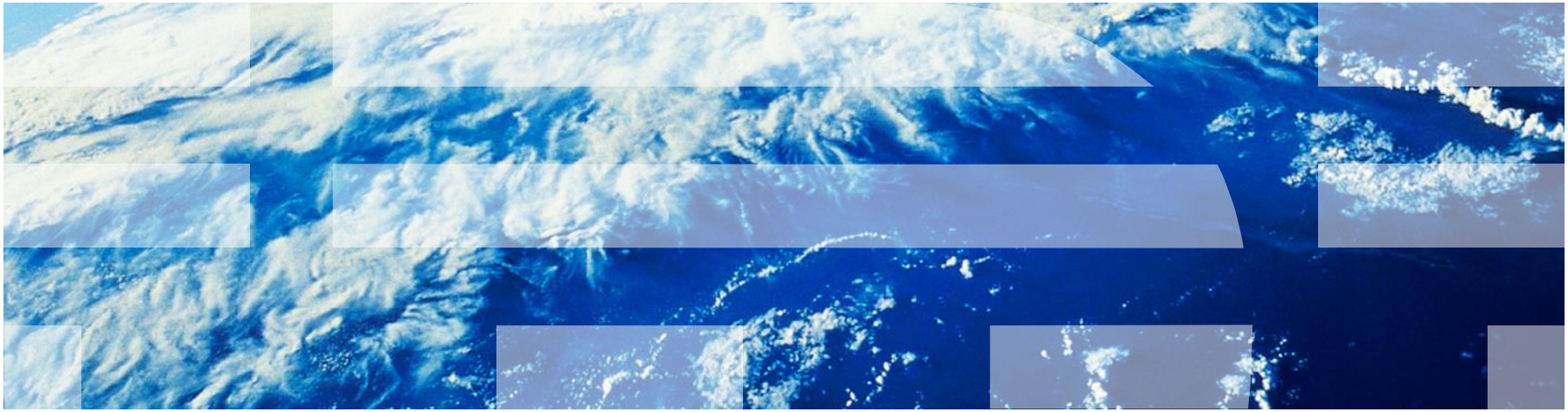
IBM Worklight Foundation V6.2.0
入門

ハイブリッド・アプリケーションでのプッシュ通知



2014 年 6 月 26 日

商標

- IBM、IBM ロゴ、ibm.com 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。
- 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

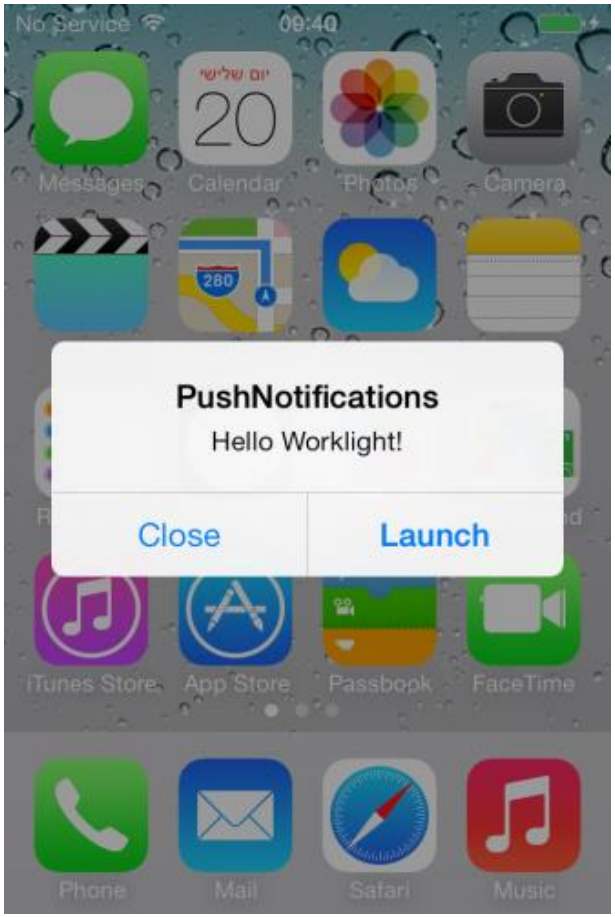
- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

プッシュ通知とは

- プッシュ通知は、サーバーから *プッシュされる* メッセージを受信する、モバイル・デバイスの機能です。
- 通知は、アプリケーションが現在実行中であるかどうかに関わらず受信されます。
- 通知は以下のようにいくつかの形式をとることができます (プラットフォームに依存)。
  - **アラート**: ポップアップ・テキスト・メッセージ
  - **バッジ、タイル**: ショート・テキストまたはイメージを使用可能なグラフィカル表現
  - **バナー、トースト**: デバイス・ディスプレイの上部にある、消えるポップアップ・テキスト・メッセージ
  - **音声アラート**



アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

デバイス・サポート

- IBM® Worklight® Foundation では、以下のモバイル・プラットフォームのプッシュ通知がサポートされます。
  - Android (2.3.5、4.x)
  - iOS 5、6、および 7
  - Windows Phone 8

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

通知アーキテクチャー
用語

- **イベント・ソース**
  - モバイル・アプリケーションが登録できるプッシュ通知チャネル。イベント・ソースは Worklight アダプター内で定義されます。
- **デバイス・トークン**
  - プッシュ・メディエーター (Apple、Google、または Microsoft) から取得した固有 ID。Worklight サーバーから通知を受け取る特定のモバイル・デバイスの要求を識別します。
- **ユーザー ID**
  - Worklight ユーザーの固有 ID。認証または他の固有 ID (パーシスタント Cookie など) を通じて取得します。
- **アプリケーション ID**
  - Worklight アプリケーション ID。モバイル・マーケットで特定の Worklight アプリケーションを識別します。

通知アーキテクチャー
サブスクリプション

- プッシュ通知の受信を開始するには、アプリケーションがまずプッシュ通知*イベント・ソースにサブスクライブする必要があります。*
- イベント・ソースは、プッシュ通知サービス用のアプリケーションが使用する Worklight アダプターで宣言されます。
- ユーザーはプッシュ通知サブスクリプションを承認する必要があります。
- ユーザーが承認を行うと、デバイスは、そのデバイスの識別に使用されるトークンを取得するために、Apple、Google、または Microsoft のプッシュ・サーバーに登録され (「デバイス Y でのアプリケーション X の通知を許可」)、サブスクリプション要求を Worklight サーバーに送信します。
  - この操作は Worklight フレームワークによって自動的に実行されます。

通知アーキテクチャー
サブスクリプション

サブスクライブ API メソッドが呼び出されると、デバイスはプッシュ・サービス・メディエーターに登録され、デバイス・トークンを取得します (IBM Worklight Foundation によって自動的に行われます)。

通知アーキテクチャー
サブスクリプション

トークンが取得されると、アプリケーションがイベント・ソースにサブスクライブします。

どちらのアクションも、後述のように１つの API メソッド呼び出しを使用して自動的に行われます。

通知アーキテクチャー
通知の送信

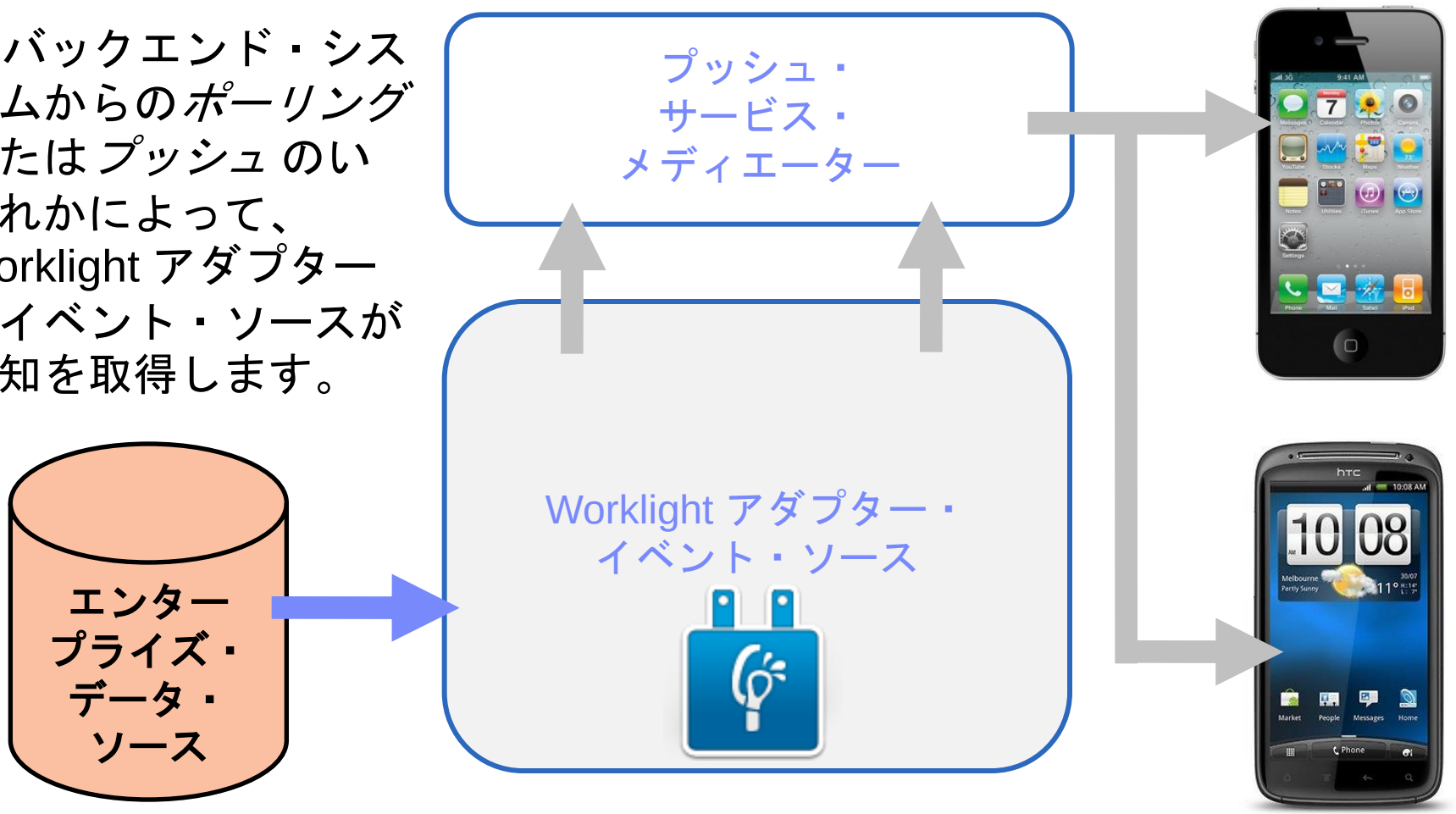
- IBM Worklight Foundation は統一されたプッシュ通知 API を提供します。
- アダプター API では以下を行うことができます。
  - サブスクリプションを管理する
  - バックエンドから通知をプッシュおよびポーリングする
  - プッシュ通知をデバイスに送信する
- アプリケーション API では以下を行うことができます。
  - プッシュ通知イベント・ソースに対してサブスクライブおよびアンサブスクライブする
  - 到着通知を処理する

通知アーキテクチャー
通知の送信

- 通知を送信するためには、まずその通知をバックエンドから取得する必要があります。
- イベント・ソースは、バックエンド・システムから通知をポーリングでき、またバックエンド・システムが新しい通知を明示的にプッシュするのを待機することもできます。
- アダプターが通知を取得すると、通知は処理され、対応するプッシュ・サービス・メディエーター (Apple、Google、または Microsoft) を通じて送信されます。
- 通知を前処理するために、カスタム・ロジックをアダプターに追加できます。
- プッシュ・サービス・メディエーターが通知を受信し、それをデバイスに送信します。

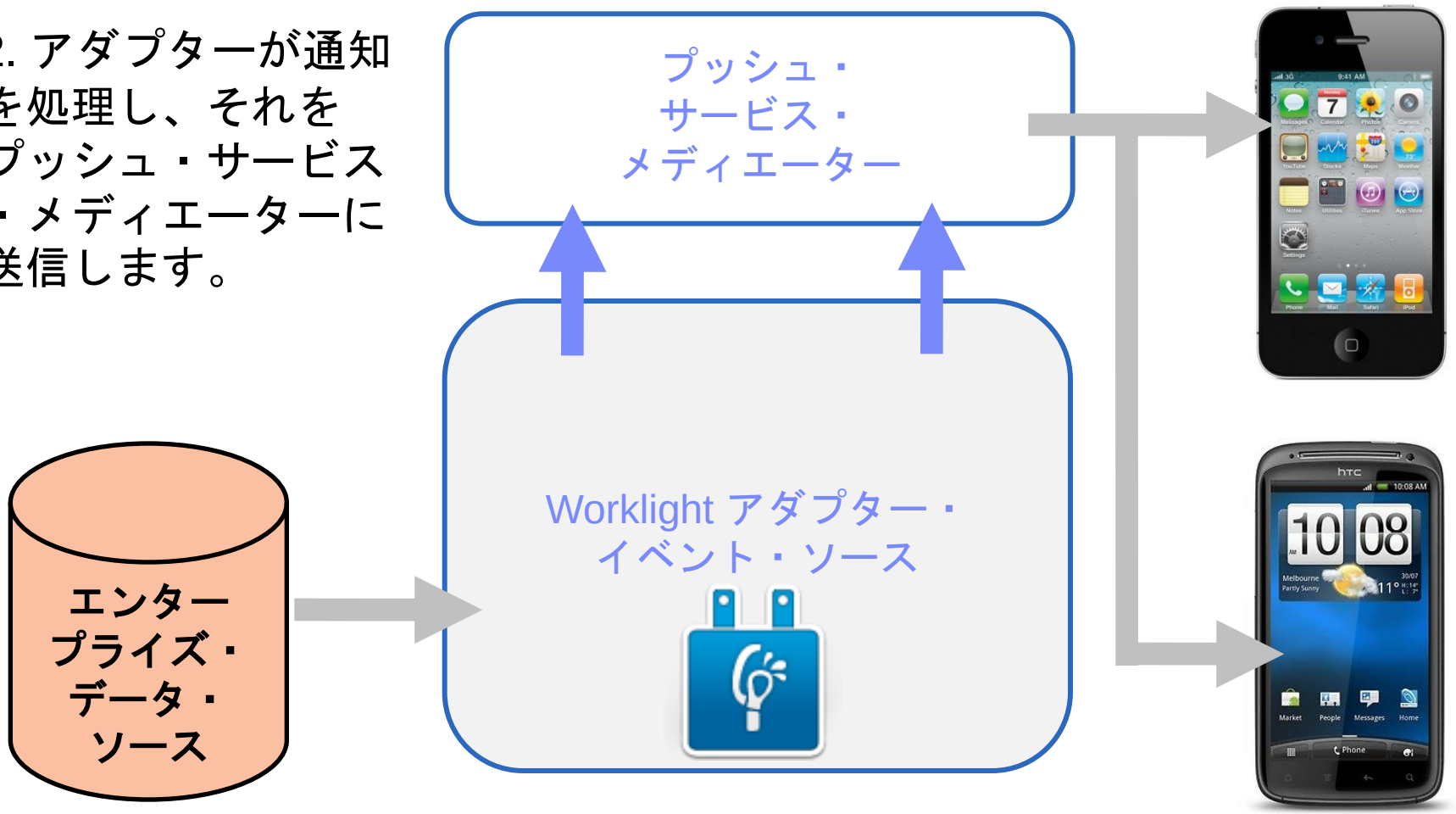
通知アーキテクチャー
通知の送信

1. バックエンド・システムからの*ポーリング*または*プッシュ*のいずれかによって、Worklight アダプター・イベント・ソースが通知を取得します。



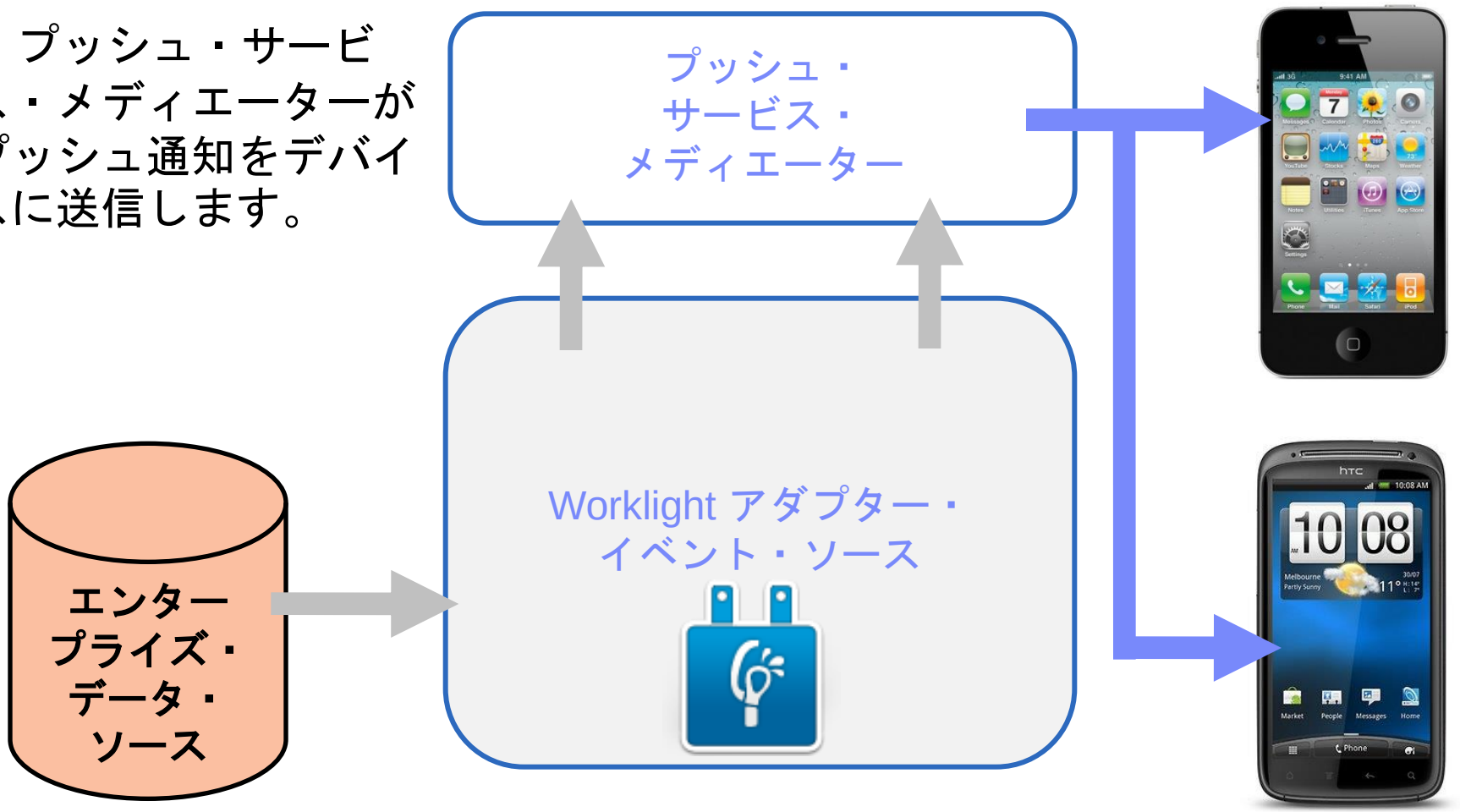
通知アーキテクチャー
通知の送信

2. アダプターが通知を処理し、それをプッシュ・サービス・メディエーターに送信します。



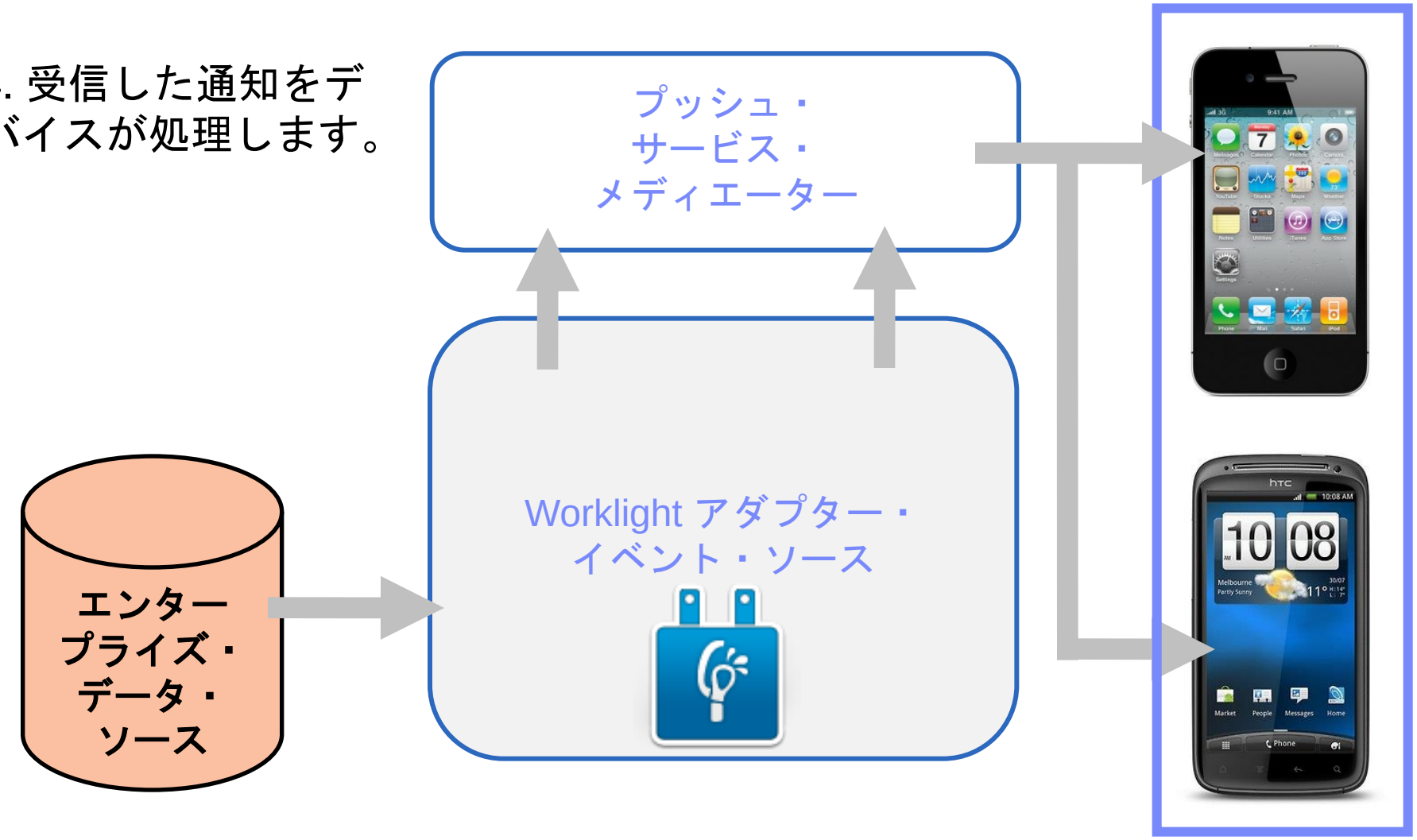
通知アーキテクチャー
通知の送信

3. プッシュ・サービス・メディエーターがプッシュ通知をデバイスに送信します。



通知アーキテクチャー
通知の送信

4. 受信した通知をデバイスが処理します。



アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

サブスクリプション管理
ユーザー・サブスクリプション

- **ユーザー・サブスクリプション**
  - ユーザー ID、デバイス ID、およびイベント・ソース ID を含むエンティティー。これは、特定のイベント・ソースから通知を受信するユーザーのインテントを表します。
- **作成**
  - イベント・ソースのユーザー・サブスクリプションは、ユーザーが最初に任意のデバイスからそのイベント・ソースにサブスクライブした時に作成されます。
- **削除**
  - ユーザー・サブスクリプションは、当該ユーザーがそのユーザーのすべてのデバイスから、イベント・ソースからのアンサブスクライブを行うと削除されます。
- **通知**
  - ユーザー・サブスクリプションが存在している間、Worklight Server は、サブスクライブ済みユーザーのプッシュ通知を生成することができます。これらの通知は、アダプター・コードによって、ユーザーがサブスクライブしたすべてのデバイスまたは一部のデバイスに送信することができます。

サブスクリプション管理
デバイス・サブスクリプション

- デバイス・サブスクリプションは、ユーザー・サブスクリプションに属し、特定のユーザーおよびイベント・ソースのスコープ内に存在します。1 つのユーザー・サブスクリプションで複数のデバイス・サブスクリプションを持つことができます。
- デバイス上のアプリケーションが `WL.Client.Push.subscribe()` を呼び出すと、デバイス・サブスクリプションが作成されます。
- デバイス・サブスクリプションは、`WL.Client.Push.unsubscribe()` を呼び出すアプリケーションによって削除されるか、デバイスが永続的にアクセスできないことをプッシュ・メディエーターが Worklight サーバーに通知した場合に削除されます。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

通知 API
サーバー・サイド

- 最初にイベント・ソースを作成する。
  - アダプター JavaScript™ コードで通知イベント・ソースをグローバル・レベル (JavaScript 関数外) で宣言します。

通知をバックエンドによってプッシュする

```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc',
    securityTest:'PushApplication-strong-mobile-se
});
```

通知をバックエンドからポーリングする

```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc',
    securityTest: 'PushApplication-strong-mobile-securityTest',
    poll: {
        interval: 3,
        onPoll: 'getNotificationsFromBackend'
    }
});
```

  - **name** – イベント・ソースの参照に使用する名前
  - **onDeviceSubscribe** – ユーザー・サブスクリプション要求の受信時に呼び出すアダプター関数
  - **onDeviceUnsubscribe** – ユーザー・アンサブスクライブ要求の受信時に呼び出すアダプター関数
  - **securityTest** – イベント・ソースの保護に使用する **authenticationConfig.xml** からのセキュリティー・テスト

通知 API
サーバー・サイド

- 最初にイベント・ソースを作成する。
  - アダプター JavaScript コードで通知イベント・ソースをグローバル・レベル (JavaScript 関数外) で宣言します。

通知をバックエンドによってプッシュする

```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc',
    securityTest:'PushApplication-strong-mobile-se
});
```

通知をバックエンドからポーリングする

```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc',
    securityTest:'PushApplication-strong-mobile-securityTest',
    poll: {
        interval: 3,
        onPoll: getNotificationsFromBackend
    }
});
```

  - **poll** – 通知を取得するために使用されるメソッド
    - 必要なパラメーターは以下のとおりです。
      - **interval** – ポーリング間隔 (秒)
      - **onPoll** – ポーリングの実装 – 指定された間隔で呼び出されるアダプター関数

通知 API
サーバー・サイド

- 通知を送信する。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId :: " + us

    var notification = WL.Server.createDefaultNotification(

    WL.Server.notifyAllDevices(userSubscription, notificati

    return { result: "Notification sent to user :: " + user
}
```

前述のとおり、通知はバックエンド・システムからポーリングするか、バックエンド・システムによってプッシュすることができます。この例では、`submitNotifications()` アダプター関数が、通知を送信する外部 API としてバックエンド・システムによって呼び出されます。

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知データを取得します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationS      ription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscripti      found for user :: " +
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId :: " + userI                    xt);

    var notification = WL.Server.createDefaultNotification(not

    WL.Server.notifyAllDevices(userSubscription, notification)

    return { result: "Notification sent to user :: " + userId
}
```

`submitNotification` 関数は、通知の送信先の `userId` および `notificationText` を受け取ります。これらの引数は、この関数を呼び出すバックエンド・システムによって提供されます。

通知 API
サーバー・サイド

- 通知を送信する。
  - アクティブ・ユーザーを検索し、それを使用してユーザーのサブスクリプション・データを取得します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subsc    tion found for user :: " + userId };
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscri

    WL.Logger.debug("submitNotification >

    var notification = WL.Server.createDe

    WL.Server.notifyAllDevices(userSubscr

    return { result: "Notification sent t
}
```

ユーザー・サブスクリプション・オブジェクトにはすべてのユーザーのサブスクリプションに関する情報が含まれています。各ユーザー・サブスクリプションは複数のデバイス・サブスクリプションを持つことができます。
オブジェクト構造は次のとおりです。

```
{
    userId: 'bjones',
    state: {
        customField: 3
    },
    getDeviceSubscriptions: function(){}
};
```

通知 API
サーバー・サイド

- 通知を送信する。
  - ユーザー・サブスクリプション・データを取得します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    var deviceSubscriptions =
        userSubscription.getDev    ubscriptions();

    WL.Logger.debug("submitNoti    tion >> userId :: "

    var notification = WL.Server.createDefaultNotificat

    WL.Server.notifyAllDevices(userSubscription, notifica

    return { result: "Notification sent to user :: " + userId };
}
```

指定のイベント・ソースに対するサブスクリプションをユーザーが持っていない場合は、ヌル・オブジェクトが返されます。

通知 API
サーバー・サイド

- 通知を送信する。
  - ユーザー・サブスクリプション・データを取得します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAda

    if (userSubscription==null){
        return { result: "No subscription found for user :
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("su otification >> userId :: " + u

    var notification = W erver.createDefaultNotification

    WL.Server.notifyAllDevices(userSubscription, notificat

    return { result: "Notification sent to user :: " + use
}
```

getDeviceSubscriptions API メソッドを使用して、ユーザーのデバイスごとのサブスクリプション・データを個別に取得することができます。この結果は、次の構造を含むオブジェクトの配列となります。

```
[
  {
    alias: "myPush",
    device: "4AooAq83gUSoas.....",
    token: 'KQz0srTUXsOqh.....',
    applicationId: 'PushApp',
    platform: 'Android',
    options: {
        customOption: 'aaa',
        alert: true,
        badge: true,
        sound: true
    }
  }
]
```

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知をユーザーの 1 つまたは複数のデバイスに送信します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscriptio
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscr

    WL.Logger.debug("submitNotification >> u                    ext);

    var notification = WL.Server.createDefaultNotification(notificationText, 3, {foo : 'bar'});

    WL.Server.notifyAllDevices(userSubscriptio      notification);

    return { result: "Notification sent to use      : " + userId };
}
```

WL.Server.createDefaultNotification API メソッドが、提供された値のデフォルト通知 JSON ブロックを作成し、返します。

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知をユーザーの 1 つまたは複数のデバイスに送信します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId :: " + userId + ",
                                                                    xt);

    var notification = WL.Server.createDefaultNotification(notificationText, 3, {foo : 'bar'});

    WL.Server.notifyAllDevices(userSubscription, notification);

    return { result: "Notification sent to user :: " + userId };
}
```

デバイスにプッシュされるテキスト

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知をユーザーの 1 つまたは複数のデバイスに送信します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId :: " + user                    xt);

    var notification = WL.Server.createDefaultNotification(notificationText, 3, {foo : 'bar'});

    WL.Server.notifyAllDevices(userSubscription, notification);

    return { result: "Notification sent to user :: " + userId };
}
```

アプリケーション・アイコンまたはタイルに表示されるバッジ番号 (これをサポートする環境内)

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知をユーザーの 1 つまたは複数のデバイスに送信します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for use
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId :: " + userId + ", text :: " + notificationText);

    var notification = WL.Server.createDefaultNotification(notificationText, 3, {foo : 'bar'});

    WL.Server.notifyAllDevices(userSubscription, notification);

    return { result: "Notification sent to user :: " + userId };
}
```

`payload` は、アプリケーションに転送され、カスタム・プロパティーを含むことができる JSON オブジェクトです。

通知 API
サーバー・サイド

- 通知を送信する。
  - 通知をユーザーの 1 つまたは複数のデバイスに送信します。

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for
    }

    var deviceSubscriptions =
        userSubscription.getDeviceSubscriptions();

    WL.Logger.debug("submitNotification >> userId ::

    var notification = WL.Server.createDefaultNotification(notificationText, 3, {foo : 'bar'});

    WL.Server.notifyAllDevices(userSubscription, notification);

    return { result: "Notificati sent to user :: " + userId };
}
```

WL.Server.notifyAllDevices API メソッドは、ユーザーにサブスクライブされているすべてのデバイスに通知を送信します。

通知 API
サーバー・サイド

- 通知を送信するための API がいくつか存在します。
  - あるユーザーのすべてのデバイスに通知を送信するには、`WL.Server.notifyAllDevices(userSubscription, options)` を使用します (スライド 33 を参照)。
  - 特定の `userSubscription` に属する特定のデバイスに通知を送信するには、`WL.Server.notifyDevice(userSubscription, device, options)` を使用します。
  - 特定のデバイスに通知を送信するには、`WL.Server.notifyDeviceSubscription(deviceSubscription, options)` を使用します。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

通知 API
クライアント・サイド

- イベント・ソース – 登録
  - 最初の作業は、アプリケーション内でイベント・ソースを登録することです。
  - IBM Worklight Foundation には、イベント・ソースの登録に使用するカスタマイズ可能な `onReadyToSubscribe` 関数が用意されています。
  - `onReadyToSubscribe` 関数は常に、グローバル JavaScript レベルでセットアップします。
  - `onReadyToSubscribe` 認証の終了時に呼び出されます。
  - API メソッド:
    `WL.Client.Push.registerEventSourceCallback(alias, adapterName, eventSourceName, callbackFunction)`

```
WL.Client.Push.onReadyToSubscribe = function(){
    WL.Client.Push.registerEventSourceCallback(
            "myPush",
            "PushAdapter",
            "PushEventSource",
            pushNotificationReceived);
};
```

通知 API
クライアント・サイド

- イベント・ソース – サブスクライブとアンサブスクライブ
  - ユーザーがサブスクライブするには、認証を受ける必要があります。
  - 以下の API を使用してイベント・ソースにサブスクライブします。

```
function subscribeButtonClicked(){
    WL.Client.Push.subscribe("myPush", {
        onSuccess: pushSubscribe_Callback,
        onFailure: pushSubscribe_Callback
    });
}

function pushSubscribe_Callback(response){
    alert("PushSubscribe_Callback invoked");
}
```

  - `WL.Client.Push.subscribe()` は以下のパラメーターを受け取ります。
    - `WL.Client.Push.registerEventSourceCallback` で宣言される**別名**
    - オプションの **onSuccess** コールバック
    - オプションの **onFailure** コールバック
  - コールバックは、必要な場合は応答オブジェクトを受け取ります。

通知 API
クライアント・サイド

- イベント・ソース – サブスクライブとアンサブスクライブ
  - 以下の API を使用してイベント・ソースからアンサブスクライブします。

```
function unsubscribeButtonClicked(){
    WL.Client.Push.unsubscribe("myPush", {
        onSuccess: pushUnsubscribe_Callback,
        onFailure: pushUnsubscribe_Callback
    });
}

function pushUnsubscribe_Callback(response){
    alert("pushUnsubscribe_Callback invoked");
}
```

  - `WL.Client.Push.unsubscribe()` は以下のパラメーターを受け取ります。
    - `WL.Client.Push.registerEventSourceCallback` で宣言される**別名**
    - オプションの **onSuccess** コールバック
    - オプションの **onFailure** コールバック
  - コールバックは、必要な場合は応答オブジェクトを受け取ります。

通知 API
クライアント・サイド

- 追加のクライアント・サイド API メソッド:
  - `WL.Client.Push.isPushSupported()` – プッシュ通知がプラットフォームでサポートされている場合は true を返し、そうでない場合は false を返します。
  - `WL.Client.Push.isSubscribed(alias)` – 現在ログインしているユーザーが指定のイベント・ソース別名にサブスクライブしているかどうかを返します。
- プッシュ通知をデバイスから受信すると、`WL.Client.Push.registerEventSourceCallback` で定義されたコールバック関数が呼び出されます。この関数は以下の引数を受け取ります。

```
function pushNotificationReceived(props, payload){
    alert("pushNotificationReceived invoked");
    alert("props :: " + Object.toJSON(props));
    alert("payload :: " + Object.toJSON(payload));
}
```

- プッシュ通知の到着時にアプリケーションがバックグラウンド・モード (または非アクティブ) であった場合、アプリケーションがフォアグラウンドに戻ると、このコールバック関数が呼び出されます。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

通知 API
タグ・ベース通知

- タグは、ユーザーが興味を示すトピックを表し、選択した興味あるトピックに従って通知を受けられる機能を提供します。
- この通知タイプでは、タグによるメッセージの送受信が可能です。
- タグ・ベース通知の受信を開始するためには、まずデバイスがアプリケーションのプッシュ通知タグにサブスクライブする必要があります。
- タグは *application-descriptor.xml* 内で次のように定義されます。

```
<tags>
    <tag>
        <name>PushTag1</name>
        <description>About PushTag1</description>
    </tag>
    <tag>
        <name>PushTag2</name>
        <description>About PushTag2</description>
    </tag>
</tags>
```

- 通知先は、アプリケーションのタグにサブスクライブしたすべてのデバイスです。

通知 API
タグ・ベース通知

- クライアント・サイド API メソッド:
  - `WL.Client.Push.subscribeTag(tagName,options)`

    デバイスを指定されたタグ名にサブスクライブします。

  - `WL.Client.Push.unsubscribeTag(tagName,options)`

    デバイスを指定されたタグ名からアンサブスクライブします。

  - `WL.Client.Push.isPushSupported()`

    プッシュ通知が当該プラットフォームによってサポートされている場合は `true` を返し、そうでない場合は `false` を返します。

  - `WL.Client.Push.isTagSubscribed(tagName)`

    デバイスが指定されたタグ名にサブスクライブされているかどうかを返します。

通知 API
タグ・ベース通知

タグ・ベース通知についての詳細は、ユーザー文書のトピック『タグ・ベース通知』を参照してください。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

通知 API
ブロードキャスト通知

- ブロードキャスト通知は、プッシュ対応の Worklight アプリケーションの場合にデフォルトで有効です。予約タグ `Push.ALL` へのサブスクリプションが、すべてのデバイスに対して作成されます。
- ブロードキャスト通知は、予約タグ `Push.ALL` からアンサブスクライブすることによって無効にすることができます。

通知 API
ブロードキャスト通知

ブロードキャスト通知についての詳細は、ユーザー文書の『ブロードキャスト通知』を参照してください。

通知 API
タグ・ベース通知とブロードキャスト通知に共通の API

- クライアント・サイド API:
  - *WL.Client.Push.onMessage (props, payload)*
    - `props` - 当該プラットフォームの通知プロパティーを含む JSON ブロック。
    - `payload` - Worklight Server から送信されるその他のデータを含む JSON ブロック。これは、タグ・ベース通知およびブロードキャスト通知のタグ名も含みます。タグ名は「tag」エレメントに現れます。ブロードキャスト通知の場合、デフォルトのタグ名は `Push.ALL` です。

```
WL.Client.Push.onMessage = function (props, payload) {
    alert("Provider notification data: " + Object.toJSON(props));
    alert("Application notification data: " + Object.toJSON(payload));
}
```

デバイスがプッシュ通知を受け取ったときに呼び出されるコールバック関数です。

この関数はグローバル JavaScript レベルで設定します。タグ名 *Push.ALL* が *payload* パラメーターに戻されます。

通知 API

タグ・ベース通知とブロードキャスト通知に共通の API

- サーバー・サイド API メソッド:
  - `WL.Server.sendMessage(applicationId,notificationOptions)`
    - `applicationId` - (必須) Worklight アプリケーションの名前。
    - `notificationOptions` - (必須) メッセージ・プロパティーを含む JSON ブロック。

指定されたターゲット・パラメーターに基づいて通知を送信します。

- メッセージ・プロパティーの完全なリストについては、「IBM Worklight Foundation ユーザー文書」を参照してください。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
  - タグ・ベース通知
  - ブロードキャスト通知
- セットアップ
  - ガイドライン
  - プロジェクト構成

セットアップ- ガイドライン

- このスライドと後続のスライドで示すように、プッシュ通知のセットアップはプラットフォームによって異なります。

Android

- プッシュ通知を Android デバイスに送信するには、Google Cloud Messaging (GCM) サービスを使用する必要があります。
- Google の GCM に登録するには、有効な Gmail アカウントが必要です。
  - GCM **プロジェクト番号**および **API キー** (これらは後で Worklight プロジェクトで使用される) を取得する方法について詳しくは、http://developer.android.com/guide/google/gcm/gs.html を参照してください。
  - **注:** API キーを作成する際には、タイプが「ブラウザー・キー」であることを確認してください。
- Android OS 2.3.5 デバイスは Gmail アカウントと同期している必要があります。
- Android OS 4.x デバイスでは Gmail アカウント同期は課されません。

セットアップ- ガイドライン

iOS

- プッシュ通知を iOS デバイスに送信するには、Apple Push Notifications Service (APNS) を使用します。
- アプリケーション用の Apple APNS 証明書を取得するには、登録済みの Apple iOS デベロッパであることが必要です。APNS 証明書には非ブランクのパスワードが必要です。
- 開発時には、証明書ファイルの名前を **apns-certificate-sandbox.p12** に変更し、環境のルート・フォルダーまたはアプリケーションのルート・フォルダーに配置してください。**環境のルート・フォルダーが最優先です。**
- 実動への移行時には、証明書ファイルの名前を **apns-certificate-production.p12** に変更し、環境のルート・フォルダーまたはアプリケーションのルート・フォルダーに配置してください。**環境のルート・フォルダーが最優先です。**
- ハイブリッド・アプリケーションに iPhone 環境と iPad 環境の両方が含まれる場合、プッシュ通知には別々の証明書が必要になります。その場合、それらの証明書は対応する環境フォルダーに配置します。

セットアップ- ガイドライン

Windows Phone 8

- プッシュ通知を Windows Phone 8 デバイスに送信するには、Microsoft Push Notifications Service (MPNS) を使用する必要があります。
- 非認証プッシュ通知は、開発者によるセットアップを必要としません。
- 認証プッシュ通知は、Windows Phone Dev Center アカウントを必要とします。
- 認証プッシュを使用するには、Microsoft トラステッド・ルート認証局によって発行された証明書を使用する必要があります。
- *実動の場合は、自分の情報が含まれないようにするために、認証プッシュ通知の使用を検討してください。非認証プッシュは開発時にのみ使用するようにしてください。*

セットアップ- ガイドライン

- プッシュ通知を送信するには、以下のサーバーが Worklight サーバーからアクセス可能でなければなりません。
- **iOS**:
  - Sandbox サーバー:
    - gateway.sandbox.push.apple.com:2195
    - feedback.sandbox.push.apple.com:2196
  - 実動サーバー:
    - gateway.push.apple.com:2195
    - Feedback.push.apple.com:2196
- **Android**:
  - オープンするポートは 5228、5229、および 5230 です。GCM は通常は 5228 のみを使用しますが、時には 5229 および 5230 を使用することもあります。
  - GCM は特定の IP アドレスを提供しないため、Google の ASN 15169 にリストされた IP ブロックに含まれるすべての IP アドレスへの発信接続をファイアウォールが受け入れられるようにする必要があります。

セットアップ- ガイドライン

Windows Phone 8:

- サーバー構成内で特定のポートをオープンする必要はありません。MPNS は正規の `http` または `https` 要求を使用します。

アジェンダ

- プッシュ通知とは
- デバイス・サポート
- 通知アーキテクチャー
- サブスクリプション管理
- 通知 API
  - サーバー・サイド
  - クライアント・サイド
- セットアップ
  - ガイドライン
  - プロジェクト構成

セットアップ– プロジェクト構成

- アプリケーションでプッシュ通知をセットアップするには、以下の行を `application-descriptor.xml` ファイルに追加します。
- これらの設定は、「設計 (Design)」モードの Application Descriptor Editor で編集することもできます。
- **Android:**

```
<android version="1.0">
    <pushSender key="GCM_key" senderId="GCM_ID"/>
```

  - GCM Web サイトで前もって作成した値を使用します。
    - **GCM_Key** に **API キー**値を入れる
    - **senderId** の代わりに**プロジェクト番号**の値を入れる

- **iOS:**

```
<iphone bundleId="com.PushNotificationsApp" version="1.0">
    <pushSender password="certificate_password"/>
```

  - Apple APNS 証明書ファイルをアプリケーション・フォルダーのルートまたは環境フォルダーのルートに配置します (スライド 51 を参照)。
  - **certificate password** を実際の証明書パスワードで置き換えます。
  - com.PushNotifications をアプリケーションの bundleId で置き換えます。Xcode プロジェクトの bundleId を作成する方法については、Apple の資料を参照してください。

セットアップ– プロジェクト構成

- **Windows Phone 8:**
- 非認証プッシュをセットアップするには、以下のようにします。

```
<windowsPhone8 version="1.0">
    <uuid>11578325-b204-42f5-acac-895430ffa09d</uuid>
    <pushSender/>
</windowsPhone8>
```

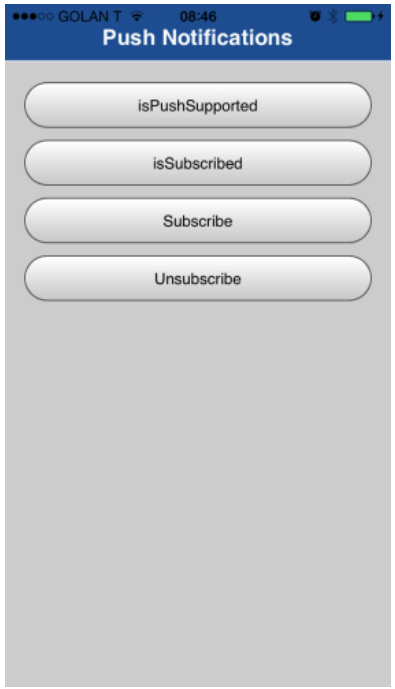
- 認証プッシュをセットアップするには、以下のようにします。

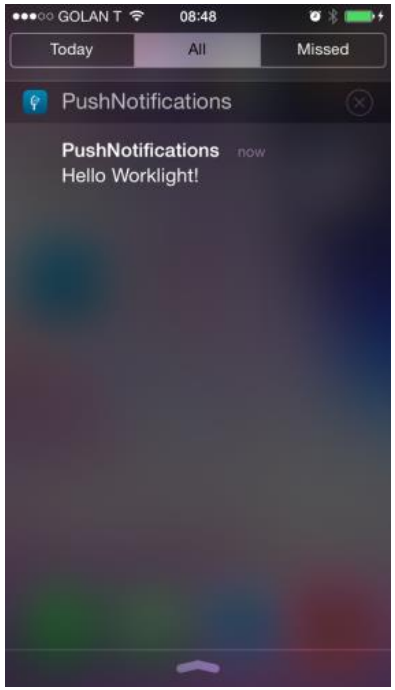
```
<windowsPhone8 version="1.0">
    <uuid>11578325-b204-42f5-acac-895430ffa09d</uuid>
    <pushSender>
        <authenticatedPush keyAlias="myAliasName" keyAliasPassword="myAliasPassword" serviceName="certificateCommonName"/>
    </pushSender>
</windowsPhone8>
```

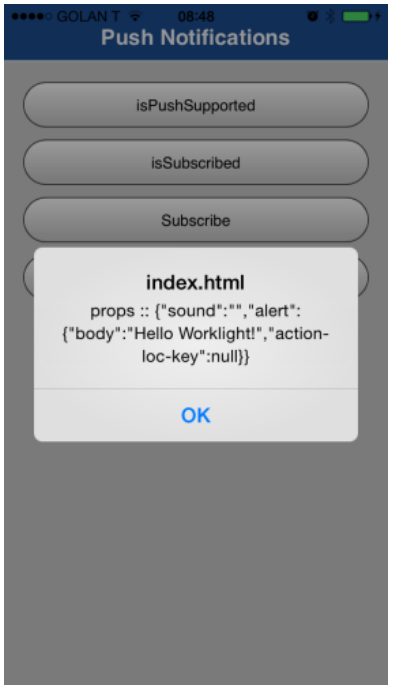
- 証明書ファイルの使用について詳しくは、「IBM Worklight Foundation ユーザー文書」のトピック「**Windows Phone 8 用プッシュ通知のセットアップ (Setting up push notifications for Windows Phone 8)**」を参照してください。

結果

- このトレーニング・モジュールのサンプルは、以下の IBM Worklight Foundation 文書 Web サイトの「入門」ページにあります。
  http://www.ibm.com/mobile-docsIMG_4185.PNG







バックエンド・エミュレーター

- このトレーニング・モジュールのサンプルには、バックエンド・エミュレーターが付属しています。このエミュレーターを使用して、バックエンド・システムによるプッシュ通知送信をシミュレートすることができます。
- このエミュレーターのソースは関連するサンプル・プロジェクトにあります。
- バックエンド・エミュレーターを実行するには、コマンド・プロンプトでサンプル・プロジェクトの **PushBackendEmulator** フォルダーを開き、提供されている JAR ファイルを以下の構文を使用して実行します。

```
java –jar PushBackendEmulator.jar <userId> <notificationText> <context root> <serverPort-optional>
```

  - `userId` は、サンプル・アプリケーションへのログインに使用したユーザー名です。
  - `contextRoot` は、Worklight プロジェクトのコンテキスト・ルートです。
- 以下に例を示します。

```
java –jar PushBackendEmulator.jar JohnDoe “My first push notification” myContextRoot 10080
```

バックエンド・エミュレーター

- バックエンド・エミュレーターは、Worklight サーバーへの接続と、`submitNotification()` アダプター・プロシージャーの呼び出しを試行します。
- コマンド・プロンプト・コンソールに呼び出し結果が出力されます。
- 成功:

```
c:\>java -jar C:\PushBackendEmulator.jar JohnDoe "hello push" PushNotificationsProject 10080
PushBackendEmulator
User Id: JohnDoe
Notification text: hello push
Server URL: http://localhost:10080/PushNotificationsProject
sending notification
Server response :: {   "isSuccessful": true,   "result": "Notification sent to user :: JohnDoe"}
```

- 失敗:

```
c:\>java -jar C:\PushBackendEmulator.jar JohnMissing "hello push" PushNotificationsProject 10080
PushBackendEmulator
User Id: JohnMissing
Notification text: hello push
Server URL: http://localhost:10080/PushNotificationsProject
sending notification
Server response :: {   "isSuccessful": true,   "result": "No subscription found for user :: JohnMissing"}
```

クイズ

答えは次のスライドにあります

- 以下の接続のうち、プッシュ通知が機能するために必要なものは次のうちどれですか。
  - クライアント・アプリケーションが APNS/GCM/MPNS サーバーに接続可能であることが必要。
  - クライアント・アプリケーションが Worklight サーバーに接続可能であることが必要。
  - Worklight サーバーが APNS/GCM/MPNS サーバーに接続可能であることが必要。
  - 上記すべて
- ユーザーは複数のデバイスを持っており、そのすべてでアプリケーションを使用しています。プッシュ通知を特定のデバイスに送信することは可能でしょうか。
  - いいえ。Worklight サーバーがプッシュ通知を送信すると、すべてのユーザー・デバイスに配信されます。
  - はい。**userSubscription** には、ユーザーがプッシュ通知への登録に使用した各デバイスの **deviceSubscription** オブジェクトが含まれています。これを使用して特定のデバイスに通知を送信できます。
  - はい。Worklight サーバーがプッシュ通知を送信すると、ユーザーがログインした最後のデバイスにこの通知が配信されます。
  - いいえ。これはセキュリティー・ブリーチと見なされます。ユーザーが複数のデバイスを持っている場合、通知はまったく送信されません。
- アプリケーション関連のカスタム・データをプッシュ通知で送信できますか。
  - いいえ。プッシュ通知には、ユーザーに対して表示される通知テキストのみを入れることができます。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。この通知を受信するために、アプリケーションはフォアグラウンドで実行されている必要があります。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。この通知を受信するために、アプリケーションはバックグラウンドで実行されている必要があります。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。プッシュ通知の到着時に実行されていなくても、アプリケーションは通知を受信できます。

クイズ- 答え

- 以下の接続のうち、プッシュ通知が機能するために必要なものは次のうちどれですか。
  - クライアント・アプリケーションが APNS/GCM/MPNS サーバーに接続可能であることが必要。
  - クライアント・アプリケーションが Worklight サーバーに接続可能であることが必要。
  - Worklight サーバーが APNS/GCM/MPNS サーバーに接続可能であることが必要。
  - 上記すべて
- ユーザーは複数のデバイスを持っており、そのすべてでアプリケーションを使用しています。プッシュ通知を特定のデバイスに送信することは可能でしょうか。
  - いいえ。Worklight サーバーがプッシュ通知を送信すると、すべてのユーザー・デバイスに配信されます。
  - はい。**userSubscription** には、ユーザーがプッシュ通知への登録に使用した各デバイスの **deviceSubscription** オブジェクトが含まれています。これを使用して特定のデバイスに通知を送信できます。
  - はい。Worklight サーバーがプッシュ通知を送信すると、ユーザーがログインした最後のデバイスにこの通知が配信されます。
  - いいえ。これはセキュリティー・ブリーチと見なされます。ユーザーが複数のデバイスを持っている場合、通知はまったく送信されません。
- アプリケーション関連のカスタム・データをプッシュ通知で送信できますか。
  - いいえ。プッシュ通知には、ユーザーに対して表示される通知テキストのみを入れることができます。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。この通知を受信するために、アプリケーションはフォアグラウンドで実行されている必要があります。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。この通知を受信するために、アプリケーションはバックグラウンドで実行されている必要があります。
  - はい。アプリケーション関連のカスタム・データをプッシュ通知で送信できます。プッシュ通知の到着時に実行されていなくても、アプリケーションは通知を受信できます。

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 Foundation 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