HCLSoftware

Unica Campaign API Guide V12.1.6

Contents

Chapter 1. Unica Campaign Rest APIs

Unica Campaign provides Rest APIs for various operations related to Campaign, Offers, Flowcharts, etc. You can find additional details on Campaign Rest APIs in Swagger documentation. To access Swagger documentation, perform the following steps.

- Log in to the Unica application.
- Navigate to **Settings > Campaign settings**.
- Click **API Documentation**.

Campaign has more APIs. Go to the following link for more details.

https://s3.amazonaws.com/help.hcltechsw.com/unica/Campaign/9.1.2/Campaign/REST_API/RESTAPI_parent.html

Use Platform user for authentication to use with Campaign Rest APIs. User authentication from third party directory services is not supported with Campaign or Optimize APIs.

Chapter 2. Unica Campaign SOAP API Overview

The Unica Campaign SOAP API specification defines version 3.0 of the Application Programming Interface, also referred to as Unica CampaignServices. This specification is delivered as part of the Unica CampaignServices Software Developer's Tookit (devkits) installed with Unica Campaign.

The `<CAMPAIGN_HOME>/devkits/CampaignServicesAPI` directory that is provided by the installer includes examples, build and text scripts, Javadoc for public classes and interfaces, and release notes.

The Unica CampaignServices SOAP API is designed to:

- Provide fine-grained create, discovery, read, and update access to Unica Campaign components, while insulating clients from underlying implementation details.
- Work along side the existing Unica Campaign user interface with minimal effects.
- Guarantee the validity of data.
- Meet the required security services of Unica Campaign.
- Support industry-standard SOAP (Simple Object Access Protocol), including secure authentication.

SOAP API design overview

The Unica Campaign Services SOAP API is a façade that provides a client view of a running Unica Campaign application instance. Only a subset of the capabilities of Unica Campaign are exposed, but it is enough to drive key aspects of Unica Campaign functionality.

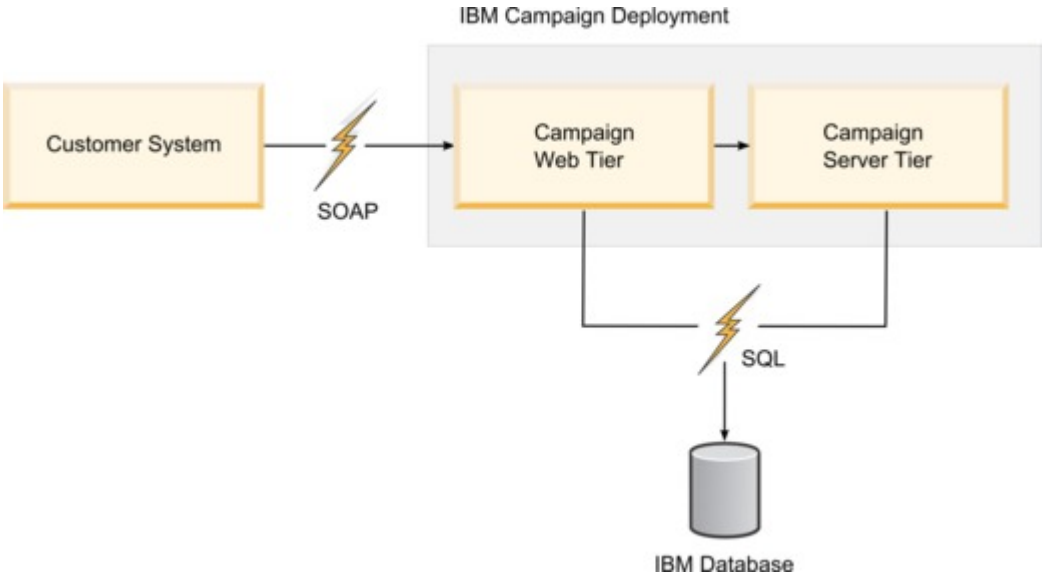
Features and diagram

The API is used concurrently with 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.



User authentication

Authentication deals with establishing a user's identity. 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 that are 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 throws `AuthorizationException`.

Locale

API requests provide for an optional **requestedLocale** parameter, which defines the locale to use for carrying out that particular request. If not defined, the API defaults to the 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.

Note: Some user-specified text, such as campaign descriptions, are in the locale of the user that specified the text. Unica Campaign does not attempt to localize such data. Only the information, warning, and error messages are 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 might change the state of underlying component instances that are managed by Unica Campaign, and these state changes might be persisted to the database.

SOAP API changes by version

The purpose of this topic is to identify changes to the Unica Campaign SOAP API for customers who are currently using the API. If you upgraded from a prior version, review the following information to determine whether you need to make changes to your application code.

Versions and backwards-compatibility

Future versions of the CampaignServices API are compatible with earlier versions, with all minor and maintenance releases that share a major version number. However, reserves the right to break compatibility with the earlier version for "dot zero" (x.0) major releases.

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

- New method added
- New data type that is added and its usage that is restricted to new methods
- New type added to an enumerated type
- a new version of an interface is defined

continues to support the published WSDL, SOAP client, and the version of Apache Axis that is used to implement the SOAP offering until at least the next major release. Practically speaking, this is accomplished by supporting several version-specific web services concurrently. (already supports several versions of this service internally.)

Changes in v9.1.2

There were no API changes in version in v9.1.2.

Note: A REST API was introduced in v9.1.2, in addition to the SOAP API described in this guide. The REST API is described separately.

Changes in v9.1.1

stopFlowchart includes a new input parameter, runId, to support a clustered listener environment.

Changes in v9.1

There were no API changes in version 9.1.

Changes in v9.0

There were no API changes in version 9.0.

Changes in v8.6

API changes implemented in v8.6:

- The SOAP engine was upgraded from AXIS version 1.4.1 to AXIS2 1.5.2.
- The WSDL was restructured to deal with issues that handle the required and optional parameters.
- The Client API `.jar` file changed due to the WSDL changes. As a result, the generated stubs and classes changed. The Client API method parameters did not change, but the constructors of supporting value objects were modified for the AXIS2 WSDL2Java converter usage.
- The Web Service URL points to:

  `http://<host>:<port>/Campaign/services/CampaignServices30Service`

  and the corresponding WSDL is retrieved at:

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

If you are upgrading to Unica Campaign version 8.6 or higher and you currently use the Unica Campaign Services API, you must modify your application code.

Depending on whether you use the client API or the WSDL, see the following sections for details:

- Existing implementations using the Client API jar file on page 5
- Existing implementations using WSDL directly on page 6

Existing implementations using the Client API jar file

This information pertains if you are upgrading to Unica Campaign version 8.6 or higher and you currently use the Client API `.jar` file to interact with the Unica Campaign web application.

Client API .jar file

Your Java™ application must use the `.jar` file that is at:

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

For a Java™ example that shows new offer creation, see OfferAPI.java on page 9. The same example can be found in your Unica Campaign installation at:

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

Dependent .jar files

As a result of the upgrade to AXIS2 version 1.5.2, your Java™ application must also upgrade to using the AXIS2 1.5.2 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 that is at `<CAMPAIGN_HOME>/Campaign.war`.

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

Client API constructor

While you construct the Client API object, change the web service URL and the Exception signature as shown in this example.

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

Parameterized constructors of the supporting classes

With the AXIS2 engine, the generated classes and stubs do not have parameterized constructors. Instead, these classes have only the default no-argument constructor with setters and getters for the members.

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

Existing implementations using WSDL directly

This information pertains if you are upgrading to Unica Campaign version 8.6 or higher and you currently use the WSDL to interact with the Unica Campaign web application. The WSDL of the Unica Campaign web service is used to generate client-side stubs and supporting classes by using any third-party converter tool. Examples provided here use the WSDL2Java tool from Apache AXIS2 1.5.2.

WSDL location and service URL

The Unica Campaign web service for Unica Campaign 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 1.5.2 can be used to generate the stubs and supporting Java™ classes from the WSDL. A sample Ant task is shown here.

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

Note: 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{
    //Please change host and port to match your environment.
    String serviceURL = "http://host:port/Campaign/services/CampaignServices30Service";
    CampaignServices30ServiceStub serviceStub = new CampaignServices30ServiceStub(serviceURL);
    long webServiceTimeout = 2*60*1000; // 2 minutes
    serviceStub._getServiceClient().getOptions().setTimeOutInMilliSeconds(webServiceTimeout); //Timeout in
 milliseconds.

    WSTextAttribute nameAttirbute = new WSTextAttribute();
    nameAttirbute.setMetadata(null);
    nameAttirbute.setName("uacOfferDescription");
    nameAttirbute.setValues(new String[]{"description " + System.currentTimeMillis()});

    WSTextAttribute[] wsAttributes = {nameAttirbute};
    // convert to WSAttributeArrays
    WSAttributeArrays obj = new WSAttributeArrays();
    obj.setTextAttributes(wsAttributes);
    //Please change the values of following variables to match your environment.
    String authorizationLoginName = "asm_admin"; //login user name
    String partitionName = "partition1"; //Use your security policy of Campaign
    String securityPolicyName = "Global Policy"; //Use your security policy of Campaign

    String offerName = "1st Offer"; //Name of the offer to be created.
    String templateName = "Offer Template"; //Existing offer template name.
    long folderID = 100; //Actual ID of the folder where this offer will be created.
    //For folderID <=0, offer will be created at root level.

    CreateOffer createOfferObject = new CreateOffer();
    createOfferObject.setAuthorizationLoginName(authorizationLoginName);
    createOfferObject.setPartitionName(partitionName);
```

```
    createOfferObject.setRequestedLocale(Locale.US.toString());
    createOfferObject.setSecurityPolicyName(securityPolicyName);
    createOfferObject.setName(offerName);
    createOfferObject.setFolderID(folderID);
    createOfferObject.setTemplateName(templateName);
    createOfferObject.setAttributes(obj);
    // make campaign WebService call
    WSCreateOfferResponse wsResponse = serviceStub.createOffer(createOfferObject);
    // process status
    WSRequestStatus status = wsResponse.getStatus();
    // done
    WSOfferInfo offerInfo = wsResponse.getOfferInfo();
    System.out.println("status = "+status.getStatusType());
    System.out.println("offerInfo = "+offerInfo.getName());
} catch (Exception exception) {
    //Handle the Exception here.
    exception.printStackTrace();
}
```

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.

SOAP API references

The following references were used to prepare the Campaign SOAP API 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`)

Chapter 3. Using the Unica Campaign SOAP API

To use the Unica Campaign Web Services SOAP API, you can use the client API .jar file or use the WSDL directly. An example shows how to use the .jar file method to create an offer.

Using the client API .jar to call Unica Campaign services

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

The `.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 Unica Campaign. The same sample can be found at:

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

Note: The example uses some dummy values for the parameters; your actual values might differ.

Also, the URL for Unica 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 Unica Campaign web application is deployed.

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

OfferAPI.java

To compile and run this Java™ sample, you must include all dependent `.jar` files in the Java™ classpath. The `CampaignServicesClient30.jar` file depends on Apache AXIS2 SOAP engine `.jar` files and other common Apache `.jar` files, which can be found in `<CAMPAIGN_HOME>/Campaign.war`. Extract the `.jar` files from `Campaign.war`, and include them in the Java™ classpath.

```
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.campaignservices.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 Unica Campaign SOAP
```

```
services API.
 * This sample uses CampaignServices30SoapClient facade to interact with Unica 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 created. 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 "
+ System.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();
        }
    }
}
```

Using the WSDL to call Unica Campaign services

Unica Campaign services can be called by using the Unica Campaign web services WSDL file: `CampaignServices30.wsdl`.

The `CampaignServices30.wsdl` file can be found at:

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

or in the Unica Campaign distribution at:

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/lib/`

The client Java™ application must use the classes and stubs that are generated from the WSDL by using any third-party WSDL-to-Java converter tool. recommends the use of Apache AXIS.

The javadocs that are created from stubs and classes that are generated from WSDL by using Apache AXIS2 can be found at:

`<CAMPAIGN_HOME>/devkits/CampaignServicesAPI/javadocs/index.html`

Note: All dependent `.jar` files must be included in the Java™ classpath. The `CampaignServicesClient30.jar` file is dependent on Apache AXIS2 SOAP engine `.jar` files and other common Apache `.jar` files, which can be found in the `Campaign.war` file that is 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 is converted into XML form, which in some cases is fairly verbose. For example, a simple loopback SOAP call can take 100 ms on a typical network (Java™ 1.4.x was even slower). The API is 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 Unica Campaign web user.

Chapter 4. SOAP API data types

The Unica Campaign Services SOAP API uses the following public data types.

WSReference

A simple wrapper around a database identifier:

- **componentTypeEnum**: an enumerated type that indicates 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 8.1.2.3.
- *maintenance*: optional Integer defining the maintenance number of the version, if applicable, such as '2' of the full version 8.1.2.3. Never supplied with an API version.
- *patch*: optional Integer defining the patch release number, if applicable, such as '3' of the full version 8.1.2.3. 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. (*apiVersion* includes only major and minor version information.)
- *campaignVersion*: a *WSVersion* instance, defining the full version of the underlying 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 that is defined by Unica Campaign.
- CUSTOM: an attribute that is defined by another application, the customer, or some other 3rd-party.
- INPUT_PARAMETER: an input parameter, such as an attribute used to run an Unica Campaign flowchart.
- OUTPUT_PARAMETER: an output parameter, such as an attribute whose value is completed as the result of a flowchart that is run in Unica Campaign.

WSAttributeStatusEnum

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

- ACTIVE: the attribute is active and might be used at will.
- RETIRED: the attribute is 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. 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 can 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 did not start yet.
- RUNNING: run in progress.
- CANCELLED: the run was canceled, either by a Unica Campaign user or this API.
- SUCCEEDED: the run completed successfully.
- FAILED: the run failed; error details are reported separately. (See WSRunResults on page 19.)

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 that is specified by another application or customer.

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

Components generally have many attributes that are associated with them, which are exposed by the CampaignServices API as a specially typed Map that is called an *AttributeMap*. Attribute data is represented as a 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 Unica Campaign user.

  Generally, this name is not the display name that would be presented to a Unica Campaign or client user. It may be standardized by the API, such as *uacDescription*, it might be assigned by Unica Campaign when publishing flowcharts, or it might be assigned by the 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). See WSAttributeMetadata on page 16.
- *Values*: an array of zero or more 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 values captured as a *Double*. If the currency code is not supplied, the default used by Unica Campaign is assumed.

    See `http://www.xe.com/symbols.php` for a list of countries, currency symbols, and codes. The locale that is used for a currency value might 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).

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

WSAttributeMetadata

WSAttributeMetadata defines information about the data of a particular typed attribute, such as value data type, localized text (display name, description, prompts), its default value, permissible value range, select type, options (if single or multiple select). As with attributes, attribute metadata is typed. 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, are all be typed *Double.*

Descriptions, labels, and other attribute metadata text are 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 that is 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.

  Note: The attribute name must be unique, is never localized, and has length restrictions (which depend on the character content and database). The name is not case-sensitive 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 or values are auto-generated by 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 that is used for the attribute, such as SINGLE. Must be NONE for Unica Campaign and cell attributes, or if no options are provided.
- *componentTypeEnum*: a *WSComponentTypeEnum* of all possible Unica Campaign components that are exposed by the API, such as CAMPAIGN, FOLDER.
- *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 typed, so for example a *WSTextAttributeMetadata* can only have a *WSTextAttributeOption* bound to it.

Note: 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. 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 tooltip 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 Unica Campaign.

WSComponentOrFolderInfo

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

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. Cell codes can be forced unique by setting the 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 that defines contact information for one response:
    - *typeCode*: a string that defines 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 refers 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 run generated the results, such as PRODUCTION_PROCESS (see *WSRunTypeEnum*).
- *statusEnum*: an enumerated type that defines 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; is null if the run did not start.
- *endDate*: as with *startDate*, but the datetime when the run ended (success or failure); is null if the run did not start or is not finished yet.
- *count*: optional total count of contacts that are selected by the run; might be zero or null if the run did not complete.

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 that indicates whether the *codeOrName* field is a presumed offer code (true) or the name of an offer list (false).
- *codeOrName*: the unique offer code (*uacOfferCode*) of 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 alphanumeric 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 that indicates the offer attributes.

WSOfferInfoStatus

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

It contains the following fields:

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

Chapter 5. SOAP API methods

The Unica Campaign Services SOAP API uses the following methods.

SOAP API methods: Service

The Unica Campaign SOAP API provides a way to obtain 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 Unica Campaign instance.

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

Parameters

None.

Returns

Returns a *WSServiceInfo* instance.

Errors

None.

SOAP API methods: Attributes

Most component instance data can be exposed by the Unica Campaign SOAP API as attributes or attribute metadata.

In some cases, the attribute metadata definitions are global to Unica Campaign (such as campaign custom attributes). In other cases, 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.

Note: Only components that are active and that the client has access to are exposed by this API. Public support is limited to a subset of the available APIs.

getAttributesByName

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

Fetches the named attributes that are 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 user's locale preferences are used. Normal locale defaulting algorithm is 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

Note: 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 or out-of-range, throw *InvalidAttributeException*.
4. Else throw *AttributeNotFoundException*if the named attribute does not exist.

Note: In the event of an exception, none of the updates are 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 24.)

attributes: an array of attributes to be updated; the attribute name is used to locate the attribute to update and the new values are used to update the existing attribute's value as a single object of the proper type or an array, if applicable. (Refer to SOAP API common exceptions on page 63.)

Returns

None.

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 definitions that are 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 that contains 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 returns 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 is 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

None.

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 that is 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 is 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.

Note: In the event of an exception, none of the updates is 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 that contains the desired attributes. If not supplied, then the update is restricted to global metadata definitions. Throws *InvalidComponentException* if the supplied reference is invalid.

allowCreate: if true, metadata definitions that currently do not exist are 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 27.)

Returns

None.

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 finds all components that reference the deleted metadata, and update as appropriate.

Note: However, in the event of an exception, none of the deletes is 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 that contains the attributes to be deleted. If not supplied, then the delete is restricted to global metadata definitions. Throws *InvalidComponentException* if the supplied reference is invalid.

Note: If the optional array of names of the attribute metadata is not supplied, then this method attempts to delete all custom attribute metadata that is 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

None.

Errors

InvalidComponentException, AttributeNotFoundException, InvalidAttributeException

AuthorizationException, DataException

SOAP API methods: Campaigns and flowcharts

The Unica Campaign SOAP API supports the following operations on campaigns and flowcharts (subject to security permissions).

- create a campaign
- discovery (list campaigns by various criteria)
- attribute create, read, and update (via attribute APIs)
- stop flowchart run

Campaigns have a number of standard attributes that are 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).

The standard campaign attributes are listed below:

- *uacName*: campaign name (not guaranteed unique).
- *uacDescription*: optional string that describes the campaign.
- *uacCampaignCode*: a string code that uniquely identifies the campaign. Typically generated by Campaign, but may be provided by the client.
- *uacCreateDate*: a Calendar that indicates the date and time when the campaign was created by the server.
- *uacUpdateDate*: a Calendar that indicates the date and time when the campaign was last updated by the server.
- *uacInitiative*: optional string that defines the campaign initiative.
- *uacObjectives*: optional string that identifies the objectives of the campaign.
- *uacStartDate*: an optional Calendar that provides the date and time when the campaign was started by the server, or is scheduled to start.
- *uacEndDate*: as with *uacStartDate*, but defines the date and time when the campaign was completed or is scheduled to be completed. Must be after the *uacStartDate*.
- *uacLastRunDate*: an optional Calendar that indicates the date and time when any flowchart bound to the campaign was last run (else null).
- *uacExternalLinkOwner*: an optional string that defines the name of the owner of an external link (see uacExternalLinkReference attribute). use only; must be one of the following:
    - "Plan" (now known as Unica Plan)
    - "Collaborate" (now known as Unica Collaborate)
- *uacExternalLinkId*: an optional numeric database identifier that is assigned by another application to an object linked to this campaign. 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 that is returned by a previous or future call to this method, or the `createCampaign()` method, or the value that is generated for a campaign that is created by the Unica Campaign GUI.

Note: The use of this method is optional, as the `createCampaign()` API generates 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 might be null.

Returns

The generated campaign code.

Errors

AuthorizationException, DataException

deleteCampaigns

Example

```
public WSDeleteCampaignsResponse deleteCampaigns(String userCredential,
String partitionName, Locale requestedLocale, WSReference[] wsReferences)
 throws CampaignServicesException
```

Deletes specified campaigns from the system.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsReference: references of campaigns to delete.

Returns

Returns object of type WSDeleteCampaignsResponse.

Errors

Throws CampaignServicesException if the campaign does not exist or the reference is invalid or no references provided.

createCampaign

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

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

To create campaigns under a specific folder, use the `WSReference` parameter to specify the campaign folder.

Example

```
private static void createCampaign
    (String userName, String partitionName, Locale loc, String securityPolicy,
                      String campaignName, long campaignfolderID)
{
    WSAttribute[] wsAttributes = { WSAttributeUtils.getWSTextAttribute
          (IAttributeMetadata.AC_CAMPAIGN_DESCRIPTION_ATTRIBUTE_NAME, null,new String[]
          { "description " + System.currentTimeMillis() }) };

  try
  {
    WSReference wsReference = WSAttributeUtils.getWSReference
          (WSComponentTypeEnum.FOLDER, campaignfolderID);
    WSCampaignInfo wsCampaignInfo = CLIENT.createCampaign
          (userName, partitionName, loc, securityPolicy, wsReference, campaignName, wsAttributes);

    System.out.println("Created Campaign with Name: " + wsCampaignInfo.getName()
           + " CampaignCode: " + wsCampaignInfo.getCampaignCode());
  }
  catch (CampaignServicesException e)
  {
    e.printStackTrace();
  }
}
```

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 use this policy. If not defined, the Global policy is used.

wsReference: optional name of the campaign folder in which the campaigns are to be created.

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

attributes: an optional array of initialization attributes; any supplied attributes overwrite the campaign's default values; others are left untouched. For example, if a *uacCampaignCode* attribute is supplied, it is used instead of an auto-generated one. It is up to the client to determine the attributes that are 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 Unica 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 values are used to determine the match; if the attribute supports arrays, then all values that are specified must match. The implied match operator is an AND, so only campaigns that match all the supplied attribute values are 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

stopFlowchart

```
stopFlowchart(int pid, int runid)
```

This API stops a running flowchart. For a single listener configuration, a flowchart run can be uniquely identified by the PID that is associated with the flowchart run. The PID indicates the process ID of the `unica_acsvr` process. If multiple Unica Campaign listeners are configured, you must include both the run-id that is associated with the flowchart run and the PID.

Parameters

pid: process ID of the `unica_acsvr` process associated with a flowchart run.

runid: run-id associated with a flowchart run. Required parameter for a clustered listener configuration. Optional parameter if a single listener is configured.

Returns

None

Errors

None

SOAP API methods: Target cells

Target cells are an abstraction for some known subsets of campaign results that are managed by 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 Unica Campaign SOAP 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 that are associated with one or more cells

Target cells have a number of standard attributes that are 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 that describes the flowchart.
- *uacCellCode*: a code string that uniquely identifies the cell. Typically auto-generated by Unica Campaign, but may be provided by the client.
- *uacCreateDate*: a Calendar instance that gives the date & time when the cell was created by the server.
- *uacUpdateDate*: a Calendar instance that defines when the last time the cell was updated by the server.
- *uacIsControl*: a boolean that indicates whether 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 that indicates whether the cell is approved (true) or not (false).
- *uacIsReadOnly*: a boolean that indicates whether the cell is read-only (true) or not (false).
- *uacDisplayOrder*: an integer that gives the order of this cell (row) relative to others in the target cell spreadsheet.
- *uacIsTopDown*: a boolean that indicates whether 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 through the 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 exist.

Once the target cell is created, attribute values can be changed by 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 that contains 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 overwrites the corresponding cell attribute's default values; others are left untouched. It is up to the client to determine the attributes that are required by the cell, their types, etc. Accumulates an *InvalidAttributeException* if there is a problem with a specified attribute.

If any exceptions are accumulated, this method throws a *CompositeException* and all creates are undone. The exception's list of causes include an exception for each attribute that caused the error and include a numeric index instead of the *reference*, the name of the attribute, and usually the offending value. The cause list is 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 exist.

After the target cell is created, attribute values can be changed by 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 that contains 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 overwrite the corresponding cell attribute's default values; others are left untouched. It is up to the client to determine the attributes that are required by the cell, their types, etc. Accumulates an *InvalidAttributeException* if there is a problem with a specified attribute.

If any exceptions are accumulated, this method throws a *CompositeException* and all creates are undone. The exception's list of causes include an exception for each attribute that caused the error and include a numeric index instead of the *reference*, and the name of the attribute, etc. The cause list is 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 are 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*.

After a target cell is located, for each attribute that is 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 or out-of-range, accumulate an *InvalidAttributeException*.
3. Else accumulate *AttributeNotFoundException* if the named attribute does not exist.

If any exceptions are accumulated, this method throws a *CompositeException* and all updates are undone. The exception's list of causes include the exceptions that are listed above. For each attribute that caused the error, both the reference and the attribute name are 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 values 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

None.

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 nonexistent cell.

Returns

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

Each run status is 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 did not start. 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).

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 accumulated, this method throws a *CompositeException* and all deletes are undone. The exception's list of causes include the exceptions that are listed above. For each cell that caused the error, the reference is recorded.

Returns

None.

Errors

CompositeException

AuthorizationException, DataException

updateTemplateAttributes

Example

```
updateTemplateAttributes
(String userCredential, String partitionName, Locale requestedLocale,
WSReference wsReference, boolean allowCreate,
boolean clearExisting, WSAttribute[] wsStaticAttributes,
WSAttribute[] wsHiddenAttributes,WSAttribute[] wsParametricAttributes)
 throws CampaignServicesException
```

Updates attributes of templates specified

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsCampaignReference: reference of the parent campaign.

allowCreate: not used now.

clearExisting: flag if set true, all earlier values in the template that are not sent in request are cleared.

wsStaticAttributes: list of static attributes in the template.

wsHiddenAttributes: list of hidden attributes in the template.

wsParametricAttributes: list of parametric attributes in the template.

Returns

None.

Errors

Throws CampaignServicesException if the offer template does not exist or the reference is invalid or no references provided.

listBottomUpTargetCells

Example

```
public List <WSTargetCellDetails>
listBottomUpTargetCells(String userCredential, String partitionName,
Locale requestedLocale, WSReference wsCampaignReference)
 throws CampaignServicesException
```

Lists information about all bottom up target cells that currently exist for the specified campaign.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsCampaignReference: reference of the parent campaign.

Returns

Returns a list of zero or more WSTargetCellDetails instances for the matched cells

Errors

Throws CampaignServicesException if the campaign does not exist or the reference is invalid.

SOAP API methods: Analytics

The Unica Campaign SOAP API supports the retrieval of simple metrics from 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

SOAP API methods: Offers, offer lists, offer templates

The Unica Campaign SOAP API supports the following operations related to 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 that are associated with them. This list can be extended by the client by adding custom attribute metadata definitions (see the Attributes APIs).

Standard offer attributes are:

- *uacName*: offer name.
- *uacDescription*: optional string that describes the offer.
- *uacOfferCode*: a code string that uniquely identifies the offer. Typically generated by Unica Campaign, but may be provided by the client.
- *uacCreateDate*: a calendar instance that indicates the date and time when the offer was created by the server.
- *uacUpdateDate*: a calendar instance that indicates 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 that describes the offer template.
- *uacCreateDate*: a calendar instance that indicates the date and time when the offer template was created by the server.
- *uacUpdateDate*: a calendar instance that indicates 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.

After retrieved, each *WSComponentOrFolderInfo* instance that is 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 are 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 is searched.

searchCriteria: the search phrase.

includeRetired: the boolean value that specifies whether search results 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.

After 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 values are used to determine the match; if the attribute supports arrays, then all values that are specified must match. The implied match operator is an OR, so components that match any of the supplied attribute values are 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 (zerovalued). 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

createSmartOfferList

Example

```
public WSCreateSmartOfferListResponse createSmartOfferList
(String userCredential, String partitionName, Locale requestedLocale,
String name, String description,String policyName,
WSReference parentFolder,WSSmartListInfo offerListInfo,
WSApplicationTypeEnum createdBy,long creatorObjectId)
 throws CampaignServicesException
```

Creates new smart offer list.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

name: name of folder to create.

description: description for new folder.

securityPolicyName: name of security policy to use.

parentFolder: id of folder where offerlist must be created.

offerListInfo: object of type WSSmartListInfo.

createdBy: (optional) object of type WSApplicationTypeEnum indicates which application created the folder. Possible values- Campaign/Plan/Collaborate/Deliver. If not provided, campaign is used.

creatorObjectId: (optional) used by Plan (HCL Plan) to link a folder in Campaign with a folder in Plan.

Returns

Returns object of type WSCreateSmartOfferListResponse.

Errors

Throws CampaignServicesException if parentFolder id is invalid or offerListInfo is not provided.

Throws CampaignServicesException if list name is duplicate.

createStaticOfferList

Example

```
public WSCreateStaticOfferListResponse createStaticOfferList
(String userCredential, String partitionName, Locale requestedLocale, String name,
String description,String policyName,WSReference parentFolder,WSReference[]
listMembers,WSApplicationTypeEnum createdBy,long creatorObjectId)
 throws CampaignServicesException
```

Creates new static offer list.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

name: name of folder to create.

description: description for new folder.

securityPolicyName: name of security policy to use.

parentFolder: id of folder where offerlist must be created.

listMembers: references to offers to be included in offerlist.

createdBy: (optional) object of type WSApplicationTypeEnum indicates which application created the folder. Possible values- Campaign/Plan/Collaborate/Deliver. If not provided, Campaign is used.

creatorObjectId: (optional) used by Plan (HCL Plan) to link a folder in Campaign with a folder in Plan.

Returns

Returns object of type WSCreateStaticOfferListResponse

Errors

Throws CampaignServicesException if parentFolder id is invalid or listMembers are invalid.

Throws CampaignServicesException if list name is duplicate.

getOffers

Example

```
public WSGetOffersResponse getOffers
(String userCredential, String partitionName,
Locale requestedLocale,WSReference[] wsReferences)
 throws CampaignServicesException
```

Lists details of offers as specified in request.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsCampaignReference: reference of the parent campaign.

Returns

Returns object of type WSGetOffersResponse.

Errors

Throws CampaignServicesException if the offer does not exist or the reference is invalid or no references provided.

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 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.

Note: 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.

editOfferList

Example

```
public WSEditOfferListResponse editOfferList(String userCredential,
String partitionName, Locale requestedLocale, WSReference listReference,
boolean isSmartList,String name, String description,
WSReference[] listMembers,WSSmartListInfo offerListInfo,
Long creatorObjectId, boolean clearExisting)
throws CampaignServicesException
```

Updates smart and static offerlist details.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

listReference: reference to offer list.

isSmartList: flag indicates whether list is smart or static.

name: name of folder to create.

description: description for new folder.

listMembers: references to offers to be included in offerlist.

offerListInfo: object of type WSSmartListInfo.

creatorObjectId: (optional) used by Plan (HCL Plan) to link a folder in Campaign with a folder in Plan.

clearExisting: flag indicates whether existing information must be cleared. If true, existing list members are cleared before adding new members. If false, new members are appended to existing members.

Returns

Returns object of type WSEditOfferListResponse.

Errors

Throws CampaignServicesException if parentFolder id is invalid or offerListInfo is not provided or listMembers are invalid.

Throws CampaignServicesException if list name is duplicate.

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 that creates 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 use this policy. If not defined, the *Global* policy is used.

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

folderID: the ID of the Offer folder where the offer is created. This ID is validated for the correctness and an exception is 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 overwrite the offer's default values; others are left untouched. For example, if a *uacOfferCode* attribute is supplied, it is used instead of an auto-generated one. It is up to the client to determine the attributes that are 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

None.

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

None.

Errors

InvalidComponentException

AuthorizationException, DataException

deleteOffersAndLists

Example

```
public WSDeleteOffersAndListsResponse deleteOffersAndLists
(String userCredential, String partitionName, Locale requestedLocale,
 WSReference[] offers)
 throws CampaignServicesException
```

Deletes the offers and lists specified.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

offers: array of references of either offer or offerlist.

Returns

Returns object of type WSGetOfferListMembersResponse.

Errors

Throws CampaignServicesException if the offer id or offerlist id is invalid.

listOfferTemplates

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

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

After retrieved, each *WSOfferTemplateInfo* instance that is 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

createTemplate

Example

```
createTemplate(String userCredential, String partitionName, Locale requestedLocale,
 String name, String securityPolicyName ,WSAttribute[]
 wsStaticAttributes,WSAttribute[] wsHiddenAttributes,
WSAttribute[] wsParametricAttributes)
 throws CampaignServicesException
```

Creates a new offer template.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

name: name of newly created offer template.

securityPolicyName: name of security policy to use.

wsStaticAttributes: list of static attributes in template.

wsHiddenAttributes: list of hidden attributes in template.

wsParametricAttributes: list of parametric attributes in template.

Returns

Returns object of type WSCreateTemplateResponse.

Errors

Throws CampaignServicesException if the offer template does not exist or the reference is invalid or no references provided.

getOfferTemplate

Example

```
public WSGetOfferTemplateResponse getOfferTemplate(String userCredential,
 String partitionName, Locale requestedLocale,WSReference[] wsReferences)
 throws CampaignServicesException
```

Lists details of offer templates as specified in references.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsCampaignReference: reference of the parent campaign.

Returns

Returns object of type WSGetOfferTemplateResponse.

Errors

Throws CampaignServicesException if the offer template does not exist or the reference is invalid or no references provided.

retireOfferTemplates

Example

```
public WSGenerateOfferCodeResponse generateOfferCodes
(String userCredential, String partitionName,Locale requestedLocale,
String offerName, WSReference template)
 throws CampaignServicesException
```

Retires one or more offer templates specified.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsCampaignReference: reference of the parent campaign.

Returns

Returns object of type WSRetireOfferTemplatesResponse.

Errors

Throws CampaignServicesException if the offer template does not exist or the reference is invalid or no references provided.

getOffersAndListsByPage

Example

```
public WSGetOffersAndListsByPageResponse getOffersAndListsByPage
(String userCredential, String partitionName, Locale requestedLocale,
 WSComponentTypeEnum type,
int pageSize, int pageOffset)
 throws CampaignServicesException
```

List offers or offerlists by page.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

type: type indicating whether offers are requested or offerlists.

pageSize: the maximum number of matched components to return for the page.

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.

Returns

Returns object of type WSGetOffersAndListsByPageResponse.

Errors

None.

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 that is specified in the *offers* parameter. All of the offers are created under the specified *folderID* by 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 is used.

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

folderID: the ID of the Offer folder where the offers are created. This ID is validated and an exception is 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

getOfferListDetails

Example

```
public WSGetOfferListDetailsResponse getOfferListDetails(String userCredential,
String partitionName, Locale requestedLocale, WSReference listReference)
 throws CampaignServicesException {
```

List details of the offer list specified.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

listReference: reference to offer list.

Returns

Returns object of type WSGetOfferListDetailsResponse.

Errors

Throws CampaignServicesException if list references are invalid.

getOfferListMembers

Example

```
public WSGetOfferListMembersResponse getOfferListMembers
(String userCredential, String partitionName, Locale requestedLocale,
 WSReference listReference)
 throws CampaignServicesException {
```

Lists offers information in specified offer list.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

listReference: reference to offer list.

Returns

Returns object of type WSDeleteOffersAndListsResponse.

Errors

Throws CampaignServicesException if offerlist id is invalid.

getOffersByQuery

Example

```
public WSGetOffersByQueryResponse getOffersByQuery(String user_credential,
 String partition_name, Locale locale, String query, Integer maxSize,
Boolean includeSubFolder, WSReference[] scopeFolders)
 throws CampaignServicesException
```

Lists offers matching provided offers.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

query: query to find offers. Format of query is the same as used in smart offerlist.

maxSize: maximum records to list.

includeSubFolder: flag that indicates whether a subfolder must be included in search.

scopeFolders: list of folder references to be searched in for offers.

Returns

Returns object of type WSGetOffersByQueryResponse.

Errors

Throws CampaignServicesException if folder references are invalid.

retireOfferLists

Example

```
public void retireOfferLists(String user_credential, String partition_name,
Locale locale, WSReference[] wsReferences)
 throws CampaignServicesException
```

Retires one or more offerlists specified.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

wsReferences: reference to offer lists.

Returns

None.

Errors

Throws CampaignServicesException if list references are invalid.

createFolder

Example

```
public WSCreateFolderResponse createFolder(String userCredential,
String partitionName, Locale requestedLocale,String name,
String description,String securityPolicyName,
long parentFolderId,WSFolderTypeEnum folderType,
WSApplicationTypeEnum createdBy,long creatorObjectId)
 throws CampaignServicesException
```

Creates a new folder of type campaign/offer/sessions/segments.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

name: name of folder to create.

description: description for new folder.

securityPolicyName: name of security policy to use.

parentFolderId: (optional) id of parent folder. If not provided, the folder is created at root.

folderType: type of folder-Offer/session/campaign/segment.

createdBy: (optional) object of type WSApplicationTypeEnum indicates which application created the folder. The possible values are Unica Campaign, Unica Plan, Collaborate, and Unica Deliver. If not provided, Unica Campaign is used.

Note: Plan=Unica Plan. Collaborate=Unica Collaborate.

creatorObjectId: (optional) used by Plan (Unica Plan) to link a folder in Unica Campaign with a folder in Unica Plan.

Returns

Returns object of type WSCreateFolderResponse.

Errors

Throws CampaignServicesException if the folder type is invalid or duplicate.

editFolder

Example

```
public WSEditFolderResponse editFolder(String userCredential,
String partitionName, Locale requestedLocale, long folderId,
String name,String description, WSFolderTypeEnum folderType,
```

```
Long creatorObjectId, boolean clearExisting)
 throws CampaignServicesException
```

Updates the specified folder.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

id: id of folder to update.

name: name of folder to create.

description: description for new folder.

folderType: type of folder-Offer/session/campaign/segment.

creatorObjectId: (optional) used by Plan (HCL Plan) to link a folder in Campaign with a folder in Plan.

clearExisting: not used now.

Returns

Returns object of type WSEditFolderResponse.

Errors

Throws CampaignServicesException if the folder type is invalid or duplicate or if folder id is invalid.

getSubFoldersList

Example

```
public WSGetSubFolderListResponse getSubFoldersList(String user_credential,
String partition_name, Locale locale, WSReference parentFolder,
WSFolderTypeEnum folderType)
 throws CampaignServicesException
```

Lists all sub folder in specified folder.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

parentFolder: reference of folder to which all sub folders are requested.

folderType: type of folder.

Returns

Returns object of type WSGetSubFolderListResponse.

Errors

Throws CampaignServicesException if folder reference is invalid.

moveFolders

Example

```
public WSMoveFolderResponse moveFolders(String userCredential,
String partitionName, Locale requestedLocale, Long[] folderId,
long parentFolder,long destinationFolder,
 WSFolderTypeEnum folderType)
 throws CampaignServicesException
```

Moves specified folder to other parent folder.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

folderId: folder id to delete.

parentFolder: id of parent folder.

destinationFolder: id of destination folder where the specified folder is moved.

folderType: type of folder-Offer/session/campaign/segment.

Returns

Returns object of type WSMoveFolderResponse.

Errors

Throws CampaignServicesException if the folder type or parent folder id is invalid or if folder id is invalid.

deleteFolders

Example

```
public WSDeleteFolderResponse deleteFolders(String userCredential,
 String partitionName, Locale requestedLocale, Long[] folderId,
 long parentFolder,boolean deleteChilds,
WSFolderTypeEnum folderType)
 throws CampaignServicesException
```

Deletes specified folders in system along with all items in folder.

Parameters

userCredential: the client user credential.

partitionName: optional name of the campaign partition to use.

requestedLocale: optional locale to use for this request.

folderId: folder id to delete.

parentFolder: id of parent folder.

deleteChilds: flag indicates whether all dependencies of the folder must be deleted. If false, no dependency is deleted.

folderType: type of folder-Offer/session/campaign/segment.

Returns

Returns object of type WSDeleteFolderResponse.

Errors

Throws CampaignServicesException if the folder type is invalid or if folder id is invalid.

Chapter 6. SOAP API common exceptions

The Unica Campaign SOAP API may throw the following common exceptions. All exception localized messages are in the requested locale if available to Unica Campaign. The usual Java™ locale fallback policies apply.

RemoteException

This item applies only to the SOAP interface.

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

Run-of-the-mill checked and unchecked API exceptions, such as DataException, are 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 Unica Campaign partition. Check the user role that is defined in Unica Platform.

Authorization Exception

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 permissions that are assigned to the user role in Unica Platform.

Data Exception

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

Check the Unica Campaign flowchart and listener logs for details.

Lock Exception

A temporary exception that is 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 and then retrying the operation. Retry logic is the responsibility of the client.

InvalidComponentException

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

InvalidAttributeException

An exception that is 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. The exception's `getAttributeName()` method returns the name of the problem attribute, `getAttributeValue()` returns the value, and `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).

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).

CompositeException

A CompositeException is used by some APIs to report multiple errors back to the caller. It typically has 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.

Note: Generally the API either completes successfully or rolls back its work before throwing a composite exception. See, for example, the bulk Target Cell Spreadsheet APIs described in SOAP API methods: Target cells on page 34.