IBM Unica Marketing Operations

버전 8 릴리스 6

2012년 5월 25일

통합 모듈

IBM

참고

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 반드시 33 페이지의 『주의사항』의 일반 정보를 읽으십시오.

이 개정판은 새 개정판에 달리 표시되어 있지 않는 한 IBM Unica Marketing Operations 버전 8, 릴리스 6, 수정 0과 모든 후속 릴리스 및 수정에 적용됩니다.

목차

제 1 장 Marketing Operations 통합 서비스 개념

Marketing Operations 통합 서비스는 다음을 복합한 것입니다.

- **Marketing Operations 통합 웹 서비스**

  통합 서비스는 IBM® UnicaMarketing Operations 고객, 파트너 및 IBM Professional Services가 Marketing Operations를 해당 환경에서 실행 중인 기타 애플리케이션과 통합할 수 있는 방법을 제공합니다.

- **Marketing Operations 프로시저 및 API**

  사용자 정의 프로시저는 임의 방법으로 Marketing Operations 비즈니스 로직을 확장하기 위해 Marketing Operations 내에서 정의할 수 있습니다. 정의되면, 이 프로시저는 다른 애플리케이션의 통합 서비스 웹 서비스 호출에 대한 대상이 될 수 있습니다. 프로시저는 또한 메시지를 다른 애플리케이션에 보내도록 정의할 수도 있습니다.

- **Marketing Operations 트리거**

  트리거는 Marketing Operations에서 이벤트 및 프로시저와 연관될 수 있습니다. 이러한 이벤트가 발생하면 연관된 트리거가 실행됩니다.

Marketing Operations 통합 서비스의 요구사항

Marketing Operations 통합 서비스는 다음을 수행해야 합니다.

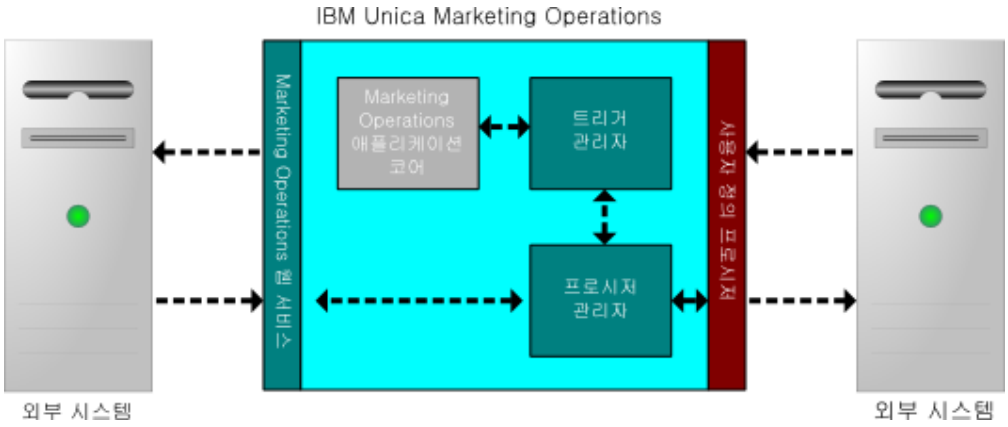
- 시스템 통합을 느슨하게 결합합니다.
- 웹 서비스를 호출을 통해 Marketing Operations에 영향을 주도록 고객 애플리케이션에 대한 메커니즘을 제공합니다.
- Marketing Operations에서 특정 이벤트를 알리도록 고객 애플리케이션에 대한 메커니즘을 제공합니다.
- 이해하고 사용하기 쉬운 단순한 프로그래밍 모델을 제공합니다.
- 실패에서 복구할 때 강력해야 합니다.
- 데이터 무결성을 보장합니다.
- 기존 Marketing Operations GUI 기반 고객과 통합되고 이 고객에 대한 노력을 최소화합니다.
- 기초적인 구현 세부사항으로부터 프로그래머를 보호하면서 Marketing Operations 구성 요소에 대한 세부적인 액세스를 제공합니다.

IBM UnicaMarketing Operations 통합 서비스 시작하기

IBM UnicaMarketing Operations 통합 서비스 기능을 사용하여 사용자 정의 프로시저를 작성할 수 있습니다. Marketing Operations 내에서 특정 이벤트가 발생할 때 외부 이벤트를 트리거하기 위해 이 프로시저를 사용할 수 있습니다. 외부 시스템 또는 프로그램에서 Marketing Operations 기능을 수행하기 위해 이 프로시저를 사용할 수 있습니다.

API를 사용하여 프로그램 수준에서 IBM UnicaMarketing Operations와 통합합니다. 어떤 경우에는 사용자 수준에서 Marketing Operations와 인터페이스하기 위해 GUI를 사용합니다. API를 사용하여 프로시저를 구성합니다. 이 프로시저를 사용하여 Marketing Operations 및 외부 시스템 사이에 통신합니다. Marketing Operations 웹 서비스는 프로시저, API 및 트리거에 대한 컨테이너 개체입니다.

다음은 Marketing Operations 통합 서비스의 아키텍처입니다.

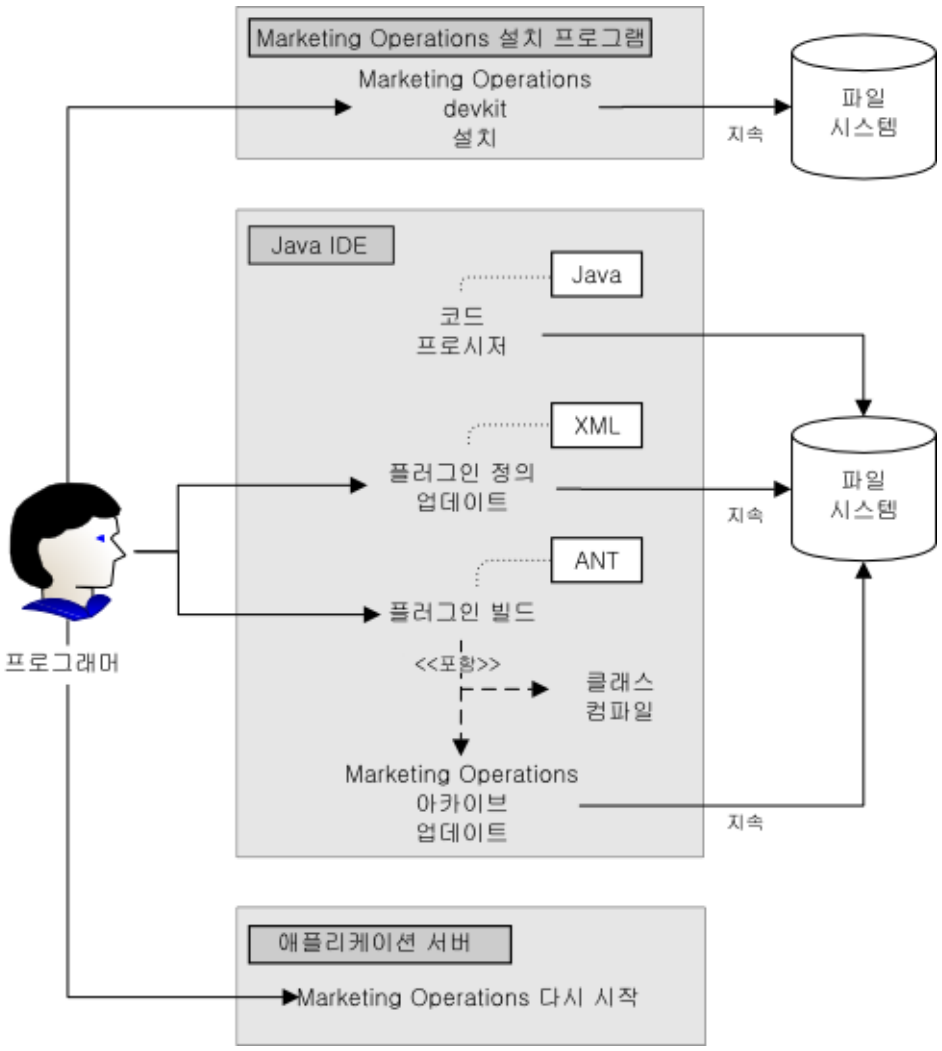


다음은 통합 서비스의 주요 구성 요소입니다.

- Marketing Operations 프로시저 관리자: API를 통해 Marketing Operations와 상호작용하여 비즈니스 로직을 확장합니다.
- Marketing Operations 트리거 관리자: 조건(예: 마케팅 개체의 상태 변경)을 조치(트리거에 대한 조건이 충족될 때 실행할 프로시저)와 연관시킵니다.

방법론

다음 다이어그램에 표시된 것처럼 IBM UnicaMarketing Operations 통합 서비스의 구성 요소를 사용하여 사용자 정의 프로시저를 개발합니다.



Marketing Operations Installer를 사용하여 개발자 킷을 설치한 후에는 다음 기본 단계를 따르십시오.

1. 사용자 정의 프로시저를 코딩하십시오. 현재, Java™를 사용해야 합니다.
2. XML 정의에서 플러그인 정의를 업데이트하십시오.
3. 플러그인을 빌드하십시오.
   a. 필요한 클래스를 컴파일하십시오.
   b. Marketing Operations 보관(WAR 파일)을 업데이트하십시오.
4. Marketing Operations를 다시 시작하십시오.

IBM UnicaMarketing Operations 및 API 사이의 통신을 위한 기본 예제

이 절에서는 API와 Marketing Operations 사이의 통신을 설정하는 기본 예제를 설명합니다. 유용한 작업을 수행하지 않고 Marketing Operations 및 통합 서비스 사이의 라운드트립을 수행합니다.

이 절에서는 Marketing Operations 통합 서비스 개발자 킷과 함께 포함된 예제 프로시저의 부분을 사용합니다. 특히, 다음 파일에서 여기에 참조된 코드를 찾을 수 있습니다.

- PlanClientFacade.java
- PlanWSNOOPTestCase.java

noop 메소드는 Marketing Operations에 대한 웹 서비스 호출입니다. 이 메소드는 PlanClientFacade 클래스에서 정의되며 배열에서 널값을 전달합니다.

```
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);
}
```

프로시저 testExecuteProcedure는 PlanClientFacade의 noop 메소드를 호출하여 Marketing Operations 응용 프로그램과의 라운드트립을 설정합니다.

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

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

NameValueArrays, ProcedureResponse 및 기타 나열된 메소드 및 데이터 유형에 대해서는, 이 안내서의 나머지와 JavaDoc에서 해당 절을 참조하십시오.

호스트된 JavaDocs

공용 API 메소드에 특정한 정보는 JavaDocs API 문서 파일의 iPlanAPI 클래스를 참조하십시오. 이러한 파일은 Marketing Operations에 로그인하여 페이지에서 **도움말 > 제품 문서**를 선택한 다음 *<version>*PublicAPI.zip 파일을 다운로드하여 볼 수 있습니다.

제 2 장 Marketing Operations 통합 웹 서비스 정보

웹 서비스는 IBM UnicaMarketing Operations 서버 개발의 일부인 Marketing Operations 통합 서비스의 클라이언트 보기를 제공합니다. 서비스는 Marketing Operations 웹 사용자와 동시에 사용하도록 디자인됩니다.

웹 서비스는 하나의 API 호출인 executeProcedure를 지원합니다.

클라이언트는 이 웹 서비스 호출을 직접 작성합니다.

인증

인증은 필수가 아닙니다. 모든 클라이언트는 알려진 IBM UnicaMarketing Operations 사용자 PlanAPIUser와 연관됩니다. 이 특수 사용자의 보안 기능은 모든 웹 서비스 클라이언트를 필요로 하는 Marketing Operations 시스템 관리자에 의해 구성되었다고 가정합니다.

서비스의 향후 버전은 보안 클라이언트 인증을 위한 한층 일반적인 메커니즘이 제공될 수 있습니다.

로케일

지원되는 유일한 로케일은 현재 IBM UnicaMarketing Operations 시스템 인스턴스에 대해 구성된 로케일입니다. 서비스를 통해 액세스 가능한 모든 로케일 종속 데이터(메시지, 통화 등)는 시스템 로케일로 되어 있다고 가정합니다.

서비스의 향후 버전은 클라이언트가 Marketing Operations에 사용할 로케일을 알려주는 메커니즘을 제공할 수 있습니다.

상태 관리

웹 서비스는 *stateless*입니다. 즉, API 호출 사이에 서비스 구현에 의해 클라이언트마다 정보가 저장되지 않습니다. 이 기능은 효율적인 서비스 구현을 위해 제공되며 클러스터 지원을 단순화합니다.

데이터베이스 트랜잭션

웹 서비스는 클라이언트에 대해 편집 잠금이나 데이터베이스 트랜잭션을 노출하지 않습니다. 그러나 프로시저 실행의 효과가 *원자적*임을 보장합니다. 이 결과는 프로시저가 성공하거나 실패함을 의미합니다. 실패는 데이터베이스를 API가 전혀 호출되지 않은 경우와 동일한 상태로 유지합니다.

Marketing Operations 통합 웹 서비스 데이터 유형

이 절에서는 특정 서비스 바인딩 또는 프로그래밍 구현과는 관계없이 웹 서비스에서 사용되는 데이터 유형을 정의합니다.

다음 표기법이 사용됩니다.

- <*type*>: <*type definition*>는 단순 데이터 유형을 정의합니다. 예를 들면 다음과 같습니다.

  Handle: string

- <*type*>: [<*type definition*>]은 복합 데이터 유형 또는 데이터 구조를 정의합니다.
- <*type*>: { <*type definition*> }은 복합 데이터 유형 또는 데이터 구조를 정의합니다.

복합 유형 요소 및 API 매개변수는 이러한 유형을 사용하여 배열을 선언할 수 있습니다. 예를 들면 다음과 같습니다.

```
Handle [] handles
```

handles 유형은 Handle 유형의 배열입니다.

기본요소 유형

기본요소 유형은 SOAP 1.1 바인딩에 대한 지원을 단순화하기 위해 따르는 테이블에 정의된 유형으로 제한됩니다. 모든 유형은 배열로 선언될 수 있습니다(예: **String [ ]**). 본질적으로, **long**과 같은 2진 데이터 유형은 프로토콜 바인딩에 의해 문자열로 표시될 수 있습니다(예: SOAP). 그러나 이러한 표시는 클라이언트가 보는 유형 시맨틱, 허용 가능한 값 등에 영향을 주지 않습니다.

표 1. 기본요소 유형

API 유형	설명	SOAP 유형	Java 유형
boolean	부울 값: **true** 또는 **false**	xsd:boolean	boolean
dateTime	날짜 시간 값	xsd:datetime	Date
decimal	임의 정밀도, 10진수 값	xsd:decimal	java.math.BigDecimal
double	부호가 있는 배정밀도의 10진수 값	xsd:double	double
int	부호가 있는 32비트 정수 값	xsd:int	int
integer	부호가 있는 임의 정밀도의 정수 값	xsd:integer	java.math.BigInteger
long	부호가 있는 64비트 정수 값	xsd:long	long
string	유니코드 문자로 된 문자열	xsd:string	java.lang.String

MessageTypeEnum

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

MessageTypeEnum은 가능한 모든 메시지 유형을 정의하는 열거 유형입니다.

- INFORMATION: 정보 메시지
- WARNING: 경고 메시지
- ERROR: 오류 메시지

메시지

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

메시지는 웹 서비스 API 호출 결과를 정의하는 데이터 구조입니다. 현지화되지 않은 코드, 현지화된 텍스트 및 로그 세부사항에 대한 선택 가능 필드를 제공합니다. 현재, 현지화된 모든 텍스트는 IBM UnicaMarketing Operations 서버 인스턴스에 대한 로케일 세트를 사용합니다.

표 2. 메시지 매개변수

매개변수	설명
유형	메시지 유형을 설정하는 MessageTypeEnum
code	메시지에 대한 선택 가능 코드(문자열 형식)
localizedText	메시지와 연관시킬 선택 가능 텍스트 문자열
logDetail	선택 가능한 스택 추적 메시지

NameValue

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

NameValue는 이름-값 쌍을 정의하는 기본 복합 유형입니다. 또한 필요에 따라 값 배열을 구성하기 위해 서비스가 사용하는 선택 가능 순서를 정의합니다(순서 기준은 0입니다).

이름이 동일하면서 순서 번호가 다른 모든 NameValue는 값 배열로 변환되고 공통 이름과 연관됩니다.

배열 크기는 최대 순서 번호로 판별됩니다. 지정되지 않은 배열 요소에는 널값이 있습니다. 배열 순서 번호는 고유해야 합니다. 값과 해당 유형은 확장 유형으로 제공됩니다.

표 3. NameValue 매개변수

매개변수	설명
이름	NameValue 유형의 이름을 정의하는 문자열
순서	NameValue 내포 값에 대한 순서 번호를 설정하는 0 기준 정수

확장된 NameValue 유형은 다음과 같이 기본요소 유형마다 정의됩니다.

표 4. 확장된 NameValue 유형

확장 유형	설명
BigDecimalNameValue: NameValue [10진수 값]	값이 임의 정밀도의 10진수인 NameValue 유형
BigIntegerNameValue: NameValue [정수 값]	값이 임의 크기 정수인 NameValue 유형
BooleanNameValue: NameValue [부울 값]	값이 부울인 NameValue 유형
CurrencyNameValue: NameValue [문자열 로케일, 10진수 값]	일부 로케일에서 통화를 표시하기에 적합한 NameValue 유형. 로케일은 ISO 언어 코드입니다. 즉, ISO-639에서 정의한 2자 코드입니다. 현재 로케일은 IBM UnicaMarketing Operations 서버 인스턴스에 설정된 로케일 세트와 일치해야 합니다.
DateNameValue: NameValue [날짜 시간 값]	값이 날짜인 NameValue 유형
DecimalNameValue: NameValue [double 값]	값이 배정밀도의 10진수인 NameValue 유형
IntegerNameValue: NameValue [long 값]	값이 64비트 정수인 NameValue 유형
String NameValue: NameValue [문자열 값]	값이 문자열인 NameValue 유형

그리고 마지막으로, 다른 유형의 NameValue 세트를 정의하기 위해 필요할 경우 확장 NameValue 유형의 배열이 사용되도록 정의됩니다.

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

executeProcedure

구문

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

리턴

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

설명

이 메소드는 선택 가능한 매개변수 배열과 함께 지정된 프로시저를 호출합니다. 호출은 동기식으로 실행됩니다. 즉, 클라이언트를 블로킹하고 완료 시 결과를 리턴합니다.

매개변수

표 5. executeProcedure 매개변수

이름	설명
key	실행할 프로시저의 고유 키. 프로시저가 **key**에 바인드되는 경우 *RemoteException* 오류가 리턴됩니다.
jobid	이 프로시저 실행과 연관되는 작업을 식별하는 선택 가능 문자열. 이 문자열은 pass-through 항목이지만, 특정 프로시저의 실행을 위해 클라이언트 작업을 연결(tie)하는 데 사용할 수 있습니다.
paramArray	프로시저를 전달할 매개변수 배열. 하나 이상의 매개변수가 올바르지 않은 경우(잘못된 유형, 부적절한 값 등) 오류 상태 및 메시지가 리턴됩니다. 프로시저에 필요한 매개변수, 해당 유형 및 값 개수는 클라이언트가 판별합니다.

리턴 매개변수

표 6. executeProcedure 리턴 매개변수

이름	설명
status	정수 코드: • 0은 프로시저가 성공적으로 실행되었음을 표시합니다. • 정수는 오류를 표시합니다. 프로시저는 상태를 사용하여 여러 오류 수준을 표시합니다.
messages	0개 이상의 메시지 데이터 구조의 배열. **status**가 0인 경우 이 배열은 오류 메시지를 포함하지 않지만 정보 및 경고 메시지를 포함할 수 있습니다. **status**가 0이 아닌 경우 메시지는 혼합된 오류, 정보 및 경고 메시지를 포함합니다.

Marketing Operations 통합 서비스 WSDL

이 주제에서는 Marketing Operations 통합 서비스에 대한 WSDL(Web Services Definition Language)을 정의합니다. WSDL은 직접 정의되었으므로 웹 서비스 정의에서 최종 단어입니다.

Axis

이 웹 서비스 버전은 Axis2 1.5.2를 사용하여 WSDL 파일에서 웹 서비스 구현을 구성하는 서버측 클래스를 생성합니다. 사용자는 모든 버전의 Axis나 비Axis 기술을 마음대로 사용하여 제공된 WSDL을 통해 API와 통합하기 위한 클라이언트측 구현을 작성할 수 있습니다.

프로토콜 버전

프로토콜 버전은 다음과 같이 명시적으로 WSDL에 바인드됩니다.

- WSDL 이름의 일부로(예: `PlanIntegrationService1.0.wsdl`)

- WSDL targetNamespace의 일부로(예: `xmlns:tns="http://webservices.unica.com /MktOps/services/PlanIntegrationServices1.0?wsdl"`)

WSDL

하나의 WSDL 파일인 `PlanIntegrationServices1.0.wsdl`이 IBM UnicaMarketing Operations 통합 서비스와 함께 제공됩니다. WSDL은 `integration/examples/soap/plan` 디렉토리에서 전달됩니다. 빌드 스크립트 예제는 이 파일을 사용하여 적절한 클라이언트측 스터브를 생성하여 웹 서비스에 연결합니다.

제 3 장 IBM UnicaMarketing Operations 프로시저

*프로시저*는 작업 단위를 수행하는 IBM UnicaMarketing Operations에 의해 호스트되는 사용자 정의 또는 표준 Java 클래스입니다. 프로시저는 고객, 파트너 및 IBM Unica Professional Services가 임의 방법으로 Marketing Operations 비즈니스 로직을 확장할 수 있도록 하는 방법을 제공합니다.

프로시저는 단순한 프로그래밍 모델을 따르며, 잘 정의된 API를 사용하여 Marketing Operations에서 관리되는 구성 요소에 영향을 줍니다. 프로시저는 단순 조회 메커니즘 및 XML 기반 정의 파일을 통해 "발견됩니다". Marketing Operations는 해당되는 "클라이언트"의 필요에 따라 프로시저를 실행합니다. 예를 들어, 통합 요청(수신) 또는 트리거 실행(내부 또는 발신)에 대한 응답으로 실행합니다.

프로시저는 해당 클라이언트에 대해 동기식으로 실행됩니다. 결과는 클라이언트에 대해 직접적으로, 그리고 지속적인 감사 메커니즘을 통해 사용 가능하게 됩니다. 프로시저를 실행하면 기타 이벤트와 트리거가 Marketing Operations에서 실행될 수도 있습니다.

프로시저는 Java로 작성해야 합니다.

가정

프로시저에 관련된 다음 가정에 유의하십시오.

- 프로시저 구현 클래스는 별도의 클래스 트리 또는 jar 파일로 패키징되고 URL 경로를 통해 IBM UnicaMarketing Operations에 사용 가능하게 됩니다. 프로시저 실행 관리자는 독립적인 클래스 로더를 사용하여 필요에 따라 이 클래스를 로드합니다. 기본적으로, Marketing Operations는 다음 디렉토리에서 찾습니다.

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

  이 기본값을 변경하려면 **설정 > 구성 > Marketing Operations > umoConfiguration > integrationServices** 아래에서 **integrationProcedureClasspathURL** 매개변수를 설정하십시오.
- 프로시저 구현 클래스 이름은 다른 공급업체의 "Unica" 및 클래스와의 패키지 충돌을 피하기 위해 수락된 Java 이름 지정 규칙 규칙을 따릅니다. 특히, 고객은 **com.unica** 또는 **com.unicacorp** 패키지 트리 아래에 프로시저를 배치하지 않아도 됩니다.
- 프로시저 구현은 응용 프로그램 서버(최소 JRE 1.5.10)에서 IBM UnicaMarketing Operations에 의해 사용되는 Java 런타임 버전에 코딩됩니다.

- IBM UnicaMarketing Operations는 몇 개의 개방 소스 및 써드파티 라이브러리를 제공합니다. 또한 응용 프로그램 서버는 이 라이브러리의 다른 버전도 사용합니다. 일반적으로, 이 목록은 릴리스 사이에 변경됩니다.

  참고: 가능한 호환성 문제점을 피하려면 프로시저가 개방 소스, 써드파티 또는 응용 프로그램 서버별 라이브러리를 사용하면 안됩니다.

  그러나 프로시저에서, 또는 프로시저에 의해 가져온 2차 클래스에서 이러한 패키지가 사용되는 경우 해당 사용은 Marketing Operations 및/또는 응용 프로그램 서버에서 제공되는 패키지에 정확히 일치해야 합니다. 이러한 경우에 Marketing Operations의 나중 버전이 라이브러리를 업그레이드하거나 버리면 프로시저 코드를 재작업해야 합니다.
- 프로시저 구현 클래스는 보통 IBM UnicaMarketing Operations에서 사용되는 클래스 로딩 정책(일반적으로 **parent-last**)에 의해 로드됩니다. 응용 프로그램 서버는 Marketing Operations 프로시저에 적용하지만 반드시 필요하지는 않은 클래스를 다시 로드하기 위한 개발 도구와 옵션을 제공합니다.
- 프로시저는 해당되는 고유 상태에 대해 스레드 안전 상태여야 합니다. 즉, 해당되는 실행 메소드는 호출 간 내부 상태 변경사항에 의존할 수 없습니다.
- 프로시저는 저절로 스레드를 작성할 수 없습니다.

디자인

디자인은 원자적으로 수행해야 하는 단일 작업 단위를 생성하는 데 초점을 맞춰야 합니다. 이상적으로, 프로시저는 나중에 실행하기 위해 비동기식으로 예약할 수 있는 작업 시리즈를 수행합니다. 이러한 "실행 후 잊기" 통합 모델의 결과 두 시스템 모두에서 최소한의 로드가 발생합니다.

프로시저 구현 클래스는 IBM UnicaMarketing Operations API를 사용하여 Marketing Operations 구성 요소를 읽고 업데이트하며 서비스를 호출하는 등의 작업을 수행합니다. 다른 작업을 수행하기 위해 다른 Java 패키지를 사용할 수 있습니다.

참고: Marketing Operations의 나중 릴리스에서는 문서화된 클래스와 메소드만 지원됩니다. Marketing Operations에 있는 다른 모든 클래스 및 메소드는 논의 금지 사항으로 간주해야 합니다.

코딩하고 컴파일하고 나면, 프로시저 구현 클래스는 Marketing Operations에서 사용 가능해야 합니다. Marketing Operations 통합 서비스와 함께 제공된 빌드 스크립트는 컴파일된 프로시저를 기본 위치에 놓아야 합니다. 최종 개발 단계는 사용자 정의 프로시저를 발견하기 위해 Marketing Operations에서 사용되는 사용자 정의 프로시저 플러그인 정의 파일을 업데이트하는 것입니다.

프로시저는 **com.unica.publicapi.plan.plugin.procedure.IProcedure** 인터페이스를 구현하고 매개변수가 없는 생성자(보통 JavaBean 모델)를 가지고 있어야 합니다. 각 프로시저의 코딩 및 컴파일은 고객 선택(예: Eclipse, Borland JBuilder, Idea 등)의 Java IDE에서 완료됩니다. 다음 위치에서 개발자 툴킷으로 IBM UnicaMarketing Operations와 함께 샘플 코드가 제공됩니다.

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

구성

설정 > 구성 > Marketing Operations > umoConfiguration > integrationServices 아래에 있는 매개변수를 사용하여 Marketing Operations 통합 모듈을 구성합니다.

세부사항은 *Marketing Operations 설치 안내서*를 참조하십시오.

프로시저 수명 주기

프로시저의 실행 시간 수명 주기는 다음과 같습니다.

1. 발견 및 초기화
2. 선택(선택사항)
3. 실행
4. 영구 삭제

발견 및 초기화

IBM UnicaMarketing Operations는 특정 설치 인스턴스에 사용 가능한 모든 표준 및 사용자 정의 프로시저를 인식해야 합니다. 이 프로시저를 발견이라고 합니다.

참고: 표준 프로시저(Marketing Operations 엔지니어링 팀이 정의한 프로시저)가 내부적으로 알려져 있으므로 발견을 위한 조치가 필요하지 않습니다.

사용자 정의 프로시저는 프로시저 플러그인 정의 파일에서 정의됩니다. Marketing Operations 플러그인 관리자는 초기화 중에 이 파일을 읽습니다. 발견되는 프로시저마다 플러그인 관리자는 다음을 수행합니다.

1. 프로시저 인스턴스를 작성하고 해당 상태를 INSTANTIATED로 전이합니다.
2. 새 프로시저 감사 레코드를 작성합니다.
3. 프로시저 인스턴스를 작성할 수 없는 경우 해당되는 플러그인 설명 파일에서 발견되는 초기화 매개변수와 함께 **initialize()** 메소드가 호출됩니다. 이 메소드가 예외 처리를 하는 경우 상태가 로그되고 프로시저는 중단됩니다. 그렇지 않으면 프로시저 상태가 INITIALIZED 상태로 전이됩니다. 이제 실행 준비가 되었습니다.
4. 새 프로시저 감사 레코드를 작성합니다.

5. 프로시저를 초기화할 수 있는 경우 해당되는 **getKey()** 메소드가 호출되어 프로시저를 참조하기 위해 클라이언트가 사용하는 키가 판별됩니다. 이 키는 인스턴스와 연관되고 향후 조회를 위해 저장됩니다.

선택

간혹 IBM UnicaMarketing Operations는 사용자가 사용 가능한 프로시저 목록을 표시해야 할 수도 있습니다(예를 들어, 관리자가 트리거를 설정할 수 있도록 하기 위해). 이는 프로시저가 초기화된 후에만 수행됩니다. 프로시저의 **getDisplayName()** 및 **getDescription()** 메소드는 이 목적으로 사용됩니다.

실행

프로시저가 초기화된 후 IBM UnicaMarketing Operations는 프로시저를 실행하기 위한 요청을 수신합니다. 이는 다른 스레드에 대해 실행 중인 다른 프로시저(또는 동일한 프로시저)와 동시에 발생할 수 있습니다.

실행 시 프로시저 실행 관리자는 다음을 수행합니다.

1. 데이터베이스 트랜잭션을 시작하십시오.
2. 프로시저 상태를 EXECUTING으로 설정하십시오.
3. 새 프로시저 감사 레코드를 작성하십시오.
4. 클라이언트가 제공하는 실행 컨텍스트 및 실행 매개변수와 함께 프로시저의 **execute()** 메소드를 호출하십시오. 메소드 구현에서는 필요에 따라 Marketing Operations API가 사용되어, 편집 잠금이 획득되고 실행 컨텍스트와 함께 전달됩니다. execute 메소드가 예외 처리하는 경우 실행 관리자는 롤백하도록 트랜잭션에 표시합니다.
5. 실행 결과에 따라 트랜잭션을 커밋 또는 롤백하십시오. 프로시저 상태를 EXECUTED로 설정하십시오.
6. 미해결 편집 잠금을 해제하십시오.
7. 새 프로시저 감사 레코드를 작성하십시오.

참고: execute() 메소드는 프로시저 인스턴스 데이터를 변경하면 안됩니다.

영구 삭제

IBM UnicaMarketing Operations가 종료될 때 프로시저 플러그인 관리자는 로드된 모든 프로시저를 워크스루합니다. 발견되는 프로시저마다 관리자는 다음을 수행합니다.

1. 프로시저의 destroy() 메소드를 호출하여 인스턴스가 영구 삭제되기 전에 프로시저가 정리하도록 합니다.
2. 프로시저 상태를 FINALIZED로 변경합니다(실행할 수 없습니다).
3. 새 프로시저 감사 레코드를 작성합니다.

4. 프로시저의 인스턴스를 영구 삭제합니다.

데이터 잠금

IBM UnicaMarketing Operations는 비관적인 편집 잠금 스킴을 사용합니다. 즉, 한 번에 한 명의 사용자에게만 구성 요소 인스턴스에 대한 업데이트 액세스가 부여됩니다. GUI 사용자의 경우 이 잠금은 비주얼 탭 수준에서 수행됩니다. 어떤 경우에는 이는 인스턴스 하위 집합(예: 프로젝트 요약 탭)을 나타내지만, 다른 경우에는 많은 인스턴스(워크플로 탭)를 나타냅니다. 사용자가 잠금을 획득하면 다른 모든 사용자는 관련 데이터에 대한 읽기 전용 액세스로 제한됩니다.

참고: 편집 잠금은 데이터베이스 트랜잭션과 동일하지 않습니다.

구성 요소 인스턴스나 인스턴스 그룹에 대해 프로시저에서 작성된 변경사항 위에 다른 사용자가 부주의로 덮어쓰지 않도록 하기 위해, 프로시저는 구성 요소 데이터를 업데이트하기 전에 적절한 잠금을 획득해야 합니다. 이를 위해 프로시저의 **execute()** 메소드에 전달된 실행 컨텍스트 개체가 사용됩니다.

프로시저가 데이터를 업데이트하기 전에, 프로시저는 필요한 잠금마다 컨텍스트의 **acquireLock()** 메소드를 호출해야 합니다. 예를 들어, 프로시저가 프로젝트 및 연관된 워크플로를 업데이트하려는 경우 프로시저는 둘 다에 대해 잠금을 획득해야 합니다.

다른 사용자가 이미 잠근 경우 **acquireLock()** 메소드는 즉시 **LockInUseException** 예외 처리합니다. 충돌을 최소화하기 위해 프로시저는 개체를 업데이트하는 즉시 잠금을 해제해야 합니다.

실행 관리자는 실행 메소드가 리턴할 때 미해결 잠금을 자동으로 해제합니다. 어떤 경우에도 잠금은 데이터베이스 트랜잭션 수명 기간에만 보유됩니다. 즉, 잠금은 데이터베이스별 트랜잭션 제한시간을 초과하면 만료됩니다.

프로시저 트랜잭션

프로시저 실행 관리자는 프로시저 실행을 데이터베이스 트랜잭션으로 자동 랩핑하여, 프로시저 실행 결과를 기반으로 적절하게 커밋하거나 롤백합니다. 이로 인해 IBM UnicaMarketing Operations 데이터베이스 업데이트는 커밋되어 업데이트가 원자적이 될 때까지 다른 사용자에게 표시되지 않습니다.

프로시저 생성자는 프로시저 실행이 완료될 때까지 다른 사용자가 데이터베이스에 변경사항을 쓰지 않도록 필요한 편집 잠금을 획득해야 합니다.

결과 통신

프로시저의 **execute()** 메소드는 IBM UnicaMarketing Operations 프로시저 감사 테이블에 기록되고 지속되는 정수 상태 코드와 하나 이상의 메시지를 리턴합니다. 클라이언트는 상태 정보를 다른 형태로 통신할 수도 있습니다.

프로시저 로깅

IBM UnicaMarketing Operations에는 프로시저에 대한 별도의 로그 파일이 있습니다.

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

프로시저 실행 관리자는 각 프로시저의 수명 주기를 로그하고 감사 레코드를 작성합니다.

- **logInfo()**: 프로시저 로그에 정보용 메시지를 씁니다.
- **logWarning()**: 프로시저 로그에 경고 메시지를 씁니다.
- **logError**: 프로시저 로그에 오류 메시지를 씁니다.
- **logException()**: 예외에 대한 스택 추적을 프로시저 로그에 덤프합니다.

주요 Java 클래스

제공된 통합 devkit에는 공용 IBM UnicaMarketing Operations API 및 지원 클래스에 대한 Javadoc 세트가 들어 있습니다. 여기에는 가장 중요한 Javadoc이 나열되어 있습니다.

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure): 모든 프로시저가 구현해야 하는 인터페이스. 프로시저는 잘 정의된 수명 주기를 조사하고 Marketing Operations API에 액세스하여 작업을 수행합니다.
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure): 모든 트리거 프로시저가 구현해야 하는 인터페이스(마커 인터페이스).
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext): 실행 관리자에 의해 프로시저에 넘겨진 불투명한 컨텍스트 개체의 인터페이스. 이 개체에는 로깅 및 편집 잠금 관리에 대한 공용 메소드가 있습니다. 프로시저는 또한 모든 PlanAPI 호출에 이 개체를 전달합니다.
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI): Marketing Operations API에 대한 인터페이스입니다. 실행 컨텍스트는 **getPlanAPI()** 메소드를 제공하여 적절한 구현을 검색합니다.

프로시저 예제

다음 예제는 통합 웹 서비스 또는 트리거에서 프로젝트 상태를 변경하기 위한 표준 프로시저를 보여 줍니다.

참고: 샘플 프로시저와 해당되는 XML 정의는 수정하지 마십시오. 사용자가 IBM UnicaMarketing Operations를 업그레이드할 때 샘플 위에 덮어쓰므로 변경사항이 손실됩니다. 다른 디렉토리에서 모든 사용자 정의 프로시저를 작성하고 수정해야 합니다.

```
// 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
    }
}
```

프로시저 플러그인 정의 파일

이 주제에서는 프로시저 플러그인 정의 파일을 설명합니다. 이 파일은 IBM UnicaMarketing Operations에 호스트될 사용자 정의 프로시저에 대한 구현 클래스, 메타데이터 및 기타 정보를 정의합니다. 기본적으로, 프로시저 플러그인 정의는 다음 경로에 있는 것으로 가정합니다.

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

이 파일은 아래에 설명된 정보를 포함하는 XML 문서입니다.

Procedures: 0개 이상의 **Procedure** 요소로 된 목록

Procedure: 프로시저를 정의하는 요소. 각 프로시저에는 다음 요소가 있습니다.

- **key**(선택사항): 프로시저에 대한 잠금 키를 정의하는 문자열. 이 키는 특정 Marketing Operations 인스턴스에 의해 호스트되는 모든 표준(IBM 제공) 및 사용자 정의 프로시저에 대해 고유해야 합니다. 키가 정의되지 않은 경우, 기본값은 **className** 요소의 완전한 버전입니다. 문자열 "uap"로 시작하는 이름은 IBM UnicaMarketing Operations에서 사용하기 위해 예약되어 있습니다.
- **className**(필수): 프로시저 클래스의 완전한 패키지 이름. 이 클래스는 IProcedure 클래스를 구현해야 합니다(com.unica.public.plan.plugin.procedure.IProcedure).
- **initParameters**(선택사항): 0개 이상의 initParameter 요소로 된 목록

  initParameter(선택사항): 프로시저의 initialize() 메소드에 전달할 매개변수. 이 요소에는 중첩 매개변수 이름, 유형 및 값 요소가 포함됩니다.
  - name: 매개변수 이름을 정의하는 문자열.
  - type: 매개변수 값의 유형을 정의하는 Java 랩퍼 클래스의 선택 가능한 클래스 이름. 다음 유형 중 하나여야 합니다.
    - java.lang.String(기본값)
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  - value: 해당 유형에 따라 속성 값의 문자열 양식.

제 4 장 IBM UnicaMarketing Operations API에 대한 정보

IBM UnicaMarketing Operations API는 실행 중인 Marketing Operations 인스턴스의 클라이언트 보기를 제공하는 façade입니다. Marketing Operations 기능의 하위 집합만 노출됩니다. API는 Marketing Operations 웹 사용자와 Marketing Operations 통합 서비스 WebService SOAP 요청 및 트리거에서 동시에 사용되도록 지정됩니다. API는 다음 유형의 작업을 지원합니다.

- 구성 요소 작성 및 삭제
- 발견(구성 요소 유형, 속성 값 등을 기준으로)
- 구성 요소 검사(해당 속성, 특수화된 링크 등을 통해)
- 구성 요소 수정사항

버전화 및 역호환성

이 API의 향후 버전은 주 버전 번호를 공유하는 모든 부 버전 및 유지보수 릴리스와 역호환 가능하게 됩니다. 그러나 비즈니스 또는 기술 케이스에서 x.0(도트 0) 주 릴리스의 이전 버전과의 호환성을 중단하는 것을 보장하는 경우 IBM에는 이를 수행할 수 있는 권한이 예약됩니다.

이 API의 주 버전 번호는 다음 변경사항이 작성되는 경우 증가합니다.

- 데이터 해석이 변경됨
- 비즈니스 로직이 변경됨(예: 서비스 메소드 기능이 변경됨)
- 메소드 매개변수 및/또는 리턴 유형이 변경됨

API의 부 버전 번호는 다음 변경사항이 작성되는 경우 증가합니다(이 변경사항은 정의 역호환성에 의한 것임).

- 새 메소드가 추가됨
- 새 데이터 유형이 추가되고 해당되는 사용이 새 메소드로 제한됨
- 새 요소가 열거 유형에 추가됨
- 인터페이스의 새 버전이 버전 접미부와 함께 정의됨

사용자 보안

인증은 프로시저의 실행 관리자에 의해 수행될 것으로 가정되고 인증된 사용자 정보는 모든 API에서 사용되는 실행 컨텍스트에 바인드됩니다. API는 인증된 사용자를 노출하지 않지만 필요에 따라 사용하기 위해 IBM UnicaMarketing Operations에 전달합니다.

그러나 인증된 사용자에게는 API에 의해 노출된 모든 작업을 수행할 수 있는 권한이 부여되지 않을 수 있습니다. 이러한 경우 API 메소드는 **AuthorizationException**을 예외 처리합니다.

로케일

이 버전에서 지원되는 유일한 로케일은 현재 IBM UnicaMarketing Operations 서버 인스턴스에 대해 구성된 로케일입니다. API를 통해 액세스 가능한 모든 로케일 종속 데이터(메시지, 통화 등)는 이 시스템 로케일로 되어 있다고 가정합니다.

상태 관리

이 API는 stateless입니다. 즉, 호출 사이에 API에 의해 클라이언트마다 정보가 저장되지 않습니다.

그러나 특정 API 호출은 IBM UnicaMarketing Operations에서 관리되는 기초적인 구성 요소 인스턴스 상태를 변경할 수 있으며, 이 상태 변경사항이 데이터베이스에 대해 지속될 수 있습니다.

데이터베이스 트랜잭션

이 API는 데이터베이스 트랜잭션을 클라이언트에 노출하지 않지만 실행 컨텍스트에 포함된 경우 이 정보를 사용할 수 있습니다. 트랜잭션이 시작된 경우 특정 프로시저에서 모든 API 호출의 효과는 원자적입니다. IBM UnicaMarketing Operations의 다른 사용자는 프로시저가 성공적으로 트랜잭션을 커밋해야 변경사항을 보게 됩니다.

데이터베이스를 업데이트하는 API 호출은 먼저 다른 Marketing Operations 사용자가 API 호출 과정 동안 기초적인 데이터를 수정하지 못하도록 편집 잠금을 획득해야 합니다. 다른 사용자는 API가 완료될 때까지 잠긴 구성 요소를 업데이트할 수 없습니다. 마찬가지로, 다른 Marketing Operations 사용자 또는 API 클라이언트는 API 호출이 완료되지 못하도록 원하는 데이터에 대해 잠금을 획득할 수 있습니다.

이벤트 처리

이 API를 통해 IBM UnicaMarketing Operations 구성 요소에 대해 수행되는 작업은 작업이 Marketing Operations 웹 사용자에 의해 수행된 경우와 동일한 이벤트를 생성합니다. 특히, 특정 이벤트를 기다리는 트리거는 두 경우 모두에서 실행됩니다. 특정 공지(예: 프로젝트 상태가 변경되는 경우)에 등록된 사용자에게는 API 호출과 웹 사용자 조치로 발생하는 상태 변경사항이 공지됩니다.

API 데이터 유형에 대한 정보

IBM UnicaMarketing Operations API에는 다음 공용 데이터 유형이 들어 있습니다.

- **ApprovalMethodEnum**: 승인 메소드에 대한 열거 유형
- **ApprovalStateEnum**: 승인 상태에 대한 열거 유형
- **AssetLibraryStateEnum**: 자산 라이브러리 상태에 대한 열거 유형
- **AssetStateEnum**: 자산 상태에 대한 열거 유형
- **AttachmentTypeEnum**: 첨부에 대한 열거 유형
- **AttributeMap**: 속성 및 해당 값을 포함하는 java 맵
- **BudgetPeriodEnum**: 예산 기간 유형에 대한 열거 유형
- **BudgetTypeEnum**: 예산 유형에 대한 열거 유형
- **Handle**: 특정 Marketing Operations 인스턴스에서 구성 요소 인스턴스 식별
- **InvoiceStateEnum**: 호출 상태에 대한 열거 유형
- **MonthEnum**: 역년에서 월에 대한 열거 유형
- **ProjectCopyTypeEnum**: 프로젝트 복사 옵션에 대한 열거 유형
- **ProjectStateEnum**: 프로젝트 상태에 대한 열거 유형
- **TaskStateEnum**: 작업 상태에 대한 열거 유형
- **QuarterEnum**: 역년에서 분기에 대한 열거 유형
- **WeekEnum**: 역년에서 주에 대한 열거 유형

각 데이터 유형의 가능한 값이 아래에 나와 있습니다.

열거 유형

ApprovalMethodEnum

ApprovalMethodEnum은 승인의 가능한 모든 메소드를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- SIMULTANEOUS
- SEQUENTIAL

ApprovalStateEnum

ApprovalStateEnum은 승인의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- NOT_STATED
- ON_HOLD
- IN_PROGRESS
- COMPLETED

- CANCELLED

AssetLibraryStateEnum

AssetLibraryStateEnum은 자산 라이브러리의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- ENABLED
- DISABLED

AssetStateEnum

AssetStateEnum은 자산의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- DRAFT
- LOCK
- FINALIZE
- ARCHIVE

AttachmentTypeEnum

AttachmentTypeEnum은 첨부 파일의 가능한 모든 유형을 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- URL
- FILE
- ASSET

BudgetPeriodEnum

BudgetPeriodEnum은 가능한 모든 예산 기간을 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- QUARTERLY
- MONTHLY
- WEEKLY
- YEARLY
- ALL

BudgetTypeEnum

BudgetTypeEnum은 가능한 모든 예산 유형을 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- TOTAL

- FORECAST
- COMMITTED
- ACTUAL
- ALLOCATED

InvoiceStateEnum

InvoiceStateEnum은 송장의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- DRAFT
- PAYABLE
- PAID
- CANCELLED

MonthEnum

MonthEnum은 월에 대해 가능한 모든 값을 정의하는 열거 유형입니다.

ProjectCopyTypeEnum

ProjectCopyTypeEnum은 프로젝트 복사의 가능한 모든 유형을 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- COPY_USING_PROJECT_METRICS
- COPY_USING_TEMMPLATE_METRICS

이 프로젝트 및 작업 상태에 대한 세부 정보는 *Marketing Operations 사용자 안내서*를 참조하십시오.

ProjectStateEnum

ProjectStateEnum은 프로젝트 또는 요청의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- NOT_STARTED
- IN_PROGRESS
- ON_HOLD
- IN_RECONCILIATION
- LATE: 프로젝트가 예약된 시작 날짜에 의해 시작되지 않았음을 표시합니다.
- OVERDUE: 프로젝트가 예약된 종료 날짜 이전에 완료되지 않았음을 표시합니다.
- DRAFT
- SUBMITTED

- RETURNED
- ACCEPTED
- COMPLETED
- CANCELLED
- DELETED

TaskStateEnum

TaskStateEnum은 워크플로 작업의 가능한 모든 상태를 정의하는 열거 유형입니다. 가능한 값은 다음과 같습니다.

- PENDING
- ACTIVE
- FINISHED
- SKIPPED
- DISABLED

QuarterEnum

QuarterEnum은 분기에 대해 가능한 모든 값을 정의하는 열거 유형입니다.

WeekEnum

WeekEnum은 주에 대해 가능한 모든 값을 정의하는 열거 유형입니다.

핸들(Handle)

핸들은 특정 Marketing Operations 인스턴스에서 구성 요소 인스턴스를 참조하는 특수한 URL 개체입니다. 핸들에는 구성 요소 유형, 내부 데이터 ID, 인스턴스 기본 URL 등이 포함됩니다. API에서 사용되거나 생성되는 핸들은 잘라서 붙이고, 전자 메일로 보내고, 다른 프로시저에서 매개변수로 사용되는 등의 Marketing Operations GUI에서 구성 요소 보기를 탐색하기 위해 사용할 수 있는 전체 URL로 표면화할 수 있습니다.

핸들은 특정 Marketing Operations 서비스 인스턴스나 클러스터된 인스턴스에 대해서만 올바르지만, 배치된 서비스의 수명 시간 동안은 올바르도록 보장됩니다. 결과적으로, 핸들은 나중 참조를 위해 파일로 저장될 수 있지만 다른 Marketing Operations 인스턴스에서 구성 요소에 액세스하기 위해 사용할 수 없습니다(동일한 머신에서도). 그러나 Marketing Operations는 서로 다른 기본 URL을 현재 인스턴스에 매핑하여 다른 머신에 대한 인스턴스 재할당을 수용하기 위한 메커니즘을 제공합니다(예를 들어, 원래 머신이 잘못 작동하는 경우에).

핸들은 클라이언트에 독립적입니다. 예를 들어, 트리거는 핸들을 프로시저에 전달할 수 있고, 이 프로시저는 써드파티 시스템에 대한 SOAP 호출에 매개변수로 이 핸들을 사용합니다. 이 시스템은 다시 Marketing Operations에 대해 SOAP 요청을 발행하여 속성을 업데이트하는 프로시저를 호출합니다.

핸들 클래스에는 URL의 다양한 유형에서 핸들을 작성하기 위한 팩토리 메소드가 있습니다.

승인 개체:

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

자산 라이브러리 개체:

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

자산 폴더 개체:

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

자산 개체:

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

첨부 개체:

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

재무 계정 개체:

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

송장 개별 항목 개체:

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

송장 개체:

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

마케팅 개체:

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

마케팅 개체 표 형태:

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

마케팅 개체 표 형태 행:

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

프로젝트:

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

프로젝트 표 형태:

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

프로젝트 표 형태 행:

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

프로젝트 개별 항목 개체:

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

팀 개체:

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

사용자 개체:

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

워크플로 작업:

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

속성 맵

속성 맵은 속성만 포함하는 맵입니다. *name* 속성은 맵 항목 키이고 *values* 배열(복수형)은 맵 항목 값입니다.

속성 맵에는 다음 필드가 있습니다.

- *Name*: 속성의 프로그램 이름. 이 이름은 속성이 발생하는 구성 요소 인스턴스에서 속성에 액세스하기 위한 고유 키 역할을 합니다.

  참고: *Name*은 반드시 GUI에서 사용자에게 표시되는 표시 이름은 아닙니다. 템플리트로 작성된 구성 요소(예: 프로젝트 또는 워크플로 작업)의 경우 속성 이름은 템플리트 요소 정의로 지정되므로 고유해야 합니다. 다른 구성 요소의 경우 속성 이름은 일반적으로 서버측 구성 요소 인스턴스에서 프로그램 방식으로 파생됩니다(예를 들어, Java 자체 조사를 통해).

  참고: 관례적으로, 사용자 정의 속성에는 편집 가능한 버전이 정의된 양식의 이름이 포함됩니다(<form_name>.<attribute_name>).

- *values*: 0개 이상의 속성 값이 포함된 Java 개체 배열. 각 값의 유형은 동일해야 하고 Marketing Operations에 정의된 속성 유형과 일치해야 합니다. 다음 Java 랩퍼 및 Marketing Operations 유형만 지원됩니다.
  - AssetLibraryStateEnum: AssetLibraryStateEnum 열거 유형 값
  - AssetStateEnum: AssetStateEnum 열거 유형 값
  - AttachmentTypeEnum: AttachmentTypeEnum 열거 유형 값
  - AttributeMap: 속성을 보유하는 맵
  - BudgetPeriodEnum: BudgetPeriodEnum 열거 유형 값
  - BudgetTypeEnum: BudgetTypeEnum 열거 유형 값
  - Handle: 구성 요소 인스턴스, 표 형태 행, 속성 등에 대한 참조
  - InvoiceStateEnum: InvoiceStateEnum 열거 유형 값
  - java.io.File: 파일 표시
  - java.lang.Boolean: 부울 값(True 또는 False)
  - java.lang.Double: 배정밀도 10진수 값
  - java.lang.Float: 단정밀도 10진수 값
  - java.lang.Integer: 32비트 정수 값
  - java.lang.Long: 64비트 정수 값
  - java.lang.Object: 일반 Java 개체
  - java.lang.String: 0개 이상의 유니코드 문자로 된 문자열
  - java.math.BigDecimal: 부호 있는 임의 정밀도 10진수 값. 통화에 적합합니다. 값 해석은 클라이언트에 대한 통화 로케일에 따라 다릅니다.
  - java.math.BigInteger: 임의 정밀도 정수 값
  - java.net.URL: URL(Universal Resource Locator) 개체
  - java.util.ArrayList: 개체 목록
  - java.util.Calendar: 특정 로케일에 대한 날짜 시간 값
  - java.util.Date: 날짜 시간 값. 이 유형은 더 이상 사용되지 않습니다. 대신 java.util.Calendar 또는 java.util.GregorianCalendar를 사용하십시오.
  - java.util.GregorianCalendar: GregorianCalendar는 java.util.Calendar의 구체적인 서브클래스이며 전세계적으로 대부분 사용하는 표준 달력 시스템을 제공
  - MonthEnum: MonthEnum 열거 유형 값
  - ProjectStateEnum: ProjectStateEnum 열거 유형 값
  - QuarterEnum: QuarterEnum 열거 유형 값
  - TaskStateEnum: TaskStateEnum 열거 유형 값
  - WeekEnum: WeekEnum 열거 유형 값

속성의 메타데이터(예: 현지화된 표시 이름 및 설명)는 속성 및 해당되는 상위 개체 인스턴스와 연관된 템플리트에 의해 정의됩니다. 속성은 프로젝트 이름, 코드 및 시작 날짜와 같은 필수 및 선택 가능 개체 인스턴스 속성을 노출하기 위한 단순하지만 확장 가능한 메커니즘을 제공합니다.

참고: 날짜를 구현하기 위해, 사용자는 java.util.Calendar 또는 java.util.GregorianCalendar를 선택할 수 있습니다.

공통 예외

이 주제에서는 API에서 발생하는 공통 예외에 대해 설명합니다.

- AuthorizationException: 클라이언트의 실행 컨텍스트와 연관된 사용자에게는 요청한 작업을 수행할 수 있는 권한이 없습니다. API 메소드에서 발생할 수 있으므로 선언되어 있지 않습니다.
- DataException: IBM UnicaMarketing Operations의 기초적인 데이터베이스 계층에서 예외가 발생했습니다. 세부사항은 SQL 로그를 확인하십시오.
- InvalidExecutionContextException: API 메소드에 전달된 실행 컨텍스트에 문제점이 있습니다(예: 메소드가 올바르게 초기화되지 않았음). API에서 발생할 수 있으므로 선언되어 있지 않습니다.
- NotLockedException: 먼저 필수 잠금을 획득하지 않고 구성 요소 데이터를 업데이트하려고 했습니다. IExecutionContext의 acquireLock() 메소드를 참조하십시오.

API 메소드

공용 API 메소드에 특정한 정보는 JavaDocs API 문서 파일의 iPlanAPI 클래스를 참조하십시오. 이러한 파일은 Marketing Operations에 로그인하여 페이지에서 **도움말 > 제품 문서**를 선택한 다음 `<version>PublicAPI.zip` 파일을 다운로드하여 볼 수 있습니다.

IBM Unica 기술 지원 담당자에게 문의

문서를 참조해도 문제점을 해결할 수 없는 경우, 회사의 지정된 지원 담당자가 IBM Unica 기술 지원 담당자와의 통화를 기록할 수 있습니다. 이 절의 정보를 사용하여 문제점을 효율적으로 해결하십시오.

회사의 지정된 지원 담당자가 아닌 경우에는 IBM Unica 관리자에게 문의하여 정보를 얻을 수 있습니다.

정보 수집

IBM Unica 기술 지원 담당자에게 문의하기 전에 다음 정보를 수집해야 합니다.

- 문제점의 특성에 대한 간단한 설명
- 해당 문제점이 발생할 때 표시되는 자세한 오류 메시지
- 문제점을 재현할 수 있는 자세한 단계
- 관련 로그 파일, 세션 파일, 구성 파일 및 데이터 파일
- "시스템 정보"에서 설명한 방법에 따라 얻을 수 있는 제품 및 시스템 환경에 대한 정보

시스템 정보

IBM Unica 기술 지원 담당자와 통화할 때 환경 정보를 요청하는 경우가 있습니다.

문제점 때문에 로그인이 불가능한 경우 외에는, 설치된 IBM Unica 응용 프로그램에 대한 정보를 제공하는 제품 정보 페이지에서 이러한 정보 대부분을 얻을 수 있습니다.

도움말 > 제품 정보를 선택하여 제품 정보 페이지에 액세스할 수 있습니다. 제품 정보 페이지에 액세스할 수 없는 경우에는 각 응용 프로그램의 설치 디렉토리 아래에 있는 `version.txt` 파일을 사용하여 모든 IBM Unica 응용 프로그램의 버전 번호를 알 수 있습니다.

IBM Unica 기술 지원 담당자에게 문의

IBM Unica 기술 지원 담당자에게 문의하는 방법은 IBM Unica 제품 기술 지원 웹 사이트(http://www.unica.com/about/product-technical-support.htm)를 참조하십시오.

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

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

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 명시적 또는 묵시적인 일체의 보증 없이 이 책을 "현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함)간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등)하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 단계의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 비IBM 제품을 반드시 테스트하지 않았으므로, 이들 제품과 관련된 성능의 정확성, 호환성 또는 기타 주장에 대해서는 확인할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

여기에 나오는 모든 IBM의 가격은 IBM이 제시하는 현 소매가이며 통지 없이 변경될 수 있습니다. 실제 판매가는 다를 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용 없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 본 샘플 프로그램은 일체의 보증 없이 "현상태 대로" 제공됩니다. IBM은 귀하의 샘플 프로그램 사용과 관련되는 손해에 대해 책임을 지지 않습니다

이 정보를 소프트카피로 확인하는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

상표

IBM, IBM 로고 및 ibm.com은 전세계 여러 국가에 등록된 IBM Corp.의 상표 또는 등록상표입니다. 기타 제품 또는 서비스 이름은 IBM 또는 타사의 상표입니다. 현재 IBM 상표 목록은 웹의 『저작권 및 상표 정보』(www.ibm.com/legal/copytrade.shtml)에 있습니다.