IBM Unica Marketing Operations
Version 8 Release 6
25. Mai 2012

Integrationsmoduln

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen" auf Seite 33 gelesen werden.

Diese Ausgabe bezieht sich auf Version 8, Release 6, Modifikation 0 von IBM Unica Marketing Operations und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM Unica Marketing Operations, Version 8 Release 6, Integration Moduls,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
TSC Germany
Kst. 2877
Mai 2012

Inhaltsverzeichnis

Kapitel 1. Was sind die Marketing Operations Integration Services?

Die Marketing Operations Integration Services setzen sich aus folgenden Komponenten zusammen.

- **Marketing Operations Integration Webservice**

  Die Integration Services bieten IBM® Unica Marketing Operations-Kunden, -Partnern und IBM Professional Services eine Möglichkeit, Marketing Operations mit anderen Anwendungen in ihrer Umgebung zu integrieren.
- **Marketing Operations-Prozeduren und -API**

  Benutzerdefinierte Prozeduren können in Marketing Operations definiert werden, um Marketing Operations-Geschäftslogiken beliebig zu erweitern. Diese definierten Prozeduren können die Ziele für Aufrufe des Web-Service Integration Services aus anderen Anwendungen sein. Prozeduren können auch zum Senden von Nachrichten an andere Anwendungen definiert werden.
- **Marketing Operations-Trigger**

  Trigger können Ereignissen und Prozeduren in Marketing Operations zugeordnet sein. Wenn eines solcher Ereignisse auftritt, wird der zugeordnete Trigger ausgeführt.

Welche Voraussetzungen gelten für Marketing Operations Integration Services?

Marketing Operations Integration Services müssen folgende Voraussetzungen erfüllen:

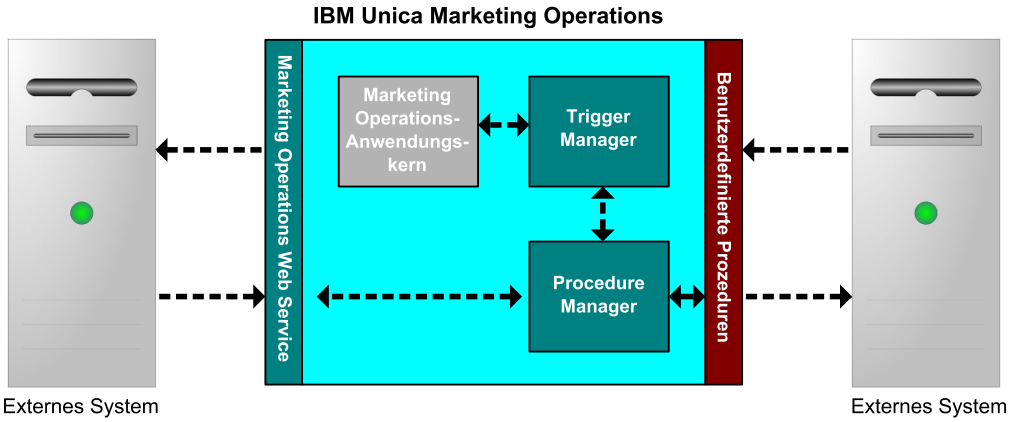
- Systemintegration flexibel verbinden.
- Einen Mechanismus für Kundenanwendungen bereitstellen, Marketing Operations über Web-Service-Aufrufe zu steuern.
- Einen Mechanismus für Kundenanwendungen bereitstellen, über bestimmte Ereignisse in Marketing Operations benachrichtigt zu werden.
- Ein einfaches Programmiermodell bereitstellen, das leicht verständlich und anwendbar ist.
- Bei der Wiederherstellung nach einer Störung stabil funktionieren.
- Datenintegrität garantieren.
- Integration mit vorhandenen grafisch orientierten Kunden von Marketing Operations bei minimalen Auswirkungen.
- Differenzierten Zugriff auf Marketing Operations-Komponenten bieten und dabei die zugrunde liegenden Implementierungsdetails vor Änderungen durch Programmierer schützen.

Einführung in IBM Unica Marketing Operations Integration Services

Mithilfe der Funktionen von IBM Unica Marketing Operations Integration Services erstellen Sie benutzerdefinierte Prozeduren. Durch diese Prozeduren können Sie externe Ereignisse auslösen, wenn bestimmte Ereignisse innerhalb von Marketing Operations auftreten. Mit diesen Prozeduren können Sie Marketing Operations-Funktionen aus externen Systemen oder Programmen ausführen.

Sie verwenden die API als Schnittstelle zu IBM Unica Marketing Operations auf Programmebene, vergleichbar mit der grafischen Benutzeroberfläche von Marketing Operations für den Benutzer. Das Erstellen von Prozeduren erfolgt unter Verwendung der API. Mithilfe dieser Prozeduren ermöglichen Sie die Kommunikation zwischen Marketing Operations und externen Systemen. Der Web-Service Marketing Operations ist ein Containerobjekt für Prozeduren, API und Trigger.

Die Architektur der Marketing Operations Integration Services wird hier dargestellt.

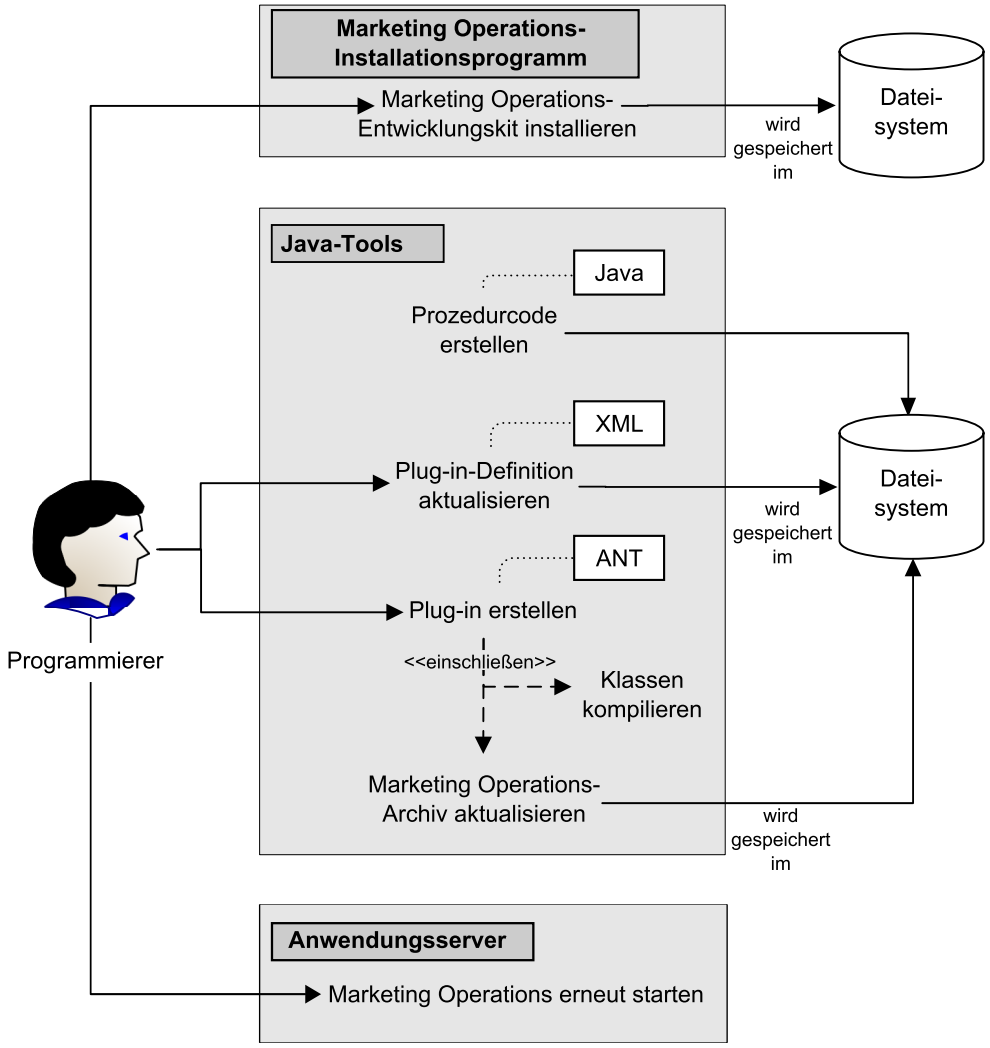


Die folgenden sind Schlüsselkomponenten der Integration Services.

- Marketing Operations Procedure Manager: erweitert die Geschäftslogik durch Interaktion mit Marketing Operations über die API.
- Marketing Operations Trigger Manager: ordnet eine Bedingung (zum Beispiel die Statusänderung eines Marketing-Objekts) einer Aktion zu (eine Prozedur, die ausgeführt wird, wenn die Bedingung für den Trigger erfüllt ist).

Methodik

Die Komponenten von IBM Unica Marketing Operations Integration Services werden wie im folgenden Diagramm dargestellt zur Entwicklung von benutzerdefinierten Prozeduren verwendet:



Nach der Installation des Developer Kits mithilfe des Marketing Operations-Installationsprogramms, führen Sie folgende grundlegende Schritte aus:

1. Geben Sie den Code für die benutzerdefinierte Prozedur ein. Derzeit müssen Sie hierzu Java verwenden.
2. Aktualisieren Sie die Plug-in-Definition in der XML-Definitionsdatei.
3. Erstellen Sie das Plug-in:
   a. Kompilieren Sie die erforderlichen Klassen.
   b. Aktualisieren Sie das Marketing Operations-Archiv (die WAR-Datei).
4. Starten Sie Marketing Operations erneut.

Einfaches Beispiel für die Kommunikation zwischen IBM Unica Marketing Operations und der API

In diesem Abschnitt wird ein einfaches Beispiel für die Einrichtung der Kommunikation zwischen der API und Marketing Operations beschrieben. Die Kommunikation in diesem Beispiel erfüllt keinen sinnvollen Zweck, es findet lediglich ein Umlauf zwischen Marketing Operations und den Integration Services statt.

In diesem Abschnitt werden Teile der Beispielprozeduren verwendet, die im Developer Kit für Marketing Operations Integration Services enthalten sind. Sie finden die hier genannten Code-Beispiele in den folgenden Dateien.

- PlanClientFacade.java
- PlanWSNOOPTestCase.java

Die noop-Methode ist ein Web-Service-Aufruf an Marketing Operations. Sie ist in der PlanClientFacade-Klasse definiert, und übergibt Nullwerte in einem Array.

```
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);
}
```

Die Prozedur "testExecuteProcedure" ruft die noop-Methode aus "PlanClientFacade" auf, um einen Umlauf mit der Anwendung Marketing Operations einzurichten.

```
public void testExecuteProcedure() throws Exception {
  // Zeitlimit überschreiten nach einer 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;

  // null oder positiver Status => erfolgreich
  System.out.println("Status: " + response.getStatus());
  System.out.println("Duration: " + duration + " ms");
  assertTrue(response.getStatus() >= 0);
  System.out.println("Done.");
}
```

Ausführliche Informationen zu "NameValueArrays", "ProcedureResponse" und den anderen genannten Methoden und Datentypen finden Sie in den entsprechenden Abschnitten in diesem Handbuch und in den JavaDocs.

Per Hosting bereitgestellte JavaDocs

Ausführliche Informationen über die öffentlichen API-Methoden finden Sie in der iPlanAPI-Klasse in den API-Dokumentationsdateien der JavaDocs. Sie können auf diese Dateien zugreifen, indem Sie sich bei Marketing Operations anmelden, auf einer beliebigen Seite **Hilfe > Produktdokumentation** auswählen und dann die Datei <*version*>PublicAPI.zip herunterladen.

Kapitel 2. Informationen zu Marketing Operations Integration Webservice

Der Web-Service stellt eine Clientansicht der Marketing Operations Integration Services im Rahmen der Bereitstellung des IBM Unica Marketing Operations-Servers zur Verfügung. Der Service ist für die gleichzeitige Verwendung mit Marketing Operations-Webbenutzern bestimmt.

Der Web-Service unterstützt einen API-Aufruf: executeProcedure.

Dieser Web-Service-Anruf wird von einem Client direkt ausgeführt.

Authentifizierung

Eine Authentifizierung ist nicht erforderlich. Alle Clients sind einem bekannten IBM Unica Marketing Operations-Benutzer mit dem Namen "PlanAPIUser" zugeordnet. Dabei ist Voraussetzung, dass die Sicherheitsfunktionen dieses besonderen Benutzers von einem Marketing Operations-Systemadministrator entsprechend den Anforderungen aller Web-Service-Clients konfiguriert wurden.

Eine zukünftige Version des Service wird eventuell über einen allgemeineren Mechanismus für die sichere Clientauthentifizierung verfügen.

Ländereinstellung

Als Ländereinstellung wird ausschließlich die derzeit für die IBM Unica Marketing Operations-Systeminstanz konfigurierte unterstützt. Alle von der Ländereinstellung abhängigen Daten, auf die Zugriff über den Service besteht (Nachrichten, Währung usw.), müssen in der Ländereinstellungen des Systems vorliegen.

Eine zukünftige Version des Service wird eventuell in der Lage sein, Marketing Operations mitzuteilen, welche Ländereinstellung verwendet werden soll.

Statusverwaltung

Der Web-Service ist *statusunabhängig*. Die Serviceimplementierung speichert keine clientspezifischen Informationen für nachfolgende API-Aufrufe. Mit dieser Funktion wird eine effiziente Serviceimplementierung ermöglicht und die Clusterunterstützung vereinfacht.

Datenbanktransaktionen

Der Web-Service legt gegenüber dem Client weder Datenbanktransaktionen noch Bearbeitungssperren offen. Er garantiert jedoch, dass die Auswirkung einer beliebigen Prozedurausführung *atomar* ist. Im Ergebnis bedeutet das, dass die Prozedur entweder erfolgreich ist oder fehlschlägt. Bei Fehlschlagen verbleibt die Datenbank in dem Status, den sie vor dem Aufruf der API hatte.

Informationen zu Datentypen im Marketing Operations Integration Webservice

In diesem Abschnitt werden die Datentypen definiert, die der Web-Service unabhängig von einer bestimmten Servicebindung oder Programmierimplementierung verwendet.

Dies erfolgt in folgender Schreibweise:

- *<Typ>*: *<Typdefinition>* definiert einen einfachen Datentyp. Beispiel:
  Handle: Zeichenfolge
- *<Typ>*: [*<Typdefinition>*] definiert einen komplexen Datentyp oder eine Datenstruktur.
- *<Typ>*: { *<Typdefinition>* } definiert einen komplexen Datentyp oder eine Datenstruktur.

Elemente, die einen komplexen Typ aufweisen, und API-Parameter können mithilfe dieser Typen Arrays deklarieren. Beispiel:

```
Handle [] handles
```

Der Typ "handles" ist ein Array aus Handle-Typen.

Primitive Typen

Primitive Typen sind auf Typen beschränkt, die in der nachfolgenden Tabelle definiert sind, um die Unterstützung von SOAP 1.1-Bindungen zu erleichtern. Alle Typen können als Arrays deklariert sein, zum Beipiel **String [ ]**. An sich können binäre Datentypen, wie **long**, von Protokollbindungen (z. B. SOAP) in Form von Zeichenfolgen dargestellt werden. Diese Darstellung hat jedoch keinerlei Auswirkungen auf die Semantik des Typs, gültige Werte usw., wie der Client diese sieht.

Tabelle 1. Primitive Typen

API-Typ	Beschreibung	SOAP-Typ	Java-Typ
boolean	Boolescher Wert: **wahr** oder **falsch**	xsd:boolean	boolesch
dateTime	Ein Datum-Zeit-Wert	xsd:datetime	Date
decimal	Ein Dezimalwert mit beliebiger Genauigkeit	xsd:decimal	java.math.BigDecimal
double	Ein signierter Dezimalwert mit doppelter Genauigkeit	xsd:double	double
int	Ein signierter 32-Bit-Ganzzahlwert	xsd:int	int
integer	Ein signierter Ganzzahlwert mit beliebiger Genauigkeit	xsd:integer	java.math.BigInteger
long	Ein signierter 64-Bit-Ganzzahlwert	xsd:long	long
string	Eine Zeichenfolge aus Unicode-Zeichen	xsd:string	java.lang.String

MessageTypeEnum

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

"MessageTypeEnum" ist ein Aufzählungstyp, der alle möglichen Nachrichtentypen definiert.

- INFORMATION: eine Informationsnachricht
- WARNING: eine Warnnachricht
- ERROR: eine Fehlernachricht

Message

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

"Message" ist eine Datenstruktur, die das Ergebnis eines Web-Service-API-Aufrufs definiert. Sie stellt optionale Felder für nicht lokalisierten Code, lokalisierten Text und Protokolldetails bereit. Derzeit verwenden alle lokalisierten Textdaten die für die IBM Unica Marketing Operations-Serverinstanz festgelegte Ländereinstellung.

Tabelle 2. Nachrichtenparameter

Parameter	Beschreibung
type	"MessageTypeEnum", legt den Typ der Nachricht fest.
code	Ein optionaler Code für die Nachricht im Zeichenfolgeformat.
localizedText	Eine optionale Textzeichenfolge, die der Nachricht zugeordnet werden soll.
logDetail	Eine optionale Stack-Trace-Nachricht.

NameValue

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

"NameValue" ist ein komplexer Basistyp, der ein Name/Wert-Paar definiert. Dieser Typ definiert auch eine optionale Sequenz, durch die der Service nach Bedarf Werte-Arrays erstellt (die Sequenzen sind nullbasiert).

Alle "NameValues" mit dem gleichen Namen und unterschiedlichen Sequenznummern werden in ein Werte-Array konvertiert und dem gemeinsamen Namen zugeordnet.

Die Array-Größe wird durch die höchste Sequenznummer bestimmt, nicht angegebene Array-Elemente haben Nullwerte. Die Array-Sequenznummern müssen eindeutig sein. Der Wert und sein Typ werden durch den erweiterten Typ angegeben.

Tabelle 3. Parameter von "NameValue"

Parameter	Beschreibung
name	Eine Zeichenfolge, die den Namen eines NameValue-Typs definiert.
sequence	Eine nullbasierte ganze Zahl, die die Sequenznummer für den durch "NameValue" angegebenen Wert festlegt.

Erweiterte NameValue-Typen werden wie folgt für die einzelnen primitiven Typen definiert:

Tabelle 4. Erweiterte NameValue-Typen

Erweiterter Typ	Beschreibung
BigDecimalNameValue: NameValue [decimal value]	Ein NameValue-Typ, dessen Wert eine Dezimalzahl mit beliebiger Genauigkeit ist.
BigIntegerNameValue: NameValue [integer value]	Ein NameValue-Typ, dessen Wert eine Ganzzahl beliebiger Größe ist.
BooleanNameValue: NameValue [boolean value]	Ein NameValue-Typ, der einen booleschen Wert hat.
CurrencyNameValue: NameValue [string locale, decimal value]	Ein NameValue-Typ, der zur Darstellung der Währung für eine Ländereinstellung geeignet ist. Die Ländereinstellung wird im ISO-Sprachencode angegeben, eine für die einzelnen Regionen in ISO-639 festgelegte Kombination aus zwei Kleinbuchstaben. Derzeit muss die Ländereinstellung mit der für IBM Unica Marketing Operations-Servreinstanz festgelegte Ländereinstellung übereinstimmen.
DateNameValue: NameValue [datetime value]	Ein NameValue-Typ, dessen Wert ein Datum ist.
DecimalNameValue: NameValue [double value]	Ein NameValue-Typ, dessen Wert eine Dezimalzahl mit doppelter Genauigkeit ist.
IntegerNameValue: NameValue [long value]	Ein NameValue-Typ, dessen Wert eine 64-Bit-Ganzzahl ist.
String NameValue: NameValue [string value]	Ein NameValue-Typ, dessen Wert eine Zeichenfolge ist.

Ein Array aus Werten vom erweiterten NameValue-Typ ist für die Verwendung definiert, wenn Sie eine Gruppe aus NameValue-Werten mit unterschiedlichen Typen definieren müssen.

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

executeProcedure

Syntax

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

Rückgabe

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

Beschreibung

Diese Methode ruft die angegebene Prozedur mit einem optionalen Array von Parametern auf. Der Aufruf wird synchron ausgeführt, das heißt, der Client wird blockiert und bei Abschluss des Aufrufs wird das Ergebnis ausgegeben.

Parameter

Tabelle 5. executeProcedure-Parameter

Name	Beschreibung
key	Der eindeutige Schlüssel der auszuführenden Prozedur. Es wird ein *RemoteException*-Fehler zurückgegeben, wenn an **key** keine Prozedur gebunden ist.
jobid	Optionale Zeichenfolge, die den dieser Prozedurausführung zugeordneten Job identifiziert. Diese Zeichenfolge ist ein Durchgriffselement, sie kann jedoch verwendet werden, um Clientjobs an die Ausführung einer bestimmten Prozedur zu binden.
paramArray	Ein Array aus Parametern, das an die Prozedur übergeben werden soll. Es werden ein Fehlerstatus und eine Fehlermeldung zurückgegeben, wenn einer oder mehrere der Parameter ungültig ist (falscher Typ, unzulässiger Wert usw.). Es ist dem Client überlassen, die Parameter, ihre Typen und die Anzahl der Werte zu bestimmen, die für die Prozedur erforderlich sind.

Rückgabeparameter

Tabelle 6. Rückgabeparameter für "executeProcedure"

Name	Beschreibung
status	Ein ganzzahliger Code: • 0 gibt an, dass die Prozedur erfolgreich ausgeführt wurde • eine ganze Zahl gibt einen Fehler an Prozeduren können über den Status verschiedene Fehlerstufen angeben.
messages	Ein Array aus null oder mehr Meldungsdatenstrukturen. Wenn **status** den Wert 0 hat, enthält dieses Array keine ERROR-Meldungen, kann jedoch INFORMATION- und WARNING-Meldungen enthalten. Wenn **status** einen anderen Wert als 0 hat, können die Meldungen aus einer beliebigen Kombination aus ERROR-, INFORMATION- und WARNING-Meldungen bestehen.

WSDL von Marketing Operations Integration Services

In diesem Abschnitt wird die Web Services Definition Language (WSDL) für die Marketing Operations Integration Services definiert. Die WSDL wurde manuell definiert und ist das letzte Wort der Web-Service-Definition.

Axis

Diese Version des Web-Service verwendet Axis2 1.5.2 zur Generation der serverseitigen Klassen, aus denen die Web-Service-Implementierung aus der WSDL-Datei besteht. Benutzer können jede beliebige Version von Axis oder auch ein anderes Verfahren als Axis verwenden, um eine clientseitige Implementierung für die Integration in die API aus der bereitgestellten WSDL zu erstellen.

Protokollversion

Die Version des Protokolls ist wie folgt explizit an die WSDL gebunden:

- Als Teil des WSDL-Namens, beispielsweise `PlanIntegrationService1.0.wsdl`
- Als Teil des WSDL-Zielnamensbereichs (targetNamespace), zum Beispiel `xmlns:tns="http://webservices.unica.com /MktOps/services/ PlanIntegrationServices1.0?wsdl"`

WSDL

Eine WSDL-Datei wird mit IBM Unica Marketing Operations Integration Services bereitgestellt: `PlanIntegrationServices1.0.wsdl`. Die WSDL befindet sich bei Auslieferung im Verzeichnis `integration/examples/soap/plan`. Das Beispiel-Erstellungsscript verwendet diese Datei bei der Generierung der erforderlichen clientseitigen Stubs für die Verbindung mit dem Web-Service.

Kapitel 3. IBM Unica Marketing Operations-Prozeduren

Eine *Prozedur* ist eine benutzerdefinierte oder eine Java-Standardklasse, die von IBM Unica Marketing Operations per Hosting bereitgestellt wird und eine Arbeitseinheit ausführt. Prozeduren bieten Kunden, Partnern und IBM Unica Professional Services die Möglichkeit, Marketing Operations-Geschäftslogiken auf beliebige Weise zu erweitern.

Prozeduren folgen einem einfachen Programmiermodell und steuern von Marketing Operations verwaltete Komponenten über eine klar strukturierte API an. Prozeduren werden mithilfe eines einfachen Suchmechanismus und einer XML-basierten Definitionsdatei "erkannt". Marketing Operations führt die Prozeduren gemäß den Bedürfnissen ihrer "Clients" aus. Beispielsweise als Reaktion auf eine Integrationsanforderung (eingehend) oder eine Initialisierung eines Triggers (intern oder ausgehend).

Prozeduren laufen synchron zu ihrem Client ab. Ergebnisse werden dem Client direkt über einen persistenten Überwachungsmechanismus zur Verfügung gestellt. Die Ausführung einer Prozedur kann auch ein Auslösen weiterer Ereignisse und Trigger in Marketing Operations mit sich bringen.

Prozeduren müssen in Java geschrieben sein.

Voraussetzungen

Beachten Sie die folgenden Voraussetzungen bezüglich Prozeduren.

- Die Prozedurimplementierungsklassen sind als Paket in einer separaten Klassenstruktur bzw. einer JAR-Datei zusammengefasst und werden IBM Unica Marketing Operations über einen URL-Pfad zur Verfügung gestellt. Der Prozedurausführungsmanager verwendet ein unabhängiges Klassenladeprogramm, mithilfe dessen diese Klassen nach Bedarf geladen werden können. Marketing Operations sucht standardmäßig im folgenden Verzeichnis:

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

  Um diese Standardeinstellung zu ändern, setzen Sie den Parameter **integrationProcedureClasspathURL** unter **Einstellungen > Konfiguration > Marketing Operations > umoConfiguration > integrationServices**.
- Der Name der Prozedurimplementierungsklasse entspricht der gültigen Java-Namenskonvention, um Konflikte der Paketbenennungen mit "unica" und Klassen anderer Anbieter zu vermeiden. Es ist wichtig, dass der Kunde Prozeduren nicht in der Paketstruktur **com.unica** oder **com.unicacorp** ablegt.
- Die Prozedurimplementierung ist entsprechend der Java-Laufzeitversion codiert, die IBM Unica Marketing Operations auf dem Anwendungsserver verwendet (JRE 1.5.10 oder höher).
- IBM Unica Marketing Operations stellt eine Reihe von Open-Source-Bibliotheken und Bibliotheken von Fremdanbietern bereit. Anwendungsserver verwenden ebenfalls verschiedene Versionen dieser Bibliotheken. Diese Liste ändert sich in der Regel von Release zu Release.

  Anmerkung: Um potenzielle Kompatibilitätsprobleme zu vermeiden, sollten Prozeduren keine Open-Source-, Fremdanbieter- oder anwendungsserverspezifische Bibliotheken verwenden.

Wenn jedoch solche Pakete von einer Prozedur oder von der Sekundärklasse, die von der Prozedur importiert wurde, verwendet werden, müssen sie genau den Anforderungen der von Marketing Operations und/oder dem Anwendungsserver bereitgestellten Paketen entsprechend eingesetzt werden. Beachten Sie, dass Sie in diesem Fall eventuell den Prozedurcode bearbeiten müssen, wenn eine spätere Version von Marketing Operations eine Bibliothek aktualisiert oder nicht weiter verwendet.

- Die Prozedurimplementierungsklasse wird von der Klassenladerichtlinie geladen, die IBM Unica Marketing Operations normalerweise verwendet (in der Regel **parent-last**). Der Anwendungsserver stellt u. U. Entwicklungstools und Optionen zum erneuten Laden von Klassen bereit, die für Marketing Operations gelten, dies ist jedoch nicht zwingend erforderlich.
- Die Prozedur muss bezüglich ihres eigenen Status threadsicher sein, das heißt, ihre Execute-Methode darf nicht von internen Statusänderungen zwischen einzelnen Aufrufen abhängen.
- Eine Prozedur ist nicht in der Lage, eigenständig Threads zu erstellen.

Design

Das Ziel des Designs sollte sein, eine einzelne Arbeitseinheit zu erstellen, die atomar ausgeführt werden muss. Im Idealfall werden bei dem Verfahren eine Reihe von Aufgaben ausgeführt, die asynchron für die Ausführung zu einem späteren Zeitpunkt eingeplant werden können. Dieses "fire and forget"-Integrationsmodell sorgt für die geringste Last in beiden Systemen.

Die Prozedurimplementierungsklasse nutzt die IBM Unica Marketing Operations-API zum Lesen und Aktualisieren von Marketing Operations-Komponenten, Aufrufen von Services und mehr. Andere Java-Pakete können zur Ausführung weiterer Aufgaben eingesetzt werden.

Anmerkung: Nur die dokumentierten Klassen und Methoden werden in zukünftigen Releases von Marketing Operations unterstützt. Alle anderen Klassen und Methoden in Marketing Operations sollten als nicht zulässig betrachtet werden.

Nachdem die Prozedurimplementierungsklassen codiert und kompiliert wurden, müssen sie Marketing Operations zur Verfügung gestellt werden. Die mit den Marketing Operations Integration Services mitgelieferten Erstellungsscripts legen die kompilierten Prozeduren an der Standardposition ab. Der abschließende Entwicklungsschritt besteht in der Aktualisierung der benutzerdefinierten Plug-in-Definitionsdatei für die Prozedur, anhand derer Marketing Operations die benutzerdefinierten Prozeduren ermittelt.

Die Prozedur muss die **com.unica.publicapi.plan.plugin.procedure.IProcedure**-Schnittstelle implementieren und einen parameterlosen Konstruktor (normales JavaBean-Modell) aufweisen. Die Codierung und Kompilierung der einzelnen Prozeduren erfolgt in einem beliebigen Java-Tool der Wahl des Kunden, z. B. Eclipse, Borland JBuilder, Idea usw. Beispielcode wird mit IBM Unica Marketing Operations in Form von Developer Toolkits an folgender Speicherposition zur Verfügung gestellt.

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

Konfiguration

Verwenden Sie die Parameter unter **Einstellungen > Konfiguration > Marketing Operations > umoConfiguration > integrationServices**, um das Marketing Operations Integration Module zu konfigurieren.

Ausführliche Informationen finden Sie im *Marketing Operations Installationshandbuch*.

Lebenszyklus von Prozeduren

Laufzeit-Lebenszyklus einer Prozedur:

1. Erkennung und Initialisierung
2. Auswahl (optional)
3. Ausführung
4. Vernichtung

Erkennung und Initialisierung

IBM Unica Marketing Operations muss über alle Standard- und benutzerdefinierten Prozeduren informiert werden, die für eine bestimmte Installationsinstanz zur Verfügung stehen. Dieser Vorgang wird als Erkennung bezeichnet.

Anmerkung: Standardprozeduren (vom Marketing Operations-Entwicklungsteam definierte Prozeduren) sind implizit bekannt, darum sind keinerlei Aktionen zur Erkennung dieser Prozeduren erforderlich.

Benutzerdefinierte Prozeduren sind in der Plug-in-Definitionsdatei für die Prozedur enthalten. Der Marketing Operations-Plug-in-Manager liest diese Datei während der Initialisierung. Für jede erkannte Prozedur führt der Plug-in-Manager die folgenden Schritte aus:

1. Instanziieren der Prozedur, Überführung ihres Status in INSTANTIATED.
2. Erstellen eines neuen Auditdatensatzes für die Prozedur.
3. Konnte die Prozedur nicht instanziiert werden, wird ihre **`initialize()`**-Methode mit allen Initialisierungsparametern aufgerufen, die in ihrer Plug-in-Beschreibungsdatei vorgefunden wurden. Wenn diese Methode eine Ausnahme auslöst, wird der Status protokolliert und die Prozedur abgebrochen. Andernfalls geht die Prozedur in den Status INITIALIZED über. Sie ist nun zur Ausführung bereit.
4. Erstellen eines neuen Auditdatensatzes für die Prozedur.
5. Wenn die Prozedur initialisiert werden konnte, wird ihre **`getKey()`**-Methode aufgerufen, um den von Clients zum Verweis auf die Prozedur verwendeten Schlüssel zu bestimmen. Dieser Schlüssel wird der Instanz zugeordnet und für spätere Suchvorgänge gespeichert.

Auswahl

Gelegentlich kann es erforderlich sein, dass IBM Unica Marketing Operations dem Benutzer eine Liste der verfügbaren Prozeduren anzeigt, zum Beispiel, um Administratoren das Einrichten eines Triggers zu ermöglichen. Dies erfolgt erst nach Initialisierung der Prozedur. Hierzu werden die Methoden **`getDisplayName()`** und **`getDescription()`** einer Prozedur verwendet.

Ausführung

Nach der Initialisierung der Prozedur erhält IBM Unica Marketing Operations eine Anforderung zur Ausführung der Prozedur. Dies kann zeitgleich mit weiteren Prozeduren (oder der gleichen Prozedur) in anderen Threads erfolgen.

Zur Ausführungszeit führt der Prozedurausführungsmanager folgende Schritte aus.

1. Datenbanktransaktion starten.
2. Prozedurstatus auf EXECUTING setzen.
3. Neuen Auditdatensatz für die Prozedur erstellen.
4. Die **execute()**-Methode der Prozedur mit einem Ausführungskontext und allen execute-Parametern aufrufen, die der Client zur Verfügung stellt. Bei der Methodenimplementierung wird bei Bedarf die Marketing Operations-API verwendet, um Bearbeitungssperren zu erwirken und den Ausführungskontext weiterzugeben. Wenn die execute-Methode eine Ausnahme auslöst, markiert der Ausführungsmanager die Transaktion für den Rollback.
5. Entsprechend den Ausführungsergebnissen Commit oder Rollback für die Transaktion durchführen, den Prozedurstatus auf EXECUTED setzen.
6. Jegliche ausstehende Bearbeitungssperren auflösen.
7. Einen neuen Auditdatensatz für die Prozedur erstellen.

Anmerkung: Die **execute()**-Methode darf die Instanzdaten der Prozedur nicht verändern.

Vernichtung

Wenn IBM Unica Marketing Operations beendet wird, geht der Plug-in-Manager der Prozedur alle geladenen Prozeduren durch. Für jede gefundene Prozedur werden folgende Schritte ausgeführt:

1. Destroy()-Methode der Prozedur aufrufen, um der Prozedur eine Bereinigung zu ermöglichen, bevor die Instanz vernichtet wird.
2. Status der Prozedur in FINALIZED ändern (kann nicht ausgeführt werden).
3. Neuen Auditdatensatz für die Prozedur erstellen.
4. Die Instanz der Prozedur vernichten.

Datensperre

IBM Unica Marketing Operations verwendet ein pessimistisches Bearbeitungssperrschema, das bedeutet, es erhält immer nur ein Benutzer den Aktualisierungszugriff auf eine Komponenteninstanz. Für den Benutzer der grafischen Benutzeroberfläche erfolgt diese Sperre auf der sichtbaren Benutzeroberfläche auf den Registerkarten. In manchen Fällen ist dies ein Subset einer Instanz (zum Beispiel eine Registerkarte mit Projektübersicht), in anderen Fällen vieler Instanzen (die Workflow-Registerkarte). Sobald ein Benutzer eine Sperre erwirkt hat, besteht für alle anderen Benutzer lediglich Lesezugriff auf die betreffenden Daten.

Anmerkung: Bearbeitungssperren sind nicht das gleiche wie Datenbanktransaktionen.

Um sicherzustellen, dass die von einer Prozedur an einer Komponenteninstanz oder einer Gruppe von Instanzen vorgenommenen Änderungen nicht versehentlich von einem anderen Benutzer überschrieben werden, muss eine Prozedur vor dem

Aktualisieren von Komponentendaten die geeigneten Sperren erwirken. Hierzu wird das an die **execute()**-Methode der Prozedur übergebene Ausführungskontextobjekt verwendet.

Bevor die Prozedur Daten aktualisiert, muss sie die **acquireLock()**-Methode des Kontexts für alle erforderlichen Sperren aufrufen. Wenn beispielsweise eine Prozedur ein Projekt und den zugehörigen Workflow aktualisiert, muss sie für beide eine Sperre erwirken.

Wenn ein anderer Benutzer bereits eine Sperre erwirkt hat, löst die **acquireLock()**-Methode unverzüglich die Ausnahme **LockInUseException** aus. Um die Anzahl an Kollisionen zu minimieren, muss die Prozedur die Sperre aufheben, sobald alle Aktualisierungen vorgenommen wurden.

Der Ausführungsmanager hebt automatisch alle ausstehenden Sperren auf, nachdem die execute-Methode zurückgegeben wurde. Sperren bleiben grundsätzlich nur für die Lebensdauer der Datenbanktransaktion bestehen. Folglich erlischen Sperren, wenn das datenbankspezifische Transaktionszeitlimit überschritten ist.

Prozedurtransaktionen

Der Prozedurausführungsmanager schließt die Ausführung der Prozedur automatisch in eine Datenbanktransaktion ein, indem er entsprechend dem Ergebnis der Prozedurausführung ein Commit oder ein Rollback durchführt. Dies garantiert, dass Updates der IBM Unica Marketing Operations-Datenbank erst von anderen Benutzern eingesehen werden können, nachdem das Commit erfolgt ist, und dass diese Updates atomar sind.

Der Autor der Prozedur muss weiterhin die nötigen Schreibsperren erwirken, um sicherzustellen, dass andere Benutzer keine Änderungen an der Datenbank vornehmen können, bevor die Ausführung der Prozedur abgeschlossen wurde.

Kommunizieren von Ergebnissen

Die **execute()**-Methode einer Prozedur gibt einen Ganzzahlstatuscode zurück und null oder mehr Nachrichten, die protokolliert und in der IBM Unica Marketing Operations-Prozeduraudittabelle gespeichert werden. Der Client kann die Statusinformationen auch auf andere Weise kommunizieren.

Prozedurprotokollierung

IBM Unica Marketing Operations legt eine separate Protokolldatei für Prozeduren an.

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

Der Prozedurausführungsmanager protokolliert den Lebenszyklus der einzelnen Prozeduren und erstellt Auditdatensätze.

- **logInfo()**: eine Informationsnachricht in das Prozedurprotokoll schreiben
- **logWarning()**: einen Warnhinweis in das Prozedurprotokoll schreiben
- **logError()**: eine Fehlernachricht in das Prozedurprotokoll schreiben
- **logException()**: Speicherauszug des Stack-Trace für die Ausnahme im Prozedurprotokoll erstellen

Wichtige Java-Klassen

Das bereitgestellte Integrationsentwicklungskit enthält eine Reihe von Javadocs für die öffentliche IBM Unica Marketing Operations-API und unterstützende Klassen. Die wichtigsten sind hier aufgelistet.

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure): Schnittstelle, die von allen Prozeduren implementiert werden muss. Prozeduren durchlaufen einen klar strukturierten Lebenszyklus und greifen auf die Marketing Operations-API zu, um Arbeiten durchzuführen.
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure): Schnittstelle, die von allen auslösenden Prozeduren implementiert werden muss (Markierungsschnittstelle).
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext): Schnittstelle des nicht transparenten Kontextobjekts, das der Ausführungsmanager an die Prozedur übergeben hat. Dieses Objekt verfügt über öffentliche Methoden zur Protokollierung und zur Verwaltung von Bearbeitungssperren. Die Prozedur übergibt dieses Objekt auch an alle PlanAPI-Aufrufe.
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI): Schnittstelle zur Marketing Operations-API. Der Ausführungskontext stellt eine **getPlanAPI()**-Methode zum Abrufen der ordnungsgemäßen Implementierung bereit.

Beispielprozedur

Dieses Beispiel zeigt eine Standardprozedur, mit der der Status eines Projekts mithilfe eines Integrations-Web-Services oder eines Triggers geändert werden kann.

Anmerkung: Verändern Sie die Beispielprozeduren und ihre XML-Definitionen nicht, da die Beispiele beim Upgrade von IBM Unica Marketing Operations überschrieben werden und dabei ihre Änderungen verloren gehen. Alle benutzerdefinierten Prozeduren sollten in einem gesonderten Verzeichnis erstellt und modifiziert werden.

```
// ProjectStateChangeProcedure
// (c) Copyright 2006 by Unica Corporation. Alle Rechte vorbehalten.

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> ist eine Marketing Operations-Standardprozedur,
 * die versucht,
 * den Status eines Projekts zu überführen.
```

```
 * <p>
 * Erwartet die folgenden Initialisierungsparameter:
 * <ul>
 *   <li>debug: Boolesches Objekt, <tt>true</tt> oder <tt>false</tt>, das angibt,
 *   ob die Tracefunktion für die Fehlerbehebung aktiviert ist oder nicht</li>
 * </ul>
 *
 * <p>
 * Erwartet die folgenden Ausführungsparameter:
 * <ul>
 *   <li>hProject: Zeichenfolgenarrayformat eines Projekthandles, z. B.,
 *     "http://mymachine:7001/MktOps/affiniumplan.jsp?
 *         cat=projecttabs&projectid=12"</li>
 *   <li>uapState: Zeichenfolgenarray eines neuen Projektstatus, z. B.
 *      "COMPLETED". Groß-/Kleinschreibung ist zu beachten!</li>
 * </ul>
 *
 */
public final class ProjectStateChangeProcedure implements IProcedure {
// Initialisierungsparameter
private final static String DEBUG_INITPARAMETER_NAME = "debug";
  // Ausführungsparameter
  private final static String HPROJECT_PARAMETER_NAME = "hProject";
  private final static String STATE_PARAMETER_NAME
    = IPlanAPI.PROJECT_ATTRIBUTE_STATEENUM; // entspricht dem Attributnamen

  // unsere Statuscodes
  private final static int STATUS_SUCCESS = 0;

    // Debug-Eigenschaft. Den Initialisierungsparameter "debug" der Prozedur
  //   auf wahr setzen, um den Debug-Trace zu aktivieren
    private boolean _debug = false;
    private boolean isDebug() { return _debug; }

  // der einfache Name ist der nicht qualifizierte Klassenname
  public String getName() {
        return "uapProjectStateChangeProcedure";
    }

  // der Anzeigename ist immer der Schlüssel
  public String getDisplayName(Locale locale) {
        // zunächst nur EN
        return getName();
    }

  // Beschreibung immer auf Englisch
    public String getDescription(Locale locale) {
        // zunächst nur EN
          return "Eine Prozedur zur Überführung des Status eines Projekts.";
    }

  // Codeversion; muss im Moment 1.0.0 sein
  public PluginVersion getVersion() {
    return new PluginVersion(1,0,0);
  }

  // Initialisieren der Instanz aus Initialisierungsparametern
    public void initialize(Map initParameters)
      throws ProcedureInitializationException {
    // der einzige vorhandene Initialisierungsparameter ist: 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());
            }
            }
        }

      // Ausführen: erwartungsgemäß hProject und state enum
        public ProcedureResult execute(IExecutionContext context, Map parameters)
          throws ProcedureExecutionException {
            // get_execute-Parameter: erwartet werden zwei:
            // - hProject: string[]-Format des Projekthandles
            // - uapState: string[]-Format von 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 muss angegeben sein.");

        int status = -1;
        ProcedureMessage[] messages = null;
            try {
               // Versuch, eine Bearbeitungssperre für das Projekt zu erwirken
                context.acquireLock(hProject, IExecutionContext.LOCK_ALL_FIELDS);

                // PlanAPIImpl zur Aktualisierung des Status verwenden
                IPlanAPI planAPI = context.getPlanAPI();
                planAPI.updateAttribute(context, hProject, STATE_PARAMETER_NAME,
                            new ProjectStateEnum[]{stateEnum});

              // erfolgreich
              status = STATUS_SUCCESS;

            } catch (Exception exception) {
              // bei Debug in Stack-Trace schreiben
                if (isDebug()) {
                    context.logError(getName(), exception);
                }
                throw new ProcedureExecutionException(exception);
            } finally {
              // Sperre freigeben
              try {
                context.releaseAllLocks();
              } catch (Exception exception) { /* ignored */ }
            }
```

```
        return new ProcedureResult(status, messages);
    }

    public void destroy( ){
         // keine Schritte erforderlich
    }
}
```

Prozedur-Plug-in-Definitiondatei

In diesem Abschnitt wird die Plug-in-Definitionsdatei für Prozeduren beschrieben. Diese Datei definiert Implementierungsklasse, Metadaten und andere Informationen zu den benutzerdefinierten Prozeduren, die IBM Unica Marketing Operations per Hosting bereitstellen soll. Die Plug-in-Definitionsdatei für eine Prozedur wird standardmäßig in folgendem Verzeichnis abgelegt:

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

Diese Datei ist ein XML-Dokument, das die nachfolgend beschriebenen Informationen enthält.

Prozeduren: eine Liste aus null oder mehr **Procedure**-Elementen.

Prozedur: ein Element, das eine Prozedur definiert. Jede Prozedur enthält die folgenden Elemente.

- **key** (optional): Zeichenfolge, die den Suchschlüssel für die Prozedur definiert. Dieser Schlüssel muss für alle Standardprozeduren (durch IBM bereitgestellt) sowie benutzerdefinierte Prozeduren, die von einer bestimmten Marketing Operations-Instanz per Hosting bereitgestellt werden, eindeutig sein. Wird kein Schlüssel definiert, dient automatisch die vollständig qualifizierte Version des **className**-Elements als Schlüssel. Namen, die mit der Zeichenfolge "uap" beginnen, sind für die Verwendung durch IBM Unica Marketing Operations reserviert.
- **className** (erforderlich): vollständig qualifizierter Paketname der Prozedurklasse. Diese Klasse muss die IProcedure-Klasse implementieren (com.unica.public-.plan.plugin.procedure.IProcedure).
- **initParameters** (optional): eine Liste aus null oder mehr initParameter-Elementen.

  initParameter (optional): Parameter, der an die Methode "initialize()" der Prozedur übergeben werden soll. Dieses Element beinhaltet den verschachtelten Parameternamen, den Typ und die Wertelemente.
  - name: Zeichenfolge, die den Parameternamen definiert
  - type: optionaler Klassenname der Java-Wrapperklasse, die den Typ des Parameterwerts definiert. Folgende Typen sind möglich:
    - java.lang.String (Standardtyp)
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  - value: Zeichenfolgenformat des Attributwerts entsprechend dem Typ

Kapitel 4. Informationen zur IBM Unica Marketing Operations-API

Die IBM Unica Marketing Operations-API ist eine Art Kulisse, auf der eine Client-Ansicht einer laufenden Marketing Operations-Instanz bereitstellt wird. Es ist nur ein Teil des Marketing Operations-Leistungsspektrums zugänglich. Die API ist für die gleichzeitige Verwendung durch Marketing Operations-Webbenutzer und SOAP-Anforderungen und Trigger des Web-Services Marketing Operations Integration Services konzipiert. Die API unterstützt die folgenden Operationstypen.

- Komponentenerstellung und Löschen von Komponenten
- Erkennung (nach Komponententyp, Attributwert usw.)
- Komponentenprüfung (anhand ihrer Attribute, spezialisierter Links usw.)
- Komponentenbearbeitung

Versionierung und Abwärtskompatibilität

Zukünftige Versionen dieser API werden abwärtskompatibel mit allen untergeordneten Releases und Wartungsreleases mit gemeinsamer Hauptversionsnummer sein. IBM behält sich jedoch das Recht vor, keine Kompatibilität mit früheren Versionen für Hauptreleases (x.0) zu gewährleisten, wenn die vorliegende Geschäftssituation bzw. die technischen Umstände dies rechtfertigen.

Die Hauptversionsnummer dieser API wird inkrementell erhöht, wenn eine der folgenden Änderungen vorgenommen wird.

- Dateninterpretation geändert
- Geschäftslogik geändert (z. B. Servicemethodenfunktionalität geändert)
- Methodenparameter und/oder Rückgabetypen geändert

Die Nebenversionsnummer der API wird inkrementell erhöht, wenn eine der folgenden Änderungen vorgenommen wurde (beachten Sie, dass diese Änderungen grundsätzlich abwärtskompatibel sind).

- Neue Methode hinzugefügt
- Neuer Datentyp hinzugefügt und dessen Verwendung auf eine neue Methode beschränkt
- Neues Element zu einem Aufzählungstyp hinzugefügt
- Eine neue Version einer Schnittstelle wird mit einem Versionssuffix definiert

Benutzersicherheit

Es wird vorausgesetzt, dass die Authentifizierung durch den Ausführungsmanager der Prozedur erfolgt und die authentifizierten Benutzerdaten an den Ausführungskontext gebunden sind, der von allen APIs verwendet wird. Die API legt den authentifizierten Benutzer nicht offen, gibt ihn aber an IBM Unica Marketing Operations zur Verwendung nach Bedarf weiter.

Der authentifizierte Benutzer ist unter Umständen nicht berechtigt, alle von der API zur Verfügung gestellten Operationen auszuführen. In diesem Fall löst die API-Methode die Ausnahme **`AuthorizationException`** aus.

Ländereinstellung

Diese Version unterstützt ausschließlich die Ländereinstellung, die derzeit für die IBM Unica Marketing Operations-Serverinstanz konfiguriert ist. Es wird vorausgesetzt, dass alle von der Ländereinstellung abhängigen Daten, die über die API zugänglich sind (Nachrichten, Währung usw.), diese Systemländereinstellung aufweisen.

Statusverwaltung

Diese API ist statusunabhängig, das bedeutet, es werden keine Client-bezogenen Informationen von der API für nachfolgende Aufrufe übernommen.

Beachten Sie jedoch, dass bestimmte API-Aufrufe den Status von zugrundeliegenden Komponenteninstanzen, die von IBM Unica Marketing Operations verwaltet werden, ändern können. Diese Statusänderungen werden gegebenenfalls in einer Datenbank gespeichert.

Datenbanktransaktionen

Diese API legt dem Client keine Datenbanktransaktionen offen, verwendet entsprechende Informationen jedoch, sofern diese in den Ausführungskontext eingeschlossen sind. Nach dem Start einer Transaktion sind die Auswirkungen aller API-Aufrufe innerhalb einer bestimmten Prozedur atomar. Andere Benutzer von IBM Unica Marketing Operations können die Änderungen erst einsehen, nachdem die Prozedur die Transaktion erfolgreich festgeschrieben hat.

API-Aufrufe, die die Datenbank aktualisieren, müssen zunächst eine Bearbeitungssperre erwirken, um zu verhindern, dass andere Marketing Operations-Benutzer während der Ausführung des bzw. der API-Aufrufe Änderungen an den zugrundeliegenden Daten vornehmen. Andere Benutzer können daraufhin die gesperrten Komponenten nicht aktualisieren, solange die API nicht alle Aufrufe abgeschlossen hat. Entsprechend kann auch ein anderer Marketing Operations-Benutzer oder API-Client eine Sperre für die gewünschten Daten erwirkt haben, was eine erfolgreiche Ausführung des API-Aufrufs verhindert.

Ereignisverarbeitung

Wenn eine Operation auf einer IBM Unica Marketing Operations-Komponente über diese API ausgeführt wird, werden dabei die gleichen Ereignisse generiert wie bei einer Ausführung der Operation durch einen Marketing Operations-Webbenutzer. Insbesondere bedeutet dies, dass Trigger, die auf bestimmte Ereignisse warten, früher oder später in beiden Fällen ausgelöst werden. Benutzer, die bestimmte Benachrichtigungen abonniert haben (beispielsweise bei Statusänderung eines Projekts), werden über Statusänderungen benachrichtigt, die aus API-Aufrufen sowie aus Aktionen von Webbenutzern folgen.

Informationen zu API-Datentypen

Die IBM Unica Marketing Operations-API enthält die folgenden öffentlichen Datentypen.

- **ApprovalMethodEnum**: Aufzählungstyp für Genehmigungsmethoden.
- **ApprovalStateEnum**: Aufzählungstyp für Genehmigungsstatus.
- **AssetLibraryStateEnum**: Aufzählungstyp für Assetbibliothekstatus.
- **AssetStateEnum**: Aufzählungstyp für Assetstatus.

- **AttachmentTypeEnum**: Aufzählungstyp für Anhänge.
- **AttributeMap**: Java-Mapping mit Attributen und ihren Werten.
- **BudgetPeriodEnum**: Aufzählungstyp für Budgetperiodentypen.
- **BudgetTypeEnum**: Aufzählungstyp für Budgettypen
- **Handle**: identifiziert eine Komponenteninstanz in einer bestimmten Marketing Operations-Instanz.
- **InvoiceStateEnum**: Aufzählungstyp für Rechnungsstatus.
- **MonthEnum**: Aufzählungstyp für Monate in einem Kalenderjahr
- **ProjectCopyTypeEnum**: Aufzählungstyp für Kopierprojektoptionen.
- **ProjectStateEnum**: Aufzählungstyp für Projektstatus.
- **TaskStateEnum**: Aufzählungstyp für Aufgabenstatus.
- **QuarterEnum**: Aufzählungstyp für Quartale eines Kalenderjahres
- **WeekEnum**: Aufzählungstyp für Wochen eines Kalenderjahres

Mögliche Werte für die einzelnen Datentypen werden nachfolgend genannt.

Aufzählungstypen

ApprovalMethodEnum

"ApprovalMethodEnum" ist ein Aufzählungstyp, der alle zulässigen Methoden einer Genehmigung definiert. Folgende Werte sind möglich:

- SIMULTANEOUS
- SEQUENTIAL

ApprovalStateEnum

ApprovalStateEnum ist ein Aufzählungstyp, der alle möglichen Status einer Genehmigung definiert. Folgende Werte sind möglich:

- NOT_STATED
- ON_HOLD
- IN_PROGRESS
- COMPLETED
- CANCELLED

AssetLibraryStateEnum

AssetLibraryStateEnum ist ein Aufzählungstyp, der alle möglichen Status einer Assetbibliothek definiert. Folgende Werte sind möglich:

- ENABLED
- DISABLED

AssetStateEnum

AssetStateEnum ist ein Aufzählungstyp, der alle möglichen Status eines Assets definiert. Folgende Werte sind möglich:

- DRAFT
- LOCK
- FINALIZE
- ARCHIVE

AttachmentTypeEnum

AttachmentTypeEnum ist ein Aufzählungstyp, der alle möglichen Typen von Anhängen definiert. Folgende Werte sind möglich:

- URL
- FILE
- ASSET

BudgetPeriodEnum

BudgetPeriodEnum ist ein Aufzählungstyp, der alle möglichen Budgetperioden definiert. Folgende Werte sind möglich:

- QUARTERLY
- MONTHLY
- WEEKLY
- YEARLY
- ALL

BudgetTypeEnum

BudgetTypeEnum ist ein Aufzählungstyp, der alle möglichen Budgettypen definiert. Folgende Werte sind möglich:

- TOTAL
- FORECAST
- COMMITTED
- ACTUAL
- ALLOCATED

InvoiceStateEnum

InvoiceStateEnum ist ein Aufzählungstyp, der alle möglichen Status einer Rechnung definiert. Folgende Werte sind möglich:

- DRAFT
- PAYABLE
- PAID
- CANCELLED

MonthEnum

MonthEnum ist ein Aufzählungstyp, der alle möglichen Werte für den Monat definiert.

ProjectCopyTypeEnum

ProjectCopyTypeEnum ist ein Aufzählungstyp, der alle möglichen Typen von Kopierprojekten definiert. Folgende Werte sind möglich:

- COPY_USING_PROJECT_METRICS
- COPY_USING_TEMMPLATE_METRICS

Details zu diesen Projekt- und Aufgabenstatus finden Sie im Benutzerhandbuch *Marketing Operations User Guide*.

ProjectStateEnum

ProjectStateEnum ist ein Aufzählungstyp, der alle möglichen Status eines Projekts oder einer Anforderung definiert. Folgende Werte sind möglich:

- NOT_STARTED
- IN_PROGRESS
- ON_HOLD
- IN_RECONCILIATION
- LATE: gibt an, dass das Projekt nicht zum geplanten Anfangsdatum begonnen hat.
- OVERDUE: gibt an, dass das Projekt nicht vor dem geplanten Enddatum abgeschlossen wurde.
- DRAFT
- SUBMITTED
- RETURNED
- ACCEPTED
- COMPLETED
- CANCELLED
- DELETED

TaskStateEnum

TaskStateEnum ist ein Aufzählungstyp, der alle möglichen Status einer Workflowaufgabe definiert. Folgende Werte sind möglich:

- PENDING
- ACTIVE
- FINISHED
- SKIPPED
- DISABLED

QuarterEnum

QuarterEnum ist ein Aufzählungstyp, der alle möglichen Werte für Quartal definiert.

WeekEnum

WeekEnum ist ein Aufzählungstyp, der alle möglichen Werte für Woche definiert.

Handle

Ein Handle ist ein spezielles URL-Objekt, das auf eine Komponenteninstanz in einer bestimmten Instanz von Marketing Operations verweist. Handles sind unter anderem der Komponententyp, IDs von internen Daten und die Basis-URL einer Instanz. Von der API verwendete oder generierte Handles können als vollständige URL exportiert werden, die Sie beispielsweise zur Navigation zu einer Sicht der Komponente auf der grafischen Benutzeroberfläche von Marketing Operations verwenden, ausschneiden und wieder einfügen, in E-Mails verschicken, in einer anderen Prozedur verwenden können.

Handles gelten nur für eine bestimmte Serviceinstanz von Marketing Operations (bzw. eine in Gruppen zusammengefasste Instanz), sind dann jedoch für die ge-

samte Lebensdauer des implementierten Service gültig. Das bedeutet, Handles können zur späteren Verwendung in einer Datei gespeichert werden, sie können jedoch nicht für den Zugriff auf Komponenten einer anderen Instanz von Marketing Operations verwendet werden (auch wenn sich diese im gleichen System befinden). Marketing Operations stellt einen Mechanismus zur Verfügung, mit dem verschiedene Basis-URLs zur aktuellen Instanz zugeordnet werden können, um das Verlagern einer Instanz in ein anderes System zu ermöglichen (beispielsweise bei Störungen im ursprünglichen System).

Handles sind clientunabhängig. Beispielsweise kann ein Auslöser ein Handle an eine Prozedur übergeben, die diesen als Parameter für einen SOAP-Aufruf an ein Drittanbietersystem verwendet. Dieses sendet daraufhin eine SOAP-Anforderung an Marketing Operations zurück, um eine Prozedur aufzurufen, die ein Attribut aktualisiert.

Die Handle-Klasse verfügt über Factory-Methoden zum Erstellen von Handles aus verschiedenen Typen von URLs.

Approval Object:

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

AssetLibrary Object:

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

Asset Folder Object:

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

Asset Object:

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

Attachment Object:

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

Financial Account Object:

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

Invoice line item object:

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

Invoice Object:

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

Marketing object:

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

Marketing object grid:

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

Marketing object grid row:

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

Project:

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

Project grid:

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

Project grid row:

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

Project line item object:

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

Team Object:

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

User Object:

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

Workflow task:

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

Attributzuordnung (AttributeMap)

Eine Attributzuordnung (AttributeMap) ist eine Zuordnung, die ausschließlich Attribute enthält. Der *Attributname* ist der Schlüssel des Zuordnungseintrags und das *Attributwerte*-Array (Plural!) ist der Zuordnungseintragswert.

Die Attributzuordnung (AttributeMap) enthält die folgenden Felder:

- *Name*: der programmorientierte Name des Attributs. Dieser Name dient als eindeutiger Schlüssel für den Zugriff auf das Attribut innerhalb der Komponenteninstanz, in der es vorkommt.

  Anmerkung: Dieser *Name* entspricht nicht unbedingt dem Anzeigenamen, den Benutzer auf der grafischen Benutzeroberfläche sehen. Der Attributname von Komponenten, die mithilfe einer Vorlage erstellt wurden (z. B. Projekte oder Workflowaufgaben), wird in der Vorlagenelementdefinition festgelegt und muss eindeutig sein. Bei anderen Komponenten wird der Attributname in der Regel programmgesteuert von der serverseitigen Komponenteninstanz abgeleitet (z. B. durch Java-Introspektion).

  Anmerkung: Gemäß Konvention umfassen kundenspezifische Attribute den Namen des Formats, in dem die bearbeitbare Version definiert ist: <formatname>.<attributname>.
- *values*: ein Java-Objekt-Array mit null oder mehr Attributwerten. Die einzelnen Werte müssen den gleichen Typ aufweisen und dem Typ des Attributs entsprechend, der in Marketing Operations definiert ist. Es werden ausschließlich die folgenden Java-Wrapper und Marketing Operations-Typen unterstützt:

- AssetLibraryStateEnum: ein AssetLibraryStateEnum-Aufzählungstypwert.
- AssetStateEnum: ein AssetStateEnum-Aufzählungstypwert.
- AttachmentTypeEnum: ein AttachmentTypeEnum-Aufzählungstypwert.
- AttributeMap: eine Zuordnung, die Attribute enthält.
- BudgetPeriodEnum: ein BudgetPeriodEnum-Aufzählungstypwert.
- BudgetTypeEnum: ein BudgetTypeEnum-Aufzählungstypwert.
- Handle: ein Verweis auf eine Komponenteninstanz, Rasterzeile, ein Attribut usw.
- InvoiceStateEnum: ein InvoiceStateEnum-Aufzählungstypwert.
- java.io.File: Darstellung einer Datei.
- java.lang.Boolean: ein boolescher Wert, entweder „wahr“ oder „falsch“.
- java.lang.Double: ein Dezimalzahlenwert mit doppelter Genauigkeit.
- java.lang.Float: ein Dezimalzahlenwert mit einfacher Genauigkeit.
- java.lang.Integer: ein 32-Bit-Ganzzahlwert.
- java.lang.Long: ein 64-Bit Ganzzahlwert.
- java.lang.Object: generisches Java-Objekt.
- java.lang.String: eine Zeichenfolge aus null oder mehr Unicode-Zeichen.
- java.math.BigDecimal: Dezimalzahlenwert mit beliebiger Genauigkeit. Geeignet für Währung; die Interpretation des Wertes hängt von der Ländereinstellung für die Währung des Clients ab.
- java.math.BigInteger: Ganzzahlwert mit beliebiger Genauigkeit.
- java.net.URL: ein URL (Universal Resource Locator)-Objekt.
- java.util.ArrayList: Liste von Objekten.
- java.util.Calendar: ein Datum/Uhrzeit-Wert für eine bestimmte Ländereinstellung.
- java.util.Date: ein Datum/Uhrzeit-Wert. Dieser Typ wird nicht weiter unterstützt. Verwenden Sie stattdessen "java.util.Calendar" oder "java.util.GregorianCalendar".
- java.util.GregorianCalendar: "GregorianCalendar" ist eine konkrete Unterklasse von "java.util.Calendar" und stellt das Standardkalendarsystem bereit, dass von den meisten Ländern verwendet wird.
- MonthEnum: ein MonthEnum-Aufzählungstypwert.
- ProjectStateEnum: ein ProjectStateEnum-Aufzählungstypwert.
- QuarterEnum: ein QuarterEnum-Aufzählungstypwert.
- TaskStateEnum: ein TaskStateEnum-Aufzählungstypwert.
- WeekEnum: ein WeekEnum-Aufzählungstypwert.

Die Metadaten eines Attributs (wie lokalisierter Anzeigename und Beschreibung) sind in der Vorlage, die dem Attribut zugeordnet ist, sowie der übergeordneten Objektinstanz definiert. Attribute stellen einen einfachen, dabei aber erweiterbaren Mechanismus zum Verfügbarmachen von erforderlichen und optionalen Objektinstanzattributen bereit, beispielsweise Projektname, Code und Startdatum.

Anmerkung: Zur Implementierung des Datums stehen dem Benutzer "java.util.Calendar" oder "java.util.GregorianCalendar" zur Verfügung.

Häufig auftretende Ausnahmen

Dieses Thema beschreibt einige Ausnahmen, die häufig von der API ausgelöst werden.

- AuthorizationException: Der dem Ausführungskontext des Clients zugeordnete Benutzer ist nicht zur Ausführung der angeforderten Operation berechtigt. Diese kann von jeder beliebigen API-Methode ausgelöst werden, sie ist also nicht deklariert.
- DataException: In der zugrundeliegenden Datenbankebene in IBM Unica Marketing Operations ist eine Ausnahme aufgetreten. Details können Sie dem SQL-Protokoll entnehmen.
- InvalidExecutionContextException: Es ist ein Problem mit einem Ausführungskontext aufgetreten, der an eine API-Methode übergeben wurde (beispielsweise wurde die Methode nicht richtig initialisiert). Diese Ausnahme von jeder beliebigen API ausgelöst werden, sie ist also nicht deklariert.
- NotLockedException: Versuch der Aktualisierung von Komponentendaten, ohne vorher die erforderliche Sperre erwirkt zu haben. Siehe acquireLock()-Methode von "IExecutionContext".

API-Methoden

Ausführliche Informationen über die öffentlichen API-Methoden finden Sie in der iPlanAPI-Klasse in den API-Dokumentationsdateien der JavaDocs. Sie können auf diese Dateien zugreifen, indem Sie sich bei Marketing Operations anmelden, auf einer beliebigen Seite **Hilfe > Produktdokumentation** auswählen und dann die Datei `<version>PublicAPI.zip` herunterladen.

Kontakt zum technischen Support von IBM Unica

Sollte sich ein Problem nicht mithilfe der Dokumentation beheben lassen, können sich die für den Kundendienst zuständigen Kontaktpersonen Ihres Unternehmens telefonisch an den technischen Support von IBM Unica wenden. Damit wir Ihnen möglichst schnell helfen können, beachten Sie dabei bitte die Informationen in diesem Abschnitt.

Wenn Sie wissen möchten, wer die zuständige Kontaktperson Ihres Unternehmens ist, wenden Sie sich an Ihren IBM Unica -Administrator.

Zusammenzustellende Informationen

Halten Sie folgende Informationen bereit, wenn Sie sich an den technischen Support von IBM Unica wenden:

- Kurze Beschreibung der Art Ihres Problems
- Detaillierte Fehlermeldungen, die beim Auftreten des Problems angezeigt werden
- Schritte zum Reproduzieren des Problems
- Entsprechende Protokolldateien, Session-Dateien, Konfigurationsdateien und Daten
- Informationen zu Ihrer Produkt- und Systemumgebung, die Sie entsprechend der Beschreibung unter „Systeminformationen“ abrufen können.

Systeminformationen

Bei Ihrem Anruf beim technischen Support von IBM Unica werden Sie um verschiedene Informationen gebeten.

Sofern das Problem Sie nicht an der Anmeldung hindert, finden Sie einen Großteil der benötigten Daten auf der Info-Seite. Dort erhalten Sie Informationen zu der installierten IBM Unica -Anwendung.

Sie können über **Hilfe > Info** (Help > About) auf die Info-Seite zugreifen. Wenn Sie nicht auf die Info-Seite zugreifen können, finden Sie die Versionsnummer der IBM Unica -Anwendung in der Datei `version.txt` im Installationsverzeichnis jeder Anwendung.

Kontaktinformationen für den technischen Support von IBM Unica

Wenn Sie sich an den technischen Support von IBM Unica wenden möchten, finden Sie weitere Informationen auf der Website des technischen Supports für IBM Unica -Produkte (http://www.unica.com/about/product-technical-support.htm).

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Défense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

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

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Alle von IBM angegebenen Preise sind empfohlene Richtpreise und können jederzeit ohne weitere Mitteilung geändert werden. Händlerpreise können unter Umständen von den hier genannten Preisen abweichen.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM, die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielprogramme entstehen.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite „Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.