IBM Unica Campaign
Version 8.5.0

Campaign Offer API Specification

Publication Date: June 7, 2011

Unica, an IBM Company
Reservoir Place North
170 Tracer Lane
Waltham, MA 02451-1379

Document Information

IBM Unica Campaign 8.5.0 *Campaign Offer API Specification*

Table of Contents

CONTACTING IBM UNICA TECHNICAL SUPPORT

This section provides information about contacting Unica Technical Support.

The key topics in this section are:

Contacting IBM Unica 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 Unica technical support. Use the information in this section to ensure that your problem is resolved efficiently and successfully.

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

Information you should gather

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

- A brief description of the nature of your issue.
- Detailed error messages 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" below.

System information

When you call IBM Unica 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 Unica applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, you can obtain the version number of any Unica application by viewing the **`version.txt`** file located under each application's installation directory.

Contact information for Unica technical support

For ways to contact IBM Unica technical support, see the IBM Unica Product Technical Support website: (**http://www.unica.com/about/product-technical-support.htm**).

1 IBM Unica Campaign Services Public API Specification 3.0

This document defines the offer portion of version 3.0 of IBM Unica Campaign Services Application Programming Interface, hereafter called CampaignServices. Only the offer services that are described within this guide are supported.

If you are upgrading to IBM Unica Campaign version 8.2 and currently have the Unica Campaign Services API implemented, changes to the API necessitated by the upgrade from AXIS version 1.3. to AXIS2 1.4.1 will require changes to your application code. For details, see "Performance considerationsChanges to the Web Services API 3.0 for IBM Unica Campaign version 8.5.0" on page 7.

This section includes the following key topics:

If you are upgrading to IBM Unica Campaign version 8.2 and currently have the Unica Campaign Services API implemented, changes to the API necessitated by the upgrade from AXIS version 1.3. to AXIS2 1.4.1 will require changes to your application code.

High level summary of changes

- Migration of SOAP engine from AXIS version 1.3. to AXIS2 1.4.1.
- The WSDL has been restructured to deal with issues handling the required/optional parameters.
- The Client API `.jar` file has changed as a result of WSDL changes, and hence the generated stubs and classes have changed. The Client API method parameters have not changed, but the constructors of supporting value objects have been modified because of AXIS2 WSDL2Java converter usage.
- The Web Service URL has been changed to point to
  `http://<host>:<port>/Campaign/services/CampaignServices30Service`

  and the corresponding WSDL can be retrieved at
  `http://<host>:<port>/Campaign/services/CampaignServices30Service?wsdl`

As a result of these changes, if you are currently using the API you must make changes to your application code. Depending on whether you use the client API or the WSDL, refer to the details in the following sections:

- Existing implementations using Client API
- Existing implementations using WSDL directly

Existing implementations using Client API

If you use the Client API .jar file to interact with the Campaign web application, the following sections detail the changes you must be aware of.

New Client API .jar file

Changes to the internal implementation of Client API classes mean that your java application must use the new `.jar` file located at

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/lib/CampaignServicesClient30.jar`

For a java example showing new offer creation, see "Code OfferAPI.java" on page 59. The same example can be found in your Campaign installation at:

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/samples/OfferAPI.java`.

New dependent .jar files

As a result of the upgrade to AXIS2 version 1.4.1, your java application must also upgrade to using the AXIS2 1.4.1 distribution .jar files, as `CampaignServicesClient30.jar` is dependent on these .jar files. All of the dependent .jar files must be included in the java class path of your

application and can be found in the `Campaign.war` file located at `<CAMPAIGN_HOME>/Campaign.war`.

Extract the .jar files from `Campaign.war`, and include them in the java class path.

Changes in Client API constructor

While constructing the Client API object, change the web service URL and the Exception signature as shown below.

```
try {
URL serviceURL = new URL(PROTOCOL, HOST, PORT,
"/Campaign/services/CampaignServices30Service");
CampaignServices30SoapClient client = new
CampaignServices30SoapClient(serviceURL, TIMEOUT);
} catch (RemoteException exception) {
exception.printStackTrace();
}
```

Changes in parameterized constructors of the supporting classes

With the AXIS2 engine, the generated classes and stubs no longer have parameterized constructors. Instead, these classes have only the default no-argument constructor with setters and getters for the members.

So,

```
WSReference wsRef = new WSReference(WSComponentTypeEnum typeEnum, Long id);
```

is changed to

```
WSReference wsRef = new WSReference();
wsRef.setComponentTypeEnum(typeEnum);
wsRef.setId(id);
```

Existing implementations using WSDL directly

The WSDL of the Campaign web service is used to generate client-side stubs and supporting classes using any third-party converter tool. If you use the WSDL to interact with the Campaign

web application, the following sections detail the changes you must be aware of. Examples below use the WSDL2Java tool from Apache AXIS2 1.4.1.

WSDL location and service URL

The Campaign web service for IBM Unica Campaign 8.5.0 is deployed at
`http://host:port/Campaign/services/CampaignServices30Service`

The corresponding WSDL can be retrieved at
`http://host:port/Campaign/services/CampaignServices30Service?wsdl`

Generating stubs and classes

The WSDL2Java tool from Apache AXIS2 1.4.1 can be used to generate the stubs and supporting java classes from the WSDL. A sample ANT task is shown below.

The tool can also be used via the command line with the similar set of arguments. The argument values can be modified to suit to your environment.

The default ADB binding is used for the following WSDL2Java converter example.

```
<java classname="org.apache.axis2.wsdl.WSDL2Java" fork="true">
   <classpath refid="axis2.class.path"/> <!-Class path having AXIS2
libraries -->
   <arg value="-uri"/>
   <arg file="CampaignServices30.wsdl"/> <!-Actual location of WSDL -
->
   <arg value="-s"/> <!-- Generate sync style code -->
   <arg value="-Euwc"/> <!-- Takes care of generating Wrapper java
types for nillable = true elements. -->
   <arg value="-uw"/> <!-- Unwrap params -->
   <arg value="-u"/> <!-- Unpack classes -->
   <arg value="-ns2p"/> <!-- Namespace to package mapping. Customer can
have their own package names. -->
   <arg
value="http://webservices.unica.com/campaign/CampaignServices/3.0=com
.unica.publicapi.campaign.campaignservices.soap.v30"/>
<arg value="-o"/> <!-- Output directory -->
   <arg file="${autogen.java.dir}"/>
</java>
```

Using generated stubs and supporting classes

The stub can be used as follows:

```
CampaignServices30ServiceStub serviceStub = new
CampaignServices30ServiceStub(serviceURL);


serviceStub._getServiceClient().getOptions().setTimeOutInMilliSeconds
(webServiceTimeout); //Timeout in milliseconds.
```

The offer can be created as follows:

```
try {
   WSAttribute[] wsAttributes = {
      WSAttributeUtils.getWSTextAttribute(
      IAttributeMetadata.AC_OFFER_DESCRIPTION_ATTRIBUTE_NAME,
      null,
      new String[]{"description " + System.currentTimeMillis()}),
};


// convert to WSAttributeArrays
WSAttributeArrays wsAttributeArrays =
WSAttributeUtils.makeWSAttributeArrays(wsAttributes);


String offerName = "1st Offer"; //Name of the offer to be created.
String templateName = "Offer Template"; //Existing offer template name.


// make campaign WebService call
WSCreateOfferResponse wsResponse = getWebService().createOffer(
   WSHelper.getCreateOffer(authorizationLoginName, partitionName,
      Locale.US.toString(), securityPolicyName, name, folderID,
      templateName, wsAttributeArrays));
} catch (Exception exception) {
   //Handle the Exception here.
}


// process status
WSRequestStatus status = wsResponse.getStatus();
```

```
// done
WSOfferInfo offerInfo = wsResponse.getOfferInfo();
```

In this example, the createOffer() now accepts only one parameter of type CreateOffer.

With the AXIS2 engine, the generated classes and stubs no longer have parameterized constructors. For details, see “Changes in parameterized constructors of the supporting classes” on page 10.

References

The following references were used to prepare this specification:

- “Basic Profile Version 1.1", Web Service Interoperability Organization (WS-I), April 10, 2006.
(http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-0310.html)
- "SOAP 1.2 (draft)", W3C Soap working group, June 24, 2003
(http://www.w3.org/TR/soap/)
- "JAX-RPC 1.1", Sun Microsystems, October 14, 2003
(http://java.sun.com/webservices/jaxrpc/index.jsp)
- Apache Web services working group
(http://ws.apache.org/axis2)

Requirements

This section summarizes the requirements addressed by the design. The CampaignServices API must:

- Provide fine-grained create, discovery, read, and update access to IBM Unica Campaign components, while insulating clients from underlying implementation details
- Co-reside with, and minimize the effect on, existing IBM Unica Campaign GUI-based users
- Guarantee data integrity
- Support security architecture of IBM Unica Campaign
- Support industry-standard SOAP, including secure authentication

Design overview

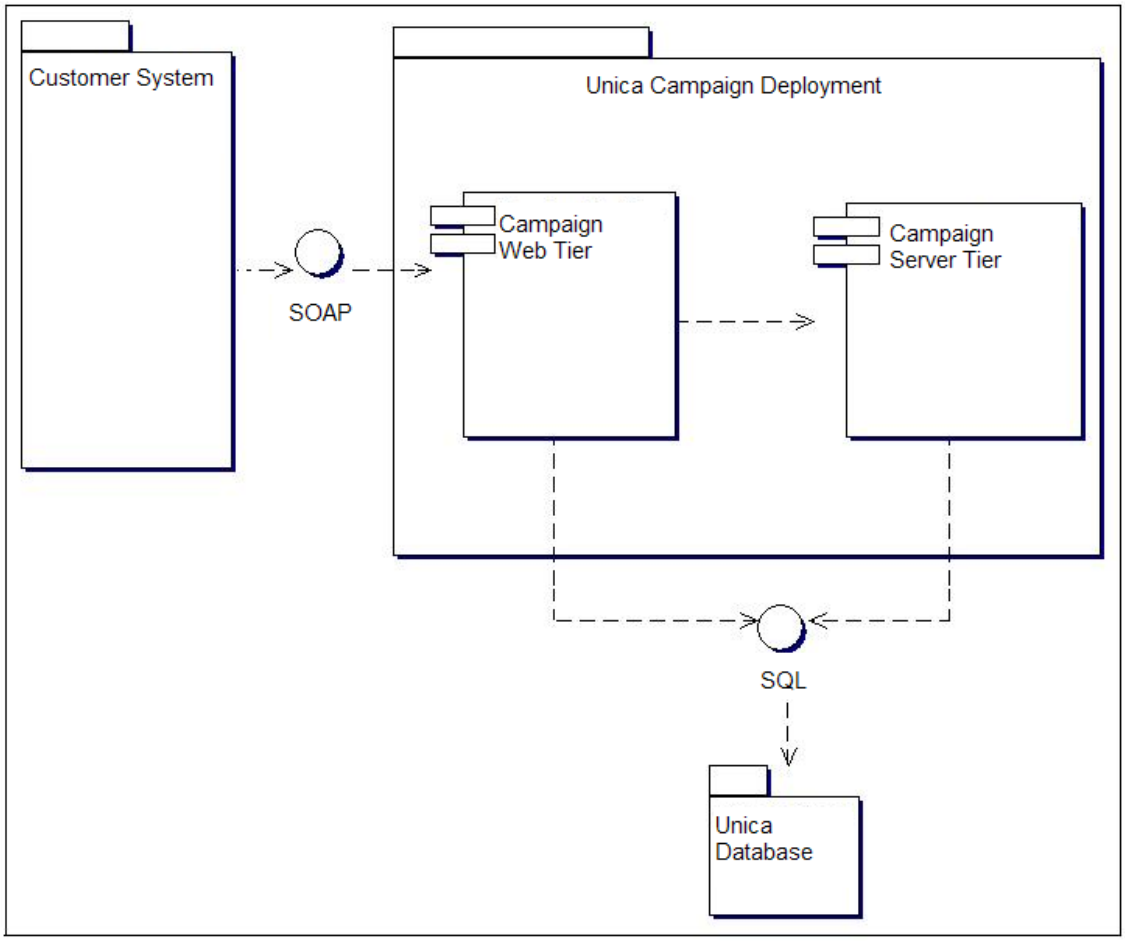
The CampaignServices API is a façade that provides a client view of a running IBM Unica Campaign application instance. Only a subset of the capabilities of IBM Unica Campaign are

exposed, but it is enough to drive key aspects of Campaign functionality. The API is designed to be used concurrently with IBM Unica Campaign web users and other API threads.

Generally, the API supports the following types of operations on campaigns, offers, and target cell components:

- Component creation
- Component discovery
- Component deletion
- Component Attribute and attribute metadata creation, inspection and modification
- Fetching of flowchart run results

The following diagram shows a sample deployment of CampaignServices 3.0.



Notes

This section addresses particular points about the design.

Versioning and backwards-compatibility

Generally, future versions of the CampaignServices API will be backward compatible with all minor and maintenance releases that share the same major version number. However, Unica reserves the right to break backward-compatibility for “dot zero” (x.0) major releases if the business or technical case warrants doing so.

The major version number of this API will be incremented if any of the following changes are made:

- Data interpretation changed
- Business logic changed, that is, the service method functionality
- Method parameters and / or return types changed

The minor version number of the API will be incremented if any of the following changes are made:

- new method added
- new data type added and its usage restricted to new method(s)
- new type added to an enumerated type
- a new version of an interface is defined

In particular, IBM Unica will continue to support the published WSDL, SOAP client, and the version of Apache Axis used to implement the SOAP offering until at least the next major IBM Unica release. Practically speaking, this will be done by supporting several version-specific web services concurrently. (IBM Unica already supports several versions of this service internally.)

User authentication

Authentication deals with establishing a user’s identity.

For this release, user authentication is the responsibility of the client application.

User authorization

Authorization deals with the permissions an authenticated user has relative to components and operations exposed by the API.

It is possible for a user to authenticate successfully, but not have sufficient permissions to perform some operations, such as edit a campaign's summary information. In this case, the API method will throw *AuthorizationException*.

Locale

API requests provide for an optional *requestedLocale* parameter, which defines the locale to use for executing that particular request. If not defined, the API will default to the IBM Unica user's preferred locale. The usual Java best-effort matching algorithm is used to return messages and other localized text in the requested locale.

This parameter is of type `java.util.Locale` class.

Some user-specified text, such as campaign descriptions, will be in the locale of the user that specified the text. IBM Unica Campaign does not attempt to localize such data. Only the information, warning, and error messages will localized by the API.

State management

The CampaignServices API is stateless, meaning that no per-client information is saved by the API across calls.

Obviously, specific API calls may change the state of underlying component instances managed by Campaign, and these state changes may be persisted to the data base.

Data Types

This section defines the public data types used by the CampaignServices API.

WSReference

A simple wrapper around a database identifier:

- ***componentTypeEnum***: an enumerated type indicating the component type the id is for. One of the following:
  - FOLDER
  - CAMPAIGN
  - FLOWCHART
  - TCS_CELL

  - OFFER
  - OFFER_LIST
  - OFFER_TEMPLATE
- ***id***: a *Long*, defining a unique numeric database-specific identifier for the reference.

WSVersion

A wrapper type that captures the various components of a version, including the following:

- *major*: an Integer defining the major version number, such as ‘8’ of the full version 8.1.2.3.
- ***minor***: an Integer defining the minor version number, such as ‘1’ of the full version above.
- ***maintenance***: optional Integer defining the maintenance number of the version, if applicable, such as ‘2’ of the full version above. Never supplied with an API version.
- ***patch***: optional Integer defining the patch release number, if applicable, such as ‘3’ of the full version above. Never supplied with an API version.

WSServiceInfo

A simple wrapper type around information about the service. It contains the following fields:

- ***apiVersion***: a *WSVersion* instance, defining the most current version of the API supported by the service. (Note that *apiVersion* will only include major and minor version information.)
- ***campaignVersion***: a *WSVersion* instance, defining the full version of the underlying IBM Unica Campaign instance.
- ***name***: internal name of the service, such as “CampaignServices30Service”.

WSAttributeTypeEnum

An enumerated type that defines all possible attribute types, one of:

- STANDARD: standard or base attribute defined by Campaign.
- CUSTOM: an attribute defined by another IBM Unica application, the customer, or some other 3rd-party.
- INPUT_PARAMETER: an input parameter, such as an attribute used to run an IBM Unica Campaign flowchart.
- OUTPUT_PARAMETER: an output parameter, such as an attribute whose value is filled in as the result of a flowchart run in IBM Unica Campaign.

WSAttributeStatusEnum

An enumeration of all possible attribute status codes, one of:

- ACTIVE: the attribute is active and may be used at will.
- RETIRED: the attribute has been removed from service and should not be used.

WSAccessTypeEnum

An enumerated type that defines all possible attribute value access types, one of:

- READ_ONLY: the attribute value can be read and displayed, but not modified
- READ_WRITE: the attribute value can be read, displayed, and modified

Attribute access is additive to security permissions, so for example, if the security policy for the client user denies read access to a particular attribute, then the attribute access cannot override that security setting. In fact, the API would never return the attribute to the client.

WSSelectTypeEnum

Defines all possible select types for a particular attribute value, one of:

- NONE: no selection (*hasOptions* is false).
- SINGLE_SELECT: only one attribute option from the list of possible options may be chosen at one time (only valid if an attribute *hasOptions*).
- MULTIPLE_SELECT: similar to SINGLE_SELECT, except that one or more options can be selected at one time.

WSRunStatusEnum

An enumerated type of all possible flowchart, branch, or cell run statuses, one of:

- NOT_STARTED: the run is scheduled, but has not started yet
- RUNNING: run in progress
- CANCELLED: the run was cancelled, either by a Campaign user or via this API
- SUCCEEDED: the run completed successfully
- FAILED: the run failed; error details are reported separately. (See "WSRunResults" on page 23.)

WSRunTypeEnum

An enumerated type of all possible run types, one of:

- NOT_RUN
- TEST_RUN
- PRODUCTION_RUN
- RUN_SKIPPED
- TEST_FLOWCHART
- PRODUCTION_FLOWCHART
- TEST_BRANCH
- PRODUCTION_BRANCH

- TEST_PROCESS
- PRODUCTION_PROCESS

WSAttribute

Attributes provide a simple, extensible mechanism for attaching arbitrary data to component instances accessible through the API, either standard data like a campaign *name*, flowchart run input parameters like *gender*, or arbitrary custom data specified by another IBM Unica application or IBM Unica customer.

In this API, attributes are used to model most component data, not only Campaign custom attributes.

Components generally will have many attributes associated with them, which are exposed by the CampaignServices API as a specially typed Map called an *AttributeMap*. Attribute data are represented as a strongly typed concrete class throughout the API, such as *WSDecimalAttribute*, for attributes that contain decimal (double precision numeric) data.

Each attribute includes the following:

- ***Name***: the unique name of the attribute. This name serves as the key for accessing the attribute and its metadata within the component instance that it occurs. The format of the name is undefined; in some cases it is assigned by the service, by the client, or by an IBM Unica Campaign user.

  Generally, this name is not the display name that would be presented to a Campaign or client user. It may be standardized by the API, such as *uacDescription*, it may be assigned by IBM Unica Campaign when publishing flowcharts, or it may be assigned by the IBM Unica application or customer when defining custom attributes. In all cases, however, the name is guaranteed unique.
- ***Metadata***: (optional) information about the attribute's data, such as value data type, display name, description, prompts, default value, select type, length (text), precision (decimals), options (if single or multiple select), etc. See "WSAttributeMetadata" on page 20.
- ***Values***: an array of zero or more strongly-typed value objects. The values field is supplied by the concrete attribute class; the type of each value must be the same and agree with the type definition in the attribute's metadata field. Not all attributes support multiple values, however.

  The following concrete attribute types are supported:

  - **WSBooleanAttribute**: an attribute whose value is a boolean, that is, *true* or *false*.
  - **WSIntegerAttribute**: an integer value (*java.lang.Long*).
  - **WSDecimalAttribute**: a double-precision decimal number value (*java.lang.Double*)

- **WSCurrencyAttribute**: a compound currency value, including an optional ISO 4217 currency code of the currency value, such as "*USD*" for the US dollar, and the currency value(s) captured as a *Double*. If the currency code is not supplied, the default used by IBM Unica Campaign is assumed.

  See `http://www.xe.com/symbols.php` for a list of countries, currency symbols, and codes. Note that the locale used for a currency value may be different from a user's preferred locale.
- **WSCalendarAttribute**: whose values are calendar dates, or datetimes, in some time-zone and locale.
- **WSTextAttribute**: a string of Unicode text (possibly null, or empty)

The list of possible attributes generally is different for each type of component, but the lists may overlap.

WSAttributeMetadata

WSAttributeMetadata defines information about the data of a particular typed attribute, such as value data type, localized text (display name, description, prompts, etc.), its default value, permissible value range, select type, options (if single or multiple select), etc. As with attributes, attribute metadata is strongly typed. So for example, a WSDecimalAttribute *myNumber* must have a WS*DecimalAttributeMetadata* bound to it, and all values, including the attribute values, metadata default value, and possible option values, will all be typed *Double.*

Descriptions, labels, and other attribute metadata text are generally localized; however, user-specified text may only be available as it was typed in by the user. Each API call includes a requested locale that client code can use to define the locale in which a particular user would like localized messages to be displayed. The usual Java locale fallback policies are used to fulfill the request.

WSAttributeMetadata includes the following fields:

- ***name***: The attribute's name, standard or custom; also the name used by the attribute that binds to this metadata. Standard attributes are defined by the system and have standard names in a reserved name space (that is, they use a "*uac*" prefix), custom names may use any other naming convention.

The attribute name must be unique, is never localized, and has length restrictions (which depend on the character content and database). The name is case-insensitive and can be composed of any combination of Unicode letter or digit characters, plus the underscore character '_', but cannot start with a digit.

- ***description***: optional description of the attribute. Suitable for a tooltip or other user-interface presentation.
- Predicates: assorted predicates that describe the attribute:
  - ***isRequired***: true if the attribute is mandatory.

- ***isInternal***: true if the attribute is defined by the system, and is for internal use only (should not be presented to a user).
- ***isGenerated***: true if the attribute's value(s) is auto-generated by IBM Unica Campaign when the component is created, such as a target cell code. Typically the *accessTypeEnum* will be READ_ONLY for generated values.
- ***hasOptions***: true if the attribute has options. Implies that options are defined for this metadata and that the *selectTypeEnum* is either SINGLE_SELECT or MULTIPLE_SELECT.

- ***typeEnum***: a *WSAttributeTypeEnum* that defines the type of attribute, such as STANDARD or CUSTOM.
- ***statusEnum:*** a *WSAttributeStatusEnum* that defines the status of the attribute, such as ACTIVE.
- ***accessTypeEnum***: a *WSAccessTypeEnum* that defines the type of access to the attribute value, such as READ_ONLY.
- ***selectTypeEnum***: a *WSAccessTypeEnum* that defines the type of selection used for the attribute, such as SINGLE. Must be NONE for campaign and cell attributes, or if no options are provided.
- ***componentTypeEnum***: a *WSComponentTypeEnum* of all possible Campaign components exposed by the API, such as CAMPAIGN, FOLDER, and so on.
- ***defaultValue*** (flowcharts only): optional type default value for the attribute. This value is provided by the concrete attribute metadata class, such as a *WSTextAttributeMetadata's* default value is of type String. (Refer to the description of the Attribute values). For components other than flowcharts, the default value is undefined.
- ***options***: optional list of options for this attribute. Taken together, an attribute's options define the exact set of permissible values for that attribute; each option is strongly typed, so for example a *WSTextAttributeMetadata* can only have a *WSTextAttributeOption* bound to it.

There is a restriction on options; only text attributes are supported.

Each option defines the following:

- ***prompt***: prompt for the option suitable for pull-down menus., such as "Male", as a gender attribute option. Note that unlike the metadata prompt, option display names usually do not include punctuation.
- ***description:*** localized description of the option, such as "A person of the male persuasion". Suitable for tool tip text.
- ***isDefault***: true if this particular option is the default. For MULTIPLE_SELECT select types, more than one option can be marked as a default.
- ***value***: the typed option value. As with the attribute metadata defaultValue, this value is provided by the concrete option subclass, such as a *WSDecimalAttributeOption's* value is of type Decimal. (Refer to the description of the Attribute values). Continuing the *gender* example, above, the value either could be declared as a string "m" (*WSTextAttributeOption*) or as a numeric code, 123 (*WSDecimalAttributeOption*).

WSCampaignInfo

A simple wrapper type around campaign attribute data.

It contains the following fields:

- ***reference***: the campaign's reference
- ***name***: campaign name (*uacName*); not guaranteed unique
- ***description***: optional campaign description (*uacDescription*)
- ***campaignCode***: the unique campaign code (*uacCampaignCode*); either assigned by the client or Campaign

WSComponentOrFolderInfo

Contains a combination of wrapped campaign or folder attribute data, like display name, its reference, etc.

It contains the following fields:

- ***reference***: the component or folder's reference
- ***name***: component or folder name (*uacName*); not guaranteed unique
- ***description***: optional component or folder description (*uacDescription*)
- ***componentCode***: unique code for the component, or null if a folder

WSTargetCellInfo

A simple wrapper around target cell row attribute data.

It contains the following fields:

- ***reference***: the cell reference
- ***name***: cell name (*uacName*); not guaranteed unique
- ***description***: optional cell description (*uacDescription*)
- ***cellCode***: the cell code (*uacCellCode*); either assigned by the client or Campaign. Note that cell codes can be forced unique by setting the IBM Unica Campaign *DuplicateCellCodesAllowed* configuration parameter to false
- ***flowchartName***: optional name of the flowchart to which the cell is bound

WSMetricsInfo

A simple wrapper type around campaign analytic data, included number of contacts. It contains the following fields:

- ***totalContacts***: a long giving the total number of contacts
- ***responses***: a typed list of *WSMetricsResponse* instances, each instance defining contact information for one response:

- ***typeCode***: a string defining the response type code, such as *PHC* for a phone call contact
- ***count***: a long giving the number of times this contact happened

WSRunResults

A wrapper type around the results of a flowchart, process box or cell run, possibly still in progress, including the run status, flowchart execution start and end date/times, and counts.

It includes the following fields:

- ***sourceReference***: optional Reference of the source of the run result. According to the context that run results are fetched, this can refer to a flowchart, a flowchart process box, or a target cell. In any case, the remaining run result data refer to this source.
- ***flowchartName***: the name of the flowchart that was run
- ***flowchartId***: the database identifier for the flowchart
- ***runId***: the database identifier of the run
- ***typeEnum***: an enumerated type that defines what kind of run generated the results, such as PRODUCTION_PROCESS (see *WSRunTypeEnum*)
- ***statusEnum***: an enumerated type defining the run status, such as RUNNING (see *WSRunStatusEnum*)
- ***statusCode***: optional integer status code
- ***statusMessage***: optional status message
- ***startDate***: optional calendar datetime of when the run started; will be null if the run has not started
- ***endDate***: as with *startDate*, but the datetime when the run ended (success or failure); will be null if the run has not started or is not finished yet
- ***count***: optional total count of contacts selected by the run; may be zero or null if the run has not completed

WSOfferInfo

A simple wrapper type around offer or offer list attribute data.

It contains the following fields:

- ***reference***: the offer or offer list's Reference
- ***name***: offer or offer list name (*uacName*); not guaranteed unique
- ***description***: optional description (*uacDescription*)
- ***offerCode***: the offer code (*uacOfferCode*) if an offer, or null if an offer list. (Not guaranteed to be unique.)

WSOfferCodeOrName

A simple wrapper type around offer codes or offer list names data.

It contains the following fields:

- ***isCode:*** boolean indicating if the *codeOrName* field is a presumed offer code (true) or the name of an offer list (false)
- ***codeOrName***: the unique offer code (*uacOfferCode*) if an offer, or the name of the offer list.

WSOfferValidationInfo

A simple wrapper type around offer validation information.

It contains the following fields:

- ***errorCode***: if not null, then defines the alpha-numeric validate error code. See the IStandardDefinitions class for error codes.
- ***errorMessage***: optional localized message that describes the error (if one occurred).
- ***codeOrName***: the validated offer code or offer list name
- ***reference***: the offer or offer list's Reference, if valid

WSOfferTemplateInfo

A simple wrapper type around offer template data.

It contains the following fields:

- ***reference***: the offer template's Reference
- ***name***: offer template name; guaranteed unique
- ***description***: optional description (*uacDescription*)
- ***offerTemplateID***: the unique offer template database identifier

WSBulkOfferInfo

Used to create offers in bulk.

It contains the following fields:

- ***offerName***: the name of the offer being created
- ***attributes***: an array of WSAttribute types indicating the offer attributes

WSOfferInfoStatus

A return type for the *bulkCreateOffers()* API method indicating the status of bulk offer creation.

It contains the following fields:

- ***name***: the name of the offer
- ***code***: the offer code. Will be null if offer creation fails
- ***description***: offer description
- ***reference***: the created offer's WSReference. Will be null if offer creation fails
- ***status***: an instance of WSRequestStatus indicating the status of offer creation

Common Exceptions

This section explains a number of common exceptions thrown by the CampaignServices API.

All exception localized messages will be in the requested locale if available to IBM Unica Campaign. The usual Java locale fallback policies apply.

RemoteException

This item applies to only the SOAP interface.

All SOAP calls to the API may throw a *RemoteException* if there is system-level error encountered, such as a problem in the SOAP envelop processing layer (Axis), a constraint defined in the web service WSDL was violated for some reason, etc.

Run-of-the-mill checked and unchecked API exceptions, such as *DataException*, will be returned as an error status, not as a RemoteException.

Refer to the SOAP interface section for details.

AuthenticationException

The user could not be authenticated for the specified Campaign partition. Check the user role set up in Platform.

AuthorizationException

The user is not authorized to perform the requested operation. This exception can be thrown by any API method, so it is undeclared (unchecked). Check the persmissions assigned to the user's role in Unica Marketing Platform.

DataException

A fatal exception occurred in the underlying database layer in Campaign (unchecked).

Check the Campaign flowchart and listener logs for details.

LockException

A temporary exception thrown when the client attempts to update a component, such as a flowchart, while it is being edited by another user. Generally, this exception can be recovered from by waiting some period of time and retrying the operation. Retry logic, however, is the responsibility of the client.

InvalidComponentException

An attempt was made to reference an invalid or unknown component (campaign, flowchart, target cell, etc.). The exception's `getComponentReference()` method returns the offending component's reference.

InvalidAttributeException

An exception thrown when the client provides or references an invalid attribute, such as if it uses the wrong data type, or uses an array of values where none are allowed, etc. The exception's `getAttributeName()` method returns the name of the problem attribute,

`getAttributeValue()` returns the value, and `getComponentReference()` identifies the component (or bulk index).

AttributeNotFoundException

Thrown whenever the client attempts to reference an unknown attribute (campaign, flowchart, target cell, etc.). The exception's `getAttributeName()` method returns the name of the unmatched attribute; `getComponentReference()` identifies the component (or bulk index).

AttributeExistsException

Thrown when the client tries to define a duplicate attribute metadata for a component. The exception's `getAttributeName()` method returns the name of the duplicate attribute; `getComponentReference()` identifies the component (or bulk index).

CompositeException

A *CompositeException* is used by some APIs to report multiple errors back to the caller. It typically will have more than one cause bound to it; all causes are captured as a list in the order that they occurred. The exception's `getCauseList()` method returns this list, which can be inspected further to get details of each error.

Generally the API will either complete successfully or rollback its work prior to throwing a CompositeException. See, for example, the bulk Target Cell Spreadsheet APIs described in "InvalidExecutionContextException, AuthorizationException" on page 40.

CampaignServices API Methods

This section defines the principal methods exported by the CampaignServices 3.0 API.

Service methods

The API provides a way to determine identifying information about the service itself.

getServiceInfo

```
WSServiceInfo getServiceInfo() throws CampaignServicesException;
```

Returns information about the service, such as the most current API version it supports, the full version of the underlying IBM Unica Campaign instance, etc.

No client information is needed by this call and no security permissions are applied.

Parameters

None

Returns

Returns a *WSServiceInfo* instance

Errors

None

Attributes

Most component instance data are exposed by the API as *attributes* or attribute *metadata*. In some cases the attribute metadata definitions are global to IBM Unica Campaign (such as campaign custom attributes), while in others they are restricted to a particular component (such as flowchart user variables).

Unless otherwise indicated, all attributes may be read if the client has sufficient security permissions to do so.

Only components that are active and that the client has access to, are exposed by this API. Public support is limited to offer attributes, offer template, and metadata.

getAttributesByName

```
Map<String, WSAttribute>
    getAttributesByName(String userCredential, String partitionName,
       Locale requestedLocale,
       WSReference reference,
       String[] names)
throws CampaignServicesException;
```

Fetches the named attributes associated with the specified component instance (may be empty).

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request; if not supplied, then the IBM Unica user's locale preferences will be used. Normal locale defaulting algorithm will be applied if necessary.

partitionName: optional name of the campaign partition to use. If not defined, then the default partition is used.

reference: the *Reference* for the component instance containing the desired attributes. *InvalidComponentException* is thrown if the reference is invalid or the component does not exist.

names: optional array of names of attributes to fetch (not display names); if not supplied, all attributes are returned. Throws *AttributeNotFoundException* if one of the named attributes does not exist.

Returns

A typed map of zero or more attributes; the attribute name is the map entry key and the attribute instance is the entry value.

Errors

InvalidComponentException, AttributeNotFoundException

AuthorizationException, DataException

All of these exceptions are wrapped inside the CampaignServicesException.

updateAttributes

```
void updateAttributes(String userCredential, String partitionName,
       Locale requestedLocale, WSReference reference,
       boolean allowCreate,
       WSAttribute[] attributes)
   throws CampaignServicesException;
```

Update one or more attributes of the component instance with the supplied attribute values.

Update logic

The update logic is as follows.

For each attribute contained in the supplied attribute map:

1. If the attribute name matches an existing attribute, attempt to overwrite its *values* field with the supplied values field.
2. If the attribute does not exist yet, *allowCreate* is true, and its metadata are known, then create the attribute. This applies to global attribute metadata as well instance attributes (except flowcharts)
3. If the value type or some other aspect of the attribute's metadata definition is not met, or one or more of the supplied values is invalid, out-of-range, etc., throw *InvalidAttributeException*.
4. Else throw *AttributeNotFoundException* if the named attribute does not exist.

In the event of an exception, none of the updates will be committed.

This particular method does not support defining new custom attributes; use the `createAttributeMetadata()` method for that.

In all cases, the attribute update operation is subject to the usual security constraints and validation. It is the client's responsibility to determine which attributes are required by a particular component instance, the correct types, etc.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

reference: the Reference for the component instance containing the attributes to be updated

allowCreate: indicates if a new attribute should be created if it does not exist yet for the component. (See "Update logic" on page 30.)

attributes: an array of attributes to be updated; the attribute name is used to locate the attribute to update and the new value(s) are used to update the existing attribute's value as a single object of the proper type or an array, if applicable. (Refer to "Common Exceptions" on page 25.)

Returns

Nothing.

Errors

InvalidComponentException, AttributeNotFoundException, InvalidAttributeException

AuthorizationException, DataException

getAttributeMetadataByName

```
Map<String, WSAttributeMetadata>
    getAttributeMetadataByName(String userCredential,
        StringpartitionName, Locale requestedLocale,
        WSReference reference, String[] names)
    throws CampaignServicesException;
```

Fetches the named attribute metadata definition(s) bound to a particular component, template, or globally defined.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

reference: optional Reference for the component or template containing the desired attribute metadata. If only the ComponentTypeEnum is provided, then the fetch is restricted to components of that type; if the reference is not supplied at all, the fetch will return all global metadata definitions, for all component types. Throws *InvalidComponentException* if the supplied reference is invalid.

names: optional array of names of attribute metadata to fetch; if not supplied, all metadata for the component, or globally defined if no Reference is provided, are returned. Throws *AttributeNotFoundException* if one or more of the specified attribute metadata definitions does not exist.

Returns

A typed map of zero or more attribute metadata definitions; the attribute name is the map entry key and the attribute's metadata is the entry value.

Errors

InvalidComponentException, AttributeNotFoundException

AuthorizationException, DataException

createAttributeMetadata

```
void createAttributeMetadata(String userCredential,
       String partitionName,
       Locale requestedLocale, WSReference reference,
       WSAttributeMetadata[] attributeMetadata)
   throws CampaignServicesException;
```

Create one or more new attribute metadata definitions and optionally bind them to a particular component or template.

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

reference: optional Reference for the component or template that the metadata should be bound to. If not supplied, the created metadata definition will be global. If the reference is supplied, but not valid, then throws *InvalidComponentException*.

attributeMetadata: an array of attribute metadata definitions to bind. If one or more of the specified metadata already is bound to the component, that is, the name is not unique, then throw *AttributeExistsException*. Throws *InvalidAttributeException* if there is a problem with one or more of the specified metadata, that is, it is internally inconsistent.

Returns

Nothing.

Errors

InvalidComponentException, AttributeExistsException, InvalidAttributeException

AuthorizationException, DataException

updateAttributeMetadata

```
void updateAttributeMetadata(String userCredential,
       String partitionName,
       Locale requestedLocale, WSReference reference,
       boolean allowCreate,
       WSAttributeMetadata[] attributeMetadata)
   throws CampaignServicesException;
```

Update one or more attribute metadata definitions of the specified component or template, optionally creating new metadata definitions if needed.

Update logic

The update logic is as follows.

For each attribute metadata definition contained in the supplied array:

1. If the attribute name does not match an existing metadata bound to the component, do the following based on the *allowCreate* parameter value:
   a. *True*: create a new metadata definition. Functionally identical to using the `createAttributeMetadata()` request.
   b. *False*: throw *AttributeNotFoundException.*

2. If the attribute metadata data type is different, throw *InvalidAttributeException*
3. Attempt to overwrite the existing attribute metadata definition with the field values of the supplied metadata, else throw *InvalidAttributeException.*

   Only the following updates are supported (else throw *InvalidAttributeException*):

   a. ***name***: cannot be changed (name is the key!)
   b. ***displayName***: accept new value
   c. ***description***: accept new value
   d. ***isRequired***: only allow change from *true* to *false*
   e. ***isInternal***: accept new value
   f. ***isGenerated***: no change allowed
   g. ***attributeTypeEnum***: no change allowed
   h. ***accessTypeEnum***: accept new value
   i. ***selectTypeEnum***: accept these transitions if options are provided:
      i. NONE to SINGLE_SELECT or MULTIPLE_SELECT
      ii. SINGLE_SELECT to MULTIPLE_SELECT
   j. ***options***: options may be added, but not deleted. Only the following option changes are supported (as per value match):
      i. ***displayName***: accept new value (no ripple)
      ii. ***description***: accept new value (no ripple)
      iii. ***isDefault***: accept new value; however must match SelectTypeEnum
      iv. ***value***: no change allowed (value is the key!)
   k. ***defaultValue*** (flowcharts only): accept the new default value
   l. ***maximumLength*** (text only): accept the new length if larger
4. If the attribute metadata definition is not internally consistent, then throw *InvalidAttributeException.*
5. If necessary, find all component instances that reference the updated attribute metadata, and update as appropriate

In the event of an exception, none of the updates will be committed.

In all cases, the attribute update operation is subject to the usual security constraints and validation.

See `createAttributeMetadata()`, `deleteAttributeMetadata()`

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

reference: optional Reference for the component instance containing the desired attributes. If not supplied, then the update will be restricted to global metadata definitions. Throws *InvalidComponentException* if the supplied reference is invalid.

allowCreate: if true, metadata definitions that currently do not exist will be created (functionally equivalent to using the `createAttributeMetadata()` method).

attributeMetadata: an array of attribute metadata definitions to be updated (and added if the *allowCreate* flag is true); the attribute name is used to locate the metadata definition to update and the remaining data are used to update the existing definition. (Refer to "Update logic" on page 33.)

Returns

Nothing.

Errors

InvalidComponentException, InvalidAttributeException

AuthorizationException, DataException

deleteAttributeMetadata

```
void deleteAttributeMetadata(String userCredential,
       String partitionName,
       Locale requestedLocale, WSReference reference,
       String[] names)
   throws CampaignServicesException;
```

Deletes one or more named attribute metadata definitions from the specified component, template (custom attribute metadata only) or global attribute metadata definitions.

As part of this task, the method will find all components that reference the deleted metadata, and update as appropriate.

However, in the event of an exception, none of the deletes will be committed.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

reference: optional Reference of the component or template containing the attributes to be deleted. If not supplied, then the delete will be restricted to global metadata definitions. Throws *InvalidComponentException* if the supplied reference is invalid.

If the optional array of names of the attribute metadata is not supplied, then this method will attempt to delete all custom attribute metadata associated with the component, or all global definitions if the reference was not provided.

---

names: optional array of names of the attribute metadata to delete. Throws *AttributeNotFoundException* if one or more of the named attribute metadata does not exist. Throws *InvalidAttributeException* if an attribute could not be removed.

Returns

Nothing.

Errors

InvalidComponentException, AttributeNotFoundException, InvalidAttributeException

AuthorizationException, DataException

Campaigns

The API supports the following operations on campaigns (subject to security permissions):

- create a new campaign
- discovery (listing of campaigns by various criteria)
- attribute create, read and update (via attribute APIs)

Campaigns have a number of standard attributes associated with them that are exposed by the API. This list can be extended at will by the client by adding custom attributes (see the Attributes APIs).

Standard campaign attributes are:

- ***uacName***: campaign name (not guaranteed unique).
- ***uacDescription***: optional string describing the campaign.
- ***uacCampaignCode***: a string code that uniquely identifies the campaign. Typically auto-generated by Campaign, but may be provided by the client.

- ***uacCreateDate***: a Calendar defining the data & time when the campaign was created by the server.
- ***uacUpdateDate***: a Calendar indicating the date and time when the campaign was last updated by the server.
- ***uacInitiative***: optional string defining the campaign initiative.
- ***uacObjectives***: optional string identifying the objectives of the campaign.
- ***uacStartDate***: an optional Calendar giving the date & time when the campaign was started by the server, or is scheduled to start.
- ***uacEndDate***: as with *uacStartDate*, but defines the date & time when the campaign was completed or is scheduled to be completed. Must be after the *uacStartDate*.
- ***uacLastRunDate***: an optional Calendar giving the date & time when any flowchart bound to the campaign was last run (else null).
- ***uacExternalLinkOwner:*** an optional string defining the name of the owner of an external link (see *uacExternalLinkReference* attribute). Unica use only; must be one of the following:
  - "Plan" (now known as Unica Marketing Operations)
  - "Collaborate" (now known as IBM Unica Distributed Marketing)
- ***uacExternalLinkId***: an optional numeric database identifier assigned by another IBM Unica application to an object linked to this campaign. Unica use only: see also the *uacExternalLinkOwner* attribute.

generateCampaignCode

```
String generateCampaignCode(String userCredential,
       String partitionName,
       Locale requestedLocale);
```

Generate a new campaign code.

This code is guaranteed unique and different from the value returned by a previous or future call to this method, or the `createCampaign()` method, or the value generated for a campaign created via the IBM Unica Campaign GUI.

The use of this method is optional, as the `createCampaign()` API will generate a campaign code for the client if one is not supplied.

See `createCampaign()`.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use. If there is only one partition in the Campaign installation, this argument may be null.

Returns

The generated campaign code.

Errors

AuthorizationException, DataException

createCampaign

```
CampaignInfo createCampaign(String userCredential,
      String partitionName,
      Locale requestedLocale,
      String securityPolicyName,
      String name, Attribute[] attributes)
      throws InvalidFolderException, AttributeNotFoundException,
      InvalidAttributeException;
```

Create a new campaign for the client, partition, and securityPolicyName, applying the specified attributes. All campaigns created by this API will be under the root folder.

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

securityPolicyName: optional name of the campaign security policy to use to create the campaign. All subsequent operations on this campaign will use this policy. If not defined, the *Global* policy will be used.

name: the name to assign the new campaign instance (its "*uacName*" attribute).

attributes: an optional array of initialization attributes; any supplied attributes will overwrite the campaign's default values; others are left untouched. For example, if a *uacCampaignCode*

attribute is supplied, it will be used instead of an auto-generated one. It is up to the client to determine the attributes required by the campaign, their types, etc.

Throws *AttributeNotFoundException* if one or more of the named attributes does not exist or *InvalidAttributeException* if an attribute value is invalid (such as incorrect data type).

Returns

A single instance of a *CampaignInfo* for the created campaign.

Errors

InvalidAttributeException, AttributeNotFoundException

AuthorizationException, DataException

listCampaignsByPage

```
List<CampaignInfo>
   listCampaignsByPage(String userCredential, String partitionName,
      Locale requestedLocale, Attribute[] attributes,
      long pageOffset, int pageSize)
   throws AttributeNotFoundException, InvalidAttributeException,
      RangeException;
```

Enumerate a "page" of campaigns that match the optional attribute values, beginning with the specified page offset. Folders are ignored.

Once retrieved, each *CampaignInfo* returned can be used as is, such as to display a summary list, or the attribute methods can be used to fetch or update the campaign's attributes.

No state is maintained by this API, so it is possible to make calls to it in any order.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

attributes: optional array of attributes to match; the attribute's name, data type, and value(s) are used to determine the match; if the attribute supports arrays, then all values specified must match. The implied match operator is an AND, so only campaigns that match all the supplied attribute values will be returned.

Throws *AttributeNotFoundException* if an attribute name does not exist or *InvalidAttributeException* if one or more of the supplied attributes is invalid.

pageOffset: the starting offset of all possible campaigns to begin the enumeration (zero-valued). For example, if the enumeration matches 1000 campaigns and this value is set to 10, then the

page would start at the 11th component. A *RangeException* is thrown if the supplied offset is out of range.

pageSize: the maximum number of matched campaigns to return for the page (cannot exceed 500).

Returns

A typed *List* of zero or more *CampaignInfo* data wrapper instances, one for each matched campaign in the page.

Errors

AttributeNotFoundException, InvalidAttributeException, RangeException

InvalidExecutionContextException, AuthorizationException

Target Cell methods

Target cells are an abstraction for some known subset(s) of campaign results that are managed by IBM Unica Campaign as a Target Cell Spreadsheet (TCS). Target cells may be global to a campaign or may be associated with a particular campaign flowchart.

The API supports the following operations on target cells:

- create one or more new global target cells
- bulk update one or more existing target cells
- discovery (listing of target cells)
- attribute create, read, and update (via attribute APIs)
- delete an existing target cell
- fetch run results associated with one or more cells

Target cells have a number of standard attributes associated with them that are exposed by the API. This list can be extended at will by the client by adding custom attribute metadata definitions (see the Attributes APIs). Each attribute metadata can be thought of as a column in the TCS—the layout of the spreadsheet is up to the client.

Standard target cell attributes are:

- ***uacName***: cell name.
- ***uacDescription***: optional string describing the flowchart.
- ***uacCellCode***: a code string that uniquely identifies the cell. Typically auto-generated by Campaign, but may be provided by the client.
- ***uacCreateDate***: a Calendar instance giving the date & time when the cell was created by the server.
- ***uacUpdateDate***: a Calendar instance defining when the last time the cell was updated by the server.

- ***uacIsControl***: a boolean indicating if this is a control cell (true) or not (false). Other cells may refer to this cell as a control cell (see *uacControlCell*)
- ***uacControlCell***: optional Reference of the control cell (not allowed if a control cell). See *uacIsControl* attribute.
- ***uacIsApproved***: a boolean indicating if the cell is approved (true) or not (false).
- ***uacIsReadOnly***: a boolean indicating if the cell is read-only (true) or not (false).
- ***uacDisplayOrder***: an integer giving the order of this cell (row) relative to others in the target cell spreadsheet.
- ***uacIsTopDown***: a boolean indicating if the cell is top-down.
- ***uacAssignedOffers***: an optional array of one or more References of offers or offer lists assigned to this cell (not allowed if a control cell).
- ***uacFlowchartName***: optional name of flowchart that this cell is linked to (read-only—must be set via the IBM Unica Campaign GUI; not allowed if a control cell).
- ***uacFlowchartId***: optional database identifier for the flowchart that this cell is linked to (read-only as above)

createTargetCell

```
TargetCellInfo
   createTargetCell(String userCredential, String partitionName,
      Locale requestedLocale,
      Reference campaignReference,
      Attribute[] attributes)
   throws InvalidComponentException, CompositeException;
```

Create a new campaign-specific target cell row, applying the specified per-cell attributes and user information.

The specified attributes can be standard or custom; however, if custom, then the corresponding global attribute metadata definitions must already exist.

Once the target cell is created, attribute values can be changed using the attributes APIs.

See `listTargetCells(), bulkCreateTargetCells()`.

See `createAttributeMetadata()`, `listAttributeMetata()`, `getAttributesByName()`

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

campaignReference: the Reference of the campaign containing the target cell spreadsheet to be updated. Accumulates an *InvalidComponentException* if the campaign does not exist or the Reference is invalid.

attributes: optional array of TCS attributes for the new cell. Each supplied attribute element will overwrite the corresponding cell attribute's default values; others are left untouched. It is up to the client to determine the attributes required by the cell, their types, etc. Accumulates an *InvalidAttributeException* if there is a problem with a specified attribute.

If any exceptions have been accumulated, this method will throw a *CompositeException* and all creates will be undone. The exception's list of causes will include an exception for each attribute that caused the error and will include a numeric index instead of the *reference*, the name of the attribute, and usually the offending value. The cause list will be ordered as with the input *attributeList*.

Returns

A *TargetCellInfo* data wrapper for the created TCS cell.

Errors

InvalidComponentException, CompositeException

AuthorizationException, DataException

bulkCreateTargetCells

```
List<TargetCellInfo>
      bulkCreateTargetCells(String userCredential,
      String partitionName,
      Locale requestedLocale,
      Reference campaignReference,
```

```
        List<Attribute[]> attributesList)
    throws InvalidComponentException, CompositeException;
```

Create many new campaign-specific target cell rows at one time, applying the specified per-cell attributes and user information.

The specified attributes can be standard or custom; however, if custom, then the corresponding global attribute metadata definitions must already exist.

Once the target cell is created, attribute values can be changed using the attributes APIs.

See `listTargetCells()`.

See `createAttributeMetadata()`, `listAttributeMetata()`, `getAttributesByName()`

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

campaignReference: the Reference of the campaign containing the target cell spreadsheet to be updated. Accumulates an *InvalidComponentException* if the campaign does not exist or the Reference is invalid.

attributeList: optional list of per-cell attribute arrays, one for each target cell row to be created. Any supplied attributes for a particular list element will overwrite the corresponding cell attribute's default values; others are left untouched. It is up to the client to determine the attributes required by the cell, their types, etc. Accumulates an *InvalidAttributeException* if there is a problem with a specified attribute.

If any exceptions have been accumulated, this method will throw a *CompositeException* and all creates will be undone. The exception's list of causes will include an exception for each attribute

that caused the error and will include a numeric index instead of the *reference*, and the name of the attribute, etc. The cause list will be ordered as with the input *attributeList*.

Returns

A list of *TargetCellInfo* data wrappers, one for each created instance, ordered according to the element order of the input *attributesList* parameter.

Errors

InvalidComponentException, CompositeException

AuthorizationException, DataException

listTargetCells

```
List<TargetCellInfo>
   listTargetCells(String userCredential,
   Reference campaignReference, Locale requestedLocale,
   Attribute[] attributes)
throws InvalidComponentException, InvalidAttributeException;
```

Lists information about all target cells that currently exist that match the specified attributes, either for the specified campaign, or globally if no campaign is specified.

See `getAttributeMetadata(), getAttributesByName()`.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

campaignReference: Reference of the parent campaign. Throws *InvalidComponentException* if the campaign does not exist or the Reference is invalid.

attributes: optional array of attributes to match. The implied match operator is an AND, so only cells that match all the supplied attribute values will be returned.

Throws *InvalidAttributeException* if one or more of the specified attributes is invalid.

Returns

Returns a list of zero or more *TargetCellInfo* instances for the matched cells.

Errors

InvalidComponentException, InvalidAttributeException

AuthorizationException, DataException

bulkUpdateTargetCells

```
void bulkUpdateTargetCells(String userCredential,
      String partitionName,
      Locale requestedLocale,
      Map<Reference, Attribute[]> attributesMap)
   throws CompositeException;
```

Update the attributes of one or more target cells.

The update logic is as follows.

For each element in the supplied *attributesMap*, the entry key is the Reference of the target cell to update and the entry value is an array of update attributes for that cell. If the target cell does not exist, accumulate an *InvalidComponentException*.

Once a target cell is located, for each attribute specified, do the following:

1. If the attribute name matches an existing attribute, attempt to overwrite its *values* field with the supplied values field.
2. If the value type or some other aspect of the attribute's metadata definition is not met, or one or more of the supplied values is invalid, out-of-range, etc., accumulate an *InvalidAttributeException*.

3. Else accumulate *AttributeNotFoundException* if the named attribute does not exist

If any exceptions have been accumulated, this method will throw a *CompositeException* and all updates will be undone. The exception's list of causes will include the exceptions listed above. For each attribute that caused the error, both the reference and the attribute name will be recorded.

In all cases, the attribute update operation is subject to the usual security constraints and validation. It is the client's responsibility to determine which attributes are required by a particular component instance, the correct types, etc.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

attributesMap: a map of target cells to update; the entry key is the Reference of the cell to update and the entry value is an array of update attributes. The attribute name is used to locate the attribute to update and the new attribute value(s) are used to update the existing attribute's value as a single object of the proper type or an array, if applicable. See exceptions above.

Returns

Nothing.

Errors

ComponentException

AuthorizationException, DataException

getRunResultsByCell

```
List<RunResults>
   getRunResultsByCell(String userCredential, String partitionName,
      Locale requestedLocale,
      Reference[] cellReferences)
   throws InvalidComponentException;
```

Get the run results of one or more target cells—possibly for a flowchart that never started, or is still in progress.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

cellReferences: an array of References of the target cells whose run results are desired. Throws *InvalidComponentException* if one or more of the cell References is invalid or references a non-existent cell.

Returns

Returns a typed list of run results for the named cells, ordered according to the input Reference array.

Each run status will be RUNNING if the underlying flowchart process box is still running, FAILED if the run failed for some reason, or NOT_STARTED if the process box run has not started. Status details are also provided.

Errors

InvalidComponentException

AuthorizationException, DataException

bulkDeleteTargetCells

```
void bulkDeleteTargetCells(String userCredential,
      String partitionName,
      Locale requestedLocale,
      Reference[] cellReferences)
   throws CompositeException;
```

Deletes one or more existing target cells and all their dependent components (that is flowchart linkage, attributes, etc.).

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

cellReferences: an array or one or more References of cells to be deleted. *InvalidComponentException* is accumulated if there is a problem with one of the specified References, or a cell does not exist.

If any exceptions have been accumulated, this method will throw a *CompositeException* and all deletes will be undone. The exception's list of causes will include the exceptions listed above. For each cell that caused the error, the reference will be recorded.

Returns

Nothing.

Errors

CompositeException

AuthorizationException, DataException

Analytics

The API supports the retrieval of simple metrics from IBM Unica Campaign.

getCampaignMetrics

```
MetricsInfo getCampaignMetrics(String userCredential,
      String partitionName,
      Locale requestedLocale,
      Reference campaignReference)
   throws InvalidComponentException;
```

Fetch the metrics for the specified campaign.

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

campaignReference: the Reference of the parent campaign. Throws *InvalidComponentException* if there is a problem with the campaign Reference or the campaign does not exist.

Returns

Returns a *MetricsInfo* instance for the campaign.

Errors

InvalidComponentException

AuthorizationException, DataException

Offer, offer list and offer template methods

The API supports the following operations on offers:

- discovery (listing by folder (offers, offer lists and subfolders), attribute (offers and offer templates), or search value (offers))
- validation
- information retrieval (retrieve attributes for a specific offer or offer template)
- create, edit, retire, and delete offers

Offers have a number of standard attributes associated with them. This list can be extended at will by the client by adding custom attribute metadata definitions (see the Attributes APIs).

Standard offer attributes are:

- ***uacName***: offer name.
- ***uacDescription***: optional string describing the offer.
- ***uacOfferCode***: a code string that uniquely identifies the offer. Typically auto-generated by IBM Unica Campaign, but may be provided by the client.
- ***uacCreateDate***: a Calendar instance giving the date & time when the offer was created by the server.
- ***uacUpdateDate***: a Calendar instance defining when the last time the offer was updated by the server.

Offer templates also have standard and custom attributes. Standard offer template attributes are:

- ***uacName***: offer template name.
- ***uacDescription***: optional string describing the offer template.
- ***uacCreateDate***: a Calendar instance giving the date & time when the offer template was created by the server.
- ***uacUpdateDate***: a Calendar instance defining when the last time the offer template was updated by the server.

listOffersAndFolders

```
List<WSComponentOrFolderInfo>
    listOffersAndFolders(String userCredential, String partitionName,
        Locale requestedLocale,
        WSReference parentReference)
    throws CampaignServicesException;
```

List all the offers, offer lists, and folders under the optional parent folder.

Once retrieved, each *WSComponentOrFolderInfo* instance returned can be used as is, for example to display the next level of the folder hierarchy, the attribute APIs can be used to fetch or update any contained offers.

Parameter

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

parentReference: optional Reference of the parent folder to list. Only the immediate child offers, offer lists and folders of this parent folder will be enumerated, so successive calls to this API are

needed to navigate the entire folder hierarchy (however, it typically is very shallow). If no parent is supplied, then all the components and folders under the root are returned.

Throws *InvalidFolderException* if there is a problem with the specified parent folder Reference.

A typed *List* of zero or more *WSComponentOrFolderInfo* data wrapper instances, one for each matched component or folder.

Errors

InvalidFolderException

InvalidExecutionContextException, AuthorizationException

searchOffersBasic

```
List<WSOfferInfo>
   searchOffersBasic(String userCredential, Locale requestedLocale,
      String partitionName, long folderID,
      String searchCriteria, boolean includeRetired,
      int pageOffset, int pageSize)
   throws CampaignServicesException;
```

Enumerate a "page" of offers that contain the given search criteria in the name, description, createBy or offer code fields, beginning with the specified page offset. Search is based on the optional Folder input. (If a folderID of 0 is provided, the root offer folder is used by default). Matches are returned based on a "contains" match for the search string.

Once retrieved, each *WSOfferInfo* returned can be used as is, for example to display a summary list, or the attribute methods can be used to fetch or update the offer's attributes.

No state is maintained by this API, so it is possible to make calls to it in any order.

Parameters

userCredential: the client user credential.

requestedLocale: optional locale to use for this request.

partitionName: optional name of the campaign partition to use.

folderID: the ID of the Offer folder to be searched; if a folderID of 0 is specified, the root folder will be searched.

searchCriteria: the search phrase.

includeRetired: the boolean value that specifies whether search results will include retired offers. Valid values are TRUE and FALSE, with TRUE indicating that retired offers are included, and FALSE indicating that retired offers are not included.

pageOffset: the starting offset of all possible components to begin the enumeration (zero-valued). For example, if the enumeration matches 1000 offers and this value is set to 10, then

the page would start at the 11th component. A RangeException is thrown if the supplied offset is out of range.

pageSize: the maximum number of matched components to return for the page (cannot exceed 500).

Returns

Returns a typed *List* of zero or more *Offer* data wrapper instances, one for each returned offer in the page.

Errors

RangeException

listOffersByPage

```
List<OfferInfo>
   listOffersByPage(String userCredential, String partitionName,
      Locale requestedLocale, Attribute[] attributes,
      long pageOffset, int pageSize)
      throws AttributeNotFoundException, InvalidAttributeException,
      RangeException;
```

Enumerate a "page" of offers that match the optional attribute values, beginning with the specified page offset. Folders are ignored. Matches are returned based on a “like” match for strings (where the match is considered sufficient if a string contains the queried value), and exact match for dates and numbers.

Once retrieved, each *OfferInfo* returned can be used as is, such as to display a summary list, or the attribute methods can be used to fetch or update the offer’s attributes.

No state is maintained by this API, so it is possible to make calls to it in any order.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

attributes: optional array of attributes to match; the attribute’s name, data type, and value(s) are used to determine the match; if the attribute supports arrays, then all values specified must match. The implied match operator is an OR, so components that match any of the supplied attribute values will be returned.

Throws *AttributeNotFoundException* if an attribute name does not exist or *InvalidAttributeException* if one or more of the supplied attributes is invalid.

pageOffset: the starting offset of all possible components to begin the enumeration (zero-valued). For example, if the enumeration matches 1000 offers and this value is set to 10, then

the page would start at the 11th component. A *RangeException* is thrown if the supplied offset is out of range.

pageSize: the maximum number of matched components to return for the page (cannot exceed 500).

Returns

A typed *List* of zero or more *OfferInfo* data wrapper instances, one for each matched component in the page.

Errors

AttributeNotFoundException, InvalidAttributeException, RangeException

InvalidExecutionContextException, AuthorizationException

validateOffers

```
List<OfferValidationInfo>
   validateOffers(String userCredential, String partitionName,
      Locale requestedLocale,
      OfferCodeOrName[] codeOrNames);
```

Validate the supplied offer codes or offer list names and return validation information for each. “Validation” consists of checking to see whether one and only one matching offer or offer list exists in the database.

The OfferValidationInfo object contains an error message instead of Offer Info if zero offers or offer lists are found matching the given code or name. An error is also returned instead of a match if the given code or name matches multiple offers or offer lists. List is returned in the same order as given. Offer codes and offer list names are validated based on exact match with offers.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

codeOrNames: an array of all offer codes or offer list names to validate

No exceptions are thrown by this method; instead validation information is returned for all codes or names supplied.

Returns

A typed *List* of zero or more *OfferValidationInfo* data wrapper instances.

Errors

None

createOffer

```
OfferInfo createOffer(String userCredential, String partitionName,
      Locale requestedLocale,
      String securityPolicyName,
      String name, String templateName,
      Attribute[] attributes)
   throws InvalidFolderException, AttributeNotFoundException,
      InvalidAttributeException;
```

```
public WSOfferInfo createOffer(String authorizationLoginName, String
      partitionName, Locale requestedLocale, String
      securityPolicyName, String name, long folderID,
      String templateName, WSAttribute[] wsAttributes) throws
CampaignServicesException;
```

Create a new offer for the client, applying the specified attributes.

Parameters

authorizationLoginName: User name of the user creating the offer. Users must be granted the Add Offers permission to use this method.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

securityPolicyName: optional name of the campaign security policy to use to create the offer. All subsequent operations on this offer will use this policy. If not defined, the *Global* policy will be used.

name: the name to assign the new offer instance (its *uacName* attribute).

folderID: the ID of the Offer folder where the offer will be created. This ID will be validated for the correctness and an exception will be thrown if the ID is invalid.

templateName: required (unique) name of an Existing Offer Template that should be used for the new Offer.

wsAttributes: an array of initialization attributes; any supplied attributes will overwrite the offer's default values; others are left untouched. For example, if a *uacOfferCode* attribute is supplied,

it will be used instead of an auto-generated one. It is up to the client to determine the attributes required by the offer, their types, etc.

Throws *CampaignServicesException* if one of the following conditions occurs:

- The folderID parameter is invalid (non-existent or not of type offer)
- The user is not authorized to perform this operation
- Invalid attributes are supplied in wsAttributes
- Other runtime exceptions occur

Returns

A single instance of *OfferInfo* for the created offer.

Errors

CampaignServicesException

retireOffers

```
void retireOffers(String userCredential, String partitionName,
      Locale requestedLocale, WSReference[] references)
   throws CampaignServicesException;
```

Retires one or more existing offers.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

references: an array of References of the offers to be retired. *InvalidComponentException* is thrown if there is a problem with a particular reference, or an offer does not exist.

Returns

Nothing.

Errors

InvalidComponentException

AuthorizationException, DataException

deleteOffers

```
void deleteOffers(String userCredential, String partitionName,
      Locale requestedLocale, WSReference[] references)
   throws CampaignServicesException;
```

Deletes one or more existing offers.

Parameters

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

reference: an array of References of the offers to be deleted. *InvalidComponentException* is thrown if there is a problem with a specified reference, or an offer does not exist.

Returns

Nothing.

Errors

InvalidComponentException

AuthorizationException, DataException

listOfferTemplates

```
List<WSOfferTemplateInfo>
    listOfferTemplates(String userCredential, String partitionName,
       Locale requestedLocale)
    throws CampaignServicesException;
```

List all offer templates that the user has permissions to view.

Once retrieved, each *WSOfferTemplateInfo* instance returned can be used as is, or one or more of the attribute APIs can be used to fetch or update any listed template.

Parameter

userCredential: the client user credential

requestedLocale: optional locale to use for this request

partitionName: optional name of the campaign partition to use

Returns

A typed *List* of zero or more *WSOfferTemplateInfo* data wrapper instances, one for each returned template.

Errors

InvalidExecutionContextException, AuthorizationException

DataException

bulkCreateOffers

```
WSOfferInfoStatus[] bulkCreateOffers(String authorizationLoginName,
String partitionName, Locale requestedLocale,
String securityPolicyName, String templateName, long folderID,
WSBulkOfferInfo[] offers)
throws CampaignServicesException;
```

Creates offers in bulk with the attributes for each offer specified in the *offers* parameter. All of the offers are created under the specified *folderID* using the specified *templateName*.

Parameter

authorizationLoginName: the client user credential

partitionName: optional name of the campaign partition to use

requestedLocale: optional locale to use for this request.

securityPolicyName: optional name of the campaign security policy to use to create the offer. If not defined, the Global policy will be used.

templateName: Name of the existing offer template in the system. All offers will be created using this template.

folderID: the ID of the Offer folder where the offers will be created. This ID will be validated and an exception will be thrown if the ID is invalid.

offers: an array of *WSBulkOfferInfo* objects that defines the offer name and attributes. See WSBulkOfferInfo data type for more details.

Returns

An array of *WSOfferInfoStatus* instances for each offer. Contains the status and offer information. The status indicates whether offer creation was successful or not.

Errors

CampaignServicesException

Using the API

This section explains how to use the Campaign Web Services API. It also shows an example of using the Campaign API Service to create an offer in Campaign.

There are two approaches for using the Campaign Services API:

- Using the Client API .jar
- Using WSDL

Using the Client API .jar

IBM Unica Campaign provide a client API that uses SOAP web services to interact with the Campaign web application. This wrapper is bundled into a `.jar` file that the client application can use to call the Campaign API.

This `.jar` file can be found at:

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/lib/CampaignServicesClient30.jar.`

The following example shows new offer creation at the root Offer folder level in Campaign. The same sample can be found at

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/samples/OfferAPI.java`

The example uses some dummy values for the parameters; your actual values may differ.

Also note that the URL for Campaign Web services is `http://host:port/Campaign/services/CampaignServices30Service`,

where host and port refer to the host name and port number of the machine where the Campaign web application is deployed.

If you use a provided sample, be sure to modify it to suit your client environment.

Code OfferAPI.java

```
import java.net.URL;
import java.util.Locale;

import com.unica.publicapi.campaign.campaignservices.CampaignServicesException;
import com.unica.publicapi.campaign.campaignservices.attribute.metadata.IAttribute-
Metadata;
import com.unica.publicapi.campaign.campaignser-
vices.soap.v30.CampaignServices30SoapClient;
import com.unica.publicapi.campaign.campaignservices.soap.v30.WSAttribute;
import com.unica.publicapi.campaign.campaignservices.soap.v30.WSOfferInfo;
import com.unica.publicapi.campaign.campaignservices.utils.WSAttributeUtils;

/**
 * This is the sample java client class that shows the usage of Campaign's SOAP ser-
vices API.
 * This sample uses CampaignServices30SoapClient facade to interact with Campaign web
service.
 * Here the creation of Offer is shown. Please refer to the API guide for more
details.
 *
 * @author AGijare
```

```
 *
 */
public class OfferAPI {

   /**
    * @param args
    */
   protected static CampaignServices30SoapClient CLIENT = null;

   private static void setup(){
      try {
         String protocol = "http"; //http or https
         String host = "localhost"; //Host name of deployed Campaign. Use proper
host name here.
         int port = 7001; //port number of deployed Campaign
         long timeOut = 2*60*1000; // 2 minutes
         String servicesURI = "/Campaign/services/CampaignServices30Service";
         CLIENT = new CampaignServices30SoapClient(
                new URL(protocol, host, port, servicesURI),
                timeOut);
      } catch (Exception exception) {
         exception.printStackTrace();
         System.exit(-1);
      }
   }

   public static void main(String[] args) {
      //Please change the values of following variables to match your environment.
      String userName = "user_name"; //login user name
      String partitionName= "partition1"; //Use proper partition name of Campaign
      Locale loc = Locale.US;
      String securityPolicy = "Global"; //Use your security policy of Campaign

      String offerName = "Offer1";
      String offerTemplate = "Offer Template"; // Template from which Offer will be
created.
      long folderID = 1002; //Actual ID of the folder where this offer will be cre-
ated. For folderID <=0, offer will be created at root level.

      //Attributes of Offer
      WSAttribute[] wsAttributes = {
            WSAttributeUtils.getWSTextAttribute(IAttributeMeta-
data.AC_OFFER_DESCRIPTION_ATTRIBUTE_NAME, null, new String[]{"description " + Sys-
tem.currentTimeMillis()})
      };

      setup();
```

```
        try {
            WSOfferInfo wsOfferInfo = CLIENT.createOffer(userName, partitionName, loc,
securityPolicy,
                    offerName, folderID, offerTemplate, wsAttributes);
            System.out.println("Created offer: " + wsOfferInfo.getName());
        } catch (CampaignServicesException e) {
            e.printStackTrace();
        }
    }
}
```

To compile and run the java sample shown above, you must include all dependent `.jar` files in the java classpath. The `CampaignServicesClient30.jar` file is dependent on Apache AXIS2 SOAP engine `.jar`s and other common Apache `.jar` files, which can be found in the `Campaign.war` file located at `<CAMPAIGN_HOME>/Campaign.war`. Extract the `.jar` files from `Campaign.war`, and include them in the java classpath.

Using WSDL

Campaign services can also be called by using the Campaign web services WSDL file, `CampaignServices30.wsdl`, which can be found at:

`http://host:port/Campaign/services/CampaignServices30Service?wsdl`

or in the Campaign distribution at `<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/lib/`.

The client java application needs to use the classes and stubs generated from the WSDL using any 3rd party WSDL-to-Java converter tool. IBM Unica recommends the use of Apache AXIS.

The javadocs created from stubs and classes generated from WSDL using Apache AXIS2 can be found at `<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/javadocs/index.html`.

All dependent `.jar` files must be included in the java classpath. The `CampaignServicesClient30.jar` file is dependent on Apache AXIS2 SOAP engine `.jar`s and other common Apache `.jar` files, which can be found in the `Campaign.war` file located at `<CAMPAIGN_HOME>/Campaign.war`. Extract the `.jar` files from `Campaign.war`, and include them in the java classpath.

Performance considerations

The current CampaignServices API implementation performance profile is similar to that of the application as experienced through the GUI. Some APIs are designed explicitly for

performance. In particular, the `listCampaignsByPage()` API allows for relatively efficient pagination.

The SOAP interface, by its nature, introduces latency and overhead because all data are converted into XML form, which in some cases is fairly verbose. For example, a simple loopback SOAP call can take 100ms on a typical network (Java 1.4.x was even slower). The API has been optimized for typical portal and other client application business use cases—such as `see listOffersByPage()`--so SOAP performance should be adequate.

However, the client must take care that it does not place too high a burden on normal CampaignServices servicing of Web user requests. In general, it is expected that an API user's processing needs do not exceed those of a typical IBM Unica Campaign web user.

Packaging

This specification is delivered as part of the CampaignServices Software Developer's Tookit (devkits) installed with IBM Unica Campaign.

The devkits directory laid down by the installer includes examples, build and text scripts, javadoc for public classes and interfaces, release notes, etc.