Tripolis Dialogue

Web services

API 2.0

Table of Contents

Introduction

What is the SOAP Web Service API 2.0
The Tripolis Dialogue SOAP API facilitates remote access for managing Dialogue, or for it's integration, without having to access Dialogue's application interface (GUI)
SOAP (Simple Object Access Protocol) is a messaging protocol that allows programs to communicate using HTTP and XML. Tripolis uses the SOAP and WSDL standards which are platform and language independent.
For usage of the web service a license is required, licenses are activated by the Tripolis System Administrators.

Why Use the SOAP Web Service API 2.0
The Tripolis Dialogue SOAP API 2.0 is an extensive web service which facilitates communication between any application and Dialogue by making use XML files containing a header with authentication Info and a body with the actual call.
The API 2.0 web service gives your developers access to many of the data, content, and campaign features of Dialogue. By making use of our API 2.0 you can build your own bi-directional data transfers, campaign management applications, and other applications that need to interface with your Dialogue client from a remote location.

Prerequisites for Using the SOAP Web Service API

- To make use of the web service, you need a Standard API user for Dialogue with Advanced API user rights (these rights need to be set by Tripolis support).
- Allow your development environment to access services listed at:
  https://**<instance>**.tripolis.com/api2/docs/api **eg** https://**td35**.tripolis.com/api2/docs/api
- Make sure to connect to the right endpoint, we have categorized our services this means each category has it's own end-point. Eg our SubscriptionService:
  https:// **<instance>**.tripolis.com/api2/soap/SubscriptionService?wsdl

Example call

subscribeContact (part of our **SubscriptionService**): subscribes (un)confirmed, unsubscribes or updates contacts with optional email publication.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
   <soap:Header>
      <ser:authInfo>
         <client>Tripolis Demo</client>
         <username>wmeijer@tripolis.com</username>
         <password></password>
      </ser:authInfo>
      <ser:responseLanguage>?</ser:responseLanguage>
   </soap:Header>
   <soap:Body>
      <sub:subscribeContact>
         <!—Most xml items can be identitfied by either name or unique encrypted id-->
         <subscribeContactRequest>
            <contactDatabase>
               <id>[contactDatabaseId]/id>
               <name>[contactDatabaseName]/</name>
            </contactDatabase>
            <!— workspace transactional message -->
            <workspace>
               <id>[workspaceId]/</id>
               <name>[workspaceName]</name>
            </workspace>
            <!—Id or unique contact field in order to create or update contact-->
            <contactId>[contactId]</contactId>
            <contactFields>
               <contactField> <!—use id or name of contactField -->
                  <id>[contactFieldId]</id>
                  <name>[contactFieldName]</name>
                  <value>[value of contactField]</value>
               </contactField>
            </contactFields>
            <contactGroupSubscriptions>
               <!--1 or more repetitions:-->
               <contactGroupSubscription>
                  <contactGroup>
                     <id>[contactGroupId]</id>
                     <name>[contactGroupName]</name>
                  </contactGroup>
                  <!—1=confirmed / 0=unconfirmed-->
                  <confirmed>?</confirmed>
               </contactGroupSubscription>
            </contactGroupSubscriptions>
            <contactGroupUnSubscriptions>
               <!--1 or more repetitions:-->
               <contactGroup>
                  <id>[contactGroupId]</id>
                  <name>[contactGroupName]</name>
               </contactGroup>
            </contactGroupUnSubscriptions>
            <directEmail>
               <id>[directEmailId]</id>
               <name>[directEmailName]</name>
            </directEmail>
            <newsletter>
               <id>[newsletterId]</id>
                <name>[newsletterName]</name>
            </newsletter>
            <!— temporary variables for content of transactional message-->
            <jobProperties>
               <property>
```

```
                  <key>[key]</key>
                  <value>[value]</value>
               </property>
            </jobProperties>
            <ip>?</ip> <!—required-->
            <reference>?</reference>
         </subscribeContactRequest>
      </sub:subscribeContact>
   </soap:Body>
</soap:Envelope>
```

Response example:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>
      <ns2:responseLanguage xmlns:ns2="http://services.tripolis.com/">en</ns2:responseLanguage>
   </soap:Header>
   <soap:Body>
      <ns3:subscribeContactResponse xmlns:ns2="http://contact.services.tripolis.com/"
xmlns:ns3="http://subscription.services.tripolis.com/" xmlns:ns4="http://services.tripolis.com/">
         <response>
            <id>yKreYZGJx3_wK2T2K1CMqw</id>
         </response>
      </ns3:subscribeContactResponse>
   </soap:Body>
</soap:Envelope>
```

Response provides created or updated (encrypted) contact id.

Services overview

We have categorized our calls over the following topics:

Generic

- ClientDomainService; for maintaining Client Domains
- FtpAccountService; for administering S/FTP accounts
- UserService; API and users information

Contacts

- ContactDatabaseService, for maintaining Contact Databases
- ContactDatabaseFieldGroupService, for maintaining Field Groups
- ContactDatabaseFieldService, for maintain Fields
- ContactGroupService, for maintaining Contact Groups
- SmartGroupService, for maintaining Smart Groups
- ContactService, for maintaining Contacts
- ImportService, for importing Contacts
- SubscriptionService, for subscribing/updating Contacts

Content

- WorkspaceService, for maintaining Workspaces
- ArticleTypeService, for maintaining Article Types
- ArticleFieldService, for maintaining Article Fields
- ArticleService, for maintaining Articles
- AttachmentService (mtom), for maintaining Attachments
- ImageService (mtom), for maintaining Images
- DirectEmailTypeService, for maintaining Direct Email Types
- DirectEmailService, for maintaining Direct Emails (editions)
- NewsletterTypeService, for maintaining Newsletter Types
- NewsletterTemplateService, for maintaining Newsletter Templates
- NewsletterSectionService, for maintaining Newsletter Sections
- NewsletterService, for maintaining Newletters (editions)
- NewsletterArticleTemplateService, for maintain Article Templates
- SmsTypeService, for maintaining SMS Types
- SmsMessageService, for maintaining SMS Messages

Publishing

- PublishingService, for publishing Mail and SMS jobs
- MailJobService, for maintaining Mail Jobs
- SmsJobService, for maintaining SMS Jobs

Campaign

- CampaignDefinitionService, for retrieving Campaign Definitions
- CampaignService, for maintaining Campaigns Runs

Reporting / analysis

- ReportingService, for retrieving response data

MTOM vs NO-MTOM

Since PHP and .NET implementations are known to experience difficulties with handling MTOM attachments we offer our web service default as NO-MTOM, with the exception of **AttachmentService**, **ImageService** and **ExportService.** These calls are available in both MTOM and NO-MTOM.

MTOM is the recommended W3C *Message Transmission Optimization Mechanism*, a method of efficiently sending binary data to and from Web services. The efficiency claim of MTOM refers to the size of the messages sent across the wire. Since SOAP uses XML, any binary data in the SOAP message will have to be encoded as text. This is usually done using Base64 encoding which increases the size of the binary data by 33%. MTOM provides a way to send the binary data in its original binary form, avoiding any increase in size due to encoding it in text.

Authorities

MODULE_CAMPAIGN = Campaign
MODULE_CONTACT = Contact
MODULE_CONTENT = Content
MODULE_OUTBOUND = Outbound
MODULE_REPORT = Report
MODULE_SETUP = Setup
MODULE_SUPPORT = Support
ROLE_ADMIN = Clientdomain Administrator
ROLE_ADVANCED = Clientdomain Advanced User
ROLE_API_ADVANCED = Advanced API user
ROLE_API_BOUNCEHANDLER = Bouncehandler user
ROLE_API_STANDARD = Standard API user
ROLE_INTERACTIVE = Interactive user
ROLE_PARTNER_ADMIN = Partnerdomain Administrator
ROLE_SYSTEM_ADMIN = System Administrator
ROLE_SYSTEM_ATTACHMENT = Content Attachment Management
ROLE_SYSTEM_USER = System User
ROLE_USER = Clientdomain User
SUBMODULE_CONTENT_DIRECT_MAIL = Content Direct Mail
SUBMODULE_CONTENT_NEWSLETTER = Content Newsletter
SUBMODULE_CONTENT_SMS = Content SMS
SUBMODULE_OUTBOUND_MAIL = Outbound Mail
SUBMODULE_OUTBOUND_SMS = Outbound SMS
SUBMODULE_REPORT_CONTACTS = Report Contacts
SUBMODULE_REPORT_PUBLISHING = Report Publishing
SUBMODULE_SETUP_APPLICATION = Setup Application
SUBMODULE_SETUP_CONTACT = Setup Contact
SUBMODULE_SETUP_CONTENT = Setup Content
SUBMODULE_SETUP_PUBLISHING = Setup Publishing

AuthInfo

In every request the authInfo information is required for authentication and permission purposes.
For security reasons we advise to configure a separate API user, not related to a person, with restricted access to your client domain data, and without access to GUI.

```
<ser:authInfo>
   <client>CLIENT NAME</client>
   <username>USER NAME</username>
   <password>PASSWORD</password>
</ser:authInfo>
```

For example:

```
<ser:authInfo>
   <client>Tripolis API</client>
   <username>apiuser@tripolis.com</username>
   <password>Xxxxx</password>
</ser:authInfo>
```

Dialogue application instance

In this document you will find the placeholder **<instance>**.
Depending on the Dialogue application instance you are running on, you must fill in the right subdomain.
For instance, if you are running on the TD42 instance, you replace **<instance>** with "td42" which would result in a url similar to: https://td42.tripolis.com/api2/soap/ClientDomainService?wsdl

getServiceInfo

All services have a method *getServiceInfo*. With this method you can check if the service is available.

Request example:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:ser="http://services.tripolis.com/">
   <soap:Header/>
   <soap:Body>
      <ser:getServiceInfo/>
   </soap:Body>
</soap:Envelope>
```

Response example:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
   <soap:Header>
      <ns2:responseLanguage xmlns:ns2="http://services.tripolis.com/">en</ns2:responseLanguage>
   </soap:Header>
   <soap:Body>
      <ns2:getServiceInfoResponse xmlns:ns2="http://services.tripolis.com/">
         <response>
            <message>The ClientDomainService is alive</message>
            <serviceInfoItems>
               <serviceInfoItem>
                  <key>buildNumber</key>
                  <value>2011-11-30-15:07:58</value>
               </serviceInfoItem>
               <serviceInfoItem>
                  <key>apiVersion</key>
                  <value>2.0</value>
               </serviceInfoItem>
               <serviceInfoItem>
                  <key>dialogueVersion</key>
                  <value>3.5.2</value>
               </serviceInfoItem>
            </serviceInfoItems>
         </response>
      </ns2:getServiceInfoResponse>
   </soap:Body>
</soap:Envelope>
```

Response codes

code	name	type	category
100	CALL_NOT_ALLOWED	FAULT	
200	SUCCESS	SUCCESS	
300	AUTHENTICATION	FAULT	AUTHENTICATION
400	VALIDATION	FAULT	VALIDATION
401	ALREADY_EXISTS	FAULT	VALIDATION
402	REQUIRED	FAULT	VALIDATION
403	INVALID	FAULT	VALIDATION
404	VALUE_TOO_LOW	FAULT	VALIDATION
405	VALUE_TOO_HIGH	FAULT	VALIDATION
406	ONLY_ONE_ALLOWED	FAULT	VALIDATION
407	NOT_FOUND	FAULT	VALIDATION
408	NOT_ALLOWED	FAULT	VALIDATION
409	VALUE_TOO_SHORT	FAULT	VALIDATION
410	VALUE_TOO_LONG	FAULT	VALIDATION
411	ONE_IS_REQUIRED	FAULT	VALIDATION
412	ATTACHMENT_NOT_POSSIBLE	FAULT	VALIDATION
413	ATTACHMENT_TOO_LARGE	FAULT	VALIDATION
414	NOT_EMPTY	FAULT	VALIDATION
401	ALREADY_ASSIGNED	FAULT	VALIDATION
500	APPLICATION	FAULT	APPLICATION
501	NOT_POSSIBLE	FAULT	APPLICATION
900	UNKNOWN	FAULT	

406: This ResponseCode is used when of two or more fields, only one is allowed. You should always check for more than one of these error codes in the response.

411: This ResponseCode is used when of two or more fields, one is required. You should always check for more than one of these error codes in the response.