する場合、または一部のロボットが多くの RAM を使用している場合は、割り当てを増やす必要があります。

インストール フォルダの `bin` サブフォルダにある `.conf` ファイルを編集することにより、任意のアプリケーションの割り当てを変更できます。

RoboServer の場合は、`bin/RoboServer.conf` を編集します。

Design Studio の場合は、`bin/DesignStudio.conf` を編集します。

ファイルを編集するには、次のステップを実行します。

1. 対応する `.conf` ファイルをテキスト エディターで開きます。
2. `wrapper.java.maxmemory` パラメーターを含む行を見つけます。
3. 行のコメントを解除し（先頭の # を削除）、その値を編集します。
   たとえば、RoboServer で最大 4GB の RAM を使用するには、以下を入力します。`wrapper.java.maxmemory=4096`

> `.conf` ファイルに `wrapper.java.maxmemory` 行が含まれていない場合は、行全体をファイルに追加します。`.conf` ファイルは、Windows 管理者など、Kofax RPA をインストールしたユーザーのみが編集できます。

RoboServer サービスの開始のトラブルシューティング

サービスが開始されない場合、Windows イベント ログで RoboServer メッセージを探します。`wrapper.syslog.loglevel=INFO` 引数を使用してサービスをインストールしたことを確認してください。詳細については、『Kofax RPA インストール ガイド』の「Kofax RPA の初期設定」を参照してください。

第 2 章

Tomcat Management Console

本番環境では、Management Console を通常の Web アプリケーションとしてスタンドアロンの Tomcat Web サーバーに展開することを強くお勧めします。

セットアップで Management Console へアクセスする必要がある場合、企業イントラネットの外部で、Tomcat サーバーと連携するように TLSv1.0 (またはそれ以降のバージョンのTLS) をセットアップします。

次の表に、機能セットの違いを示します。

Management Console 機能と設定

機能	組み込み	スタンドアロン **J2SE Web** コンテナー
認証	Management Console で定義された単一の `admin` スーパーユーザーとユーザー Management Console 管理者によって管理されるユーザーとロール	Management Console 管理者によって管理されるユーザーとロール Active Directory または他の LDAP プロバイダーを介した役割ベースのセキュリティ。 CA Single Sign-On を使用したシングル サインオン。
Management Console データストア	組み込み Derby データベース	コンテナ管理のデータソース (サポートされているプラットフォーム)

Derby JDBC ドライバーは、エンタープライズ Management Console とともには分配されません。Derby JDBC ドライバのダウンロード情報については、Apache Derby の Web サイトを参照してください。MySQL または他のエンタープライズクラスのデータベースをエンタープライズ Management Console で使用することをお勧めします。

組み込み Management Console の設定手順は、Kofax RPA のオンライン ヘルプで確認することができます。組み込み Management Console を起動するには、「Management Console」セクションの「Management Console の起動」を参照してください。

Tomcat の展開

この章では、スタンドアロンの J2SE Web コンテナ上で、Management Console を手動でインストールする方法について詳しく説明します。このガイドでは、Tomcat を選択しました。J2SE Web コンテナで

サポートされている Java バージョンについては、『Kofax RPA RPA 技術仕様』ドキュメントを参照してください。Oracle Java SE のダウンロード サイトにアクセスして、最新の Java リリースをダウンロードしてください。

セットアップで Management Console へアクセスする必要がある場合、企業イントラネットの外部で、Tomcat サーバーと連携するように TLSv1.0 (またはそれ以降のバージョンのTLS) をセットアップします。

Tomcat での Management Console のインストール

前提条件

- Oracle の Web サイトから最新の Java アップデートをインストールします: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Apache の Web サイト https://tomcat.apache.org から Tomcat をダウンロードします。
  - Tomcat をインストールし、ユーザー パスワードを設定します。

インストール

1. Kofax RPA インストーラーをすべてダウンロードし、インストールを続行します。インストール中に **[Management Console WAR]** オプションを選択します。
2. Kofax RPA インストール先の `WebApps` フォルダから、Tomcat サーバー上の `webapps` フォルダに `ManagementConsole.war` ファイルをコピーし、.war ファイルを設定します。
3. Tomcat サーバーの `conf/Catalina/localhost/` に `ManagementConsole.xml` という Tomcat コンテキスト ファイルを作成します。詳細については Tomcat コンテキスト ファイルの作成 を参照してください。
4. Tomcat で `webapps/manager/WEB-INF/web.xml` ファイルを編集します。
   - アプリケーションをアップロードするには、以下を編集します。

```
<multipart-config>
<!-- 150MB max -->
 <max-file-size>152428800</max-file-size>
 <max-request-size>152428800</max-request-size>
 <file-size-threshold>0</file-size-threshold>
</multipart-config>
```

   - ネットワークで HTTPS プロトコル を強制的に使用する必要がある場合は、`web.xml` で `org.apache.catalina.filters.HttpHeaderSecurityFilter` classを使用して、Tomcat サーバー上の HTTP Strict Transport Security Header (HSTS ヘッダー) を有効にします。詳細については、Apache Tomcat のドキュメントを参照してください。
   - Management Console は特定の class の攻撃の検出と緩和に役立つ追加の防御層としてコンテンツ セキュリティ ポリシーを実装します。すべての URL にマップされる `SecurityHeadersFilter` フィルターは、`web.xml` で宣言および設定します。ヘッダーの値を設定するには、`contentSecurityPolicy` 変数を設定します。これは、`web.xml` ファイルでのヘッダー フィルタリングの設定例です:

```
    <filter>
        <filter-name>SecurityHeadersFilter</filter-name>
        <filter-
class>com.kapowtech.scheduler.server.servlet.SecurityHeadersFilter</filter-
class>
        <init-param>
```

```
            <param-name>contentSecurityPolicy</param-name>
            <param-value>default-src 'self' data: blob: 'unsafe-inline' 'unsafe-
eval'</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>SecurityHeadersFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

ヘッダー フィルタリングが不要な場合は、フィルター コードと対応するフィルター マッピングを無効にするか削除します。可能なオプションとヘッダー フィルタリングの詳細については、Web でコンテンツ セキュリティ ポリシーを検索してください。

5. Tomcat を起動します。
6. ライセンス情報の入力。

ManagementConsole.war の設定

Management Console アプリケーションは、`ManagementConsole.war` という名前の Web アプリケーション アーカイブ (WAR ファイル) の形式で提供されます。これは、Kofax RPA インストール フォルダの `/ WebApps` フォルダにあります。

Kofax RPA 付属の `ManagementConsole.war` のバージョンは、RoboServer 内部に組み込まれて実行するように設定されています。Tomcat でスタンドアロン アプリケーションとしてデプロイする前に、環境に合わせた再設定が必要になる場合があります。

WAR ファイルは、圧縮された zip ファイルを使用して圧縮されます。構設定ファイルにアクセスするには、zip ファイルを抽出する必要があります。設定ファイルが更新されたら、`ManagementConsole.war` を再圧縮して Tomcat サーバーにデプロイします。

以下の表には、解凍された `ManagementConsole.war` のルートに関連する設定ファイルのリストが含まれています。

設定ファイル

ファイル	設定	注意
`WEB-INF/Configuration.xml`	クラスタリング、パスワード暗号化、REST プラグイン	以前のバージョンからファイルをコピーする場合、Management Console を起動すると自動的にアップグレードされます
`WEB-INF/login.xml`	管理者とユーザー、これは LDAP と統合する場所です	
`WEB-INF/classes/log4j2.properties`	アプリケーションのログ	
`WEB-INF/spring/authentication.xml`	ユーザー認証	
`WEB-INF/roles.xml`	Management Console で組み込みのロール。	

Spring 設定ファイル

`Configuration.xml`、`login.xml`、`roles.xml`、および `authentication.xml` はすべて Spring 設定ファイル (www.springsource.org) であり、ここで概説したものと同じ一般的な構文を共有しています。

Spring は一連の Bean を介して設定され、各 Bean にはアプリケーション内のコードを設定するプロパティがあります。一般的な構文:

```
<bean id="id" class="SomeClass">
   <property name="myName" value="myValue"/>
</bean>
```

ファイル	設定
`id="id"`	Bean の ID は、アプリケーションが Bean を参照するために使用する内部ハンドルです。Bean の名前とも呼ばれます。
`class="SomeClass"`	Classは、Bean が設定するコード コンポーネントを識別します。
`<property name="myName" value="myValue"/>`	名前が `myName` で、値が `myValue` のプロパティを定義します。これにより、class 属性で定義されたコード コンポーネントのプロパティが設定されます。

Kofax RPA では、Management Console の [管理] メニューの [ユーザーおよびグループ] セクションを使用してユーザー管理を実行します。Kofax RPA のエンタープライズ バージョンでは、ユーザー管理はデフォルトで有効になっています。以前のインストールからのユーザー作成は、Kofax RPA バックアップ機能を使用して実行できます。LDAP 統合と SAML 統合の場合は、login.xml ファイルも編集する必要があります。

トラブルシューティング

インストール中に問題が発生した場合は、Tomcat インストール先の `/logs` フォルダで Tomcat ログを確認する必要があります。設定プロセス中に、コマンド ライン ウィンドウにエラー メッセージが直接出力されるため、多くの場合、コマンド ラインから Tomcat を実行する方が簡単です。

新しいデータベースの作成

Management Console と Kapplets は、Management Console、Kapplets、およびそのデータを含むデータベース間に高帯域幅で低遅延のネットワークを必要とします。これらは同じリージョンまたは近くに存在している必要があります。Management Console と Kapplets が遠く離れた場所に配置されている場合にデータベースや相互にアクセスすると、どちらもデッドロックやタイムアウトが発生する可能性があります。パラメータを使用してタイムアウトを延長し、更新頻度を減らすと、サービス品質が低下する可能性があります。

データの不整合や Management Console と Kapplets のロックを防ぐために、データベースのメンテナンスを実行している間は Management Console と Kapplets を停止します。

Management Console が使用するテーブル用に新しいデータベースを作成することを強くお勧めします。データベースには 2 つの要件があります。

- Unicode サポート
- 大文字と小文字の区別

デンマーク語の Æ、ドイツ語の ß、キリル文字の Ё などの非 ASCII 文字がロボットに入力されることもあるため、Unicode のサポートが必要です。この入力はデータベースに保存され、Unicode のサポートがない場合、これらの文字は正しく保存されないことがあります。

`a.robot` という口ボットの名前と `A.robot` という別の名前をアップロードすることが可能であるため、大文字と小文字の区別が必要です。大文字と小文字の区別を行わない場合、別の名前のアップロードによって前の名前が上書きされます。

製品ドキュメントの推奨事項に従って、Kofax RPA 製品データベースを作成して保守運用してください。データベースの変更またはカスタマイズを検討している場合は、Kofax に確認したうえで続行するようにしてください。確認せずに実行した場合、結果が予測不能になり、ソフトウェアが動作しなくなる可能性があります。

データベース サーバーは、Unicode の制御と大文字と小文字を区別の制御を異なる方法で行います。次のリストには、サポートされているデータベース システムの推奨事項が含まれています。

Unicode サポートと大文字と小文字を区別する場合の推奨事項

データベース	推奨事項
MySQL	Utf8mb4_bin 照合を使用してデータベースを作成します。 `CREATE DATABASE KAPOW_MC COLLATE utf8mb4_bin`
Oracle	NVARCHAR2、NCLOB タイプは Unicode に使用します。大文字と小文字を区別する場合は、NLS_COMP が BINARY に設定されていることを確認してください。
Microsoft SQL Server	NVARCHAR、NTEXT タイプは Unicode に使用します。大文字と小文字を区別する場合は、Latin1_General_100_BIN2 などの大文字と小文字の区別がある照合でデータベースを作成します。 `CREATE DATABASE KAPOW_MC COLLATE Latin1_General_100_BIN2` 詳細については、Tomcat コンテキスト ファイルの作成 で 「Windows 統合セキュリティの Microsoft SQL Server の設定」を参照してください。
PostgreSQL	CODESET UTF-8 を使用してデータベースを作成します。 `CREATE DATABASE KAPOW_MC ENCODING 'UTF8'`

Management Console で使用されるテーブルは次の 3 つのカテゴリにグループ化できます: プラットフォーム テーブル、ロギング テーブル、およびデータビュー テーブル。ロギング テーブルおよびデータビュー テーブルが RoboServer と共有されている間は、アップロードされた口ボットおよびそのスケ

ジューリング情報などの Management Console 専用の情報がプラット フォーム テーブルに保持されます。

サポートされているデータベースとバージョンのリストについては、『Kofax RPA Technical Specifications』(Kofax RPA 技術仕様) ドキュメントを参照してください。

ユーザー権限

Management Console を開始すると、必要なプラットフォーム テーブルとロギング テーブルを自動的に作成しようとします (RoboServer によってまだ作成されていない場合) 。つまり、データベースへのアクセスに使用するユーザー アカウントには、バックアップを復元するための CREATE TEMPORARY TABLES 特権だけでなく、CREATE TABLE および ALTER TABLE 特権も必要です。Oracle ユーザーには、CREATE SEQUENCE 特権も必要です。これが不可能な場合は、以下のスクリプトを使用してテーブルを作成するようデータベース管理者に依頼できます。

さらに、システムが正常に機能するためには、ユーザーによる選択、挿入、更新、削除が許可されている必要があります。

Management Console テーブルの SQL スクリプト

SQL スクリプトは、Kofax RPA in the `documentation\sql` ディレクトリのコピーに含まれています。詳細については、Kofax RPA テーブルの SQL スクリプト を参照してください。

Management Console は Quartz と呼ばれるサードパーティのスケジューリング コンポーネントを使用します。Quartz にも、その他のプラットフォーム テーブルに存在する必要のある多数のテーブルが必要です。これらのテーブルは Management Console が起動するときに自動で作成されますが、Kofax RPA テーブルの SQL スクリプト のスクリプトを使用して手動で作成することもできます。

Tomcat コンテキスト ファイルの作成

エンタープライズ環境では、多くの場合、データベースへのアクセスはデータソースを介して実行されます。このセクションでは、ローカル MySQL データベース サーバーに接続するデータソースで Tomcat を設定する方法を示します。

Tomcat では、データソースはアプリケーション コンテキスト内で定義されています。コンテキストは、組み込みまたはアプリケーションの外部で宣言できます。コンテキストが組み込まれると、ファイル `context.xml` で定義されます。このファイルは、META-INF フォルダの WAR ファイル内に配置する必要があります。外部で宣言する場合、ファイルは Tomcat の `/conf/Catalina/localhost` フォルダにあり、ファイルの名前は `ManagementConsole.xml` (デプロイされた WAR ファイルと同じ名前) である必要があります。Tomcat は単一のデプロイメント ユニットを提供するため、組み込みコンテキストでデプロイすることが推奨されますが、このガイドでは外部コンテキスト定義を使用します。これにより、ファイルの変更が容易になります。設定を調整したら、コンテキスト ファイルを組み込み、WAR ファイルを運用環境に展開できます。

プラットフォーム データソースを追加

`conf/Catalina/localhost` にある Tomcat の `ManagementConsole.xml` ファイルを作成し、次のコンテンツを追加します。

次の抜粋はあくまで例であり、実際には他の設定が含まれる場合もあります。

MySQL データベースへの接続を作成するには:

```
<Context useHttpOnly="true">
         <!-- Default set of monitored resources -->
         <WatchedResource>WEB-INF/web.xml</WatchedResource>

         <Resource name="jdbc/kapow/platform" auth="Container"
             type="javax.sql.DataSource"
             maxTotal="100" maxIdle="30" maxWaitMillis="-1"
             factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
             validationQuery="/* ping */" testOnBorrow="true"
             username="MyUser" password="MyPassword"
             driverClassName="com.mysql.jdbc.Driver"
             url="jdbc:mysql://localhost:3306/KAPOW_MC?
useUnicode=yes&amp;characterEncoding=UTF-8&amp;rewriteBatchedStatements=true"/>

     </Context>
```

PostgreSQL への接続を作成するには:

```
<Context useHttpOnly="true">
         <!-- Default set of monitored resources -->
         <WatchedResource>WEB-INF/web.xml</WatchedResource>

         <Resource name="jdbc/kapow/platform" auth="Container"
             type="javax.sql.DataSource"
             maxActive="100" maxIdle="30" maxWait="-1"
             validationQuery="SELECT 1" testOnBorrow="false"
             username="username" password="password"
             driverClassName="org.postgresql.Driver"
             url="jdbc:postgresql://<URL>:<PORT>/<Database>"/>
</Context>
```

Oracle への接続を作成するには:

```
<Resource name="jdbc/kapow/platform" auth="Container" type="javax.sql.DataSource"
   maxTotal="100" maxIdle="30" maxWaitMillis="10000"
   validationQuery="SELECT 1 FROM DUAL" testOnBorrow="false"
   username="user" password="password" driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@<IP>:<Port>:DB_name"/>
```

上記の `url` パラメーターは JDBC URL です。ユーザー名とパスワードの属性は、データベースへの接続時に使用される接続プールを作成するために Tomcat によって使用されます。

データソースは、他のデータベースに対して異なる方法で定義されています。たとえば、Microsoft SQL Server を使用している場合、上記の関連する 3 行は次のようになります:

```
username="MyUser" password="MyPassword"
            driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
            validationQuery="SELECT 1" testOnBorrow="true"
            url="jdbc:sqlserver://localhost:1433;DatabaseName=MyDbName"/>
```

`jdbc:mysql://localhost:3306/KAPOW_MC?useUnicode=yes&characterEncoding=UTF-8` の URL は、ローカル MySQL にある KAPOW_MC という名前のデータベースを参照します。MySQL の場合、`?useUnicode=yes&characterEncoding=UTF-8` をすべての接続文字列に追加することをお勧めします。追加しない場合、JDBC ドライバーは中国語、日本語、およびその他の 3 バイトの utf-8 文字を正し

く処理できず、コンテキスト xml ファイルには「&」を直接含めることはできないため、「&」としてエンコードする必要があります。

`rewriteBatchedStatements=true` は、MySQL JDBC ドライバーのバッチ挿入/更新を指示し、kapplet ロボットに対する挿入パフォーマンスを改善します。

`driverClassName` パラメーターは、使用する JDBC ドライバーを制御します。各データベース ベンダーは、データベース用の JDBC ドライバーを提供しています (ダウンロードが必要) 。JDBC ドライバー (通常は単一の `.jar` ファイル) は、Tomcat で `/lib` フォルダにコピーする必要があります。

`validationQuery` は、接続プールから取得した接続が有効であることを確認するために Tomcat によって使用されます (データベース サーバーが接続を閉じた可能性があるため)。検証クエリは軽量で、データベース サーバー上のリソースをほとんど使用しません。このリストには、サポートされているデータベースの検証クエリが含まれています。

検証クエリ

データベース	クエリ
MySQL	/ * ping * /
Microsoft SQL Server	SELECT 1
Oracle	デュアルから 1 を選択
PostgreSQL	SELECT 1

IBM DB2 は Management Console データベースとしてはサポートされていませんが、LogDB やロボットで利用可能なデータベース向けにユーザーのデータベース タイプとして使用できます。

MySQL JDBC ドライバーが特別な軽量の /* ping */ 'request' をサポートしていることに注意してください。詳細は、「JConnector Manual」(JConnector マニュアル) のセクション 6.1 を確認してください。

コンテキストの設定とデータソースの詳細については、「JNDI リソース HOW-TO」および「JNDI データソース HOW-TO」を参照してください。

Windows 統合セキュリティでの **Microsoft SQL Server** の設定

Microsoft SQL Server と Windows 統合セキュリティを使用している場合、Design Studio と RoboServer を実行しているコンピューターは以下の条件に従う必要があります:

- Windows で実行する
- データベースへのアクセスが許可されたユーザー アカウントで実行されている
- この部分で後述するように、JDBC ドライバーが手動でインストールされている

Windows 統合セキュリティで Microsoft SQL Server を設定するには、次の操作を実行します。

- JDBC ドライバーを Tomcat のフォルダにインストールします。JAR ファイルを `lib` フォルダにコピーして、DLL ファイルを `nativelib` フォルダにコピーします。
- Management Console は JDBC ドライバーを分配できないため、Jar ファイルを Management Console にアップロードしないでください。
- `ManagementConsole.xml` (または `context.xml`) と Management Console のデータベース タイプの定義、および Design Studio のローカルの定義で、接続 URL に `;integratedSecurity=true`

が追加されていることを確認してください。『Kofax RPA のヘルプ』にある「データベース タイプの追加」を参照してください。

これで、Tomcat サーバーを起動する準備が整いました。

Tomcat の起動

Tomcat サーバーを起動し、アプリケーションがデプロイされるまで数秒待機します。

http://localhost:8080/ManagementConsole を開きます。ログイン画面が表示されます。

ユーザー名に `admin`、パスワードに `admin` と入力し、**[ログイン]** をクリックします。

ライセンス情報の入力

ログインすると、ライセンス ウィンドウが表示されます。

Kofax RPA ライセンス情報を入力して、[保存] をクリックします。ライセンス キーによって有効になる機能を示すダイアログ ボックスが表示されます。

定義済みのユーザー ロール

Management Console には、ユーザーに割り当てることができる組み込み済みロールが用意されています。ロールはユーザーまたはサービスにマッピングされます。ユーザー権限は、ユーザーが所属するセキュリティ グループに割り当てられたロールに基づいて計算されます。組み込み済みロールを変更したり、その他のロールを追加したりすることができます。組み込み済みロールは、`roles.xml`ファイルで定義されます。詳細については プロジェクト権限 を参照してください。

組み込み済みロール

Management Console によって提供される組み込みロールを以下に示します。

サービス ロールは API アプリケーションでのみ使用することを目的としているため、ブラウザの Management Console への対話型ログインには使用しないでください。

- プロジェクト管理者 (Project Administrator) : 1 つまたは複数のプロジェクトを管理し、これらのプロジェクトのグループにロールを割り当てる権限を持ちます。このロールにより、RoboServer およびクラスタの設定を変更せずに表示する権限が付与されます。プロジェクト管理者は、RPA 管理者グループのメンバーではありません (詳細については、このセクションの後半を参照してください)。
- 開発者: リポジトリ内のすべてのリソース タイプをアップロード、ダウンロード、表示する権限があります。このロールにより、スケジュールの作成、編集、削除、ロボットの実行、実行ログとクラスタの表示を行う権限が付与されます。
- 閲覧者: [スケジュール]、[リポジトリ]、[データ ビュー]、[ログ ビュー]、および一部の [設定] を表示できます。このロールにより、[管理者] セクションの下で制限付きアクセスが付与されますが、ロボットを変更したり実行したりする権限は付与されません。
- **API** (サービス ロール): リポジトリ API を使用してリポジトリの読み取りおよびリポジトリへの書き込みを行う権限を付与します。このロールでは、REST を使用したロボットの実行は許可されませんが、RQL を使用したロボットの実行は許可されます。

- サービス認証 **API** (サービス ロール): リポジトリ API を使用して、リポジトリの読み取りおよびリポジトリへの書き込みを行います。ユーザーは OAuth 認証方法を使用してログインします。
- **RoboServer** (サービス ロール): リポジトリからの読み取りのみができます。このロールは、クラスタにアクセスする場合、リポジトリ アイテムを取得する場合、およびパスワード ストアからパスワードを要求する場合に RoboServer で使用されます。
- **Kapplet** 管理者: Kapplets から Management Console のプロジェクトへの読み取り/書き込みアクセス権を付与します。Kapplets では、このロールを持つユーザーは、Kapplet を管理し、これらのテンプレートに必要なロボットを含むプロジェクトの Kapplet テンプレートを作成および管理できます。

  このロールを持つユーザーは、他の権限がない場合、Management Console にアクセスできません。

  詳細については、『Kofax RPA のヘルプ』の「Kapplet ユーザー管理」と「ユーザーとユーザー グループ」を参照してください。
- **Kapplet** ユーザー: Kapplets から Management Console のプロジェクトへの読み取り専用アクセス権を付与します。Kapplets では、このロールを持つユーザーは、アクセス権のあるプロジェクトに属するロボットの Kapplets のみを表示および実行できます。

  このロールを持つユーザーは、他の権限がない場合、Management Console にアクセスできません。

  詳細については、『Kofax RPA のヘルプ』の「Kapplet ユーザー管理」と「ユーザーとユーザー グループ」を参照してください。
- **Kapplet** サービス ユーザー (サービス ロール): リポジトリからの読み取りのみができます。これは特定のプロジェクトで利用できるロボット、タイプ、スニペット、リソースに関する情報の取得にのみ使用されるロールなので、Kapplets と Management Console の間の通信のみを目的に使用します。このロールはすべての Management Console プロジェクトに自動的に適用されます。
- パスワード ストア クライアント (サービス ロール): このアドオン ロールにより、Management Console のパスワード ストアへのアクセス権を付与します。このロールは、開発者ロールと同様に、他のロールの上位に提供されます。
- **DAS** クライアント ユーザー (サービス ロール): このロールを持つユーザーはリモートの Desktop Automation サービス (DAS) クライアント用に作成され、DAS API へのアクセス権のみを持ちます。DAS クライアント ユーザーには DAS を Management Console に提示し、DAS 構成を取得する権限があります。
- **VCS** サービス ユーザー (サービス ロール): シンクロナイザーに特別な権限セットを付与します。このロールにより、リソースを追加、編集、削除する権限を付与します。他のユーザーの代理で展開を実行してバージョン コントロール サービスで「deployer」機能を使用できるのはこのロールのみです。
- **Process Discovery** クライアント (サービス ロール): このロールを使用すると、Process Discovery コンポーネントを Management Console と連携させることができます。
- **KTA** クライアント**:** (サービス ロール): このロールを使用すると、KTA コンポーネントを Management Console と連携させることができます。

組み込み **admin** ユーザー

admin は、すべてにアクセスできるスーパーユーザーです。admin は RPA 管理者グループのメンバーではありません。また、どのグループのメンバーにも属しません。 このユーザーは、デフォルトの admin ユーザー パスワードを使用できます (ユーザー名 - `admin`、パスワード - `admin`)。『Kofax RPA のヘルプ「」』の「ユーザーおよびグループ」の説明に従って、管理ユーザーのパスワードを変更できます。

また、`initialAdminUser` プロパティと `initialAdminPassword` プロパティを設定することで、`WEB-INF/login.xml` ファイル内の初期の admin ユーザー名とパスワードを変更できます。データベースが空であるか admin ユーザーが存在せず、システムがローカル認証を使用するように設定されているか、(LDAP などの場合) `createAdmin` プロパティが `true` に設定されている場合は、admin ユー

ザーが作成されます。admin ユーザーがデータベースにすでに含まれている場合、そのユーザーは作成されません。

LDAP 統合セットアップでは、管理者グループは LDAP 設定の一部として定義されます。admin はログインを行い、開発者、プロジェクト管理者、RoboServer などのロールにマッピングする LDAP グループを定義できます。

内部ユーザー設定では、admin ユーザーは初期開始時に作成します。また、このユーザーはログインおよび管理者や開発者などのユーザーの作成が可能です。

組み込み管理者ユーザーの特別な権限

admin は、初期ユーザーであることに加えて、次のような特別な権限を持ちます。

- Management Console の [RoboServer] セクションで、admin は RoboServer ノードをクリックして、対応する RoboServer からスタック トレースをリクエストすることができます。
- admin のみがインポート バックアップを作成できます。
- パスワード ストアで、admin はパスワードを別のプロジェクトに移動できます。

Management Console のバックアップを復元すると、デフォルトの `admin` スーパーユーザーがバックアップのスーパーユーザーに置き換えられます。復元した Management Console で指定された資格情報を使用します。

組み込みグループ

RPA 管理者グループに属するユーザーは、特別な admin ユーザー権限を除く、すべてのプロジェクトに対するすべての権限を持ちます。RPA 管理者ユーザーは、任意のプロジェクトに対して新しい管理者とユーザーを作成します。ユーザーを管理者にするには、そのユーザーをこのグループに追加します。

- RPA 管理者グループは、内部ユーザー管理が有効な場合に表示され、デフォルトでは空の状態です。
- 10.7 より前のバージョンで作成されたバックアップを復元した場合、管理者ロールを持つユーザーは RPA 管理者グループのメンバーになります。

ログイン試行回数のチェック

デフォルトでは、ユーザーが行ったログイン試行回数と次の試行までの待ち時間のチェックがオフになっています。

1. この機能を有効にするには、`authentication.xml` ファイル内のコードを編集します。

   このファイルは次の場所にあります。`<Tomcat installation folder>\WebApps\Management Console\WEB-INF\spring` にある authentication.xml ファイル内の次のセクションを編集します。以下はコード サンプルです。

```
  <bean id="loginAttemptService"

 class="com.kapowtech.scheduler.server.spring.security.LoginAttemptService" lazy-init="true">
      <constructor-arg type="boolean" value="false"/>
      <constructor-arg type="int" value="3"/>
      <constructor-arg type="int" value="10"/>
  </bean>
```

2. 最初の値を **true** に設定します。
3. 2 番目と 3 番目の値を必要に応じて指定します。

2 番目と 3 番目の値は、ログイン試行回数 (この例では 3) と次の試行までの待ち時間 (この例では 10) です。

プロジェクト権限

admin はスーパーユーザーであるため、ログインに使用される admin ユーザー アカウントは、通常のユーザーに適用される通常のプロジェクト権限を無視します。スーパーユーザーはどのグループのメンバーにも属さず、すべてのプロジェクトに無制限にアクセスできます。

管理者パスワードを変更して新しいユーザーとグループを作成するには、Management Console > [管理] > [ユーザーとグループ] に移動します。セキュリティ モデルはロールベースです。ユーザーを作成したら、次の手順例に示すように、このユーザーを 1 つ以上のプロジェクトの特定のロールに関連付けられている 1 つ以上のグループに追加する必要があります。

「**DEVELOPER**」グループの作成

1. [グループ] タブで、プラス記号をクリックします。
   [新しいグループを作成] ダイアログ ボックスが表示されます。
2. 「DEVELOPER」という名前を指定して、説明を入力します。
3. [**OK**] をクリックします。
   「Developers」グループがテーブルに表示されます。

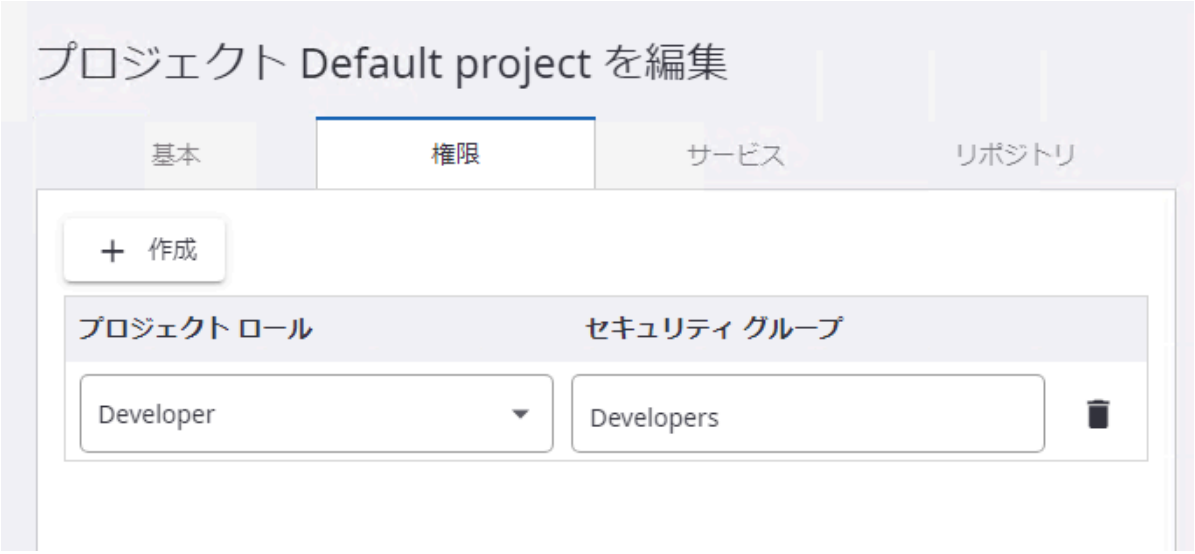
「**Dev**」ユーザーの作成

1. [ユーザー] タブで、プラス記号をクリックします。
   [新しいユーザーを作成] ダイアログボックスが表示されます。
2. ユーザー名「Dev」、パスワード、フル ネーム、ユーザーのメール アドレスを指定して、「DEVELOPER」グループを選択します。
3. [**OK**] をクリックします。
   「Dev」ユーザーがテーブルに表示されます。

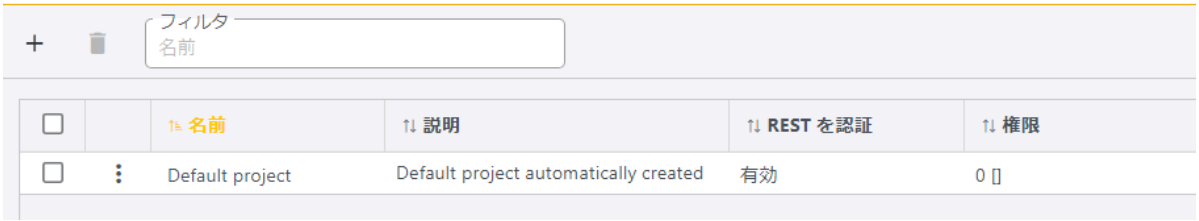
ここで、ユーザーがログインできるように、ユーザーに権限を割り当てる必要があります。[管理] > [プロジェクト] に移動します。このセクションの右側にある メニュー アイコンをクリックして、[権限] 列をクリックして表示します。現在、このプロジェクトの権限数は「0」と表示されています。

初期プロジェクト権限

1. デフォルト プロジェクトの ⋮ コンテキスト メニューで [編集] をクリックし、[権限] タブに移動します。[プロジェクト ロール] 列と [セキュリティ グループ] 列を含む枠があります。プロジェクト ロールは、ロボットのアップロード、スケジュールの作成、ログの表示といった Management Console 内部で実行できるアクションのセットを決定します。プロジェクト内で、プロジェクト ロールをセキュリティ グループに割り当てます。これにより、選択したセキュリティ グループのすべてのユーザーは、割り当てられたプロジェクト ロールで許可されているアクションを実行できます。
2. プラス記号をクリックして、このプロジェクトに権限を追加します。これにより、枠内に新しい行が追加され、プロジェクト ロールを選択できるドロップダウン ボックスが挿入されます。プロジェクト ロール [**DEVELOPER**] を選択します。
3. ここで、[セキュリティ グループ] 列をクリックして、**DEVELOPER** セキュリティ グループ (「Dev」ユーザーがメンバーになっているグループ) を選択します。次のようになります:



4. **[OK]** をクリックします。
「DEVELOPER」グループのすべてのメンバーが、開発者 (DEVELOPER) ロールで許可されているアクションを実行できるようになりました。デフォルト プロジェクトの [権限] 列に権限数が「1」と表示されるようになりました。



次に、「Dev」ユーザーとしてログインし、Management Console に権限がどのように反映されているのかを確認します。右上隅のメニュー ボタンをクリックしてログアウトし、dev ユーザーとしてログインします。ここで、[ログ ビュー] に移動し、左側のペインで RoboServer のログインを選択します。削除ボタンが無効になっていることに注目してください。このボタンの上にマウスを置くと、RoboServer メッセージを削除する権限がないことを示すツールチップ メッセージが表示されます。

同じセキュリティ グループには複数のロールを割り当てることができ、複数のセキュリティグループには同じロールを割り当てることができます。ユーザーが複数のロールを保持している場合、ユーザーは少なくとも 1 つのロールで許可されている操作を実行できます。Management Console の複数のプロジェクトでグループをプロジェクト固有のロールに割り当てることで、さまざまなプロジェクトのユーザーを完全に分離できます。

事前定義されたロールは推奨ロールですが、`roles.xml` ファイルを使用して、必要に応じて任意の個数のロールを追加したり、既存のロールを変更したりできます。

[設定]、[バックアップ]、および [ライセンス] セクションで実行できるアクションは、RPA Administrators グループのメンバーであるユーザーのみが使用できます。

LDAP ユーザー アカウントの使用については、高度な構成 > LDAP 統合 のトピックを参照してください。

セキュリティ

WEB-INF/Configuration.xml ファイルでは、Management Console 用にいくつかの追加のセキュリティ設定を設定できます。

これはセキュリティ設定の例です。

```
<bean id="securityConfiguration" class="com.kapowtech.mc.config.SecurityConfiguration">
   <property name="jdbcDriverUpload" value="LOCALHOST"/>
</bean>
```

この動作を変更するには、jdbcDriverUpload プロパティを変更します。可能な値は次のとおりです。プロパティには次のものがあります:

- なし: JDBC ドライバーのアップロードはどのユーザーにも許可されていません
- LOCALHOST: デフォルト値。admin ユーザーはローカルホストから Management Console にアクセスする場合にドライバーをアップロードできます。
- ANY_HOST: admin ユーザーは、任意のホストからドライバーをアップロードできます

Kofax RPA を Docker にデプロイする場合は、CONFIG_SECURITY_JDBCDRIVERUPLOAD 環境変数を使用し、値を NONE、LOCALHOST、または ANY_HOST に設定して、docker-compose ファイルでこの設定を行います。Docker デプロイメントで JDBC ドライバー アップロードを許可するには、この環境変数を ANY_HOST に設定します。詳細については Kofax RPA の展開のための Docker ツール を参照してください。

テレメトリ機能

Kofax はテレメトリ機能を使用して特定の指標を収集することで製品を分析し、改善します。

Kofax RPA ではテレメトリ機能がデフォルトで有効になっており、収集されたデータは 24 時間ごとに Kofax テレメトリ サービスに報告されます。

テレメトリ機能を無効にするには、次のいずれかの方法を使用します。

- ManagementConsole.war Web アーカイブ内にある WEB-INF/spring/common.xml ファイルを編集します。このファイルは解凍する必要があります。

  telemetryEnabled プロパティを見つけて、その値を false に設定します。
- Docker 環境では、SETTINGS_TELEMETRY_ENABLED 変数を false に設定します。

展開チェックリスト

次のチェックリストを使用して、すべての展開タスクが適切な順序で完了するようにします。

展開チェックリスト

アイテム	説明
サポートされているバージョンの Apache Tomcat サーバーをダウンロードしてインストールします	企業イントラネットの外部でセットアップが Management Console にアクセスする必要がある場合、Tomcat サーバーと連携するように SSL が設定されていることを確認してください。

アイテム	説明
Java をダウンロードしてインストールします	Management Console はサポートされていない Java のバージョンを使用して Apache Tomcat を起動した場合、正しく初期化されません。詳細については、Kofax RPA のドキュメント サイトにある『https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm 技術仕様』を参照してください。
JAVA_HOME 変数が Java インストール フォルダを指定していることを確認してください。	
Management Console アプリケーションをデプロイする前に、Apache Tomcat を起動し、Apache Tomcat サーバーがオンラインになることを確認します。	• `\bin` ディレクトリから catalina run コマンドを使用して、サーバーを起動できます。 • ブラウザで http://localhost:8080 にアクセスすると、デフォルトの Apache Tomcat 起動ページが表示されます。
Kofax RPA インストールから、Apache Tomcat `\webapps` ディレクトリの下にデプロイされる `ManagementConsole.war` ファイルを配置します。	この時点ではサーバーの設定を行っていないため、アプリケーションの最初の起動時にエラーが発生します。インストールおよび構成プロセスを完了するために編集する必要があるファイルに簡単にアクセスできるように、.war ファイルの解凍のみを行います。
Apache Tomcat サーバーをオフにします。	
Management Console エンタープライズで使用される新しいデータベースを作成し、その構成を保持します。	• いずれかのサポート プラットフォームを選択します。 • Kofax RPA インストール フォルダには、必要なすべてのデータベース テーブルを作成するために必要な CREATE および DROP SQL スクリプトが含まれています。詳細については Kofax RPA テーブルの SQL スクリプト を参照してください。これらのスクリプトは、アプリケーションがデータベースに接続するために使用するユーザー アカウントにスキーマに対する CREATE 特権がない場合にのみ使用する必要があることに注意してください。

アイテム	説明
JDBC を介してデータベースに接続するためにアプリケーションで使用される DB ユーザー アカウントを作成します。	
Tomcat コンテキスト ファイルを作成します: `ManagementConsole.xml`。	• 指定する検証クエリはデータベース固有です。オンライン ヘルプには、サポートされるすべてのデータベースで受け入れられるすべての値が含まれています。 • JDBC URL 文字列の Username、Password、および DatabaseName パラメーターを、データベースの正しい値で必ず更新するようにしてください。
アプリケーションを起動する前に、Management Console データベースに（`ManagementConsole.xml` に指定されているデータベース）、Management Console プロセスに関連するテーブルがないことに注意します。	• アプリケーションの起動時に、DBUser アカウントに CREATE 権限があると仮定して、必要なデータベーステーブルが存在しない場合は自動的に作成されます。 • ユーザーに CREATE 権限がない場合、CREATE SQL スクリプトは、Kofax RPA インストール ディレクトリにある`documentation\sql` ディレクトリで見つかります。詳細については、Kofax RPA テーブルの SQL スクリプト を参照してください。
Apache Tomcat のインストール ディレクトリの下に、必要な JDBC .jar ファイルを必ずデプロイするようにしてください。	• `<APACHE_TOMCAT_INSTALL_DIR>\lib` にデプロイすると、Tomcat の下で実行されているすべてのアプリで使用できるようになります。 • `<APACHE_TOMCAT_INSTALL_DIR>\webapps\ManagementConsole\WEB-INF\lib` により、JDBC .jar ファイルが Management Console アプリケーションにのみアクセスできます。
Apache Tomcat サーバーを再起動します	Tomcat のメイン ホームページが読み込まれることを確認します: http://localhost:8080
ユーザー、管理者、パスワード、管理者として Management Console へのログインを試行します。	`http://localhost:8080/ManagementConsole`
Kofax RPA ソフトウェア ライセンス キーを入力します	ライセンス情報の入力を参照してください。

Kofax RPA の展開のための Docker ツール

Kofax は、Linux および Windows 環境への Kofax RPA の展開を迅速かつ簡単に行うための Docker ツールを提供しています。Kofax RPA の Docker ツールは、RPA コンポーネントの Docker イメージをビルドする場合に役立ちます。現在、イメージは次のコンポーネントで利用できます。

Windows Docker コンテナのリソース使用量が大幅に増加するため、可能な場合は Linux バージョンを使用することをお勧めします。Windows でのみ実行されるコンポーネントには、Windows Docker コンテナ バージョンのみを使用できます。また、Windows Docker コンテナの実行は、Windows Server ホストでのみサポートされます。

公式の Kofax RPA イメージが Docker ハブを通じて提供されるようになりました。以下のリンクを使用して必要なイメージをダウンロードするか、docker-compose ファイルを使用してイメージをビルドで

きます (詳細については、docker-compose ファイルを使用して Kofax RPA を展開する を参照してください)。Linux イメージのみが利用可能であることに注意してください。

コンポーネント	Linux	Windows
Management Console https://hub.docker.com/r/kofax/rpa-managementconsole	はい	はい
RoboServer https://hub.docker.com/r/kofax/rpa-roboserver	はい	はい
Robot File System	はい	はい
Synchronizer https://hub.docker.com/r/kofax/rpa-synchronizer	はい	はい
Kapplets https://hub.docker.com/r/kofax/rpa-kapplets	はい	はい
Document Transformation	いいえ	はい
Kofax Analytics for RPA	いいえ	はい

この章では、Docker ツールの詳細と使用例を示します。Docker に関する詳細については、https://www.docker.com を参照してください。スタンドアロン サーバーで Kofax RPA を手動で展開する方法については、Tomcat の展開を参照してください。

Windows Docker に関する注意事項

本番環境では、上記の表で指定されている推奨オペレーティング システム コンテナの下で Docker コンテナを実行することをお勧めします。Windows コンテナのイメージは、高可用性機能またはヘルスチェック設定を備えていません。Windows Docker コンテナが Kofax RPA 設定に含まれていて、現在はこの設定が Document Transformation および Kofax Analytics for RPA でのみ必須な場合は、Linux ノードと Windows Server 2019 (以降) のノードを含むハイブリッド Kubernetes クラスタ内に設定を展開することをお勧めします。以前のバージョンでは、イメージが大幅に大きくなります。

Windows Docker コンテナを実行する場合は、このオペレーティング システムに固有の次の決定を行う必要があります。

- 分離レベル

  使用するランタイム分離モードを決定します。「プロセス」と「Hyper-V」を使用できます。Windows Server のデフォルトは「プロセス」、Windows 10 のデフォルトは「Hyper-V」です。「プロセス」分離モードを使用する場合、コンテナの基本バージョンは、ホストのオペレーティング システムのバージョンと一致している必要があります。Windows Server 2019 ホストでコンテナを実行する場合、servercore のバージョン `ltsc2019` に基づいてコンテナを構築する必要があります。

  servercore バージョン `ltsc2019` 以降を使用する場合、オペレーティング システム コンポーネントによっては、最適化を行ってから RoboServer イメージに追加しないと機能しない場合があります。`docker-win\roboserver` にある RoboServer の Dockerfile を確認し、dxva2.dll スクリプトがコピーされているコメント部分に注目します。コピーをビルド フォルダに移動するか、コメントで指示されている場所を指すように COPY コマンドを変更します。

  また、docker-compose ファイルを実行する前に、必須の Dockerfile 内で `SERVERCORE_VERSION` ビルド引数を適切に設定します。たとえば、Windows Server 2019 を使用している場合は引数を `ltsc2019` に設定し、Windows Server 2022 を使用している場合は引数を `ltsc2022` に設定します。

- バージョン

  使用する Windows Server のバージョンを決定します。Windows Server 2019 バージョン以降、Windows ベースのコンテナ イメージは大幅に最適化されています。リソース使用量が増加するため、以前のバージョンを使用することはお勧めしません。

  ただし、Microsoft SQL Server イメージなどの一部の Windows コンテナは、2019 バージョンでは使用できません。公式の Microsoft イメージは Linux Docker コンテナのみをサポートしています。Microsoft が提供する他のイメージは、2019 バージョンとしては使用できません。使用できるオプションは次のとおりです。

  - Windows Server 2016 など、すべての前提条件が利用可能なオペレーティング システムで「プロセス」分離モードで実行し、これらのイメージに基づいてビルドします。この場合、追加のランタイム リソース コストが発生します。
  - コンテナごとに最適な設定を使用して、「Hyper-V」分離モードで実行します。この場合、仮想マシンを実行するというオーバーヘッドが生じます。
  - 最適なバージョンで使用できないサードパーティのイメージは (MS SQL Server など)、別のコンピューターに展開します。この場合は、公式の Microsoft イメージ (Linux) を推奨します。
  - 「servercore:2019ltsc」以降に基づいて Microsoft SQL Server バージョンが使用できるようになるまでお待ちください。
  - Windows Docker が、「プロセス」分離モードで、Windows コンテナと Linux コンテナを 1 台のコンピューターで同時に実行できるようになるまでお待ちください。

  本番環境では、すべてのコンポーネントを 1 台のコンピューターで実行することを目的としていないため、Windows Docker コンテナが必要な場合は、ハイブリッド Kubernetes クラスタを使用するか、別の形式のハイブリッド設定を使用することをお勧めします。

RoboServer の前提条件

RoboServer が含まれている docker-compose ファイルを実行する前に、Kofax RPA のインストール フォルダ内の `docker-win\roboserver` フォルダに移動します。このフォルダで、`copy_fonts.ps1` スクリプトを探して実行し、Windows システム フォントを `roboserver\Fonts` フォルダにコピーします。

Kofax Analytics for RPA の前提条件

Kofax Analytics for RPA が含まれている docker-compose ファイルを実行する前に、次の手順を実行します。

1. Kofax RPA インストール フォルダ内の `docker-win\kafrpa` フォルダに移動し、`copy_fonts.ps1` スクリプトを検索して実行し、Windows システム フォントを `docker-win\kafrpa\Insight\Fonts` フォルダにコピーします。
2. Kofax Analytics for RPA プロジェクト バンドルの名前を **kafrpa_bundle.zip** に変更して、`docker-win\kafrpa\Insight` フォルダにコピーします。
3. Insight インストール パッケージの名前を **KofaxInsightSetup.msi** に変更して、`docker-win\kafrpa\Insight` フォルダにコピーします。
4. 必要なライセンス ファイル `Altosoft.Insight.License.xml` を取得して、`docker-win\kafrpa` フォルダにコピーします。

Document Transformation Service の前提条件

Kofax Analytics for RPA が含まれている docker-compose ファイルを実行する前に、次の手順を実行します。

1. Kofax RPA インストール フォルダ内の `docker-win\compose-examples` に移動し、`docker-compose-dt.yml` ファイルを開き、必要に応じて次の行を変更します。
   `DT_LICENSE_SERVER=put-your-license-server-here.example.com # Kofax license server`
2. 変更した `docker-compose-dt.yml` ファイルを `docker-win\documenttransformation` にコピーします。
3. NLP バンドル `KofaxTransformation-6.3.1.0_NLP-LanguageBundles` (`KofaxTransformation_Salience6.4_LanguageBundle_extended.msi`、`KofaxTransformation_Sal default.msi`、および `KofaxTransformation_Salience6.4_LanguageBundle_western-extended.msi`) を `docker-win\documenttransformation` にコピーします。
4. `KofaxRPA_DocumentTransformationService-11.5.0.0.msi` を `docker-win\documenttransformation` にコピーします。

docker-compose ファイルを使用して Kofax RPA を展開する

このトピックでは、Kofax RPA を Linux ベース システムまたは Windows ベース システムのスタンドアロン サーバーに展開するための基本手順を示します。Kofax RPA で指定された Windows および Linux の docker-compose サンプル ファイルの構成については、次のセクションを参照してください。

現在、Docker Compose バージョン 2 は `docker compose` 構文を使用しています。ただし、`docker-compose` 機能は引き続きサポートされ、`docker-compose` 構文の `docker compose` へのエイリアシングはデフォルトで有効になっています。ハイフン (-) をスペースに置き換え、`docker-compose` の代わりに `docker compose` を使用することで、Compose V2 を実行できます。詳細については、Docker ドキュメントを参照してください。

docker-compose ファイルを使用して Kofax RPA を展開するには、次の手順を実行します。

1. コンピューター で Kofax RPA をインストールします。
2. https://www.docker.com から Docker をダウンロードし、コンピューターにインストールします。
3. Linux にのみ適用されます。新しいユーザーを Docker グループに追加します。たとえば、「Kofax」ユーザーを追加してユーザーに Docker コンテナへのアクセスを許可するには、`/etc/group` ファイル内の `docker:x:<n>` を `docker:x:Kofax` に置き換えます。権限を更新するには、ログアウトしてからログインするか、コンピューターを再起動します。
4. `compose-examples` フォルダ内で、ニーズに最適な docker-compose ファイルを選択します。ファイルを `compose-examples` フォルダから Kofax RPA インストール フォルダのルートにコピーし、必要に応じてファイル名を `docker-compose.yml` に変更します。
   たとえば、Windows で作業している場合は、`C:\Program Files\Kofax RPA 11.5.0.0\docker-win\compose-examples` から `C:\Program Files\Kofax RPA 11.5.0.0` に `docker-compose.yml` をコピーします。
5. 環境とニーズに応じて、docker-compose ファイルを編集します。
   - `CONFIG_LICENSE_` 変数にライセンス情報を追加します。

Management Console を実行中は、ライセンスを変更することはできません。ライセンス設定を変更するには、設定を停止し docker-compose ファイルのライセンス変数を更新します。ライセンス設定が docker-compose ファイルに含まれていない場合は、コンテナを停止しなくても Management Console 内でライセンス設定を変更できます。

- 初期の admin ユーザー名とパスワードを設定します。

  デフォルトでは、次のデフォルトの admin スーパーユーザー名とパスワードが Management Console への認証に使用されます。

  ユーザー名: `admin`

  パスワード: `admin`

  カスタムの認証情報を使用するように、`LOGIN_INITIAL_ADMIN_USER` 環境変数と `LOGIN_INITIAL_ADMIN_PASSWORD` 環境変数を設定します。クレデンシャルを設定すると、この情報はデータベースに保存され、再度変更することはできなくなります。

  データベースが空であるか admin ユーザーが存在せず、システムがローカル認証を使用するように設定されているか、(LDAP などの場合) `LOGIN_CREATE_ADMIN_USER` 変数が `true` に設定されている場合は、admin ユーザーが作成されます。admin ユーザーがデータベースにすでに含まれている場合、そのユーザーは作成されません。

Kofax RPA サービスを開始します。製品のインストール フォルダから次のコマンドを実行してイメージをビルドし、サービスを開始して、現在のコンテナに「kofaxrpa」という名前を設定します。

```
docker-compose -p kofaxrpa up -d
```

初めてイメージをビルドするときは、準備に時間がかかる場合があります。

次のように、個別のコマンドを使用してイメージをビルドし、サービスを開始することができます。

Docker ハブのタグにはビルド番号が含まれているため、docker-compose ファイルはデフォルトで「最新の」イメージを使用します。

- イメージをビルドするには、次のコマンドを使用します。
  - Linux の場合: `docker build -f docker/managementconsole/Dockerfile -t managementconsole: 11.5.0.0`
  - Windows の場合: `docker build -f docker-win\managementconsole\Dockerfile -t managementconsole: 11.5.0.0`

  行末のスペースとピリオドに注意してください。ピリオドは現在のディレクトリを指します。
- サービスを開始します。

  Linux および Windows に対して: `docker-compose -p kofaxrpa up -d`

docker-compose ファイルを実行すると、コンテナの起動後、すぐに Docker ホストを開けるようになり (デフォルトの場所は `http://localhost`)、Management Console が実行されていて、RoboServer が別のコンテナに格納されていることを確認できます。また、docker-compose ファイルに含まれていた、Kofax RPA によってコンテナ化された他のコンポーネントにアクセスできるようになります。

RPA バージョン 11.2 以降では、RoboServer は UTC 時間でログを書き込みます。11.2 より前のバージョンの RoboServer は、デフォルトではローカル サーバー時間でログを書き込みます。そのため、11.2 とそれ以前のバージョンの RoboServer の両方が同じログ データベースにログを記録すると、タイムスタンプの不一致が生じることがあります。11.2 より前のバージョンの RoboServer をバージョン 11.2 以降の Management Console に接続する場合、Docker 設定ファイルの `roboserver-service>environment` セクションに次のオプションを指定することで、ローカル サーバー時間ではなく UTC 時間でログ メッセージを書き込むように設定できます。

```
- WRAPPER_JAVA_ADDITIONAL_1=-DwriteLogdbUtc=true
```

RoboServer は、このパラメータをサポートするフィックスパック バージョンに更新する必要があります。詳細については、対応する RPA フィックスパックの ReadMe ファイルを参照してください。

次のセクションでは、docker-compose の例と設定について詳しく説明します。

Docker-compose の例

Kofax RPA には、`docker/compose-examples` フォルダにいくつかの単純な構成を持つ docker-compose ファイルが複数含まれています。Windows の場合、ファイルは `docker-win\compose-examples` に配置されています。

Linux の場合、次のすべての例で、Management Console の設定データベースとして PostgreSQL が利用されます。Management Console は他のデータベースで実行できますが、Management Console の設定データとしては PostgreSQL が推奨されます。PostgreSQL docker イメージのドキュメントは https://hub.docker.com/_/postgres で入手できます。Windows の場合、docker-compose の例では、設定データベースとして Microsoft SQL Server が利用されます。

ロギングおよびロボット データ ストレージには、サポートされているデータベースを使用できます。『Kofax RPA インストール ガイド』の「サポートされているプラット フォーム」を参照してください。

Linux 環境用の Docker-compose ファイル

Linux 環境用に、次の docker-compose ファイルが提供されています。

docker-compose-basic.yml

この設定により、Management Console、RoboServer、PostgreSQL データベースを起動し、それらを接続します。この設定を開始する前に、ライセンス情報を設定に入力して、Management Console の起動時に入力する必要がないようにすることができます。RoboServer (「Production」クラスタ内) の数をスケーリングするには、次のコマンドを使用します。

```
docker-compose -p kofaxrpa up -d --scale roboserver-service=2
```

docker-compose-ha.yml

この設定により Management Console、RoboServer、PostgreSQL データベース、および軽量の Traefik イメージに基づくロード バランサーを起動します。

Management Console マルチキャスト検出を使用して高可用性を有効にし、実行するように設定されています (エンタープライズ ライセンスが必要です)。詳細については、Docker Swarm 上での Management Console の高可用性モードによる実行を参照してください。

マルチキャスト検出には、UDP マルチキャストをサポートするネットワーク オーバーレイが必要です。

この設定を最適に機能させるには、サービスを起動する前にライセンス情報を docker-compose ファイルに入力します。また、コンテナに割り当てられたネットワークに合わせて次の行を編集します。

```
- CONFIG_CLUSTER_INTERFACE=172.20.0.*
```

詳細な手順と変数は、Kofax RPA インストール内の `docker` フォルダにある `.md` ファイルにあります。

次のステップを実行して、使用されているネットワークを確認します。

1. コンポジションを開始します。
   ```
   docker-compose -p kofaxrpa up -d
   ```
2. 開始されたコンテナをリストします。
   ```
   docker container ls
   ```
3. 次のコマンドを使用して、ホスト名を取得します。
   ```
   docker exec kofaxrpa-managementconsole-service-1 hostname -i
   ```
4. docker-compose ファイルを編集する前に構成を停止します。
   ```
   docker-compose -p kofaxrpa down
   ```

`traefik.docker.network` 変数値には、現在のコンテナの名前が含まれている必要があります。この例では、コンテナ名は 「kofaxrpa」 であるため、`traefik.docker.network=`**`kofaxrpa`**`_net` となります。

ワイルドカードを使用できるため、IP アドレスは `172.*.*.*` のようになります。Hazel cast はすべての数字の代用としてのワイルド カードを受け入れないため、`*.*.*.*` は許可されないことに注意してください。

設定を開始するときに、次のコマンドを使用する Management Console インスタンス実行中の数をスケーリングできます。

```
docker-compose -p kofaxrpa up -d --scale managementconsole-service=2
```

Management Console の 2 つ以上のインスタンスを実行することは可能ですが、その場合はデータベースの負荷が増加します。ロード バランサーは、スティッキー セッションを使用するように設定されています。

docker-compose-kapplets.yml

この設定により、Management Console、RoboServer、PostgreSQL データベースを起動するとともに、Kofax RPA Kapplets の追加と設定を行います。

docker-compose-ldap.yml

これは、LDAP を使用する設定例です。この設定は、コンテナ内の `OpenLDAP` コマンド (例として含まれていますが、通常は不要) を起動します。

また、テスト用に関連ファイル `ldap_ad_content.ldif` が含まれています。次のコマンドを実行して、この設定をテストします。

```
docker-compose -p kofaxrpa up -d && docker -vv cp ./docker/compose-examples/
ldap_ad_content.ldif kapow_ldap-service_1:/ldap_ad_content.ldif && docker exec
```

```
kapow_ldap-service_1 ldapadd -x -D "cn=admin,dc=example,dc=org" -w admin -f /
ldap_ad_content.ldif
```

docker-compose-rfs.yml

この設定により、Management Console、RoboServer、PostgreSQL データベースを起動するとともに、Robot File System の追加と設定を行います。

docker-compose-synchronizer.yml

この設定により、Management Console、PostgreSQL、RoboServer を起動するとともに、Synchronizer の追加と設定を行います。

Docker JMX ヘルス チェックを無効にする場合は、Dockerfile 内の次の間にある行を削除します。

`# ### start cutting here ###` と `# ### end cutting here ###`

Windows 環境用の Docker-compose ファイル

Windows 環境用に、次の docker-compose ファイルが提供されています。

docker-compose.yml

この設定は、使用可能な次のすべての Kofax RPA コンポーネントを起動します (Document Transformation は除く): Management Console、RoboServer、Synchronizer、Robot File System、Kapplets、Kofax Analytics for RPA、および MS SQL データベース。

docker-compose-dt.yml

この構成ファイルは、Document Transformation Windows Docker コンテナで使用されるパラメータを文書化する際に使用します。Document Transformation が必要な場合は、次のように `docker-compose-dt.yml` サンプル ファイルを使用できます。このファイルを実行する前に、Windows Docker に関する注意事項で示されている Document Transformation Service の前提条件を満たしていることを確認してください。

```
docker-compose -f .\docker-compose-dt.yml up
```

Docker ビルドのコンテキスト サイズの最適化

このセクションの情報は、Linux にのみ適用されます。

独自のイメージをビルドすることはお勧めしません。ただし、ビルドする場合、このセクションが適用されます。

すべての Docker イメージは、ビルド コンテキストとしてディストリビューションのルート フォルダを使用してビルドされるため、イメージのビルドに極端に長い時間がかかることがあります。ビルド コンテキストのサイズを最適化するために、`dockerignore` ファイルがディストリビューションに追加されています。コンポーネントによって要件が異なるため、コンポーネントごとに個別の `dockerignore` ファイルが提供されています。このファイルは Dockerfile の隣に配置して同じ命名規則に従うようにする必要があります。ただし、`Dockerfile.dockerignore` のように `.dockerignore` サフィックスを付ける必要があります。

この機能を使用するには、BuildKit によるビルドを有効にする必要があります。現在、標準の Docker ビルドはグローバルな `.dockerfile` のみをサポートしています。この機能を有効にするには、環境内で `DOCKER_BUILDKIT = 1` を設定するか、`/etc/docker/daemon.json` 内で定義します。

docker-compose を使用してビルドしている場合は、環境内で COMPOSE_DOCKER_CLI_BUILD = 1 も設定する必要があります。BuildKit でビルドしていない場合は、Dockerfile.dockerignore ファイルの名前を .dockerignore に変更して、ビルド コンテキストのルートに移動することができます。docker-compose などを使用して複数のコンポーネントをビルドする場合は、dockerignore コンテンツを結合する必要があります。ただし、BuildKit によるビルドはより効率的であるため、この方法をお勧めします。

Docker シークレット機能を使用してパスワードを保存する

Linux と Windows の docker-compose ファイルで接続パスワードを直接指定しないようにするには、Docker シークレット機能を使用してパスワードを安全な場所に保存します。

環境変数にシークレット機能を使用できます。

次の例では、Linux ベース環境で PostgreSQL データベースに接続するためのパスワードを指定します。

1. パスワードを含むテキストファイルを作成します。1 つのファイルに 1 つのパスワードを含める必要があります。この例では、PostgreSQL データベースのパスワードが含まれる postgrespassword.txt ファイルを作成しました。
2. 次のように、使用する docker-compose ファイル内の最初のレベル (インデントなし) で secrets というセクションを作成し、2 番目のインデント レベルで変数名を指定して、3 番目のインデント レベルでパスワードを含むファイルの相対または絶対パスを指定します。

```
secrets:
  postgrespassword:
    file: postgrespassword.txt
```

3. .yml ファイルの services > postgres-service > environment セクションで、POSTGRES_PASSWORD_FILE 変数を POSTGRES_PASSWORD 変数で置き換え、以下のように secrets セクションで前に指定した変数を参照します。

```
services:
  postgres-service:
    image: postgres:10
    environment:
      - POSTGRES_DB=postgresdatabase
      - POSTGRES_USER=postgresuser
      - POSTGRES_PASSWORD_FILE=/run/secrets/postgrespassword
    secrets:
      - postgrespassword
```

例としてこの手順を使用すると、docker-compose ファイル内の他のパスワードのシークレット機能を設定できます。

データベースのセットアップ

Kofax RPA では、コンテナ化されたデータベースに Management Console の設定およびリポジトリ データを保存し、データ ストレージと (監査) ログ用の外部データベースを追加することをお勧めします。

ただし、外部または企業データベースに Management Console 内部設定データを保存する必要がある場合もあります。Management Console データベースを変更し、データベース コンテキストの環境変数を修正するには、適切な JDBC ドライバーをイメージ/コンテナーに追加します。

Management Console の Dockerfile (Linux の場合は `docker/managementconsole`、Windows の場合は `docker-win\managementconsole`) の次の行は、現在の JDBC ドライバーを JDBC フォルダ内の Management Console イメージに追加します。

Linux: `ADD --chown=${user}:${group} https://jdbc.postgresql.org/download/postgresql-[バージョン].jar /usr/local/tomcat/lib/jdbc/`

Windows: `ADD https://clojars.org/repo/com/microsoft/sqlserver/sqljdbc4/4.0/sqljdbc4-4.0.jar${CATALINA_HOME}/lib/jdbc/`。ここで、`CATALINA_HOME=${KAPOW_HOME}\tomcat\apache-tomcat-@tomcatNumericVersion@` および `KAPOW_HOME=c:\kapow`

この行を変更するか、Dockerfile に行を追加して、Java に適した JDBC ドライバーと正しいバージョンのデータベース コネクタを追加します。利用可能な最新バージョンのドライバーを使用します。

Dockerfile を編集した後に、次のコマンドを実行してイメージを再ビルドする必要があります。

Linux: `docker build -f docker/managementconsole/Dockerfile .-t managementconsole:@productVersion@`

Windows: `docker build -f docker-win\managementconsole\Dockerfile .-t managementconsole:@productVersion@`

または、シンプル バージョンを実行します:

```
docker-compose -p kofaxrpa up -d
```

バックアップと復元

このセクションの情報は、Linux にのみ適用されます。Management Console イメージには、リポジトリおよび設定情報をバックアップおよび復元するための 2 つのスクリプトが含まれています。

backup.sh

`/kapow/backup` に保存するバックアップを作成するには、次の Docker コマンドを実行します。

```
docker exec kapow_managementconsole-service_1 backup.sh [オプション]
```

次のスクリプト使用オプションが利用可能です:

```
backup.sh [options]
```

オプション

オプション	説明
`-u, --username`	必須。Management Console を呼び出すときに使用するユーザー名です。
`-p, --password`	必須。Management Console を呼び出すときに使用するパスワードです。
`-h, --host`	Management Console (デフォルト: localhost) のホスト名です。
`-i, --project <Id>`	特定のプロジェクトのみをバックアップするために使用するプロジェクト ID。

オプション	説明
`-c, --configurationOnly <Id>`	プロジェクトデータではなく、設定のみをバックアップする ID。
`-n, --postfix`	作成されたバックアップ ファイルの接尾辞を設定します (デフォルトは <datetime>)。

restore.sh

`/kapow/backup` に保存されたバックアップを復元するには、次の docker コマンドを実行します。

```
docker exec kapow_managementconsole-service_1 restore.sh [オプション]
```

次のスクリプト使用オプションが利用可能です:

```
restore.sh [options]
```

オプション

オプション	説明
`-u, --username`	必須。Management Console を呼び出すときに使用するユーザー名です。
`-p, --password`	必須。Management Console を呼び出すときに使用するパスワードです。
`-h, --host`	Management Console (デフォルト: localhost) のホスト名です。
`-f, --filename`	必須。復元するバックアップ ファイルの名前。
`-a, --path`	復元するバックアップ ファイルへのパス (デフォルト: `/kapow / backup /`)。

事前開始チェック

コンテナを起動すると、`ManagementConsole` イメージはデフォルトで次のチェックを実行します。

1. 指定された JDBC URI を使用してデータベースに接続することにより、データベース設定が機能することを確認します。また、検証クエリを 1 回実行しようとします。
2. LDAP 構成が機能することを確認します。LDAP が設定されている場合、接続とバインドを試行し、クエリを実行してすべてのグループを取得することにより、各 LDAP ディレクトリがチェックされます。より多くのルックアップに使用するテスト ユーザー名を追加することにより、デバッグまたは検証目的でこのテストを拡張できます。

チェックが失敗した場合、イメージは設定された時間で再試行を繰り返します。設定されたタイムアウト内にチェックのいずれかが成功しなかった場合、コンテナは終了し、設定パラメータでその内容を確認できます。

チェックのタイムアウトをゼロ (0) に設定することで、チェックを実行しないようにすることができます。詳細については、環境変数 セクションを参照してください。

データフォルダ

コンテナーからのログ、データベースデータ、および設定は、以下にリストされているフォルダに保存されます。コンテナが大きくならないようにするには、これらのフォルダをボリュームまたはローカルのファイル システムにマウントする必要があります。ボリュームの Docker ドキュメントを参照してください。

これらのボリュームに保存されているログ、データベース、およびその他のデータは、Kofax RPA のバージョンのアップグレード時には再利用できません。

RoboServer

RoboServer コンテナのデータ、ログ、および設定は、次のフォルダ内にあります。

Linux: `/kapow/data`

Windows: `C:\kapow\data`

ManagementConsole

Tomcat ログは次のフォルダにあります:

Linux: `/usr/local/tomcat/logs`

Windows: `C:\kapow\tomcat\apache-tomcat-<tomcat` のバージョン`>\logs`

環境変数

次の表に、Docker を使用して RPA を展開する際に設定が必要になる共通の変数をいくつか示します。docker-compose ファイルに関連するさまざまな Docker コンテナの環境変数の完全なリストは、Kofax RPA インストール内の `docker` フォルダ内にある `readme.md` ファイルに含まれています。

変数	デフォルト値	説明
`ROBOSERVER_CONNECTION_TYPE`	(NoConnectionType)	RoboServer と Management Console の間の接続タイプを指定します。必要な値を選択します。 • `ClientConnectionType` • `SocketConnectionType` • `SocketSSLConnectionType` 詳細については RoboServer を参照してください。
`RFS_MC_SHARED_SECRET`	( )	Robot File System を Management Console で認証するために使用される共有シークレット。
`ROBOSERVER_MC_SHARED_SECRET_FILE`	( )	Management Console に登録する際に使用する共有シークレットを含むファイルへのパス
`LOGIN_LDAP_DIRECTORY_CONVERTTOUPPERCASE_N`	(true)	グループ名を大文字に変換します。
ロギング用の変数		

変数	デフォルト値	説明
LOG4J_LOGGER_ADDITIONAL_COUNT	(0)	log4j.properties ファイルに追加する追加プロパティの数。 ⚠ log4j 設定の構文が変更され、log4j2 が使用されるようになりました。log4j2.properties の構文を確認し、それに応じてパラメータを指定してください。
LOG4J_LOGGER_ADDITIONAL_KEY_N	( )	追加プロパティ N のキー。例: LOG4J_LOGGER_ADDITIONAL_KEY_1
LOG4J_LOGGER_ADDITIONAL_VALUE_N	( )	追加プロパティ N の値。ロギングレベルをデバッグに設定する場合は、値として DEBUG を指定します。以下の「例: デバッグ ロギングの値」を参照してください。

例: デバッグ ロギングの値

次の例では、ロギングをデバッグ レベルに設定し、詳細なロギング情報を Tomcat サーバー上のファイルに書き込みます。詳細については、apache.org Web サイトの Apache Log4j 2 を参照してください。

⚠ log4j 設定の構文が変更され、log4j2 が使用されるようになりました。log4j2.properties の構文を確認し、それに応じてパラメータを指定してください。

```
- LOG4J_LOGGER_ADDITIONAL_COUNT=10
- LOG4J_LOGGER_ADDITIONAL_KEY_1=rootLogger.level
- LOG4J_LOGGER_ADDITIONAL_VALUE_1=DEBUG
- LOG4J_LOGGER_ADDITIONAL_KEY_2=logger.spring.level
- LOG4J_LOGGER_ADDITIONAL_VALUE_2=DEBUG
- LOG4J_LOGGER_ADDITIONAL_KEY_3=appenders
- LOG4J_LOGGER_ADDITIONAL_VALUE_3=A, file
- LOG4J_LOGGER_ADDITIONAL_KEY_4=appender.file.type
- LOG4J_LOGGER_ADDITIONAL_VALUE_4=File
- LOG4J_LOGGER_ADDITIONAL_KEY_5=appender.file.name
- LOG4J_LOGGER_ADDITIONAL_VALUE_5=file
- LOG4J_LOGGER_ADDITIONAL_KEY_6=appender.file.fileName
- LOG4J_LOGGER_ADDITIONAL_VALUE_6=/usr/local/tomcat/logs/output.log
- LOG4J_LOGGER_ADDITIONAL_KEY_7=appender.file.layout.type
```

```
- LOG4J_LOGGER_ADDITIONAL_VALUE_7=PatternLayout
- LOG4J_LOGGER_ADDITIONAL_KEY_8=appender.file.layout.pattern
- LOG4J_LOGGER_ADDITIONAL_VALUE_8=%d{HH:mm:ss,SSS} %-5p %c %equals{%x}{[]}{} - %m%n
- LOG4J_LOGGER_ADDITIONAL_KEY_9=rootLogger.appenderRefs
- LOG4J_LOGGER_ADDITIONAL_VALUE_9=A, file
- LOG4J_LOGGER_ADDITIONAL_KEY_10=rootLogger.appenderRef.file.ref
- LOG4J_LOGGER_ADDITIONAL_VALUE_10=file
```

Docker Swarm 上での Management Console の高可用性モードによる実行

次の手順と設定は例としてのみ意図されており、推奨事項ではありません。

このセクションの例は、Kofax RPA (高可用性 モードが有効になっている場合) の複数のインスタンスを持つ Docker swarm での Management Console 設定方法を示しています。

このセクションにはマネージャーとワーカーの 2 つのノードで構成される Docker swarm をセットアップする例が含まれており、1 つのノードのみを使用するよりも高いレベルのフォールト トレランスを提供します。この例では PostgreSQL を使用しています。

Management Console はクラスタ内の複数のインスタンスとして実行でき、データベースを介して設定、ログおよびリポジトリ データを共有し、Hazelcast プラットフォームを介してクラスタの揮発性状態を共有できます。このプラットフォームでは Management Console のインスタンスがそれぞれクラスタ内の他のインスタンスを検出して接続できるようにします。現在実装されている 2 つの検出方法は、マルチキャストと TCP です。この手順例ではマルチキャスト (UDP) ディスカバリー方式を使用します。TCP ディスカバリー方式も正常にテストされました。

マルチキャスト UDP を Docker swarm で機能させるには、この手順で Weave Net プラグイン 2.5.1 を使用します。

Management Console のある最小限の Docker swarm をセットアップ

続行する前に、Linux カーネル 3.8 以降で実行されている 2 つの Docker ホストがあることを確認してください。この手順では、ホストを `host1` および `host2` と呼びます。

1. Docker swarm クラスタをセットアップします。
   a. `host1` でマネージャー ノードを作成します。

      ```
      host1$ docker swarm init --advertise-addr <host1_IP>
      ```

   b. `host2` をワーカー ノードとして swarm に参加させます。

      ```
      host2$ docker swarm join --token <token> --advertise-addr <host2_IP>
       <host1_IP>:<port>
      ```

      `<token>` を最初のコマンドから取得したトークンに置き換えます。
2. 2 つのホストに Weave Net プラグインをインストールし、次のようにマルチキャスト機能を有効にします。

   ```
   host1$ docker plugin install store/weaveworks/net-plugin:2.5.2 --grant-all-
   permissions --disable
   ```

```
host1$ docker plugin set store/weaveworks/net-plugin:2.5.2 WEAVE_MULTICAST=1
host1$ docker plugin enable store/weaveworks/net-plugin:2.5.2
```

```
host2$ docker plugin install store/weaveworks/net-plugin:2.5.2 --grant-all-
permissions --disable
host2$ docker plugin set store/weaveworks/net-plugin:2.5.2 WEAVE_MULTICAST=1
host2$ docker plugin enable store/weaveworks/net-plugin:2.5.2
```

3. マネージャー ノードで、Weave Net プラグインをドライバーとして使用して、次のように Docker ネットワークを作成します。

   ```
   host1$ docker network create --driver=store/weaveworks/net-plugin:2.5.2 --
   attachable weavenet
   ```

4. Docker swarm のすべてのノード上で Management Console と RoboServer の Docker イメージを構築します。Docker リポジトリを使用すると、そこにイメージをプッシュできます。Docker イメージの構築の詳細については、Kofax RPA の展開のための Docker ツール を参照してください。

5. **docker-compose.yml** というファイルを作成し、次の行を環境に合わせて変更します。

```
version: '3.2'
    networks:
      net:
        external:
          name: weavenet
    services:
      loadbalancer:
        image: traefik
        command: --docker --docker.swarmmode --docker.watch --web --loglevel=DEBUG
        ports:
          - 80:80
        volumes:
          - /var/run/docker.sock:/var/run/docker.sock
        networks:
          - net
        deploy:
          mode: global
          placement:
            constraints: [node.role == manager]
      postgres-service:
        image: postgres:10
        environment:
          - POSTGRES_USER=scheduler
          - POSTGRES_PASSWORD=schedulerpassword
          - POSTGRES_DB=scheduler
        networks:
          - net
      managementconsole-service:
        image: managementconsole:@productVersion@
        depends_on:
          - postgres-service
        environment:
          - CONTEXT_RESOURCE_VALIDATIONQUERY=SELECT 1
          - CONTEXT_RESOURCE_USERNAME=scheduler
          - CONTEXT_RESOURCE_PASSWORD=schedulerpassword
          - CONTEXT_RESOURCE_DRIVERCLASSNAME=org.postgresql.Driver
          - CONTEXT_RESOURCE_URL=jdbc:postgresql://postgres-service:5432/scheduler
          # enter your license here, or type it through the GUI in first login
          - CONFIG_LICENSE_NAME=
          - CONFIG_LICENSE_EMAIL=
          - CONFIG_LICENSE_COMPANY=@licenseCompany@
          - CONFIG_LICENSE_PRODUCTIONKEY=@licenseProduction@
          - CONFIG_LICENSE_NONPRODUCTIONKEY=@licenseNonProduction@
```

```
            # change to use your own, match with the Roboserver setting
ROBOSERVER_MC_SHARED_SECRET
            -
SERVICE_AUTHENTICATION_ROBOSERVER_SHARED_SECRET=@RoboserverSharedSecret@
          - CONFIG_CLUSTER_JOINCONFIG=multicastCluster
          # change to fit your network
          - CONFIG_CLUSTER_INTERFACE=10.*.*.*
          - CONFIG_CLUSTER_MULTICAST_GROUP=224.2.2.3
          - CONFIG_CLUSTER_MULTICAST_PORT=54327
          - LOG4J_LOGGER_COM_HAZELCAST=ERROR, A
        deploy:
          replicas: 2
          labels:
            traefik.docker.network: weavenet
            traefik.port: 8080
            traefik.backend.loadbalancer.sticky: "true"
            traefik.frontend.rule: PathPrefix:/
        networks:
          - net
      roboserver-service:
        image: roboserver:@productVersion@
        depends_on:
          - postgres-service
        networks:
          - net
        environment:
          - ROBOSERVER_ENABLE_MC_REGISTRATION=true
          - ROBOSERVER_MC_URL=http://loadbalancer/
          - ROBOSERVER_MC_CLUSTER=Production
          # change to use your own, match with the Management Console setting
SERVICE_AUTHENTICATION_ROBOSERVER_SHARED_SECRET
          - ROBOSERVER_MC_SHARED_SECRET=@RoboserverSharedSecret@
          - ROBOSERVER_CONNECTION_TYPE=SocketConnectionType
          - WRAPPER_MAX_MEMORY=2048
```

ファイルをコピーする場合は、必ず元のレイアウトを保持してください。レイアウトが正しくないと、ファイルが無効になる場合があります。

注意:

- **@productVersion@**、**@licenseCompany**、**@licenseProduction@**、**@licenseNonProduction@**、および **@RoboserverSharedSecret@** を適切な値に置き換えます。
- **CONFIG_CLUSTER_INTERFACE** は、Docker swarm の Weave Net サブネットに適合する必要があります。マネージャー ノードで次のコマンドを使用してサブネットを見つけることができます:
  `host1$ docker network inspect weavenet`
- 共通ネットワークの **net**とは、以前に作成した外部ネットワーク **weavenet** を指します。
- **Traefik** はスティッキー セッションのロード バランサーとして使用されます。
- このセットアップは、一時ボリュームで PostgreSQL データベースを実行します。実際の運用セットアップでは、データベースをクラスタ化して永続的なデータ ボリュームで実行する必要もあります。
- RoboServer の複数のインスタンスを実行するには、デプロイの制約を **roboserver-service** に追加します。
- すべてのサービスで、**endpoint_mode:dnsrr** を使用する必要があります (デフォルト設定の **endpoint_mode:vip** は Weave Net プラグインではサポートされていません)

6. 次のように、マネージャノードにサービス スタックをデプロイします。

```
host1$ docker stack deploy -c docker-compose.yml rpa
```

空のデータベースで 2 つのレプリカを使用して **managementconsole-service** を開始すると、競合状態になる可能性があります。このような状況が発生した場合は、**replicas: 2** という行を **replicas: 1** に変更し、前述のコマンドを実行します。サービスが開始されるまで待って行を元に戻してから、前のコマンドを再度実行します。

サービスを停止するには、次のコマンドを使用します。

```
host1$ docker stack down rpa
```

7. スタックをデプロイした後で、次の URL から Management Console にアクセスできます。
   `http://host1/`

高度な構成

LDAP と CA シングル サインオンの統合

このトピックでは、LDAP および CA シングル サインオン認証の使用方法について説明します。

Kofax RPA は LDAPS (LDAP over SSL) もサポートしています。このセクションの後半にある「セキュアな LDAPS」および「LDAPS 使用時の SSL 接続エラーを解決するためのチェックリスト」を参照してください。

ユーザーのオリジンと認証

Management Console にログインしたユーザーは、ユーザー名とオリジンで識別されます。Management Console の [ユーザーおよびグループ] ページの [ユーザーのオリジン] フィールドに、下表のようにユーザー作成方法の情報が示されます。

ユーザーのオリジン	説明
`unknown`	ユーザーはバックアップの復元後に作成されました。
`internal`	ユーザーは [ユーザーおよびグループ] ページで手動で作成されました。
`saml`	ユーザーは SAML 経由でのログイン後に作成されました。
`siteminder`	ユーザーは SiteMinder 経由でのログイン後に作成されました。
`ldap#{ldapDirectoryIdentifier}`	ユーザーは LDAP 経由でのログイン後に作成されました。

ユーザー認証については、次の点に注意してください。

- あるユーザーがログインしたときに、名前が同じでオリジンが `unknown` であるユーザーがすでに存在する場合、新しいユーザーが作成されることはなく、すでに存在するオリジンが新しいログイン方法に基づいて変更されます。

- 外部 ID プロバイダー (SAML、LDAP、SiteMinder など) を使用していない場合、Management Console の [ユーザーおよびグループ] ページでユーザーを選択し [内部オリジンを設定] をクリックすることで、`unknown` オリジンを `internal` オリジンに変更できます。
- `ldapDirectoryIdentifier` は、`login.xml` の必須プロパティです。
- `LdapDirectory` はオプションであり、明示的に割り当てられていない場合は、デフォルト値の 0 (ゼロ) になります。
- `ldapDirectoryIdentifier` が同じ LDAP ディレクトリが複数存在する場合、Management Console は起動しません。
- `login.xml` の `authenticationConfiguration` Bean で `useLdap` オプションと `useSaml` オプションに `true` を指定することで、LDAP 認証と SAML 認証を同時に使用できます。
- LDAP 認証と SAML 認証の両方を使用する場合、Management Console への SSO には SAML が使用され、サービスの認証 (Design Studio への接続など) には LDAP が使用されます。

LDAP 統合

認証に LDAP を使用するには、LDAP 認証を有効にし、`login.xml` ファイルの LDAP 定義を編集します。

LDAP 認証を有効にするには、次のように `useLdap` プロパティを `true` に設定します:

```
<bean id="authenticationConfiguration"
 class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration">
        <property name="useLdap" value="true"/>
        <property name="useSiteMinder" value="false"/>
</bean>
```

`login.xml` には、次の定義があります:

```
<bean id="ldapDirectories" class="com.kapowtech.mc.config.LdapDirectories" lazy-
init="true">
        <property name="directories">
            <list>
                <bean class="com.kapowtech.mc.config.LdapDirectory">
                  <!-- Property defining unique ldap directory name, used as part of
 user's origin field.
                  Must be different for each LdapDirectory -->
                  <property name="ldapDirectoryIdentifier" value="0"/>
                    <!-- List of security groups which will be application
 administrators.
                    Users in these groups will have all permissions. Only users in
 these groups can
                    access the backup tab and create and restore backups -->
                    <property name="adminGroups">
                        <list>
                            <value>KAPOWADMIN</value>
                            <value>ENGINEERING/value>

                   <property name="administratorGroups">
                    <list>
                        <value>RPAADMINISTRATORS</value>
                    </list>
                   </property>
                    </property>
                    <property name="ldapServerURL" value="ldap://
ldap.kapowdemo.com:389"/>
                    <property name="userDn" value="CN=LDAP
 test,CN=Users,DC=kapowdemo,DC=local"/>
                    <property name="password" value="change-me"/>
```

```
                    <property name="userSearchBase"
value="OU=Users,OU=TheEnterprise,DC=kapowdemo,DC=local"/>
                    <property name="userSearchFilter"
value="(userPrincipalName={0}@kapowdemo.local)"/>
                    <property name="userSearchSubtree" value="true"/>
                    <property name="groupSearchBase" value="OU=Security
Groups,OU=TheEnterprise,DC=kapowdemo,DC=local"/>
                    <property name="groupSearchFilter" value="(member={0})"/>
                    <property name="groupRoleAttribute" value="cn"/>
                    <property name="groupSearchSubtree" value="true"/>
                    <property name="allGroupsFilter" value="(cn=*)"/>
                    <property name="fullNameAttribute" value="displayName"/>
                    <property name="emailAttribute" value="userPrincipalName"/>
                    <property name="referral" value="follow"/>
                </bean>
            </list>
        </property>
    </bean>
```

これは、`ldap` という名前の `ldapDirectory` Beans のリストを定義し、LDAP サーバーへの接続リストを表します。Kofax RPA はマルチフォレスト LDAP 統合をサポートしているため、LDAP ディレクトリへの複数の接続を指定できます。各 Bean は、LDAP 統合を制御する多くのプロパティを定義します。Tomcat を LDAP に統合する方法に精通している方は、このことにも詳しいはずです。

グループ名は、リスト内のすべての LDAP サーバーで一意である必要があります。

セキュアな **LDAP**

Kofax RPA は Management Console による LDAPS (SSL の LDAP) をサポートします。LDAPS を使用するには、`ldapServerURL` プロパティを次のように設定する必要があります:

```
<property name="ldapServerURL" value="ldaps://<hostname>:<port>"/>
```

デフォルトでは、LDAPS ポートは 636 です。

展開チェックリスト

プロパティ	説明
`ldapDirectoryIdentifier`	LDAP ディレクトリ名。Management Console の [ユーザーのオリジン] フィールドで使用されます。この名前は、LDAP ディレクトリごとに一意である必要があります。
`adminGroups`	すべてに対するアクセス権を持つ Management Console の admin スーパーユーザーにマップされた LDAP グループのリスト。
`administratorGroups`	Management Console の RPA 管理者にマップされた LDAP グループのリスト。これらの LDAP グループに属するユーザーは、Management Console 内の LDAP グループとロールのマッピングなど、すべてのプロジェクトに対するすべての権限を保持しています (特別な管理者ユーザー権限は除く)。
`ldapServerURL`	LDAP サーバーの URL。これは、ldap:// または ldaps:// プロトコルを使用します。
`userDn`	LDAP にログインして他のユーザーを認証するために使用される DN (識別名)。
`password`	userDN アカウントのパスワード。パスワードはこのファイルにクリア テキストで保存されるため、「読み取り」アクセス権のみを持つアカウントを使用する必要があります。

プロパティ	説明
userSearchBase	ユーザーを検出する LDAP ツリーのサブディレクトリ。
userSearchFilter	ユーザー名を検出するために適用するフィルター。
userSearchSubtree	ユーザーが userSearchBase のサブディレクトリにある場合は、true に設定します。
groupSearchBase	グループを検出する LDAP ツリーのサブディレクトリ。
groupSearchFilter	このグループのユーザーを識別するために適用するフィルター。
groupRoleAttribute	グループ名を保持する属性。
groupSearchSubtree	グループが groupSearchBase のサブディレクトリにある場合は true に設定します
convertToUpperCase	グループ名が大文字に変換される場合は、デフォルトで true になっています。
allGroupsFilter	オプション。プロジェクトの権限を作成するときに表示されるグループを制御します。以下を参照してください。
fullNameAttribute	ユーザーのフルネームを取得するための属性。
emailAttribute	ユーザーのメールを取得するための属性。
referral	LDAP ツリーのサブノードへのリダイレクトを許可するには、「フォロー」に設定します。

LDAP アカウントを使用して Management Console を管理するには、自分が属しているグループの 1 つを、`login.xml` 内の `administratorGroups` Bean に追加する必要があります。手順については、プロジェクト権限を参照してください。`administratorGroups` にリストされているグループのメンバーであれば誰でも、Management Console の管理者になるため、この目的のために新しい LDAP グループの作成が必要になることがあります。`convertToUpperCase` が true の場合、大文字のグループ名を使用します。

Management Console でプロジェクト権限を選択すると、すべてのグループ名が LDAP から取得されてリストに入力されていることがわかります。グループは、`groupRoleAttribute` を使用して、すべてのグループを取得するためのフィルターを構築することにより配置されます。ここにすべての LDAP グループを表示したくない場合があります。その場合、独自のフィルターを指定してこの動作をオーバーライドします。フィルターを指定するには、`LdapLogin.` にさらにプロパティを追加します。

`<property name="allGroupsFilter" value="(cn=*)"/>`: グループ名が cn 属性にある場合、これによってすべてのグループ名が検出されます (これがデフォルトです) 。

文字「e」で始まるグループのみを検索する場合は、次のコードを使用できます

```
<property name="allGroupsFilter" value="(cn=E*)"/>
```

フィルタは基本的な LDAP クエリを使用します。より複雑なクエリについては、LDAP のドキュメントを参照してください。

LDAPS 使用時の SSL 接続エラーを解決するためのチェックリスト

LDAPS を使用した接続エラーが発生した場合は、次を確認してください。

- LDAPS では、LDAP サーバーによって提示された証明書が Tomcat を実行している Java によって信頼されている必要があります。アプリケーションが使用する Java キーストアにパブリック証明書をインポートします。

- JRE または JDK のインスタンスが複数ある場合など、証明書が正しいトラスト ストアにインポートされていることを確認してください。
- 正しいトラスト ストアが使用されていることを確認してください。- `Djavax.net.ssl.trustStore` が設定されている場合、デフォルトのトラスト ストアの位置をオーバーライドします。
- Exchange などのメールサーバーに接続する場合は、認証でプレーン テキストが許可されていることを確認してください。
- ターゲットサーバーが SSL を正しく提供するように設定されていることを確認します。これは、SSL サーバー テスト ツールで実行できます。
- ウイルス対策ツールに SSL および TLS をブロックする「SSL スキャン」があるかどうかを確認します。この機能を無効にするか、ターゲット アドレスに例外を設定します。

CA シングル サインオン統合

Management Console CA シングル サインオンを使用した事前認証をサポートします。CA シングル サインオン では、Management Console にアクセスする前にユーザーの ID が確立され、ユーザーの ID は HTTP ヘッダーを介して送信されます。Management Console がユーザーの LDAP グループ メンバーシップを解決するには、ユーザーの ID は LDAP 識別名の形式である必要があります。

CA シングル サインオン 統合はデフォルトで無効になっています。有効にするには、次のように `login.xml` ファイルで `useSiteMinder` プロパティを `true` に設定します。

```
   <bean
class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration"
id="authenticationConfiguration">
       <property name="useLdap" value="false"/>
       <property name="useSiteMinder" value="true"/>
   </bean>
```

```
   <bean class="com.kapowtech.mc.config.SiteMinderConfiguration"
id="siteMinderConfiguration">
       <property name="headerName" value="sm_userdn"/>
       <property name="accountAttribute" value="sAMAccountName"/>
       <property name="accountAttributePattern" value="(.*)"/>
   </bean>
```

CA シングル サインオン 統合を有効にした後、ユーザーの識別名を含む HTTP ヘッダーの名前を指定します。`accountAttribute` プロパティは、ユーザーのどの LDAP 属性がアカウント名として使用されるかを識別します (デフォルトでは `sAMAccountName` となっており、これはユーザーの Windows ログイン名です) 。`accountAttributePattern` プロパティが、アカウント名が属性値からどのように解析されるかを指定します。この属性値は正規表現 (アカウント名を識別する単一の括弧のセット) である必要があり、`(.*)` の値 `accountAttributePattern` プロパティは、属性内のすべてを意味します。設定内のユーザーの電子メール アドレスからアカウント名を抽出するには、次を指定します。

```
   <bean class="com.kapowtech.mc.config.SiteMinderConfiguration"
id="siteMinderConfiguration">
       <property name="headerName" value="sm_userdn"/>
       <property name="accountAttribute" value="userPrincipalName"/>
       <property name="accountAttributePattern" value="([^@]*)@.*"/>
   </bean>
```

`([^@]*)@.*` は電子メール アドレスと一致し、アカウント名として `@` の前にすべてを抽出します。

CA シングル サインオンでは LDAP ログイン設定の一部が使用されるため、`administratorGroups` Bean にユーザー グループを追加してから、Management Console の設定を開始する必要があります。

システムからログアウトすることはできません。CA シングル サインオン ヘッダーの存在は、常に認証されることを意味するためです。ログアウトするには、ブラウザを閉じます。

制限事項

CA シングル サインオンの統合は、ブラウザから Management Console がアクセスされた場合にのみ機能します。ただし、Management Console がブラウザではないアプリケーションによってアクセスされる場合、CA シングル サインオン認証メカニズムは使用されず、そのようなサービスには Management Console で設定された資格情報 (ユーザー名とパスワード) のセットが必要です。これらのクライアントには、次のものが含まれますが、これらに限定されません:

Design Studio

ロボット開発者が開発者シート ライセンスを取得し、ロボットをアップロードして、Management Console に格納されているデータベース設定およびその他の設定を入手するには、Management Console にアクセスする必要があります。これには次の URL へのアクセスが必要です (Management Console がデプロイされるコンテキスト パスに対して)。

- /License/*
- /secure/*
- /IDESettings/*
- /rest/*
- /ws/*

上記の URL へのアクセスは、Management Console で設定されるパスワードの基本認証により保護されています。

REST サービス

REST サービスはデフォルトで基本認証によって保護されていますが、ロボットは認証なしで REST サービスとして公開できます。このようなサービスは通常、外部アプリケーションによって呼び出されます。RESTは /rest/* URL を使用します。

RoboServer

RoboServer は、クラスタへの登録とリソース要求に共有シークレットを使用して認証を行います。

Desktop Automation サービス

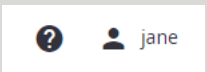
Desktop Automation サービスは、共有シークレットまたは Management Console を使用して登録とステータス更新の認証を行います。

Kofax RPAJava または **.Net API** に基づくアプリケーション

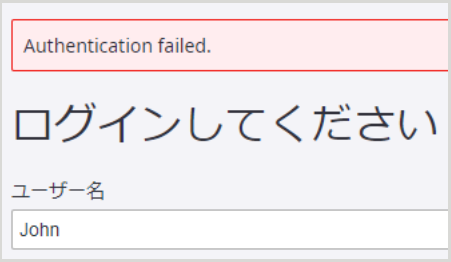
Kofax RPA Java または .Net API に基づくアプリケーションは、Management Console アクセス時に基本認証によって保護されます。

例: 使用方法

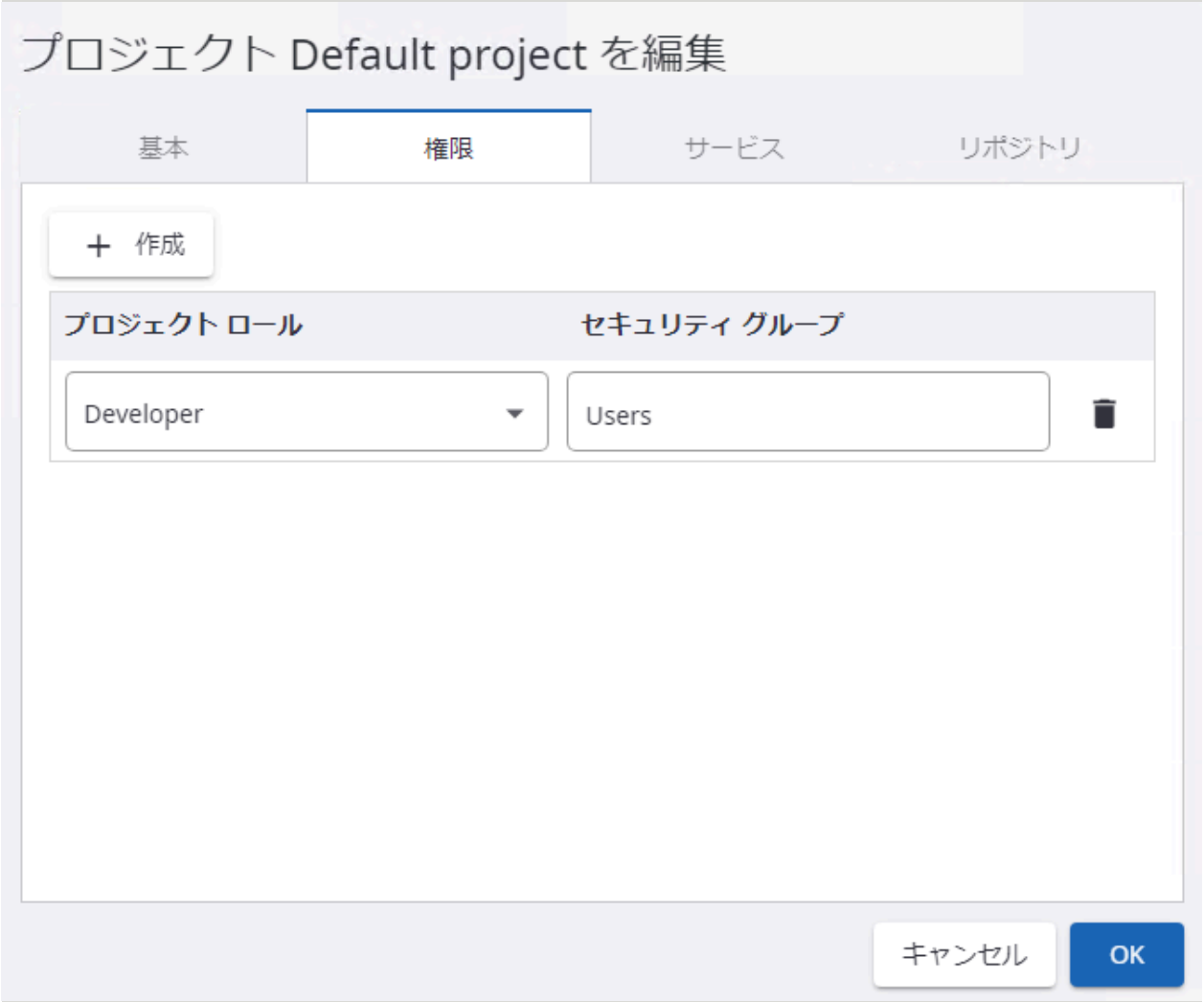
- ユーザー「Jane」は Management Console 管理者として指定されています。このユーザーは Active Directory (AD) の [**Administrators**] グループに追加されているため、このグループは [`login.xml`] に記載されており、CA シングル サインオンでブラウザを認証するとすぐに Management Console にログインします。



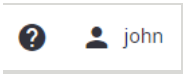
- **John** は AD 内の ユーザー グループのメンバーであるロボット開発者であり、現在は、Management Console にログインしようとしても拒否されます。



John がログインする前に、Jane は Management Console > [管理] > [プロジェクト] で、選択したプロジェクトの ユーザー グループに権限を割り当てる必要があります。たとえば、次に示すように、ユーザー グループに開発者ロールを割り当てることができます。



これで「John」は Management Console CA シングル サインオンにログインできるようになりました。



John は Design Studio を起動するときに、[ライセンス情報の入力] ダイアログ ボックスに必要な URL を入力し、開いているブラウザで Management Console を認証してライセンスを取得する必要があります。

Jane は、Management Console にログインできないローカル サービス ユーザーも作成できます (RoboServer サービスや Desktop Automation サービスのユーザーなど)。次のことに注意してください:

- サービス ユーザー用に選択するグループは、Active Directory からのみ生成する必要があります。
- このグループには、Management Console で設定した適切な権限が必要です。

SAML シングル サインオンの統合

Management Console はSAML シングル サインオンを使用したユーザー事前認証をサポートします。

次の手順は、ID プロバイダーで SAML アプリケーションをすでに作成および設定していることを前提に書かれています。設定手順の例については、OneLogin 設定の例を参照してください。

ライセンスがない場合、SAML 統合の Management Console では開始できません。アプリケーションを使用する前に有効なライセンスをインストールします。

SAML シングル サインオンでの Management Console 統合は、デフォルトで無効になっています。有効にするには、`login.xml` ファイルで `useSaml` プロパティを探して `true` に設定します。

`login.xml` の `authenticationConfiguration` Bean で `useLdap` オプションと `useSaml` オプションに `true` を指定することで、LDAP 認証と SAML 認証を同時に使用できます。詳細については、「ユーザーのオリジンと認証」を参照してください。

```
 <bean
class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration"
id="authenticationConfiguration">
   <property name="useLdap" value="false"/>
   <property name="useSiteMinder" value="false"/>
  <property name="useSaml" value="true"/>
```

`login.xml` で統合を有効にした後、ID プロバイダーとサービス プロバイダーを設定する必要があります。`\WEB-INF\spring` フォルダにある `saml.xml` ファイルで、セットアップに合わせて次のプロパティを変更します。

`login.xml` および `saml.xml` ファイルを変更した後に、Tomcat を再起動します。

プロパティ	説明
`assignGroups`	`false` に設定すると、管理者は手動でユーザーをグループに割り当て、ユーザーが Management Console にアクセスできるようにします。デフォルトは `true` です。
`forceAuthN`	`true` に設定すると、ID プロバイダーはユーザーを再認証し、以前の認証イベントに依存しません。デフォルトは `false` です。

プロパティ	説明
groupsAttributes	グループ化属性。Management Console へのユーザー アクセスは特定のユーザーが属するグループによって管理されます。ID プロバイダーでユーザーに割り当てられているグループのリストを示す SAML アサーション メッセージの属性名に一致する名前を、1 つまたはリストで指定します。
adminGroups	管理者グループ。すべてに対するアクセス権を持つ Management Console の admin スーパーユーザーにマップされた ID プロバイダー グループのリスト。
administratorGroups	管理者グループ。このグループのユーザーは、すべての RPA プロジェクトに対する管理者権限を取得します。また、SAML を使用してカスタム管理者グループを作成することもできます。ユーザー ロールとグループの詳細については、『Kofax RPA 管理者ガイド』の「定義済みのユーザー ロール」を参照してください。
email	これは、ユーザーの電子メールを含む SAML 応答からの属性の名前です。
firstname	これは、ユーザーの名を含む SAML 応答からの属性の名前です。
lastname	これは、ユーザーの姓を含む SAML 応答からの属性の名前です。
idpName	ID プロバイダーの名前。可能な値は、OKTA、ONELOGIN、および AZURE です。それ以外の場合は、DEFAULT に設定します。
idpGroupNameSeparator	ID プロバイダーのグループ名を区切るために使用する区切り文字。idpName で指定した ID プロバイダーが OneLogin である場合にのみ有効になります。それ以外の場合、このプロパティは無視されます。
idpUserNameRegex	一部のユーザー名に特殊文字が含まれている場合は、^[\p{L}0-9]*$ パターンを追加して、既存のパラメータを更新します。
entityId	Management Console の URL に saml/login を加えたもの。この URL は SAML アプリケーションから取得できます。例: http://localhost:8080/ManagementConsole/saml/login
entityBaseURL	Management Console のベース URL。例: http://localhost:8080/ManagementConsole
maxAuthenticationAge	秒単位のシングル サインオンの有効性。ユーザーが最初に IDプロバイダーで認証された後にサインインできる時間枠。デフォルトでは 86400 秒、つまり 24 時間です。 このプロパティを使用して、デフォルトを変更します。
responseSkew	ID プロバイダー サーバーと Management Console が展開されているコンピュータ間の時刻を比較するための秒単位の不正確さの許容誤差。時刻の同期が完全に一致しない可能性があるため、デフォルトでは 60 秒が適用されます。 デフォルトの値を変更するには、このプロパティを使用します。
maxAssertionTime	シングル サインオン中に処理されたアサーションの妥当性。アサーション時間が設定された制限に達すると、認証は無効になります。デフォルトでは 6000 秒、つまり 100 分に制限されています。 このプロパティを使用して、デフォルトを変更します。

プロパティ	説明
`HTTPMetadataProvider` Bean の `java.lang.String`	ID プロバイダーのメタデータへの URL。この URL は SAML アプリケーションから取得できます。例: `http://example.okta.com/saml/metadata/222670a0-2b96-48ef-975a-b7267446d09e` Microsoft Azure などの一部の ID プロバイダーは、このプロパティを必要としません。
`FilesystemMetadataProvider` Bean の `java.io.File`	ID プロバイダーのメタデータを含む XMLファイルへのパス。ID プロバイダーがリアルタイムでメタデータの読み取りを許可していない場合、このプロパティを使用します。 OneLogin などの一部の ID プロバイダーは、このプロパティを必要としません。
`useSamlSingleLogout`	デフォルトは `false` です。 `false` に設定した場合は、Management Console からログアウトした後に Management Console セッションが破棄され、ログアウトが成功したことを示すメッセージが表示されます。 `true` に設定した場合は、Management Console で **[ログアウト]** をクリックしたときに、ユーザーは ID プロバイダの SAML アプリケーションからログアウトします。

サンプルの `saml.xml` ファイルの次のスニペットには、設定する必要があるプロパティが含まれています。

```
<bean id="samlEntryPoint" class="org.springframework.security.saml.SAMLEntryPoint"
 lazy-init="true">
    <property name="defaultProfileOptions">
        <bean class="org.springframework.security.saml.websso.WebSSOProfileOptions">
            <property name="includeScoping" value="false"/>
            <property name="forceAuthN" value="false"/>
            <property name="passive" value="false"/>
        </bean>
    </property>
    <property name="filterProcessesUrl" value="/saml/entry"/>
</bean>
```

```
<bean id="samlAuthenticationProvider" class="com.kapowtech.scheduler.server.spring.sec
urity.GroupProvidingSAMLAuthenticationProvider" lazy-init="true"> <constructor-arg ref
="platformEMF"/>
    <property name="internalAuthenticationProvider"
 ref="internalAuthenticationProvider"/>
    <property name="customerNameMapper" value="false"/>
    <property name="customerNameMapperIdentifier" value=""/>
    <property name="groupsAttributes">
        <list>
            <value>groups</value>
        </list>
    </property>
    <property name="adminGroups">
        <list>
            <value>KapowAdmins</value>
        </list>
    </property>
    <property name="assignGroups" value="true"/>
    <property name="consumer" ref="webSSOprofileConsumer"/>

    <property name="email" value="email"/>
    <property name="firstname" value="firstname"/>
    <property name="lastname" value="lastname"/>
```

```
    <property name="idpName" value="ONELOGIN"/>
    <property name="idpGroupNameSeparator" value=";"/>
    <property name="idpUserNameRegex" value="^[a-zA-Z0-9]*$"/>
    <property name="idpEmailRegex" value="^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$"/>
</bean>
```

```
<bean id="useSamlSingleLogout" class="java.lang.Boolean">
      <constructor-arg value="false"/>
</bean>
```

```
<bean id="metadataGeneratorFilter"
 class="org.springframework.security.saml.metadata.MetadataGeneratorFilter">
    <constructor-arg>
        <bean class="org.springframework.security.saml.metadata.MetadataGenerator">
            <property name="entityId" value="http://localhost:8080/ManagementConsole/
saml/login"/>
            <property name="requestSigned" value="false"/>
            <property name="entityBaseURL" value="http://localhost:8080/
ManagementConsole"/>
               <property name="extendedMetadata">
                <bean
 class="org.springframework.security.saml.metadata.ExtendedMetadata">
                    <property name="idpDiscoveryEnabled" value="false"/>
                    <property name="signMetadata" value="false"/>
                </bean>
            </property>
        </bean>
    </constructor-arg>
</bean>
```

```
<bean id="webSSOprofileConsumer"
 class="org.springframework.security.saml.websso.WebSSOProfileConsumerImpl">
        <property name="maxAuthenticationAge" value="86400"/>
        <property name="responseSkew" value="600"/>
        <property name="maxAssertionTime" value="6000"/>
    </bean>
```

```
<bean id="metadata"
 class="org.springframework.security.saml.metadata.CachingMetadataManager">
    <constructor-arg>
        <list>
           <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider"
 lazy-init="true">
                <constructor-arg>
                    <value type="java.lang.String">http://example.okta.com/saml/
metadata/222670a0-2b96-48ef-975a-b7267446d09e</value>
                </constructor-arg>
                <constructor-arg>
                    <value type="int">10000</value>
                </constructor-arg>
                <property name="parserPool" ref="parserPool"/>
            </bean>

            <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                    <constructor-arg>
                        <value type="java.io.File">classpath:security/idp.xml</value>
                    </constructor-arg>
                    <property name="parserPool" ref="parserPool"/>
                </bean>
        </list>
    </constructor-arg>
```

```
</bean>
```

OneLogin 設定の例

このセクションでは、Kofax RPA で使用するために OneLogin ID プロバイダで SAML アプリケーションを設定する手順の例を示します。

1. OneLogin Web サイトで、アカウントを作成します。
2. アカウントの作成後に、<使用するドメイン名>.onelogin.com ページを開き、[Administration] ページを開きます。
3. メニューで、[**APPS**] をクリックして、新しいアプリケーションを追加するためのオプションを選択します。次のアプリケーション タイプを選択します: **SAML Test Connector (Advanced)**。
4. [**Configuration**] タブで、次の表に示すようにアプリケーション パラメータを設定します。他のパラメータはそのままにしておきます。

パラメータ	値
RelayState	http://[IP アドレス]:8080/ManagementConsole/saml/login
Audience	http://[IP アドレス]:8080/ManagementConsole/saml/login
Recipient	http://[IP アドレス]:8080/ManagementConsole/saml/login
ACS (Consumer) URL Validator	http://[IP アドレス]:8080/ManagementConsole/saml/login
ACS (Consumer) URL	http://[IP アドレス]:8080/ManagementConsole/
Single Logout URL	http://[IP アドレス]:8080/ManagementConsole/saml/SingleLogout
Login URL	http://[IP アドレス]:8080/ManagementConsole/saml/login
SAML Initiator	OneLogin
nameID format	Email
issuer type	Specific
signature element	Response
encryption method	TRIPLEDES-CBC

5. [**Parameters**] タブで、次の表に示すようにアプリケーション パラメータを設定します。

パラメータ	値
NameID	Email name part
email	Email
firstname	First Name
group	ユーザー ロール
lastname	Last Name

また、[**Include in SAML assertion**] オプションを選択します。

6. [**SSO**] タブには、Kofax RPA `saml.xml` ファイルの構成に必要な発行者の URL が含まれています。
   [**X.509 Certificate**] が [**Standard Strength Certificate (2048-bit)**] に設定されており、[**SAML Signature Algorithm**] が [**SHA-1**] に設定されていることを確認します。
   [**Issuer URL**] のプロパティ値をコピーして、`saml.xml` ファイルの「IDP Metadata configuration」セクションで使用します。例については、以下を参照してください。
7. メニューで、[**USERS**] > [**Roles**] をクリックし、新しいロールを追加するオプションを選択します。
   Management Console グループに対応するロールを追加し、前に作成したアプリケーションを割り当てます。Kofax RPA では [**KapowAdmins**] ロールが必須であるため、必ず追加してください。
8. メニューで、[**USERS**] > [**All users**] をクリックし、このユーザーに必要なロールを選択します。
   これで、OneLogin アプリケーションの設定が完了しました。
9. 次に、フォルダ `\WEB-INF\spring` にある `saml.xml` でグループ属性を設定して、作成した OneLogin アプリケーションと一致させる必要があります。
   ファイルの変更後に、Tomcat を再起動します。
   例

```
        <property name="groupsAttributes">
            <list>
                <value>group</value>
            </list>
        </property>
        <property name="adminGroups">
            <list>
                <value>KapowAdmins</value>
            </list>
        </property>
...

<property name="idpName" value="ONELOGIN"/>

...

    <bean id="metadataGeneratorFilter"
 class="org.springframework.security.saml.metadata.MetadataGeneratorFilter" lazy-
init="true">
        <constructor-arg>
            <bean
 class="org.springframework.security.saml.metadata.MetadataGenerator">
                <property name="entityId" value="http://<IP_address>:8080/
ManagementConsole/saml/login"/>
                <property name="requestSigned" value="false"/>
                <property name="entityBaseURL" value="http://<IP_address>:8080/
ManagementConsole"/>

                <property name="extendedMetadata">
                    <bean
 class="org.springframework.security.saml.metadata.ExtendedMetadata">
                        <property name="idpDiscoveryEnabled" value="false"/>
                        <property name="signMetadata" value="false"/>
                    </bean>
                </property>
            </bean>
        </constructor-arg>
```

```
    </bean>

...

    <!-- IDP Metadata configuration - paths to metadata of IDPs in circle of trust
 is provided here
    -->
    <bean
 class="org.springframework.security.saml.metadata.CachingMetadataManager"
 id="metadata" lazy-init="true">
        <constructor-arg>
            <list>

                <!--  <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                    <constructor-arg>
                        <value type="java.io.File">classpath:security/idp.xml</
value>
                    </constructor-arg>
                    <property name="parserPool" ref="parserPool"/>
                </bean> -->
                <bean
 class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider" lazy-
init="true">
                    <constructor-arg>
                        <value type="java.lang.String">https://
app.onelogin.com/saml/metadata/d237b5c5-b110-4f42-a646-7678ae08feae</value>
                    </constructor-arg>
                    <constructor-arg>
                        <value type="int">10000</value>
                    </constructor-arg>
                    <property name="parserPool" ref="parserPool"/>
                </bean>
                <!--  <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                    <constructor-arg>
                        <value type="java.io.File">classpath:security/idp.xml</
value>
                    </constructor-arg>
                    <property name="parserPool" ref="parserPool"/>
                </bean> -->
            </list>
        </constructor-arg>
    </bean>
```

高可用性

高可用性 (フェイルオーバー) が必要な場合は、クラスタとして連携するように複数の Management Console インスタンスを設定できます。完全なフェールオーバーを実現するには、次のコンポーネントをクラスタ化する必要があります。

クラスタ コンポーネント

コンポーネント	説明
ロード バランサー	HTTP ロード バランサーは、複数の Tomcat サーバー間でリクエストを分散するために必要です。 高可用性モードを設定する場合は、RoboServer や Kapplet などの他のすべてのサービスは、Management Console に直接接続するのではなく、ロード バランサーを介して Management Console にアクセスするように設定する必要があります。
クラスタ化されたプラットフォーム データベース	Management Console のプラットフォーム データベースには、スケジュール、ロボットなどが格納されます。フェイルオーバー シナリオでは、単一障害点を回避するために、プラットフォーム データベースをクラスタ化された DBMS で実行する必要があります。
Tomcat セッション レプリケーション	Management Console はユーザーのセッションにデータを直接保存しませんが（インポート/エクスポート時を除く）、セッションはユーザーの認証情報を保持します。 セッション レプリケーションが有効になっていない場合、現在接続している Tomcat がクラッシュした場合、ユーザーは再度ログインする必要があります。
Apache Tomcat コネクタ	mod_jk は、Apache、iPlanet、Sun ONE（以前の Netscape）などの Web サーバー、および Apache Jserv プロトコルを使用する IIS に Tomcat サーブレットコンテナを接続するために使用される Apache モジュールです。
Hazelcat	Hazelcast（www.hazelcast.com）は、複数の JVM でデータ構造をクラスタ化するために使用されます。Management Console 内で、これは重要なデータ構造のクラスタリングを提供し、アプリケーション インスタンス間の相互通信を提供するために使用されます。 以下に例を示します: ロボットを RoboServer 上で実行するとき、RoboServer によって返されるステータス メッセージを処理するにはスレッドが必要です。このスレッドは、特定の Tomcat インスタンス内で実行されます。クラスタ環境では、ロボットを停止しようとするユーザーが実際に、ロボットを実行しているインスタンス以外の Tomcat インスタンスで停止要求を生成している可能性があります。その場合、停止要求は Hazelcast を介してすべてのインスタンスに送信され、ロボットを実行しているインスタンスはそれを受信し、ロボットを停止するように動作します。

複数 Management Console インスタンス

2 つ以上の同一の Tomcat インストールがあり、それらのすべてに同じバージョンの `ManagementConsole.war` をデプロイする必要があります。`web.xml`、`Configuration.xml`、`login.xml`、および `roles.xml` ファイルがすべてのインスタンスで同じであることを確認してください。

コンポーネントのインストールと設定

このトピックでは、マルチキャスト クラスタリングを使用して高可用性設定に必要なコンポーネントをインストールおよび設定する方法について説明します。この設定では、2 台のホスト コンピューターをセットアップします。1 台のホスト（host1）には Tomcat サーバーとデータベースが含まれ、もう 1 台のホスト（host2）には Tomcat サーバーとロード バランサーとしての Apache サーバーが含まれます。

ステップ バイ ステップの手順

次の手順は、高可用性設定のためのコンポーネントのインストールに役立ちます。

1. 「host1」コンピューターにデータベースをセットアップします。
2. Apache Web サイトから Tomcat をダウンロードします: https://tomcat.apache.org
3. Tomcat を両方のホストにインストールし、ユーザー パスワードを設定します。詳細については、「Tomcat の展開 」を参照してください。
4. 両方のホストの Tomcat で Management Console をインストール。
5. 両方のコンピューターで Tomcat アプリケーションを起動し、それらがオンラインになることを確認します。2 つの同一の Tomcat インストールがあり、それらのすべてに同じバージョンの ManagementConsole.war をデプロイする必要があります。`web.xml`、`Configuration.xml`、`login.xml`、および `roles.xml` ファイルが両方のインストールで同じであることを確認してください。
6. Tomcat サーバーをシャットダウンします。
7. Apache Web サイトから Apache サーバーをダウンロードします: http://httpd.apache.org/download.cgi#apache24)「host2」に Apache サーバーをインストールし、Apache サービスを開始します。詳細については、Apache のドキュメントを参照してください: http://httpd.apache.org/docs/
8. Apache Web サイトから Apache `mod_jk` コネクターをダウンロードします: https://tomcat.apache.org/download-connectors.cgi。
   - ファイルをディスク上のディレクトリに解凍します。
   - `mod_jk.so` ファイルを `<apache>\module` ディレクトリにコピーします。
   - 次のように `<apache>\conf\httpd.conf` を編集します:

     ```
     LoadModule jk_module modules/mod_jk.so
     <IfModule mod_jk.c>
      JkWorkersFile "<apache>\conf\workers.properties"
      JkLogFile "<apache>\logs\mod_jk.log"
      JkLogLevel error
      JkLogStampFormat "[%a %b @d %H:%M:%S %Y] "
      JkRequestLogFormat "%w %V %T"
     </IfModule>
     JkMount /ManagementConsole/* loadbalancer
     JkMount /ManagementConsole loadbalancer
     ```

   - 次の内容で `<apache>\conf\workers.properties` ファイルを作成します:

     ```
     worker.list=host1, host2, loadbalancer

     worker.host1.host=<ip address or host name>
     worker.host1.port=8009
     worker.host1.type=ajp13
     worker.host1.lbfactor=1
     worker.host2.host=<ip address or host name>
     worker.host2.port=8009
     worker.host2.type=ajp13
     worker.host2.lbfactor=1
     worker.loadbalancer.type=lb
     worker.loadbalancer.balance_workers=host1, host2
     ```

     `<ip address or host name>` は、Tomcat を実行しているホスト コンピュータのアドレスまたはホスト名です。
9. 両方の Tomcat サーバーについて、次の行を `conf\server.xml` に入力します。

```
<Connector protocol="AJP/1.3"
address=<host name or IP address>
port="8009"
redirectPort="8443"
secretRequired="false" />
<Engine name="Catalina" defaultHost="localhost" jvmRoute="<host number from
 workers.properties>">
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

詳細については、Tomcat セッション レプリケーションを参照してください。

10. 各 Tomcat サーバーで、`webapps\ManagementConsole\WEB-INF\Configuration.xml` ファイルを次のように編集します。ネットワーク上のホスト コンピューターの有効な IP アドレスを指定する必要があることに注意してください。

```
<!-- Cluster configuration -->
<bean id="cluster" class="com.kapowtech.mc.config.ClusterConfig" >
<property name="port" value="5701"/>
<!-- In "<mask for the IP address>"/>, insert the values, such as "192.168.1.*" or
 "10.10.10.*"-->
<property name="interface" value="<mask for the IP address>"/>
<!-- Uncomment the line below to enable clustering via multicast. Your license
must support High Availability for this to work -->
<!--property name="joinConfig" ref="multicastCluster"/> -->
<!-- or uncomment this line to enable clustering via TCP-IP. Your license must
support High Availability for this to work-->
<property name="joinConfig" ref="tcpCluster"/>
<property name="managementCenterUrl" value=""/>
</bean>
 <!-- definition for a TCP cluster. You need to add peers to this list, so each
 client can locate at least one other functioning cluster member -->
 <bean id="tcpCluster" class="com.kapowtech.mc.config.TcpJoinConfig" lazy-
init="true">
 <property name="peers">
 <list>
 <bean class="com.kapowtech.mc.config.TcpPeer">
 <property name="host" value="<ip address or host name>"/>
 <!-- port is only needed if the other machine is not using the same port as this
 instance-->
 <!--property name="port" value="5701"/-->
 </bean>
 <bean class="com.kapowtech.mc.config.TcpPeer">
 <property name="host" value="<ip address or host name>"/>
 <!-- port is only needed if the other machine is not using the same port as this
 instance-->
 <!--property name="port" value="5701"/-->
 </bean>
 </list>
 </property>
 </bean>
```

詳細については、「Hazelcast の複製」を参照してください。

これで、`<host2>:80/ManagementConsole` に移動することで、ロード バランサー上で Management Console にログインできます。ここでは「host2」は Apache サーバーを実行しているコンピューターの名前です。ログイン後に **[管理]** > **[高可用性ノード]** に移動すると、正しい IP アドレスが設定された 2 つのノードが表示されます。

高可用性モードを設定する場合は、RoboServer や Kapplets などの他のすべてのサービスは、Management Console に直接接続するのではなく、ロード バランサーを介して Management Console にアクセスするように設定する必要があります。

ロード バランサーの起動

このセクションでは、アプリケーションが正常に起動したかどうかを判断する方法について説明します。

`ManagementConsole.xml`（コンテキスト設定）または `web.xml` ファイルが無効な場合、アプリケーションは Tomcat にデプロイできず、リクエストは通常エラー コード 404 を返します（`/ManagementConsole/` にデプロイされていない Tomcat の ROOT アプリケーションにヒットするため）。

アプリケーションの起動中に発生したその他のエラーは、アプリケーションのロード時にユーザーに表示されます。この方法では、アプリケーションを正しくロードできなかった理由を把握するために必ずしもログを確認する必要はありません。ただし、起動中にエラーが発生してもアプリケーションが 200 OK を返すため、これは実用的であるとは言えません。また、認証が有効になっている場合は、エラー メッセージを表示する前にログインする必要があります。

ロード バランサーがアプリケーションが正しく起動したかどうかを確認しやすくするために、URL `/ManagementConsole/Ping` へリクエストを送信できます。これにより、アプリケーションが正しくロードされた場合は HTTP ステータス コード 200 が返され、エラーのスタック トレースがある場合は 500 が返されます。

Tomcat セッション レプリケーション

セッション複製は `/conf/server.xml` で設定されます。Tomcat でのインスタンス検出にマルチキャストを使用する例を以下に示します。

```
<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
          managerClassName="org.apache.catalina.cluster.session.DeltaManager"
          expireSessionsOnShutdown="false"
          useDirtyFlag="true"
          notifyListenersOnReplication="true"
          printToScreen="true">

        <Membership
            className="org.apache.catalina.cluster.mcast.McastService"
            mcastAddr="228.0.0.4"
            mcastPort="45564"
            mcastFrequency="500"
            mcastDropTime="3000"/>

        <Receiver
            className="org.apache.catalina.cluster.tcp.ReplicationListener"
            tcpListenAddress="auto"
            tcpListenPort="4002"
            tcpSelectorTimeout="100"
            tcpThreadCount="6"/>

        <Sender
            className="org.apache.catalina.cluster.tcp.ReplicationTransmitter"
            replicationMode="pooled"
            ackTimeout="150000"
```

```
            waitForAck="true"/>

        <Valve className="org.apache.catalina.cluster.tcp.ReplicationValve"
               filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.htm;.*\.html;.*\.css;.*
\.txt;"/>

        <Deployer className="org.apache.catalina.cluster.deploy.FarmWarDeployer"
                  tempDir="/tmp/war-temp/"
                  deployDir="/tmp/war-deploy/"
                  watchDir="/tmp/war-listen/"
                  watchEnabled="false"/>

        <ClusterListener
 className="org.apache.catalina.cluster.session.ClusterSessionListener"/>
    </Cluster>
```

また、`server.xml` の <Engine> 要素で jvmRoute 属性を設定する必要があります:

```
<Engine jvmRoute="tomcat2" name="Catalina" defaultHost="MyHost">
```

mod_jk を簡易的なロード バランサーとして使用している場合は、jvmRoute の値を mod_jk 構成による `workers.properties` ファイル参照にリストされている名前と一致させる必要があります。

詳細については、Tomcat のドキュメントを参照してください。

Hazelcast の複製

最も基本的な Hazelcast 設定は `Configuration.xml` で編集できますが、SSL 暗号化などのより高度な設定は `/WEB-INF/Hazelcast.xml` で設定する必要があります

Management Consoleを開始すると、ポート 5701 (または 5701 が利用できない場合は次に利用可能なポート) に Hazelcast ノードが作成されます。デフォルトで、この Hazelcast ノードは IP アドレス 127.0.0.1 にバインドします。バインド アドレスをパブリック IP/ホスト名に変更しないと、クラスタに参加できません。これは、`Configuration.xml`で クラスタ Bean のインターフェイス プロパティを変更することにより行われます。次のようになります:

```
    <bean id="cluster" class="com.kapowtech.mc.config.ClusterConfig" >
        <property name="port" value="26000"/>
        <property name="interface" value="10.0.0.*"/>
    .......
    </bean>
```

* はワイルドカードとして使用されます。この場合、アプリケーションは 10.0.0 で始まる IP アドレスを持つ「最初」のインターフェイスにバインドしようとします。インターフェイスへのバインドは可能ですが、最終的に 127.0.0.1 または別の仮想インターフェイスにバインドされる可能性があるため、*.*.*.* の使用は推奨されません。

Management Console の追加インスタンスを開始するとき、Hazelcast インスタンスは既存の Hazelcast ノードを見つけてクラスタに参加しようとします。この検出は、マルチキャストまたは TCP/IP を介して実行できます。

マルチキャスト検出を使用するには、`Configuration.xml` のクラスタ Bean を変更する必要があります。これは、次の行のコメントを外して行われます:

```
<property name="joinConfig" ref="multicastCluster"/>
```

multicastCluster は、マルチキャスト グループとポートを定義する multicastCluster Bean への参照です。ネットワーク トポロジに合わせて変更できます。

ネットワークでマルチキャストが許可されていない場合は、tcpCluster を使用する必要があります。その代わりに、この行のコメントを解除することにより行われます:

```
<property name="joinConfig" ref="tcpCluster"/>
```

tcpCluster Bean には、他の Hazelcast ノードごとに 1 つずつ、TcpPeer のリストが含まれています。すべての Hazelcast ノードに同じ TCP ポートを使用する場合、ポート番号を指定する必要はありません (各ノードは、ピアが自身と同じポートで実行されていると想定します)。TCP クラスタに 2 つのノードが設定されている場合は、次のようになります。

```
    <bean id="tcpCluster" class="com.kapowtech.mc.config.TcpJoinConfig">
         <property name="peers">
             <list>
                 <bean class="com.kapowtech.mc.config.TcpPeer">
                     <property name="host" value="10.0.0.25"/>
                 </bean>
                 <bean class="com.kapowtech.mc.config.TcpPeer">
                     <property name="host" value="10.0.0.26"/>
                 </bean>
             </list>
         </property>
     </bean>
```

両方のノードがリストにあることに注意してください。これは、どのノードが最初に起動するかに関係なく、ピアを見つけることができることを意味します。また、両方のアプリケーションで同一の `Configuration.xml` ファイルを使用できます。また、TCP ポート番号は定義されていないため、各ピアはそのピア自体でリッスンしている場合と同じポート上の他のピアに接続しようとします。

HA サーバー ノード

アプリケーションが適切にクラスタ化されていることを確認するには、**[管理]** > **[高可用性ノード]** に移動します。

[インターフェイス] 列に、Hazelcast がクラスタ間の通信に使用している IP/ホストとポートが一覧表示されます。[接続先] 列には、2 つのノードのうち、現在接続しているノードが表示されます。現在接続しているサーバーをシャットダウンすると、ロード バランサーによって別のライブ インスタンスに自動的に再ルーティングされます。

ノードのコンテキスト メニューで、デバッグに役立つ可能性があるスレッド ダンプを要求できます。

URI エンコーディング

デンマーク語の ÆØÅ、ドイツ語の ß などの非 ASCII 文字を含む名前のロボットをリポジトリにアップロードする場合、Web コンテナーの URI エンコードを UTF-8 に設定する必要があります。

Tomcat では、これは `/conf` フォルダ内の `server.xml` ファイルにある <connector> の定義で行います。ここで、以下のように属性 URIEncoding="UTF-8" を追加します:

```
<Connector port="8080" URIEncoding="UTF-8"...../>
```

パスワード暗号化

Management Console はパスワードを保存するときに、証明書ベースの (公開鍵、秘密鍵) 暗号化を使用します。以前のバージョンからインポートすると、新しい証明書ベースのアルゴリズムを使用してパスワードが自動的に再暗号化されます。

証明書と一致する秘密鍵は Java キーストアに保存されますが、Management Console にはデフォルトの証明書と秘密鍵を含むキーストアが付属しています。すべての顧客が同じキーストアを取得するため、独自のキーストアを作成することをお勧めします。作成しない場合、誰でもエクスポートをロードしてパスワードを取得できる可能性があります。

独自のキーストアを作成する

Management Console をすでに開始している場合、証明書をアップグレードする必要があります。キーストアは pkcs12 形式である必要があり、Java SDK に付属の keytool アプリケーションを使用して作成できます (Java SDK は Oracle.com からダウンロードできます)。次のコマンドは、365 日間有効な証明書を使用して新しい pkcs 12 キーストアを作成します。

```
keytool -genkey -alias mc -keyalg RSA -validity 3650 -keystore mc.p12 -
storetype pkcs12
```

パスワードと、X.509 秘密鍵に保存される情報の入力を求められます。このコマンドは、現在のディレクトリにファイル mc.p12 (-keystore 引数の値) を作成します。-validity 3650 は、証明書が 10 年間有効であることを意味します。

pkcs12 は秘密鍵と公開証明書を保持するため、認証局 (CA) によって発行された証明書の使用はお勧めしません。秘密鍵へのパスワードはアプリケーション構成の一部としてクリア テキストで書き込まれます。

Management Console に新しい証明書を使用させるには、`Configuration.xml` ファイルを変更します。このファイルは `ManagementConsole.war` web アーカイブ内にあるため、解凍する必要があります。詳細については、Tomcat のデプロイを参照してください。`Configuration.xml` 内には、次のエントリーがあります。

```
<bean id="keyStore" class="com.kapowtech.mc.config.KeyStoreConfig" >
      <property name="location" value="/WEB-INF/mc.p12"/>
      <property name="password" value="changeit"/>
      <property name="alias" value="mc"/>
</bean>
```

ここで、キーストアの場所、パスワード、エイリアスを指定する必要があります。キーストアを `ManagementConsole.war` にコピーする場合、場所はアプリケーションのルートに対して相対的である必要があります。ファイル システムに保存されているキーストアを参照する場合、場所は file:// で始まり、キーストアの場所への絶対参照である必要があります。

キーストアをアップグレードする

初めて Management Console を開始すると、キーストアの秘密鍵を使用してチェックサムを作成します。これにより、キーストアが置き換えられたことを検出し、提供された証明書で実際にパスワード

を復号化できることを確認できます。独自のキーストアをインストールする前にすでに Management Console を開始している場合、パスワード変換を実行するように設定する必要があります。

キーストアをアップグレードするには、現在のキーストア ファイルをユーザーのホームフォルダなどの新しい場所にコピーし、`Configuration.xml` を変更して古いキーストアへの参照を含むパスワード コンバーターを作成します。

```
     <bean id="oldKeyStore" class="com.kapowtech.mc.config.KeyStoreConfig" >
        <property name="location" value="file:///home/roboserver/mc.p12"/>
        <property name="password" value="changeit"/>
        <property name="alias" value="mc"/>
     </bean>

     <bean id="passwordConverter"
class="com.kapowtech.scheduler.server.service.PasswordConverter">
        <constructor-arg ref="oldKeyStore"/>
     </bean>
```

これにより、以前の証明書を使用して既存のパスワードとチェックサムを復号化するパスワード コンバーターが設定され (古いキーストアの正しい場所、エイリアス、パスワードを提供する必要があります)、新しい秘密鍵 (上記で設定) を使用して再暗号化するパスワードを入力し、新しいチェックサムを作成します。次回、Management Console を開始したときにアプリケーションの開始中に変換が行われ、多くのスケジュールがある場合は時間がかかる場合があります。`Configuration.xml` から oldKeyStore Bean と passwordConverter Bean を削除する必要はありません。パスワードの変換は、チェックサムとキーストアが非同期で、変換後にチェックサムが新しいキーストアと一致する場合にのみトリガーされるためです。

SSL エンドポイント検証

新しいクラスタの作成時に、RoboServer との通信を SSL 暗号化することができます。これにより、任意のユーザーがネットワークを「リッスン」できないようになるため、2 者間で交換した重要な情報の流出を防ぐことができます。

暗号化に加えて、SSL はエンドポイント検証も提供します。これは、設定の誤りや DNS のハッキングによる第三者への重要な情報の漏洩を防ぐためのものです。この機能が動作するようにするには、RoboServer で Management Console が信頼されるように設定し、Management Console で RoboServer が信頼されるように設定する必要があります。

これには、`ManagementConsole.war` 内のファイルを編集する必要があるため、この変更を実行するときに Tomcat サーバーが実行されていないことを確認してください。

証明書

2 つの証明書を作成する必要があります。1 つは Management Console、もう 1 つは RoboServer で、それぞれの証明書には秘密鍵と公開鍵が含まれています。ここでは、証明書の作成と公開キーのエクスポートについて説明します。一般に、証明書について説明しているヘルプのセクション全体、特に API クライアント/サーバー証明書のセクションを読むことをお勧めします。

エンドポイント検証は、Management Console を RoboServer に信頼させること、および RoboServer を Management Console に信頼させることの 2 つに分けることができますが、両者は別々に設定を行うことができ、必ずしも両者を同時に設定する必要はありません。

RoboServer を Management Console に信頼させる

次に Management Console への SSL 接続を作成するときに秘密鍵を使用する RoboServer を設定する必要があります。これは、WAR ファイル内にある `/WEB-INF/certs.xml` を変更することにより行われます。証明書の場所とパスワードを指定します。これは次のようになります:

```
<bean id="sslCertificationConfiguration"
 class="com.kapowtech.mc.config.SSLVerificationConfiguration">
                    ...
     <property name="privateCertificateLocation" value="file:///home/roboserver/
client.p12"/>
     <property name="privateCertPassword" value="changeit"/>

 </bean>
```

Management Console は SSL 接続を確立するときに秘密鍵を使用するようになりました。一度 Management Console 公開鍵は RoboServer `/TrustedClients` フォルダ内にデプロイすると、RoboServer の右側に Management Console が接続されていることが確認できます。RoboServer の設定で [**API** クライアント証明書を検証] を有効にして、クラスタ内のすべての RoboServer に公開鍵を展開する必要があります。

Management Console に RoboServer を信頼させる

RoboServer は API 証明書がすでにインストールされているため、新しい証明書を作成し、事前にインストールされた証明書を置き換える必要があります。最初に上記のように証明書を作成し、次に RoboServer 設定を開始して、[証明書] タブに移動します。変更ボタンをクリックし、証明書を選択してプロンプトが表示されたらパスワードを入力します。RoboServer は、Management Console（および他の API クライアント）と SSL 接続を作成するときに新しい証明書を使用するようになりました。

次に、証明書が正しい場合に RoboServer からの SSL 接続を信頼するように Management Console を設定する必要があります。Management Console クライアント証明書と同様に、これは `/WEB-INF/certs.xml` で（部分的に）設定され、次の 2 つのオプションを使用します。

```
 <bean id="sslCertificationConfiguration"
 class="com.kapowtech.mc.config.SSLVerificationConfiguration">
     <property name="verifyRoboServerCert" value="true"/>
     <property name="checkHostName" value="true"/>
     ...
 </bean>
```

検証のためのオプションの RoboServer 証明書は単純なブール フラグ（true/false）です。これは、RoboServer JRE のデフォルト キーストアに公開キーをインポートする必要があるためです。JRE のデフォルト キー ストアは、`/jre/lib/security/` にある `cacerts` という名前のファイルです。

`Cacerts` へ RoboServer 公開鍵をインポートするには、次のコマンドを使用します:

```
keytool -import -alias RoboServer -keystore cacerts -trustcacerts -file
server.pub.cer
```

パスワードの入力を求められますが、変更していない場合、このパスワードは `changeit` です。エイリアスは一意である必要があるため、それぞれの RoboServer に対して個別の証明書を作成した場合は、接尾辞を追加します。また、この例では cacerts と server.pub.cer への参照が相対的であることに注意してください。

checkHostName オプションは、Management Console が正しい証明書を提示し、RoboServer 証明書の内部に記述されたホスト名を使用して連絡している場合にのみ、RoboServer と通信することを保証します。ホスト名がチェックされている場合、localhost と 127.0.0.1 は同じホストとは見なされないことに注意してください。

トラブルシューティング

SSL 接続を確立できない場合、利用可能な情報がほとんどないため、トラブルシューティングは非常に困難になりますが、次のことを把握しておくことが重要です。

- Management Console は、証明書が見つからない場合、またはパスワードが間違っている場合は起動しません。
- RoboServer 設定内の RoboServer 証明書を変更するときに、証明書を保存する前にパスワードが正しいことを確認します。

Management Console が RoboServer に接続できない場合は、以下の確認がトラブル シューティングに役立つ場合があります:

- RoboServer は実行されていますか? ソケットへの telnet を試行してください。
- RoboServer ホスト名は正しいですか? (checkHostName が有効な場合)
- V 公開鍵は cacerts にインポートされていますか? keytool -list -v -keystore cacerts -alias RoboServer を使用してください。-alias を指定すると、すべての証明書がリストアップされます。
- Management Console 公開証明書は RoboServer `/TrustedClients` フォルダにコピーされましたか?
- 有効期限を確認してください。公開鍵には秘密鍵の有効期限データが含まれており、Windows と Linux で開くことや表示することができます。

ユーザー アカウントの同時セッション

デフォルトでは、単一のユーザー アカウントを複数の場所から同時に認証できます。単一のユーザー アカウントの同時セッションの可能性を制限するには、`WEB-INF/spring` にある `authentication.xml` ファイルの設定を調整します。

`Authentication.xml` で、次のセクションを見つけ、コメント タグ (ここでは太字で表示) を削除します。

```
<!--
<bean class="com.kapowtech.scheduler.server.spring.security.KapowConcurrentSes
sionControlAuthenticationStrategy" lazy-init="true"> <constructor-arg ref="ses
sionRegistry"/> <constructor-arg ref="platformEMF"/> <property name="maximumSessi
ons" value="1"/> <property name="exceptionIfMaximumExceeded" value="true"/> </bea
n>
-->
```

また、ユーザーがアクションを実行しない場合にセッションを自動的に終了するタイムアウトを定義するには、`WEB-INF` にある `web.xml` ファイルで `session-timeout` プロパティを設定します。デフォルトでは、タイムアウトは 30 分です。

セキュリティが統合された Microsoft SQL Server の使用

セキュリティが統合された Microsoft SQL Server データベースを使用して Kofax RPA Management Console を実行する場合やこのようなデータベースにデータを保存する場合は、次のステップを実行して環境をセットアップします。JDBC ドライバーは Management Console に保存されないため、指定したフォルダに JAR ファイルと DLL ファイルを配置する必要があります。

Tomcat サーバー上で

- SQL Server 用の Microsoft JDBC ドライバーの JAR ファイルを Tomcat インストール フォルダの **lib** フォルダにコピーします。
- SQL Server 用の Microsoft JDBC ドライバーの DLL ファイルを、Tomcatインストール フォルダの **bin** フォルダにコピーします。

Design Studio ユーザー用の開発者コンピューター上で

- SQL Server 用の Microsoft JDBC ドライバーの JAR ファイルを、Design Studio インストール フォルダの **lib** フォルダにコピーします。
- SQL Server 用の Microsoft JDBC ドライバーの DLL ファイルを、 Design Studio インストール フォルダの **jre\bin** フォルダにコピーします。

RoboServer コンピューター上で

- SQL Server 用の Microsoft JDBC ドライバーの JAR ファイルを、RoboServer インストール フォルダの **lib** フォルダにコピーします。
- SQL Server 用の Microsoft JDBC ドライバーの DLL ファイルを、 RoboServer インストール フォルダの **jre\bin** フォルダにコピーします。

Design Studio を実行するユーザーと RoboServer にデータベースへのアクセス権があり、Windows で実行されている必要があります。

ロボット ファイル システム サーバーのセットアップ

Robot File System (RFS) サーバーは、RoboServer、Design Studio インスタンス、および Desktop Automation エージェントに共有ストレージを提供します。Tomcat サーバーで RFS サーバーをセットアップするには、次の手順を実行します。

Robot File System にアップロードまたはダウンロードできるファイルの最大サイズは、使用可能なメモリによって制限されます。

1. Kofax RPA インストール フォルダの中にある `WebApps` フォルダで、`rfs.war` ファイルを見つけます。
   たとえば、Windows システムではフォルダは次の場所にあります: `C:\Program Files\Kofax RPA 11.5.0.0\WebApps`
2. `rfs.war` を Apache Tomcat インストール フォルダの `webapps` フォルダにコピーします。
   たとえば、Windows システムではフォルダは次の場所にあります: `C:\apache-tomcat\webapps`
3. Tomcat サーバーを再起動します。
   Tomcat サーバーを再起動すると、Tomcat インストール フォルダの`webapps` フォルダに、`rfs` サブフォルダが作成されます。

4. テキスト エディターで、Tomcat インストール フォルダの `webapps\rfs\WEB-INF` フォルダにある `web.xml` ファイルを開きます。
5. `mc-path` パラメータを見つけ、`param-value` で Management Console URL を指定します。
   例：`http://localhost:50080`

   自己署名証明書を使用した HTTPS 経由の Robot File System へのアクセスはサポートされていません。

6. Management Console を開き、**[管理]** > **[サービス認証]** に移動します。**[RFS]** のコンテキスト メニューをクリックし、**[共有シークレットを表示]** を選択して、共有シークレットをクリップボードにコピーします。
   共有シークレットをファイルに保存し、`shared-secret-file` パラメータを見つけて、このファイルへのパスを `param-value` に指定します。もう 1 つの方法として、プレーン テキストの共有シークレットを `shared-secret` パラメータに直接貼り付けます (`shared-secret-file` が空であるか、ファイルを読み取れない場合にのみ使用します)。
7. `allow-absolute-paths` パラメータ見つけ、`true` または `false` を設定します。
   `allow-absolute-paths` が `true` に設定されている場合、`c:\files`、`z:\data` などのパス、および RFS サービス ユーザーがアクセスできるその他のパスを使用して、RFS ファイル共有を作成できます。`allow-absolute-paths` を `false` に設定すると `data-path` は特定のフォルダに設定されます。たとえば、Linuxの `/data` のように、`/data` と指定したフォルダ内のパスで共有するアクセスのみを許可するサービスのフォルダに設定されます。
8. `data-path` で一時的な口ボット実行データを保存するフォルダを`C:/RFSData` のように指定します。`allow-absolute-paths` が `true` に設定される場合のみ、絶対パスを指定できることに注意してください。
   一時的な共有は、指定されたフォルダのサブフォルダとして作成および削除されます。
9. 他の設定はそのままにして、Tomcat サーバーを再起動します。
10. Management Console を開き、**[設定]** > **[一般]** > **[口ボット ファイル システム サーバー]** に移動します。
    [口ボット ファイル システム サーバーを使用] を選択し、RFS サーバーを設定した Tomcat サーバーの URL を指定します。たとえば、`http://myserver.mydomain:8080/rfs` です。

これで口ボットが使用および/または生成したデータを共有および保存するように設定されたファイル システムを使用できます。ファイル システムの設定を追加するには、『Kofax RPA のヘルプ』の「口ボット ファイル システム」を参照してください。

例: 口ボット ファイル システムへのフォルダのマッピング

この例では、Windows のフォルダを Management Console の Robot File System にマッピングし、Design Studio の口ボットにステップを追加して、この口ボット ファイル システムにデータを書き込む方法についての一般的な手順を示します。

この例を実行する前に、Robot File System の機能の設定と使用方法に関する詳細情報が記載された上記の「口ボット ファイル システム サーバーの設定」の手順、および『Kofax RPA のヘルプ』の「口ボット ファイル システム」を確認しておくことをお勧めします。

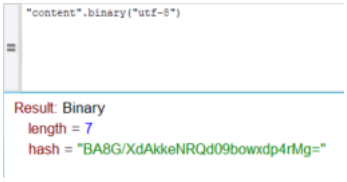
この手順では、「ロボット ファイル システム サーバーのセットアップ」の手順 1 ~ 8 を事前に確認していることが前提になります。

1. `web.xml` ファイルの `data-path` パラメータで、一時的なロボット実行データを保存するフォルダへのパスを`c:/rfs` のように指定します。

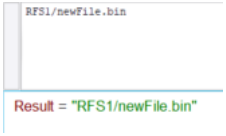
```
<init-param>
            <!-- the path to where local data is stored -->
            <param-name>data-path</param-name>
            <param-value>c:/rfs</param-value>
     </init-param>
```

   Tomcat サーバーを再起動して設定を適用します。
2. **[Management Console]** > **[設定]** > **[一般]** > **[ロボット ファイル システム サーバー]** で、Robot File System サーバーを使用することを選択します。
3. **[Management Console]** > **[リポジトリ]** > **[ロボット ファイル システム]** で、ファイル システムの設定を追加します。
   a. [一般情報] タブで、必要なパラメータをすべて指定します。
      [ファイル システム名] パラメータで [**RFS1**] を指定します。[パス] パラメータで、[**rfs1_folder**] を指定します。
      この場合、`rfs1_folder` は [**RFS1**] のルート フォルダであるため、絶対パスは `c:/rfs/rfs1_folder` となります。手動で作成しない場合は、`rfs1_folder` が自動的に作成されます。
   b. **[認証されたアクセス トークン]** タブで、ロボットのアクセス トークンを貼り付け、現在のバージョンのロボットのみがファイル システムにアクセスできるようにします (ロボットを Management Console にアップロードする必要があります)。トークンをコピーするには、**Management Console** > **[リポジトリ]** > **[ロボット]** に移動し、必要なロボットのコンテキスト メニューを開いて、**[リソース アクセス トークンを取得]** をクリックします。
4. 設定を保存します。
5. Design Studio のロボットで、次のプロパティを使用してファイル出力ステップを作成します。

   コンテンツ: ロボット ファイル システム に書き込むデータを含むバイナリ変数を指定します。この例では、「content」はファイルに書き込まれた文字列です。ファイル コンテンツはバイナリである必要があります。



ファイル名: データを書き込むファイルへのパスを指定します。ロボット ファイル システムの名前は、大文字と小文字が区別されます。



6. ステップを実行すると、newFile.bin ファイルに「content」のデータが含まれます。このファイルは `c:/rfs/rfs1_folder/` に保存されます。

一時的な RFS セッション ストレージを設定する

実行処理中に、ロボットは Robot File System にファイルを保存して読み取ることができます。この目的のために、ロボットはロボットのみが使用する一時的な RFS セッション ストレージを作成します。

たとえば、一時的な RFS セッション ストレージは、ロボットが同じような名前で異なる内容のファイルを保存する場合に役立つことがあります。このオプションは、[ファイルの読み取り] や [ファイル出力] など、いくつかの手順で使用できます。

フォルダを一時的な RFS セッション ストレージにマッピングするには、RFS サーバーをセットアップした後に、必要なロボットのステップを設定します。**[ファイル名]** プロパティで、`robot/newFile.bin` などのパスを指定します。

ファイルは `C:/rfs/8c8a89ca-fd06-4580-99a7-b7ffb7288080` などのフォルダに保存されます。

このフォルダは各ロボットの実行に固有であり、実行の完了後に削除されます。

第 3 章

サービスとしての RPA コンポーネントの実行

この章では、ServiceInstaller.exe プログラムを使用して、さまざまな Kofax RPA コンポーネントをサービスとして実行する方法について説明します。

ServiceInstaller.exe の説明

Kofax RPA コンポーネントをサービスとして実行するためには、まず ServiceInstaller.exe プログラムを使用してそのコンポーネントをインストールする必要があります。以下は、「RPAComponent」プログラムのコマンドライン引数の概要を示す一般的な例です (ここでは複数行に表示されていますが、これは 1 行のコマンドです)。

```
ServiceInstaller.exe -i RPAComponent.conf wrapper.ntservice.account=Account
 wrapper.ntservice.password.prompt=true wrapper.ntservice.name=Service-
name wrapper.ntservice.starttype=Start-method wrapper.syslog.loglevel=INFO
 wrapper.app.parameter.1="First-Argument" wrapper.app.parameter.2="Second-argument"
```

wrapper.ntservice.account

「RPAComponent」を実行する必要があるユーザーのアカウント。Kofax RPA がユーザーのディレクトリに設定を保存します。正しい設定を持つユーザーを選択することが重要です。

「RPAComponent」をドメイン ユーザーとして実行する必要がある場合は、domain\account の形式でアカウントを入力します。

「RPAComponent」を通常のユーザーとして実行する必要がある場合は、.\account の形式でアカウントを入力します。

セキュリティ上の理由から、RoboServer サービスのログイン用の LocalSystem アカウントは使用しないでください。LocalSystem が使用されている場合、WebKit (デフォルト) ロボットの実行時に以下のエラーが発生します:「WebKitBrowser への接続を確立できませんでした。パスへの接続に失敗しました。」

wrapper.ntservice.password.prompt

値が true の場合、アカウントパスワードの入力をユーザーに求めます。コマンド ラインにパスワードを入力する場合は、wrapper.ntservice.password=<your-password> を使用します。

wrapper.ntservice.name

インストールするサービスの名前。サービスの名前にスペースを含めることはできません。

wrapper.ntservice.starttype

以下の値を指定します。

- **AUTO_START**: システムの再起動時にサービスを自動的に開始します。
- **DELAY_START**: 少し時間を空けてサービスを開始します。

- **DEMAND_START**: サービスを手動で開始します。

wrapper.syslog.loglevel
「RPAComponent」からイベント ログへコンソール出力をリダイレクトします。

wrapper.app.parameter.
「RPAComponent」の引数。必要な数だけ入力できます。

サービスがインストールされると、ユーザーには「サービスとしてログオン」権限が付与されます。サービスの開始に失敗した場合は、gpedit.msc を開いて権限が付与されていることを確認し、(Windows 10 の場合) **Administrative Tools** > **Local Security Policy** > **Local Policy** > **User Rights Assignment** > **Log on as a service** > **Properties** に移動して、ユーザーを追加します。

サービスとしての RoboServer と Management Console の実行

RoboServer と Management Console はどちらも、プログラムの起動時に指定した引数に応じて、同じ RoboServer サーバー プログラムによって起動されます。

RoboServer プログラム用のコマンドライン引数の詳細な説明については、「RoboServer の開始」の「RoboServer パラメータ」セクションを参照してください。

RoboServer と Management Console を Windows と Linux で自動的に起動する方法の例を以下に示します。

Windows で RoboServer を起動する
RoboServer を Windows で自動的に起動するには、Windows サービスとして追加する必要があります。このトピックでは、Kofax RPA インストールに含まれる `ServiceInstaller.exe` プログラムを使用して Windows サービスを追加および削除する方法について説明します。

Windows サービスの追加
以下に、さまざまな設定で RoboServer をインストールする例を示します。例では、MC は Management Console、RS は RoboServer です。

- 次のスクリプトによって、デフォルトのパラメータを使用して RoboServer を起動するサービスをインストールします。サービスの名前は必要に応じて変更できます。

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_MC"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="<PORT-NUMBER>"
wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
SHARED-SECRET>"
```

- このスクリプトは、Management Console を開始するだけの Windows サービスを生成します。Management Console は独自の JVM で実行する必要があるので、可能であればこれは推奨される設定です。Windows サービスの名前は、必要に応じて変更できます。

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_MC"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="<PORT-
NUMBER>" wrapper.app.parameter.3="-MC" wrapper.app.parameter.4="-
pauseAfterStartupError" wrapper.app.parameter.5="-mcUrl"
wrapper.app.parameter.6="<MC-URL>" wrapper.app.parameter.7="-ss"
wrapper.app.parameter.8="<MC-SHARED-SECRET>"
```

- 次のスクリプトによって、2 つの RoboServer (ポート 50000 で 1 つ、ポート 50001 でもう 1 つ) を起動するサービスをインストールします。サービス名は異なる場合があります。

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_50000"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="50000"
wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
SHARED-SECRET>"
```

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_50001"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="50001"
wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
SHARED-SECRET>"
```

Windows サービスの削除

サービスをアンインストールするには、次のコマンドを実行できます。

```
ServiceInstaller.exe -r RoboServer.conf wrapper.ntservice.name=Service-name
```

wrapper.ntservice.name
削除するサービスの名前。

Linux での RoboServer の起動

最も簡単な方法で RoboServer を Linux で自動的に起動させるには、crontab を使用します。次のコマンドを使用して、Linux で特定のユーザーのスケジュール済みジョブのリストを作成または編集します。

```
crontab -u someUser -e
```

たとえばスケジュールされたジョブリストに、以下を追加します:

```
@reboot $HOME/Kofax RPA_11.5.0/bin/RoboServer -mcUrl http://localhost:8080/
ManagementConsole -p 50000 -ss [MC 共有シークレット]
```

このように RoboServer プログラムは、再起動時に指定されたコマンド ライン引数で起動します。実際のインストール フォルダの下の bin ディレクトリを識別する必要があることに注意してください。

サービスとしての Synchronizer の実行

Kofax RPASynchronizer により、Management Console と使用しているリポジトリとの間でオブジェクトの状態が比較および同期されます。Synchronizer の詳細については、『Kofax RPA のヘルプ』の「ロボット ライフサイクル マネジメント」を参照してください。

このトピックでは、Synchronizer を Windows サービスとして開始する例を示します。

Windows サービスの追加

Synchronizer をサービスとしてインストールする例を以下に示します。`ServiceInstaller.exe` については、「ServiceInstaller.exe の説明」を参照してください。

1. コマンド プロンプト ウィンドウで、Synchronizer を実行するために必要なパラメータを指定します。詳細については、『Kofax RPA のヘルプ』の「同期の開始」を参照してください。

   構成設定はコマンドを実行するユーザーの AppData に保存されるため、コマンド ラインは Synchronizer サービスを実行するユーザーとして実行する必要があります。

2. Synchronizer が正しく動作することを確認し、コマンド ラインで `-s` パラメータを使用して構成設定を保存します。
3. 次のスクリプトを使用して、Synchronizer サービスをインストールします。

```
ServiceInstaller.exe -i Synchronizer.conf wrapper.ntservice.account=domain
\account wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="Synchronizer11.5.0_MC"
wrapper.ntservice.starttype=MANUAL wrapper.syslog.loglevel=INFO
```

パラメータを変更する必要がある場合は、Synchronizer サービスを停止し、新しいパラメータを指定してコマンド ラインから Synchronizer を実行して、新しい構成設定を保存してから、Synchronizer サービスを再起動します。

Windows サービスの削除

サービスをアンインストールするには、次のコマンドを実行できます。

```
ServiceInstaller.exe -r Synchronizer.conf
 wrapper.ntservice.name="Synchronizer11.5.0_MC"
```

wrapper.ntservice.name
削除するサービスの名前。

第 4 章

Management Console 監査ログ

Management Console 監査ログでは、API コールを含めたすべてのユーザー操作を Management Console で記録しています。`Log4j2.properties` ファイルを構成することにより、ファイルまたはデータベースに情報を記録できます。Windows システムで、`log4j2.properties` ファイルは以下の場所にあります: `User home\AppData\Local\Kofax RPA\version\Configuration`.

ファイルへのロギング

```
#Log4j2 log to file configuration example

name = PropertiesConfig
appenders = auditLogAppender

appender.auditLogAppender.name = auditLog
appender.auditLogAppender.type = File
appender.auditLogAppender.fileName=logFilePath/logFileName.log
appender.auditLogAppender.layout.type = PatternLayout
appender.auditLogAppender.layout.pattern=%d - %m%n

logger.auditLog.name = auditLog
logger.auditLog.level = INFO
logger.auditLog.appenderRef.auditLog.ref = auditLog
logger.auditLog.additivity = false
```

データベースのロギング

以下の手順では例として、MySQL データベースを使用していますが、サポートされている他のデータベースは、特定の JDBC ドライバーと URL 接続を使用して口ギングに使用できます。

組み込みの RoboServer で実行されている Management Console の MySQL データベースに対する監査口グを有効にするには、次の手順を実行します。

1. MySQL コネクタの JAR ファイルを Kofax RPA インストール フォルダの `lib` サブフォルダ (`Kapow tech-common.jar` と `platform.jar` がある場所) にコピーします。たとえば、`mysql-connector-java-<`バージョン`>.jar`。Java 用のドライバーの最新バージョンを使用します。詳細については、https://repo1.maven.org/maven2/mysql/mysql-connector-java/ を参照してください。
2. データを記録するデータベース テーブルを生成します。テーブルを作成するための MySQL スクリプトは以下の通りです:

```
CREATE TABLE LOGS
   (
    DATED   timestamp      NOT NULL,
    LEVEL   VARCHAR(10)    NOT NULL,
    MESSAGE VARCHAR(1000)  NOT NULL
   );
```

情報の損失を防ぐために、メッセージ列の VERCHAR サイズが最小 600 であることを確認します。

3. `log4j2.properties` ファイルに次の行を追加します。

```
#Log4j2 log to MySQL database configuration example

name = PropertiesConfig
appenders = auditLogAppender

appender.auditLogAppender.name = auditLogAppender
appender.auditLogAppender.type = JDBC
appender.auditLogAppender.connectionSource.type = DriverManager
appender.auditLogAppender.connectionSource.connectionString = jdbc:mysql://
localhost/YourDatabaseSchemaName
appender.auditLogAppender.connectionSource.username = user
appender.auditLogAppender.connectionSource.password = password
appender.auditLogAppender.connectionSource.driverClassName = com.mysql.jdbc.Driver

appender.auditLogAppender.tableName = LOGS

appender.auditLogAppender.columnConfigs[0].type = Column
appender.auditLogAppender.columnConfigs[0].name = DATED
appender.auditLogAppender.columnConfigs[0].pattern = %d{yyyy-MM-dd HH:mm:ss}

appender.auditLogAppender.columnConfigs[1].type = Column
appender.auditLogAppender.columnConfigs[1].name = LEVEL
appender.auditLogAppender.columnConfigs[1].pattern = %p

appender.auditLogAppender.columnConfigs[2].type = Column
appender.auditLogAppender.columnConfigs[2].name = MESSAGE
appender.auditLogAppender.columnConfigs[2].pattern = %msg

logger.auditLog.name = auditLog
logger.auditLog.level = INFO
logger.auditLog.appenderRef.auditLogAppender.ref = auditLogAppender
logger.auditLog.additivity = false
```

データベースで作成したデータベース スキーマ名、ユーザー名、パスワード、テーブル名を定義します。

上記の例で使用したタイムスタンプの形式は一般的ではありません。クエリの正しい処理は、データベースで使用したデータベース タイプとデータベースによって異なります。
タイムスタンプの形式がデータベースの要件を満たしているか確認します。

Tomcat で実行している Management Console の MySQL データベースに監査ログを有効にするには、次の手順を実行します:

1. MySQL コネクタ JAR ファイルを Apache Tomcat の下の `lib` ディレクトリにコピーします。たとえば、`mysql-connector-java-8.0.16.jar` 。Java 用のドライバーの最新バージョンを使用します。詳細については、https://repo1.maven.org/maven2/mysql/mysql-connector-java/ を参照してください。

2. 手順 2 と 3 は、組み込みの RoboServer で実行されている Management Console の場合と同じです。`log4j2.properties` ファイルは、`tomcat directory\webapps\Management Console\WEB-INF\classes` にあります。

監査ログのリファレンス

このセクションでは正常に実行されたとき、またはアクセス制限によって実行できなかったときにログに記録される操作リストを提供します。ご参考までに、ログファイルの例も提供します。

ログイン イベント
RoboServer は Management Console への登録にクレデンシャルを使用しているため、すべてのユーザー ログインが記録されます。RoboServer が始まると、監査ログには RoboServer へのアクセスが許可されたユーザーからのログイン イベントが記録されます。

ログに記録された操作
Management Console では、ログに記録された操作はセクション別にグループ化されています。

例: ログ

さまざまなシナリオでのロボット実行ログの例を以下に示します。MySQL はログ データベースとして使用され、ログ メッセージはタイムスタンプ、ログ レベル、および詳細メッセージで構成されます。

REST 呼び出しから口ボットを実行
REST 呼び出しによって開始した口ボットは、最初に ID、実行 ID、タスク ID で口ボット名を記録し、次にプロジェクト名とタスク ID で口ボットを起動したユーザーを記録します。

```
2019-11-27 16:42:26        INFO      Robot Wait60 with id = 17 execution id =
 -1-9-67f20877bebb task id = 9 has requested to start
2019-11-27 16:42:26        INFO      admin run Robot f1/f2/f3/Wait60.robot from Project
 Default project with task id = 9 from REST
```

SOAP 呼び出しから口ボットを実行

```
2019-11-27 16:34:24        INFO      Robot Wait60 with id = 17 execution id =
 -1-1-67f20877bebb task id = 1 has requested to start
2019-11-27 16:34:24        INFO      admin run Robot f1/f2/f3/Wait60 from Project
 Default project with task id = 1 from SOAP
```

口ボットの実行は **UI** から開始しました
このロボットは、Management Console の [口ボット] セクションから開始されました。

```
2019-11-27 16:35:56        INFO      Robot Wait60 with id = 17 execution id =
 -1-2-67f20877bebb task id = 2 has requested to start
2019-11-27 16:35:56        INFO      admin run Robot Wait60 with id = 17 task id = 2
```

時間によってトリガーされるスケジュール
ログなし。

ユーザーによってトリガーされるスケジュール
ログメッセージは、スケジュール ID とタスク ID で参照できます。接尾辞が「- from schedule, started by user」となっている Execution ログメッセージは、このロボットが手動によるスケジュールの実行で

トリガーされたことも示しています。たとえば、「MultipleTaskSchedule」スケジュールには 4 つの口ボット ジョブが含まれます: ExampleRobot1、ExampleRobot2、ExampleRobot2、ExampleRobot3。ユーザーが手動でスケジュールを実行すると、ログメッセージは次のようになります:

```
2019-12-02 10:50:22        INFO      admin start Schedule MultipleTaskSchedule with id
 = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot1 with task id = 53 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with task id = 54 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with task id = 55 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot3 with task id = 56 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot1 with id = 1 execution id =
 3816-53-1dc33f3a2a44b task id = 53 has requested to start - from schedule, started by
 user
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with id = 39 execution id =
 3816-54-1dc33f3a2a44b task id = 54 has requested to start - from schedule, started by
 user
2019-12-02 10:50:23        INFO      Robot ExampleRobot2 with id = 39 execution id =
 3816-55-1dc33f3a2a44b task id = 55 has requested to start - from schedule, started by
 user
2019-12-02 10:50:23        INFO      Robot ExampleRobot3 with id = 20 execution id =
 3816-56-1dc33f3a2a44b task id = 56 has requested to start - from schedule, started by
 user
```

第 5 章

Kofax RPA テーブルの SQL スクリプト

データベースにテーブルを作成、削除するための SQL スクリプトは、Kofax RPA インストール ディレクトリの `documentation\sql` ディレクトリにあります。たとえば、Windows システムでは、`C:\Program Files\Kofax RPA 11.5.0\documentation\sql` にあります。スクリプト ファイルの名前には、スクリプトが対象とするデータベースの名前が含まれます。

SQL スクリプトは Design Studio をインストールする時に Kofax RPA ドキュメントとともにインストールされます。

データベース テーブルの **SQL** スクリプト

`sql` ディレクトリには、以下のように異なるスクリプトを持つ 4 つのサブディレクトリが含まれています。

- `Kapplet`: Kapplets テーブルの作成およびドロップを行うためのスクリプト
- `logdb`: logdb テーブルの作成および削除を行うためのスクリプト
- `mc`: Management Console テーブルの作成およびドロップを行うためのスクリプト
- `statistics`: 統計 (Kofax Analytics for RPA) テーブルを作成および削除するためのスクリプト

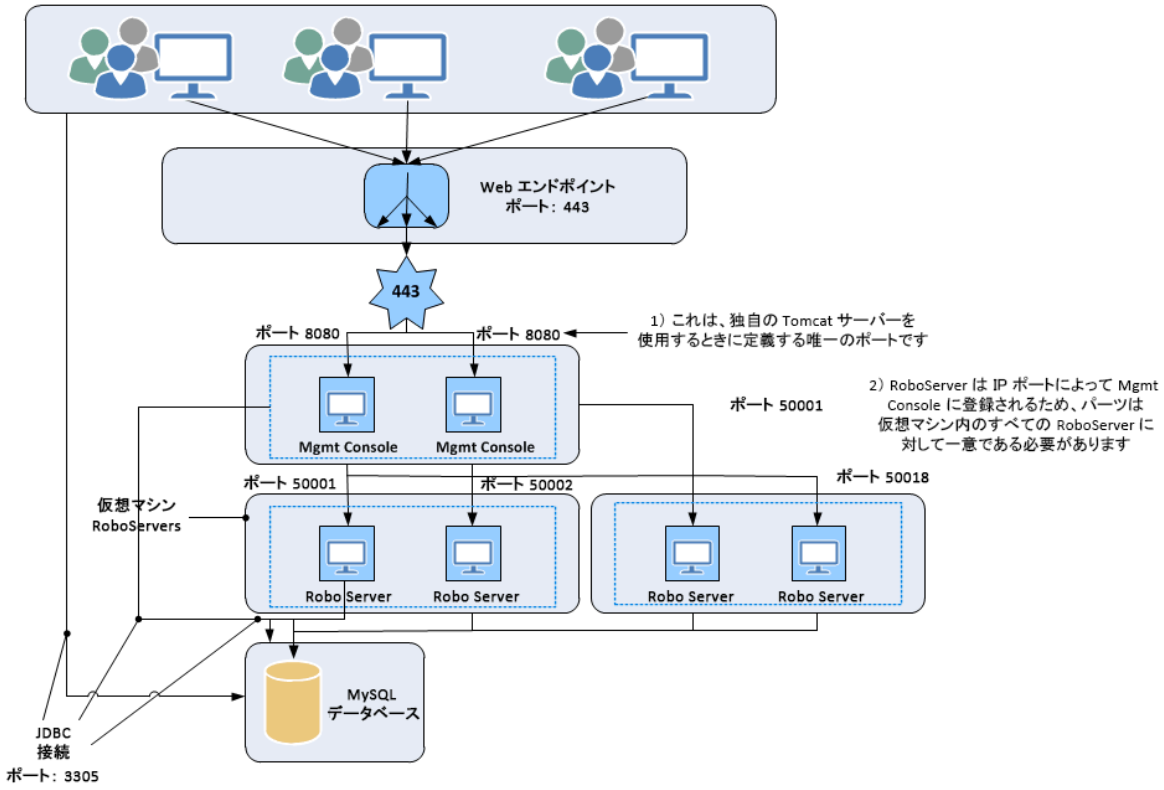
Management Console は Quartz と呼ばれるサードパーティのスケジューリング コンポーネントを使用します。Quartz にも、その他のプラットフォーム テーブルに存在する必要のある多数のテーブルが必要です。これらのテーブルは、Management Console が開始したときに自動的に作成されます。または、スクリプトを使用して手動で作成することも可能です。

以下は、Quartz 検証スクリプトです。

```
select count(*) from QRTZ_SIMPLE_TRIGGERS;
select count(*) from QRTZ_BLOB_TRIGGERS;
select count(*) from QRTZ_CRON_TRIGGERS;
select count(*) from QRTZ_CALENDARS;
select count(*) from QRTZ_FIRED_TRIGGERS;
select count(*) from QRTZ_LOCKS;
select count(*) from QRTZ_PAUSED_TRIGGER_GRPS;
select count(*) from QRTZ_SCHEDULER_STATE;
select count(*) from QRTZ_TRIGGERS;
select count(*) from QRTZ_JOB_DETAILS;
```

付録 A

Kofax RPA セキュリティ モデル



A.ユーザーのログインと認証

カテゴリー	認証と承認
説明	Kofax アプリケーションのログイン クレデンシャルは、ユーザーが入力します。
セキュリティの詳細	Kofax RPA はActive Directory / LDAP のユーザー/グループの同期をサポートします。これにより Kofax RPA は認証および資格情報管理のために企業インフラストラクチャを活用します。 Kofax RPA 利便性のために、アプリケーション固有の認証および承認メカニズムも備えています。これには資格情報管理とストレージが含まれます。保存されたパスワードは暗号化されます。

B.クライアントは **Kofax RPA** サーバーに送信

カテゴリー	転送中のデータ
ポート	80 または 443
プロトコル	HTTP または HTTPS
説明	クライアントは Kofax RPA サーバーに送信します。
セキュリティの詳細	Kofax RPA クライアント（ Management Console と Design Studio ）から Kofax RPA へのすべての接続は HTTP / HTTPS 経由です。HTTPS は、最高レベルのセキュリティを設定する必要があります。

C. Kofax RPA サーバーが別の **Kofax RPA** サーバーに送信

カテゴリー	転送中のデータ
ポート	設定可能。デフォルト 80、443、50000、50443、49999、49998
プロトコル	HTTP / HTTPS、ソケットTCP / IP
説明	Kofax RPA サーバーは別の Kofax アプリケーションやサーバーとの間で送受信を行います。
セキュリティの詳細	すべての Kofax RPA コンポーネントは、カスタム証明書のある安全な暗号化通信（TLS 1.2 ）を使用するように設定できます。

D. Kofax RPA サーバーはデータベース サーバーに送信

カテゴリー	転送中のデータ
ポート	プロトコルによって異なります
プロトコル	TCP / IP
説明	Kofax RPA サーバーがデータベースと送受信します。
セキュリティの詳細	Kofax RPA サーバーは SQL データベースに接続します。 通常、データベース サーバー システムは同じ場所に配置されるか物理的に保護されるため、送信を暗号化する必要はありません。 ただし、このような暗号化が必要な場合は、SSL を介してデータベース接続を暗号化できます。

E.ロボットとデータ ストレージ

カテゴリー	REST のデータ
説明	ロボット、設定、関連するメタデータは、 Management Console を介して保存されます。ロボットは顧客データをデータベースに保存できます。

セキュリティの詳細	ロボット、設定、関連するメタデータは、Kofax データベースに保存され、設定されたシステム アカウントを介してアクセスされます。データベース自体の暗号化機能を使用して、データベース レベルの暗号化も利用できます。 ファイル システムやデータベースの暗号化が有効かどうかに関係なく、パスワード (外部システムまたはアプリケーション固有のユーザー用) はさらに保護されます。パスワード ストアでの保存、あるいはスケジュールへの入力として保存されたパスワードは、顧客が生成した証明書を使用して暗号化されます。証明書に選択された暗号を使用して、保存されているパスワードを暗号化します。デフォルトでは、インストールは RSA 1024 ビット暗号化証明書が付いていますが、お客様が独自の証明書を生成することを強くお勧めします。詳細については パスワード暗号化 を参照してください。