IBM Unica Marketing Operations
バージョン 8 リリース 6
2012 年 5 月 25 日

統合モジュール

注

本書および本書で紹介する製品をご使用になる前に、 33 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM Unica Marketing Operations バージョン 8 リリース 6 モディフィケーション 0 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： IBM Unica Marketing Operations
Version 8 Release 6
May 25, 2012
Integration Module

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2012.6

© Copyright IBM Corporation 2002, 2012.

目次

© Copyright IBM Corp. 2002, 2012

第 1 章 Marketing Operations 統合サービスの説明

Marketing Operations 統合サービスは次のものの複合体です。

- **Marketing Operations 統合 Web サービス**

  統合サービスは、IBM® Unica Marketing Operations顧客、パートナー、および IBM Professional Services が Marketing Operations を、自分の環境内で実行されている他のアプリケーションと統合する手段を提供します。

- **Marketing Operations プロシージャーおよび API**

  カスタム・プロシージャーを Marketing Operations 内部で定義して Marketing Operations ビジネス・ロジックを任意の方法で拡張できます。いったん定義されると、これらのプロシージャーを、他のアプリケーションからの統合サービス Web サービス呼び出しのターゲットにすることが可能です。また、他のアプリケーションにメッセージを送信するようにプロシージャーを定義することもできます。

- **Marketing Operations トリガー**

  トリガーを Marketing Operations 内のイベントやプロシージャーと関連付けることができます。そのようなイベントの 1 つが発生した際、関連トリガーを実行します。

Marketing Operations 統合サービスの要件の説明

Marketing Operations 統合サービスは次のことを行う必要があります。

- 疎結合システム統合。
- カスタマー・アプリケーションが Web サービス呼び出しを介して Marketing Operations に作用するメカニズムの提供。
- Marketing Operations のあるイベントをカスタマー・アプリケーションに通知するメカニズムの提供。
- 容易に理解し使用できる単純なプログラミング・モデルの提供。
- 障害からの堅固なリカバリー。
- データ保全性の保証。
- 既存 Marketing Operations GUI ベースのカスタマーとの、影響を最小限にした統合。
- 基礎になっている実装詳細からプログラマーを隔離しつつ、Marketing Operations コンポーネントへのきめ細かいアクセスを提供。

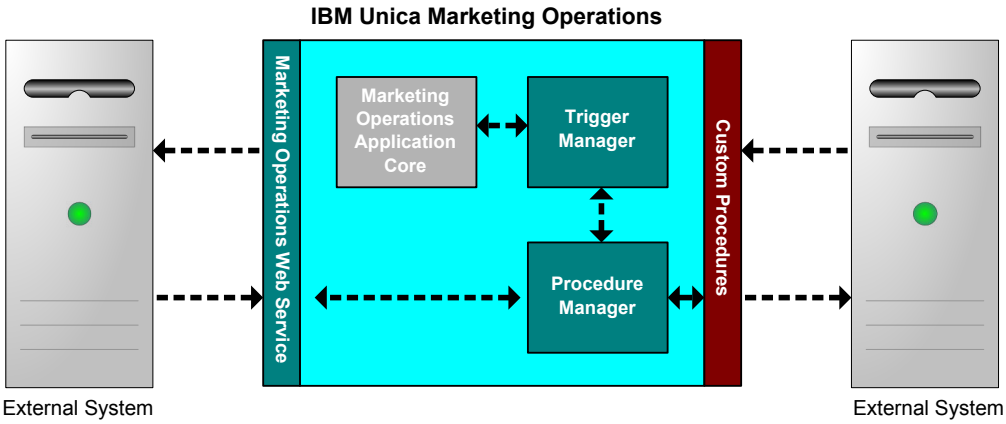
© Copyright IBM Corp. 2002, 2012

IBM Unica Marketing Operations統合サービス入門

IBM Unica Marketing Operations統合サービスの機能を使用して、カスタム・プロシージャーを作成できます。これらのプロシージャーを使用して、Marketing Operations 内部で特定のイベントが発生したときに、外部イベントをトリガーできます。これらのプロシージャーを使用して、外部システムまたは外部プログラムから Marketing Operations の機能を実行できます。

ユーザー・レベルで GUI を Marketing Operations とのインターフェースとして使用するのと同じようにして、この API を IBM Unica Marketing Operationsとのインターフェースとしてプログラム・レベルで使用します。この API を使用して、プロシージャーを構成します。これらのプロシージャーを使用して、Marketing Operations と外部システム間の通信を行います。Marketing Operations Webservice は、プロシージャー、API、およびトリガーのコンテナー・オブジェクトです。

Marketing Operations 統合サービスのアーキテクチャーをここに示します。

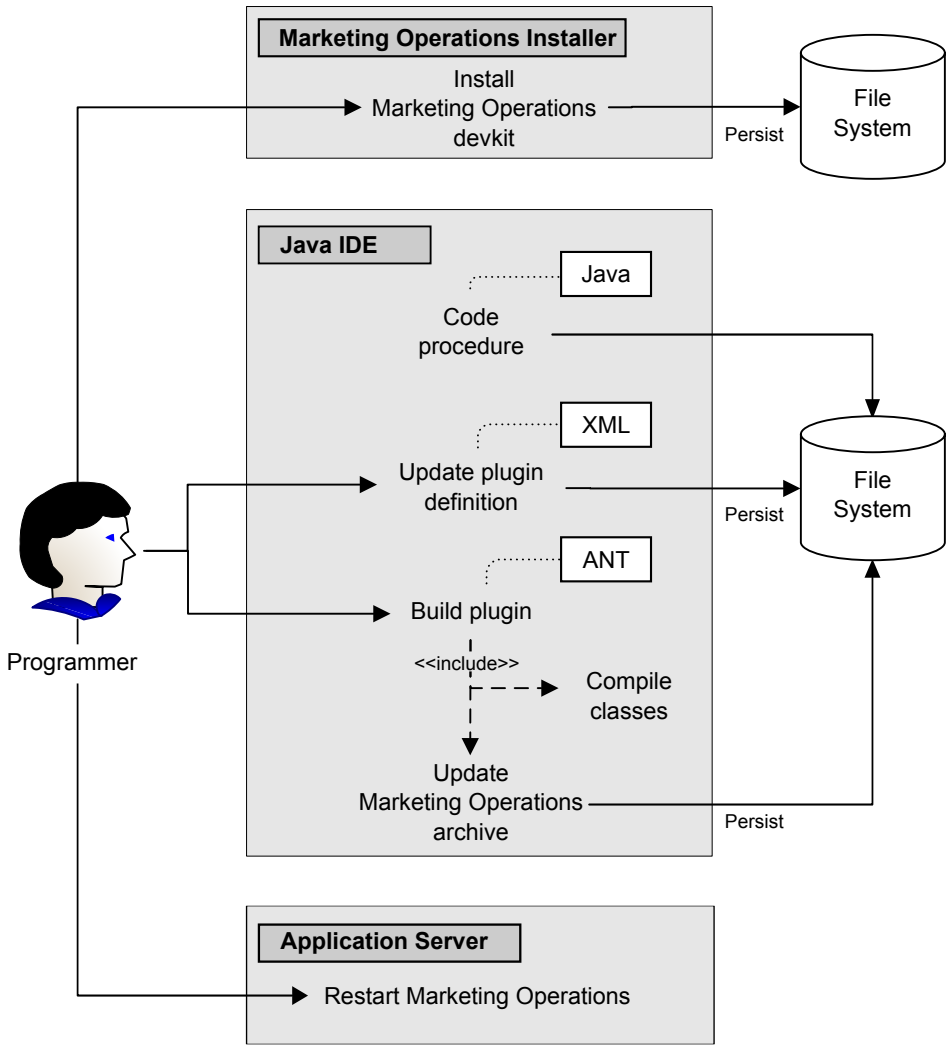


統合サービスのキー・コンポーネントは次のとおりです。

- Marketing Operations Procedure Manager: API により Marketing Operations と対話して、ビジネス・ロジックを拡張します。
- Marketing Operations Trigger Manager: 条件 (例えば、マーケティング・オブジェクトの状態変更) をアクション (トリガーの条件が合致したときに実行するプロシージャー) と関連付けます。

方法論

この図に示されているように、IBM Unica Marketing Operations統合サービスのコンポーネントを使用して、カスタム・プロシージャーを開発します。



Marketing Operations インストーラーを使用してデベロッパーズ・キットをインストールした後、これらの基本的な手順に従います。

1. カスタム・プロシージャーのコーディングを行います。現在のところ、Java を使用する必要があります。
2. XML 定義ファイルのプラグイン定義を更新します。
3. 次のようにして、プラグインをビルドします。
   a. 必要なクラスをコンパイルします。
   b. Marketing Operations アーカイブ (WAR ファイル) を更新します。
4. Marketing Operations を再始動します。

IBM Unica Marketing Operationsと API の間で通信する基本的なサンプル

このセクションでは、API と Marketing Operations の間で通信を確立する基本的なサンプルについて記述します。これは、実際的な作業は何も行いません。Marketing Operations と統合サービス間の往復を実行します。

このセクションでは、Marketing Operations 統合サービスのデベロッパーズ・キットに含まれているサンプル・プロシージャーの一部を利用します。特に、ここで参照されているコードは次のファイルにあります。

- PlanClientFacade.java
- PlanWSNOOPTestCase.java

noop メソッドは、Marketing Operations への Web サービス呼び出しです。これは、PlanClientFacade クラス中に定義されており、配列にヌル値を入れて渡します。

```
public ProcedureResponse noop(String jobId)
  throws RemoteException, ServiceException {
  NameValueArrays parameters =
    new NameValueArrays(null, null, null, null, null, null, null, null);
  return _serviceBinding.executeProcedure("uapNOOPProcedure", jobId, parameters);
}
```

プロシージャー testExecuteProcedure は、noop メソッドを PlanClientFacade から呼び出し、Marketing Operations アプリケーションとの間で往復を確立します。

```
public void testExecuteProcedure() throws Exception {
  // Time out after a minute
  int timeout = 60000;
  PlanClientFacade clientFacade = new PlanClientFacade(urlWebService, timeout);
  System.out.println("noop w/no parameters");
  long startTime = new Date().getTime();
  ProcedureResponse response = clientFacade.noop("junit-jobid");
  long duration = new Date().getTime() - startTime;

  // zero or positive status => success
  System.out.println("Status: " + response.getStatus());
  System.out.println("Duration: " + duration + " ms");
  assertTrue(response.getStatus() >= 0);
  System.out.println("Done.");
}
```

NameValueArrays、ProcedureResponse、および他のリストされているメソッドとデータ型の詳細については、このガイドの残りの部分にある特定のセクションおよび JavaDoc を参照してください。

ホストされている JavaDoc

公式の API メソッドに関する特定の情報については、JavaDoc API ドキュメンテーション・ファイルの iPlanAPI クラスを参照してください。これらのファイルは、Marketing Operations にログインし、任意のページから 「**ヘルプ**」>「**製品資料**」を選択して、 *<version>*PublicAPI.zip をダウンロードすることによって参照できます。

第 2 章 Marketing Operations 統合 Web サービスについて

Web サービスは、Marketing Operations 統合サービスのクライアント・ビューを提供します。これは、IBM Unica Marketing Operationsサーバーの配置の一部です。このサービスは、複数の Marketing Operations Web ユーザーで同時に使用できるように設計されています。

Web サービスは、1 つの API 呼び出し、executeProcedure をサポートしています。

クライアントは、この Web サービス呼び出しを直接行います。

認証

認証は必須ではありません。すべてのクライアントは、既知の PlanAPIUser という名前の IBM Unica Marketing Operationsユーザーと関連付けられます。この特殊ユーザーのセキュリティー機能は、すべての Web サービス・クライアントの必要に合わせて、Marketing Operations システム管理者により構成されているものと想定されています。

このサービスの将来のバージョンでは、保護クライアント認証のためのさらに汎用化されたメカニズムが提供される可能性があります。

ロケール

サポートされているロケールは、現在 IBM Unica Marketing Operationsシステム・インスタンスに構成されているロケールだけです。このサービスを介してアクセス可能なロケールに依存したすべてのデータ (メッセージ、通貨など) は、システム・ロケールにあるものと想定されます。

このサービスの将来のバージョンでは、どのロケールを使用するのかを Marketing Operations に指示するメカニズムがクライアントに提供される可能性があります。

状態管理

Web サービスはステートレス です。これは、複数の API 呼び出し間でクライアントごとの情報をサービス実装が保存しないことを意味します。これにより、効果的なサービス実装が提供され、クラスター・サポートが簡単になります。

データベース・トランザクション

Web サービスは、データベース・トランザクションも編集ロックもクライアントに公開しません。しかし、すべてのプロシージャー実行の影響がアトミック になることを保証します。これは、プロシージャーが成功するか失敗するかのいずれかになるということであり、失敗した場合、データベースは API がまったく呼び出されていないかのように元と同じ状態のままになります。

© Copyright IBM Corp. 2002, 2012

Marketing Operations 統合 Web サービス・データ型について

このセクションでは、特定のサービス・バインディングまたはプログラミング実装とは独立した Web サービスで使用されるデータ型について定義します。

次の記法が使用されます。

- <型>: <型定義> は単純データ型を定義します。以下に例を示します。

  Handle: string
- <型>: [<型定義>] は複合データ型またはデータ構造を定義します。
- <型>: { <型定義> } は複合データ型またはデータ構造を定義します。

複合型エレメントおよび API パラメーターは、これらの型を使用して配列を宣言できます。以下に例を示します。

```
Handle [] handles
```

型 handles は Handle 型の配列です。

プリミティブ型

プリミティブ型は、SOAP 1.1 バインディングのサポートを簡単にするため、次のテーブルに定義されている型に制限されます。すべての型は配列として宣言できます (例えば、**String [ ]**)。本質的に、**long** などのバイナリー・データ型は、プロトコル・バインディング (例えば、SOAP) によりストリングとして表現できます。しかし、この表現は、クライアントに対して表示される時に、型のセマンティクス、暗黙的値などに影響を及ぼしません。

表 1. プリミティブ型

API 型	説明	SOAP 型	Java タイプ
boolean	ブール値: **true** または **false**	xsd:boolean	boolean
dateTime	日時値	xsd:datetime	Date
decimal	任意精度の 10 進値	xsd:decimal	java.math.BigDecimal
double	倍精度、符号付き 10 進値	xsd:double	double
int	符号付き 32 ビット整数値	xsd:int	int
integer	任意精度の符号付き整数値	xsd:integer	java.math.BigInteger
long	符号付き 64 ビット整数値	xsd:long	long
string	Unicode 文字のストリング	xsd:string	java.lang.String

MessageTypeEnum

```
MessageTypeEnum: { INFORMATION, WARNING, ERROR }
```

MessageTypeEnum は、使用可能なすべてのメッセージ型を定義する列挙型です。

- INFORMATION: 通知メッセージ
- WARNING: 警告メッセージ
- ERROR: エラーメッセージ

Message

```
Message: [MessageTypeEnum type, string code, string localizedText, string logDetail]
```

Message は、Web サービス API 呼び出しの結果を定義するデータ構造です。これには、ローカライズされていないコード、ローカライズされているテキスト、およびログ詳細のオプションのフィールドがあります。現在、すべてのローカライズされているテキストは、IBM Unica Marketing Operationsサーバー・インスタンスに設定されているロケールを使用します。

表 2. Message パラメーター

パラメーター	説明
type	MessageTypeEnum、メッセージの型を設定。
code	メッセージのストリング形式のオプションのコード。
localizedText	メッセージに関連付けるオプションのテキスト・ストリング。
logDetail	オプションのスタック・トレース・メッセージ。

NameValue

```
NameValue: [string name, int sequence]
```

NameValue は、名前と値のペアを定義する基本的な複合型です。これは、必要に応じて値配列 (シーケンスは 0 から始まる) を構成するためにサービスが使用するオプションのシーケンスも定義します。

同じ名前でシーケンス番号が異なるすべての NameValue は、値の配列に変換され、共通名に関連付けられます。

配列サイズは、最大シーケンス番号により決定されます。未指定の配列エレメントは null 値になります。配列シーケンス番号は固有でなければなりません。値とその型は、拡張型により提供されます。

表 3. NameValue パラメーター

パラメーター	説明
name	NameValue 型の名前を定義するストリング。
sequence	NameValue 暗黙値のシーケンス番号を設定する 0 から始まる整数。

拡張 NameValue 型は、次のように、各プリミティブ型に対して定義されています。

表 4. 拡張 NameValue 型

拡張型	説明
BigDecimalNameValue: NameValue [decimal value]	その値が任意精度の 10 進数である NameValue 型。
BigIntegerNameValue: NameValue [integer value]	その値が任意サイズの整数である NameValue 型。

表 4. 拡張 NameValue 型 (続き)

拡張型	説明
BooleanNameValue: NameValue [boolean value]	その値がブール値である NameValue 型。
CurrencyNameValue: NameValue [string locale, decimal value]	あるロケールの通貨を表現するのに適した NameValue 型。ロケールは ISO 言語コード、つまり、ISO-639 で定義されている小文字の 2 文字コードです。 現在のところ、ロケールは、IBM Unica Marketing Operationsサーバー・インスタンスに設定されているロケールと一致していなければなりません。
DateNameValue: NameValue [datetime value]	その値が日付である NameValue 型。
DecimalNameValue: NameValue [double value]	その値が倍精度の 10 進数である NameValue 型。
IntegerNameValue: NameValue [long value]	その値が 64 ビットの整数である NameValue 型。
String NameValue: NameValue [string value]	その値がストリングである NameValue 型。

さらに、異なる型の NameValue のセットを定義する必要があるときに使用するために、拡張 NameValue 型の配列が定義されています。

```
     NameValueArrays: [
BooleanNameValue[]    booleanValues,
StringNameValue[]     stringValues,
IntegerNameValue[]    integerValues,
BigIntegerNameValue[] bigIntegooleanNameValue,
DecimalNameValue[]    decimalValues,
BigDecimalNameValue[] bigDecimalValues
DateNameValue[]       dateNameValues
CurrencyNameValue[]   currencyValues
     ]
```

executeProcedure

構文

```
executeProcedure(string key, string jobid, NameValueArrays paramArray)
```

戻り値

```
int: status
Message[]: messages
```

説明

このメソッドは、指定されたプロシージャーを、オプションでパラメーターの配列を指定して呼び出します。この呼び出しは同期的に実行されます。つまり、クライアントをブロックし完了時に結果を返します。

パラメーター

表 5. executeProcedure パラメーター

名前	説明
key	実行するプロシージャーの固有キー。**key** に何もプロシージャーがバインドされていない場合、*RemoteException* エラーが返されます。
jobid	このプロシージャー実行に関連したジョブを指定するオプションのストリング。このストリングはパススルー項目ですが、クライアント・ジョブを特定のプロシージャーの実行と結びつけるために使用できます。
paramArray	プロシージャーに渡すパラメーターの配列。1 つ以上のパラメーターが無効な場合 (間違った型、正しくない値など)、エラー・ステータスとメッセージが返されます。プロシージャーが必要とするパラメーター、その型、および値の数を決めるのは、クライアントの責任です。

戻りパラメーター

表 6. executeProcedure 戻りパラメーター

名前	説明
status	整数コード。 • 0 は、プロシージャーが正常に実行されたことを示します。 • 他の整数値は、エラーを示します。 プロシージャーは、このステータスを使用して、異なるレベルのエラーを示せます。
messages	0 個以上のメッセージ・データ構造の配列。**status** が 0 の場合、この配列には ERROR メッセージは含まれませんが、INFORMATION および WARNING メッセージを含めることは可能です。 **status** が 0 でない場合、メッセージには ERROR、INFORMATION、および WARNING メッセージのいずれでも混合して含められます。

Marketing Operations 統合サービス WSDL

このトピックでは、Marketing Operations 統合サービス用の Web サービス記述言語 (WSDL) を定義します。WSDL は手動で定義され、Web サービス定義における最終的なものです。

Axis

この Web サービスのバージョンは、Axis2 1.5.2 を使用して WSDL ファイルから Web サービス実装を構成するサーバー・サイド・クラスを生成します。ユーザーは、任意のバージョンの Axis または Axis 以外の技法を使用して、提供されている WSDL からの API を統合するクライアント・サイド実装を作成できます。

プロトコル・バージョン

プロトコルのバージョンは、次のように明示的に WSDL とバインドされます。

- WSDL 名の一部として (例えば、`PlanIntegrationService1.0.wsdl`)

- WSDL targetNamespace の一部として (例えば、xmlns:tns="http://webservices.unica.com /MktOps/services/PlanIntegrationServices1.0?wsdl")

WSDL

次の 1 つの WSDL ファイルが IBM Unica Marketing Operations統合サービスで提供されます: PlanIntegrationServices1.0.wsdl。この WSDL は、integration/examples/soap/plan ディレクトリーに配置されています。サンプル・ビルド・スクリプトはこのファイルを使用して、Web サービスに接続するためのクライアント・サイドの適切なスタブを生成します。

第 3 章 IBM Unica Marketing Operationsプロシージャー

プロシージャー は、IBM Unica Marketing Operationsによりホストされるカスタムまたは標準の Java クラスであり、ある作業単位を行います。プロシージャーは、顧客、パートナー、および IBM Unica Professional Services が Marketing Operations ビジネス・ロジックを任意の方法で拡張する手段を提供します。

プロシージャーは、単純なプログラミング・モデルに従い、明確に定義された API を使用して Marketing Operations により管理されるコンポーネントに作用します。プロシージャーは、単純な検索メカニズムおよび XML ベースの定義ファイルにより「ディスカバー」されます。Marketing Operations は、「クライアント」の必要に応じでプロシージャーを実行します。例えば、統合要求 (入力) またはトリガー起動 (内部または出力) に応答します。

プロシージャーは、クライアントに関しては同期的に実行します。結果はクライアントが直接使用可能で、永続的な監査メカニズムで監査されます。プロシージャーの実行により、他のイベントやトリガーを発生させ、Marketing Operations を起動することもできます。

プロシージャーは Java で書かなければなりません。

前提事項

プロシージャーに関する次の前提事項に注意してください。

- プロシージャー実装クラスは、個別のクラス・ツリーまたは jar ファイルにパッケージされ、URL パスにより IBM Unica Marketing Operationsで使用可能になっています。プロシージャー実行マネージャーは、独立クラス・ローダーを使用してこれらのクラスを必要に応じてロードします。デフォルトでは、Marketing Operations は次のディレクトリーを探します。

  <Marketing Operations_home>/devkits/integration/examples/classes

  このデフォルトを変更するには、**「設定」>「構成」>「Marketing Operations」>「umoConfiguration」>「integrationServices」**で **integrationProcedureClasspathURL** パラメーターを設定します。
- プロシージャー実装クラス名は受け入れられている Java 命名規則に従うようにして、「unica」と他のベンダーからのクラスのパッケージが衝突しないようにします。特に、ユーザーはプロシージャーを **com.unica** または **com.unicacorp** パッケージ・ツリーの下に置いてはなりません。
- プロシージャー実装は、アプリケーション・サーバーで IBM Unica Marketing Operationsが使用する Java ランタイム・バージョン (JRE 1.5.10 以上) に合わせてコーディングしなければなりません。
- IBM Unica Marketing Operationsは、いくつかのオープン・ソースとサード・パーティーのライブラリーを提供しています。アプリケーション・サーバーは、これらのライブラリーの別のバージョンを使用することもあります。一般的に、このリストはリリースごとに変わります。

© Copyright IBM Corp. 2002, 2012

注: 互換性の問題を回避するため、プロシージャーはオープン・ソース、サード・パーティーまたはアプリケーション・サーバー特定のライブラリーを何も使用すべきではありません。

しかし、そのようなパッケージがプロシージャーで使用されたり、プロシージャーがインポートする 2 次クラスにより使用されたりする場合、Marketing Operations またはアプリケーション・サーバー (またはその両方) が提供するパッケージに完全に適合した使用法でなければなりません。この場合、Marketing Operations の将来のバージョンがライブラリーをアップグレードするか使用しなくなるかした場合、プロシージャーのコードを修正しなければならなくなる可能性があります。

- プロシージャー実装クラスは、IBM Unica Marketing Operationsが通常使用するクラス・ロード・ポリシーに従ってロードされます (通常は **parent-last**)。アプリケーション・サーバーは、Marketing Operations プロシージャーに適用するクラスを再ロードする開発ツールとオプションを提供するかもしれません。しかし、これは必須ではありません。
- プロシージャーは、自分自身の状態に関してスレッド・セーフでなければなりません。つまり、その実行メソッドは、呼び出しごとに変わる内部状態に依存することはできないということです。
- プロシージャーは、自分自身でスレッドを作成することはできません。

設計

設計は、アトミックに実行される単一の作業単位の作成に焦点を当てて行わなければなりません。理想的には、プロシージャーは、将来のある時点で実行されるよう非同期的にスケジュールできるいくつかのタスクのシリーズを実行します。この「応答不要送信」統合モデルは、双方のシステムの負荷を最小にします。

プロシージャー実装クラスは IBM Unica Marketing Operations API を使用して、Marketing Operations コンポーネントの読み取りと更新、サービスの呼び出しなどを行います。他の Java パッケージを使用して他のタスクを実行できます。

注: 文書化されているクラスとメソッドだけが、Marketing Operations の将来のリリースでサポートされます。Marketing Operations の他のすべてのクラスとメソッドは使用できないものと見なす必要があります。

コーディングしてコンパイルされれば、プロシージャー実装クラスは Marketing Operations で使用可能になるはずです。Marketing Operations 統合サービスで提供されるビルド・スクリプトは、コンパイルしたプロシージャーをデフォルトの場所に置きます。開発の最終ステップは、カスタム・プロシージャーをディスカバーするために Marketing Operations が使用するカスタム・プロシージャー・プラグイン定義ファイルを更新することです。

プロシージャーは **com.unica.publicapi.plan.plugin.procedure.IProcedure** インターフェースを実装し、パラメーターのないコンストラクター (通常の JavaBean モデル) を持たなければなりません。各プロシージャーのコーディングとコンパイルは、Eclipse、Borland JBuilder、Idea などの Java IDE のうち好みのもので行えます。デベロッパー・ツールキットとして IBM Unica Marketing Operationsで提供されているサンプル・コードは、次の場所にあります。

<*Marketing Operations-home*>/devkits/integration/examples/src/procedure

構成

「設定」>「構成」>「Marketing Operations」>「umoConfiguration」>「integrationServices」にあるパラメーターを使用して、Marketing Operations 統合モジュールを構成します。

詳細については、『*Marketing Operations*インストール・ガイド』を参照してください。

プロシージャーのライフサイクル

プロシージャーのランタイム・ライフサイクルは次のとおりです。

1. ディスカバリーと初期化
2. 選択 (オプション)
3. 実行
4. 消滅

ディスカバリーと初期化

IBM Unica Marketing Operationsは、特定のインストール・インスタンスで使用可能なすべての標準およびカスタム・プロシージャーを知ることができなければなりません。この処理は、ディスカバリーと呼ばれます。

注: 標準プロシージャー (Marketing Operations エンジニアリング・チームにより定義されたプロシージャー) は暗黙的に知られているので、ディスカバーされるためのアクションは何も必要ありません。

カスタム・プロシージャーは、プロシージャー・プラグイン定義ファイルで定義します。Marketing Operations プラグイン・マネージャーは、初期化中にこのファイルを読み取ります。見つかったそれぞれのプロシージャーについて、プラグイン・マネージャーは次のことを行います。

1. プロシージャーのインスタンスを生成します。状態を INSTANTIATED に遷移します。
2. 新規プロシージャー監査レコードを作成します。
3. プロシージャーがインスタンス化された場合、プラグイン記述ファイルにある初期化パラメーターを指定して **`initialize()`** メソッドを呼び出します。このメソッドが例外をスローした場合、そのステータスはログに記録され、プロシージャーは中止します。そうでない場合、プロシージャーは INITIALIZED 状態に遷移します。これで、実行の準備ができました。
4. 新規プロシージャー監査レコードを作成します。
5. プロシージャーが初期化できた場合、**`getKey()`** メソッドが呼び出され、そのプロシージャーを参照するためにクライアントが使用しているキーを特定します。このキーは、インスタンスと関連付けられ、後で検索するために保存されます。

選択

時々、IBM Unica Marketing Operationsは使用可能なプロシージャーのリストをユーザーに表示する必要があります。例えば、管理者がトリガーをセットアップできるようにするような場合です。これは、プロシージャーが初期化されてからでなければ行われません。プロシージャーの **getDisplayName()** および **getDescription()** メソッドがこの目的で使用されます。

実行

プロシージャーが初期化された後のある時点で、IBM Unica Marketing Operationsはそのプロシージャーの実行要求を受け取ります。これは、他のスレッドで実行される他のプロシージャー (または同じプロシージャー) と同時に発生する可能性があります。

実行時に、プロシージャー実行マネージャーは次のことを行います。

1. データベース・トランザクションを開始します。
2. プロシージャーの状態を EXECUTING に設定します。
3. 新規プロシージャー監査レコードを作成します。
4. プロシージャーの **execute()** メソッドを、クライアントが提供する実行コンテキストと実行パラメーターを指定して呼び出します。メソッド実装は Marketing Operations API を必要に応じて使用し、編集ロックを取得して実行コンテキストを渡します。execute メソッドが例外をスローした場合、実行マネージャーは、そのトランザクションにロールバックのマークを付けます。
5. 実行結果に応じて、トランザクションのコミットまたはロールバックが行われます。プロシージャーの状態は EXECUTED に設定されます。
6. 未処理のすべての編集ロックが解放されます。
7. 新規プロシージャー監査レコードを作成します。

注: execute() メソッドは、プロシージャー・インスタンス・データを変更してはなりません。

消滅

IBM Unica Marketing Operationsがシャットダウンする際、プロシージャー・プラグイン・マネージャーは、ロードされたすべてのプロシージャーをウォークスルーします。見つかったそれぞれのプロシージャーについて、次のことを行います。

1. プロシージャーの destroy() メソッドを呼び出し、インスタンスが破棄される前にプロシージャーがクリーンアップを行えるようにします。
2. プロシージャーの状態を FINALIZED に変更します (実行できない場合)。
3. 新規プロシージャー監査レコードを作成します。
4. プロシージャーのインスタンスを破棄します。

データ・ロック

IBM Unica Marketing Operationsは、排他的編集ロッキング方式を使用します。つまり、コンポーネント・インスタンスに対する更新アクセスを付与されるのは、一度に 1 ユーザーだけです。GUI ユーザーについては、このロッキングは表示されるタブ・レベルで行われます。これは 1 つのインスタンスの 1 つのサブセット (例えば、「プロジェクト・サマリー」タブ) を表わす場合もあれば、複数のインスタンス (「ワークフロー」タブ) を表わす場合もあります。ユーザーがロックを取得すると、関連データに関し、他のすべてのユーザーは読み取り専用アクセスに制限されます。

注: 編集ロックはデータベースのトランザクションと同じではありません。

プロシージャーによってコンポーネント・インスタンスまたはインスタンスのグループになされる変更が、別のユーザーによって不注意に上書きされないようにするため、プロシージャーはコンポーネント・データを更新する前に適切なロックを取得しなければなりません。プロシージャーの **execute()** メソッドに渡される実行コンテキスト・オブジェクトを使用してこのことを行います。

何らかのデータを更新する前に、プロシージャーは必要とする各ロックについてコンテキストの **acquireLock()** メソッドを呼び出さなければなりません。例えば、プロシージャーがプロジェクトとその関連ワークフローを更新しようとする場合、プロシージャーはその両方のロックを取得する必要があります。

別のユーザーが既にロックを取得している場合、**acquireLock()** メソッドは直ちに **LockInUseException** をスローします。衝突を最小限に抑えるために、プロシージャーはオブジェクトを更新したらすぐにロックを解放する必要があります。

実行マネージャーは、実行メソッドが戻るときに、あらゆる未処理のロックを自動的に解放します。どの場合でも、ロックはデータベース・トランザクションの存在期間しか保持されません。つまり、データベース固有のトランザクション・タイムアウトになった場合、ロックは解放されます。

プロシージャー・トランザクション

プロシージャー実行マネージャーは、自動的にプロシージャーの実行をデータベース・トランザクションでラップし、プロシージャー実行の結果に基づき適切にコミットまたはロールバックを行います。これにより、IBM Unica Marketing Operations データベースへの更新はコミットされるまで他のユーザーには不可視で、更新はアトミックであることが保証されます。

それでも、プロシージャーの書き込みプログラムは、そのプロシージャーの実行が完了する前に他のユーザーがデータベースに変更を書き込めないようにするため、必要な編集ロックを取得する必要があります。

通信結果

プロシージャーの **execute()** メソッドは、IBM Unica Marketing Operationsプロシージャー監査テーブルに記録されて保管される、整数の状況コードおよび 0 個以上のメッセージを返します。クライアントは、なんらかの他の方法で状況情報について通信する可能性もあります。

プロシージャーのログ

IBM Unica Marketing Operationsでは、プロシージャー毎に個別のログ・ファイルがあります。

<Marketing Operations-home>¥logs¥procedure.log

プロシージャー実行マネージャーは、各プロシージャーのライフサイクルをログに記録し、監査レコードを作成します。

- **logInfo()**: 情報メッセージをプロシージャー・ログに書き込みます
- **logWarning()**: 警告メッセージをプロシージャー・ログに書き込みます
- **logError()**: エラー・メッセージをプロシージャー・ログに書き込みます
- **logException()**: 例外のスタック・トレースをプロシージャー・ログにダンプします

キーとなる Java クラス

提供されている統合 devkit には、公開 IBM Unica Marketing Operations API とサポートするクラスの Javadoc のセットが含まれています。最も重要なものを、ここにリストします。

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure): すべてのプロシージャーで実装する必要があるインターフェース。プロシージャーは、明確に定義されたライフサイクルをたどり、作業を実行するために Marketing Operations API にアクセスします。
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure): すべてのトリガー・プロシージャーで実装する必要があるインターフェース (マーカー・インターフェース)。
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext): 実行マネージャーによりプロシージャーに渡される不透明なコンテキスト・オブジェクトのインターフェース。このオブジェクトには、ログと編集ロック管理のための public メソッドがあります。プロシージャーは、また、このオブジェクトをすべての PlanAPI 呼び出しに渡します。
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI): Marketing Operations API へのインターフェース。実行コンテキストは、適切な実装を取得する **getPlanAPI()** メソッドを提供します。

プロシージャーのサンプル

このサンプルは、統合 Web サービスまたはトリガーから、プロジェクトの状態を変更する標準プロシージャーを示しています。

注: このサンプル・プロシージャーとその XML 定義を変更しないでください。このサンプルは、IBM Unica Marketing Operationsをアップグレードすると上書きされるので、行った変更が失われます。すべてのカスタム・プロシージャーの作成および変更は、別のディレクトリーで行なわなければなりません。

```
// ProjectStateChangeProcedure
// (c) Copyright 2006 by Unica Corporation. All rights reserved.

package com.unica.uap.plugin.procedure.standard;

import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Map;

import com.unica.publicapi.plan.api.Handle;
import com.unica.publicapi.plan.api.IExecutionContext;
import com.unica.publicapi.plan.api.IPlanAPI;
import com.unica.publicapi.plan.api.LockInUseException;
import com.unica.publicapi.plan.api.ProjectHandle;
import com.unica.publicapi.plan.api.ProjectStateEnum;
import com.unica.publicapi.plan.plugin.PluginVersion;
import com.unica.publicapi.plan.plugin.procedure.IProcedure;
import com.unica.publicapi.plan.plugin.procedure.ProcedureExecutionException;
import com.unica.publicapi.plan.plugin.procedure.ProcedureInitializationException;
import com.unica.publicapi.plan.plugin.procedure.ProcedureMessage;
import com.unica.publicapi.plan.plugin.procedure.ProcedureMessageTypeEnum;
import com.unica.publicapi.plan.plugin.procedure.ProcedureResult;

/**
 * <b>ProjectStateChangeProcedure</b> is a standard Marketing Operations procedure
 * that attempts to
 * transition the state of a project.
 * <p>
 * Expects the following initialization parameters:
 * <ul>
 *   <li>debug: Boolean object, <tt>true</tt> or <tt>false</tt>, indicating
 *   if debug tracing is enabled or not</li>
 * </ul>
 *
 * <p>
 * Expects the following execute parameters:
 * <ul>
 *   <li>hProject: string array form of project handle, e.g.,
 *     "http://mymachine:7001/MktOps/affiniumplan.jsp?
 *         cat=projecttabs&projectid=12"</li>
 *   <li>uapState: string array form of new project state, e.g.,
 *      "COMPLETED". Note, case matters!</li>
 * </ul>
 *
 */
public final class ProjectStateChangeProcedure implements IProcedure {
// initialization parameters
private final static String DEBUG_INITPARAMETER_NAME = "debug";
  // execute parameters
  private final static String HPROJECT_PARAMETER_NAME = "hProject";
  private final static String STATE_PARAMETER_NAME
    = IPlanAPI.PROJECT_ATTRIBUTE_STATEENUM; // same as attribute name

  // our status codes
  private final static int STATUS_SUCCESS = 0;

    // debug property. set the procedure's "debug" init parameter
  //   to true to enable debug trace
    private boolean _debug = false;
    private boolean isDebug() { return _debug; }
```

```
  // simple name is unqualified class name
  public String getName() {
        return "uapProjectStateChangeProcedure";
    }

  // display name is always key
  public String getDisplayName(Locale locale) {
        // only do EN for now
        return getName();
    }

  // description always in english
    public String getDescription(Locale locale) {
        // only do EN for now
          return "A procedure to transition the state of a project.";
    }

  // version we're coded to; must be 1.0.0 for now
  public PluginVersion getVersion() {
    return new PluginVersion(1,0,0);
  }

  // initialize instance from init parameters
    public void initialize(Map initParameters)
      throws ProcedureInitializationException {
    // the only init parameter we have is: debug, Boolean
        if (initParameters.containsKey(DEBUG_INITPARAMETER_NAME)) {
            try {
                _debug = ((Boolean)initParameters.get(DEBUG_INITPARAMETER_NAME)).
                  booleanValue();
            } catch (Exception exception) {
                throw new ProcedureInitializationException("Problem using "
                                        + DEBUG_INITPARAMETER_NAME
                                        + " init parameter: "
                                        + exception.getMessage());
        }
        }
    }

  // execute: expect hProject and state enum
    public ProcedureResult execute(IExecutionContext context, Map parameters)
      throws ProcedureExecutionException {
        // get execute parameters: we expect two:
        // - hProject: string[] form of project handle
        // - uapState: string[] form of ProjectStateEnum
        ProjectHandle hProject = null;
    if (parameters.containsKey(HPROJECT_PARAMETER_NAME)) {
      try {
        hProject = (ProjectHandle)
Handle.makeHandle(((String[])parameters.get(HPROJECT_PARAMETER_NAME))[0]);
      } catch (Exception exception) {
        throw new ProcedureExecutionException("Problem using "
                            + HPROJECT_PARAMETER_NAME
                            + " parameter: "
                            + exception.getMessage());
      }
    } else throw new ProcedureExecutionException(HPROJECT_PARAMETER_NAME
                            + " parameter must be provided.");

        ProjectStateEnum stateEnum = null;
    if (parameters.containsKey(STATE_PARAMETER_NAME)) {
      try {
        stateEnum =
          ProjectStateEnum.valueOf(((String[])parameters.
            get(STATE_PARAMETER_NAME))[0]);
      } catch (Exception exception) {
```

```
            throw new ProcedureExecutionException("Problem using "
                                + STATE_PARAMETER_NAME
                                + " parameter: "
                                + exception.getMessage());
          }
        } else throw new ProcedureExecutionException(STATE_PARAMETER_NAME
                                + " parameter must be provided.");

        int status = -1;
        ProcedureMessage[] messages = null;
            try {
               // try to acquire an edit lock for the project
                context.acquireLock(hProject, IExecutionContext.LOCK_ALL_FIELDS);

                // use PlanAPIImpl to update state
                IPlanAPI planAPI = context.getPlanAPI();
                planAPI.updateAttribute(context, hProject, STATE_PARAMETER_NAME,
                            new ProjectStateEnum[]{stateEnum});

              // success
              status = STATUS_SUCCESS;

            } catch (Exception exception) {
              // write stack trace if debug
                if (isDebug()) {
                    context.logError(getName(), exception);
                }
                throw new ProcedureExecutionException(exception);
            } finally {
              // release our lock
              try {
                context.releaseAllLocks();
              } catch (Exception exception) { /* ignored */ }
            }

            return new ProcedureResult(status, messages);
        }

        public void destroy( ){
             // we don't need to do anything
        }
    }
```

プロシージャー・プラグイン定義ファイル

このトピックでは、プロシージャー・プラグイン定義ファイルについて説明します。このファイルは、IBM Unica Marketing Operations内にホストされるカスタム・プロシージャーに関する実装クラス、メタデータ、および他の情報を定義します。デフォルトでは、プロシージャー・プラグイン定義は次のパスにあるものと想定されます。

```
<Marketing Operations-home>/devkits/integration/examples/src/
  procedures/procedure-plugins.xml
```

このファイルは、下記の情報を含む XML 文書です。

Procedures: 0 個以上の **Procedure** 要素のリスト。

Procedure: プロシージャーを定義する要素。各プロシージャーには、次の要素が含まれます。

- **key** (オプション): プロシージャーの検索キーを定義するストリング。このキーは、特定の Marketing Operations インスタンスによりホストされるすべての標準

(IBM 提供) およびカスタム・プロシージャーに関して固有でなければなりません。定義されない場合、デフォルトは、**className** 要素の完全修飾バージョンになります。ストリング「uap」で始まる名前は、IBM Unica Marketing Operations の使用のために予約されています。

- **className** (必須): プロシージャー・クラスの完全修飾パッケージ名。このクラスは、IProcedure クラス (com.unica.public.plan.plugin.procedure.IProcedure) を実装しなければなりません。
- **initParameters** (オプション): 0 個以上の initParameter 要素のリスト。

  initParameter (オプション): プロシージャーの initialize() メソッドに渡されるパラメーター。この要素には、ネストされたパラメーター名、型、および値要素が含まれます。

  – name: パラメーター名を定義するストリング
  – type: パラメーター値の型を定義する Java ラッパー・クラスのオプションのクラス名。次の型のいずれかである必要があります。
    - java.lang.String (デフォルト)
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  – value: その型に応じた属性値のストリング形式

第 4 章 IBM Unica Marketing Operations API について

IBM Unica Marketing Operations API は、実行中の Marketing Operations インスタンスのクライアント・ビューを提供するファサードです。Marketing Operations 機能のサブセットだけが公開されます。API は、Marketing Operations Web ユーザーおよび Marketing Operations 統合サービス WebService SOAP 要求およびトリガーと共に使用するように設計されています。API は次のタイプの操作をサポートします。

- コンポーネントの作成と削除
- ディスカバリー (コンポーネント・タイプ、属性値、などによる)
- コンポーネント検査 (属性、特殊リンク、などによる)
- コンポーネント変更

バージョンと後方互換性

この API の将来のバージョンでは、メジャー・バージョン番号が同じすべてのマイナー・リリースおよびメンテナンス・リリースとの後方互換性を持ちます。しかし、ビジネス上または技術上必要になる場合、メジャー・リリースの初期のバージョン (x.0) との互換性を保たない権利を IBM は留保します。

この API のメジャー・バージョン番号は、次のいずれかの変更がなされた場合に増加します。

- データ解釈の変更
- ビジネス・ロジックの変更 (例えば、サービス・メソッド機能の変更)
- メソッドのパラメーターまたは戻りの型 (またはその両方) の変更

この API のマイナー・バージョン番号は、次のいずれかの変更がなされた場合に増加します (これらの変更は、後方互換性の定義によるものであることに注意してください)。

- 新規メソッドの追加
- 新規データ型の追加かつその使用が新規メソッドに限定
- 列挙型への新規エレメントの追加
- インターフェースの新規バージョンを、バージョン・サフィックスを付けて定義

ユーザー・セキュリティー

認証はプロシージャーの実行マネージャーにより行われているものと想定され、すべての API により使用される実行コンテキストに認証ユーザー情報がバインドされます。API は認証ユーザーを公開しませんが、使用する必要がある場合は IBM Unica Marketing Operations に渡されます。

しかし、認証ユーザーは、API によって公開されているすべての操作の実行を許可されているわけではないかもしれません。この場合、API メソッドは **AuthorizationException** をスローします。

© Copyright IBM Corp. 2002, 2012

ロケール

このバージョンでサポートされているロケールは、IBM Unica Marketing Operations サーバー・インスタンスに現在構成されているロケールだけです。API を介してアクセス可能なロケールに依存したすべてのデータ (メッセージ、通貨など) は、このシステム・ロケールにあるものと想定されます。

状態管理

この API はステートレスです。これは、複数の呼び出し間でクライアントごとの情報が API によって保存されないことを意味します。

しかし、特定の API 呼び出しは、IBM Unica Marketing Operationsにより管理されている基礎となっているコンポーネント・インスタンスの状態を変更するかもしれず、これらの状態変更はデータベースに保存される可能性があることに注意してください。

データベース・トランザクション

この API はデータベース・トランザクションをクライアントには公開しませんが、実行コンテキストにそのような情報が含まれている場合はそれを使用します。トランザクションが開始すると、特定のプロシージャー内部のすべての API 呼び出しの影響はアトミックになります。IBM Unica Marketing Operationsの他のユーザーは、トランザクションをプロシージャーが正常にコミットするまでその変更を知ることはありません。

データベースを更新する API 呼び出しは、最初に編集ロックを取得し、API 呼び出しの間に他の Marketing Operations ユーザーが基礎となっているデータを変更しないようにしなければなりません。他のユーザーは、その API が完了するまでロックされたコンポーネントを更新することはできません。同じように、使用したいデータのロックを別の Marketing Operations ユーザーまたは API クライアントが取得している場合もあり、その場合 API 呼び出しが完了しません。

イベント処理

この API を介して IBM Unica Marketing Operationsコンポーネントで実行される操作は、その操作が Marketing Operations Web ユーザーにより実行される場合と同じイベントを生成します。特に、特定のイベントを待機するトリガーは、両方の場合に起動します。特定の通知 (例えば、プロジェクト状態が変更したとき) にサブスクライブしているユーザーは、API 呼び出しの結果生じる状態変更を Web ユーザーのアクションの結果生じる状態変更と同じように通知されます。

API データ型について

IBM Unica Marketing Operations API は、次の公用データ型を含んでいます。

- **ApprovalMethodEnum**: 承認メソッドの列挙型。
- **ApprovalStateEnum**: 承認状態の列挙型。
- **AssetLibraryStateEnum**: 資産ライブラリー状態の列挙型。
- **AssetStateEnum**: 資産状態の列挙型。
- **AttachmentTypeEnum**: 添付ファイルの列挙型。

- **AttributeMap**: 属性とその値を含む Java Map。
- **BudgetPeriodEnum**: 予算期間型の列挙型。
- **BudgetTypeEnum**: 予算型の列挙型。
- **Handle**: 特定の Marketing Operations インスタンス中のコンポーネント・インスタンスを特定します。
- **InvoiceStateEnum**: 請求状態の列挙型。
- **MonthEnum**: 暦年の月の列挙型。
- **ProjectCopyTypeEnum**: コピー・プロジェクト・オプションの列挙型。
- **ProjectStateEnum**: プロジェクト状態の列挙型。
- **TaskStateEnum**: タスク状態の列挙型。
- **QuarterEnum**: 暦年の四半期の列挙型。
- **WeekEnum**: 暦年の週の列挙型。

各データ型で使用できる値は、以下で説明されています。

列挙型

ApprovalMethodEnum

ApprovalMethodEnum は、承認の使用可能なすべてのメソッドを定義する列挙型です。使用可能な値は次のとおりです。

- SIMULTANEOUS
- SEQUENTIAL

ApprovalStateEnum

ApprovalStateEnum は、承認の使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- NOT_STATED
- ON_HOLD
- IN_PROGRESS
- COMPLETED
- CANCELLED

AssetLibraryStateEnum

AssetLibraryStateEnum は、資産ライブラリーの使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- ENABLED
- DISABLED

AssetStateEnum

AssetStateEnum は、資産の使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- DRAFT

- LOCK
- FINALIZE
- ARCHIVE

AttachmentTypeEnum

AttachmentTypeEnum は、添付ファイルの使用可能なすべての型を定義する列挙型です。使用可能な値は次のとおりです。

- URL
- FILE
- ASSET

BudgetPeriodEnum

BudgetPeriodEnum は、使用可能なすべての予算期間を定義する列挙型です。使用可能な値は次のとおりです。

- QUARTERLY
- MONTHLY
- WEEKLY
- YEARLY
- ALL

BudgetTypeEnum

BudgetTypeEnum は、使用可能なすべての予算型を定義する列挙型です。使用可能な値は次のとおりです。

- TOTAL
- FORECAST
- COMMITTED
- ACTUAL
- ALLOCATED

InvoiceStateEnum

InvoiceStateEnum は、請求の使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- DRAFT
- PAYABLE
- PAID
- CANCELLED

MonthEnum

MonthEnum は、月に使用可能なすべての値を定義する列挙型です。

ProjectCopyTypeEnum

ProjectCopyTypeEnum は、コピー・プロジェクトの使用可能なすべての型を定義する列挙型です。使用可能な値は次のとおりです。

- COPY_USING_PROJECT_METRICS
- COPY_USING_TEMMPLATE_METRICS

これらのプロジェクトとタスク状態についての詳細は、「*Marketing Operations* ユーザー・ガイド」を参照してください。

ProjectStateEnum

ProjectStateEnum は、プロジェクトまたは要求の使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- NOT_STARTED
- IN_PROGRESS
- ON_HOLD
- IN_RECONCILIATION
- LATE: プロジェクトがスケジュールされた開始日までに開始しなかったことを示します。
- OVERDUE: プロジェクトがスケジュールされた終了日までに完了しなかったことを示します。
- DRAFT
- SUBMITTED
- RETURNED
- ACCEPTED
- COMPLETED
- CANCELLED
- DELETED

TaskStateEnum

TaskStateEnum は、ワークフロー・タスクの使用可能なすべての状態を定義する列挙型です。使用可能な値は次のとおりです。

- PENDING
- ACTIVE
- FINISHED
- SKIPPED
- DISABLED

QuarterEnum

QuarterEnum は、四半期の使用可能なすべての値を定義する列挙型です。

WeekEnum

WeekEnum は、週の使用可能なすべての値を定義する列挙型です。

Handle

Handle は、特定の Marketing Operations インスタンス中のコンポーネント・インスタンスを参照する特別 URL オブジェクトです。ハンドルには、コンポーネント・タイプ、内部データ ID、インスタンス・ベースの URL などが含まれます。API により使用されるか生成されるハンドルは、Marketing Operations GUI のコンポーネントのビューにナビゲートするために使用できる完全 URL に外部化したり、カット・アンド・ペーストしたり、E メールで送信したり、別のプロシージャーでパラメーターとして使用したりすることなどができます。

ハンドルは特定の Marketing Operations サービス・インスタンスまたはクラスター化されたインスタンスだけで有効ですが、配置されたサービスの存続期間、有効であることが保証されます。結果として、後で参照するためにハンドルをファイルに保存できますが、別の Marketing Operations インスタンス (同じマシンに存在する場合であっても) のコンポーネントにアクセスするために使用することはできません。しかし、Marketing Operations は、異なるベース URL を現在のインスタンスにマッピングし、インスタンスを別のマシンに再配置して接続する手段を提供します (例えば、元のマシンが誤動作する場合)。

ハンドルは、クライアントに依存していません。例えば、トリガーはハンドルをプロシージャーに渡すことができます。プロシージャーはハンドルを、サード・パーティー・システムへの SOAP 呼び出しのパラメーターとして使用し、そのシステムがターンアラウンドして SOAP 要求を発行して Marketing Operations に戻し、属性を更新するプロシージャーを呼び出します。

Handle クラスには、様々なタイプの URL からハンドルを作成するファクトリー・メソッドがあります。

Approval オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=approvaldetail&approvalid=101
```

AssetLibrary オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=library&id=101
```

Asset Folder オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=folder&id=101
```

Asset オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=asset&assetMode=VIEW_ASSET
  &assetid=101
```

Attachment オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=attachmentview&
  attachid=101&parentObjectId=101&parentObjectType=project
```

Financial Account オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=accountdetails&accountid=101
```

請求書の明細項目オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=invoicedetails&
  invoiceid=134&line_item_id=101
```

Invoice オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=invoicedetails&invoiceid=134
```

マーケティング・オブジェクト:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=componenttabs&
  componentid=creatives &componentinstid=1234"
```

マーケティング・オブジェクト・グリッド:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=componenttabs&
  componentid=creatives &componentinstid=1234&gridid=grid"
```

マーケティング・オブジェクト・グリッド行:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=componenttabs&
  componentid=creatives &componentinstid=1234&gridid=grid&gridrowid=101"
```

プロジェクト:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=projecttabs&projectid=1234"
```

プロジェクト・グリッド:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=projecttabs&
  projectid=1234&gridid=grid"
```

プロジェクト・グリッド行:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=projecttabs&
  projectid=1234&gridid=grid &gridrowid=101"
```

プロジェクト明細項目オブジェクト:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=projecttabs&
  projectid=1234&projectlineitemid=123&projectlineitemisversionfinal=false"
```

Team オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=teamdetails&
  func=edit&teamid=100001
```

User オブジェクト:

```
http://mymachine:7001/MktOps/affiniumplan.jsp?cat=adminuserpermissions&
  func=edit&userId=101
```

ワークフロー・タスク:

```
"http://mymachine:7001/MktOps/affiniumplan.jsp?cat=projectworkflow&
  projectid=1234&taskid=5678"
```

属性マップ

AttributeMap は、属性だけを含む Map です。属性 *name* はマップ・エントリー・キーであり、属性 *values* 配列 (複数であることに注意) はマップ・エントリー値です。

AttributeMap には、次のフィールドが含まれています。

- *Name*: 属性のプログラマチック名。この名前は、発生したコンポーネント・インスタンス内部の属性にアクセスするための固有キーとなります。

注: *Name* は、GUI でユーザーに表示される表示名である必要はありません。テンプレート (プロジェクトまたはワークフロー・タスクなど) で作成されたコンポーネントでは、属性名はテンプレート・エレメント定義で指定され、固有でなければなりません。他のコンポーネントでは、通常、属性名はサーバー・サイドのコンポーネント・インスタンス (例えば、Java イントロスペクション) からプログラマチックに派生させられます。

注: 規約では、カスタム属性には、編集可能バージョンが定義されているフォームの名前が含まれます: <form_name>.<attribute_name>。

- *values*: Java オブジェクト配列。0 個以上の属性値を含みます。各値の型は同じでなければならず、Marketing Operations で定義されている属性の型と一致していなければなりません。次の Java ラッパーおよび Marketing Operations 型だけがサポートされています。
  - AssetLibraryStateEnum: AssetLibraryStateEnum 列挙型値。
  - AssetStateEnum: AssetStateEnum 列挙型値。
  - AttachmentTypeEnum: AttachmentTypeEnum 列挙型値。
  - AttributeMap: 属性を保持するマップ。
  - BudgetPeriodEnum: BudgetPeriodEnum 列挙型値。
  - BudgetTypeEnum: BudgetTypeEnum 列挙型値。
  - Handle: コンポーネント・インスタンス、グリッド行、属性などへの参照。
  - InvoiceStateEnum: InvoiceStateEnum 列挙型値。
  - java.io.File: ファイルの表現。
  - java.lang.Boolean: ブール値、True か False のいずれか
  - java.lang.Double: 倍精度 10 進数値。
  - java.lang.Float: 単精度 10 進数値
  - java.lang.Integer: 32 ビット整数値
  - java.lang.Long: 64 ビット整数値
  - java.lang.Object: 総称 Java オブジェクト
  - java.lang.String: 0 個以上の Unicode 文字のストリング
  - java.math.BigDecimal: 任意精度符号付き 10 進数値。通貨での使用に適しています。値の解釈方法は、クライアントの通貨ロケールに依存します。
  - java.math.BigInteger:任意精度整数値。
  - java.net.URL: Universal Resource Locator (URL) オブジェクト。
  - java.util.ArrayList: オブジェクトのリスト。
  - java.util.Calendar: 特定のロケールの日時値。
  - java.util.Date: 日時値。この型は非推奨です。代わりに、java.util.Calendar または java.util.GregorianCalendar を使用してください。
  - java.util.GregorianCalendar: GregorianCalendar は java.util.Calendar の具象サブクラスで、世界中のほとんどで使用されている標準カレンダー・システムを提供します。
  - MonthEnum: MonthEnum 列挙型値。
  - ProjectStateEnum: ProjectStateEnum 列挙型値。
  - QuarterEnum: QuarterEnum 列挙型値。

– TaskStateEnum: TaskStateEnum 列挙型値。
– WeekEnum: WeekEnum 列挙型値。

属性のメタデータ (ローカライズされた表示名や説明など) は、その属性や親オブジェクト・インスタンスに関連したテンプレートにより定義されます。属性は、プロジェクト名、コード、および開始日などの、必須およびオプションのオブジェクト・インスタンス属性を公開するための、単純でありながら拡張可能な仕組みを提供します。

注: 日付を実装するには、java.util.Calendar または java.util.GregorianCalendar のいずれかを選択できます。

一般的な例外

このトピックは、API によりスローされるいくつかの一般的な例外について記述します。

- AuthorizationException: クライアントの実行コンテキストに関連したユーザーに、この要求操作を実行する権限がありません。これはどの API メソッドでもスローする可能性があるので、宣言されていません。
- DataException: IBM Unica Marketing Operationsの基礎となるデータベース層で発生する例外。詳しくは、SQL ログを確認してください。
- InvalidExecutionContextException: API メソッドに渡された実行コンテキストに問題があります (例えば、メソッドが正しく初期化されていない)。これはどの API でもスローする可能性があるので、宣言されていません。
- NotLockedException: 必要なロックを最初に獲得せずに、コンポーネント・データを更新しようとしました。IExecutionContext の acquireLock() メソッドを参照してください。

API メソッド

公式の API メソッドに関する特定の情報については、JavaDoc API ドキュメンテーション・ファイルの iPlanAPI クラスを参照してください。これらのファイルは、Marketing Operations にログインし、任意のページから **「ヘルプ」>「製品資料」**を選択して、 *<version>*PublicAPI.zip をダウンロードすることによって参照できます。

IBM Unica 技術サポートへの連絡

ドキュメンテーションを参照しても解決できない問題があるなら、指定されているサポート窓口を通じて IBM Unica 技術サポートに電話することができます。このセクションの情報を使用するなら、首尾よく効率的に問題を解決することができます。

サポート窓口が指定されていない場合は、IBM Unica 管理者にお問い合わせください。

収集する情報

IBM Unica テクニカル・サポートに連絡する前に、次の情報を収集しておいてください。

- 問題の性質の要旨。
- 問題発生時に表示されるエラー・メッセージの詳細な記録。
- 問題を再現するための詳しい手順。
- 関連するログ・ファイル、セッション・ファイル、構成ファイル、およびデータ・ファイル。
- 「システム情報」の説明に従って入手した製品およびシステム環境に関する情報。

システム情報

IBM Unica 技術サポートに電話すると、実際の環境に関する情報について尋ねられることがあります。

問題が発生してもログインは可能である場合、情報の大部分は「バージョン情報」ページで入手できます。そのページには、インストールされている IBM Unica のアプリケーションに関する情報が表示されます。

「バージョン情報」ページは、**「ヘルプ」>「バージョン情報」**を選択することにより表示できます。「バージョン情報」ページを表示できない場合、どの IBM Unica アプリケーションについても、そのインストール・ディレクトリーの下にある `version.txt` ファイルを表示することにより、各アプリケーションのバージョン番号を入手できます。

IBM Unica 技術サポートの連絡先情報

IBM Unica 技術サポートとの連絡を取る方法については、IBM Unica 製品技術サポートの Web サイト (http://www.unica.com/about/product-technical-support.htm) を参照してください。

© Copyright IBM Corp. 2002, 2012

特記事項

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。

© Copyright IBM Corp. 2002, 2012

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
170 Tracer Lane
Waltham, MA 02451
U.S.A.

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。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

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。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

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。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴ、および ibm.com は、世界の多くの国で登録された International Business Machines Corp. 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、『www.ibm.com/legal/copytrade.shtml』 をご覧ください。

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21