Tribute Store API Documentation 1.1

Provided by: Tribute Technology and the Tribute Store

Updated March 25, 2019
(clarity update)

Tribute Store

Table of Contents

Introduction

The Tribute Store API is a REST-style json API provided for users who plan to integrate with the Tribute Technologies Tribute Store. Details are provided below on how to get an Authorization token, how to set up funeral home rooftops, and how to push obituaries and photos through the Tribute Store API so as to automatically set up a personalized Tribute Store page for each deceased.

API URL

Development

http://api.demo.tributecenteronline.com/

Production

http://api.tributecenteronline.com/

Authorization

There is a multi-level authorization scheme in place. You will be provided with a {Provider} credential string.

Each {Provider} also has an IP Whitelist – make sure that the IP of your development server has been added to the IP whitelist.

For each Funeral Home you will be provided a {HostName, UserName, Password} triple. That triple can be used to hit the token/ endpoint to be assigned an html bearer token. That token can then be used to hit other endpoints to push or retrieve data.

Note that currently the funeral home {HostName, UserName, Password} triples are assigned either manually or in bulk, but that an automatic method of generating live funeral home credential triples is in development.

In the meantime, you will be provided with a {Provider} and a single {HostName, UserName, Password} triple for development — within that single 'Funeral Home' entity you can create multiple 'Serving Locations' — e.g. rooftops.

Token

Authorization is by html bearer tokens.

Hitting any endpoint/method combination other than a POST to the token/ endpoint will require a bearer token generated by a correctly formatted POST to the token/ endpoint.

```
Method: POST
Endpoint: token/
Headers: Content-Type: application/x-www-form-urlencoded
f-hostname: {HostName}
f-provider: {Provider}
Body:
grant_type=password&username={UserName}&password={Password
}&scope=external-api

Response:Json

Failure:  {“error”:”unsupported_grant_type”}

Success: {
  “access_token”: “TOKEN”,
  “token_type”: “bearer”,
  “expires_in”: SECONDS
}
```

The endpoint checks that the request provides a valid {HostName} in the header that was set up for your {Provider}, and ensures that the {UserName,Password} pair in the body are valid and match the provided {HostName}.

If all that checks out, then a TOKEN is provided, which is valid for SECONDS seconds. Note that the TOKEN *is* your identification to other API endpoints — it automatically specifies the associated {HostName} when hitting other endpoints, and only entities created for/with that {HostName} are valid/available.

POST External-Location

Before you can post any obituaries, you have to post serving locations. You can think of the {HostName} as the Funeral Home corporate entity, whereas each 'Serving Location' is a rooftop. A Funeral Home may have a single or multiple rooftops.

There are other types of 'Serving Locations', but for the purpose of pushing obituaries through to the Tribute Store, only Funeral Home rooftops need to be explicitly pushed through.

Authorization is with the TOKEN received earlier.

```
Method:         POST
Endpoint:       api/external-location/post
Headers:        Content-Type: application/json
                Authorization: Bearer TOKEN

{
    "Id": string,
    "LocationType": "Funeral Home",
    "Name": string,
    "Address1": string,
    "Address2": string,
    "City": string,
    "State": string,
    "Country": string,
    "Phone": string,
    "PostalCode": string,
    "WebSite": string,
    "Email": string,
    "Fax": string,
    "CouldServeObituary": true
}

Response:

Failure (json): {"Message":MESSAGE}
Success(integer) : SERVING_LOCATION_ID
```

Parameter	Description
Id	Unique identifier of this location from your application
LocationType	enum ('Cemetery','Church','Funeral Home','Memorial','Other')
Name	Funeral Home Company/Branch Name
Address1	
Address2	
City	
State	Should be 2 letter Abbreviation
Country	US or CA
Phone	
Postal Code	US Zip or Canadian Postal Code
Website	
Email	
Fax	
CouldServeObituary	Whether obituaries will be attached

A couple of the above fields are simplified – you only need to add Funeral Homes, and you are presumably adding one so you can attach obituaries, so LocationType should always be "Funeral Home" and CouldServeObituary should always be true.

Id should be the unique identifier in your application. If this Id has been stored before, then the POST is treated as an UPDATE. If this is a new Id, then the POST is treated as an ADD. In either case, the response is the SERVING_LOCATION_ID in the Tribute Store if the transaction was successful, and a hopefully informative MESSAGE if there was an error.

GET Location

This is primarily to verify the current information stored about a Serving Location and/or to verify that all the information you POSTed was successfully received.

```
Method:      GET
Endpoint:    api/locations/{SERVING_LOCATION_ID}
Headers:
     Content-Type: application/x-www-form-urlencoded
     Authorization: Bearer TOKEN

Response:

{
 “Id”: 1,
 “LocationTypeId”: 2,
 “Name”: “sample string 3”,
 “AddressId”: 1,
 “WebSite”: “sample string 4”,
 “Email”: “sample string 5”,
 “ShowOnWebSite”: true,
 “Address1”: “sample string 7”,
 “Address2”: “sample string 8”,
 “City”: “sample string 9”,
 “State”: “sample string 10”,
 “PostalCode”: “sample string 11”,
 “Country”: “sample string 12”
}
```

This will get back a json-data structure representing the information stored about a location that has been POSTed. Note that ‘ShowOnWebSite’ is equivalent to CouldServeObituary from the POST, and that the LocationTypeId will be 3 for Funeral Home (although see api/locations/getlocationtypes below for the complete list).

If you want to get back *all* the locations at once (not recommended for the demo user):

```
Method:     GET
Endpoint:   api/locations/
Headers:
     Content-Type: application/x-www-form-urlencoded
     Authorization: Bearer TOKEN
```

This will get back a json array of all the locations.

For completeness there is a GET api/locations/getlocationtypes – retrieve location types (predefined values), which will return:

```
[
 {“Id”: 1,”Name”: “Cemetery”  },
 {“Id”: 2,”Name”: “Church” },
 {“Id”: 3,”Name”: “Funeral Home” },
 {“Id”: 4,”Name”: “Memorial” }
]
```

POST External-Case

Once the serving location required for your obituary has been pushed, you can start passing through obituaries.

You will need your provider name credential directly in the body of this post, as well as the TOKEN from the token endpoint.

```
Method:     POST
Endpoint:   api/external-case/post
Headers:    Content-Type: application/json
            Authorization: Bearer TOKEN
{
  “CaseId”: “5f54ae57-bbbe-4519-866b-cb1bb7cc8b4b”,
  “ProviderName”: {Provider},
  “FirstName”: “John”,
  “LastName”: “Doe”,
  “BirthDate”: “1926-09-01T00:00:00”,
  “DeathDate”: “2015-09-01T00:00:00”,
  “Services”:  [{EVENT_OBJECT}] ,
  “Obituary”: “Test Test”,
  “Photo”:””,
  “IsPhotoChanged”:true,
  “IsPublished”: true,
}

Response:

Failure (json): {“Message”:MESSAGE}
Success(integer) : OBITUARY_ID
```

Parameter	Is Required	Description
CaseId	yes	Your unique identifier
ProviderName	no	
FirstName	yes	string
LastName	yes	string
BirthDate	no	datetime

Parameter	Is Required	Description
DeathDate	yes	datetime
Services	no	array of event objects (e.g. visitation, service)
Obituary	no	string
Photo	no	(base64 string)
IsPhotoChanged	no	Boolean (default value false)
IsPublished	no	Boolean

CaseId is your unique Id for the obituary, and will be used as an ADD/UPDATE Key, much as the Id field in the Serving Location. A POST using a CaseId already used will update the previously posted information.

Services takes a set of event names/times/locations and will be outlined below.

Obituary is the text obituary of the deceased, and IsPublished provides a way to specify whether this obituary is actively viewable or not – this is probably 'true' if you are pushing through to the Tribute Store. Photo is a base64 encoded string representation of a 60x70 pixel thumbnail photo of the deceased, and IsPhotoChanged should be true if you are pushing through a new photo.

Each Service Looks like the following:

```
{

  "Type": "Visitation",
  "StartTime": "2016-09-04T08:00:00",
  "EndTime":"2016-09-04T09:00:00",
  "Location":
   {
     "Name": "United Methodist Church",
     "Address1": "587 Arapaho -6Th. Arapaho",
     "Address2": "",
     "City": "Eakly",
     "State": "OK",
     "PostalCode":"12345",
     "Phone": "123456789",
   }
}
```

Parameter	Is Required	Description
Type	yes	string
StartTime	yes	datetime
EndTime	no	datetime
Location	yes	location object
Location Object		
Name	yes	string
Address1	yes	string
Address2	no	string
City	yes	string
State	yes	string
PostalCode	yes	string
Phone	yes	string

You can provide your own event Type, and the Start / EndTime and Location information all follow and accept standard formats.

GET Obituaries

To check the details on an obituary that you have posted, you can hit the following endpoint:

```
Method:     GET
Endpoint:   api/obituaries/GetObituary/{OBITUARY_ID}
Headers:
      Content-Type: application/x-www-form-urlencoded
      Authorization: Bearer TOKEN

Response:  fully specified json obituary object
```

Note that while you can ADD and UPDATE Obituaries with your own reference number (CaseId), to retrieve the details from the Tribute Store you need to use the Tribute Store OBITUARY_ID that you receive as the result of a successful POST api/external-case/post.

We've omitted above the details of the fully specified json object, but you should be able to verify all the data that you passed through the API.

To review multiple obituaries, you can review:

```
Method:     GET
Endpoint:   api/obituaries
Headers:
      Content-Type: application/x-www-form-urlencoded
      Authorization: Bearer TOKEN

Response:  json array of Obit Summary objects
[{
  "Id": 1,
  "ImageUrl": "sample string 2",
  "Name": "sample string 4",
  "FirstName": "sample string 5",
  "MiddleName": "sample string 6",
  "LastName": "sample string 7",
  "BirthDate": "2016-02-11T13:55:01.9377822-06:00",
  "DeathDate": "2016-02-11T13:55:01.9377822-06:00"
}]
```

This summary list has name, dates, thumbnail URL and the OBITUARY_ID for each obit.

This endpoint also supports standard OData Query Options, so for example if you wanted the 21st through 30th obituaries ordered by DeathDate, you could hit:

```
GET api/obituaries?$orderby=DeathDate
desc&$skip=20&$top=10
```

Obituary Store URL

You will be given a base URL

e.g.

http://demo.tributecenterstore.com

To the base Store URL for your site, you just add a query string ?oId=OBITUARY_ID (Protip: that is “oh-eye-dee”, not the word “old”). e.g. you will end up with a URL something like

http://demo.tributecenterstore.com?oId=1064677

Usage Expectations

Please feel free to push an update to a serving location or an obituary when the information changes. However, please refrain from pushing obituary or serving location updates when the information is *not* changing (e.g. do not push through the API on every save on your end if that save doesn’t change the information in the API contract).


TRIBUTE
TECHNOLOGY