Version 9 Release 1.2
September 23, 2015

IBM Marketing Operations Integration Module

Note
Before using this information and the product it supports, read the information in "Notices" on page 33.

This edition applies to version 9, release 1, modification 2 of IBM Marketing Operations and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2002, 2015.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. What is IBM Marketing Operations Integration Services? 1

What are the requirements for Marketing Operations Integration Services? 2

IBM Marketing Operations Integration Services basics 3

Installing Integration Services 5

Software developer kit contents 5

Hosted JavaDocs 6

Marketing Operations documentation and help . .. 6

Chapter 2. Marketing Operations Integration Webservice 9

Marketing Operations Integration Services WSDL .. 9

executeProcedure 9

Marketing Operations Integration Webservice data types 10

Chapter 3. IBM Marketing Operations procedures 15

Assumptions 15

Configuration parameters. 17

Design 17

Procedure lifecycle 18

Key Java classes 19

Data locking 19

Procedure transactions. 20

Procedure communication 20

Procedure logging 20

Procedure plug-in definition file 21

Chapter 4. IBM Marketing Operations API 23

Contents of the IBM Marketing Operations API .. 23

API Interfaces 23

Common exceptions 24

Handles 24

Attribute Map 26

Enumerated data types 27

Before you contact IBM technical support 31

Notices 33

Trademarks 35

Privacy Policy and Terms of Use Considerations .. 35

© Copyright IBM Corp. 2002, 2015

Chapter 1. What is IBM Marketing Operations Integration Services?

IBM®Marketing Operations Integration Services combines the Marketing Operations Integration Webservice, API procedures, and triggers to extend business capabilities.

IBM Marketing Operations Integration Services is a composite of the following.

- **Marketing Operations Integration Webservice**

  Integration Services provide a way for Marketing Operations customers and IBM Professional Services to integrate Marketing Operations with other applications that run in their environment.
- **Marketing Operations procedures and API**

  Custom procedures can be defined within Marketing Operations to extend Marketing Operations business logic in arbitrary ways. After you define procedures, these procedures can be the targets for the Integration Services webservice calls from other applications. Procedures also can be defined to send messages to other applications.
- **Marketing Operations triggers**

  Triggers can be associated with events and procedures in Marketing Operations. When one such event occurs, the associated trigger is run.

Versions and backwards-compatibility

Future versions of the integration services will be backwardly compatible with all minor and maintenance releases that share a major version number. However, IBM reserves the right to break compatibility with an earlier version for dot zero (x.0) major releases if the business or technical case warrants.

The major version number of this API is incremented if any of the following changes are made.

- Data interpretation changes
- Business logic changes (for example, service method functions changes)
- Method parameters, return types, or both change

The minor version number of the API is incremented if any of the following changes are made. These changes are compatible with an earlier version by definition.

- New method added
- New data type is added and its usage is restricted to a new method
- New element added to an enumerated type
- A new version of an interface is defined with a version suffix

Authentication

Authentication is not required; all clients are associated with a known IBM Marketing Operations user named PlanAPIUser. A system administrator configures the security capabilities of this special user to meet the needs of all webservice clients.

© Copyright IBM Corp. 2002, 2015

Locale

The only locale that is supported is the locale that is currently configured for the IBM Marketing Operations system instance. All locale-dependent data, such as messages and currency, are assumed to be in the system locale.

State management

The API and webservice are *stateless*; no per-client information is saved by the service implementation across API calls. This feature provides for an efficient service implementation and simplifies cluster support.

Database transactions

Marketing Operations Integration Services does not show database transactions to the client, but uses such information if it is included in the execution context. If a transaction is started, then the effect of all API calls within a particular procedure is atomic. In other words, a failed API call leaves the database in the same state as if the API was never called at all. Other users of Marketing Operations do not see the changes until the procedure successfully completes the transaction.

API calls that update the database must first acquire an edit lock to prevent other users from modifying the underlying data during the API calls. Other users cannot update locked components until the API call completes. Likewise, the next Marketing Operations user or API client must acquire the lock on the data before another API call is submitted.

Event processing

Operations on IBM Marketing Operations components through the API generate the same events as if a Marketing Operations user did the operation. Users that subscribed to certain notifications, such as project state changes, are notified of state changes that result from API calls and user actions.

What are the requirements for Marketing Operations Integration Services?

Marketing Operations Integration Services has the following requirements.

Marketing Operations Integration Services must:

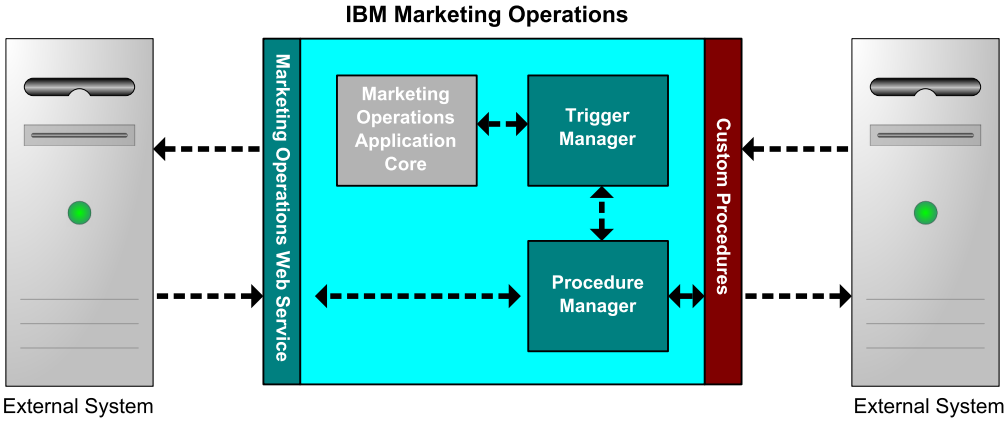
- Loosely couple system integration.
- Provide a mechanism for customer applications to affect Marketing Operations through webservice calls.
- Provide a mechanism for customer applications to be notified of certain events in Marketing Operations.
- Provide a simple programming model that is easy to understand and use.
- Be robust when recovering from failure.
- Guarantee data integrity.
- Integrate with, and minimize the effect on, existing Marketing Operations GUI-based customers.
- Provide fine-grained access to Marketing Operations components while insulating programmers from underlying implementation details.

IBM Marketing Operations Integration Services basics

You use IBM Marketing Operations Integration Services to create custom procedures. You can use these procedures to trigger external events when certain events occur within Marketing Operations. You can use these procedures to run Marketing Operations functions from external systems or programs.

The API interface interacts with IBM Marketing Operations at the programmatic level, in the same way the GUI interfaces with Marketing Operations at a user level. Using the API, you construct procedures. Using these procedures, you communicate between Marketing Operations and external systems. The Marketing Operations Webservice is the container object for the procedures, API, and triggers.

The architecture of the Marketing Operations Integration Services is shown here.

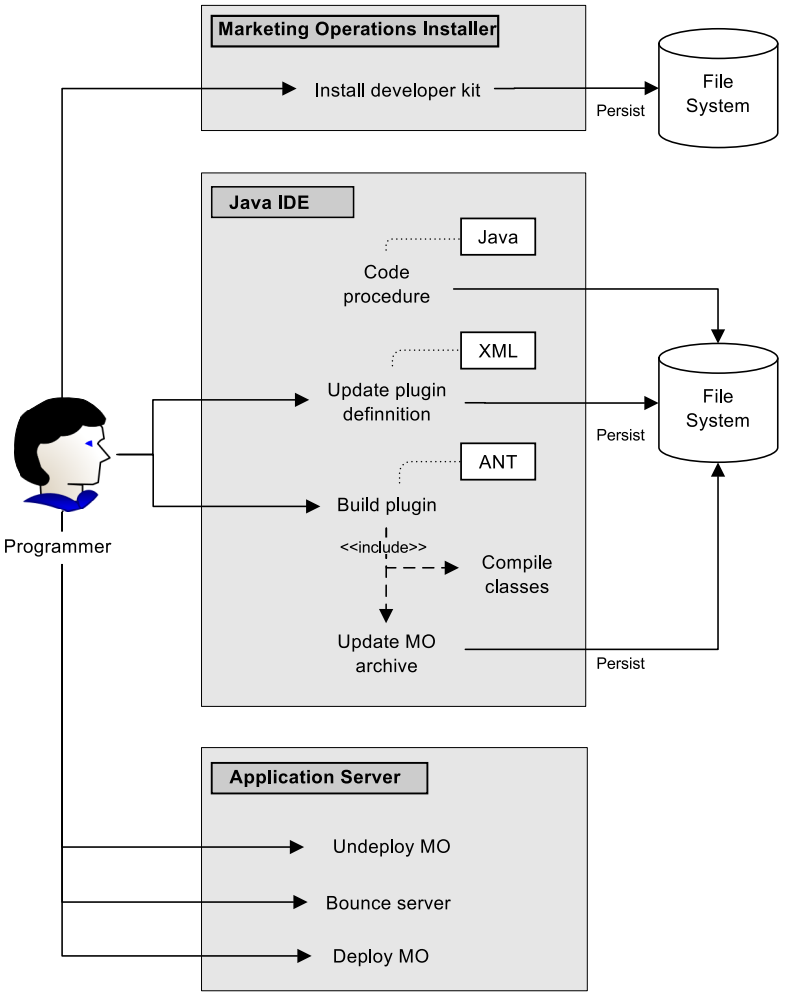


The following are key components of the Integration Services.

- Marketing Operations Procedure Manager: extends the business logic by interacting with Marketing Operations through the API.
- Marketing Operations Trigger Manager: associates a condition (for example, the state change of a marketing object) with an action (a procedure to run when the condition for the trigger is met).

Methods

You use the components of IBM Marketing Operations Integration Services to develop custom procedures, as shown in the following diagram.



After you install the developer's kit, you follow these basic steps:

1. Code the custom procedure. Currently, you must use Java™.
2. Update the plug-in definition in the XML definition file.
3. Build the plug-in:
   a. Compile the necessary classes.
   b. If you are using a third-party library that is not in the Marketing Operations archive, bundle the library inside the `plan.war` file and redeploy.
4. Restart Marketing Operations. Changes to the procedure classes are applied when you restart the application server.

   Note: If you change the **plan.war** file, you must undeploy and redeploy Marketing Operations with the new **plan.war** file. Undeploy and redeploy Marketing Operations if you use a third-party library that is not in the Marketing Operations archive and you edit the **plan.war** file.

Basic Example to communicate between IBM Marketing Operations and the API

The following basic example describes establishing communication between the API and Marketing Operations. It does not do any useful work; it performs a round trip between Marketing Operations and the Integration Services.

This example uses portions of the example procedures included with the Marketing Operations Integration Services developer's kit. Specifically, you can find the code that is referenced here in the following files.

- `PlanClientFacade.java`
- `PlanWSNOOPTestCase.java`

The `noop` method is a webservice call to Marketing Operations. It is defined in the PlanClientFacade class, and passes null values in an 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);
}
```

The procedure testExecuteProcedure calls the `noop` method from PlanClientFacade to establish a round trip with the Marketing Operations application.

```
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.");
}
```

For details of NameValueArrays, ProcedureResponse, and other listed methods and data types, refer to the *Marketing Operations Integration Module* and the JavaDocs.

Installing Integration Services

The IBM Marketing Operations Integration Services module is a separate, paid component. If you purchase the Integration Services module, you must install it.

1. Download the IBM Marketing Operations Integration Services installers.
2. The IBM EMM installers detect the Integration Services module.
3. The installer sets configuration properties at **Marketing Operations | umoConfiguration | integrationServices | enableIntegrationServices**. You can customize your installation by changing configuration parameters. For more information, see "Configuration parameters" on page 17.

Software developer kit contents

Typically, all the Marketing Operations Integration Services components are installed under a folder labeled "devkits."

The software developer kit contains documentation, Javadoc containing all publicapi classes and interfaces, and example code.

Example code is installed in the following folders.

- The **build** folder contains scripts to build and deploy custom procedures.
- The **Classes** folder contains the compiled procedure classes.

Users must deploy the compiled classes of their custom procedures at the path that is specified by the configuration parameter **integrationProcedureClasspathURL**. Then, the IBM Marketing Operations Procedure Manager loads them as specified in the `procedure-plugins.xml` configuration file.

- The **lib** folder contains the necessary libraries for developing and compiling custom procedures.
- The **src** folder contains source files for custom procedures. Users can place custom procedures to be started as triggers or web-services here.
  - The **src/procedure** folder contains `procedure-plugins.xml` configuration file. Every custom procedure that runs as a trigger based an event or through an external web-service must have an entry in this file. The entries must contain a fully qualified class path of procedure and required initialization parameters.
  - The **src/procedure** folder also contains some sample procedures that are included with IBM Marketing Operations. These procedures can be used to understand and develop your custom procedures.

    Place custom procedures under the **src** directory in a new folder structure, such as `com/<mycompany>/<mypackage>`. Do not place custom procedures in the sample procedures folder.
  - The **src/soap** folder contains sample web service clients that are developed in Java. Use these samples as a starting point for developing web service-based clients for Integration Services. This folder also contains binary scripts to start sample clients over the command line.

Hosted JavaDocs

For specific information about the public API methods, refer to the iPlanAPI class in the JavaDocs API documentation files.

These files are available in the following ways:

- By the files in the `<IBM_EMM>/<MarketingOperations_Home>/devkits/integration/javadocs` directory on the server that hosts Marketing Operations.
- By logging in to Marketing Operations and selecting **Help > Product Documentation** from any page, and then downloading the `IBM <version>PublicAPI.zip` file.

Marketing Operations documentation and help

Different people in your organization use IBM Marketing Operations to accomplish different tasks. Information about Marketing Operations is available in a set of guides, each of which is intended for use by team members with specific objectives and skill sets.

The following table describes the information available in each guide.

Table 1. Guides in the Marketing Operations documentation set.

The following three-column table describes tasks in one column, guide names in the second column, and audience in the third column.

If you	See	Audience
• Plan and manage projects • Establish workflow tasks, milestones, and personnel • Track project expenses • Get reviews and approvals for content • Produce reports	*IBM Marketing Operations User's Guide*	• Project managers • Creative designers • Direct mail marketing managers
• Design templates, forms, attributes, and metrics • Customize the user interface • Define user access levels and security • Implement optional features • Configure and tune Marketing Operations	*IBM Marketing Operations Administrator's Guide*	• Project managers • IT administrators • Implementation consultants
• Create marketing campaigns • Plan offers • Implement integration between Marketing Operations and Campaign • Implement integration between Marketing Operations and IBM Digital Recommendations	*IBM Marketing Operations and IBM Integration Guide*	• Project managers • Marketing execution specialists • Direct marketing managers
• Learn about new system features • Research known issues and workarounds	*IBM Marketing Operations Release Notes*	Everyone who uses Marketing Operations
• Install Marketing Operations • Configure Marketing Operations • Upgrade to a new version of Marketing Operations	*IBM Marketing Operations Installation Guide*	• Software implementation consultants • IT administrators • Database administrators
Create custom procedures to integrate Marketing Operations with other applications	*IBM Marketing Operations Integration Module* and the API JavaDocs available when you click **Help > Product Documentation** in Marketing Operations, and then download the `IBM<version>PublicAPI.zip` file	• IT administrators • Database administrators • Implementation consultants
Learn about the structure of the Marketing Operations database	*IBM Marketing Operations System Schema*	Database administrators

Table 1. Guides in the Marketing Operations documentation set (continued).

The following three-column table describes tasks in one column, guide names in the second column, and audience in the third column.

If you	See	Audience
Need more information while you work	• Get help and search or browse the *Marketing Operations User's, Administrator's,* or *Marketing Operations Installation* guides: Click **Help > Help for this page** • Access all of the Marketing Operations guides: Click **Help > Product Documentation** • Access guides for all IBM Enterprise Marketing Management (EMM) products: Click **Help > All IBM EMM Suite Documentation**	Everyone who uses Marketing Operations

Chapter 2. Marketing Operations Integration Webservice

The webservice provides a client view of the Marketing Operations Integration Services, which is part of the deployment of the IBM Marketing Operations server. The service is used concurrently with Marketing Operations web users.

The webservice supports one API call, executeProcedure.

A client makes this webservice call directly.

Marketing Operations Integration Services WSDL

The Web Services Definition Language (WSDL) was defined by hand and is the final word on the webservice definition.

Axis

This version of the webservice uses Axis2 1.5.2 to generate the server-side classes that make up the web service implementation from the WSDL file. Users can use any version of Axis, or a non-Axis technique, to create a client side implementation for integrating with the API from the supplied WSDL.

Protocol version

The version of the protocol is explicitly bound to the WSDL as follows:

- As part of the WSDL name, for example, `PlanIntegrationService1.0.wsdl`
- As part of the WSDL targetNamespace, for example, `xmlns:tns="http://webservices.unica.com /MktOps/services/PlanIntegrationServices1.0?wsdl"`

WSDL

One WSDL file is provided with IBM Marketing Operations Integration Services: `PlanIntegrationServices1.0.wsdl`. The WSDL is delivered in the `integration/examples/soap/plan` directory. The example build script uses this file to generate the appropriate client-side stubs to connect to the webservice.

executeProcedure

executeProcedure is the on API call that is supported by the webservice.

Syntax

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

Returns

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

Description

This method invokes the specified procedure with an optional array of parameters. The call executes synchronously; that is, it blocks the client and returns the result upon completion.

© Copyright IBM Corp. 2002, 2015

Parameters

Table 2. executeProcedure parameters

Name	Description
key	The unique key of the procedure to run. A *RemoteException* error is returned if no procedure is bound to **key**.
jobid	Optional string that identifies the job that is associated with this procedure execution. This string is a pass-through item, but it can be used to tie client jobs to the execution of a particular procedure.
paramArray	An array of parameters to pass to the procedure. An error status and message is returned if one or more of the parameters is invalid (such as, the wrong type or an incorrect value). It is up to the client to determine the parameters, their types, and the number of values that are required by the procedure.

Return Parameters

Table 3. executeProcedure return parameters

Name	Description
status	An integer code: • 0 indicates that the procedure ran successfully • an integer indicates an error Procedures can use the status to indicate different levels of errors.
messages	An array of zero or more message data structures. If **status** is 0, this array does not contain ERROR messages, but might contain INFORMATION and WARNING messages. If **status** is non-zero, messages can contain any mix of ERROR, INFORMATION, and WARNING messages.

Marketing Operations Integration Webservice data types

The data types that are used by the webservice are independent of any particular service binding or programming implementation.

The following notation is used.

- <*type*>: <*type definition*> defines a simple data type. For example:
  Handle: string
- <*type*>: [<*type definition*>] defines a complex data type or a data structure.
- <*type*>: { <*type definition*> } defines a complex data type or a data structure.

Complex type elements and API parameters can use these types to declare arrays. For example:

```
Handle [] handles
```

The type, handles, is an array of Handle types.

Primitive types

Primitive types are restricted to the types defined in the table that follows to simplify support for SOAP 1.1 bindings. All types can be declared as arrays, for example, **String [ ]**. Inherently, binary data types, such as **long**, can be represented

as strings by a protocol binding (for example, SOAP). This representation, however, has no effect on the semantics of the type, permissible values, and so on, as seen by the client.

Table 4. Primitive types

API Type	Description	SOAP Type	Java Type
Boolean	Boolean value: **true** or **false**	xsd:Boolean	Boolean
dateTime	A date time value	xsd:datetime	Date
decimal	An arbitrary-precision, decimal value	xsd:decimal	java.math.BigDecimal
double	A double-precision, signed, decimal value	xsd:double	double
int	A signed, 32-bit, integer value	xsd:int	int
integer	An arbitrary-precision, signed, integer value	xsd:integer	java.math.BigInteger
long	A signed, 64-bit, integer value	xsd:long	long
string	A string of Unicode characters	xsd:string	java.lang.String

MessageTypeEnum

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

MessageTypeEnum is an enumerated type that defines all possible message types.

- INFORMATION: an informational message
- WARNING: a warning message
- ERROR: an error message

Message

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

Message is a data structure that defines the result of a webservice API call. It provides optional fields for a non-localized code, localized text, and log detail. Currently, all localized text uses the locale that is set for the IBM Marketing Operations server instance.

Table 5. Message parameters

Parameter	Description
type	A MessageTypeEnum, setting the type of the message.
code	An optional code, in string format, for the message.
localizedText	An optional text string to associate with the message.
logDetail	An optional stack trace message.

NameValue

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

NameValue is a base complex type that defines a name-value pair. It also defines an optional sequence that the service uses to construct value arrays as needed (the sequences are zero-based).

All NameValues with the same name, but different sequence numbers, are converted into an array of values and associated with the common name.

The array size is determined by the maximum sequence number; unspecified array elements have null values. Array sequence numbers must be unique. The value and its type are provided by the extended type.

Table 6. NameValue parameters

Parameter	Description
name	A string that defines the name of a NameValue type.
sequence	A zero-based integer that sets the sequence number for the NameValue implied value.

Extended NameValue types are defined for each primitive type, as follows:

Table 7. Extended NameValue types

Extended type	Description
BigDecimalNameValue: NameValue [decimal value]	A NameValue type whose value is an arbitrary-precision, decimal number.
BigIntegerNameValue: NameValue [integer value]	A NameValue type whose value is an arbitrarily sized integer.
BooleanNameValue: NameValue [Boolean value]	A NameValue type whose value is a Boolean.
CurrencyNameValue: NameValue [string locale, decimal value]	A NameValue type suitable for representing currency in some locale. Locale is an ISO Language Code, that is, the lowercase, two-letter codes as defined by ISO-639. Currently, the locale must agree with the locale set in the IBM Marketing Operations server instance.
DateNameValue: NameValue [datetime value]	A NameValue type whose value is a date.
DecimalNameValue: NameValue [double value]	A NameValue type whose value is a double-precision, decimal number.
IntegerNameValue: NameValue [long value]	A NameValue type whose value is a 64-bit integer.
String NameValue: NameValue [string value]	A NameValue type whose value is a string.

And finally, an array of the extended NameValue types is defined for use when you must define a set of NameValues of with different types.

```
     NameValueArrays: [
BooleanNameValue[]    booleanValues,
StringNameValue[]     stringValues,
IntegerNameValue[]    integerValues,
BigIntegerNameValue[] bigIntegooleanNameValue,
DecimalNameValue[]    decimalValues,
```

```
BigDecimalNameValue[] bigDecimalValues
DateNameValue[]       dateNameValues
CurrencyNameValue[]   currencyValues
     ]
```

Chapter 3. IBM Marketing Operations procedures

A "procedure" is a custom or standard Java class hosted by IBM Marketing Operations that does some unit of work. Procedures provide a way for customers and IBM Professional Services to extend Marketing Operations business logic in arbitrary ways.

Procedures follow a simple programming model with a well-defined API to affect components that are managed by Marketing Operations. Procedures are "discovered" through a simple lookup mechanism and XML-based definition file. Marketing Operations runs the procedures according to needs of their "clients." For example, in response to an integration request (incoming) or a trigger firing (internal or outgoing).

Procedures run synchronously with their client; results are made available directly to the client, and through a persisted auditing mechanism. The execution of a procedure can also cause other events and triggers to fire in Marketing Operations.

Procedures must be written in Java.

Assumptions

The procedure implementation classes are packaged into a separate classes tree or JAR file and made available to IBM Marketing Operations through a URL path.

Procedure implementation

The procedure execution manager uses an independent class loader to load these classes as needed. By default, Marketing Operations looks in the following directory.

<MarketingOperations_Home>/devkits/integration/examples/classes

To change this default, set the **integrationProcedureClasspathURL** parameter under **Settings > Configuration > Marketing Operations > umoConfiguration > integrationServices**.

The procedure implementation class name follows the accepted Java naming conventions, to avoid package collisions with "unica" and classes from other vendors. In particular, customers must not place procedures under the "com.unica" or "com.unicacorp" package tree.

The procedure implementation is coded to the Java runtime version used by IBM Marketing Operations on the application server (at least JRE 1.5.10).

The procedure implementation class is loaded by the class loading policy that is normally used by IBM Marketing Operations (typically **parent-last**). The application server might provide development tools and options to reload classes that would apply to Marketing Operations procedures, but that is not required.

© Copyright IBM Corp. 2002, 2015

Libraries

IBM Marketing Operations provides some open source and third-party libraries; application servers also use different versions of these libraries.

Generally, this list changes from release to release. For version 9.1.0, the following third-party libraries are supported.

- Ant 1.6.5 (`ant.jar`)
- Axis2 1.5.2 and dependencies
  - `axiom-api-1.2.9.jar`
  - `axiom-impl-1.2.9.jar`
  - `axis2-adb-codegen-1.5.2.jar`
  - `axis2-codegen-1.5.2.jar`
  - `axis2-adb-1.5.2.jar`
  - `axis2-kernel-1.5.2.jar`
  - `axis2-transport-http-1.5.2.jar`
  - `axis2-transport-local-1.5.2.jar`
  - `commons-codec.jar`
  - `commons-httpclient-3.1.jar`
  - `commons-logging.jar`
  - `httpcore-4.0.jar`
  - `neethi-2.0.4.jar`
  - `geronimo-stax-api_1.0_spec-1.0.1.jar`
  - `jaxrpc.jar`
  - `xlxpScanner.jar`
  - `xlxpScannerUtils.jar`
  - `xlxpWASParsers.jar`
  - `wsdl4j-1.6.2.jar`
  - `XmlSchema-1.4.3.jar`
- JavaMail 1.4.3 (`activation.jar`, `mail.jar`)
- JUnit 4.4 (`junit-4.4.jar`)
- IBM Marketing Operations APIs (`affinium_plan.jar`)
- IBM Marketing Platform APIs (`unica-common.jar`)

If a procedure, or the secondary classes the procedure imports, does use such packages, their use must agree exactly with the packages provided by Marketing Operations or the application server. In this case, rework of your procedure code is required if a later version of Marketing Operations upgrades or abandons a library.

Procedures and threads

The procedure must be thread-safe concerning its own state; that is, its run method cannot depend on internal state changes from call to call. A procedure cannot create threads on its own.

Configuration parameters

When you install the Marketing Operations Integration Module, the installer sets three configuration properties. You can modify the configuration properties to customize the behavior of the Integration Module.

Configuration properties for the Integration Module are under **Marketing Operations | umoConfiguration | integrationServices**.

- The **enableIntegrationServices** configuration property to turns the Integration Services module on and off.
- The **integrationProcedureDefinitionPath** parameter contains the full file path to the custom procedure definition XML file.

  The default value is `<IBM_EMM_Home><MarketingOperations_Home>/devkits/ integration/ examples/src/procedure/procedure-plugins.xml/`.
- The **integrationProcedureClasspathURL** parameter contains the URL to the class path for custom procedures.

  The default value is `file:///<IBM_EMM_Home><MarketingOperations_Home>/ devkits/ integration/examples/classes/`.

  Note: The '/' at the end of the `integrationProcedureClasspathURL` path is required for loading procedure classes correctly.

Design

The procedure implementation class uses the IBM Marketing Operations API to read and update Marketing Operations components, start services, and so on. Other Java packages can be used to do other tasks.

In your design, focus on producing a single unit of work that operates atomically. Ideally, a procedure performs some series of tasks that can be scheduled asynchronously to run at some later time. This "fire and forget" integration model results in the least load on both systems.

Note: Only the documented classes and methods will be supported in future releases of Marketing Operations. Consider all other classes and methods in Marketing Operations to be off-limits.

After you code and compile the procedure implementation classes, you make them available to Marketing Operations. The build scripts that are supplied with the Marketing Operations Integration Services place the compiled procedures in the default location. The final development step is to update the custom procedure plug-in definition file that is used by Marketing Operations to discover the custom procedures.

The procedure must implement the **`com.unica.publicapi.plan.plugin.procedure.IProcedure`** interface and have a parameter-less constructor (usual JavaBeans model). Coding and compilation of each procedure is done in a Java IDE of the customer's choice, such as Eclipse, Borland JBuilder, or Idea. Sample code is provided with IBM Marketing Operations as developer toolkits, in the following location:

`<MarketingOperations_Home>/devkits/integration/examples/src/procedure`

Procedure lifecycle

Each procedure runs through a complete lifecycle.

The runtime lifecycle of a procedure includes the following steps.

1. Discovery and initialization
2. Selection (optional)
3. Execution
4. Destruction

Discovery and initialization

IBM Marketing Operations must be made aware of all standard and custom procedures available for a particular installation instance. This process is called discovery.

Note: Standard procedures (procedures that are defined by the Marketing Operations engineering team) are known implicitly and so do not need any action to be discovered.

Custom procedures are defined in the procedure plug-in definition file. The Marketing Operations plug-in manager reads this file during initialization. For each procedure found, the plug-in manager completes the following steps.

1. Instantiate the procedure; transition its state to INSTANTIATED.
2. Create a procedure audit record.
3. If the procedure was instantiated, its **`initialize()`** method is called with any initialization parameters found in its plug-in description file. If this method throws an exception, the status is logged and the procedure is abandoned. Otherwise, the procedure state changes to the INITIALIZED state. It is now ready to run.
4. Create a procedure audit record.
5. If the procedure was initialized, its **`getKey()`** method is called to determine the key that is used by clients to reference the procedure. This key is associated with the instance and saved for later lookup.

Selection

From time to time, IBM Marketing Operations might present a list of available procedures to users, for example to enable administrators to set up a trigger. Marketing Operations only presents this list after the procedure is initialized, using the procedure's **`getDisplayName()`** and **`getDescription()`** methods.

Execution

At some point after the procedure is initialized, IBM Marketing Operations receives a request to run the procedure. This request might happen concurrently with other procedures (or the same procedure) running on other threads.

At run time, the procedure execution manager completes the following steps.

1. Start a database transaction.
2. Set the procedure state to EXECUTING.
3. Create a procedure audit record.

4. Call the procedure's **execute()** method with an execution context and any run parameters that are provided by the client. The method implementation uses the Marketing Operations API as needed, acquiring edit locks, and passing along the execution context. If the run method throws an exception, the execution manager marks the transaction for rollback.
5. Commit or rollback the transaction according to the execution results; set procedure state to EXECUTED.
6. Release any outstanding edit locks.
7. Create a procedure audit record.

Note: The **execute()** method is not intended to alter the procedure instance data.

Destruction

When IBM Marketing Operations shuts down, the procedure plug-in manager walks through all loaded procedures. For each procedure found, it completes the following steps.

1. Calls the procedure's destroy() method to allow the procedure to clean up before the instance is destroyed.
2. Changes the state of the procedure to FINALIZED (it cannot be run).
3. Creates a procedure audit record.
4. Destroys the instance of the procedure.

Key Java classes

The supplied integration developer's kit contains a set of Javadoc for the public IBM Marketing Operations API and supporting classes.

The most important Java classes are listed here.

- IProcedure (com.unica.publicapi.plan.plugin.procedure.IProcedure): interface that all procedures must implement. Procedures go through a well-defined lifecycle and access the Marketing Operations API to do work.
- ITriggerProcedure (com.unica.publicapi.plan.plugin.procedure.ITriggerProcedure): interface that all trigger procedures must implement (marker interface).
- IExecutionContext (com.unica.publicapi.plan.plugin.procedure.IExecutionContext): interface of opaque context object that is handed to the procedure by the execution manager. This object has public methods for logging and edit lock management. The procedure also passes this object to all PlanAPI calls.
- IPlanAPI (com.unica.publicapi.plan.api.IPlanAPI): interface to the Marketing Operations API. The execution context provides a **getPlanAPI()** method to retrieve the proper implementation.

Data locking

IBM Marketing Operations uses a pessimistic edit locking scheme; that is, only one user is granted update access to component instances at a time. For the GUI user, this locking is done at the visual tab level. In some cases, data is locked for a subset of an instance, for example, a project summary tab. In other cases, data is locked across many instances, for example, the workflow tab. After a user acquires a lock, all other users are restricted to read-only access to the related data.

Rto ensure that the changes made by a procedure to a component instance or group of instances are not inadvertently overwritten by another user, a procedure must acquire the appropriate locks before it updates component data. The execution context object that is passed to the procedure's **execute()** method is used to accomplish lock the data.

Before the procedure updates any data, it must call the context's **acquireLock()** method for each lock it needs. For example, if a procedure is going to update a project and the associated workflow, the procedure must acquire locks for both.

If another user already has a lock, the **acquireLock()** method throws a **LockInUseException** immediately. To minimize collisions, the procedure must release the lock as soon as it updates the object.

The execution manager automatically releases any outstanding locks when the execute method returns. In any case, locks are only held for the life of the database transaction. That is, locks expire if the database-specific transaction timeout is exceeded.

Note: Edit locks are not the same as database transactions.

Procedure transactions

The procedure execution manager automatically wraps execution of the procedure with a database transaction, committing or rolling it back based on the outcome of the procedure execution.

Wrapping the procedure execution and database transaction ensures that updates to the IBM Marketing Operations database are not visible to other users until committed. It also makes the updates atomic.

The procedure writer still must acquire the necessary edit locks to ensure that other users cannot write changes to the database before the procedure execution completes.

Procedure communication

The **execute()** method of a procedure returns an integer status code to the IBM Marketing Operations procedure audit table. The **execute()** method of a procedure can also return zero or more messages to the procedure audit table, which are logged and persisted.

The client might also communicate the status information in some other way.

Procedure logging

IBM Marketing Operations has a separate log file for procedures:
<MarketingOperations_Home>\logs\procedure.log

The procedure execution manager logs the lifecycle of each procedure and creates audit records.

- **logInfo()**: write an informational message to the procedure log.
- **logWarning()**: write a warning message to the procedure log.
- **logError()**: write an error message to the procedure log.
- **logException()**: dump the stack trace for the exception to the procedure log.

Procedure plug-in definition file

The procedure plug-in definition file defines implementation class, metadata, and other information about the custom procedures to be hosted in IBM Marketing Operations.

By default, the procedure plug-in definition is assumed to be in the following path:

`<MarketingOperations_Home>/devkits/integration/examples/src/procedures/procedure-plugins.xml`

This file is an XML document that contains the following information.

Procedures: a list of zero or more **Procedure** elements.

Procedure: an element that defines a procedure. Each procedure contains the following elements.

- **key** (optional): string that defines the lookup key for the procedure. This key must be unique among all standard (IBM-supplied) and custom procedures that are hosted by a particular Marketing Operations instance. If not defined, defaults to the fully qualified version of the **className** element. Names starting with the string "uap" are reserved for use by IBM Marketing Operations.
- **className** (required): fully qualified package name of the procedure class. This class must implement the IProcedure class (com.unica.public.plan.plugin.procedure.IProcedure).
- **initParameters** (optional): a list of zero or more initParameter elements.

  initParameter(optional): parameter to be passed to the procedure's initialize() method. This element includes the nested parameter name, type, and value elements.
  - name: string that defines the parameter name
  - type: optional class name of the Java wrapper class that defines the type of the parameter value. Must be one of the following types:
    - java.lang.String (the default)
    - java.lang.Integer
    - java.lang.Double
    - java.lang.Calendar
    - java.lang.Boolean
  - value: string form of the attribute value according to its type

Chapter 4. IBM Marketing Operations API

The IBM Marketing Operations API is a façade that provides a client view of a running Marketing Operations instance.

Only a subset of the Marketing Operations capabilities is shown to users. The API is used concurrently by Marketing Operations web users and Marketing Operations Integration Services WebService SOAP requests and triggers. The API supports the following types of operations.

- Component creation and deletion
- Discovery (by component type, attribute value, and more values)
- Component inspection (through its attributes, specialized links, and more values)
- Component modification

Contents of the IBM Marketing Operations API

The com.unica.publicapi.plan.api package delivers the IBM Marketing Operations API.

This package offers interfaces and exceptions, and contains the following types of classes:

- Enumerated data types.
- Handles to identify object and component instances.
- A Java map, AttributeMap.

Complete documentation of the API, including all methods and possible values, is available by clicking **Help > Product Documentation** in an instance of Marketing Operations, then downloading the `IBM <version>PublicAPI.zip` file.

API Interfaces

The IBM Marketing Operations application programming interface (API) includes **IPlanAPI** and **IExecutionContext**.

The Marketing Operations API includes the following interfaces.

IPlanAPI

Defines the public API for Marketing Operations. Provides methods for creating, discovering, and modifying objects, including folders, projects, programs, workflow tasks, and team members.

For systems that have the optional integration with IBM Campaign enabled, also provides methods for creating, discovering, and modifying offers.

IExecutionContext

Defines the triggers and locks that execute methods in the API.

API methods

For specific information about the public API methods, refer to the iPlanAPI class in the JavaDocs API documentation files.

© Copyright IBM Corp. 2002, 2015

These files are available by logging in to Marketing Operations and selecting **Help > Product Documentation** from any page, and then downloading the `<version>PublicAPI.zip` file.

Common exceptions

Common exceptions that are thrown by the API include NotFoundException, AuthorizationException, DataException, InvalidExecutionContextException, and NotLockedException.

The following list explains why these common exceptions might occur.

- *<object type>*NotFoundException: The system is unable to return the specified item or object.
- AuthorizationException: The user who is associated with the execution context is not authorized for the requested operation. This exception can be thrown by any API method, so is undeclared.
- DataException: An exception occurred in the underlying database layer in IBM Marketing Operations. Check the SQL log for details.
- InvalidExecutionContextException: There is a problem with an execution context passed to an API method (for example, the method was not initialized correctly). This exception can be thrown by any API, so is undeclared.
- NotLockedException: Attempt to update component data without first acquiring the required lock. See the acquireLock() method of the IExecutionContext interface.

Handles

A Handle is special URL object that references a particular object instance in an IBM Marketing Operations instance. Handles include the component type, internal data identifier, and an instance base URL.

Handles used or generated by the API can be externalized to a full URL. You can use the resulting URL in different ways. You can use the URL to open a view of the component in the Marketing Operations GUI, send it in email messages, or use it in another procedure as a parameter.

Handles are valid only for a particular Marketing Operations service instance or clustered instance, but are valid for the lifetime of the deployed service. As a result, handles can be saved in a file for later reference, but they cannot be used to access components on another Marketing Operations instance. This restriction also applies to instances on the same physical host server. Marketing Operations does provide, however, a mechanism for mapping different base URLs to the current instance to accommodate relocating an instance to another server (for example, if the equipment malfunctions).

Handles are client-independent. For example, a trigger can pass a handle to a procedure, which uses it as a parameter in a SOAP call to a 3rd-party system. The 3rd-party system can then issue a SOAP request back to Marketing Operations to start a procedure that updates an attribute.

Members of the Handle class have factory methods for creating handles from various types of URLs. Examples follow.

Approval

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

Asset

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

Asset Folder

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

Asset Library

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

Attachment

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

Financial Account

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

Folder

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=grouping_folder&
folderid=1234
```

Invoice

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

Invoice line item

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

Marketing object

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

Marketing object grid

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

Marketing object grid row

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

Plan team

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

Plan user

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

Program

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=programtabs&programid=125
```

Program grid

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=programtabs&
programid=1234&gridid=grid
```

Program grid row

```
http://mymachine:7001/plan/affiniumplan.jsp?cat=programtabs&
programid=1234&gridid=grid&gridrowid=101
```

Project

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

Project grid

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

Project grid row

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

Project line item

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

Workflow stage

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

Workflow task

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

Attribute Map

The AttributeMap class is a Java map that contains only attributes. The attribute <*Name*> is the map entry key, and the attribute <*values*> array (note plural) is the map entry value.

The AttributeMap class includes the following fields.

- <*Name*>: the programmatic name of the attribute. This name serves as a unique key for accessing the attribute within the component instance in which it occurs.

  Note: <*Name*> is not necessarily the display name that is presented to a user in the GUI. For components that are created from templates (such as projects or workflow tasks), the attribute name is specified by the template element definition. The attribute name must be unique. For other components, the attribute name typically is derived programmatically from the server-side component instance (for example, through Java introspection).

  Note: By convention, custom attributes include the name of the form in which the editable version is defined: <*form_name*>.<*attribute_name*>.
- *Values*: a Java object array, containing zero or more attribute values. The type of each value must be the same and agree with the type of the attribute as it is defined in Marketing Operations. Only the following Java wrapper and Marketing Operations types are supported:
  - AssetLibraryStateEnum: a AssetLibraryStateEnum enumerated type value.
  - AssetStateEnum: a AssetStateEnum enumerated type value.
  - AttachmentTypeEnum: a AttachmentTypeEnum enumerated type value.
  - AttributeMap: a map that holds attributes.
  - BudgetPeriodEnum: a BudgetPeriodEnum enumerated type value.
  - BudgetTypeEnum: a BudgetTypeEnum enumerated type value.
  - Handle: a reference to a component instance, grid row, attribute, and so on.
  - InvoiceStateEnum: an InvoiceStateEnum enumerated type value.
  - java.io.File: representation of a file.
  - java.lang.Boolean: a Boolean value, either True or False
  - java.lang.Double: a double-precision decimal number value.
  - java.lang.Float: a single-precision decimal number value
  - java.lang.Integer: a 32-bit integer value
  - java.lang.Long: a 64-bit integer value
  - java.lang.Object: Generic Java object

- java.lang.String: a string of zero or more Unicode characters
- java.math.BigDecimal: arbitrary-precision signed decimal number value. Suitable for currency; the interpretation of the value depends on the currency locale for the client.
- java.math.BigInteger: arbitrary-precision integer value.
- java.net.URL: a Universal Resource Locator (URL) object.
- java.util.ArrayList: List of objects.
- java.util.Calendar: a date-time value for a particular locale.
- java.util.Date: a date-time value. This type is deprecated. Use java.util.Calendar or java.util.GregorianCalendar instead.

  Note: To implement date, users can select either java.util.Calendar or java.util.GregorianCalendar.
- java.util.GregorianCalendar: GregorianCalendar is a concrete subclass of java.util.Calendar and provides the standard calendar system in use by most of the world.
- MonthEnum: a MonthEnum enumerated type value.
- ProjectStateEnum: a ProjectStateEnum enumerated type value.
- QuarterEnum: a QuarterEnum enumerated type value.
- TaskStateEnum: a TaskStateEnum enumerated type value.
- WeekEnum: a WeekEnum enumerated type value.

The metadata of an attribute (such as translated display name and description) is defined by the template that is associated with the attribute and its parent object instance. Attributes provide a simple yet extensible mechanism for showing both required and optional object instance attributes, such as project name, code, and start date.

Enumerated data types

The IBM Marketing Operations API supports the following enumerated data types and values.

ApprovalMethodEnum

ApprovalMethodEnum defines valid approval methods. Possible values are:

- SEQUENTIAL
- SIMULTANEOUS

ApprovalStateEnum

ApprovalStateEnum defines valid approval states. Possible values are:

- CANCELLED
- COMPLETED
- IN_PROGRESS
- NOT_STATED
- ON_HOLD

AssetLibraryStateEnum

AssetLibraryStateEnum defines valid asset library states. Possible values are:

- DISABLED
- ENABLED

AssetStateEnum

AssetStateEnum defines valid asset states. Possible values are:

- ARCHIVE
- DRAFT
- FINALIZE
- LOCK

AttachmentTypeEnum

AttachmentTypeEnum defines valid attachment types. Possible values are:

- ASSET
- FILE
- URL

BudgetPeriodEnum

BudgetPeriodEnum defines the possible budget periods. Possible values are:

- ALL
- MONTHLY
- QUARTERLY
- WEEKLY
- YEARLY

BudgetTypeEnum

BudgetTypeEnum defines valid budget types. Possible values are:

- ACTUAL
- ALLOCATED
- COMMITTED
- FORECAST
- TOTAL

ComponentTypeEnum

ComponentTypeEnum identifies the accessible Marketing Operations component types. Possible values are:

- APPROVAL
- ASSET
- ASSET_FOLDER
- ASSET_LIBRARY
- ATTACHMENT
- FINANCIAL_ACCOUNT
- GROUPING_FOLDER
- INVOICE
- MARKETING_OBJECT
- PLAN_TEAM
- PLAN_USER
- PROGRAM
- PROJECT
- PROJECT_REQUEST
- TASK
-

InvoiceStateEnum

InvoiceStateEnum defines valid invoice states. Possible values are:

- CANCELLED
- DRAFT
- PAID
- PAYABLE

MonthEnum

MonthEnum defines valid values for the month.

OfferStateEnum

OfferStateEnum defines valid offer states. Possible values are:

- STATE_OFFER_DRAFT
- STATE_OFFER_PUBLISHED
- STATE_OFFER_RETIRED

ProjectCopyTypeEnum

ProjectCopyTypeEnum defines valid methods for copying a project. Possible values are:

- COPY_USING_PROJECT_METRICS
- COPY_USING_TEMMPLATE_METRICS

ProjectParticipantLevelEnum

ProjectParticipantLevelEnum identifies the roles that users can have in a project. Possible values are:

- OWNER
- PARTICIPANT
- REQUESTER

ProjectStateEnum

ProjectStateEnum defines valid project and request states. Possible values are:

- ACCEPTED
- CANCELLED
- COMPLETED
- DRAFT
- IN_PROGRESS
- IN_RECONCILIATION
- LATE: the project did not start by its scheduled begin date.
- NOT_STARTED
- ON_HOLD
- OVERDUE: the project was not completed before its scheduled end date.
- RETURNED
- SUBMITTED

For more information about project and task statuses, see the *IBM Marketing Operations User's Guide*.

QuarterEnum

QuarterEnum defines the valid values for quarters: Q1, Q2, Q3, and Q4.

TaskStateEnum

TaskStateEnum defines valid workflow task states. Possible values are:

- ACTIVE

- DISABLED
- FINISHED
- PENDING
- SKIPPED

WeekEnum

WeekEnum defines valid values for weeks in a year, from WEEK_1 to WEEK_53.

Before you contact IBM technical support

If you encounter a problem that you cannot resolve by consulting the documentation, your company's designated support contact can log a call with IBM technical support. Use these guidelines to ensure that your problem is resolved efficiently and successfully.

If you are not a designated support contact at your company, contact your IBM administrator for information.

Note: Technical Support does not write or create API scripts. For assistance in implementing our API offerings, contact IBM Professional Services.

Information to gather

Before you contact IBM technical support, gather the following information:

- A brief description of the nature of your issue.
- Detailed error messages that you see when the issue occurs.
- Detailed steps to reproduce the issue.
- Related log files, session files, configuration files, and data files.
- Information about your product and system environment, which you can obtain as described in "System information."

System information

When you call IBM technical support, you might be asked to provide information about your environment.

If your problem does not prevent you from logging in, much of this information is available on the About page, which provides information about your installed IBM applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, check for a `version.txt` file that is located under the installation directory for your application.

Contact information for IBM technical support

For ways to contact IBM technical support, see the IBM Product Technical Support website: (http://www.ibm.com/support/entry/portal/open_service_request).

Note: To enter a support request, you must log in with an IBM account. This account must be linked to your IBM customer number. To learn more about associating your account with your IBM customer number, see **Support Resources > Entitled Software Support** on the Support Portal.

© Copyright IBM Corp. 2002, 2015

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 2002, 2015

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Privacy Policy and Terms of Use Considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer's website terms of use (e.g. privacy policy) which includes a link to IBM's and Client's data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor's computer by IBM on the Client's behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client's behalf on website visitor's devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Online Privacy Statement at: http://www.ibm.com/privacy/details/us/en section entitled "Cookies, Web Beacons and Other Technologies."

Printed in USA