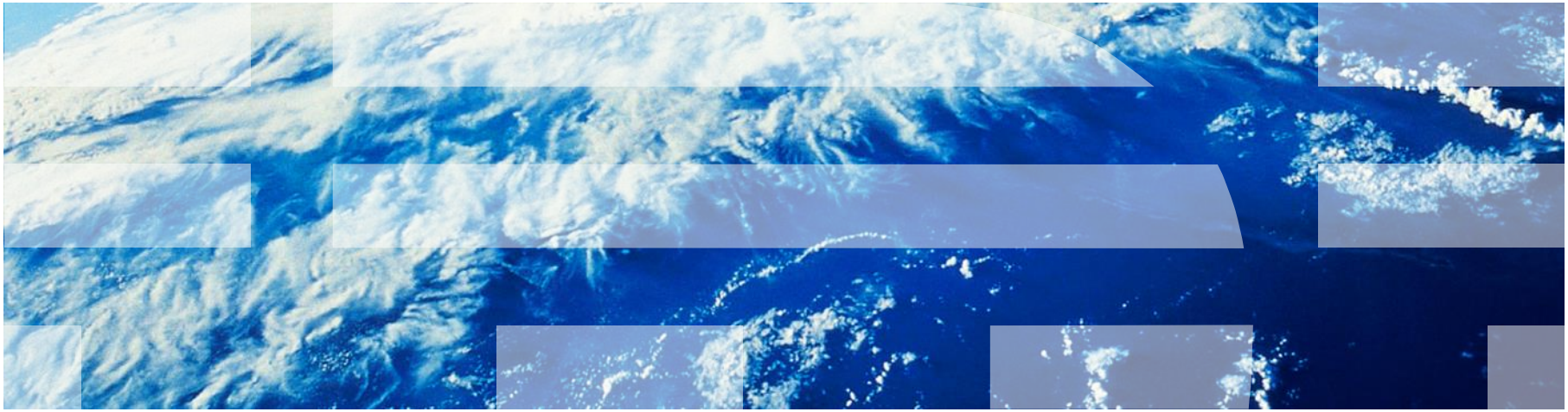
IBM Worklight Foundation V6.2.0
入門

クライアントの X.509 証明書認証とユーザー登録



2014 年 6 月 20 日

商標

- IBM、IBM ロゴ、ibm.com、WebSphere および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。
- Linux は、Linus Torvalds の米国およびその他の国における登録商標です。
- Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM® について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

概要 (1/3)

- X.509 ユーザー証明書認証機能は、新たに導入されたユーザー・レルムであり、X.509 クライアント証明書を使用してユーザー ID を確立します。
- ユーザー ID は、特定のデバイスとアプリケーション上で特定ユーザーに対して確立されます。
- この機能には、SSL クライアント・サイド証明書認証機能とユーザー登録機能が備わっています。
- SSL クライアント・サイド証明書認証では、Worklight® のクライアントとサーバー間で両方向の SSL ハンドシェークが確立されます。これにより、クライアントとサーバーの両方が自身の ID を提示できるようになり、その結果 SSL/TLS プロトコルを介して相互に信頼を確立できます。

概要 (2/3)

- ユーザー登録機能を使用して、Worklight モバイル・アプリケーション管理システムと選択した PKI に新規ユーザーを登録できます。
- ユーザー登録では、今後の使用のために、必要な X.509 ユーザー資格情報を使用してデバイスのプロビジョンも行います。
- この機能には、すぐに使い始めることができる、教育環境と非実稼働環境専用の基本的な組み込み PKI が備わっています。
- 実稼働環境の場合、この機能を使用すると、既存の PKI との統合が容易になります。
  - PKI Bridge Java™ インターフェースまたは Worklight 組み込みアダプターを使用して、証明書管理機能を外部の PKI システムに委任できます。

概要 (3/3)

- このモジュールでは、「ユーザー証明書認証」ユーザー・レルムの有効化方法と構成方法について学習します。
- このモジュールでは、以下のことも説明します。
  - Worklight に備わっている組み込み PKI の使用方法
- 実稼働環境の場合、この機能を使用すると、既存の PKI との統合が容易になります。
  - PKI Bridge Java インターフェースまたは Worklight 組み込みアダプターを使用して、証明書管理機能を外部の PKI システムに委任できます。

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

前提条件

- Worklight ユーザー・レルムとアダプター全般について理解しておく必要があります。フォーム・ベースの認証と HTTP アダプターの関連モジュールについて詳しくは、以下を参照してください。
  http://ibm.biz/knowctr#SSZH4A_6.2.0/com.ibm.worklight.getstart.doc/start/c_gettingstarted.html

- フォーム・ベースの認証を現在サポートしているアプリケーションを使用して、以下の指示に従うことを前提としています。
  - フォーム・ベースの認証モジュールでは、検証なしのログイン・モジュールを使用します。これらのログイン・モジュールは、実稼働環境では推奨されません。
  - その他のユーザー認証レルム (WASLTPA など) は、実稼働環境で使用します。

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

ユーザー証明書認証の仕組みについて (1/4)

- ユーザー証明書認証は、既存の PKI によって Worklight Server を介して、特定のアプリケーションやデバイス上の特定のユーザーに対して X.509 証明書が発行されるプロセスです。
- 関連するユーザー情報は、指定された従属ユーザー・レルムでのユーザー登録プロセス中に取得されます。
- ユーザー登録プロセスでは、X.509 証明書の発行先となる初期ユーザー ID を確立するために、従属ユーザー・レルムを必要とします。
- 次に Worklight は、今後のサーバー接続に使用するために、X.509 クライアント証明書を使用してデバイスをプロビジョンします。

ユーザー証明書認証の仕組みについて (2/4)

- ユーザーが Worklight Server に初めて接続する際には、従属レルムを介して認証を行い、登録プロセスを開始する必要があります。ユーザーがユーザー証明書認証レルムに登録されたら、それ以降のサーバー接続は両方向の SSL/TLS ハンドシェークによって行われます。このハンドシェークでは、クライアント証明書が SSL クライアント・エンティティーとして提示されます。

ユーザー証明書認証の仕組みについて(3/4)

ユーザー登録のフロー

モバイル・デバイス

Worklight Server

保護リソースに接続

従属レルム・チャレンジ

従属レルムへのログイン

CSR 要求

CSR 要求の送信

証明書を保存

Worklight サーバーは PKI と連携して、ユーザー ID を作成し、CSR を検証し、ユーザー証明書を生成します

この時点で、ユーザーは従属レルムを介してユーザー証明書認証レルムに認証され、ユーザー証明書は後で使用できるようにデバイスにプロビジョンされます

ユーザー証明書認証の仕組みについて(4/4)

クライアント証明書認証のフロー

モバイル・デバイス

Worklight は、SSL ハンドシェークの X509 クライアント証明書を使用して保護リソースに接続します

Worklight Server

ユーザーは、ユーザー証明書認証レルムに正常に認証されました

Worklight サーバーは PKI と連携して、クライアント証明書を検証し、ユーザー ID を確立します

この時点で、ユーザーは X509 クライアント証明書によりユーザー証明書認証レルムに認証され、Worklihgt は次の一連のセキュリティー・レルム・チャレンジを処理する準備が整います

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

X.509 証明書と認証局 (CA)

- セキュリティー上の理由のため、テスト中に、インフラストラクチャー内で組み込み PKI を使用する、確立された CA を使用することはお勧めできません。
- サーバー証明書とユーザー証明書の両方に署名できる自己署名 CA を作成できます。
- このモジュールでは、OpenSSL コマンド・ライン・ユーティリティーを使用します。
- OpenSSL は、大半の Linux ディストリビューションと Mac OS X に組み込まれています。Windows ユーザーは、OpenSSL Web サイトから OpenSSL バイナリーを取得できます。
- このモジュールに示されるコマンドは、Linux と Mac OS X 上で動作します。Windows の場合は、相当する MS-DOS コマンドを使用してください。

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

ルート CA、署名 CA、証明書の作成 (1/2)

- 空のディレクトリーを作成し、システムの端末でこのパスにナビゲートします。
- `openssl.cnf` という名前の基本的な OpenSSL 構成ファイルを作成します (次のスライドにサンプルを示します)。作成したディレクトリーにこのファイルを入れます。
- 別のポリシー要件が必要な場合は、各種オプションの構成方法の説明について OpenSSL の構成に関する資料を参照してください。

ルート CA、署名 CA、証明書の作成 (2/2)

- openSSL.cnf サンプル・ファイル:

```
[ req ]
default_bits                    = 2048                          # size of keys
default_keyfile                 = key.pem                       # name of generated keys
default_md                      = sha1                          # message digest algorithm
string_mask                     = nombstr                       # permitted characters
distinguished_name              = req_distinguished_name

[ req_distinguished_name ]
0.organizationName              = Organization Name (company)
organizationalUnitName          = Organizational Unit Name (department, division)
emailAddress                    = Email Address
emailAddress_max                = 40
localityName                    = Locality Name (city, district)
stateOrProvinceName             = State or Province Name (full name)
countryName                     = Country Name (2-letter code)
countryName_min                 = 2
countryName_max                 = 2
commonName                      = Common Name (hostname, IP, or your name)
commonName_max                  = 64

[ policy_match ]
countryName                     = optional
stateOrProvinceName             = optional
localityName                    = optional
organizationName                = optional
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

ルート CA の作成 (1/5)

- 以下のセクションを openssl.cnf 構成ファイルに追加して、ルート CA 要件をセットアップします。

```
[ root_authority ]
basicConstraints       = CA:TRUE
subjectKeyIdentifier = hash

[ root_authority_ca_config ]
dir                    = ./rootca
certs                  = $dir/certs
new_certs_dir          = $dir/newcerts
database                       = $dir/index.txt
certificate            = $dir/root_ca.crt
private_key            = $dir/root_ca_key.pem
serial                 = $dir/serial
RANDFILE               = $dir/.rand
policy                 = policy_match
```

ルート CA の作成 (2/5)

- 端末を使用して、ルート CA のフォルダー構造と要件を作成します。

```
# Create a root CA certificate directory structure
mkdir rootca
mkdir rootca/certs rootca/crl rootca/newcerts
touch rootca/serial

export HEXOUT=0123456789ABCDEF
# Create a serial list of random numbers
for y in {1..2048}
do
export output="";
for i in {1..16}
do
      export randomnum=$((RANDOM%16))
      export output=$output${HEXOUT:$randomnum:1};
done
echo "$output" >> rootca/serial
done

touch rootca/index.txt
```

ルート CA の作成 (3/5)

- Windows の場合、ルート CA のフォルダー構造および要件を作成します。

```
REM Create a root CA certificate directory structure
MKDIR rootca
MKDIR rootca¥certs
MKDIR rootca¥crl
MKDIR rootca¥newcerts

REM Create a serial list of random numbers for the root CA
openssl rand -hex -out rootca¥serial 8

REM Create index for root CA
COPY NUL rootca¥index.txt
```

ルート CA の作成 (4/5)

- 端末を使用して、RSA 鍵ペアを生成し、次にルート CA 証明書に自己署名します。テスト環境の場合でも、パスワードをセキュアな状態に保つ必要があります。以下の例では、パスワードは `passRoot` です。

```
export ROOT_CA_SUBJECT="Development Root CA"

# Create the RSA key pair
# The parameter, 2048, represents the key length
openssl genrsa -des3 -out rootca/root_ca_key.pem -passout
pass:passRoot 2048

# Sign a certificate with the key pair
openssl req -new -x509 -nodes -sha1 -days 365 -key
rootca/root_ca_key.pem -out rootca/root_ca.crt -config
openssl.cnf -subj "/CN=$ROOT_CA_SUBJECT" -extensions
root_authority -passin pass:passRoot
```

ルート CA の作成 (5/5)

- Windows の場合、RSA 鍵ペアを生成してから、ルート CA 証明書に自己署名します。テスト環境の場合でも、パスワードをセキュアな状態に保つ必要があります。以下の例では、パスワードは passRoot です。

```
REM Create the RSA key pair
REM The parameter, 2048, represents the key length
openssl genrsa -des3 -out rootca¥root_ca_key.pem -passout
pass:passRoot 2048

REM Sign a certificate with the key pair
openssl req -new -x509 -nodes -sha1 -days 365 -key
rootca¥root_ca_key.pem -out rootca¥root_ca.crt -config openssl.cnf
-subj "/CN=Development Root CA" -extensions root_authority -passin
pass:passRoot
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

署名 CA の作成 (1/5)

- これより前に作成した openssl.cnf ファイルを編集し、以下の構成を追加して、署名 CA 構成オプションをセットアップします。

```
[ signing_authority ]
basicConstraints                = CA:TRUE,pathlen:0
subjectKeyIdentifier            = hash
authorityKeyIdentifier  = keyid:always, issuer:always

[ signing_authority_ca_config ]
dir                             = ./signingca
certs                           = $dir/certs
new_certs_dir                   = $dir/newcerts
database                        = $dir/index.txt
certificate                     = $dir/signing_ca.crt
private_key                     = $dir/signing_ca_key.pem
serial                  = $dir/serial
RANDFILE                        = $dir/.rand
policy                  = policy_match
email_in_dn                     = false
```

署名 CA の作成 (2/5)

- 端末を使用して、署名 CA のフォルダー構造と要件を作成します。基本ディレクトリーから以下のコマンドを実行します。

```
# Create a signing CA certificate directory structure
mkdir signingca
mkdir signingca/certs signingca/crl signingca/newcerts
touch signingca/serial

export HEXOUT=0123456789ABCDEF
# Create a serial list of random numbers
for y in {1..2048}
do
export output="";
for i in {1..16}
do
      export randomnum=$((RANDOM%16))
      export output=$output${HEXOUT:$randomnum:1};
done
echo "$output" >> signingca/serial
done

touch signingca/index.txt
```

署名 CA の作成 (3/5)

- Windows の場合、署名 CA のフォルダー構造と要件を作成します。基本ディレクトリーから以下のコマンドを実行します。

```
REM Create a signing CA certificate directory structure
MKDIR signingca
MKDIR signingca¥certs
MKDIR signingca¥crl
MKDIR signingca¥newcerts

REM Create a serial list of random numbers for the signing CA
openssl rand -hex -out signingca¥serial 8

REM Create index for signing CA
COPY NUL signingca¥index.txt
```

署名 CA の作成 (4/5)

- 端末を使用して、RSA 鍵ペアを生成し、次にルート CA で署名 CA CSR に署名します。この例では、パスワードは passSigning です。基本ディレクトリーから以下のコマンドを実行します。

```
export SIGNING_CA_SUBJECT="Development Signing CA"

openssl genrsa -des3 -out signingca/signing_ca_key.pem -passout
pass:passSigning 2048

openssl req -new -key signingca/signing_ca_key.pem -out
signingca/signing_ca.csr -config openssl.cnf -subj
"/CN=$SIGNING_CA_SUBJECT" -passin pass:passSigning

openssl ca -in signingca/signing_ca.csr -out signingca/signing_ca.crt -
keyfile rootca/root_ca_key.pem -cert rootca/root_ca.crt -config openssl.cnf
-name root_authority_ca_config -extensions signing_authority -md sha512 -
days 365 -passin pass:passRoot
```

署名 CA の作成 (5/5)

- Windows の場合、RSA 鍵ペアを生成し、次にルート CA で署名 CA CSR に署名します。この例では、パスワードは passSigning です。基本ディレクトリーから以下のコマンドを実行します。

```
openssl genrsa -des3 -out signingca¥signing_ca_key.pem -passout
pass:passSigning 2048

openssl req -new -key signingca¥signing_ca_key.pem -out
signingca¥signing_ca.csr -config openssl.cnf -subj "/CN=Development Signing
CA" -passin pass:passSigning

openssl ca -in signingca¥signing_ca.csr -out signingca¥signing_ca.crt -
keyfile rootca¥root_ca_key.pem -cert rootca¥root_ca.crt -config openssl.cnf
-name root_authority_ca_config -extensions signing_authority -md sha512 -
days 365 -passin pass:passRoot
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

サーバー証明書の作成 (1/3)

- 前に作成した openssl.cnf ファイルを編集し、以下の構成を追加して、サーバー証明書構成オプションを設定します。

```
[ server_identity ]
basicConstraints        = CA:FALSE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always;issuer:always
```

サーバー証明書の作成 (2/3)

- 端末を使用して、RSA 鍵ペアを生成し、署名 CA で新規証明書に署名します。この証明書がサーバー ID 証明書です。この例では passServer をパスワードとして使用します。基本ディレクトリーから以下のコマンドを実行します。

```
# Use the full hostname of your Worklight Server. SSL will break if the full hostname
is not provided, or if an IP address is used as the hostname.
export SERVER_FULL_HOSTNAME=dev.yourcompany.com
mkdir server

# Create the RSA key pair and generate a CSR
openssl genrsa -des3 -out server/server_key.pem -passout pass:passServer 2048
openssl req -new -key server/server_key.pem -out server/server.csr -config openssl.cnf
-subj "/CN=$SERVER_FULL_HOSTNAME" -passin pass:passServer

# Sign the CSR with the signing CA
openssl ca -in server/server.csr -out server/server.crt -keyfile
signingca/signing_ca_key.pem -cert signingca/signing_ca.crt -config openssl.cnf -name
signing_authority_ca_config -extensions server_identity -md sha512 -days 365 -passin
pass:passSigning
```

サーバー証明書の作成 (3/3)

- Windows の場合、RSA 鍵ペアを生成し、署名 CA で新規証明書に署名します。この証明書がサーバー ID 証明書です。この例では passServer をパスワードとして使用します。基本ディレクトリーから以下のコマンドを実行します。

```
REM Use the full hostname of your Worklight Server. SSL will break if the full
REM hostname is not provided, or if an IP address is used as the hostname.
SET HOSTNAME=dev.yourcompany.com
mkdir server

REM Create the RSA key pair and generate a CSR
openssl genrsa -des3 -out server¥server_key.pem -passout pass:passServer 2048
openssl req -new -key server¥server_key.pem -out server¥server.csr -config openssl.cnf
-subj "/CN=%HOSTNAME%" -passin pass:passServer

REM Sign the CSR with the signing CA
openssl ca -in server¥server.csr -out server¥server.crt -keyfile
signingca¥signing_ca_key.pem -cert signingca¥signing_ca.crt -config openssl.cnf -name
signing_authority_ca_config -extensions server_identity -md sha512 -days 365 -passin
pass:passSigning
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

サーバー証明書用の証明書チェーンの作成 (1/2)

- 端末を使用して、iOS 環境と Android 環境のトラスト・アンカー (ルート CA) までの完全な証明書チェーンを送信します。証明書ファイルをトラスト・アンカー (ルート CA) に連結できます。

```
# Create a chain for the signing CA
cat signingca/signing_ca.crt rootca/root_ca.crt >
signing_ca_chain.crt

# Create a chain for the server certificate
cat server/server.crt signingca/signing_ca.crt
rootca/root_ca.crt > server_chain.crt
```

サーバー証明書用の証明書チェーンの作成 (2/2)

- Windows の場合、iOS 環境と Android 環境のトラスト・アンカー (ルート CA) までの完全な証明書チェーンを送信します。証明書ファイルをトラスト・アンカー (ルート CA) に連結できます。

```
REM Create a chain for the signing CA
COPY rootca¥root_ca.crt+signingca¥signing_ca.crt
signing_ca_chain.crt

REM Create a chain for the server certificate
COPY
rootca¥root_ca.crt+signingca¥signing_ca.crt+server¥server.crt
server_chain.crt
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

署名 CA 用の PKCS12 ファイルのエクスポート

- 署名 CA 用の秘密鍵と証明書を .p12 鍵ストア・ファイルにエクスポートして、組み込み PKI が署名 CA でユーザー証明書に署名できるようにします。

```
openssl pkcs12 -export -in signingca/signing_ca.crt -inkey
signingca/signing_ca_key.pem -out signingca/signing_ca.p12 -
passin pass:passSigning -passout pass:passSigningP12
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

サーバー証明書用の PKCS12 ファイルのエクスポート

- サーバー用の秘密鍵と証明書を .p12 鍵ストア・ファイルにエクスポートして、サーバーが有効なサーバー証明書をクライアントに送信できるようにします。

```
openssl pkcs12 -export -in server_chain.crt -inkey
server/server_key.pem -out server/server.p12 -passout
pass:passServerP12 -passin pass:passServer
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

IBM WebSphere® Application Server Liberty プロファイル (Liberty) の構成 (1/3)

- 以下のように、`ssl-1.0` および `appSecurity-2.0` 機能を `server.xml` ファイルで有効にします。

```
<featureManager>

     <feature>ssl-1.0</feature>
     <feature>appSecurity-2.0</feature>

</featureManager>
```

IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成 (2/3)

- Liberty を使用するには、鍵ストアとトラストストアをセットアップして、生成されたクライアント証明書に対して信頼を確立する必要があります。詳しくは、以下を参照してください。
  http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.wlp.nd.doc/ae/rwlp_ssl.html
  - 以前に生成された `.p12` ファイル (`server.p12`) を使用するようにサーバーの鍵ストアをセットアップします。
  - 以前に生成された `.p12` ファイル (`signing_ca.p12`) を使用するようにトラストストアをセットアップします。
  - サーバーの HTTP エンドポイントを構成し、クライアント・サイド証明書を許可します (ただし要求はしません)。この構成を使用するには、Liberty SSL エレメントで `clientAuthenticationSupported="true"` プロパティーを使用します。
- 注: 完全な例については、次のスライドで示します。

IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成 (3/3)

- 以下の例は、更新後の SSL 構成を示しています。

```
<!-- default SSL configuration is defaultSSLSettings -->
<sslDefault sslRef="defaultSSLSettings" />
<ssl clientAuthenticationSupported="true" id="defaultSSLSettings"
keyStoreRef="defaultKeyStore" trustStoreRef="defaultTrustStore"
/>
<keyStore id="defaultKeyStore" location="server.p12"
password="passServerP12" type="PKCS12" />
<keyStore id="defaultTrustStore" location="signing_ca.p12"
password="passSigningP12" type="PKCS12" />
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

authenticationConfig.xml の構成 (1/3)

- authenticationConfig.xml ファイルの「User Certificate Login Module」セクションのコメントを外します。

```
<!-- Login Module for User Certificate Authentication -->
<loginModule name="WLUserCertificateLoginModule">
  <className>com.worklight.core.auth.ext.UserCertificateLoginModule</className>
</loginModule>
```

- wl_userCertificateAuthRealm セクションのコメントを外します。

```
<-- For User Certificate Authentication -->
<realm name="wl_userCertificateAuthRealm" loginModule="WLUserCertificateLoginModule">
  <className>com.worklight.core.auth.ext.UserCertificateAuthenticator</className>
  <parameter name="dependent-user-auth-realm" value="SampleAppRealm" />
  <parameter "name=pki-bridge-class"
value="com.worklight.core.auth.ext.UserCertificateEmbeddedPKI" />
```

authenticationConfig.xml の構成 (2/3)

- `embedded-pki-bridge-ca-p12-file-path` エレメントの値を、署名 CA `.p12` ファイルの絶対パスに更新します。
- `embedded-pki-bridge-ca-p12-password` エレメントの値を、`.p12` ファイルの作成に使用したパスワード (`passSigningP12`) に更新します。
- `dependent-user-auth-realm` エレメントの値を、使用する従属レルム (`SampleAppRealm`) に更新します。
- レルム名 (`wl_userCertificateAuthRealm`) は変更できません。
- 注: 完全な例については、次のスライドで示します。

authenticationConfig.xml の構成 (3/3)

- 以下の例は、前のスライドでリストした更新内容を示しています。

```
<!-- For User Certificate Authentication -->
<realm name="wl_userCertificateAuthRealm" loginModule="WLUserCertificateLoginModule">
  <className>com.worklight.core.auth.ext.UserCertificateAuthenticator</className>
  <parameter name="dependent-user-auth-realm" value="SampleAppRealm" />
  <parameter name="pki-bridge-class"
value="com.worklight.core.auth.ext.UserCertificateEmbeddedPKI" />
  <parameter name="embedded-pki-bridge-ca-p12-file-path"
value="YOUR_BASE_DIRECTORY/signing_ca.p12" />
  <parameter name="embedded-pki-bridge-ca-p12-password" value="passSigningP12" />
</realm>
```

- `wl_userCertificateAuthRealm` を使用するセキュリティー・テストを定義します。

```
<customSecurityTest name="customx509Tests">
  <test realm="wl_antiXSRFRealm" step="1" />
  <test realm="wl_authenticityRealm" step="1" />
  <test realm="wl_directUpdateRealm" mode="perSession" step="1" />
  <test realm="wl_userCertificateAuthRealm" isInternalUserID="true" step="1" />
  <test realm="wl_deviceNoProvisioningRealm" isInternalUserID="true" step="2" />
</customSecurityTest>
```

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

application-descriptor.xml の構成

- iOS 環境または Android 環境が Worklight アプリケーションに追加されたことを確認します。
- カスタム・セキュリティー・テストを使用して、アプリケーションまたは環境を保護します。
  - `<android securityTest="customx509Tests" version="1.0">`
  - `<iPhone bundleId="com.SampleApp" securityTest="customx509Tests" version="1.0">`
- アプリケーションとアダプターをビルドし、Worklight Server にデプロイします。

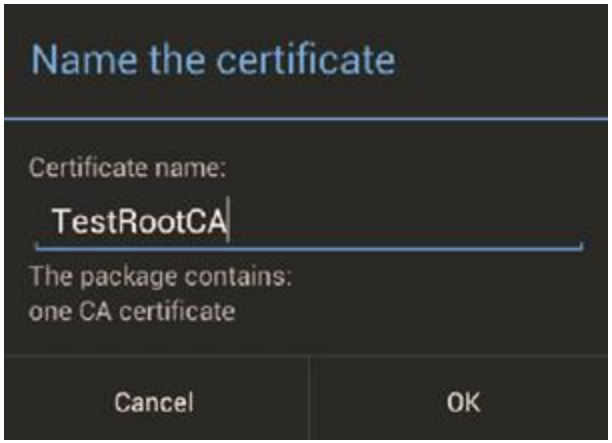
アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

iOS と Android へのルート CA のインストール

- デバイスが SSL を介して Worklight Server を信頼するには、以前のステップで生成したルート CA をクライアント・デバイスにインストールする必要があります。`root_ca.crt` ファイルを E メールで送信するかホストし、デバイス上でこのファイルを開きます。証明書を手動でインストールしようとすると、iOS および Android デバイスから承認を求められます。

iOS



Android

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

アプリケーションのインストールとテスト (1/2)

1. アプリケーションを Worklight Server にデプロイします。
   - 「実行」->「<*server name*> 上で実行 (Run on <server name>)」
2. HTTPS のデプロイ・ターゲットを更新します。
   - 「実行」->「ビルド設定およびデプロイ・ターゲット...(Build Setting and Deploy Target…)」
   - **「他の Worklight サーバーと連携するようにアプリケーションをビルドする (Build the application to work with a different Worklight Server)」**チェック・ボックスにマークを付けます。
   - サーバー HTTPS アドレスを入力します。
     `https://<host>:<https port #>`
   - アプリケーション・コンテキスト・パスを入力します。
     `/<path>`

アプリケーションのインストールとテスト (2/2)

3. 更新されたデプロイ・ターゲットでアプリケーションをビルドします。
   - 「実行」->「すべての環境のビルド (Build All Environments)」
4. 指定環境でアプリケーションを実行します。

アジェンダ

- 概要
- 前提条件
- ユーザー証明書認証の仕組みについて
- X.509 証明書と認証局 (CA)
  - ルート CA、署名 CA、証明書の作成
  - ルート CA の作成
  - 署名 CA の作成
  - サーバー証明書の作成
  - サーバー証明書用の証明書チェーンの作成
  - 署名 CA 用の PKCS12 ファイルのエクスポート
  - サーバー証明書用の PKCS12 ファイルのエクスポート
- IBM WebSphere Application Server Liberty プロファイル (Liberty) の構成
- authenticationConfig.xml の構成
- application-descriptor.xml の構成
- iOS と Android へのルート CA のインストール
- アプリケーションのインストールとテスト
- 結果の確認

結果の確認

- 正常に構成されたことを確認するには、保護リソースへのアクセスを初めて試みる際に、ログイン・フォームが表示されることを確認します。`main.js` ファイルで `WL.Client.connect()` のコメントを外した場合、アプリケーションの開始時にログイン・フォームが表示されます。コメントを外していない場合、アダプターの呼び出し後にログイン・フォームを表示するには、アダプター・プロシージャーを呼び出す前に、`WL.Client.connect()` を呼び出す必要があります。
- 従属レルムを使用してログインすると、ユーザーが正常に登録されたことを示す成功応答がアダプター呼び出しから返されます。
- それ以降のサーバー接続では、ユーザーがログインを求められることはなく、アダプター呼び出しから正常に応答が返されます。

詳細情報

- *ユーザー証明書認証機能*について詳しくは、「IBM Worklight Foundation ユーザー文書」(以下の URL) を参照してください。
  - http://ibm.biz/knowctr#SSZH4A_6.2.0/com.ibm.worklight.monitor.doc/monitor/c_user_CA.html

特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作または サービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました