# Oracle Mobile Application Framework

## MAF MCS ユーティリティ開発者ガイド

ORACLE ホワイト・ペーパー | 2015 年 8 月

# 目次

## 概要

MAF MCS ユーティリティは、MAF を使用して構築されたモバイル・アプリケーションから Oracle Mobile Cloud Services (MCS)へのアクセスを簡単にする Oracle Mobile Application Framework (MAF)パブリック・サンプルおよびライブラリです。

---

*MAF MCS ユーティリティの使用は簡単です。*

---

Oracle MCS の一部として提供される Android および iOS 用の Oracle MCS モバイル・クライアント SDK と同様に、MAF MCS ユーティリティは Oracle MCS への REST サービス・リクエストをラップし、REST レスポンスとリクエスト・ペイロードを Java エンティティに変換するインフラストラクチャを提供します。

MAF MCS ユーティリティは、Oracle JDeveloper および Oracle Enterprise Packs for Eclipse (OEPE)で Oracle MAF 2.1.3 とともに提供されるパブリック・サンプルの一部です。サンプルにはユーティリティ・ソースが含まれ、すぐに使用できるように Java アーカイブ・ファイル(mafmcsutility.jar)にコンパイルされたバージョンも含まれています。また、MAF MCS ユーティリティの使用方法を開発者に示すために、任意の Oracle MCS クラウド・サービスに接続可能な汎用の Oracle MCS テスターである MAF パブリック・サンプル・アプリケーションが提供されています。

このホワイトペーパーでは、MAF MCS ユーティリティの概要を示し、Oracle MCS パブリック・クラウドに対するカスタム MAF アプリケーション開発でのユーティリティの使用方法について MAF アプリケーション開発者に説明します。ペーパーでは、MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の追加コード例の場所も示します。

## Oracle MCS プラットフォーム API

Oracle Mobile Cloud Service は、ユーザー管理サービス、データ・オフライン同期、ストレージ、分析、プッシュ通知、およびカスタム API へのアクセスを含むそのすべての機能に対する REST インタフェースを公開します。

図 1 に、モバイル・クライアント・アプリケーションから直接アクセス可能な API をハイライトして Oracle MCS プラットフォーム API を示します。モバイル・クライアント・アプリケーションから直接アクセスできない API は、Oracle MCS カスタム API 機能を使用して公開できます。

---

*Oracle Mobile Cloud Service では標準の REST インタフェースが公開されます。したがって、JSON メッセージを処理可能な REST 対応のモバイル・テクノロジまたはフレームワークは、Oracle MCS クラウド・サービス機能にクライアントとしてアクセスできます。*

---

すべてのモバイル・クライアントと Oracle MCS との対話は標準の REST 呼出しを介して行われるため、JSON (JavaScript Object Notation)ペイロードを使用した REST 呼出しを処理可能な任意のモバイル・アプリケーション開発テクノロジおよびフレームワークを使用して、Oracle MCS の機能を活用したモバイル・アプリケーションを構築できます。

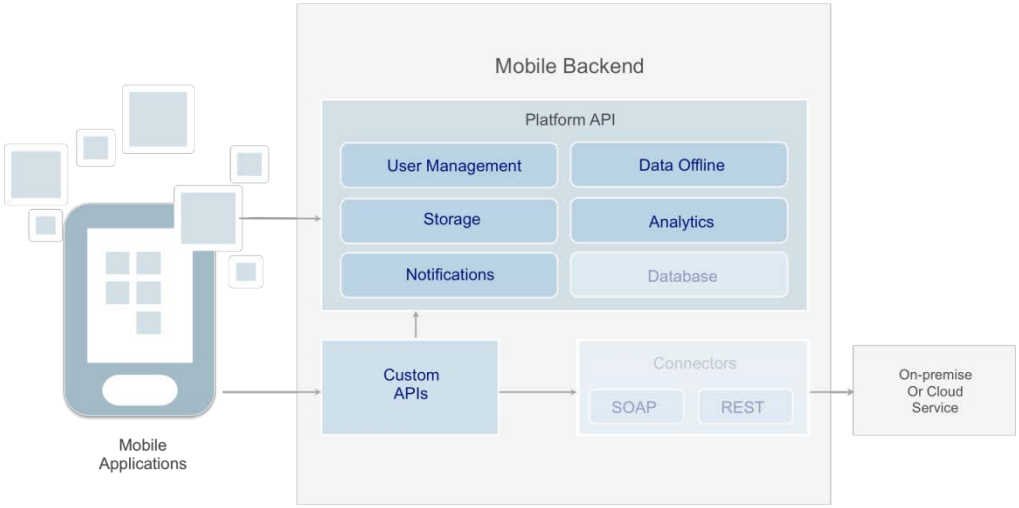


図 1: Oracle MCS プラットフォーム API の概要

Oracle MCS にアクセスするモバイル・クライアント・アプリケーションの開発を簡単にするために、Oracle MCS には Oracle MCS REST API の REST 呼出しをネイティブ API 内にラップするプラットフォーム固有のソフトウェア開発キット(SDK)が用意されています。その最初のパブリック・リリースで、Oracle MCS は Android および Objective C 用の SDK を提供します。

Oracle MCS では Oracle MAF 用の SDK は提供されません。MAF MCS ユーティリティが有用なのはこのためです。Android および iOS SDK と同様の機能が Oracle MAF アプリケーション開発者に提供されます。

## MAF MCS ユーティリティ概要

Oracle MAF MCS ユーティリティは、Oracle MCS パブリック・クラウド・サービスに簡単にアクセスするための Oracle Mobile Application Framework 用のパブリック・サンプル・ライブラリです。REST が、公開されているプラットフォーム API を介してモバイル・クライアントと対話するために Oracle MCS で使用される通信トランスポート・メカニズムです。ただし、テクノロジとしての REST はその使用において冗長で、アプリケーションの開発者がモバイル・プラットフォームによって使用されるプログラミング言語内でのみ作業するために理想的にはアプリケーションの開発者から遮蔽されます。Java が Oracle MAF モバイル・プラットフォームのプログラミング言語で、MAF MCS ユーティリティが MAF アプリケーション開発者による Java から Oracle MCS への直接アクセスを可能にするツールです。

*MAF MCS ユーティリティによって MAF モバイル・アプリケーション開発者は REST や未加工の HTTP ではなく Java で考えることができます。*

MAF MCS ユーティリティを AMX データ・コントロールまたはマネージド Bean クラスに追加されたカスタム Java コードで使用するために MAF アプリケーション開発者に必要なのは、MAF アプリケーションまたはビュー・コントローラ・プロジェクト内で MAF MCS ユーティリティ Java アーカイブ(JAR)を参照することのみです。

MAF アプリケーション開発者がユーティリティの初期動作および機能をカスタマイズまたは拡張できるよう、MAF MCS ユーティリティは Oracle MAF のパブリック・サンプルとして提供され、ソース・コードも含まれています。

注意: 設計上、MAF MCS ユーティリティの初期バージョンは MAF 2.1.2 以上で機能します。

## MAF MCS ユーティリティ・アーキテクチャ

MAF MCS ユーティリティは Android および iOS 用の Oracle MCS SDK と同様の機能を提供するだけでなく、同様のアーキテクチャも実装します。図 2 に、MAF AMX データ・コントロールおよびマネージド Bean クラスおよび Oracle MCS プラットフォーム REST API のコンテキストで表した MAF MCS ユーティリティのアーキテクチャを示します。

*Oracle MAF MCS ユーティリティは、Android および iOS 用 Oracle MCS SDK の機能およびアーキテクチャに相当します。*

**MBE マネージャ** – モバイル・バックエンド(MBE)マネージャは、Oracle MCS クラウド・インスタンスで構成および公開されたモバイル・バックエンド(MCS MBE)インスタンスを表す MBE オブジェクトのインスタンスを管理するシングルトン・オブジェクトです。

各 MBE オブジェクトは Oracle MCS MBE ルート URL、MCS MBE ID、MCS MBE 匿名キー、MCS MBE クライアント・アプリケーション・キーおよびその他のオプション設定で構成されます。Oracle MCS MBE で公開された機能にアクセスするには、MAF アプリケーション開発者は、MBE マネージャ・シングルトンに渡される MBE 構成オブジェクトを作成して MBE オブジェクト・インスタンスを作成または取得します。

**サービス・プロキシ** – 各 MBE オブジェクトは、MAF アプリケーション開発者が特定のプラットフォーム API 用に Oracle MCS への MAF REST 呼出し対話をラップするプロキシ・オブジェクトへのハンドルを取得するためのメソッドを公開します。

- **ユーザー管理** – アプリケーション開発者は、ユーザーが指定したユーザー名とパスワードのペアと Oracle MCS で有効なアカウント情報との照合を行えます。さらに、モバイル・アプリケーションで Oracle パブリック・クラウド・レルムに格納されている認証済ユーザー情報にアクセスできます。情報はすべて読み取り可能で、ユーザー権限に応じて部分的に更新可能です。
- **ストレージ** – モバイル・アプリケーションで、ドキュメントまたはイメージなどの不透明オブジェクトをアップロードまたはダウンロードし、他のモバイル・ユーザーと共有したり、特定のユーザー専用に保存したりできます。ストレージでは、カスタマ・サービス・リクエストに関連する写真の保存などのモバイル・ユース・ケースに固有のコンテンツを保存します。
- **通知** – モバイル・クライアントは Oracle MCS MBE に登録して MCS MBE インスタンスからプッシュ・メッセージを受信できます。メッセージは Oracle MCS から直接は送信されず、Google Messaging Cloud (GMC)や Apple Push Notification Service (APNS)などのモバイル・デバイス固有のプッシュ・メッセージ・プロバイダを経由します。MAF アプリケーションで通知プロキシ・サービスを使用するには、Oracle Mobile Application Framework の製品ドキュメントに記載されている追加の構成手順が必要な場合があります。

Oracle MCS MBE 通知プラットフォーム・サービスの 2 つ目の機能にはモバイル・クライアントは直接アクセスできません。GMC または APNS に送信するためには Oracle MCS でメッセージをキューイングします。この Oracle MCS API にモバイル・クライアントからアクセスするには、モバイル・アプリケーション開発者は Oracle MCS カスタム API を記述し、MAF アプリケーションでアクセスできるよう MCS MBE で公開する必要があります(カスタム API サービス・プロキシを参照してください)

- **分析** – Oracle MCS では分析イベントがサポートされます。これはモバイル・クライアントによって使用されてモバイル・アプリケーションのヘルスと使用法に関する情報、およびカスタマ・サービス・コンサルタントがカスタマ・コールに応答したかどうか、応答していない場合は拒否した理由を示すために入力する情報などのビジネス関連の情報を送信します。分析イベントはモバイル・クライアントで収集され、一括(セッションと呼ばれる)して MCS MBE インスタンスに送信されます。このインスタンスで Oracle MCS 分析エンジンは Oracle MCS 管理ポータルでのグラフィック表示用にイベントを保存および収集します。
  MAF MCS ユーティリティの分析サービス・プロキシによって MAF アプリケーション開発者はイベントを作成して Oracle MCS に送信できます。ネットワークの中断によって送信に失敗したイベントは、後でネットワークが使用可能になったときのためにローカルに保存されます。

- **カスタム API** – MAF アプリケーション開発者は、MCS MBE で公開された REST API を起動できます。この機能は、主に Oracle MCS に公開されたカスタム API にアクセスするために設計されていますが、任意のプラットフォーム API にも使用できます。カスタム API サービス・プロキシとその他のサービス・プロキシとの違いは、JSON REST メッセージが自動的には Java オブジェクトにシリアライズされず、未加工の Java String または byte[]形式で提供されることです。それに応じてメッセージを解析することは MAF アプリケーション開発者の責任です。この場合に MAF MCS ユーティリティで提供されるのは、必要な Authorization ヘッダーの送信およびエラー処理ルーチンを含む Oracle MCS へのアクセスです。

**Rest サービス・アダプタ** – MAF MCS ユーティリティは Oracle MAF Rest サービス・アダプタをラップし、Oracle MCS との REST 対話を簡易化します。このラッピングの一環として Oracle MAF MCS ユーティリティは、呼出しおよび実行時に動的にオン/オフの切替えが可能な MBE インスタンスに固有のロギングに認可ヘッダーを追加します。REST サービス・アダプタ・ラッパーは MAF MCS ユーティリティの実装に固有と考えられるため、モバイル・アプリケーション開発者に公開されません。これによって、MAF MCS ユーティリティがユーティリティを使用して構築されるモバイル・アプリケーションに影響することなく、Oracle MAF または Oracle MCS の将来的な変更に適応できるようにするために必要な余地が与えられます。

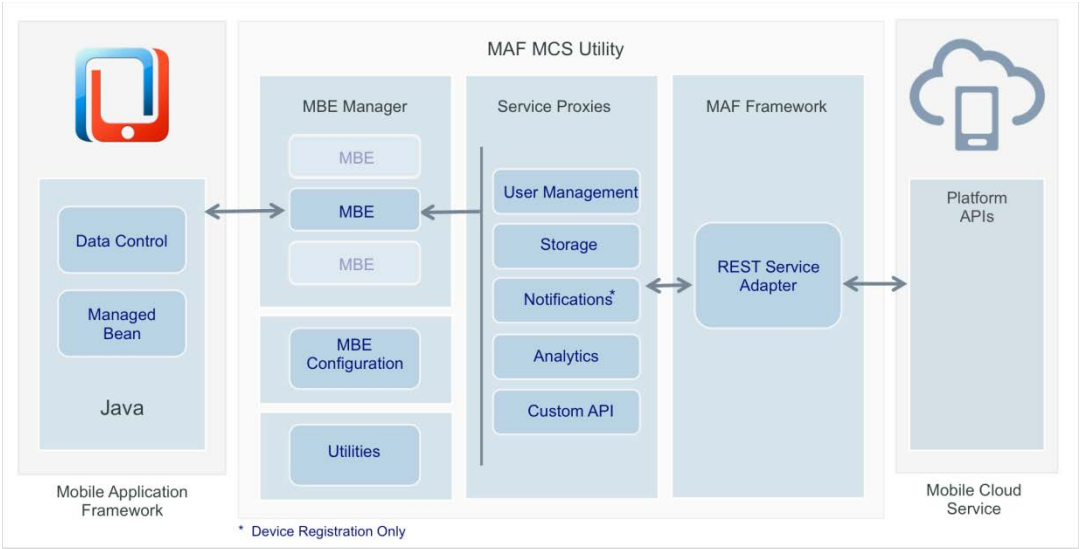


図 2: MAF MCS ユーティリティ・アーキテクチャ

MAF MS ユーティリティには、ロギング用のハッシュ・マップの内容のダンプ、日付の書式設定やその他の一般的な関連タスクなどにモバイル・アプリケーションで使用できるユーティリティ・クラスおよびメソッドのセットが用意されています。

注意: 図 2 では、MAF アプリケーションと MAF MCS ユーティリティが 2 つの別個のボックスで示されています。ボックスは、MAF MCS ユーティリティに属するものと MAF に属するものを示すためのものです。物理的には MAF MCS ユーティリティは MAF アプリケーションとともにモバイル・デバイスにデプロイされます。

Android および iOS 用 Oracle MCS SDK で提供され、Oracle MAF MCS ユーティリティで提供されない機能はオフライン・ストレージおよび同期です。現在の MAF では、モバイル・アプリケーションにオフライン機能を必要とするモバイル・アプリケーション開発者は、デバイス上のコンテンツの永続性および同期を手動で実装する必要があります。MAF MCS ユーティリティはこのコンテキストでも使用できます。

## MAF MCS ユーティリティ・サンプル・アプリケーション

MAF MCS ユーティリティには、ユーティリティを使用して Oracle Mobile Cloud Services にアクセスする方法を示すサンプル・コードを含む Oracle MAF パブリック・サンプルも付属しています。パブリック・サンプル・アプリケーションは、アプリケーション・コントローラ・プロジェクトに MAF MCS ユーティリティが構成された汎用 MCS テスターです。MAF MCS ユーティリティの機能はデータ・コントロールで公開されており、開発者はすべてのサンプル・コードを 1 箇所で見つけることができます。

---

*Oracle MAF 内の MAF MCS ユーティリティのパブリック・サンプルは、Oracle MCS パブリック・クラウドの試用または有料サブスクリプション・インスタンスに対して実行される汎用 Oracle MCS テスターです。*

---

MAF MCS ユーティリティ・パブリック・サンプルは Apple シミュレータまたは Android エミュレータで実行したり、iOS または Android デバイスにデプロイできます。最高のユーザー・エクスペリエンスを得るには、より大きな画面領域を利用できるよう MAF MCS ユーティリティ・パブリック・サンプルをモバイル・タブレットで実行するか、シミュレータまたはエミュレータで同等のフォーム・ファクタを使用することをお薦めします。

図 3 に、アプリケーションを起動する際ただちに指定する必要のある、あらかじめ必要な情報についてアプリケーション・ユーザーに案内している初期サンプル・アプリケーション画面を示します。情報を読み終えたら、アプリケーション・ユーザーはハンバーガー・アイコンを押して、デモ起動エントリを含むスライド・メニューを開きます。

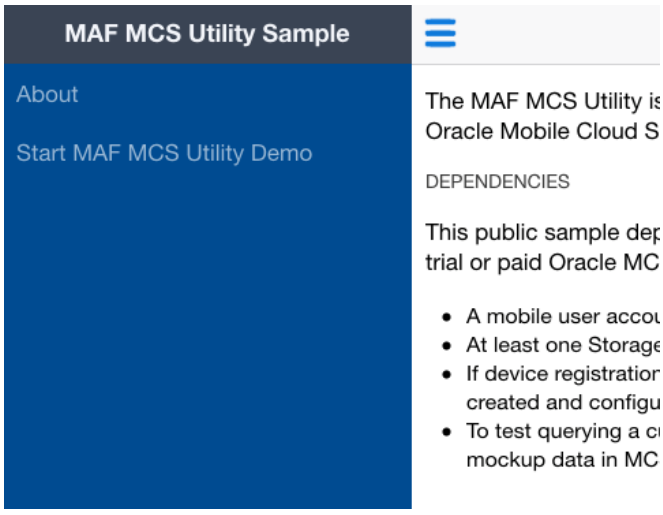


図 3: MAF MCS ユーティリティ・サンプルようこそ画面

ユーザーが初めてアプリケーションを実行する場合、図 4 に示すプリファレンス画面が表示され、ユーザーはサンプル・アプリケーションを介してアクセスおよびテストする Oracle MCS モバイル・バックエンド(MBE)インスタンスに関する情報を指定します。

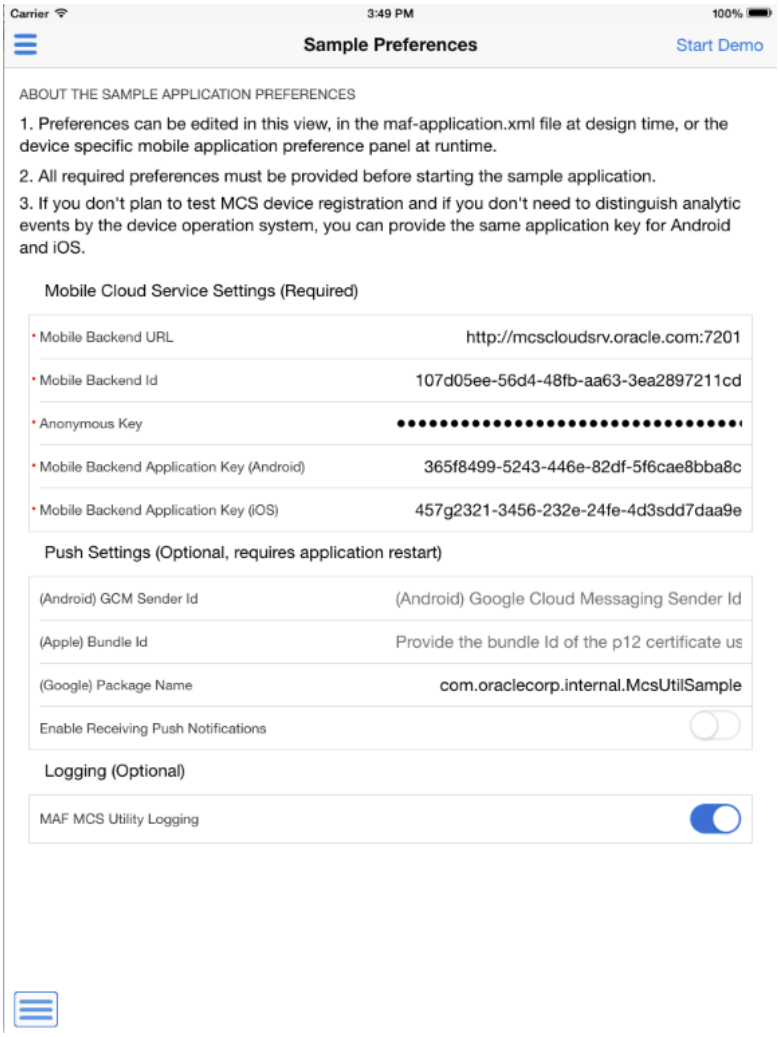


図 4: MAF MCS ユーティリティ・サンプル・プリファレンス画面

注意: プリファレンス設定は、ログイン画面(図 5)の上部に表示される"Edit Preferences"リンクをクリックするか、モバイル・デバイス固有のプリファレンス設定に移動して後でいつでも変更できます。

図 5 に示すログイン画面は、アプリケーション・プリファレンスの設定後サンプル・アプリケーションを起動したときにアプリケーション・ユーザーに表示される最初の画面です。以降のアプリケーションの起動では、プリファレンス画面は自動的に省略されます。

図 5 のログイン画面で、ユーザーはモバイル・ユーザー名およびパスワードを使用して MCS に対する認証を受けるか、匿名ユーザーの資格証明を使用して認証を受けることができます。

匿名ユーザーには図 6 に示したメニューと同じものではなく、選択肢の少ないメニューが表示されることに注意してください。すべての Oracle MCS プラットフォームを匿名ユーザーが使用できるわけではなく、サンプル・アプリケーション・メニューにはこれが反映されます。

注意: 匿名ユーザーはカスタム API テスターにアクセスできます。ただし、認証を受けたユーザー・アカウントが必要なカスタム API の起動はエラーで失敗することに注意してください。

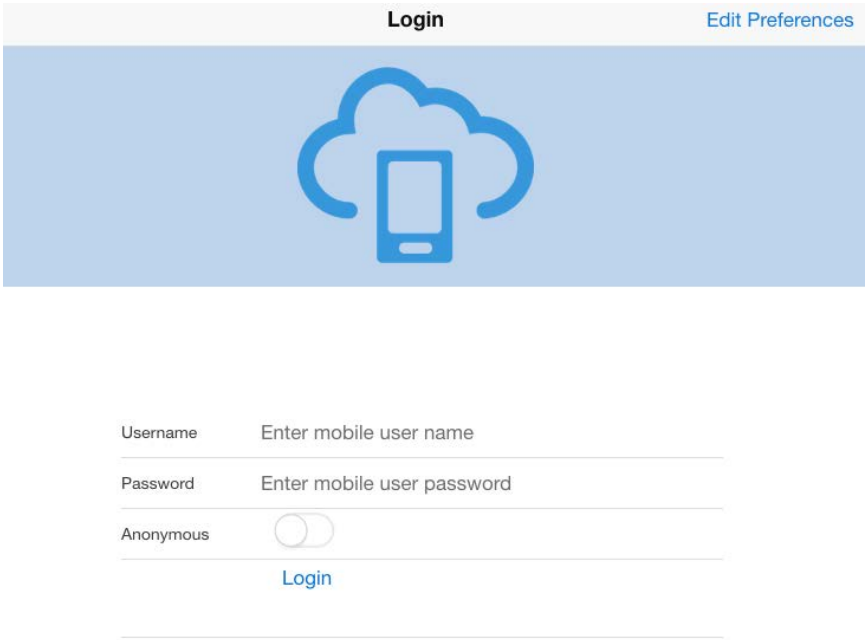


図 5: MAF MCS ユーティリティ・サンプル・ログイン

図 6 に、ユーザーが特定の API を選択してリモートの Oracle MCS MBE に対するテストを実行できるパブリック・サンプル・テスター・メニューを示します。**「Analytics API」メニュー**・アイテムでは、ショッピング・カードの選択をリモート MCS 分析エンジンにレポートする分析イベントをキューイングできる画面が表示されます。

注意: 分析イベントを送信するために Oracle MAF はジオロケーション情報の取得を試みますが、これはシミュレータおよびエミュレータでは使用できません。この場合、および分析イベント API が初めて起動される場合のみ、Oracle MAF はタイムアウトが 1 分後に起こるのを待ちます。これによって、ユーザーによる分析イベントの MCS への送信と MCS ポータルの分析ダッシュボードでのイベントの表示との間に 1 分間の遅延が発生します。アプリケーションをモバイル・デバイスにデプロイした場合はこのかぎりではありません。

**「MCS Storage API」メニュー・エントリ**では、MCS MBE インスタンスで公開されている使用可能なコレクションのリストを表示し、格納されているコンテンツをアップロード、ダウンロードおよび削除できるページのセットに移動します。モバイル・デバイスから Oracle MCS へのコンテンツのアップロードのデモのために、MAF サンプル・アプリケーションは PNG、PDF および MP4 ファイルなどのサンプル・ドキュメントとともにデプロイされています。

**「Custom API」メニュー・エントリ**では、ユーザーがカスタム API URI、使用する HTTP メソッド、および送信する JSON ペイロード(オプション)に関する情報を指定する表示に移動します。Oracle MCS から返されるメッセージは未加工の形式で表示されます。

**「MCS Device Registration API」メニュー・エントリ**では、Google および Apple からメッセージを受信するために Oracle MAF によって必要とされる、あらかじめ必要な構成についてユーザーに知らせる追加指示の表示に移動します。

構成後、サンプル・アプリケーションは Google または Apple から取得したクライアント・トークン、MCS クライアント登録の確認および Oracle MCS によってキューイングされた着信プッシュ・メッセージのコンテンツを表示します。

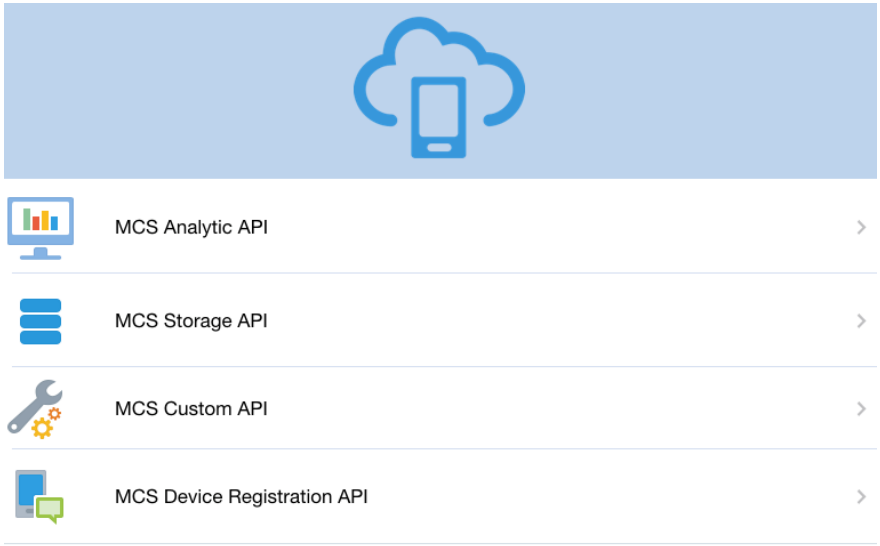


図 6: MAF MCS ユーティリティ・テスター・メニュー

MAF MCS ユーティリティ・パブリック・サンプルでは、MAF アプリケーション開発者に MAF MCS ユーティリティ API の同期および非同期の起動例を含むサンプル・コードが提供されます。

## MAF MCS ユーティリティ構成

MAF MCS ユーティリティをカスタム MAF アプリケーション開発で使用するには、mafmcsutility.jar アーカイブ・ファイルをアプリケーションのアプリケーション・コントローラおよびビュー・コントローラ・プロジェクトに追加する必要があります。デフォルトの MAF プロジェクト名を使用する MAF アプリケーションの場合、アプリケーション・コントローラ・プロジェクト名は"ApplicationController"で、ビュー・コントローラ・プロジェクト名は"ViewController"です。

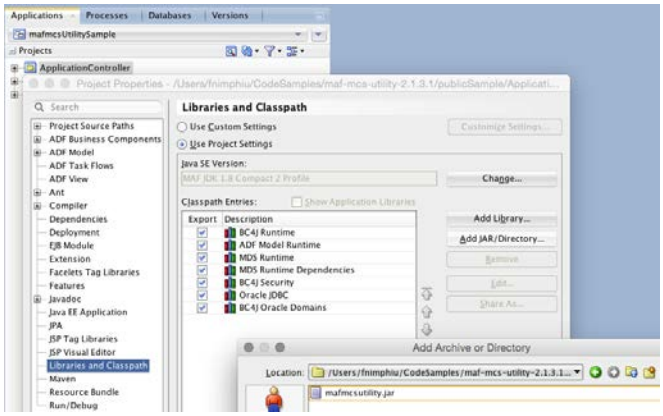


図 7: mafmcsutility.jar 構成

mafmcsutility.jar ファイルを"ApplicationController"プロジェクトに追加するには、Oracle JDeveloper でプロジェクト・ノードを選択し、右マウス・メニューから「プロジェクト・プロパティ」を選択します。「プロジェクト・プロパティ」ダイアログで「ライブラリとクラスパス」エントリを選択し、「JAR/ディレクトリの追加」ボタンを押して、コンピュータのファイル・システム上の mafmcsutility.jar ファイルを選択します。アーカイブ・ファイルを選択した状態で「開く」、「OK」の順に押します。図 7 に、前述のダイアログのスクリーンショットを示します。

注意: Oracle JDeveloper の場合を説明しましたが、MAF MCS ユーティリティは Oracle Enterprise for Eclipse (OEPE)で開発された MAF アプリケーションについても構成できます。OEPE でも mafcsutilit.jar をアプリケーションまたはビュー・コントローラ・プロジェクトについて構成できます。

"ViewController"プロジェクトでの mafmcsutility.jar ファイルの構成は、図 7 に示す ApplicationController プロジェクトの構成と同様です。唯一の違いは、ビュー・コントローラ・プロジェクト・ノードで「プロジェクト・プロパティ」メニュー・エントリを選択することです。

**MAF MCS ユーティリティのスコープについて**

アプリケーション・コントローラ・プロジェクトで構成およびインスタンス化されたデータ・コントロールは、アプリケーション・スコープのマネージド Bean 同様 MAF アプリケーション全体で使用できます。したがって、Oracle MCS からの問合せのデータは MAF 機能間で共有されます。

反対に、ビュー・コントローラ・プロジェクトで定義されたデータ・コントロールでは MAF の機能間でインスタンスは共有されず、そのため、データも共有されません。

通常のアプリケーション開発同様ユース・ケースが重要で、データ・コントロールをどこで定義するかについて何が正しくて何が間違っているということはありません。MAF MCS ユーティリティでは、MAF の機能と Oracle MCS MBE との 1 対 1 のマッピングが必要な場合、ビュー・コントローラ・プロジェクトでデータ・コントロールを作成します。これは mafmcsutility.jar ファイルを ViewController プロジェクトで構成することも意味します。この場合、MAF 機能での各データ・コントロールの使用で独自の MAF MCS ユーティリティ MBE マネージャのインスタンスおよび含まれる MBE インスタンスが作成されます。

ただし、Oracle MCS MBE 機能および MCS から問い合されたデータが異なる MAF 機能からアクセスされる必要がある場合、ApplicationController プロジェクトでデータ・コントロールを作成し、アプリケーション・コントローラ・プロジェクトおよびオプションで ViewController プロジェクトで MAF MCS ユーティリティ・アーカイブを構成します。ViewController では、MAF 機能で使用するために MAF MCS ユーティリティ・クラスがデータ・コントロールで公開される場合 mafmcsutility.jar が構成されている必要があります。この設定を使用する場合、MBEManager インスタンスがデータ・コントロールのみを介して公開され、機能のマネージド Bean などから直接アクセスされないようにする必要があることに注意してください。

**パブリック・サンプル内の MAF MCS ユーティリティ構成**

Oracle MAF 2.1.3 以上のバージョンに付属の MAF MCS ユーティリティ・パブリック・サンプル・アプリケーションでは、ユーティリティ・ライブラリが ApplicationController プロジェクトで構成されており、`MobileBackendDC.java` データ・コントロール・クラス内で使用されます。さらに、ViewController プロジェクトでは mafmcsutility.jar は使用されず、かわりにデータ・コントロールは Oracle MCS からの問合せの結果用に汎用オブジェクトを公開します。データ型変換をしない場合、かわりに mafcsutility.jar を ViewController プロジェクトで構成し、MAF データ・コントロールの結果を MAF MCS ユーティリティの型に解析します(マネージド Bean からデータ・コントロールへのアクセスと仮定します)。繰り返しになりますが、ここではどの実装が正しいということではなく、ユース・ケースによってデータ・コントロールをどこで実装するか、MAF MCS ユーティリティ・ライブラリをどこで構成するかが定義されます。

---

*MAF MCS ユーティリティ・パブリック・サンプルは、開発者がコード例を見つけたり、ユーティリティの機能について調べる場合に最適のリソースです。*

---

MAF MCS ユーティリティ・パブリック・サンプルは実例を示すためのもので、Oracle MAF MCS ユーティリティの使用方法を説明するコード・コメントが多数含まれています。開発者がコード例を見つけたり、ユーティリティの機能について調べる場合に最適のリソースです。MAF パブリック・サンプルの一部として提供される MAF MCS ユーティリティ・ソース・コードに詳細なコード・コメントが含まれていることにも注意してください。

# はじめに: MAF MCS ユーティリティ・ベース・クラス

Oracle MCS パブリック・クラウドに対する MAF アプリケーション開発の出発点となる MAF MCS ユーティリティ・クラスは次の 3 つのです

- com.oracle.maf.sample.mcs.shared.mbe.MBEConfiguration
- com.oracle.maf.sample.mcs.shared.mbe.MBEManager
- com.oracle.maf.sample.mcs.shared.mbe.MBE

以降の項でクラスの用途および公開されるメソッドについて説明します。

## MBEConfiguration クラス

`MBEConfiguration` オブジェクトを作成して、要求した `MBE` インスタンスの初期構成設定を `MBEManager` クラスに渡す必要があります。MBEConfiguration オブジェクトは MBE インスタンスにコピーされ、そこからロギングや分析イベントの動作を変更するなどのために実行時にアクセスされます。

### コンストラクタ

コンストラクタには、Oracle MCS MBE ベース URL を指す MAF REST 接続、一意の MCS MBE ID、MCS MBE 匿名キー、Android および iOS 用のクライアント・アプリケーション・キーおよび認証タイプに関する情報が必要です。Oracle MCS パブリック・クラウドの最初のリリースでは Basic 認証のみサポートされるため、認証タイプは MBEConfiguration.AuthenticationType.BASIC_AUTH に設定される必要があります。

```
MBEConfiguration(String mafRestConnectionName, String mobileBackendId,
                 String mbeAnonymousKey, String mbeClientApplicationKey,
                 AuthenticationType authenticationType)
```

注意: 構成オブジェクトを作成する前に、(http://mcscloudsrv.oracle.com:7201 などのような)Oracle MCS MBE ベース URL を保持するための MAF REST 接続を作成する必要があります。

いずれかの引数が null 値を持つ場合、または値の長さがゼロの場合、コンストラクタは `oracle.adfmf.framework.exception.IllegalArgumentException` 例外をスローします。

### メソッド

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| getAppleBundleId | Apple iOS デバイスにデプロイされた MAF アプリケーションのアプリケーション・バンドル ID が返されます。バンドル ID は MAF アプリケーションを MCS に登録し、プッシュ・メッセージを受信するために使用されます。 |
| getAuthenticatedUsername | 認証されたユーザーのユーザー名が返されるか、MBE インスタンスが認証されていない場合は null が返されます。 |
| getAuthorizationToken | 認可のために Oracle MCS への各 REST リクエストに追加された、Base 64 エンコードされた認可トークン |
| getAuthtype | MBE インスタンスが Basic 認可を使用して認証されるか、OAUTH を使用して認証されるか。Oracle MCS の最初のリリースでは、認証は常に Basic です。 |

| | |
|---|---|
| getDeviceToken | 通知が有効な場合、Google GMC または Apple APNS プッシュ・プロバイダによって返されるデバイス・トークンが返されます。情報は MAF MCS ユーティリティによって使用され、MAF アプリケーションを MCS に登録し、プッシュ・メッセージを登録します |
| getGooglePackageName | Android デバイスにデプロイされた MAF アプリケーションのアプリケーション・パッケージ名が返されます。パッケージ名は MAF アプリケーションを MCS に登録し、プッシュ・メッセージを受信するために使用されます。 |
| getLogger | ログ・メッセージを書き込む MBE 固有のロガーへのアクセスが提供されます。ログ・メッセージは構成オブジェクトとともに使用される MBE インスタンスのコンテキストで出力されます。MBE ロガーの出力では常に MBE ID がログ・メッセージに追加されます |
| getMobileBackendBaseURL | MAF アプリケーション開発者が、MBE とともに使用される MAF REST 接続で構成されている Oracle MCS MBE ベース URL 文字列を取得できます。この情報は、MCS 接続の問題のデバッグなどに有用です |
| getMobileBackendClientApplicationKey | MBEConfiguration オブジェクトとともに使用される MBE インスタンスに対して MCS MBE で定義されるクライアント・アプリケーション・キーが返されます。キーは通常 iOS と Android アプリケーションで異なり、分析および通知で使用されます。 |
| getMobileBackendId | MAF MCS ユーティリティの MBE インスタンスの接続先である Oracle MCS 内のモバイル・バックエンドの MCS MBE ID が返されます |
| isEnableAnalytics | 分析は実行時に有効および無効にし、Oracle MCS 分析エンジンに送信される情報の量を削減できます。MAF アプリケーションでこの設定を確認し、分析イベントをキューイングしたり、分析を無視できます |
| isLoggingEnabled | ロギングは実行時に有効または無効にできます。このメソッドによって MAF 開発者はログ・メッセージが書き込まれたかどうかを確認できます |
| isManualAuthentication | MAF MCS ユーティリティでは、MCS への認証が MAF MCS ユーティリティを介して処理される手動認証、および認証が MAF によって処理される MAF 機能レベルの認証がサポートされます。このメソッドによって MAF 開発者は特定の MBE で使用される認証のタイプを確認できます |

### MBEManager クラス

MAF MCS ユーティリティ `MBEManager` クラスはファクトリ・クラスで、出発点となるファサードです。`MBEManager` クラスは、指定された `MBEConfiguation` オブジェクトに基づいて `MBE` インスタンスを作成および管理します。`MBE` インスタンスは、公開したサービス・プロキシ・クラスを介してすべての Oracle MCS プラットフォーム API 機能への Java アクセスを提供します。

**コンストラクタ**

MBEManager はシングルトンでファクトリ・メソッドを介して呼び出されます。MAF の MBEManager インスタンスはクラス・ローダーごとに作成されます。つまり、アプリケーション・コントローラ・プロジェクトで定義された MBEManager インスタンスは、MAF 機能ごとにインスタンスが作成されるビュー・コントローラ・プロジェクトで定義された MBEManager インスタンスとは異なります。

推奨される方法は、アプリケーション・コントローラ・プロジェクトで MAF データ・コントロールに MAF MCS ユーティリティ参照を定義し、常にデータ・コントロールを介してアクセスする方法です。このようにすると、MBE インスタンスの数を最小限に保て、Oracle MCS から問い合せたデータを機能間で共有することもできます。

ただし、モバイル・アプリケーション・ユース・ケースで MAF 機能とサーバー側 MCS MBE との 1 対 1 マッピングが予測される場合、ビュー・コントローラ・プロジェクトでデータ・コントロールまたはマネージド Bean に MBEManager を定義します。

```
MBEManager.getManager()
```

**メソッド**

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| createOrRenewMobileBackend | 新規 MBE インスタンスが MBEConfiguration オブジェクト設定に基づいて作成されます。指定した名前の MBE インスタンスが存在する場合、それは取り消され、新しい構成設定を使用して再作成されます。<br>注意: 登録されている MBE インスタンスの名前を一意にするには、MCS MBE ID を名前として使用できます。 |
| existMobilBackendWithName | MAF アプリケーション開発者が、指定した名前の MBE がすでに MBEManager インスタンスに存在するかどうかをテストできます。 |
| getMobileBackend | 指定された名前に関連付けられている MBE インスタンスが取得されるか、その名前に対する MBE インスタンスが見つからない場合は null が返されます |
| releaseAllMobileBackend | この MBEManager によって管理されているすべての MBE インスタンスが解放および削除されます |
| releaseNamedMobileBackend | この MBEManager によって管理されている、指定された MBE インスタンスが解放および削除されます |

## MBE クラス

モバイル・バックエンド(MBE)クラスは、パブリック・クラウド内の特定の Oracle MCS MBE へのアクセス・ファサードです。MBE クラスは、Oracle MCS プラットフォーム API の機能をラップするサービス・プロキシ・クラス(後述)を公開します。

**コンストラクタ**

MBE コンストラクタは MBEManager によって呼び出され、MAF アプリケーション開発者によって呼び出されません。

```
MBE(String mobileBackendName, MBEConfiguration mbeConfig)
```

**メソッド**

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| getApplicationFeatureName | この MBE が使用されている現在の機能の名前が取得されます。このメソッドは、Oracle MAF で MAF 機能と MCS MBE との間に 1 対 1 マッピングがあるユース・ケースの場合に有用です。 |
| getAuthorizationProvider | MAF アプリケーション開発者が MCS MBE パブリック・クラウドに対する `MBE` の手動認証に使用する認可プロバイダのインスタンスが返されます。<br>最初のリリースでは、Basic 認証のみサポートされます。今後のバージョンでは OAUTH もサポートされます。<br>`com.oracle.maf.sample.mcs.shared.authorization.auth.Authorization` |
| getMbeConfiguration | `MBE` インスタンスの `MBEConfiguration` オブジェクトが返されます。 |
| getServiceProxy | `MBE` によって公開されているサービス・プロキシの 1 つにアクセスする汎用メソッド。MAF 開発者は、型安全なメソッドを使用してサービス・プロキシ・インスタンスにアクセスすることをお薦めします。 |
| getServiceProxyAnalytics | Oracle MCS で分析プラットフォーム API にアクセスするためのサービス・プロキシ・クラスのインスタンスが返されます |
| getServiceProxyStorage | Oracle MCS でストレージ・プラットフォーム API にアクセスするためのサービス・プロキシ・クラスのインスタンスが返されます |
| getServiceProxyUserInfo | Oracle MCS でユーザー管理 API プラットフォーム API にアクセスするためのサービス・プロキシ・クラスのインスタンスが返されます |
| getServiceProxyCustomAPI | Oracle MCS MBE で定義されているカスタム API にアクセスするためのサービス・プロキシ・クラスのインスタンスが返されます |
| getServiceProxyNotifications | Oracle MCS で通知プラットフォーム API にアクセスするためのサービス・プロキシ・クラスのインスタンスが返されます |

### 例: MAF MCS ユーティリティ MBE インスタンスの作成

この項のコード例は、`MBEConfiguration` のインスタンスの作成および構成から始まります。その後の構成オブジェクトの定義の方が、Oracle MCS パブリック・クラウドへの MAF MCS ユーティリティ・アクセスの作成においてより大きな部分を占めます。

```
/* Create a handle to an MBE class instance */

mobileBackend = MBEManager.getManager().createOrRenewMobileBackend(
                                    mobileBackendId,mbeConfiguration);

//access the MBE specific logger instance
logger = mobileBackend.getMbeConfiguration().getLogger();
```

`MBE` インスタンスは `MBEManager` によって作成および管理されますが、構成オブジェクト `MBEConfiguration` を介して構成されます。構成オブジェクトを作成する前に、(http://mcscloudsrv.oracle.com:7201 などのような)Oracle MCS MBE ベース URL を保持するための MAF REST 接続を作成する必要があります。この例では、REST 接続名は"mcsmbeurl"とします

```
/* global variable declaration e.g. in data control

private MBE mobileBackend = null;
// If you need to write log messages for an MBE instance
private MBELogger logger = null;

…
/* Create MBEConfig configuration object */
String mobileBackendId = "107d05ee-56d4-48fb-aa63-3ea2897211cd"
String mbeAnonymousKey = "UFJJTUVfREVDRVBUSUNPTl9NT0JJTE"


// Provide the client application keys created in MCS MBE.The client application
// is a configuration in MCS MBE that is mainly used for configuring push messages
// MAF MCS Utility however needs a the key for Analytics also, even if push is not
// configured
String appKeyAndroid = "365f8499-5243-446e-82df-5f6cae8bba8c";
String appKeyiOS = "457g2321-3456-232e-24fe-4d3sdd7daa9e";


// MBE specific logging in MAF MCS Utility can be enabled / disabled at runtime
boolean loggingEnabled = false;

// Detect platform MAF runs on
DeviceManager deviceManager = DeviceManagerFactory.getDeviceManager();
String _toUppercaseManufacturerOS = deviceManager.getOs().toUpperCase();
boolean isIOS = _toUppercaseManufacturerOS.equals("IOS")?true:false;
// Assign application client key
String appKey = isIOS? appKeyiOS : appKeyAndroid;

// MBE instance need to be configured to support a specific authentication mechanism.
// At the moment only Basic Authentication is supported.

MBEConfiguration mbeConfiguration = new MBEConfiguration("mcsmbeurl", mobileBackendId,
                 mbeAnonymousKey,appKey, MBEConfiguration.AuthenticationType.BASIC_AUTH);
```

```
// Enable analytic event feature for MBE
mbeConfiguration.setEnableAnalytics(true);
// logging can be enabled / disabled at runtime for a MBE instance. Note that logging for
// An MBE requires logging to be enabled for the MAF application too.
mbeConfiguration.setLoggingEnabled(loggingEnabled);

// Try to identify the device so that analytics can distinguish between the devices owned
// by a person
mbeConfiguration.setMobileDeviceId(DeviceManagerFactory.getDeviceManager().getName());

/* Get handle to MBE instance */

mobileBackend = MBEManager.getManager().createOrRenewMobileBackend(
                                    mobileBackendId, mbeConfiguration);

//access the MBE specific logger instance
logger = mobileBackend.getMbeConfiguration().getLogger();
```

前述のコードで MAF アプリケーションは、`MBE` インスタンスで公開されたプロキシ・サービスを介して Oracle MCS にアクセスする準備が整います。前述のコードの大部分が MBE インスタンスの構成を定義するためのものです。

MBE オブジェクトの構成オブジェクトは、MBE インスタンスで `getMbeConfiguration()` を呼び出して実行時にアクセスでき、開発者がロギングなどの機能のオン/オフを切り替えたり、構成設定にアクセスできます。

注意: もともと作成されていた構成オブジェクトを変更することはそこから作成された MBE インスタンスに影響しません。MBE 構成を変更するには、必ず `MBE` インスタンスまたは公開されているサービス・プロキシの 1 つを介して `MBEConfiguration` にアクセスする必要があります。

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC.java` クラスで MAF MCS ユーティリティ `MBE` インスタンスをインスタンス化します。アプリケーション・ユーザーがアプリケーション・プリファレンス・ページでアクセスする MCS MBE の詳細情報を指定すると、`prepareMCSAccess()` メソッドがタスク・フロー・メソッド・アクティビティから起動されます。サンプル・コードでは、アプリケーション・プリファレンスからの `MBEConfiguration` オブジェクトの構成方法も示され、プリファレンスの情報で MAF Rest 接続をオーバーライドする方法も含まれます。

## MAF MCS ユーティリティ MBE サービス・プロキシの使用

Oracle MCS MBE プラットフォーム API は、MAF MCS ユーティリティでサービス・プロキシ・クラスによって表されます。サービス・プロキシ・クラスは基礎となる Oracle MCS の REST 呼出しをラップするだけでなく、Java でサーバー・レスポンスを処理するインフラストラクチャ・クラスも提供します。MAF MCS ユーティリティに含まれるサービス・プロキシ・クラスは次のとおりです

- com.oracle.maf.sample.mcs.apis.userinfo.UserInfo
- com.oracle.maf.sample.mcs.apis.analytics.Analytics
- com.oracle.maf.sample.mcs.apis.storage.Storage
- com.oracle.maf.sample.mcs.apis.custom.CustomAPI
- com.oracle.maf.sample.mcs.apis.notifications.Notifications

### UserInfo サービス・プロキシ

`com.oracle.maf.sample.mcs.apis.userinfo.UserInfo` プロキシ・クラスでは、MAF アプリケーションで Oracle MCS セキュリティ・レルムに保存されている認証済ユーザー・プロファイル情報(ユーザーのカスタム・プロパティを含む)にアクセスできます。更新可能なレルム属性の場合、UserInfo プロキシを使用してユーザー情報を変更できます。

#### コンストラクタ

MAF アプリケーション開発者はコンストラクタを直接呼び出すことはできません。`MBE` インスタンスを介してプロキシ・クラスにアクセスします。

```
mbe.getServiceProxyUserInfo()
```

#### メソッド

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| getCurrentUserInformation | 指定したユーザー・プロパティを含む `com.oracle.maf.sample.mcs.apis.userinfo.User` のオブジェクト・インスタンスおよび特定のセキュリティ・レルム用の追加プロパティを含む HashMap が返されます |
| updateCurrentUserInformation | MAF アプリケーション開発者がプロパティの名前と値を含む HashMap オブジェクトを介して MCS セキュリティ・レルム内のユーザー情報を更新できます。更新可能ではないレルム・プロパティ(ユーザー名など)もあることに注意してください。エラーが発生しないようにするには、MCS MBE API テスター、cURL や Google Postman などのテスト・ツールでプロパティの更新をテストすることをお薦めします。 |

#### MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスから `UserInfo` プロキシ・クラスにアクセスします。getUserInformation メソッドでは、ビュー・コントローラ・プロジェクトにある `UserManagementBacking` Bean でアクセスおよび解析される JSON 文字列が返されます。図 8 に、実行時の UserManagement.amx の表示を示します。

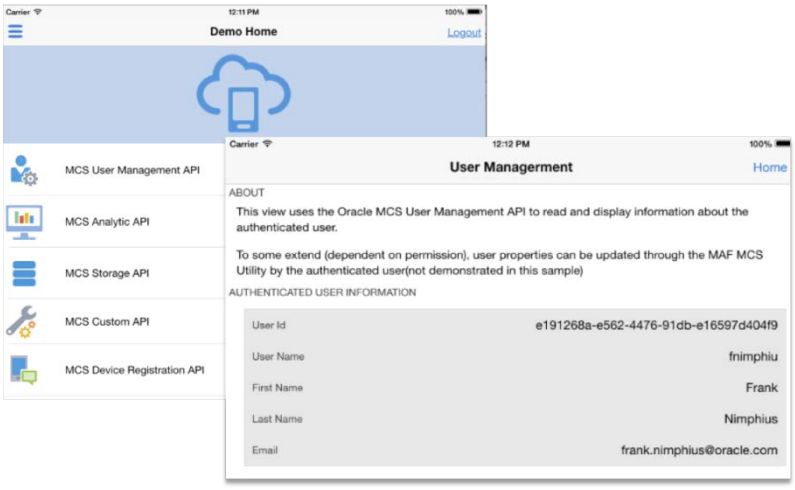


図 8: MAF MCS ユーティリティ・パブリック・サンプル内の UserInfo プロキシ

**例: ユーザー・プロファイル情報の読取り**

```
MBE mbe = … get MBE instance from MBEManager
UserInfo userInfo = mbe.getServiceProxyUserInfo();

User user = userInfo.getCurrentUserInformation();

//get information from User object properties or dump user information into
//JSON string.The latter is shown below
userInfoJson = user.toJSONString();
```

### Analytics サービス・プロキシ

`com.oracle.maf.sample.mcs.apis.analytics.Analytics` プロキシ・クラスでは、MAF アプリケーションで Oracle MCS 分析エンジンにカスタム・イベントをキューイングおよび送信できます。イベントは分析セッションのコンテキストで送信されます。これはアプリケーション・セッションと同じではありません。分析セッションは、モバイル・プロジェクト・リードまたは管理者が情報を収集する対象のモバイル・タスクまたはユース・ケースに対応します。たとえば、カスタマ・サービス技術者は携帯電話を使用してカスタマ・サービス・リクエストに応答または拒否したり、修理に関する情報でリクエストをクローズしたりします。この情報が分析イベントで収集されて MCS に送信され、モバイル・プロジェクト・リードはどのようにモバイル・アプリケーションまたはサポート・サービスを改善したらより効率的な運営が可能になるかを知ることができます。別のユース・ケースでは、イベントを使用してモバイル・アプリケーションの購入についてレポートされ、モバイル・プロジェクト・リードはアプリケーション・ユーザーによる製品の検索および最も多く選択される製品カテゴリについて理解します。

**コンストラクタ**

MAF アプリケーション開発者はコンストラクタを直接呼び出すことはできません。`MBE` インスタンスを介してプロキシ・クラスにアクセスします。

```
mbe.getServiceProxyAnalytics()
```

**メソッド**

表には、MAF アプリケーション開発者によって使用される可能性の高いパブリック・メソッドのみリストします。MAF 内部操作に関連するメソッドについては説明しません。

| メソッド名 | 説明 |
|---|---|
| addCustomEvent | カスタム・イベントがイベント・キューに追加されます。分析セッションが存在しない場合は作成されます。メソッドの引数は次のインスタンスです<br>`com.oracle.maf.sample.mcs.apis.analytics.Event`<br>例:<br>`Event customEvent =`<br>`new Event(customEventName, null, properties);`<br>`analyticsProxy.addCustomEvent(customEvent);`<br>"properties"引数は、イベントの内容を表すカスタムの名前と値のペアである HashMap<String, String>です。 |
| endSession | 分析セッションの記録が終了され、セッション・イベントが MCS 分析エンジンに送信されます。セッションにはカスタム・イベントだけでなく、MAF MCS ユーティリティによって収集されたシステムおよびコンテキスト・イベントも含まれます。 |

| | |
|---|---|
| purgeAnalyticMessagesForMobileBackend | ネットワーク障害が原因でセッションが MCS に送信されない場合、分析イベントは SQLite データベースに保存されます。このハウスキーピング・メソッドで MAF アプリケーション開発者は MBE インスタンスに対するすべての保存済セッションをクリアできます。 |
| refreshGeoLocationInformation | 分析セッション・コンテキスト・イベントの一部として現在のジオ・ロケーションが MCS に渡されます。MAF アプリケーションでこのメソッドを呼び出し、MAF MCS ユーティリティでジオ・ロケーションを再度読み取ります。 |
| startSession | 分析セッションが開始されます。カスタム・イベントの追加の前にこのメソッドが呼び出されない場合、セッションが自動的に作成されます。このメソッドの呼出し時にタイムスタンプが取得されます |

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスから `Analytics` プロキシ・クラスにアクセスします。データ・コントロールで次のメソッドが公開されます。このメソッドは一見しておくことをお薦めします: `addCustomAnalyticEvent`、`postEventsToServer`。

パブリック・サンプルでは Analytics.amx でショッピング・カートの例を示します。図 9 に、このページの実行時の表示を示します。

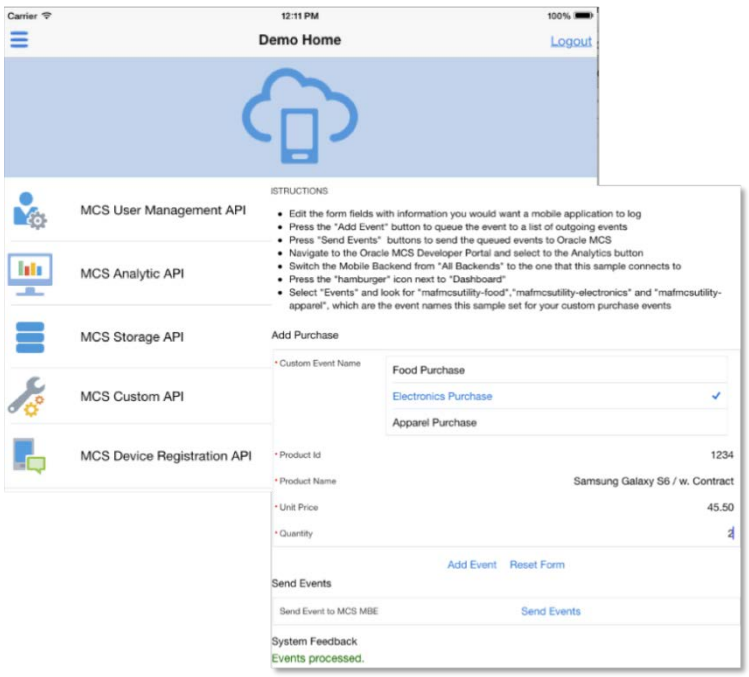


図 9: MAF MCS ユーティリティ・パブリック・サンプル内の Analytics プロキシ

**例: カスタム・イベントのキューイングおよび分析セッションの MCS への送信**

```
HashMap<String, String> properties = new HashMap<String, String>();

// Some sample data
String productId = "12345";
String productName = "iPhone";
String category = "electronics-purchase";
Double singlePrice = new Double(588);
Double quantity = new Double(2);
Double total = singlePrice * volume;
String totalString = total.toString();

// Create HashMap with values
properties.put("ProductId", productId);
properties.put("Name", producName);
properties.put("ProductType", category);
properties("Price",singlePrice.toString());
properties.put("Quantity", quantity.toString());
properties.put("Discount", "0");
properties.put("Total", totalString);

// Queue custom event locally
Analytics analyticsProxy = this.mobileBackend.getServiceProxyAnalytics();

if (isAnalyticEventOngoing == false) {

   analyticsProxy.startSession();
   isAnalyticEventOngoing = true;
}

Event customEvent = new Event(customEventName, null, properties);
analyticsProxy.addCustomEvent(customEvent);

// Repeat the steps above for as long as the analytic session (or mobile
// Use case) lasts

…

// Finally, send event to MCS - this would be in a separate method unless you
// immediately want to post events. analyticsProxy.endSession();

analyticsProxy.endSession();
```

### Storage サービス・プロキシ

`com.oracle.maf.sample.mcs.apis.storage.Storage` プロキシ・クラスでは、Oracle MCS コレクション情報を読み取ったり、その内容を更新するメソッドおよびヘルパー・オブジェクトが公開されます。ストレージ・プロキシで、アプリケーション・ユーザーはアクセス権を持つ共有または独立のコレクションにアクセスまたは書き込むことができます。

#### コンストラクタ

MAF アプリケーション開発者はコンストラクタを直接呼び出すことはできません。`MBE` インスタンスを介してプロキシ・クラスにアクセスします。

```
mbe.getServiceProxyStorage()
```

#### HTTP ETag の使用について

HTTP エンティティ・タグ(ETag)で、Web クライアントによるサーバー側のオブジェクトの問合せまたは更新に対して読取りおよび書込みの一貫性が保証されます。読取り一貫性を保証し、問合せのパフォーマンスを向上させるために、クライアントはクライアント・キャッシュ内の要求したオブジェクトのバージョンに関する情報を含む条件付きリクエストを送信します。サーバー上のオブジェクトがクライアントでキャッシュされているオブジェクトより新しい場合、サーバーは新しいバージョンのオブジェクトを含む完全なリプライで応答します。サーバー上のバージョンがキャッシュ内のものと同じ場合、これを示す省略版のレスポンスが送信されます。

同様に、書込み一貫性を保証するために、クライアントから発行される更新リクエストには更新されたオブジェクトのバージョンが示され、サーバーはこれを認識しているオブジェクトのバージョンと比較します。オブジェクトのバージョンが一致する場合、サーバーは更新を許可しますが、一致しない場合はエラー・メッセージを返し、クライアントは更新を送信する前に最新バージョンのオブジェクトを問い合せます。

ストレージ・サービス・プロキシでは `StorageCollection` オブジェクトで公開されるメソッドにおいて ETag パラメータがサポートされ、MAF アプリケーション開発者はコレクション・コンテンツ・オブジェクトに対する条件付きの更新および削除操作を実装できます。サポートされている場合、ETag パラメータは `HashMap` であるキーと値のペアとして指定されるオプションの引数です。

#### メソッド

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| getThisUserId | 認証済ユーザーID を返す便宜メソッド。これを使用しない場合、MCS ユーザーには UserManagement API を介してアクセスします。ユーザーID は特定のユーザーの共有コレクション・オブジェクトの問合せに必要で、このために使用されることがあります。 |
| querySingleCollection | 指定したコレクションのコレクション属性およびコレクション・オブジェクト情報を表す `StorageCollection` クラスのインスタンスが返されます。`StorageCollection` クラスでは保存済コレクション・オブジェクトの読取り(ダウンロード)および更新(作成、変更および削除)を行うメソッドも公開されます。<br>`com.oracle.maf.sample.mcs.apis.storage.StorageCollection` |
| querySingleCollectionForUserById | 2 つ目の引数として渡す userId で識別されるユーザーが所有する、指定した ID のコレクションが取得されます。userId はログインに使用されるユーザー名と同じではなく、MCS 内のモバイル・ユーザー・アカウントの内部ユーザーID です。ユーザー別に独立したコレクションの場合、問合せでユーザー ID を指定することで READ または READ_ALL 権限を持つ認証済ユーザーがこのユーザーのコレクション・オブジェクトにアクセスできます。 |

| | |
|---|---|
| | READ 権限ではユーザー自身の独立オブジェクトのみを読み取れることに注意してください。他のユーザーの独立オブジェクトを読み取るには、READ_ALL が必要です |
| queryStorageInformation | 特定の MCS MBE に関連付けられているコレクションのリストの問合せが行われます。コレクション情報の問合せをフェッチ・サイズと開始索引で制限し、増分フェッチを行うことができます。問合せで `com.oracle.maf.sample.mcs.apis.storage.StorageInformation` のインスタンスが返されます。<br><br>`StorageInformation getItems()` メソッドで、共有/独立状態やコレクション・コンテンツの合計サイズなどの各コレクションに関する情報を含む `StorageCollection` のリストが返されます |

**StorageCollection**

`com.oracle.maf.sample.mcs.apis.storage.StorageCollection` クラスは、MAF アプリケーション開発者がコレクション・オブジェクトのコンテンツのダウンロードおよびアップロードに使用するオブジェクトです。

次の表に、MAF 開発者が使用可能な一般的なメソッドを示しますが、これが完全なリストではありません。

| メソッド名 | 説明 |
|---|---|
| contains | 特定のオブジェクト ID がコレクションの一部であるかどうかがチェックされます |
| createObject | コンテンツ・オブジェクト(イメージなど)をモバイル・デバイスから MCS にアップロードするためのオーバーロードされたメソッド。MAF アプリケーション開発者は、アップロードするオブジェクトの byte[]配列を `StorageObject` オブジェクトまたは表示名と MIME タイプの情報とともに渡します。Oracle MCS コレクションにアップロードされたコンテンツの ID は、Oracle MCS によって自動生成されます。 |
| createOrUpdateObject | `createObject` メソッドに似ています。このメソッドでは既存のオブジェクトの更新もでき、指定したオブジェクト ID に一致するオブジェクトが存在しない場合にのみ MCS コレクションに新規オブジェクトを作成します。このメソッドでは MAF アプリケーション開発者がコンテンツの Oracle MCS での保存時に使用する一意の ID を定義することもできます。 |
| downloadByteContentForObjectId | 引数として渡された objectId またはオブジェクト URI に対応する、Oracle MCS に保存されている不透明オブジェクトをダウンロードするためのオーバーロードされたメソッド。2 つ目のメソッド引数で開発者は MCS に送信する http Accept ヘッダーの MIME タイプを指定できます。レスポンスは byte[]配列です。 |
| dump | String 表現のコレクション・オブジェクト情報(ダウンロードしたコンテンツではない)が返されます |
| isUserIsolated | コレクションがユーザー別かどうかの情報が示されます |

| | |
|---|---|
| querySingleStorageObjectById | コレクション ID で識別されたコレクションに関する情報が返されます。メソッドで `StorageInformation` のインスタンスが返されます |
| queryStorageObjectsByRange | フェッチ・サイズおよび開始索引でコレクション・オブジェクト情報の問合せが行われます。メソッドで `StorageInformation` のリストが返されます。 |
| removeCollectionObjectWithURI | StorageCollection 内の StorageObject の正規のリンク・プロパティでリモート MCS コレクション内のオブジェクトが一意に識別されます。独立したコレクションの場合、このリンクにはコレクションを所有するユーザーのユーザーID などが含まれます。<br><br>removeCollectionObjectWithURI メソッドでサーバー側のストレージ・オブジェクトが削除されます。オプションの 2 つ目の引数で HTTP ETag のキーと値のペアを含む HashMap が受け入れられ、条件付き削除が実装されます。 |
| removeCollectionObject | 次の引数に基づいてコレクション内のストレージ・オブジェクトが削除されます。<br><br>String objectId<br><br>削除するオブジェクトの ID 値<br><br>HashMap<String,String> etagHashMap<br><br>たとえば、リクエストで送信した ETag と一致しない場合にのみオブジェクトを削除する(If-Match)などのための ETag パラメータ・プロパティと値を含む HashMap。 |
| setObjectOwnerUserID | StorageCollection はサーバー側 Oracle MCS クラウド・インスタンスのコレクションを表します。<br><br>コレクションが独立コレクションの場合、すべての問合せに"user"問合せパラメータ(?user=<user id>)を付加する必要があります。MAF MCS ユーティリティでは認証済ユーザーのユーザーID が自動的に付加されます。<br><br>read_all または read_write_all 権限を持つユーザーが他のユーザーの独立コレクション領域にアクセスする場合、このメソッドでユーザー問合せパラメータ値を他の(そのコレクション・オブジェクトを更新または削除する)ユーザーのユーザーID に変更できます。<br><br>その後、`resetObjectOwnerUserIdToAuthenticatedUserId` メソッドを使用してユーザー問合せパラメータ値を認証済ユーザーのユーザーID に戻すことができます |
| updateCollectionObjectWithURI | StorageCollection 内の StorageObject の正規のリンク・プロパティでリモート MCS コレクション内のオブジェクトが一意に識別されます。<br><br>updateCollectionObjectWithURI メソッドでサーバー側のストレージ・オブジェクトが更新されます。<br><br>displayName 引数で開発者は MCS ポータルで表示される際のオブジェクトの名前を定義できます。<br><br>contentType 引数でコンテンツの MIME タイプを定義します。たとえば、アップロードするコンテンツが PNG 形式のイメージの場合は image/png です。オプションの 4 つ目の引数で HTTP ETag のキーと値のペアを含む HashMap を受け入れ、条件付き削除を実装します。 |

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスから `Storage` プロキシ・クラスにアクセスします。サンプル・アプリケーションに、MCS MBE インスタンスでの含まれるコレクションの問合せ、コレクションおよび含まれるオブジェクト情報のリスト、オブジェクトの読取り(ダウンロード)、作成(アップロード)、削除および更新(アップロード)のためのサンプル・コードがあります。

ビュー・コントローラ・プロジェクトの Storage.amx ページはすべてのストレージ操作のランディング・ページです。関連付けられている Java ロジックはビュー・コントローラ・プロジェクトの `com.oracle.maf.sample.mobile.mbeans.storage` パッケージにあるマネージド Bean に格納されています。図 10 に、MAF MCS ユーティリティ・パブリック・サンプルの実行時の表示を示します

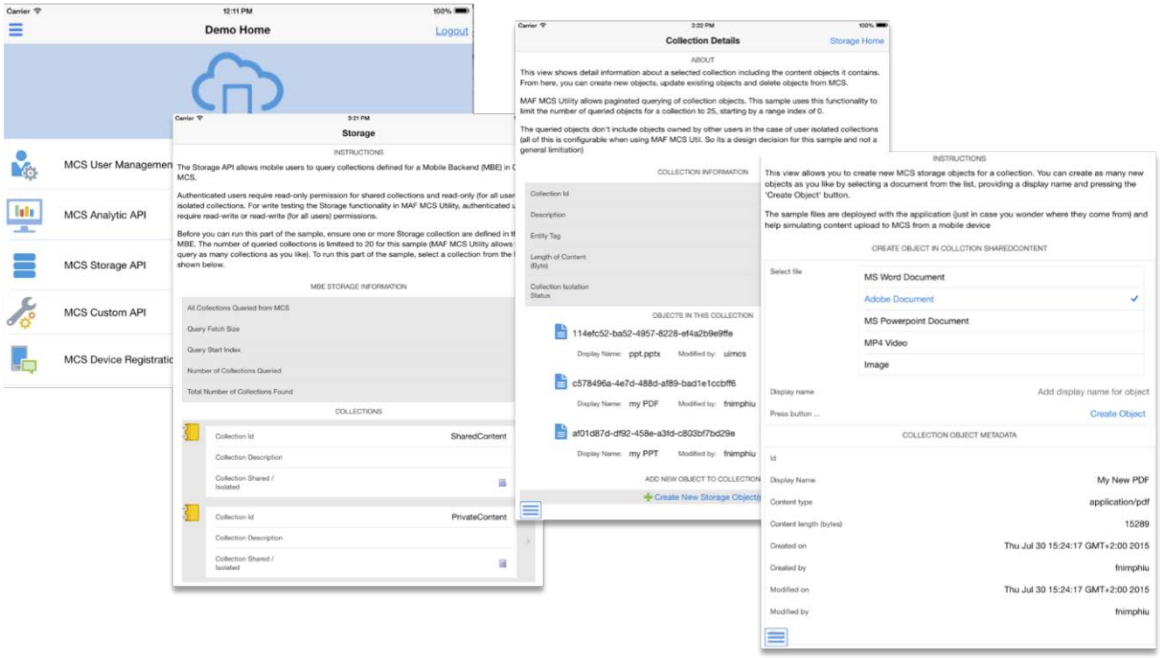


図 10: MAF MCS ユーティリティ・パブリック・サンプル内のストレージ・プロキシの表示

**例: 他のユーザーが所有するストレージ・オブジェクトの作成、更新、削除**

独立コレクションではユーザーに固有のプライベート領域内にユーザー・オブジェクトが保存されます。独立コレクション内のこのプライベート領域に対応するために、Oracle MAF MCS ユーティリティは認証済ユーザーのユーザーID を問合せパラメータとしてすべてのオブジェクト操作および問合せ操作に付加します。これは自動的に行われ、開発者はこれをコーディングする必要はありません。

ただし、認証済ユーザーが別のユーザーのプライベート領域内のコレクション・オブジェクトにアクセスする必要があるユース・ケースの場合(認証済ユーザーに必要な read_all または read_write_all 権限があるものとします)、オブジェクト操作および読取り操作に付加されるユーザーID はオブジェクト所有者の ID である必要があります。

MAF MCS ユーティリティでは、独立コレクション内の他のユーザーのオブジェクトにアクセスするための 2 つのオプションが認識されます

- アクセスするオブジェクトの正規の URI の使用。オブジェクトの正規の URI にはプライベート領域内のオブジェクトへのアクセスに必要なものがすべて含まれます。MAF MCS ユーティリティで `StorageObject` の各インスタンスは正規の URI を取得するメソッドを公開します。

  注意: このペーパーの「正規の URI を使用したコンテンツのダウンロード」というタイトルの例を参照してください

- `StorageCollection` オブジェクトはオブジェクト所有者のユーザーID を設定するプロパティを公開します。これは、開発者が問合せおよび操作の対象のコンテンツ・オブジェクトが含まれるプライベート領域を変更できるコンテキスト・スイッチです。`StorageCollection` で実行される操作はすべて所有者 ID のコンテキストで行われます。

次のコードに、オブジェクト所有者 ID を認証済ユーザーではないユーザーに設定する場合の 2 つのオプションを示します。

1. コンストラクタでのユーザーID の設定

```
StorageCollection privateCollection = storageProxy.querySingleCollectionForUserId(
                                                    "<collection name>", "<user id>")
```

2. 既存のコレクションのユーザーID の設定

```
StorageCollection privateCollection = storageProxy.querySingleCollection(
                                                                  "<collection name>")

privateCollection.setObjectOwnerUserID("<user id>");
```

いずれの場合も、次のような呼出しで所有者 ID を認証済ユーザーに戻すことができます

```
privateCollection.resetObjectOwnerUserIdToAuthenticatedUserId();
```

注意: 前述のコードではストレージ・コンテンツ・オブジェクトを所有するユーザーを非パーソナライズしません。認証済ユーザーは、独立コレクションの他のユーザーのプライベート領域で自分の権限のみを使用して操作を実行します

**例: コンテンツのダウンロード**

```
String objectId = "… object Id of content in collection …";
String mimeTye = "… e.g. image/png";
StorageCollection storageCollection = storageProxy.queryCollection("…a collection Id …");

byte[] downloadedContent =
     storageCollection.downloadByteContentForObjectId(objectId,mimeType);

//write to file.Define new file in download directory
String outFile =
             AdfmfJavaUtilities.getDirectoryPathRoot(AdfmfJavaUtilities.DownloadDirectory)
             + "/"+targetFileName;
FileOutputStream fos = new FileOutputStream(outFile);
fos.write((byte[]) downloadedContent);
fos.close();
return outFile;
```

**例: 正規の URI を使用したコンテンツのダウンロード**

コレクション内の各オブジェクトはその正規の URI で明示的にアクセスされます。たとえば、独立コレクション内のオブジェクトの場合、正規の URI にすでに"?user="パラメータとオブジェクトを作成したモバイル・ユーザーのユーザーID が含まれます。

```
String mimeTye = "… e.g. image/png";
StorageCollection storageCollection = storageProxy.queryCollection("…a collection Id …");
List<StorageObject> storageObjectList = new ArrayList<StorageObject>(); ;
//query the first 25 objects in the collection.If the collection is isolated (assuming
//the authenticated user has //read_write_all permission), then query objects of other
//users as well.Note that in such a use case it makes sense to use the canonical URI as
//you don't need to switch the user owner Id in the StorageCollection

storageObjectList = currentStorageCollection.queryStorageObjectsByRange(
                                        0, 25, this.showObjectsOwnedByOtherUsers, null);
//... lets assume the user selected an object from this list
StorageObject storageContentObject = storageObjectList.get(… index of object …)
String canonicalLink = storageContentObject.getCanonicalLink();
byte[] downloadedContent =
        currentStorageCollection.downloadByteContentForObjectUri(canonicalLink, mimeType);

//… see previous example for how to save files
```

**例: コンテンツのアップロード**

```
String displayName = "fnimphiu photo";
String contentType = "image/png";
byte[]byteContent = … photo from camera or file …
StorageCollection storageCollection = storageProxy.queryCollection("…a collection Id …");
StorageObject updateStorageObject = storageCollection.createObject(
                                                displayName, contentType, byteContent);

// Access JSON object string with new item's meta-data
String json = updateStorageObject.toJSONString();
```

**例: コンテンツの削除**

```
String objectId = "… object Id of content in collection …";
StorageCollection storageCollection = storageProxy.queryCollection("…a collection Id …");
… browse collection …
// No entity tag, passing null instead
storageCollection.removeCollectionObject(objectId, null);
…
```

### CustomAPI サービス・プロキシ

`com.oracle.maf.sample.mcs.apis.custom.CustomAPI` プロキシ・クラスは、Authorization、Oracle-Mobile-Backend-Id、Accept などの MCS で必要とされる HTTP ヘッダーを MAF から Oracle MCS に発行されるリクエストに追加する汎用 REST プロキシです。

厳密には、CustomAPI プロキシは Oracle MAF RestServiceAdapter クラスをラップし、MCSRequest および MCSResponse という 2 つの便宜クラスを追加して、MAF REST リクエスト構成およびレスポンス処理を簡単にします。

Oracle MCS プラットフォーム API にアクセスできますが、CustomAPI は主に Oracle MCS MBE で公開されたカスタム API へのアクセスに使用されます。MAF MCS カスタム API を MAF RestServiceAdapter から直接呼び出すかわりに CustomAPI プロキシを使用することによって MAF アプリケーション・ユーザーが得られる利点は、MCS 固有のヘッダーが設定されること、一貫したエラー処理(このペーパーの「ServiceProxyException クラス」を参照)および MBE 固有のロギングです。

**コンストラクタ**

MAF アプリケーション開発者はコンストラクタを直接呼び出すことはできません。MBE インスタンスを介してプロキシ・クラスにアクセスします。

```
mbe.getServiceProxyCustomApi)
```

**メソッド**

CustomAPI プロキシは 2 つのメソッドを公開します

| メソッド名 | 説明 |
|---|---|
| sendForStringResponse | String レスポンス用に同期カスタム REST リクエストを MCS サーバー・インスタンスに送信するメソッド。メソッドでは、HTTP 2XX 以外のレスポンス・コードはすべてアプリケーション・エラーとして処理されます。<br>`sendForStringResponse(MCSRequest request)`<br>MCSRequest オブジェクトには、MAF REST 接続名、リクエスト URI、ヘッダー・パラメータなどの呼出しを発行するために必要なすべての構成が保持されます。メソッドで、レスポンス・ペイロードおよびその他の情報を保持する MCSResponse が返されます。MAF アプリケーションで MCSResponse オブジェクトにおいて getMessage() を呼び出し、レスポンス・ペイロードを取得しますが、これは String にキャストできるオブジェクトです。 |
| sendReceiveBytes | byte[]レスポンス用に同期カスタム REST リクエストを MCS サーバー・インスタンスに送信するメソッド。メソッドでは、HTTP 2XX 以外のレスポンス・コードはすべてアプリケーション・エラーとして処理されます。<br>`sendReceiveBytes(MCSRequest request)`<br>MCSRequest オブジェクトには、MAF REST 接続名、リクエスト URI、ヘッダー・パラメータなどの呼出しを発行するために必要なすべての構成が保持されます。メソッドで、レスポンス・ペイロードおよびその他の情報を保持する MCSResponse が返されます。MAF アプリケーションで MCSResponse オブジェクトにおいて getMessage() を呼び出し、レスポンス・ペイロードを取得しますが、これは byte[]にキャストできるオブジェクトです。 |

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある MobileBackendDC データ・コントロール・クラスから CustomAPI プロキシ・クラスにアクセスします。invokeCustomMcsAPI メソッドは、com.oracle.maf.sample.mobile.mbeans.custom パッケージにある CustomAPIBacking バッキング Bean のビュー・レイヤーから起動されます。図 11 に、パブリック・サンプルの CustomAPI.amx ページの実行時の表示を示します。

サンプルでは、ユーザーは HTTP メソッド(Get、Post、Put、Delete)を選択し、MCS で起動するカスタム API の URI を指定します。HTTP メソッドが Put または Post の場合、ペイロードが必須である可能性があります。サンプル・アプリケーションでは、複雑にならないように文字列のペイロードおよびレスポンスのみサポートされます。ただし、カスタム API がその目的で設計されている場合、基礎となる MAF MCS ユーティリティ・プロキシでバイナリ・コンテンツのアップロードまたはダウンロードもできます。

http ヘッダー・パラメータ・フィールド(1 つは名前でもう 1 つは値)でユーザーはリクエストとともに送信されるヘッダー・パラメータおよび値を追加できます。そうすると、カスタム API の起動から返されたレスポンスが、画面の下部に未加工の形式(JSON またはテキスト)で表示されます。

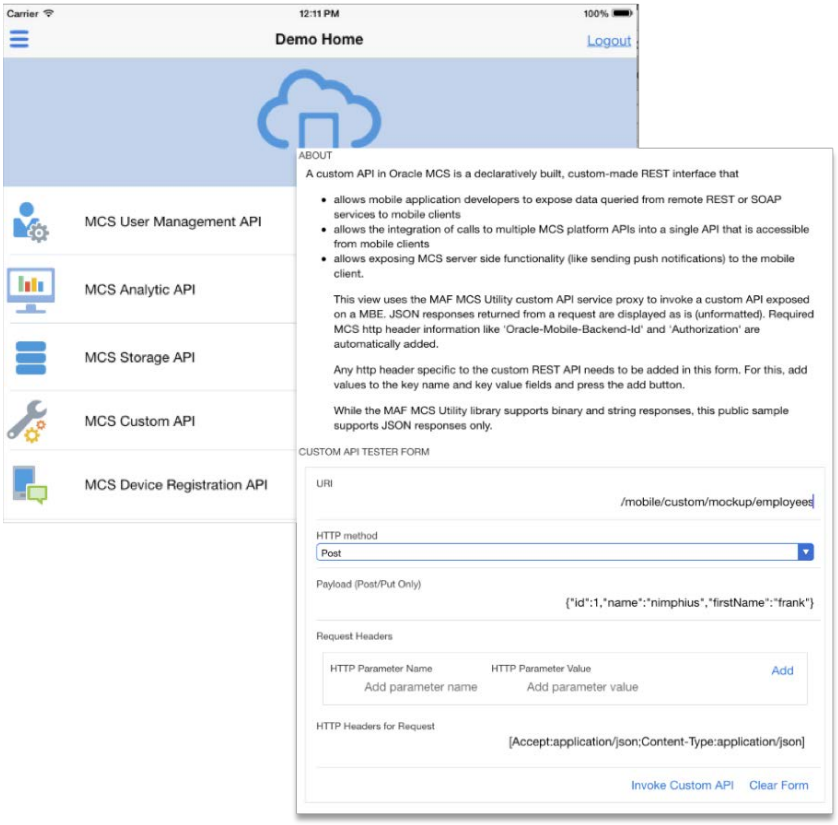


図 11: MAF MCS ユーティリティ・パブリック・サンプル内の CustomAPI の表示

**例: カスタム API の起動**

```
//gGet access to the service proxy for custom API calls
CustomAPI customApi = mobileBackend.getServiceProxyCustomApi();

// MAF MCS Utility provides to configuration objects, MCSRequest and
// MCSResponse, that guide you through setting up the MCS request and
//to read the MCS response.First, a MCSRequest object is created
```

```
MCSRequest request = new MCSRequest(mobileBackend.getMbeConfiguration());

// The MSF REST connection as defined in Application Resources --> Connections --> REST.
// In this example the REST connection name is "mcsrestconn"
request.setConnectionName("mcsrestconn");

// Save the custom MCS API URI
request.setRequestURI("/moile/custom/mockup/employees");
// Set the request method
request.setHttpMethod(MCSRequest.HttpMethod.POST);

// Set payload - a JSON string in this example
request.setPayload("{¥"id¥":¥"1¥"¥"name¥":¥"nimphius¥",¥"firstName¥":¥"frank¥"}");
request.setRetryLimit(0);

// Define the headers that need to be sent with the custom API request
HashMap<String, String> headers = new HashMap<String, String>();
headers.put("Content-Type","application/json");
request.setHttpHeaders(headers);

// This call returns a String response.For binary responses (binary[]) you use
// CustomApi.sendReceiveBytes(request) instead.

MCSResponse response = customApi.sendForStringResponse(request);
String jsonResponse = (String) response.getMessage();
```

### Notifications サービス・プロキシ

Oracle MCS の通知プラットフォーム API ではクライアントおよびサーバー側のサービスが提供されます。サーバー側のサービスでは、Node.js に記述されたカスタム API コードが Google Cloud Messaging (GCM)サービスまたは Apple Push Notification Service (APNS)にプッシュ・メッセージをキューイングし、送信します。

クライアント・サービスではモバイル・クライアントがモバイル・デバイスを Oracle MCS に登録します。その後メッセージが発せられた場合、Oracle MCS がプッシュ・メッセージを Google GMC または Apple APNS プッシュ・プロバイダにキューイングし、登録されたデバイスに配信されます。

名前が示すとおり、クライアント・サービスにはモバイル・クライアントから直接アクセスでき、この用途に MAF MCS ユーティリティには `com.oracle.maf.sample.mcs.apis.notifications.Notifications` プロキシ・クラスが用意されています。

MAF MCS ユーティリティの `MBE` で公開されている他のプロキシ・クラスとは異なり、`Notifications` プロキシはそのままでは機能せず、MAF アプリケーションでアプリケーションの起動時に GMC または APNS からのデバイス・トークンを要求するために必要とされる Google 送信者 ID、Apple p12 証明書およびバンドル ID を MAF アプリケーション開発者が取得する必要があります。

注意: MAF MCS ユーティリティ・パブリック・サンプル・アプリケーションには、アプリケーション・コントローラ・プロジェクトで簡単に設定できるデバイス・トークンを取得するコードが含まれています。`PushEventListener` クラスおよび `LifeCycleListenerImpl` クラスを参照してください。送信者 ID および p12 証明書の取得方法の詳細は、Oracle MAF の製品ドキュメントを参照してください。

`Notifications` プロキシ・サービスでは、モバイル・アプリケーション・クライアントはモバイル・デバイスを登録および登録解除し、MCS メッセージ・ペイロードが Android および iOS 用に正しく書式設定され、適切に構成されるようにします。

#### コンストラクタ

MAF アプリケーション開発者はコンストラクタを直接呼び出すことはできません。`MBE` インスタンスを介してプロキシ・クラスにアクセスします。

```
mbe.getServiceProxyNotifications()
```

**メソッド**

| メソッド名 | 説明 |
|---|---|
| deregisterDeviceFromMCS | MCS からのデバイスの登録解除が試行されます。このデバイスに対するメッセージは Google または Apple プッシュ・サービスに送信されなくなります。<br><br>構成情報は `MBE` の `MBEConfiguation` オブジェクトから読み取られるため、MAF アプリケーション開発者はこのメソッドを起動する際引数を渡す必要はありません。アプリケーションがアクティブでない間はメッセージが送信されないようにする場合、アプリケーションを閉じる際にこのメソッドを呼び出す必要があります。 |
| isDeviceRegisteredForPush | アプリケーションが MCS に登録されているかどうかを MAF アプリケーション開発者が確認するための便宜メソッド。 |
| registerDeviceToMCS | MCS からのデバイスの登録が試行されます。このデバイスに対するメッセージは MCS を介して Google または Apple プッシュ・サービスに送信されます。<br><br>構成情報は `MBE` の `MBEConfiguation` オブジェクトから読み取られるため、MAF アプリケーション開発者はこのメソッドを起動する際引数を渡す必要はありません。<br><br>このメソッドはアプリケーションの起動時に必ず呼び出します。MCS ではデバイス登録が永久に保持されるわけではなく、使用されないトークンはクリーンアップされることに注意してください。<br><br>メソッドは MCS デバイス登録情報を含む JSON 文字列を返します。この情報はメッセージの取得およびプッシュには不要で、無視できます。 |

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスから `Notifications` プロキシ・クラスにアクセスします。さらに、サンプル・アプリケーションには、Google または Apple からデバイス・トークンを受信するよう構成されているライフサイクルおよびプッシュ・イベント・リスナーがあります。

注意: Apple および Google のライセンスの制限から、MAF MCS ユーティリティ・サンプル・アプリケーションには MAF 開発者がすぐに通知をテストできるような Google 送信者 ID または Apple バンドル ID および p12 証明書は付属していません。自分で送信者 ID および p12 証明を取得し、(Apple デプロイメントの場合)デプロイメント時にアプリケーションに署名してデプロイメントのバンドル ID を変更する必要があります。

図 12 に、ビュー・コントローラ・プロジェクトにある DeviceRegistration.amx ページの実行時の表示を示します。プッシュが設定され、MAF サンプル・アプリケーションに対して有効な場合、アプリケーションが Google または Apple から受信したデバイス・トークン、デバイス登録の成功を表す JSON オブジェクト、着信プッシュ・メッセージの未加工のペイロードなどの情報がこのページに表示されます。図 12 に、(当座無効な)デバイス・トークン、MCS デバイス登録通知および MCS から送信されたプッシュ・メッセージを含むページを示します。

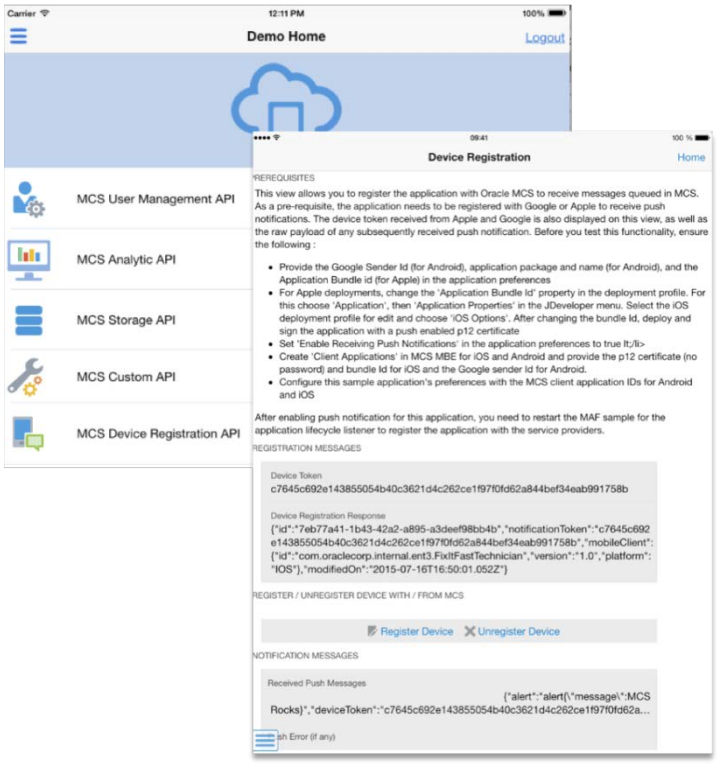


図 12: MAF MCS ユーティリティ・パブリック・サンプル内の CustomAPI の表示

**例: デバイスの登録**

```
// Get the _token retrieved from Google or Apple, assuming the
// MAF lifecycle listener stored this information in memory: //
("#{appliationScope.devicetoken}");
String _token =
    (String) AdfmfJavaUtilities.getELValue("#{appliationScope.devicetoken}");

// If we have a _token, set it to the MBE configuration for the Notifications
// service proxy to access when registering and de-registering the application
// to MCS

if(_token != null && !_token.isEmpty()){
   mobileBackend.getMbeConfiguration().setDeviceToken(_token);
} else {

     // No _token -- cancel request
     return;

}

// Does MAF run on iOS or Android ?
DeviceManager deviceManager = DeviceManagerFactory.getDeviceManager();
```

```
String _toUppercaseManufacturerOS = deviceManager.getOs().toUpperCase();
boolean isIOS = _toUppercaseManufacturerOS.equals("IOS") ? true : false;

if (isIOS) {
  //assume Apple bundle Id is available in "#{applicationScope.bundleId}"
   String appleBundleId = (String)
   AdfmfJavaUtilities.getELValue("#{applicationScope.bundleId}");
   mobileBackend.getMbeConfiguration().setAppleBundleId(appleBundleId);
// else Android
} else {
  // Assume Google package name is available in "#{applicationScope.packageName}"
  String packageName =
     (String) AdfmfJavaUtilities.getELValue("#{applicationScope.packageName}");
   mobileBackend.getMbeConfiguration().setGooglePackageName(packageName);
}

Notifications notification = mobileBackend.getServiceProxyNotifications();
String registrationInfo = notification.registerDeviceToMCS();

…
```

## ServiceProxyException クラス

MAF MCS ユーティリティでは 2 つのタイプの例外がスローされ、MAF アプリケーション開発者がエラー処理します。

- `oracle.adfmf.framework.exception.IllegalArgumentException`

  必要なメソッドまたはコンストラクタ引数がない(null に設定されているなど)場合、または引数の長さがゼロの場合にスローされます。

- `com.oracle.maf.sample.mcs.shared.exceptions.ServiceProxyException`。

  アプリケーション・エラー、REST トランスポート・レイヤー例外の両方の場合にスローされます。アプリケーション・エラーは Oracle MCS の起動の失敗です。たとえば、ユーザーが READ または READ_ALL 権限のないコレクション内のオブジェクトにアクセスを試みた場合などです。

  REST トランスポート・レイヤー例外は、ネットワークやサーバーのアクセス失敗などのランタイム例外で、MAF MCS ユーティリティは処理方法を知りません。

`ServiceProxyException` には MAF アプリケーション開発者によるメッセージ・タイトルや説明などの MCS エラー・メッセージの詳細へのアクセスを可能にする便宜メソッドが用意されており、MAF MCS ユーティリティのすべてのメソッド内でスローされます。

**メソッド**

| メソッド名 | 説明 |
|---|---|
| getErrorResponseHeaders | エラー・レスポンスに HTTP ヘッダー情報が含まれている場合、これが HashMap で返されます |
| getExceptionClassName | `ServiceProxyException` オブジェクトでラップされた例外クラスが返されます |
| getHttpResponseCode | アプリケーション・エラーに対する HTTP エラー・レスポンスが返されます。<br>たとえば、Oracle MCS はプラットフォーム API の起動の失敗に対し 4xx の HTTP ステータス・コードを返します。<br>各 MCS API がエラーの場合に返されるステータス・コードのリストは、MAF MCS ユーティリティ・ソース・コードまたは Oracle MCS API テスターで取得できます。 |
| getMessage | エラー・メッセージが返されます。JSON ペイロード文字列(エラーが MCS API の起動の失敗の場合)またはテキスト・メッセージ(エラーがランタイムまたはネットワーク・エラーの場合)です |
| isApplicationError | エラーが Oracle MCS アプリケーション・エラーであることが示されます。この場合、次の呼出しを行えます<br>`String errorMessage = "";`<br>`if (e.isApplicationError()) {`<br>`  // If this is a well formatted Oracle Mobile error,`<br>`  // we can display a user friendly error message`<br>`  try {`<br>`  // OracleMobileError is a MAF NCS Utility class`<br>`  OracleMobileError mobileError =`<br>`  OracleMobileErrorHelper.getMobileErrorObject(`<br>`  e.getMessage());`<br>`  errorMessage = mobileError.getTitle();`<br>`  } catch (JSONException f) {`<br>`  //not a JSN formatted MCS error message`<br>`  errorMessage = e.getMessage();`<br>`  }`<br>`…`<br>`}` |
| isException | エラーが MCS に関係なく、ランタイムまたはネットワークに関係することが示されます |

**MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所**

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスで `ServiceProxyException` クラスを使用します。

データ・コントロール・クラス内のすべての MAF MCS ユーティリティ・サービス・プロキシの起動は、起動が失敗した場合にエラーがアプリケーション・エラーかランタイム例外かを確認する try-catch ブロックで囲まれます。アプリケーション・エラーの場合、サンプルは MAF MCS ユーティリティ `OracleMobileErrorHelper` ユーティリティ・クラスを使用して MCS エラー・メッセージの詳細を Java オブジェクト `OracleMobileError` クラスに読み込み、エラー・タイトルをユーザー・インタフェースに表示します。

**例: エラー処理**

`MobileBackendDC` から抜粋したサンプル・コード:

```
if ((collectionId != null) && (!collectionId.isEmpty())) {
  //use of the typed interface to access a service proxy
  Storage storageProxy = this.mobileBackend.getServiceProxyStorage();
  try {
     currentStorageCollection = storageProxy.queryCollection(collectionId);
     // Indicates successful change of storagee selection
     currentStorageCollectionChangedFlag = true;
     return currentStorageCollection;
  } catch (ServiceProxyException e) {
       if (e.isApplicationError()) {
          try {
             OracleMobileError mobileError =
                      OracleMobileErrorHelper.getMobileErrorObject(e.getMessage());
                      // Print short description of error
                      logDcErrorForUserInterfaceDisplay(mobileError.getTitle());
          } catch (JSONException f) {
                logDcErrorForUserInterfaceDisplay(e.getMessage());
          }
       } else {
            logDcErrorForUserInterfaceDisplay(e.getMessage());
       }
  }

} else {
       logDcErrorForUserInterfaceDisplay("collectionId value cannot be null or empty");
}

…
```

## 非同期 API 起動

設計では、すべての MAF MCS ユーティリティ・メソッドは、常に非同期に実行される分析セッションの Oracle MCS への送信以外同期的に起動されます。

Java ライブラリを設計する場合、開発者には非同期メソッド起動の処理について 2 つの選択肢があります。

- 非同期に実行するメソッドをライブラリで公開し、メソッドに渡されるオブジェクトでコールバック・メソッドを起動するコールバック・メカニズムを用意します。このオプションでは設計がクリーンになり、非同期タスクが完了すると制御が呼出し元のアプリケーションに戻ります。デメリットはコールバック・ハンドラを作成することがプログラミング・オーバーヘッドであることで、メソッド起動の成功およびエラーの場合にコールバックが必要な場合に顕著です。この方法を使用する場合、アプリケーション開発者はメソッドが必ず適切な順序で実行されるようにする必要もあり、注意を要します。コールバックは JavaScript 開発でよく使用されることに注意してください。
- 同期的に実行するメソッドをライブラリ内に含め、アプリケーション開発者に API を非同期メソッド起動内でラップする方法を説明します。このオプションの利点は、アプリケーション開発者がメソッドの実行順序を制御できることです。また、UI コントロール(通常はメイン・スレッドで実行される)を管理し、メソッドの起動に呼応してリフレッシュできます。

MAF MCS ユーティリティでは 2 つ目の選択肢(同期メソッドを非同期呼出し内にラップするオプション)が選択されています。この項では、MAF アプリケーション開発者が MAF MCS ユーティリティ・サービス・プロキシ・メソッドを非同期に起動する方法について説明します。

MAF は Java フレームワークで、メイン・アプリケーション・スレッド以外に追加の実行スレッドを起動できます。このために、MAF アプリケーション開発者は MCS MCS ユーティリティ・サービス・プロキシ・メソッドの起動を `Runnable` を実装するオブジェクト内にラップします。`Runnable` オブジェクトは `ExecutorService` オブジェクトから実行されます。図 13 に、この起動のブロック図を示します。

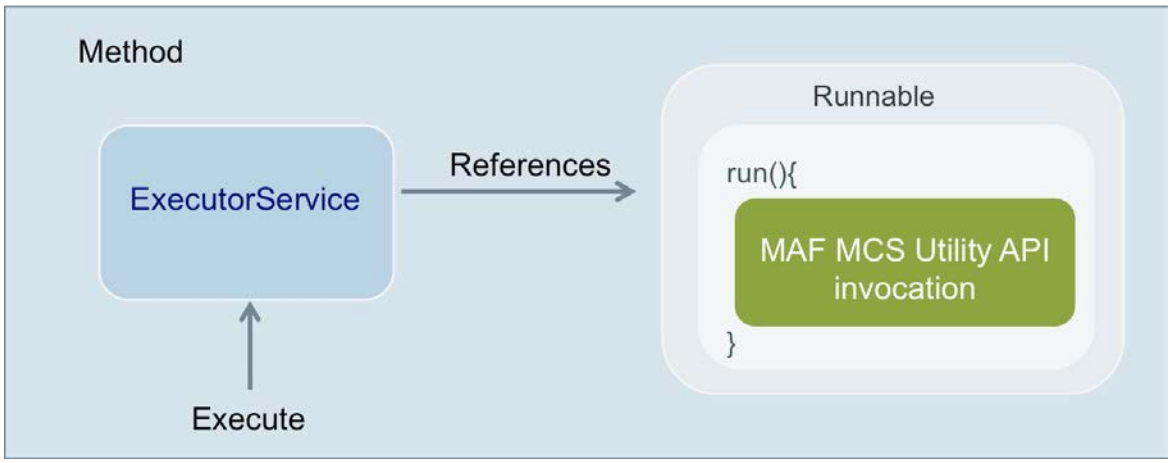


図 13: MAF MCS ユーティリティ・アーキテクチャ

例を見てみましょう。次のコードはすべて同一メソッド(たとえば、データ・コントロールまたはマネージド Bean で公開されているもの)に追加できることに注意してください。

```
Runnable mcsutilityJob = new Runnable(){
     public void run(){

       //this is where MAF developer put the MAF MCS Utility method
       //invocation …
        … do some more work …
       //refresh UI controls that are impacted by the data changes
```

```
        //triggered by the method invocation.For this in MAF you call
        //AdfmfJavaUtilities.flushDataChangeEvent()

      }

}

… optionally do some other work here …

//execute runnable
ExecutorService executor = Executors.newFixedThreadPool(2);
executor.execute(mcsutilityJob);
executor.shutdown();
```

アプリケーションは `ExecutorService` の実行後にあるコード行で続くことに注意してください。UI の変更が MAF UI に必ず表示されるようにするためには、`AdfmfJavaUtilities.flushDataChangeEvent()` を呼び出し、メイン・スレッドでそのコンテンツをリフレッシュする必要があります。さらに、メソッドがメイン・スレッドでコレクションを更新する場合、MAF でプロバイダ変更イベントを発生させる必要があります。

注意: MAF におけるプロバイダ変更イベントについては、Oracle Mobile Application Framework YouTube チャンネルのビデオ (https://www.youtube.com/watch?v=ZJePFhfVqMU)で詳しく説明されています

### MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所

MAF MCS ユーティリティ・パブリック・サンプルでは、アプリケーション・コントロール・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスでデバイス登録(`registerDeviceWithMCS` メソッドを参照)およびカスタム API 起動(`invokeCustomMcsAPI` メソッドを参照)に対する非同期メソッド起動を紹介しています。

注意: MAF MCS ユーティリティ・ソース・コードを見ると、同じ非同期メソッド起動が `Analytics` プロキシで使用されていることがわかります。分析セッションの MCS へのポストで呼出し元アプリケーションの値は変更されません。そのため、MAF MCS ユーティリティで直接この呼出しを非同期として実装しても、UI が適切にリフレッシュされないなどの望ましくない結果をもたらしません。

### 例: 非同期 MAF MCS ユーティリティ・メソッド起動

```
Runnable mcsJob = new Runnable(){
  public void run(){
    try {
      Notifications notification = mobileBackend.getServiceProxyNotifications();
      notification.deregisterDeviceFromMCS();
      logDcInfoForUserInterfaceDisplay("Application has been deregistered from MCS");
      AdfmfJavaUtilities.setELValue(PushConstants.MCS_REGISTRATION_STRING, "");
   } catch (ServiceProxyException e) {
       if (e.isApplicationError()) {
           …
        }

          else {?

            logDcErrorForUserInterfaceDisplay(e.getMessage());
```

```
        }

    }
   //ensure main thread is synchronized with result

   AdfmfJavaUtilities.flushDataChangeEvent();
  }
 };

//Invoke asynchronous invocation

   ExecutorService executor = Executors.newFixedThreadPool(2);
   executor.execute(mcsJob);
   executor.shutdown();
}
```

## ユーザー認証

Oracle MCS パブリック・クラウド、バージョン 1.0 の認証は Basic 認証を介してのみ行われ、MAF 機能レベルで構成したり、MAF MCS ユーティリティ認可プロバイダを使用して手動で実行できます。

ユーザー認証が MAF MCS ユーティリティを介して明示的に実行されない場合、ユーティリティは Oracle MCS へのリクエストごとに Oracle MAF によって HTTP Authorization ヘッダーが設定されるものと見なします。

### MAF を介した認証

MAF MCS ユーティリティ呼出しの認証に MAF セキュリティ・フレームワークを使用するには、MAF MCS ユーティリティ MBE インスタンスと MAF の機能との間に 1 対 1 マッピングを実装する必要があります。MAF MCS ユーティリティで MAF の機能と MBE インスタンスとの 1 対 1 マッピングを作成するには、mafmcsutility.jar ファイルをビュー・コントローラ・プロジェクト内に構成し、ビュー・コントローラ・プロジェクトで定義されたデータ・コントロールまたはマネージド Bean から `MBEManager` にアクセスします。

これを行う場合、`MBEManager` インスタンスおよび `MBE` インスタンスが作成され、MAF 機能に関連付けられている子クラス・ローダーで管理されていることを確認します。

次に、MAF 機能でセキュリティを有効にし(maf-features.xml ダイアログ)、maf-application.xml エディタを使用して認証用に Oracle MCS MBE を指すログイン・サーバーを作成します。

また、maf-application.xml ダイアログで、MAF ログイン・サーバー構成を Oracle MCS MBE に対して認証される MAF 機能に関連付ける必要もあります。

図 14 に、maf-application.xml ダイアログおよびログイン・サーバー接続の作成および MAF MCS MBE に対して認証される MAF 機能との関連付けに使用するセキュリティ・パネルを示します


図 14: セキュリティ・パネルが表示された maf-application.xml ダイアログ

MAF 宣言セキュリティを認証に使用することの利点は、宣言的に構成されること、およびオフライン認証と Basic 認証の remember-me 機能が使用できることです。

注意: Oracle MCS MBE に対する MAF 機能の認証を可能にするには、MAF 2.1.3 以上を使用する必要があります

### MAF MCS ユーティリティを介した認証

MAF MCS ユーティリティでは、アプリケーション開発者が MAF MCS ユーティリティ認可プロバイダ・クラスを介して Oracle MCS MBE へのリクエストを認可できます。MAF MCS ユーティリティでの手動の MCS MBE 認可は MAF 2.1.2 以上で可能で、Oracle MCS へのリクエストごとに MAF MCS ユーティリティによって HTTP Authorization ヘッダーが設定されます。

注意: MAF MCS ユーティリティの認可は、MAF 機能にマップされている MBE インスタンスおよびアプリケーション・コントローラ・プロジェクトで MAF データ・コントロールを介してグローバル・アプリケーションの使用に対して作成された MBE インスタンスに対して機能します。

```
Authorization authorization = mobileBackend.getAuthorizationProvider();

Boolean authenticationSuccess = Boolean.FALSE;

try {

    // ServiceProxyException is thrown if authentication fails
    authorization.authenticate(username, password);
    authenticationSuccess = Boolean.TRUE;
} catch (Exception e) {
    authenticationSuccess = Boolean.FALSE;
      …
}
```

MAF MCS ユーティリティを介したコード中心の手動認証を使用する利点は、アプリケーション開発者が匿名認証とユーザー名ベースの認証とを切り替えることができることです。これは、MAF での宣言認証より MAF MCS ベースの認証を使用する方が簡単に行えます。

さらに、プログラムによる認証では認証されたユーザーのユーザーID が `MBEConfiguration` オブジェクトに暗黙的に設定されます。この設定はストレージ・サービス・プロキシ内の独立コレクションにアクセスするために必要です。MAF 宣言セキュリティ機能を介して認証する場合、`UserInfo` サービス・プロキシに直接アクセスし、認証されたユーザーのユーザーID(ユーザー名ではない)を取得して `setAuthenticatedUserMCSUserId()` の呼出しで `MBEConfiguration` オブジェクトに手動で設定する必要があります。

#### MAF MCS ユーティリティ・パブリック・サンプル・アプリケーション内の該当箇所

これより前の図 5 に、アプリケーション・ユーザーがユーザー名とパスワードで、またはパブリック API アクセス専用の匿名ユーザーとして認証を受けることのできる MAF MCS ユーティリティ・パブリック・サンプルの手動ログイン画面を示しています。

サンプル・ログイン画面(ビュー・コントローラ・プロジェクトの Authentication.amx)では、`authenticateUser` および `anonymousLogin` メソッドにアプリケーション・コントローラ・プロジェクトにある `MobileBackendDC` データ・コントロール・クラスでアクセスします。

**例: 匿名ログイン**

```
Boolean authenticationSuccess = Boolean.FALSE;
if (mobileBackend != null) {
  Authorization authorization = mobileBackend.getAuthorizationProvider();
  try {
      authorization.authenticateAsAnonymous();
      authenticationSuccess = Boolean.TRUE;
  } catch (ServiceProxyException e) {
      authenticationSuccess = Boolean.FALSE;
      if (e.isApplicationError()) {
        try {
           OracleMobileError mobileError =
                OracleMobileErrorHelper.getMobileErrorObject(e.getMessage());
          …
         } catch (JSONException f) {
  …
         }
      }else {

   …
}
```

## デバッグ

Oracle MAF パブリック・サンプルには mafmcsutility.jar およびユーティリティの使用方法を示すパブリック・サンプルだけでなく、MAF MCS ユーティリティ・ソース・コードも JDeveloper ワークスペースに含まれています。MAF MCS ユーティリティ・ソース・コードにはコメントが付いており、開発者が特定のサービス・プロキシ・メソッドの機能および用途を理解するのに役立ちます。

開発およびデバッグを簡単にするために、MAF MCS ユーティリティ・ソース・コード・プロジェクトをアプリケーション・ワークスペースに追加できます。図 15 に、JDeveloper メニューで「ファイル」→「開く」を選択すると開くダイアログを示します。

mafmcsutility.jpr プロジェクト・ファイルを選択して開きます。変更が MAF MCS ユーティリティ・ワークスペースに適用されることを知らせるダイアログで「OK」をクリックします。

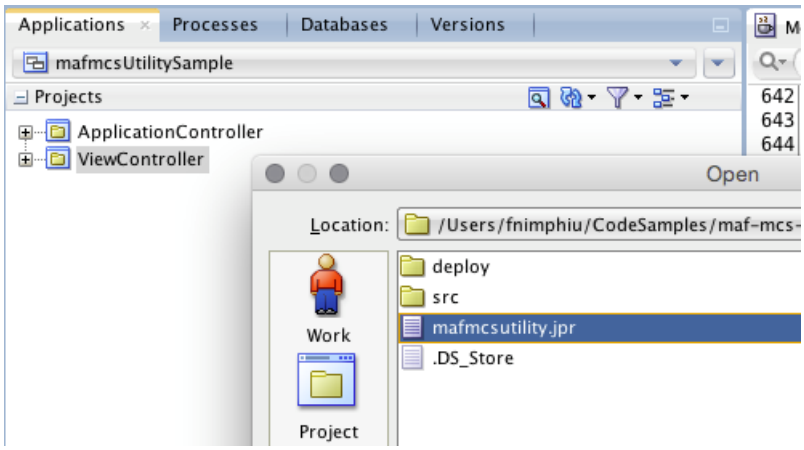


図 15: カスタム MAF アプリケーション・ワークスペースへの MAF MCS ユーティリティ・プロジェクトの追加

MAF MCS ユーティリティ・プロジェクトを MAF アプリケーション・コントローラまたはビュー・コントローラ・プロジェクトの依存性として構成します。これによって特定のユーティリティ・クラスに簡単に移動でき、MAF MCS ユーティリティ・コードにブレークポイントを設定することもできます。

アプリケーション開発の最後に、MAF MCS ユーティリティ・プロジェクトの依存性を mafmcsutility.jar ファイルへのライブラリ参照に置き換えます。図 16 に、MAF MCS ユーティリティ・パブリック・サンプル・ワークスペースに追加されている "mafmcsutility"プロジェクトを示します

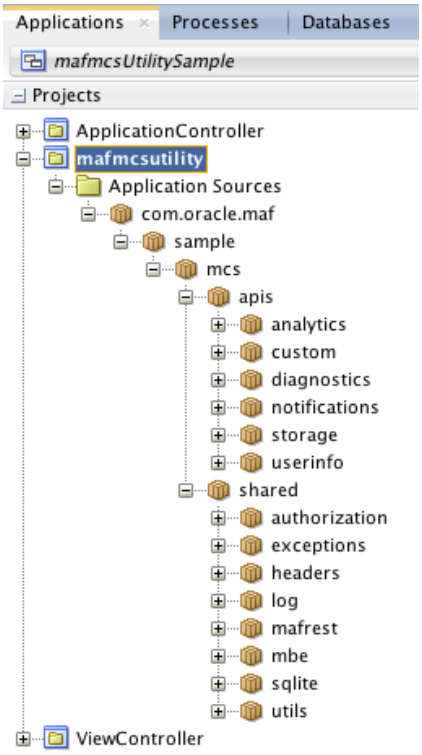


図 16: パブリック・サンプル・ワークスペースに追加された MAF MCS ユーティリティ・プロジェクト

## まとめ

このホワイトペーパーでは、MAF アプリケーション開発者に Oracle MAF アプリケーションでの Oracle Mobile Cloud Services (MCS)へのアクセスを簡単にするパブリック・サンプルおよびライブラリである MAF MCS ユーティリティの概要を示しました。Oracle MAF MCS ユーティティは Oracle MCS 用 SDK ではありませんが、同様の機能およびインフラストラクチャを提供します。

このホワイトペーパー内の API ドキュメントとコード例、Oracle MAF MCS ユーティリティ・パブリック・サンプル・アプリケーションおよび MAF MCS ユーティリティ・ソース・コードによって、MAF アプリケーション開発者は Oracle MCS パブリック・クラウドの呼出しをカスタム・モバイル・アプリケーションに簡単に統合できます。

**Oracle Corporation, World Headquarters**
500 Oracle Parkway
Redwood Shores, CA 94065, USA

**海外からのお問合せ窓口**
電話: +1.650.506.7000
ファックス: +1.650.506.7000

CONNECT WITH US


blogs.oracle.com/oracle


facebook.com/oracle


twitter.com/oracle


oracle.com

Hardware and Software, Engineered to Work Together

Copyright (c) 2015, Oracle and/or its affiliates.All rights reserved.本文書は情報提供のみを目的として提供されており、ここに記載される内容は予告なく変更されることがあります。本文書は一切間違いがないことを保証するものではなく、口頭での明示か法律による暗示かを問わず、特定の目的に対する商品性または適合性についての暗黙的な保証を含め、いかなる保証も条件も提供いたしかねます。本文書に関するいかなる法的責任も負わず、また本文書によって直接的または間接的に契約上の義務は確立されないものとします。本文書は書面による許可をあらかじめ得ることなく、いかなる目的のためにも、また電子的、機械的を問わずいかなる形式や手段によっても複製または送出することはできません。

Oracle および Java はオラクルおよびその関連会社の登録商標です。その他の社名、商品名等は各社の商標または登録商標である場合があります。

Intel、Intel Xeon は、Intel Corporation の商標または登録商標です。すべての SPARC の商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMD ロゴ、AMD Opteron ロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIX は、The Open Group の登録商標です。0815

Oracle Mobile Application Framework - MAF MCS ユーティリティ開発者ガイド
2015 年 8 月
原本著者: Frank Nimphius
原本協力者: Chris Muir、Mireille Duroussaud、Lynn Munsiger、Gary Williams、Jeff Gallus

Oracle is committed to developing practices and products that help protect the environment