IBM Unica Marketing Operations

V8R6

2012 年 5 月 25 日

集成模块

注意

在使用本信息及其支持的产品前，请先阅读第 29 页的『声明』中的信息。

此版本适用于 IBM Unica Marketing Operations V8.6.0 及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

第 1 章 什么是 Marketing Operations Integration Services?

Marketing Operations Integration Services 是以下对象的组合体。

- **Marketing Operations 集成 Webservice**

  Integration Services 提供了一种方式，使 IBM® Unica® Marketing Operations 客户、合作伙伴和 IBM Professional Services 可以将 Marketing Operations 与环境中运行的其他应用程序集成。

- **Marketing Operations 过程和 API**

  可以在 Marketing Operations 中定义定制过程，以通过任意方式扩展 Marketing Operations 业务逻辑。定义后，这些过程就可以作为从其他应用程序进行 Integration Services Webservice 调用的目标。还可以定义过程来将消息发送至其他应用程序。

- **Marketing Operations 触发器**

  可以将触发器与 Marketing Operations 中的事件和过程关联。发生某个此类事件时，将执行关联的触发器。

Marketing Operations Integration Services 的要求有哪些?

Marketing Operations Integration Services 必须满足以下要求:

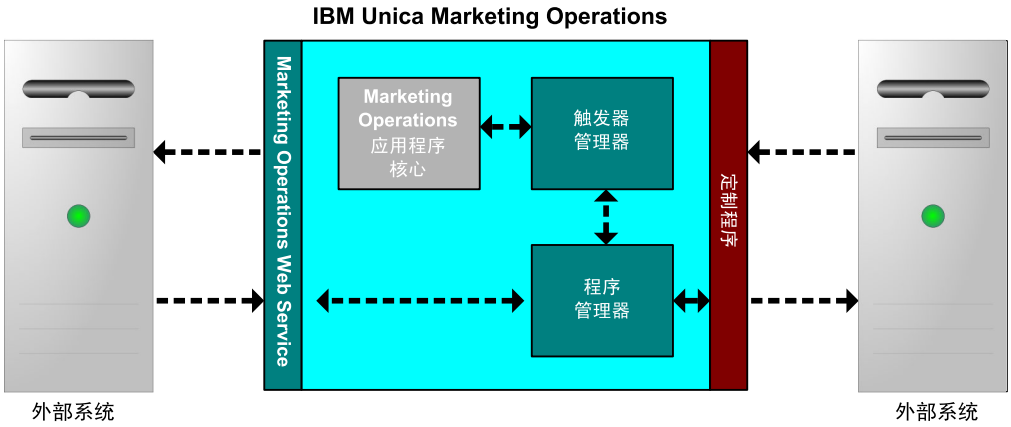
- 以松散方式进行系统集成的耦合。
- 提供一种机制，客户机应用程序可使用此机制通过 Webservice 调用来影响 Marketing Operations。
- 提供一种机制，可使用此机制在 Marketing Operations 中向客户应用程序通知某些事件。
- 提供易于理解和使用的简单编程模型。
- 具有强大的故障恢复功能
- 保证数据完整性。
- 与基于 GUI 的现有 Marketing Operations 客户集成，并将对这些客户的影响降至最低。
- 对 Marketing Operations 组件提供细颗粒度的访问，同时使编程人员免于接触底层实现详细信息。

IBM Unica Marketing Operations Integration Services 入门

IBM Unica Marketing Operations Integration Services 功能用于创建定制过程。当 Marketing Operations 中发生特定事件时，可以使用这些过程来触发外部事件。可以使用这些过程来从外部系统或程序执行 Marketing Operations 功能。

使用 API 与 IBM Unica Marketing Operations 在编程级别上连接，正如使用 GUI 与 Marketing Operations 在用户级别上连接一样。使用 API 可以构建过程。可以使用这些过程在 Marketing Operations 与外部系统之间通信。Marketing Operations Webservice 是过程、API 和触发器的容器对象。

以下显示了 Marketing Operations Integration Services 的体系结构。

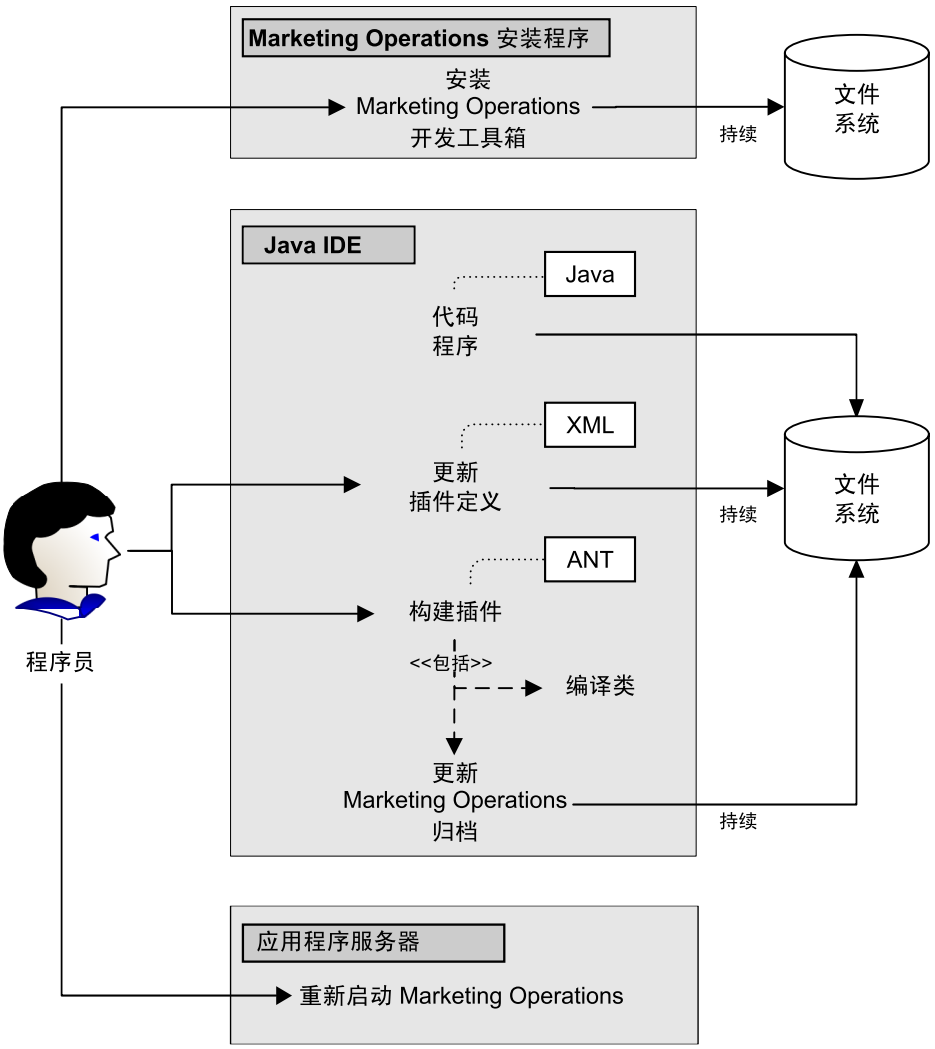


以下是 Integration Services 的主要组件。

- Marketing Operations 过程管理器: 通过借助 API 与 Marketing Operations 交互，扩展业务逻辑。
- Marketing Operations 触发器管理器: 将条件（例如市场营销对象的状态更改）与操作（当满足触发器的条件时要执行的过程）关联。

方法

使用 IBM Unica Marketing Operations Integration Services 的组件来开发定制过程，如下图中所示:



使用 Marketing Operations 安装程序安装开发者工具包后，请遵循以下基本步骤:

1. 对定制过程进行编码。当前必须使用 Java™。
2. 更新 XML 定义文件中的插件定义。
3. 构建插件:
   a. 编译必要的类。
   b. 更新 Marketing Operations 归档（WAR 文件）。
4. 重新启动 Marketing Operations。

在 IBM Unica Marketing Operations 与 API 之间通信的基本示例

本节描述在 API 与 Marketing Operations 之间建立通信的基本示例。它并不执行任何有用的工作；它将执行 Marketing Operations 与 Integration Services 之间的往返通信。

本节使用 Marketing Operations Integration Services 开发者工具包随附的示例过程其中的部分。即，可以在以下文件中找到此处引用的代码。

- PlanClientFacade.java
- PlanWSNOOPTestCase.java

noop 方法是对 Marketing Operations 的 Webservice 调用。它在 PlanClientFacade 类中定义，并在数组中传递空值。

```
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);
}
```

过程 testExecuteProcedure 从 PlanClientFacade 调用 noop 方法，以建立与 Marketing Operations 应用程序的往返通信。

```
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.");
}
```

有关 NameValueArrays、ProcedureResponse 以及其他列出的方法和数据类型的详细信息，请参阅本指南其余部分中的特定章节和 Javadoc。

托管的 Javadoc

有关公共 API 方法的特定信息，请参阅 JavaDocs API 文档文件中的 iPlanAPI 类。通过登录到 Marketing Operations 并可从任何页面选择**帮助> 产品文档**，然后下载 <V>PublicAPI.zip 文件来获取这些文件。

第 2 章 关于 Marketing Operations 集成 Webservice

Webservice 提供 Marketing Operations Integration Services 的客户机视图，而 Integration Services 是 IBM Unica Marketing Operations 服务器部署的一部分。该服务旨在供 Marketing Operations Web 用户并发使用。

Webservice 提供一个 API 调用，即 executeProcedure。

客户机直接进行此 Webservice 调用。

认证

无需认证；所有客户机都与名为 PlanAPIUse 的已知 IBM Unica Marketing Operations 用户关联。我们假定此特殊用户的安全功能由 Marketing Operations 系统管理员配置，以满足所有 Webservice 客户机的需求。

该服务的将来版本可能会提供一种更为通用的机制，以进行安全客户机认证。

语言环境

支持的唯一语言环境是当前为 IBM Unica Marketing Operations 系统实例配置的语言环境。假定可通过该服务访问的所有依赖于语言环境的数据（消息、货币等）都使用此系统语言环境。

该服务的将来版本可能会提供一种机制，以供客户机告知 Marketing Operations 应使用哪种语言环境。

状态管理

该 Webservice *无状态*；即跨 API 调用的服务实现不会保存每个客户机的信息。次能提供更有效率的服务实现，并且简化了集群支持。

数据库事务

该 Webservice 不显示数据库事务，也不编辑对客户机的锁定。但是，它确实保证了所有过程执行的作用都将采用*原子方式*。此结果表示过程将会成功或失败；如果失败，数据库就仍处于与从未调用过 API 一样的状态。

关于 Marketing Operations 集成 Webservice 数据类型

本节定义 Webservice 使用的数据类型，它们与特定服务绑定或编程实现无关。

将使用以下表示法。

- <*type*>: <*type definition*> 定义简单的数据类型。例如:

  Handle: string

- <*type*>: [<*type definition*>] 定义复杂的数据类型或数据结构。
- <*type*>: { <*type definition*> } 定义复杂的数据类型或数据结构。

复杂的类型元素和 API 参数可以使用这些类型来声明数组。例如:

```
Handle [] handles
```

类型 handles 是 Handle 类型的数组。

基本类型

基本类型仅限于下表中定义的类型，用于简化 SOAP 1.1 绑定的支持。可以将所有类型声明为数组，例如 **String [ ]**。实质上，二进制数据类型（如 **long**）可以由协议绑定（例如 SOAP）表示为字符串。但是，此表示对于客户机看到的类型语义、许可的值等没有作用。

表 1. 基本类型

API 类型	描述	SOAP 类型	Java 类型
boolean	布尔值: **true** 或 **false**	xsd:boolean	boolean
dateTime	日期时间值	xsd:datetime	Date
decimal	任意精度十进制值	xsd:decimal	java.math.BigDecimal
double	带有符号的双精度十进制值	xsd:double	double
int	带有符号的 32 位整数值	xsd:int	int
integer	带有符号的任意精度整数值	xsd:integer	java.math.BigInteger
long	带有符号的 64 位整数值	xsd:long	long
字符串	Unicode 字符的字符串	xsd:string	java.lang.String

MessageTypeEnum

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

MessageTypeEnum 是定义所有可能消息类型的枚举类型。

- INFORMATION: 参考消息
- WARNING: 警告消息
- ERROR: 错误消息

消息

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

Message 是定义 Webservice API 调用结果的数据结构。它提供非本地化代码、本地化文本和日志详细信息的可选字段。当前，所有本地化文本都使用为 IBM Unica Marketing Operations 服务器实例设置的语言环境。

表 2. 消息参数

参数	描述
type	MessageTypeEnum，设置消息的类型。
code	消息的可选代码，采用字符串格式。
localizedText	用于与消息关联的可选文本字符串。
logDetail	可选堆栈跟踪消息。

NameValue

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

NameValue 是定义名称/值对的基本复杂类型。它还定义服务用于根据需要构造值数组的可选序列（序列基于零）。

名称相同但序号不同的所有 NameValue 都将转换为值的数组，并与公共名称关联。

数组大小由最大序号确定；未指定的数组元素具有空值。数组序号必须唯一。值和值的类型由扩展类型提供。

表 3. NameValue 参数

参数	描述
name	定义 NameValue 类型的名称的字符串。
顺序	基于零的整数，它设置 NameValue 暗含值的序号。

为每个基本类型定义了扩展 NameValue 类型，如下所示:

表 4. 扩展 NameValue 类型

扩展类型	描述
BigDecimalNameValue: NameValue [decimal value]	值为任意精度十进制数字的 NameValue 类型。
BigIntegerNameValue: NameValue [integer value]	值为任意大小整数的 NameValue 类型。
BooleanNameValue: NameValue [boolean value]	值为布尔值的 NameValue 类型。
CurrencyNameValue: NameValue [string locale, decimal value]	适用于表示某些语言环境中的货币的 NameValue 类型。语言环境是 ISO 语言代码，即 ISO-639 定义的两个小写字母的代码。 当前，语言环境必须与 IBM Unica Marketing Operations 服务器实例中设置的语言环境一致。
DateNameValue: NameValue [datetime value]	值为日期的 NameValue 类型。
DecimalNameValue: NameValue [double value]	值为双精度十进制数字的 NameValue 类型。
IntegerNameValue: NameValue [long value]	值为 64 位整数的 NameValue 类型。
String NameValue: NameValue [string value]	值为字符串的 NameValue 类型。

最终，将定义扩展 NameValue 类型的数组，以供在需要定义不同类型 NameValue 集合时使用。

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

executeProcedure

语法

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

返回值

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

描述

此方法使用可选的参数数组调用指定过程。调用将以同步方式执行；即，它将阻止客户机并在完成时返回结果。

参数

表 5. executeProcedure 参数

名称	描述
key	要执行的过程的唯一键。如果没有为 **key** 绑定任何过程，那么将返回 *RemoteException* 错误。
jobid	可选字符串，用于标识与此过程执行关联的作业。此字符串是传递项目，但它可用于将客户机任务与特定过程的执行关联在一起。
paramArray	要传递至过程的参数数组。如果一个或多个参数无效（错误类型、非法值等），那么将返回错误状态和消息。将由客户机来确定参数、参数类型以及过程所需值的数目。

返回参数

表 6. executeProcedure 返回参数

名称	描述
status	整数代码: • 0 指示过程已成功执行 • 整数指示错误 过程可以使用状态来指示不同级别的错误。
messages	零到多个消息数据结构的数组。如果 **status** 为 0，那么此数组不包含 ERROR 消息，但可以包含 INFORMATION 和 WARNING 消息。 如果 **status** 非零，那么 messages 可以包含 ERROR、INFORMATION 和 WARNING 消息的任何组合。

Marketing Operations Integration Services WSDL

本主题定义 Marketing Operations Integration Services 的 Web service 定义语言 (WSDL)。WSDL 通过手动定义，并且是关于 Webservice 定义的最终表述。

Axis

此版本的 Webservice 使用 Axis2 1.5.2 来通过 WSDL 文件生成构成 Web service 实现的服务器端类。用户可以通过提供的 WSDL，随意使用任何版本的 Axis 或任何其他

非 Axis 技术来创建客户机端实现，以与 API 集成。

协议版本

协议的版本与 WSDL 显式绑定，如下所示:

- 作为 WSDL 名称的一部分，例如，`PlanIntegrationService1.0.wsdl`
- 作为 WSDL targetNamespace 的一部分，例如，`xmlns:tns="http://webservices.unica.com /MktOps/services/PlanIntegrationServices1.0?wsdl"`

WSDL

IBM Unica Marketing Operations Integration Services 随附一个 WSDL 文件: `PlanIntegrationServices1.0.wsdl`。该 WSDL 在 `integration/examples/soap/plan` 目录中提供。示例构建脚本使用此文件来生成相应的客户机端存根以连接至 Webservice。

第 3 章 IBM Unica Marketing Operations 过程

*过程*是 IBM Unica Marketing Operations 托管的定制或标准 Java 类，它执行某些工作单元。过程提供了一种方式，使客户、合作伙伴和 IBM Unica Professional Services 可以通过任意方式扩展 Marketing Operations 业务逻辑。

过程遵循简单的编程模型，它使用正确定义的 API 来影响 Marketing Operations 管理的组件。通过简单的查找机制和基于 XML 的定义文件来“发现”过程。Marketing Operations 根据过程的“客户机”的需求来执行这些过程。例如，对集成请求（入局）或触发器触发（内部或出局）进行响应。

过程与其客户机以同步方式运行；结果通过持久的审计机制直接提供给客户机。过程的执行可能还会导致在 Marketing Operations 中触发其他事件和触发器。

必须使用 Java 编写过程。

假设

请注意以下有关过程的假设。

- 过程实现类封装在单独的类树或 JAR 文件中，并通过 URL 路径提供给 IBM Unica Marketing Operations。过程执行管理器使用独立的类装入器来根据需要装入这些类。缺省情况下，Marketing Operations 在以下目录中查找:

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

  要更改此缺省值，请对**设置 > 配置 > Marketing Operations > umoConfiguration > integrationServices** 下的 **integrationProcedureClasspathURL** 参数进行设置。
- 过程实现类名遵循公认的 Java 命名约定，以避免与“unica”和其他供应商提供的类发生程序包冲突。特别是客户不得将过程置于 **com.unica** 或 **com.unicacorp** 程序包树下。
- 已使用应用程序服务器上的 IBM Unica Marketing Operations 所使用的 Java 运行时版本（至少为 JRE 1.5.10）来对过程实现进行编码。
- IBM Unica Marketing Operations 提供一些开放式源代码和第三方库；应用程序服务器也使用这些库的不同版本。通常，此列表随发行版的不同而更改。

  注: 为避免出现可能的兼容性问题，过程不应使用任何特定于开放式源代码、第三方或应用程序服务器的库。

  但是，如果过程或过程导入的次级类使用此类程序包，那么这些包的使用必须与 Marketing Operations 和/或应用程序提供的包完全一致。请注意，在这种情况下，如果 Marketing Operations 的将来版本对库进行升级或放弃，那么您可能需要重新编写过程代码。
- 过程实现类由 IBM Unica Marketing Operations 通常使用的类装入策略（通常是 **父类后装入**）来装入。应用程序服务器可能会提供开发工具和选项以重新装入适用于 Marketing Operations 过程的类，但此行为并非必需。

- 对于过程自己的状态，过程必须是线程安全的；即它的 execute 方法不能依赖于不同调用间的内部状态更改。
- 过程不能自己创建线程。

设计

设计应着重于生成必须以原子方式执行的单个工作单元。在理想情况下，过程执行一系列任务，可通过异步方式将这些任务调度为在将来某时执行；此“一劳永逸”的集成模型导致两个系统上都存在最小的负载。

过程实现类使用 IBM Unica Marketing Operations API 来读取和更新 Marketing Operations 组件、调用服务等等。其他 Java 包可用于执行其他任务。

注: 在 Marketing Operations 的将来发行版中，将仅支持文档化的类和方法。应将 Marketing Operations 中的所有其他类和方法都视为禁止使用。

对过程实现类进行编码和编译后，就必须将这些类提供给 Marketing Operations。随 Marketing Operations Integration Services 提供的构建脚本将编译后的过程置于缺省位置。最后的开发步骤是更新 Marketing Operations 使用的定制过程插件定义文件，以发现定制过程。

过程必须实现 **com.unica.publicapi.plan.plugin.procedure.IProcedure** 接口，并必须具有无参数的构造方法（常见的 JavaBean 模型）。每个过程的编码和编译将在客户选择的 Java IDE（如 Eclipse、Borland JBuilder、Idea 等）中完成。IBM Unica Marketing Operations 随附了样本代码作为开发者工具包，它们位于以下位置。

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

配置

使用**设置 > 配置 > Marketing Operations > umoConfiguration > integrationServices** 下的参数来配置 Marketing Operations 集成模块。

有关详细信息，请参阅《*Marketing Operations 安装指南*》。

过程生命周期

过程的运行时生命周期是:

1. 发现和初始化
2. 选择（可选）
3. 执行
4. 销毁

发现和初始化

必须使 IBM Unica Marketing Operations 了解可用于特定安装实例的所有标准和定制过程。此过程称为发现。

注: 标准过程（由 Marketing Operations 工程团队定义的过程）是隐式已知的，因此无需任何操作就可以发现这些过程。

定制过程在过程插件定义文件中定义。Marketing Operations 插件管理器在初始化期间读取此文件。对于找到的每个过程，插件管理器将执行以下操作:

1. 对过程进行实例化；将其状态转换为 INSTANTIATED。
2. 创建新的过程审计记录。
3. 如果可以对过程进行实例化，那么将使用其插件描述文件中找到的任何初始化参数来调用其 **initialize()** 方法。如果此方法抛出异常，那么将记录状态，并且会放弃该过程。否则，该过程的状态将转换为 INITIALIZED。现在该过程已准备就绪可以执行。
4. 创建新的过程审计记录。
5. 如果可以对过程进行初始化，那么将调用其 **getKey** 方法来确定客户机用于引用该过程的键。会将此键与实例关联，并保存以供将来查找。

选择

有时，IBM Unica Marketing Operations 可能需要向用户提供可用过程的列表，例如使管理员可以设置触发器。只有在对过程进行初始化后才能完成此操作。过程的 **getDisplayName()** 和 **getDescription()** 方法用于此目的。

执行

在已对过程进行初始化后的某个时刻，IBM Unica Marketing Operations 将收到执行该过程的请求。这可能会与其他线程上执行的其他过程（或同一过程）同时发生。

在执行时，过程执行管理器将执行以下操作:

1. 启动数据库事务。
2. 将过程状态设置为 EXECUTING。
3. 创建新的过程审计记录。
4. 使用执行上下文和由客户机提供的任何执行参数来调用过程的 **execute()** 方法。方法实现将根据需要使用 Marketing Operations API，同时获取编辑锁定并沿执行上下文方向传递。如果 execute 方法抛出异常，那么执行管理器将标记事务以进行回滚。
5. 根据执行结果落实或回滚事务；将过程状态设置为 EXECUTED。
6. 释放任何未决的编辑锁定。
7. 创建新的过程审计记录。

注: **execute()** 方法不应改变过程实例数据。

销毁

当 IBM Unica Marketing Operations 关闭时，过程插件管理器将遍历所有装入的过程。对于找到的每个过程，它将执行以下操作:

1. 调用过程的 destroy() 方法，以使过程可在销毁实例之前进行清理。
2. 将过程的状态更改为 FINALIZED（无法执行该过程）。
3. 创建新的过程审计记录。
4. 销毁过程的实例。

数据锁定

IBM Unica Marketing Operations 使用保守式编辑锁定方案；即，每次只向一个用户授予组件实例的更新访问权。对于 GUI 用户，此锁定在可视选项卡级别上完成。在某些情况下，这表示实例的一部分（例如，项目摘要选项卡），而在其他情况下，它表示许多实例（工作流程选项卡）。一旦用户获取锁定，所有其他用户对于相关数据都只能进行只读访问。

注: 编辑锁定与数据库事务不同。

为确保其他用户不会在无意中覆盖过程对组件实例或实例组进行的更改，过程在更新组件数据之前必须获取相应的锁定。将使用传递至过程 **execute()** 方法的执行上下文对象来实现此目的。

在过程更新任何数据之前，它必须为所需的每个锁定调用上下文的 **acquireLock()** 方法。例如，如果过程将更新项目和关联的工作流程，那么该过程需要获取这两者的锁定。

如果其他用户已拥有锁定，那么 **acquireLock()** 方法将立即抛出 **LockInUseException**。为尽量减少冲突，过程应在更新对象后立即释放锁定。

当 execute 方法返回时，执行管理器将自动释放所有未决的锁定。在任何情况下，仅在数据库事务的生存期中保持锁定。这就是说，如果超出了特定于数据库的事务超时值，那么锁定将到期。

过程事务

过程执行管理器自动使用数据库事务对过程的执行进行打包，根据过程执行的结果进行相应的落实或回滚。这就保证了在落实之前其他用户不会看到对 IBM Unica Marketing Operations 数据库进行的更新，并保证了更新以原子方式进行。

过程写程序仍然必须获取必要的编辑锁定，以确保在过程执行完成之前其他用户无法向数据库写入更改。

传递结果

过程的 **execute()** 方法返回整数状态码以及零到多条消息，会将这些信息记录并持久保存在 IBM Unica Marketing Operations 过程审计表中。客户机也可以使用某些其他的方式来传递状态信息。

过程日志记录

IBM Unica Marketing Operations 具有用于过程的单独日志文件。

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

过程执行管理器将记录每个过程的生命周期并创建审计记录。

- **logInfo()**：将参考消息写入到过程日志
- **logWarning()**：将警告消息写入到过程日志
- **logError**：将错误消息写入到过程日志

- **logException()**：将异常的堆栈跟踪转储到过程日志

关键 Java 类

提供的集成开发者工具包包含公用 IBM Unica Marketing Operations API 和支持类的一组 Javadoc。此处列出了最重要的一些 Javadoc。

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure)：所有过程必须实现的接口。过程经历正确定义的生命周期，并访问 Marketing Operations API 以执行工作。
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure)：所有触发器过程必须实现的接口（标记接口）。
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext)：由执行管理器传递至过程的不透明上下文对象的接口。此对象具有用于记录和编辑锁定管理的公用方法。过程还会将此对象传递至所有 PlanAPI 调用。
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI)：Marketing Operations API 的接口。执行上下文提供 **getPlanAPI()** 方法以检索正确的实现。

过程示例

本示例显示用于通过集成 Webservice 或触发器更改项目状态的标准过程。

注：请勿修改样本过程及其 XML 定义，因为当您升级 IBM Unica Marketing Operations 时将覆盖这些样本，从而会丢失更改。应在其他目录中创建和修改所有定制过程。

```
// 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
    }
}
```

过程插件定义文件

本主题描述过程插件定义文件。此文件定义有关要在 IBM Unica Marketing Operations 中托管的定制过程的实现类、元数据和其他信息。缺省情况下，假定过程插件定义位于以下路径中:

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

此文件是包含以下所述信息的 XML 文档。

Procedures: 零到多个 **Procedure** 元素的列表。

Procedure: 定义过程的元素。每个过程包含以下元素。

- **key**（可选）: 定义过程的查找键的字符串。对于特定 Marketing Operations 实例托管的所有标准（IBM 提供的）过程和定制过程，此键必须唯一。如果未定义，那么将缺省为 **className** 元素的标准版本。以字符串“uap”开头的名称保留供 IBM Unica Marketing Operations 使用。
- **className**（必需）: 过程类的标准程序包名。此类必须实现 IProcedure 类 (com.unica.public.plan.plugin.procedure.IProcedure)。
- **initParameters**（可选）: 零到多个 initParameter 元素的列表。

  initParameter（可选）: 要传递至过程的 initialize() 方法的参数。此元素包括嵌套的参数 name、type 和 value 元素。
  - name: 定义参数名称的字符串
  - type: 定义参数值类型的 Java 包装器类的可选类名。必须是以下某个类型:
    - java.lang.String（缺省值）
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  - value: 根据类型而定的属性值的字符串格式

第 4 章 关于 IBM Unica Marketing Operations API

IBM Unica Marketing Operations API 是一种外观，它提供正在运行的 Marketing Operations 实例的客户机视图。将仅显示部分 Marketing Operations 功能。该 API 旨在供 Marketing Operations Web 用户与 Marketing Operations Integration Services WebService SOAP 请求和触发器并发使用。该 API 支持以下类型的操作:

- 组件创建和删除
- 发现（按组件类型、属性值等）
- 组件检查（通过其属性、专用链接等）
- 组件修改

版本控制和向后兼容性

此 API 的将来版本将与共享同一主版本号的所有次要版本和维护版向后兼容。但是，对于点零 (x.0) 主要版本的较早版本，如果有业务理由或技术理由证明必要，那么 IBM 将保留中断与该版本兼容性的权限。

如果进行了任何以下更改，那么此 API 的主要版本号将递增。

- 更改了数据解释
- 更改了业务逻辑（例如更改了服务方法功能）
- 更改了方法参数和/或返回类型

如果进行了任何以下更改，那么此 API 的次要版本号将递增（请注意，根据定义，这些更改具有向后兼容性）。

- 添加了新方法
- 添加了新数据类型并且其用途限于某个新方法
- 向枚举类型添加了新元素
- 使用版本后缀定义了接口的新版本

用户安全性

假定认证由过程的执行管理器完成，而与执行上下文绑定的已认证用户信息由所有 API 使用。API 不会显示已认证的用户，而是将其传递到 IBM Unica Marketing Operations 以便在需要时使用。

然而，已认证的用户可能无权执行 API 显示的所有操作；在这种情况下，API 方法将抛出 **AuthorizationException**。

语言环境

此版本支持的唯一语言环境是当前为 IBM Unica Marketing Operations 服务器实例配置的语言环境。假定可通过 API 访问的所有依赖于语言环境的数据（消息、货币等）都使用此系统语言环境。

状态管理

此 API 无状态，即 API 交叉调用将不会保存每个客户机的信息。

然而，请注意，特定 API 调用可以更改 IBM Unica Marketing Operations 管理的底层组件实例的状态，并且这些状态更改可对于数据库具有持久性。

数据库事务

此 API 不会向客户机显示数据库事务，但如果执行上下文中包含此类信息，那么此 API 将使用这些信息。如果已启动事务，那么特定过程中所有 API 调用的作用将采用原子方式。在过程成功落实事务之前，IBM Unica Marketing Operations 的其他用户将不会看到更改。

用来更新数据库的 API 调用必须首先获取编辑锁定，以防止其他 Marketing Operations 用户在该 API 调用过程中修改底层数据。在该 API 完成之前，其他用户将无法更新锁定的组件；与此类似，其他 Marketing Operations 用户或 API 客户机可能已获取所需数据上的锁定，从而使该 API 调用无法完成。

事件处理

通过此 API 在 IBM Unica Marketing Operations 组件上执行的操作将如同操作由 Marketing Operations Web 用户执行一样生成相同的事件。具体而言，等待特定事件的触发器最终将在两种情况下都进行触发。订阅特定通知（例如，项目状态更改时）的用户将收到由 API 调用以及 Web 用户操作而引起的状态更改通知。

关于 API 数据类型

IBM Unica Marketing Operations API 包含以下公用数据类型。

- **ApprovalMethodEnum**：核准方法的枚举类型。
- **ApprovalStateEnum**：核准状态的枚举类型。
- **AssetLibraryStateEnum**：资产库状态的枚举类型。
- **AssetStateEnum**：资产状态的枚举类型。
- **AttachmentTypeEnum**：附件的枚举类型。
- **AttributeMap**：包含属性及其值的 Java 映射。
- **BudgetPeriodEnum**：预算期类型的枚举类型。
- **BudgetTypeEnum**：预算类型的枚举类型
- **Handle**：识别特定 Marketing Operations 实例中的组件实例。
- **InvoiceStateEnum**：发票状态的枚举类型。
- **MonthEnum**：日历年中月份的枚举类型
- **ProjectCopyTypeEnum**：副本项目选项的枚举类型。
- **ProjectStateEnum**：项目状态的枚举类型。
- **TaskStateEnum**：任务状态的枚举类型。
- **QuarterEnum**：日历年中季度的枚举类型
- **WeekEnum**：日历年中星期的枚举类型

下文说明了每种数据类型的可能值。

枚举类型

ApprovalMethodEnum

ApprovalMethodEnum 是定义核准的所有可能方法的枚举类型。可能的值有:

- SIMULTANEOUS
- SEQUENTIAL

ApprovalStateEnum

ApprovalStateEnum 是定义核准的所有可能状态的枚举类型。可能的值有:

- NOT_STATED
- ON_HOLD
- IN_PROGRESS
- 已完成
- CANCELLED

AssetLibraryStateEnum

AssetLibraryStateEnum 是定义资产库的所有可能状态的枚举类型。可能的值有:

- ENABLED
- DISABLED

AssetStateEnum

AssetStateEnum 是定义资产的所有可能状态的枚举类型。可能的值有:

- 草稿
- LOCK
- FINALIZE
- ARCHIVE

AttachmentTypeEnum

AttachmentTypeEnum 是定义附件的所有可能状态的枚举类型。可能的值有:

- URL
- FILE
- ASSET

BudgetPeriodEnum

BudgetPeriodEnum 是定义所有可能预算期的枚举类型。可能的值有:

- QUARTERLY
- MONTHLY
- WEEKLY
- YEARLY
- ALL

BudgetTypeEnum

BudgetTypeEnum 是定义所有可能预算类型的枚举类型。可能的值有:

- TOTAL
- FORECAST
- COMMITTED
- ACTUAL
- ALLOCATED

InvoiceStateEnum

InvoiceStateEnum 是定义发票的所有可能状态的枚举类型。可能的值有:

- 草稿
- PAYABLE
- PAID
- CANCELLED

MonthEnum

MonthEnum 是定义月份的所有可能值的枚举类型。

ProjectCopyTypeEnum

ProjectCopyTypeEnum 是定义副本项目的所有可能类型的枚举类型。可能的值有:

- COPY_USING_PROJECT_METRICS
- COPY_USING_TEMMPLATE_METRICS

有关这些项目和任务状态的详细信息，请参阅《*Marketing Operations 用户指南*》。

ProjectStateEnum

ProjectStateEnum 是定义项目或请求的所有可能状态的枚举类型。可能的值有:

- NOT_STARTED
- IN_PROGRESS
- ON_HOLD
- IN_RECONCILIATION
- LATE: 指示项目未在其调度开始日期之前启动。
- OVERDUE: 指示项目未在其调度结束日期之前完成。
- 草稿
- SUBMITTED
- RETURNED
- ACCEPTED
- 已完成
- CANCELLED
- DELETED

TaskStateEnum

TaskStateEnum 是定义工作流程任务的所有可能状态的枚举类型。可能的值有:

- PENDING
- ACTIVE
- FINISHED
- SKIPPED
- DISABLED

QuarterEnum

QuarterEnum 是定义季度的所有可能值的枚举类型。

WeekEnum

WeekEnum 是定义星期的所有可能值的枚举类型。

句柄

句柄是引用特定 Marketing Operations 实例中某个组件实例的特殊 URL 对象。句柄包括组件类型、内部数据标识和实例基本 URL 等。可以对 API 使用或生成的句柄进行外部化，以使其成为可用于浏览至 Marketing Operations GUI 中组件视图的完整 URL，可以剪切并粘贴这些句柄，可以在电子邮件中发送这些句柄，以及可以将这些句柄作为参数用在其他过程中等等。

句柄仅对于特定 Marketing Operations 服务实例（或集群实例）有效，但对于已部署服务的整个生存期都保证有效。因此，可以将句柄保存在文件中以供将来引用，但无法将其用于访问其他 Marketing Operations 实例上的组件（即使这些实例位于同一台机器上）。但是，Marketing Operations 确实提供了一种将不同的基本 URL 映射至当前实例的机制，以适应将实例重新放置至其他机器（例如，当原始机器发生故障时）。

句柄独立于客户机。例如，触发器可以将句柄传递至过程，该过程将其作为参数用在对第三方系统的 SOAP 调用中，而第三方系统回转方向并向 Marketing Operations 发回 SOAP 请求，以调用更新属性的过程。

Handle 类具有从各种类型的 URL 创建句柄的工厂方法。

Approval 对象:

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

AssetLibrary 对象:

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

Asset Folder 对象:

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

Asset 对象:

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

Attachment 对象:

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

Financial Account 对象:

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

Invoice line item 对象:

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

Invoice 对象:

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

Marketing 对象:

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

Marketing 对象网格:

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

Marketing 对象网格行:

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

项目:

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

项目网格:

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

项目网格行:

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

Project line item 对象:

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

Team 对象:

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

User 对象:

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

工作流程任务:

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

属性映射

AttributeMap 是仅包含属性的映射。属性 *name* 是映射条目键，而属性 *values* 数组（请注意复数）是映射条目值。

AttributeMap 包括以下字段:

- *Name*: 属性的编程名称。此名称用作在发生映射的组件实例中访问该属性的唯一键。

  注: *Name* 不一定是 GUI 中向用户呈示的显示名。对于使用模板（如项目或工作流程任务）创建的组件，属性名由模板元素定义指定，并且必须唯一。对于其他组件，属性名通常以编程方式从服务器端组件实例派生而来（例如，通过 Java 自省）。

  注: 按照惯例，定制属性包括用于定义可编辑版本的格式的名称: <form_name>.<attribute_name>。

- *values*: Java 对象数组，它包含零到多个属性值。每个值的类型必须相同并且与 Marketing Operations 中定义的属性类型一致。仅支持以下 Java 包装器和 Marketing Operations 类型:
  - AssetLibraryStateEnum: AssetLibraryStateEnum 枚举类型值。
  - AssetStateEnum: AssetStateEnum 枚举类型值。
  - AttachmentTypeEnum: AttachmentTypeEnum 枚举类型值。
  - AttributeMap: 保存属性的映射。
  - BudgetPeriodEnum: BudgetPeriodEnum 枚举类型值。
  - BudgetTypeEnum: BudgetTypeEnum 枚举类型值。
  - Handle: 对组件实例、网格行、属性等的引用。
  - InvoiceStateEnum: InvoiceStateEnum 枚举类型值。
  - java.io.File: 文件的说明。
  - java.lang.Boolean: 布尔值，True 或 False
  - java.lang.Double: 双精度十进制数值。
  - java.lang.Float: 单精度十进制数值
  - java.lang.Integer: 32 位整数值
  - java.lang.Long: 64 位整数值
  - java.lang.Object: 类属 Java 对象
  - java.lang.String: 零到多个 Unicode 字符组成的字符串
  - java.math.BigDecimal: 带有符号的任意精度十进制数值。适用于货币；该值的解释取决于客户机的货币语言环境。
  - java.math.BigInteger: 任意精度整数值。
  - java.net.URL: 统一资源定位器 (URL) 对象。
  - java.util.ArrayList: 对象列表。
  - java.util.Calendar: 特定语言环境的日期时间值。
  - java.util.Date: 日期时间值。建议不要使用此类型。请改为使用 java.util.Calendar 或 java.util.GregorianCalendar。
  - java.util.GregorianCalendar: GregorianCalendar 是 java.util.Calendar 的具体子类，并提供世界上大部分地区使用的标准日历系统。
  - MonthEnum: MonthEnum 枚举类型值。
  - ProjectStateEnum: ProjectStateEnum 枚举类型值。
  - QuarterEnum: QuarterEnum 枚举类型值。

– TaskStateEnum: TaskStateEnum 枚举类型值。

– WeekEnum: WeekEnum 枚举类型值。

属性的元数据（如本地化的显示名和描述）由与该属性及其父对象实例关联的模板定义。属性提供了一种简单而又可扩展的机制，可用于显示必需和可选的对象实例属性，如项目名称、代码和开始日期。

注: 要实现日期，用户可以选择 java.util.Calendar 或 java.util.GregorianCalendar。

常见异常

本主题描述 API 抛出的某些常见异常。

- AuthorizationException: 与客户机的执行上下文关联的用户无权执行请求的操作。任何 API 方法都可以抛出此异常，因此未声明此异常。
- DataException: IBM Unica Marketing Operations 中的底层数据库层中发生了异常。请查看 SQL 日志以获取详细信息。
- InvalidExecutionContextException: 传递至 API 方法的执行上下文发生问题，例如，未对该方法正确初始化。任何 API 都可以抛出此异常，因此未声明此异常。
- NotLockedException: 未首先获取必需的锁定就尝试更新组件数据。请参阅 IExecutionContext 的 acquireLock() 方法。

API 方法

有关公共 API 方法的特定信息，请参阅 JavaDocs API 文档文件中的 iPlanAPI 类。通过登录到 Marketing Operations 并可从任何页面选择**帮助> 产品文档**，然后下载 <V>PublicAPI.zip 文件来获取这些文件。

联系 IBM Unica 技术支持中心

如果遇到无法通过查阅文档解决的问题，那么贵公司的指定支持联系可致电 IBM Unica 技术支持中心。请使用此部分中的信息以确保高效并成功地解决问题。

如果您不是贵公司的指定支持联系，请与 IBM Unica 管理员联系以了解相关信息。

要收集的信息

联系 IBM Unica 技术支持中心前，请收集以下信息:

- 有关问题性质的简短说明。
- 发生问题时看到的详细错误消息。
- 重现该问题的详细步骤。
- 相关的日志文件、会话文件、配置文件和数据文件。
- 关于产品和系统环境的信息，可按“系统信息”中所述获得此信息。

系统信息

致电 IBM Unica 技术支持中心时，可能会要求您提供有关系统环境的信息。

如果问题不妨碍登录，则可在“关于”页面上获得大部分此类信息，该页面提供有关所安装的 IBM Unica 应用程序的信息。

可通过选择**帮助 > 关于**访问“关于”页面。如果无法访问“关于”页面，那么通过查看位于每个应用程序的安装目录下的 `version.txt` 文件，可以获取任何 IBM Unica 应用程序的版本号。

IBM Unica 技术支持中心的联系信息

有关联系 IBM Unica 技术支持中心的方法，请参见 IBM Unica 产品技术支持中心网站: (http://www.unica.com/about/product-technical-support.htm)。

声明

本信息是为在美国提供的产品和服务而编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本资料中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: (i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系:

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

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估算的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时变更或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

所显示的所有 IBM 的价格均是 IBM 当前的建议零售价，可随时更改，而不另行通知。经销商的价格可能会有所不同。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口 (API) 进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。样本程序都是“按现状”提供的，不附有任何种类的保证。对于因使用样本程序所引起的任何损害，IBM 概不负责。

如果您正以软拷贝格式查看本信息，那么图片和彩色图例可能无法显示。

商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp., 在全球许多管辖区域内的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。当前的 IBM 商标可在 Web 站点 www.ibm.com/legal/copytrade.shtml 上『版权和商标信息』部分获取。

Printed in China