DonorPerfect Online XML API Documentation

User Manual

SofterWare, Inc.

Table of Contents

1.   Executive Summary

The DonorPerfect Online (DPO) Application Programming Interface (API) makes it possible for you to integrate DonorPerfect Online with your existing web environment. You can, for example, access donor and gift information on secure (login protected) portions of your intranet.

The API proves secure connections to the DPO database, allowing read and write access in real time.

Common uses of the API allow developers to create content rich applications for web sites to display donor and gift information. For example, a religious organization may choose to display a list of church contact records from the DonorPerfect Online database on their website. Then, when the contact records are updated in the DonorPerfect Online database, the website will automatically display the updated records. Extending the same scenario, the API could also be configured to allow users to update the contact records from their website.

If you are not sure whether your DonorPerfect configuration includes the API, contact your DonorPerfect account representative!

Intended Audience

This document is written for customers who currently have an existing DonorPerfect Online license, or for those developers that wish to integrate their third party applications with the overall DPO CRM solution.

Version Control

This document has been updated to Version 5.16 to correspond to DonorPerfect Online Version 2020.09.   This update includes the following changes:

- Updated the section title of dp_savemultivalues_xml to dp_savemultivalue_xml.

- Added additional note to mergemultivales API call showing example of how to retrieve list of set flags for a specified flag and donor

- Updates to dp_savepledge notes

2.   API Overview

What is the DPO API?

The DonorPerfect Online (DPO) Application Programming Interface (API) is a set of calls and parameters that one can use to interface external applications with DPO. The API uses XML to return data back to the client, and these XML data 'streams' can be easily parsed using many popular XML parsers. Simply put, the DPO API is a way for programmers and webmasters to make their existing websites or any other applications synchronize with DPO on a variety of levels.

Common API Uses

The DPO API includes the most commonly requested functions such as:

- Saving Donor/Constituent Information – includes demographic Name, Address, Email, Phone Numbers, etc.

- Saving a Donor/Constituent's extended information (User Defined Fields) – Unlimited ability to save to any field in the DPO system at the donor/constituent level- including ones that are created by the client.

- Retrieving a Donor/Constituent's record from the database – retrieves all demographic information and User Defined Fields.

- Searching for a Donor/Constituent, and returning the Donor_ID.

- Saving a Gift – saves all aspects of a gift or pledge transaction, including Date, Start Date, Amount, Gift Type, General Ledger, Solicitation, Thank You Letter Type, Memo/Comments, etc.

- Saving a Gift's extended information (User Defined Fields) – unlimited ability to save any field at the Gift/Pledge level.

- Listing Gifts for a selected Donor/Constituent – retrieves Gift or Pledge Transactions for the selected Donor.

The list of these functions is constantly expanding as clients request more commonly used functions exposed to the API. It is also possible to have a custom call designed for your organization, and it has to be arranged through your Account Manager.

Before You Begin

It is important to understand that the DonorPerfect Online (DPO) Application Programming Interface (API) comprises a set of API calls that can be used within application programming code written by, or for, your organization. Your organization is responsible for the development, testing, implementation and maintenance of your API based application(s) and you should be aware that there are certain limitations involved with using the API:

- There is a row retrieval limit of 500 rows of data. See following section on Row Retrieval Limit for more info.

- Predefined Procedures should be used where possible rather than Dynamic API calls

- Limited SQL functionality is supported in Dynamic API calls and no SQL functionality beyond that which is represented in this API User Manual is supported.

- SmartActions are not supported via the API – i.e.; changes to the DonorPerfect data made via the API will <u>not</u> trigger SmartActions.

- Acknowledgement response time for support calls is not guaranteed and is independent of any other support guidelines. Calls are answered in the order they are received and under normal circumstances, it's reasonable to expect a response within one business day. For emergencies, send an additional email to support@donorperfect.com. DonorPerfect support services extend to your use of API calls but do not include supporting your API based application.

- SofterWare makes every effort to maintain the constant availability of the API connection but is unable to guarantee a specific service level guarantee due to factors such as network congestion and transient API traffic levels.

- While we avoid doing so when we can, SofterWare reserves the right to change API interface as needed to service the greater needs of our clients.

Row Retrieval Limit

There is a 500 row retrieval limit for data retrieved via API calls from DonorPerfect.

To avoid encountering this limit, modify your queries to pull subsets of the desired data where applicable.

Example1: Retrieval of DPCODES Solicitation Codes

```
...&action=select top 500 code, description from dpcodes where
field_name='SOLICIT_CODE' order by code

...&action=select top 500 code, description from dpcodes where
field_name='SOLICIT_CODE' and code > '{last code}' order by code

etc… (i.e.; repeats of 2nd API call until all values have been retrieved)
```

- where {last code} is last CODE value retrieved in the previous query

Example 2: Retrieval of Recent Gift Fields

```
...&action=select top 500 gift_id, amount, created_date, modified_date,
gl_code from dpgift WHERE COALESCE(DPGIFT.MODIFIED_DATE,
DPGIFT.CREATED_DATE) >='{date}' order by gift_id

...&action=select top 500 gift_id, amount, created_date, modified_date,
gl_code from dpgift WHERE COALESCE(DPGIFT.MODIFIED_DATE,
DPGIFT.CREATED_DATE) >='{date'} and gift_id > {last gift_id} order by
gift_id

etc...
```

- where {date} is the last time you ran this - e.g..; yesterday's date and {last gift_id} is the highest gift_id number retrieved in the previous query

- the COALESCE function is an SQL statement returns the first value that does not evaluate to NULL. So if this is a new item with no modified_date, it returns the created_date. If, on the other hand, it is an item that has been updated, there will be a modified_date value and it will return that.

How do I use the DPO API?

The DPO API can be accessed via HTTPS (Hyper Text Transfer Protocol Secure) requests made to a specific page located on the secure DPO web server. Anyone who has familiarity with XML and has possibly used an outside credit card processor (which requires an HTTPS request) should be comfortable with using the DPO API. XML is a structured data set that is sent from our DPO system to your application (website, etc.) that is then translated to a meaningful form whether for the web user to view or to be stored by your application (database, file, email, etc.). Some calls are used exclusively to save information in DPO such as Main, Gift, Pledge or Contact screen information.

API Example Call Breakdown

A sample HTTPS request is analyzed below. This one returns all the gifts that are in the DPO system for a donor with id=1269525. This example assumes a username of 'xxx' and a password of 'yyy':

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_gifts&params=1
269525&login=xxx&pass=yyy

Or, if you prefer, you can place the login credentials before the action:

https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxx&act
ion=dp_gifts&params=1269525
```

Each XML API call has a number of key components:

1. ***https://www.donorperfect.net/prod/xmlrequest.asp*** - a constant location of the page that accepts the parameters and returns the requested XML information.

2. ***action=dp_gifts*** – action (function) that one would like to perform.

3. ***params=1269525*** – parameters listing. This particular example only has one parameter but in most cases there will be more than one for other functions (especially that deal with saving information into DPO). These may include numeric, date and text parameters. Enclose any date or text parameters in single quotes as shown.

   There are two ways to input parameters:

   Unnamed parameters can be listed in order and separated by a comma (,).

   ```
   …&params=1, 'Apple', '01/15/2018
   ```

   The preferred method is to use **named parameters**.

   ```
   …&params=@this=1, @that='Apple', @somedate='01/15/2018'
   ```

   - When you use named parameters, the order in which they are listed is flexible

   - Using named parameters does not allow you to skip required parameters

   - You cannot mix named parameters and unnamed parameters within a single API call

   - Additional information on using named parameters in the Using Named Parameters section of this document.

   Non numeric parameters must be enclosed in single quotes

4. ***login=xxx*** – where "xxx" is a User ID (username) created in the DPO User Security screen for communication with DPO via XML API

5. ***pass=yyy*** – where "yyy" " is a password created in the DPO User Security screen for communication with DPO via XML API

   NOTE: See following section on API Connection Credentials for information on the alternative **API Key** connection method.

API Connection Credentials

There are two methods for supplying connection credentials to DonorPerfect Online (DPO). Either connection method will work, but the new API Key method has the benefit

of enhanced security through the use of long (over 100 characters) and very strong key values:

1. Using a username and corresponding password value from a user created within the DPO user interface (Settings > User Security)
2. Using an API Key value provided by DonorPerfect. This is the recommended connection method.

API Key (apikey=)

This is the recommended connection method and has the benefit of providing enhanced security through the use of very strong passwords of over one hundred characters in length.

To get an API Key value for your DonorPerfect site, contact the API Help Desk (mwarren@softerware.com) or request one through your regular DonorPerfect support channel. If you are an API integrator acting on behalf of a DonorPerfect Online (DPO) client, please have your DPO client request the API Key.

Username/Password (&login=)

This is the traditional method for supplying connection credentials. A sample HTTPS request is analyzed below. This particular one returns all the gifts that are in the DPO system for a donor with id=1269525:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_gifts&params=1
269525&login=xxx&pass=yyy
```

It is also worth noting that when/if you connect using a login username and password, these connections are subject to the Invalid Login Attempt Count in the same way that a non-API account would and so it may become necessary to unlock the account through the DPO User Security screen. See the DPO Knowledge Base topic on: **Using the "Forgot Password?" Link**

Sample API Call

A sample HTTPS request is analyzed below. This particular one returns all the gifts that are in the DPO system for a donor with id=1269525. The actual key values are much longer than the partial key shown here but does not exceed 200 characters:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=
95LTf7cfVoBJnHqILHv1jN2ZYbdBR5U…… &action=dp_gifts&params=1269525
```

NOTE:

Turns out there is an important difference between these two:

- `https://www.donorperfect.net/prod/xmlrequest.asp?apikey=`

and this:

- `https://www.donorperfect.net/prod/xmlrequest.asp? apikey=`

If you have a space after the question mark and your API call isn't set up exactly right, you may miss out on important error information.

For example, with the extra space, a dp_tribAnon_Summary API call that was missing the @user_id parameter returns this:

- `<return />`

But once you remove the extra space, the API call (still with missing @user_id) returns this:

`<result><field name="success" reason="user not authorized for this api call." id="success" value="false"/><error>[Microsoft][ODBC SQL Server Driver][SQL Server]`<mark>`Procedure or function 'dp_tribAnon_MyTribSummary' expects parameter '@userId', which was not supplied.`</mark>`</error></result>`

```
- <result>
  - <record>
      <field name="gift_date2" id="gift_date2" value="09/30/2005" />
      <field name="amount" id="amount" value="5000" />
      <field name="total" id="total" value="0" />
      <field name="sub_solicit_code" id="sub_solicit_code" value="" />
      <field name="campaign" id="campaign" value="KA0905" />
      <field name="balance" id="balance" value="0" />
      <field name="gl" id="gl" value="" />
      <field name="solicit_code" id="solicit_code" value="0223AF" />
      <field name="reference" id="reference" value="56724" />
      <field name="record_type" id="record_type" value="G" />
      <field name="gift_id" id="gift_id" value="384441" />
      <field name="donor_id" id="donor_id" value="1269525" />
    </record>
  - <record>
      <field name="gift_date2" id="gift_date2" value="09/28/2005" />
      <field name="amount" id="amount" value="5000" />
      <field name="total" id="total" value="0" />
      <field name="sub_solicit_code" id="sub_solicit_code" value="" />
      <field name="campaign" id="campaign" value="KA0905" />
      <field name="balance" id="balance" value="0" />
      <field name="gl" id="gl" value="" />
      <field name="solicit_code" id="solicit_code" value="0223AF" />
      <field name="reference" id="reference" value="56724" />
      <field name="record_type" id="record_type" value="G" />
      <field name="gift_id" id="gift_id" value="384382" />
      <field name="donor_id" id="donor_id" value="1269525" />
    </record>
  <result>
```

Using Named Parameters

Most of the examples in this document specify parameter values based on their position. For example, the example shown for the dp_savedonor API call specifies parameters in this way. The parameter names are not specified but are listed in order and are separated by commas:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_savedonor&para
ms=0,'Orianthi', 'Panagaris', null, null, null, null, null, null, '4240
Main St.', null, 'North Woodstock', 'NH', '12345', 'US', null, '205-555-
1212', null, null, '205-987-6543', 'orianthi@bigstar.com', null,
null,'N', null, null,'API User' &login=xxx &pass=yyy
```

It is also possible, however, to enter parameters using the parameter names <u>instead</u> of entering them by position. When you use named parameters in an API call, you do not need to follow a specific order as long as you enter all the fields that DonorPerfect needs to process the command.

dp_savedonor example:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxxxxx&
action=dp_savedonor&params=@donor_id=0 ,@first_name='Orianthi'
,@last_name='Panganaris' ,@middle_name=null ,@suffix=null ,@title=null
,@salutation=null ,@prof_title=null ,@opt_line=null ,@address='4240 Main
St.' ,@address2=null ,@city='North Woodstock' ,@state='NH' ,@zip='12345'
,@country='US' ,@address_type=null ,@home_phone='205-555-1212'
,@business_phone=null ,@fax_phone=null ,@mobile_phone='205-987-6543'
,@email='orianthi@bigstar.com' ,@org_rec='N' ,@donor_type='IN'
,@nomail='N' ,@nomail_reason=null ,@narrative='Narrative field contents
here' ,@user_id='My AppName'
```

dp_savegift example:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxxxxx&
action= dp_savegift&params=@gift_id=0,@donor_id=16, @record_type='G',
@gift_date='2017/05/02',@amount=23.45,@gl_code='GEN',
@solicit_code='BQ10', @sub_solicit_code='TS', @gift_type='VISAIC',
@split_gift='N',@pledge_payment='N',@reference='1234567890123',@memory_ho
nor=null,
@gfname=null,@glname=null,@fmv=0,@batch_no=0,@gift_narrative='Gift
narrative field
value.',@ty_letter_no='TY',@glink=null,@plink=null,@nocalc='N',@receipt='
N',@old_amount=null,@user_id='My AppName'
```

API Administration and Limitations

Logging

DonorPerfect Online logs all API requests for record keeping and administration purposes. These logs are kept to help troubleshoot common problems as well as to identify fraudulent or suspicious activity.

Restrictions

In addition, the calls are limited to return a maximum amount of data so that overall application performance will not be affected. Since some calls allow multiple transactions to be returned, these calls are intentionally limited. The default limit is set to 500 rows, though DonorPerfect may increase this on an individual client basis depending on need and scheduling concerns. Finally, there is no limit to the number of consecutive calls to the server.

It is also important to note that SmartAction triggers are not activated by any API actions (e.g.; donor/gift additions or updates).

No Blocking

The API is designed such that the API user is exempted from the DPO restriction on maximum number of concurrent users, so API calls are never blocked regardless of the number of concurrent users connected to the DPO system.

Change Notification

Any changes to the DonorPerfect API will be noted in the DonorPerfect Community Release Notes:

> https://softerware.force.com/dpcommunity/s/global-search/%22Release%20Notes%22

Additional details on these changes, as appropriate, will be documented in this API User Manual. The Version Control section of this document lists any changes since the previous release of the document.

3. Dynamic Queries

SELECT Statements

This query format is based on the ANSI[1] SQL implementation of Structured Query Language (SQL). SQL is the universal language used to communicate with databases.

The dynamic query feature of the DPO API allows users to build API calls using SQL syntax.

Example:

DPO contains a table called DP which contains donor information including people's first and last names and their donor ID numbers.   The column names are DONOR_ID, FIRST_NAME and LAST_NAME. The values are capitalized for clarity, but the query does not expect capitalization.

An SQL query to retrieve DONOR_ID, FIRST_NAME and LAST_NAME from the DP table for people with the last name 'Bacon'would look like this:

```
Select donor_id, first_name, last_name from dp where last_name = 'bacon'

Which, by the way, is identical to sending this API call:

SELECT DONOR_ID, FIRST_NAME, LAST_NAME FROM DP WHERE LAST_NAME = 'BACON'
```

This SQL gets placed into an XML API call and follows the `&action` = parameter like this:

```
https://www.donorperfect.net/prod/xmlrequest.asp?login=xxx
&pass=yyy&action= Select donor_id, first_name, last_name from dp where
last_name = 'bacon'
```

Please note that you can include multiple tables in queries including the use of JOIN statements. For example:

```
Select d.donor_id, d.first_name, d.last_name, du.mcat from dp d, dpudf du
where d.donor_id = du.donor_id and last_name = 'bacon'
```

---

[1] ANSI is the American National Standards Institute. MSSQL uses ANSI SQL.

If there is a donor in your system named Kevin Bacon and the above command was submitted through a web browser with the appropriate username and password, the raw (unprocessed) XML result will look like this:

```
- <result>
  - <record>
      <field name="donor_id" id="donor_id" value="68" />
      <field name="first_name" id="first_name"
        value="Kevin" />
      <field name="last_name" id="last_name"
        value="Bacon" />
    </record>
  </result>
```

To retrieve all fields from the DP table, you would use the asterisk wildcard (*) instead of field names.

NOTE: This can be a good way to identify all the field names in a table **BUT** make sure you limit the result set by specifying a particular donor_id or gift_id as appropriate. Otherwise your system will be burdened with a query retrieving thousands of data values. The query to retrieve all DP table field data for Kevin (donor_id 68) would look like this:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action= Select * from dp
where donor_id = 68 &login=xxx &pass=yyy
```

While retrieving all fields for a particular donor or gift with a SELECT * query can be useful in a test command to view all fields for a specified table, it is not good practice to retrieve all fields in your finished application. Doing so would slow down your application retrieval. Instead, just retrieve the fields you actually need by listing them by name:

https://www.donorperfect.net/prod/xmlrequest.asp?login=xxxxx&pass=yyyyy&action=SELECT DONOR_ID, FIRST_NAME, LAST_NAME, EMAIL, ADDRESS, CITY, STATE, ZIP FROM DP WHERE LAST_NAME='Dotsaur' AND FIRST_NAME LIKE 'J%25'

- Note: The %25 is a multicharacter wildcard which is the URL encoded equivalent to the percent (%) sign

DPO Table List

This is a list of tables in DPO you can run Dynamic Queries On

Table Name	Comments
DP	Contains donor information
DPUDF	User Defined Fields associated with the DP table
DPCODES	This table contains code values shown in drop-down lists – e.g;. General Ledger code values, Solicitation codes, Campaign codes, etc.

DPUSERMULTIVALUES	Checkbox field values
DPGIFT	Contains gift information
DPGIFTUDF	User Defined Fields associated with the DPGIFT table
DPADDRESS	Allows storages of additional addresses for donors (in DPO Address tab)
DPADDRESSUDF	User Defined Fields associated with the DPADDRESS table
DPLINK	Associated with values seen in the DPO Link tab
DPOTHERINFO	Associated with values seen in the DPO Other Info tab
DPOTHERINFOUDF	User Defined fields added to the DPO Other Info tab
DPTRIBUTESANON	This table contains the list of tributes including tribute name, active status, type, and notification person name
DPTRIBUTESANON_GIFTASSOC	This table creates the link between the gift and the tribute by containing the GIFT_ID and the TRIBUTE_ID.
DPPAYMENTMETHOD	Contains payment information associated with EFT Transactions for systems where this feature has been enabled. Two examples of the data returned from this table are shown in the section on the dp_paymentMethodInsert command.

4.   Predefined Procedures

You may notice that many of these API calls have a USER_ID field**:**

- **We recommend** that you use a name here, such as the name of your API application, for auditing purposes.

- The @user_id value does not need to match the name of an actual DPO user account.

- This can be very useful if you are trying to track whether a particular entry in DPO was created by your application or by another user.

dp_donorsearch

Introduction

Searching for a Donor—used to search for the donor based on several search criteria (similar to the search functionality offered in DPO). You can use "%" for wildcards.

Parameters:

Parameter	Type	Notes
@donor_id	numeric	Enter as NULL to search all donors
@last_name	Nvarchar(75)	If you aren't searching for a specific field value or the field is empty, use NULL instead of ''.
@first_name	Nvarchar(50)	As above
@opt_line	Nvarchar(100)	As above
@address	Nvarchar(100)	As above
@city	Nvarchar(50)	As above
@state	Nvarchar(30)	As above
@zip	Nvarchar(20)	As above
@country	Nvarchar(30)	As above
@filter_id	numeric	Always enter this parameter as NULL
@user_id	Nvarchar(20)	If you aren't searching for a specific field value or the field is empty, use NULL instead of ''.

Sample Call:

```
Esample 1:
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_donorsearch&pa
rams=null,'Pa%','Ori%',null, null, null, null, null, null, null, null

Example 2:
https://www.donorperfect.net/prod/xmlrequest.asp?&action=dp_donorsearch&p
arams=@donor_id=null,@last_name='Panagaris',@first_name='O%',@opt_line=nu
ll,@address='4240 PRS Ave.',@city='North
Woodstock',@state='NH',@zip='12345',@country=null,@filter_id=null,@user_i
d=null
```

Returns:

```
<result>
-<record>
<field name="donor_id" id="donor_id" value="147"/>
<field name="first_name" id="first_name" value="Orianthi"/>
<field name="last_name" id="last_name" value="Panagaris"/>
<field name="title" id="title" value=""/>
<field name="suffix" id="suffix" value=""/>
<field name="address" id="address" value="4240 Main St."/>
<field name="opt_line" id="opt_line" value=""/>
<field name="city" id="city" value="North Woodstock"/>
<field name="state" id="state" value="NH"/>
<field name="zip" id="zip" value="12345"/>
<field name="gifts" id="gifts" value="0"/>
<field name="gift_total" id="gift_total" value="0"/>
<field name="address2" id="address2" value=""/>
<field name="donor_name" id="donor_name" value="Orianthi Panagaris"/>
<field name="city_state_zip" id="city_state_zip" value="North Woodstock,
NH 12345"/>
</record>
</result>
```

Notes:

- If you require different fields to be returned, or if you need the city_state_zip or donor_name fields to appear as separate fields then use a Dynamic Query SELECT statement instead to achieve the desired result set.

- Alphanumeric fields – e.g.; Address, Zip, City – must be wrapped in single quotes as shown in the examples above

- When/if a field is empty, it is set to NULL rather than '' which is why a search on '' returns no values.

- Searching for NULL is the same as searching for '%'

dp_savedonor

Introduction

Saving a New/Existing Donor—used to save changes to the existing donor/constituent or save the new donor/constituent into the DPO system.

Parameters:

Parameter	Type	Notes
@donor_id	numeric()	Enter 0 (zero) to create a new donor/constituent record or an existing donor_id. Please note: If you are updating an existing donor, all existing values for the fields specified below will be overwritten by the values you send with this API call.
@first_name	varchar(50)	
@last_name	varchar(75)	
@middle_name	varchar(50)	
@suffix	varchar(50)	
@title	varchar(50)	
@salutation	varchar(130)	
@prof_title	varchar(100)	
@opt_line	varchar(100)	Enter as NULL if other field value not required.
@address	varchar(100)	
@address2	varchar(100)	
@city	varchar(50)	
@state	varchar(30)	
@zip	varchar(20)	
@country	varchar(30)	
@address_type	varchar(30)	
@home_phone	varchar(40)	
@business_phone	varchar(40)	
@fax_phone	varchar(40)	
@mobile_phone	varchar(40)	
@email	varchar(75)	

@org_rec	varchar(1)	Enter 'Y' to check the Org Rec field (indicating an organizational record) or 'N' to leave it unchecked indicating an individual record.
@donor_type	varchar(30)	Set to 'IN' for Individual donor or 'CO' for Corporate donor. You can also check your DPO system for additional DONOR_TYPE field value choices.
@nomail	varchar(1)	
@nomail_reason	varchar(30)	
@narrative	text(2147483647)	
@donor_rcpt_type	Nvarchar(1)	'I' for individual or 'C' for consolidated receipting preference
@user_id	Nvarchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxxxxx&
action=dp_savedonor&params=@donor_id=0 ,@first_name='Orianthi'
,@last_name='Panganaris' ,@middle_name=null ,@suffix=null ,@title=null
,@salutation=null ,@prof_title=null ,@opt_line=null ,@address='4240 Main
St.' ,@address2=null ,@city='North Woodstock' ,@state='NH' ,@zip='12345'
,@country='US' ,@address_type=null ,@home_phone='205-555-1212'
,@business_phone=null ,@fax_phone=null ,@mobile_phone='205-987-6543'
,@email='orianthi@bigstar.com' ,@org_rec='N' ,@donor_type='IN'
,@nomail='N' ,@nomail_reason=null ,@narrative='Narrative field contents
here' , @donor_rcpt_type='C', @user_id='My AppName'
```

Returns:

This command returns the donor ID value of the new donor. In the case of an update, the returned donor_id value is always zero (0).

```
<result>
        <record>
                <field name="" id="" value="147" />
        </record>
</result>
```

Notes:

- 147 is the donor_id of the created donor
- Do not have any spaces in the first part of your command (https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_savedonor&para

ms=) otherwise the API will return an error indicating that the parameter @donor_id was not supplied.

- The null values act as placeholders and must be included in your command

- Character values and date values must be enclosed in single quotes (e.g; '11/01/2010')

- The @user_id value is just there for transaction reference purposes and is not validated against the list of DPO users.

- If you need to save an apostrophe in a name – e.g.; O'Reilly – you must send two apostrophes in a row – e.g; O"Reilly .   This tells the API handler that to interpret this as an apostrophe within the name rather than as the end of a quoted text value.

dp_gifts

Introduction

This procedure returns a predefined set of fields associated with all gifts given by the specified donor.

Parameters:

Parameter	Type	Notes
@donor_id	numeric	

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_gifts&params=1
47 &login=xxx &pass=yyy
```

Returns:

```
<result>
-<record>
<field name="gift_date2" id="gift_date2" value="10/27/2010"/>
<field name="amount" id="amount" value="14.98"/>
<field name="total" id="total" value="0"/>
<field name="sub_solicit_code" id="sub_solicit_code" value="TS"/>
<field name="campaign" id="campaign" value=""/>
<field name="balance" id="balance" value="0"/>
<field name="gl" id="gl" value="NO Rcpt: BOOKS & TAPES"/>
<field name="solicit_code" id="solicit_code" value="BQ10"/>
<field name="reference" id="reference" value=""/>
<field name="record_type" id="record_type" value="G"/>
<field name="gift_id" id="gift_id" value="10230"/>
```

```
<field name="donor_id" id="donor_id" value="147"/>
<field name="anongift" id="anongift" value=""/>
<field name="gift_aid_date" id="gift_aid_date" value=""/>
</record>
</result>
```

Notes:

- If you need different or additional fields returned from the system, use a Dynamic Query SELECT command.

dp_savegift

Introduction

This procedure is used to save changes to an existing gift or to save a new gift into the DPO system

Parameters:

Parameter	Type	Notes
@gift_id	numeric()	Enter 0 in this field to create a new gift or the gift ID of an existing gift. Please note: If you are updating an existing gift, all existing values for the fields specified below will be overwritten by the values you send with this API call.
@donor_id	numeric()	Enter the donor_id of the person for whom the gift will be created
@record_type	varchar(1)	Set as 'G' for a regular gift or for an individual split gift entry within a split gift (i.e.; not the Main top level split), 'P' for Pledge, 'M' for the Main gift in a split gift
@gift_date	datetime()	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@amount	money()	
@gl_code	varchar(30)	If desired, enter the CODE value of the General Ledger code this gift will be associated with. Code values can be found in the DPCODES table. Note: If you are not setting a gl_code value in this field, set the field to null. Do not ever set this field to empty ('').

Parameter	Type	Notes
@solicit_code	varchar(30)	If desired, enter the CODE value of the desired solicit code Note: If you are not setting a solicit_code value in this field, set the field to null. Do not ever set this field to empty ('').
@sub_solicit_code	varchar(30)	If desired, enter the CODE value of the desired sub-solicit code Note: If you are not setting a sub_solicit_code value in this field, set the field to null. Do not ever set this field to empty ('').
@campaign	varchar(30)	If desired, enter the CODE value of the desired campaign code Note: If you are not setting a campaign code value in this field, set the field to null. Do not ever set this field to empty ('').
@gift_type	varchar(30)	
@split_gift	varchar(1)	Set to 'Y' for each of the splits within a split gift but set to 'N' for the Main gift in a split gift or for any gift that is not a split gift.
@pledge_payment	varchar(1)	
@reference	varchar(100)	The associated SafeSave Transaction ID number is normally entered here.
@memory_honor	varchar(30)	
@gfname	varchar(50)	
@glname	varchar(75)	
@fmv	money()	
@batch_no	numeric()	
@gift_narrative	nvarchar(4000)	
@ty_letter_no	varchar(30)	Note: If you are not setting a ty_letter_no code value in this field, set the field to null. Do not ever set this field to empty ('').
@glink	numeric()	In a split gift (not the Main), set the glink value to the gift_id value of the Main gift in the split Also, if you are creating a soft credit gift (record_type='S'), use the GLINK field to identify the gift_id of the actual gift.

Parameter	Type	Notes
		Note: If you are not setting a gift_id value in this field, set the field to null. Do not ever set this field to empty ('') or zero.
@plink	numeric()	This field should be blank for a non-recurring gift but if you are creating a gift that is to be associated with a pledge, set the plink value of the gift to the gift_ID value of the associated pledge. Note: If you are not setting a gift_id value in this field, set the field to null. Do not ever set this field to empty ('') or zero.
@nocalc	varchar(1)	
@receipt	varchar(1)	
@old_amount	money	
@user_id	Nvarchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.
@gift_aid_date	datetime()	This field relates to the UK based Gift Aid Program. See Supplemental Information > Gift Aid Program for more information.
@gift_aid_amt	Money()	This field relates to the UK based Gift Aid Program. See Supplemental Information > Gift Aid Program for more information.
@gift_aid_eligible_g	Char(1)	This field relates to the UK based Gift Aid Program. See Supplemental Information > Gift Aid Program for more information.
@currency		If you use the multi-currency feature, enter appropriate code value per your currency field – e.g; 'USD', 'CAD', etc.
@receipt_delivery_g	varchar(1)	This field sets receipt delivery preference for the specified gift. Supply one of the following single letter code values: • N = do not acknowledge • E = email • B = email and letter • L = letter

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxx&action=d
p_savegift&params=@gift_id=0, @donor_id=640, @record_type='G',
```

```
@gift_date='3/29/2018',  @amount=149.95,  @gl_code='4540-N',
@solicit_code='BQ10',  @sub_solicit_code='TS',  @campaign='CAM2018',
@gift_type='VISAIC',  @split_gift='N',  @pledge_payment='N',
@reference=1234567890123,  @memory_honor=NULL,  @gfname=NULL,
@glname=NULL,  @fmv=0,  @batch_no=0,  @gift_narrative='Gift narrative
text',  @ty_letter_no='TY',  @glink=NULL,  @plink=NULL,  @nocalc='N',
@old_amount=NULL, @receipt='N',  @user_id='My App Name',
@gift_aid_date=NULL, @gift_aid_amt=NULL, @gift_aid_eligible_g=NULL,
@currency='USD'
```

Returns:

```
<result>
        <record>
                <field name="" id="" value="10230" />
        </record>
</result>
```

Notes:

- 10230 is the gift_id of the created gift

dp_savepledge

Introduction

This procedure is used to create, or save changes to, a pledge. It is not used for pledge payments. In DPO, there is a parent pledge (which this command is used to create) that shows up in the DPO pledges tab.   Then, when pledge payments are made, they are created as gifts (record_type='G') using the dp_savegift procedure with a gift_type of 'G' like a regular gift, but add in a '**plink**' value with the gift_id of the parent pledge.

Parameters:

Parameter	Type	Notes
@gift_id	numeric()	Enter 0 in this field to create a new pledge or the gift ID of an existing pledge.
@donor_id	numeric()	Enter the donor_id of the person for whom the pledge is being created/updated
@gift_date	datetime()	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@start_date	datetime()	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@total	money()	Enter either the total amount to be pledged (the sum of all the expected payment amounts) or enter 0 (zero) if

Parameter	Type	Notes
		the pledge amount is to be collected ad infinitum
@bill	money()	Enter the individual monthly/quarterly/annual billing amount
@frequency	varchar (30)	Enter one of: M (monthly), Q (quarterly), S (semi-annually), A (annually)
@reminder	varchar (1)	Sets the pledge reminder flag
@gl_code	varchar(30)	Note: If you are not setting a gl_code value in this field, set the field to null. Do not ever set this field to empty ('').
@solicit_code	varchar(30)	Note: If you are not setting a solicit_code value in this field, set the field to null. Do not ever set this field to empty ('').
@initial_payment	varchar (1)	Set to ''Y' for intial payment, otherwise 'N'
@sub_solicit_code	varchar(30)	Note: If you are not setting a sub_solicit_code value in this field, set the field to null. Do not ever set this field to empty ('').
@writeoff_amount,	money()	
@writeoff_date	datetime()	
@user_id	varchar(20),	
@campaign	varchar(30)	Note: If you are not setting a campaign_code value in this field, set the field to null. Do not ever set this field to empty ('').
@membership_type	varchar(30)	Or NULL
@membership_level	varchar(30)	Or NULL
@membership_enr_date	datetime	Or NULL
@membership_exp_date	datetime	Or NULL
@membership_link_ID	numeric	Or NULL
@address_id	numeric	Or NULL
@gift_narrative	nvarchar(4000)	Or NULL
@ty_letter_no	varchar(30)	Or NULL
@vault_id	numeric()	This field must be populated from the Vault ID number returned by SafeSave for the pledge to be listed as active in the user interface.

Parameter	Type	Notes
@receipt_delivery_g	varchar(1)	'E' for email, 'B' for both email and letter, 'L' for letter, 'N' for do not acknowledge or NULL
@contact_id	numeric()	Or NULL

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxx&action=d
p_savepledge&params=@gift_id=0, @donor_id=640,@gift_date='10/12/2018',
@start_date='10/15/2018',@total=600.00,@bill=30.00,@frequency='M',@remind
er='Y',@gl_code='4540-
N',@solicit_code='BQ10',@initial_payment='N',@sub_solicit_code='TS',
@writeoff_amount=0.00,@writeoff_date='12/29/2018',@user_id='MyApp_Name',@
campaign='REG', @membership_type=NULL, @membership_level=NULL,
@membership_enr_date=NULL, @membership_exp_date=NULL,
@membership_link_id=NULL, @address_id=NULL,@gift_narrative='Gift
narrative text', @ty_letter_no='TY',
@vault_id=123321,@receipt_delivery_g='E',@contact_id=NULL
```

Returns:

```
<result>
        <record>
                <field name="" id="" value="12902" />
        </record>
</result>
```

Notes:

- 12902 is the gift_id of the created gift

dp_savecontact

Introduction

This procedure saves fields to the DPCONTACT table. It will create a new or updated Contact record for the specified donor_id.

Parameters:

Parameter	Type	Notes
@contact_id	numeric()	Enter 0 to create a new record or the other_id record number of an existing dpcontact record

@donor_id	numeric()	Enter the Donor ID of the donor for whom the contact record is to be created or retrieved
@activity_code	varchar(30)	CODE value for the Activity Code field. See DPO Settings > Code Maintenance > Activity Code / Contact Screen. The required values will be listed in the Code column of the resulting display.
@mailing_code	varchar(30)	CODE value for Mailing Code field
@by_whom	varchar(30)	CODE value for the By Whom/Contact Screen field in DPO Description value of selected code shows in the 'Assigned To' field of the contact record.
@contact_date	datetime	Contact / Entry Date field in DPO
@due_date	datetime	Due Date field in DPO. Set as 'MM/DD/YYYY' only. Setting time values here is <u>not</u> supported.
@due_time	varchar(20)	Time field in DPO. Enter as 'hh:mm xx' where xx is either AM or PM – e.g; '02:00 PM'.
@completed_date	datetime	Completed Date field in DPO. Set as 'MM/DD/YYYY' only. Setting time values is <u>not</u> supported.
@comment	varchar(4000)	Contact Notes field in DPO
@document_path	varchar(200)	Type a URL/File Path field in DPO
@user_id	varchar (20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_savecontact&pa
rams=0, 135, 'TE', 'BW02', 'ELF', '09/13/2012', '09/18/2012', '12:24:00
PM', '09/28/2012', 'Your comment on this contact item.',
'http://api.markwarren.ca', 'APIUser' &login=xxx &pass=yyy
```

Returns:

```
<result>
-<record>
<field name="" id="" value="123 />
</record>
</result>
```

Notes:

- 123 is the contact_id associated with the created value

- In this example, the contact_date is 09/13/2012. Typically the current date. The due_date is 09/18/2012 and the completed_date is 09/28/2012 but you would typically leave this field blank by entering null in this field to allow the DPO User to whom the activity was assigned to mark the item as completed when they performed the required activity: e.g.;

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_savecontact&pa
rams=0, 135, 'TE', 'BW02', 'ELF', '09/13/2012', '09/18/2012', '12:24:00
PM', null, 'Your comment on this contact item.',
'http://api.markwarren.ca', 'API User' &login=xxx &pass=yyy
```

dp_saveotherinfo

Introduction

This procedure saves fields to the dpotherinfo table. It will create a new or updated 'Other **Info'** record for the specified donor_id.

Parameters:

Parameter	Type	Notes
@other_id	numeric()	Enter 0 to create a new record or the other_id record number of an existing dpotherinfo record
@donor_id	numeric()	Enter the donor_id for whom the record is to be created / updated.
@other_date	datetime()	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@comments	nvarchar(4000)	
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_saveotherinfo&
params=0,147,'11/18/2010','my comment', 'MyAppName' &login=xxx &pass=yyy
```

Returns:

```
<result>
-<record>
<field name="" id="" value="160" />
</record>
</result>
```

Notes:

- 160 is the other_id associated with the updated value

- This other_id value can now be used as a @matching_id value to save additional fields associated with the dpotherinfoudf table associated with this entry.

dp_saveaddress

Introduction

Use this API call to add or update secondary address values that appear in the DonorPerfect Online Address tab. These secondary addresses are also referred to as Seasonal Addresses in the DPO Knowledge Base.

Parameters

Parameter	Type	Notes
@address_id	numeric	Enter 0 in this field to create a new address value or the address_id number of an existing address_id to update the existing value
@donor_id	numeric	Specify the donor_id of the donor associated with this
@opt_line	varchar(100)	Enter a secondary name or company name if appropriate
@address	varchar(100)	
@address2	varchar(100)	
@city	varchar(50)	
@state	varchar(30)	
@zip	varchar(20)	
@country	varchar(30)	
@address_type	varchar(30)	Enter the CODE value associated with the address type
@getmail	varchar(1)	Enter 'Y' or 'N' to indicate whether the Receive Mail box will be checked and to

		indicate whether mail can be sent to this address.
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.
@title	varchar(50)	Enter a value to be stored in the Professional Title field.
@first_name	varchar(50)	
@middle_name	varchar(50)	
@last_name	varchar(75)	
@suffix	varchar(50)	
@prof_title	varchar(100)	
@salutation	varchar(130)	Enter desired salutation value (e.g.; 'Dear Bob')
@seasonal_from_date	varchar(4)	Enter the 'from' date as **MMYY** – e.g; November 2017 would be represented as 1117
@seasonal_to_date	varchar(4)	Enter the 'to' date as **MMYY**
@email	varchar(75)	
@home_phone	varchar(40)	
@business_phone	varchar(40)	
@fax_phone	varchar(40)	
@mobile_phone	varchar(40)	
@address3	varchar(100)	
@address4	varchar(100)	
@ukcountry	varchar(100)	
@org_rec	varchar(1)	Enter 'Y' to check the Org Rec field (indicating an organizational record) or 'N' to leave it unchecked to indicate an individual record.

Sample Call

```
…&action=dp_saveaddress&params='167','4220',null,'12 Main
St.',null,'Potsdam','NY','33132','USA','Home','N','UserID','title','first
_name','middle_name','last_name','suffix','prof_title','salutation','0717
','0817','email@address.com','home_phone','business_phone','fax_phone','m
obile_phone','address3','address4','ukcountry','Y'
```

Returns

This API call returns the @address_id value of the newly created (or updated) DPADDRESS value.

```
<result>
  <record>
    <field name="" id="" value="167"/>
  </record>
</result>
```

dp_save_udf_xml

Introduction

This procedure saves a Donor's extended information (User Defined Fields) —used to save changes to the user-defined fields that are custom for each client and are not part of the standard DPO system.

This procedure will save a single parameter for a specified User Defined Field (UDF).

Parameters:

Parameter	Type	Notes
@matching_id	Numeric()	Specify either an existing donor_id value if updating a donor record, a gift_id value if updating a gift record, a contact_id number if updating a contact record or an other_id value if updating a dpotherinfo table value (see dp_saveotherinfo).
		If you are updating a value in the DPADDRESSUDF, specify the @address_ID as the matching ID. Also, FYI, you can link the DPADDRESS.ADDRESS_ID=DPADDRESSUDF.ADDRESS_ID in any of your SELECT queries.
@field_name	varchar(20)	
@data_type	varchar(1)	C- Character, D-Date, N- numeric
@char_value	Nvarchar(2000)	Null if not a Character field
@date_value	Datetime()	Null if not a Date field. Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@number_value	numeric()	Null if not a Number field
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing

		purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxx&action=dp_save_udf_xml&params=16013,'ALT_PHONE','C','555-1212',null,null,'MyAppName'

https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxx&action=dp_save_udf_xml&params=@matching_id=348,@field_name='FIELDNAME',@data_type='C',@char_value='new value here',
@date_value=null,@number_value=null,@user_id='MyAppName'
```

Returns:

```
<result>
-<record>
<field name="" id="" value=" 16013" />
</record>
</result>
```

Notes:

- 16013 is the donor_id associated with the updated value

- This example assumes that a user defined field called ALT_PHONE already exists in your system

dp_savecode

Introduction

This procedure allows you to add new code and description values to the DPCODES table. This would allow you to add things like new GL Code, Solicitation, Campaign code and other code values administered in the Code Maintenance screen of DPO.

Edit	Field Name	Code	Description ▲	Active (Click to Change)	Delete
✎	EMAIL_TYPE	OTHER	Other	✔	N/A
✎	EMAIL_TYPE	PERSONAL	Personal	✔	N/A
✎	EMAIL_TYPE	WORK	Work	✔	N/A

While this API call requires the specification of many fields, you will typically only supply the Field Name, Code, Description, Inactive field value and we also recommend you supply the @user_id value as well.

Parameters

Parameter	Type	Notes
@field_name	varchar(20)	Enter the name of an existing field type from the DPCODES table
@code	varchar(30)	Enter the new CODE value
@description	varchar(100)	Enter the description value that will appear in drop-down selection values
@original_code	varchar(20)	Enter NULL
@code_date	Datetime()	Enter NULL
@mcat_hi	Money()	Enter NULL
@mcat_lo	Money()	Enter NULL
@mcat_gl	varchar(1)	Enter NULL
@acct_num	varchar(30)	Enter NULL
@campaign	varchar(30)	Enter NULL
@solicit_code	varchar(30)	Enter NULL
@overwrite		
@inactive	varchar(1)	Enter 'N' for an active code or 'Y' for an inactive code. Inactive codes are not offered in the user interface dropdown lists.
@client_id		Enter NULL
@available_for_sol		Enter NULL
@user_id	varchar(20)	Enter the name of your API application
@cashact		Enter NULL
@membership_type		Enter NULL
@leeway_days		Enter NULL
@comments	varchar(2000)	You may enter a comment of up to 2000 characters if you like.
@begin_date	datetime	. Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@end_date	datetime	. Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@ty_prioritize	char(1)	Enter NULL
@ty_filter_id		Enter NULL
@ty_gift_option		Enter NULL
@ty_amount_option		Enter NULL
@ty_from_amount		Enter NULL
@ty_to_amount		Enter NULL

@ty_alternate		Enter NULL
@ty_priority		Enter NULL

Sample Call

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=...&action=dp_sav
ecode&params=@field_name='GL_CODE', @code='4800-03',
@description='Building Fund:
Furnishings',@original_code=null,@code_date=null,@mcat_hi=null,@mcat_lo=n
ull,@mcat_gl=null,@reciprocal=null,@mailed=null,@printing=null,@other=nul
l,@goal=null,@acct_num=null,@campaign=null, @solicit_code=null,
@overwrite=null,@inactive='N',@client_id=null, @available_for_sol=null,
@user_id='My API App', @cashact=null, @membership_type=null,
@leeway_days=null,@comments=null, @begin_date=null, @end_date=null,
@ty_prioritize=null, @ty_filter_id=null, @ty_gift_option=null,
@ty_amount_option=null, @ty_from_amount=null, @ty_to_amount=null,
@ty_alternate=null, @ty_priority=null
```

Returns

```
<result>
<record>
<field name="result" id="result" value="0"/>
</record>
</result>
```

Notes

Set @inactive='N' to indicate that this entry is Active and will appear in the appropriate drop-down field in the user interface.

dp_savelink

Introduction

This procedure creates a link between two donors. These links show in the Links tab of the DPO user interface. Note that reciprocal links are created automatically. For example, if you create a Friend link (FR) link with donor A as the donor_id value and donor B as the donor_id2 value, you will automatically see the Friend link in the DPO user interface if you look at the Links tab in either donor A or donor B.   Please also note that DPO will create appropriate reciprocal links in the same way it does via the user interface. So, if you create a Parent link from donor A to donor B, then donor B will show donor A as a Child link.

Parameters:

Parameter	Type	Notes
@link_id	numeric()	Enter zero (0) to create a new link entry of the link_id of an existing entry if you are updating an existing entry
@donor_id	Numeric()	Enter the donor_id of the donor where the link is being created
@donor_id2	Numeric()	Enter the donor_id of the OTHER donor involved in the link
@link_code	Varchar(30)	Enter the CODE value of the link per the Link Type values shown in the Code Maintenance screen of the DPO user interface
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=
dp_savelink&params=0,144,135,'PA','MyApplication'&login=…
```

Returns:

```
<result>
<record>
<field name="" id="" value="12345"/>
</record>
</result>
```
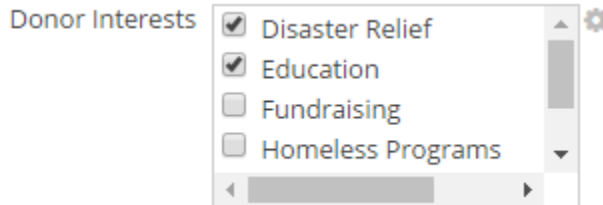
Notes:

- The returned value is the link_id of the created link

- The User ID value does not have to correspond to an actual user and simply provides a reference for who checked the box. In the case of API applications, we recommend you use your API application name

dp_savemultivalue_xml

Introduction

This procedure allows you to set a checkbox value. Checkbox fields in DPO are stored in the DPUSERMULTIVALUES table and use a MATCHING_ID value as the index. The MATCHING_ID value corresponds to the associated DONOR_ID. Note that even though a checkbox field may contain many checkboxes, there will only be an entry present in DPUSERMULTIVALUES for checkboxes that are set (checked).

Parameters:

Parameter	Type	Notes
@matching_id	numeric	Specify the desired donor_id
@field_name	varchar(20)	Use the **code** value associated with the flag. For example, the 'AL', flag in this example had a description value of 'Alumni'.
@code	Varchar(30)	Enter the Code value of the checkbox entry you wish to set
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_savemultivalue
_xml&params=12345,'FIELDNAME','CODE','AppName'&login=xxx &pass=yyy
```

Returns:

```
<result>
<record>
<field name="" id="" value="12345"/>
</record>
</result>
```

Notes:

- This example assumes the existence of a checkbox field with FIELD_NAME = 'FIELDNAME' and a checkbox with the CODE = 'CODE'

- The User ID value does not have to correspond to an actual user and simply provides a reference for who checked the box. In the case of API applications, we recommend you use your API application name

mergemultivalues

Introduction

This procedure allows you to set field values for an individual checkbox field. Any values not specified will be unset (unchecked).

Parameters:

Parameter	Type	Notes
@matchingid	numeric	Specify the desired donor_id
@fieldname	varchar(20)	Enter the name of the checkbox field name.
@valuestring	varchar(20)	Enter any CODE values to be set. Separate with commas. Any code values not specified will be unset (unchecked).
@debug	Numeric	Specification of this field is optional but if you want to return the list of checkbox fields and the values in them after running this command then add @debug=1 as a parameter to this API call. If a code was previously set but was not specified in your mergemultivalues API call then it will show as a DeletedCode value. If a value was not previously set but was specified in your API call, then it will show as an InsertedCode. See example below.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxxx
&action=mergemultivalues&params=@matchingid=16,@fieldname='GTYPES',@value
string='CR, RE, ST',@debug=1
```

Returns:

```
<result>
<record>
<field name="ActionType" id="ActionType" value="INSERT"/>
<field name="DeletedID" id="DeletedID" value=""/>
<field name="DeletedCode" id="DeletedCode" value=""/>
<field name="DeletedFieldName" id="DeletedFieldName" value=""/>
<field name="InsertedTargetID" id="InsertedTargetID" value="16"/>
<field name="InsertedCode" id="InsertedCode" value="CR"/>
<field name="InsertedFieldName" id="InsertedFieldName" value="GTYPES"/>
</record>
<record>
<field name="ActionType" id="ActionType" value="INSERT"/>
<field name="DeletedID" id="DeletedID" value=""/>
<field name="DeletedCode" id="DeletedCode" value=""/>
<field name="DeletedFieldName" id="DeletedFieldName" value=""/>
<field name="InsertedTargetID" id="InsertedTargetID" value="16"/>
<field name="InsertedCode" id="InsertedCode" value="RE"/>
<field name="InsertedFieldName" id="InsertedFieldName" value="GTYPES"/>
</record>
<record>
<field name="ActionType" id="ActionType" value="INSERT"/>
<field name="DeletedID" id="DeletedID" value=""/>
<field name="DeletedCode" id="DeletedCode" value=""/>
<field name="DeletedFieldName" id="DeletedFieldName" value=""/>
<field name="InsertedTargetID" id="InsertedTargetID" value="16"/>
<field name="InsertedCode" id="InsertedCode" value="ST"/>
<field name="InsertedFieldName" id="InsertedFieldName" value="GTYPES"/>
</record>
<record>
<field name="ActionType" id="ActionType" value="DELETE"/>
<field name="DeletedID" id="DeletedID" value="16"/>
<field name="DeletedCode" id="DeletedCode" value="BE"/>
<field name="DeletedFieldName" id="DeletedFieldName" value="GTYPES"/>
<field name="InsertedTargetID" id="InsertedTargetID" value=""/>
<field name="InsertedCode" id="InsertedCode" value=""/>
<field name="InsertedFieldName" id="InsertedFieldName" value=""/>
</record>
<record>
<field name="ActionType" id="ActionType" value="DELETE"/>
<field name="DeletedID" id="DeletedID" value="16"/>
<field name="DeletedCode" id="DeletedCode" value="PI"/>
<field name="DeletedFieldName" id="DeletedFieldName" value="GTYPES"/>
<field name="InsertedTargetID" id="InsertedTargetID" value=""/>
<field name="InsertedCode" id="InsertedCode" value=""/>
<field name="InsertedFieldName" id="InsertedFieldName" value=""/>
</record>
</result>Notes:
```

If the @debug=1 is not included in the API call, it returns this:

```
<result/>
```

Notes

This is the call to use instead of dp_deletemultivalues_xml if you ever need to unset (uncheck) a checkbox field because it allows you to modify a single checkbox field instead of having to retrieve all checkbox fields for the screen tab and re-set them.

Depending on what you need to do, you can probably also use this instead of the dp_savemultivalues_xml API call as well – as it allows you to set (check) checkboxes as well as un-set them.

Also, please note that the field specifications are slightly different than the usual when you are entering the command string – e.g.; @matchingid is correct. @matching_id is incorrect.

You can retrieve a list of the possible code values for the desired field with a SELECT query like this:

```
&action=select code, description from dpcodes where field_name='{field
name here}'
```

You can retrieve a list of flags set for a specific flag for a specific donor_id with a SELECT query like this:

```
&action=select id, matching_id, code from dpusermultivalues where
field_name='{field name}' and matching_id={donor_id}
```

dp_deletemultivalues_xml

Introduction

This procedure deletes ALL checked checkbox values on the specified screen tab (e.g.; Main, Gift, Pledge,Bio, Other) for the specified donor.
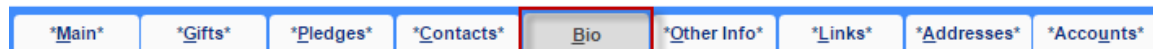
| *Main* | *Gifts* | *Pledges* | *Contacts* | Bio | *Other Info* | *Links* | *Addresses* | *Accounts* |

Checkbox fields in DPO are stored in the DPUSERMULTIVALUES table and use a MATCHING_ID value as the index. The MATCHING_ID value corresponds to the associated DONOR_ID.   Note that even though a checkbox field may contain many checkboxes, there will only be an entry present in DPUSERMULTIVALUES for checkboxes that are set (checked).

We recommend that before using this API call, you use a dynamic SELECT statement to retrieve all set (checked) values for the specified donor. This will allow you to follow the delete API call with dp_savemultivalues_xml API calls to re set/check the checkbox fields you did not want deleted.

```
Example: …&action=select field_name, code from dpusermultivalues where
field_name='{field name}' and matching_id={donor_id)
```

Parameters:

Parameter	Type	Notes

@matching_id	numeric	Specify the desired donor_id
@table_name	varchar(20)	Enter the name of the DPO screen tab (i.e. MAIN, GIFT, PLEDGE, BIO, OTHER) where <u>all</u> checked fields are to be deleted
@user_id	varchar(20)	Enter a user name that will be associated with this update. Normally with will be the name you associate with your API application.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxxx&act
ion=dp_deletemultivalues_xml&params=135,'BIO','Set via API'
```

Returns:

```
<result>
-<record>
<field name="" id="" value=" 135" />
</record>
</result>
```

Notes:

- 135 is the donor_id associated with the updated value
- This API call removes ALL checked values from the specified DPO screen tab.

dp_saveflag_xml

Introduction

This procedure allows you to set flags as shown in the top section of the Main tab. The Flags field is on the Main tab in the DPO user interface.

Flags must have been previously created in Settings > Code Maintenance and the value you set corresponds to the Code value, not the description value.

Parameters:

Parameter	Type	Notes
@donor_id	Numeric()	Specify either a donor_id value if updating a donor record, a gift_id value if updating a gift record or an other_id value if updating a dpotherinfo table value (see dp_saveotherinfo)
@flag	varchar(30)	Use the **code** value associated with the flag. For example, the 'AL', flag in this example had a description value of 'Alumni'.
@user_id	varchar(20)	We recommend that you use a name here, such as the name of your API application, for auditing purposes. The user_id value does not need to match the name of an actual DPO user account.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_saveflag_xml&params=135,'AL','xmluser'&login=xxx &pass=yyy
```

Returns:

```
<result>
-<record>
<field name="" id="" value=" 135" />
</record>
</result>
```

Notes:

- 135 is the donor_id associated with the updated value

- This example assumes that a user defined field called ALT_PHONE already exists in your system

- If the flag was already set, the system returns an error like this:
  [Microsoft][ODBC SQL Server Driver][SQL Server]Violation of PRIMARY KEY constraint 'PK_dpflags'. Cannot insert duplicate key in object

- To view the flags set for a specified donor, use this command format:

<u>https://www.donorperfect.net/prod/xmlrequest.asp?action=select * from dpflags where donor_id=135&login='xxx'&pass='yyy'</u>

dp_delflags_xml

Introduction

This procedure **removes (deletes) all flags for the specified donor**. Flags are shown on the main donor screen in DPO.

Parameters:

Parameter	Type	Notes
@donor_id	numeric	Specify the donor_id of the donor for whom the flags (all of them) are to be deleted
@user_id	varchar(20)	

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_delflags_xml&params=135,'xmluser'&login=xxx &pass=yyy
```

Returns:

```
<result>
<record>
<field name="" id="" value="135"/>
</record>
</result>
```

Notes:

- In this example, 135 is the donor_id associated with the deleted flags

- It is not currently possible to delete individual flags for a specified donor. This command deletes all flags set for the specified donor_id.

- To view the flags set for a specified donor, use a SELECT query to retrieve a list of all set flags for the specified donor. You will use this list of set flags to re-set all flags you did not want un-set for the specified donor: SELECT * FROM DPFLAGS WHERE DONOR_ID={desired donor_id}

- See dp_saveflag_xml for information on setting flags.

dp_tribAnon_MyTribSummary

Introduction

This API call has been added to support the new DPO tribute functionality. It retrieves either a list of

- All tributes **or**

- All Active tributes (<field id="ActiveFlg" value="True" name="ActiveFlg")

Parameters

Parameter	Type	Notes
@ShowAllRecords	numeric	If set to 1, retrieves all tributes If set to 0, retrieves all ACTIVE tributes but does not include inactive tributes – i.e.; tributes where the ActiveFlg='False'
@userId	numeric	Enter null. This parameter is not applicable to API calls.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxxxx&action=dp_tribAnon_MyTribSummary&params=@showallrecords=1,@userId=null
```

Returns:

```
<record>
<field name="TributeID" id="TributeID" value="4"/>
<field name="ActiveFlg" id="ActiveFlg" value="True"/>
<field name="Type" id="Type" value="In Honor Of"/>
<field name="Name" id="Name" value="In Honor of Ishmael"/>
<field name="Created" id="Created" value="8/22/2011"/>
<field name="Gift_Amt" id="Gift_Amt" value="534"/>
<field name="Gift_Count" id="Gift_Count" value="12"/>
<field name="Notif_Name" id="Notif_Name" value="Mark Panagaris"/>
</record>
…
```

Notes

The User Identification Number parameter must be entered as 'null' to process this API call.

Either of these two parameter combinations work (depending on whether you want all tributes or just the active ones:

- &params=0,null

- &params=1,null

dp_tribAnon_Search

Introduction

This API call allows you to search for a tribute by the tribute name and either include or exclude inactive tributes.

Parameters

Parameter	Type	Notes
@Keywords	varchar(200)	Enter all or part of the tribute name to retrieve data on each matching tribute.
@IncludeInactive	Int()	Enter 1 to include inactive tributes (ActiveFlg=False) or enter 0 to exclude these.

Sample Call

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxx&action=dp_t
ribAnon_Search&params=@keywords='Trib',@includeinactive=1
```

Returns

```
<record>
<field name="TributeID" id="TributeID" value="8"/>
<field name="Name" id="Name" value="My New Tribute"/>
<field name="DPCode_ID" id="DPCode_ID" value="707"/>
<field name="ActiveFlg" id="ActiveFlg" value="True"/>
<field name="UserCreateDT" id="UserCreateDT" value="4/1/2017"/>
<field name="Created_Date" id="Created_Date" value="4/1/2017 11:23:00
AM"/>
<field name="Legacy_ID" id="Legacy_ID" value=""/>
<field name="CodeDescription" id="CodeDescription" value="In Honor Of"/>
</record>
<record>
<field name="TributeID" id="TributeID" value="9"/>
<field name="Name" id="Name" value="My New Tribute"/>
<field name="DPCode_ID" id="DPCode_ID" value="707"/>
<field name="ActiveFlg" id="ActiveFlg" value="False"/>
<field name="UserCreateDT" id="UserCreateDT" value="4/1/2017"/>
<field name="Created_Date" id="Created_Date" value="4/4/2017 3:26:29
PM"/>
<field name="Legacy_ID" id="Legacy_ID" value=""/>
<field name="CodeDescription" id="CodeDescription" value="In Honor Of"/>
</record>
```

Notes

If you already have a tribute that has the same type (CodeDescription) as one you are trying to create, you can use the matching DPCode_ID value.

Example from above: If you needed to create a new tribute that is an 'In Honor Of' tribute, you can safely assume the code of 707 is the correct DPCodeID value in the dp_tribAnon_Create API call for a new tribute of the same type.

dp_tribAnon_Create

Introduction

This API call is used to create a new tribute in DonorPerfect Online (DPO) .

Parameters

Parameter	Type	Notes
@Name	varchar(200)	Specify the name that will be used for the tribute
@DPCodeID	int()	This is the numeric code_ID value that is associated with the tribute type. The standard values are M (In Memory Of) and H (In Honor Of) but you will not be specifying the letter value here but rather the numeric value of the code_ID. You can get the required @code_id value with this SQL SELECT query: SELECT CODE, CODE_ID, DESCRIPTION FROM DPCODES WHERE FIELD_NAME = 'MEMORY_HONOR' Run the query and record the CODE_ID values. They will not change for your DonorPerfect system. If you are connecting to multiple DonorPerfect systems, you will need to run this once for each system you are connecting to and store the values.
@ActiveFlg	Bit()	Enter 1 here to make the tribute active or 0 to make it inactive.
@UserCreateDt	Datetime()	Enter the current date in this format: mm/dd/yyyy and place single quotes around the date. Entry of time values is NOT supported.
@Recipients	Numeric()	Enter either the donor_id of a single recipient OR multiple donor_id values separated by the pipe symbol and wrapped in single quotes. See examples below. The second example shows four donor ID numbers (11101, 22202, 33303, 44404) being assigned as Recipients and then in the Returns section below that, you can see the

		names of the donors who correspond to those donor ID numbers.
		Note: It is possible to create a tribute without specifying any @recipients by omitting this parameter.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxx
&action=dp_tribAnon_Create&params=@name= 'My First Tribute',
@dpcodeid=708, @activeflg=1, @usercreatedt='03/15/2017', @recipients=575

https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxx&actio
n= dp_tribAnon_Create&params=@name='My Second Tribute', @dpcodeid=707,
@activeflg=1, @usercreatedt='03/15/2017',
@recipients='11101|22202|33303|44404'
```

Returns:

```
<result/>
```

Notes

Sample returned @dpcodeid values from a SQL SELECT query:

```
SELECT CODE, CODE_ID, DESCRIPTION FROM DPCODES WHERE FIELD_NAME =
'MEMORY_HONOR'.
```

You need to run this query for your DPO system to determine the @DPCodeID values you need for your DPO system :

```
<result>
<record>
<field name="CODE" id="CODE" value="H"/>
<field name="CODE_ID" id="CODE_ID" value="707"/>
<field name="DESCRIPTION" id="DESCRIPTION" value="In Honor Of"/>
</record>
<record>
<field name="CODE" id="CODE" value="M"/>
<field name="CODE_ID" id="CODE_ID" value="708"/>
<field name="DESCRIPTION" id="DESCRIPTION" value="In Memory Of"/>
</record>
</result>
```
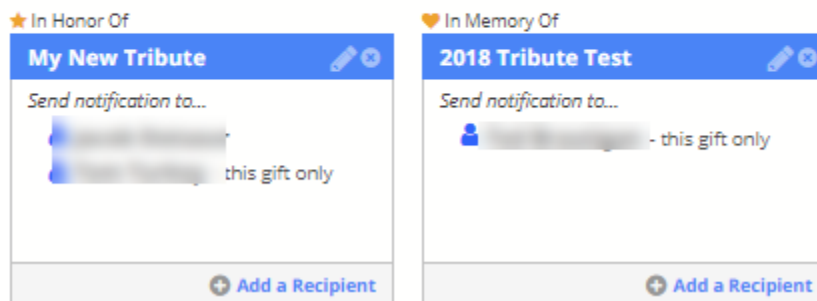
dp_tribAnon_AssocTribsToGift

Introduction

This API call is used to create the association between a gift and a tribute by specifying the GIFT_ID and the TributeID_List

Once you have associated a tribute to a gift, the tribute will show in the Tribute Details section of the gift screen.

Parameters

Parameter	Type	Notes
@Gift_ID	numeric	Specify the GIFT_ID of the gift that will be associated with the specified tribute
@TributeID_List	numeric	Enter the TributeID of the tribute. You can optionally enter a comma separated list of tribute ID numbers – e.g.; 9, 12, 33 Tribute IDs are included in the data retrieved in the dp_tribAnon_MyTribSummary API call.

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?login=xxx&pass=yyy&action=dp_tribAnon_AssocTribsToGift&params=@giftid=10561, @tributeid_list=12
```

Returns:

```
<result>
<record>
```

```
<field name="TributeID" id="TributeID" value="12"/>
</record>
</result>
```
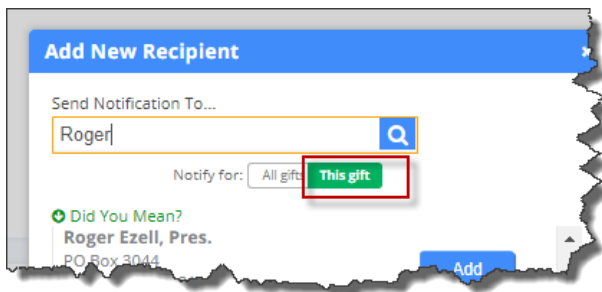
Notes

Use the dp_tribAnon_MyTribSummary API call to retrieve the TributeID values.

# dp_tribAnon_SaveTribRecipient

Introduction

Use this API call to add a new recipient notification for a particular gift (This gift):

When you have a gift that already has tribute(s) applied to it, the user interface shows an option to **Add a Recipient**. The resulting window gives you the opportunity to send a notification to an additional recipient, and specifically, for just this gift. This API call supports that functionality.
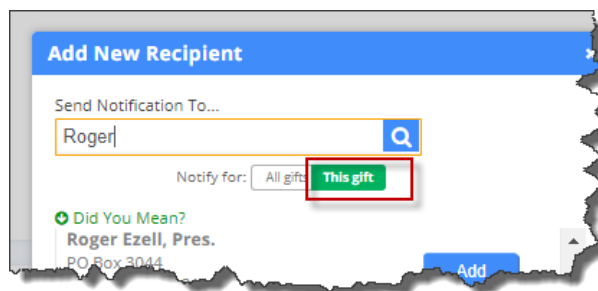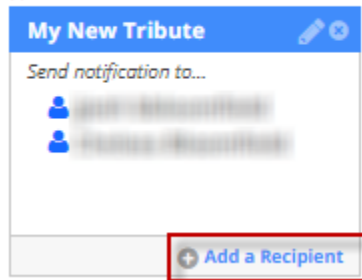
Parameters

Parameter	Type	Notes

@DonorId	numeric	The Donor ID number of the new tribute notification recipient. Example, if you are notifying Susan about Frank's gift, you would specify Susan's Donor_ID number here
@TributeID	numeric	The ID number of the tribute
@GiftID	numeric	The Gift ID number of the gift this donor will be notified about. Per example above, you would specify the Gift_ID number of Fred's gift here.
@Level	varchar (20)	**Always set this to 'L'**

Sample Call

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxxxxxxxx
&action=dp_tribAnon_SaveTribRecipient&params=@donorid=640, @tributeid=9,
@giftid=12879, @level='L'
```

Notes

To use this API call, it is **important** to use the named field syntax shown in the example – e.g; @this=1, @that=2,.

To locate the recipient in the DPO user interface

- Go to Tributes under the cog icon (top right hand corner of user interface)
- Search for and select the tribute by its Tribute ID number (above)
- You should see the recipient donor under the 'Send Notifications To' text

The dp_tribNotif_Save API call should be run after the dp_tribAnon_SaveTribRecipient API call to make sure that the actual notification record is created for the recipient of the notification.

dp_tribNotif_Save

Introduction

This API call creates a tribute gift notification record for a person who is to be notified of a tribute. The tribute gift notification records added with this API call are for the notification recipient for **this gift only**.

Parameters

Parameter	Type	Notes
@Gift_ID	numeric	Enter 0 to create a new tribute gift notification record or the gift_id number of an existing tribute gift notification record if you are updating one.
@Donor_Id	numeric	The Donor ID number of the new notification record recipient. Please note that this may be a different donor than the person who gave the original tribute gift.
@glink	numeric	The Gift ID number of the gift this donor will be notified about
@tlink	numeric	The Tribute ID number of the tribute
@smount	Money	Enter the amount of the gift
@total	Money	Enter the amount of the gift
@bill	Money	Enter 0.00
@start_date	Date	Enter null
@frequency	Varchar	Enter null
@gift_type	Varchar	Enter 'SN'
@record_type	Varchar	Enter 'N' to indicate this is a notification record
@gl_code	Varchar	Enter same value that was set for the original gift
@solicit_code	Varchar	Enter same value that was set for the original gift
@sub_solicit_code	Varchar	Enter same value that was set for the original gift
@campaign	Varchar	Enter same value that was set for the original gift
@ty_letter_no	Varchar(30)	Enter 'NT' for this field.
@fmv	Money	Enter same value that was set for the original gift
@reference	Varchar	Enter SafeSave transaction record or null
@gfname	Varchar(50)	null
@glname	Varchar(75)	null
@gift_narrative	Nvarchar(4000)	If desired, enter gift narrative text
@membership_type	Varchar	Null
@membership_level	Varchar	Null

@membership_enr_date	Date	Null
@membership_exp_date	Date	Null
@address_id	Numeric	Enter 0 unless the notification letter is to go to an address other than the one shown on the Main screen.
@user_id	Varchar(20)	Enter the name of your API application – max 20 characters

Sample Call

```
https://www.donorperfect.net/prod/xmlrequest.asp?apikey=xxxxx&action=dp_t
ribNotif_Save&params=@Gift_ID=0, @Donor_Id=642, @glink=12879, @tlink=12,
@amount=23.23, @total=23.23, @bill=0.00, @start_date=null,
@frequency=null, @gift_type='SN', @record_type='N', @gl_code='MIS',
@solicit_code='BD04', @sub_solicit_code=null, @campaign=null,
@ty_letter_no='NT', @fmv=0.00, @reference=null, @gfname=null,
@glname=null, @gift_narrative='This is a gift narrative comment',
@address_id=0, @user_id='APIApp'
```

Returns

```
<result>
<record>
<field name="" id="" value="12911"/>
</record>
</result>
```

Notes

The notification shows up as a new gift for donor ID 642 with the Type of Gift field set to Send Notification.

The dp_tribNotif_Save API call should be run after the dp_tribAnon_SaveTribRecipient API call to make sure that the actual notification record is created for the recipient of the notification.

Tribute gift notification records will show in the Linked Gift screen of the original gift (i.e.; the gift_id specified in the @glink field in a table called Linked Gifts. This table will include all notification gifts for all tributes linked to this gift.

dp_tribAnon_Update

Introduction

User this API call if you need to update the list of recipients for all gifts that are associated with a particular tribute. This API call is different that the dp_tribAnon_SaveTribRecipient and dp_tribNotif_Save API calls which add a recipient to a tribute for one specified gift.

It is API equivalent to the Add Recipient link in the Edit Tribute screen (shown below):

Provide Tribute Details

Type*

In Honor Of

Name*

In Honour of Ishmael

Date*

04/08/2019

✅ Active

Send Notifications To...

👤 Mark Panagaris ✏ ✖

👤 Tom Turkey ✏ ✖

👤 Ted Brautigan ✏ ✖

➕ Add Recipient

Parameters

Parameter	Type	Notes
@TributeID	Numeric ()	Enter the ID number of the tribute you are updating
@name	Nvarchar (200)	Enter the existing name of the tribute to be updated. See Notes section below for example of a SELECT statement that will give you this info.
@dpcode_id	Numeric ()	This is the numeric code_ID value that is associated with the tribute type. See Notes section below for example of a SELECT statement that will give you this info.
@ActiveFlg	Bit	Set as 1 for True (active) or 0 for False (inactive)

@UserCreateDt	Date	Enter existing user create date in this format: 'MM/DD/YYYY'
@recipients	Numeric()	Enter the list of all existing and any new recipients using the pipe | symbol as a delimiter. ALSO, prefix the list with capital letter N and also wrap the recipients list in single quotes. Example: `@recipients=N'105|43256|323387|137'`

Sample Call

```
… &action=dp_tribAnon_Update&params=@tributeID=4,@name='In Honour of
Ishmael',@DPCode_id=707,@ActiveFlg=1,@UserCreateDt='04/08/2019',@Recipien
ts=N'135|640|642'
```

Returns

The donor_id numbers you supplied in the API call are replaced with the names associated with the specified donor_id numbers:

```
<result>
<record>
<field name="TributeID" id="TributeID" value="4"/>
<field name="Name" id="Name" value="In Honour of Ishmael"/>
<field name="ActiveFlg" id="ActiveFlg" value="True"/>
<field name="UserCreateDt" id="UserCreateDt" value="4/8/2019"/>
<field name="DPCode_ID" id="DPCode_ID" value="707"/>
<field name="CodeDescription" id="CodeDescription" value="In Honor Of"/>
<field name="NotificantNames" id="NotificantNames" value="Ted Brautigan,
Mark Panagaris, Tom Turkey"/>
</record>
</result>
```

Notes

IMPORTANT: You need to first retrieve info on the existing tribute @recipients as you must include these in the dp_tribAnon_Update call otherwise they will no longer be associated with the tribute.

Here is an example of a SELECT statement that will give you the field values you will need for an existing tribute:

```
…&action=select r.tributeid, t.name, r.donor_id, r.createddate,
t.activeflg from dptributesanon t, dptributesanon_recipients r where
t.tributeid=r.tributeid and t.tributeid={desired tributeid number}
```

It is not necessary to follow this API call with the dp_tribNotif_Save API call.

dp_PaymentMethod_Insert

Introduction

This procedure allows insertion of DPO Payment Method values.   This table is used on systems with the EFT Transactions feature enabled.

This procedure will save a single parameter for a specified User Defined Field (UDF).

Parameters:

Parameter	Type	Notes
@CustomerVaultID	Nvarchar(55)	Enter -0 to create a new Customer Vault ID record
@donor_id	Int()	
@IsDefault bit	Bit	Enter 1 if this is will be the default EFT payment method
@AccountType	Nvarchar(256)	e.g. 'Visa'
@dpPaymentMethodTypeID	Nvarchar(20)	e.g.; 'creditcard'
@CardNumberLastFour	Nvarchar(16)	e.g.; '4xxxxxxxxxxx1111
@CardExpirationDate	Nvarchar(10)	e.g.; '0810'
@BankAccountNumberLastFour	Nvarchar(50)	
@NameOnAccount	Nvarchar(256)	
@CreatedDate	datetime	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@ModifiedDate	datetime	Set as 'MM/DD/YYYY' only. Setting time values is not supported.
@import_id	int	
@created_by	Nvarchar(20)	
@modified_by	Nvarchar(20)	
@selected_currency	Nvarchar(3)	

Sample Call:

```
https://www.donorperfect.net/prod/xmlrequest.asp?action=dp_paymentmethod_
insert&params=0,147,1,'Visa','creditcard','4xxxxxxxxx1234','0412',null,'O
riana',null,null,null,null,null,'USD' &login=xxx &pass=yyy
```

Returns:

```
<result>
```

```
-<record>
<field name="DpPaymentMethodID" id="DpPaymentMethodID" value="2"/>
</record>
</result>
```

Notes:

- This table would normally only be populated from an ecommerce API where the DPO system has EFT Transactions enabled.

Here are two samples of XML data retrieved from a Dynamic SELECT Query on the DpPaymentMethod table from a system with EFT Transactions enabled. :

Sample Check Item:

```
- <record>
  <field name="DpPaymentMethodID" id="DpPaymentMethodID" value="402" />
  <field name="donor_id" id="donor_id" value="123456" />
  <field name="IsDefault" id="IsDefault" value="True" />
  <field name="AccountType" id="AccountType" value="Bank Account" />
  <field name="dpPaymentMethodTypeID" id="dpPaymentMethodTypeID"
value="check" />
  <field name="CardNumberLastFour" id="CardNumberLastFour" value="" />
  <field name="CardExpirationDate" id="CardExpirationDate" value="" />
  <field name="BankAccountNumberLastFour" id="BankAccountNumberLastFour"
value="7xxxx6543" />
  <field name="NameOnAccount" id="NameOnAccount" value="Arthur
Roundtable" />
  <field name="CreatedDate" id="CreatedDate" value="10/15/2010 12:01:48
PM" />
  <field name="ModifiedDate" id="ModifiedDate" value="10/15/2010 12:01:46
PM" />
  <field name="CustomerVaultID" id="CustomerVaultID" value="1742923032"
/>
  <field name="import_id" id="import_id" value="" />
  <field name="created_by" id="created_by" value="" />
  <field name="modified_by" id="modified_by" value="" />
  <field name="selected_currency" id="selected_currency" value="CAD" />
  </record>
```

Sample Credit Card Item:

```
- <record>
  <field name="DpPaymentMethodID" id="DpPaymentMethodID" value="392" />
  <field name="donor_id" id="donor_id" value="21245" />
  <field name="IsDefault" id="IsDefault" value="True" />
  <field name="AccountType" id="AccountType" value="MasterCard" />
  <field name="dpPaymentMethodTypeID" id="dpPaymentMethodTypeID"
value="creditcard" />
  <field name="CardNumberLastFour" id="CardNumberLastFour"
value="1xxxxxxxxxx2345" />
  <field name="CardExpirationDate" id="CardExpirationDate" value="0614"
/>
```

```
  <field name="BankAccountNumberLastFour" id="BankAccountNumberLastFour"
value="" />
  <field name="NameOnAccount" id="NameOnAccount" value="" />
  <field name="CreatedDate" id="CreatedDate" value="10/14/2010 1:11:30
PM" />
  <field name="ModifiedDate" id="ModifiedDate" value="10/14/2010 1:11:29
PM" />
  <field name="CustomerVaultID" id="CustomerVaultID" value="1234567890"
/>
  <field name="import_id" id="import_id" value="" />
  <field name="created_by" id="created_by" value="Diane Warner" />
  <field name="modified_by" id="modified_by" value="" />
  <field name="selected_currency" id="selected_currency" value="USD" />
  </record>
```

5. Supplemental Information

Gift Aid Program

The Gift Aid program is a tax incentive program used in the United Kingdom to encourage people to give to registered charities. DonorPerfect Online provides these Gift Aid fields.

Table	Field	Notes
DP	GIFT_AID_ELIGIBLE	'Y' or 'NO' to indicate whether this donor is gift aid eligible
DPUDF	GA_TITLE	Per title field in gift aid charity donation
DPUDF	GA_ADDRESS	Text box for foreign (non-UK) address
DPGIFT	GIFT_AID_ELIGIBLE_G	'Y' or 'N' to indicate whether the gift is gift aid eligible
DPGIFT	GIFT_AID_DATE	Date when gift was submitted to HM Revenue & Customs
DPGIFT	GIFT_AID_AMT	Portion of gift eligible for gift aid tax relief
DPOTHERINFO	GA_START_DT	Start date of gift aid declaration
DPOTHERINFO	GA_END_DT	End date of gift aid declaration
DPOTHERINFO	GA_DECL_ACTIVE	'Y' or 'N' to indicate whether the gift aid declaration is active

If you are not sure what the United Kingdom Gift Aid program is or how it works, please consult the United Kingdom Government site for more information.

Soft Credits

To create a soft credit gift, create that gift with RECORD_TYPE='S' and use the GLINK field to connect this to the standard gift (record_type='G') by its GIFT_ID number .

Here's an example: Donor_id 1336 actually gave $10, but we recognize that they gave because of donor_id 640 so we create the soft credit back to donor_id 640. Here's a SELECT showing the relevant fields:

```
<result>
<record>
<field name="donor_id" id="donor_id" value="1336"/>
<field name="gift_id" id="gift_id" value="12857"/>
```

```
<field name="amount" id="amount" value="10"/>
<field name="record_type" id="record_type" value="G"/>
<field name="glink" id="glink" value=""/>
</record>

<record>
<field name="donor_id" id="donor_id" value="640"/>
<field name="gift_id" id="gift_id" value="12858"/>
<field name="amount" id="amount" value="10"/>
<field name="record_type" id="record_type" value="S"/>
<field name="glink" id="glink" value="12857"/>
</record>
</result>
```

Pledge Notes

When you create a pledge in DonorPerfect Online to represent a recurring gift, you create a parent gift that represents the overall commitment – e.g; Dave Smith pledges to give $40 every month.   If Dave also has opted to give his first $40 gift when he set up the pledge, this $40 gift is a second gift separate from the parent pledge – but - associated with the parent pledge.

So in this scenario,

- a $40 per month pledge gift is created with a RECORD_TYPE field value of 'P' for pledge.

- The initial $40 gift Dave gives when he's making the pledge comes in as a second gift with RECORD_TYPE = 'G' for gift. Also, the PLINK field of this gift gets the GIFT_ID value of the parent pledge so that it this gift is linked to this specific pledge

- Also, to allow DonorPerfect to use its' EFT transactions functionality so that subsequent pledge payments can be run from within the DPO user interface, the parent pledge gift must have the DPGIFTUDF.EFT field set to 'Y'. This is done using the **dp_save_udf_xml** API call.

- Also, make sure you populate the @vault_id field with the Vault ID number returned by SafeSave.

- Note: The EFT field is shown as the Monthly Giving field in the user interface on the Pledge screen:

    o dp_save_udf_xml&params={parent pledge gift_id},EFT,'C',Y,null,null,'API User'

- DonorPerfect will also need an associated Payment Method to process pledge payments. Payment Method(s) are displayed under the Accounts tab in the DonorPerfect screen interface.

    o Payment methods are set up using the dp_PaymentMethod_insert API call

    o To use this API call, you will first need to create a Customer Vault ID in your SafeSave account using the SafeSave API

> o Creating the payment method with it's associated
> customer vault ID information in DonorPerfect allows
> all subsequent pledge payments to be managed through
> the DonorPerfect user interface without the need for
> any subsequent API application coding. It's like
> telling DonorPerfect "Here's a pledge, here's how much
> it's for, how often to collect payments and a payment
> method you can use for the payments."

Gift Adjustments

You can apply gift adjustments using the API if your application is handling refunds.

In the user interface, Gift Adjustments are done using the Adjust Gift screen:

 If you are using the API to do this, you will

- make adjustments to the **Original Gift** (gift_id 12885 in this example)
- create a new **Adjustment Gift** (gift_id 12891 in this example)
- create a new **Resulting Gift** (gift_id 12890 in this example)

Your gift_id numbers will, of course, be different than the ones shown here:

Original Gift:

- In this example, the original gift_id number is 12885
- Use the dp_savegift API call to set the @record type value of A

- o This will automatically update the @modified_date to todays date and the @modified_by field to the @user_id field value you specify.
    - o When you are using dp_savegift API call you will want to first use a SELECT query to retrieve all the values in this gift that you are 'updating' to the existing values
    - o When dp_savegift is used to update a gift rather than to create a new one, use the existing gift gift_id value in the @gift_id field instead of zero
- This gift shows original amount (29.95)

Adjustment Gift

- Use dp_savegift to create this new Adjustment Gift
- The new Adjustment gift gift_id number in this example is 12891
- Set todays date as the gift_date
- Record type in this case of H
- Set the amount is just the adjustment amount (-10)
- Include all the field data from the original (e.g.; GL Code, Solicit Code, Thank-You Letter, etc.)
- Set the @glink field to the gift_id number of the Original Gift

Resulting Gift

- Use dp_savegift to create this new Resulting Gift
- The new Resulting gift gift_id number in this example is 12890
- This gift gets the Gift Memo (gift_narrative) note <u>appended</u> to the end of any existing gift_narrative field contents from the original gift so you will need to retrieve the Original Gift gift_narrative field contents.
- Include all the field data from the original (e.g.; GL Code, Solicit Code, Thank-You Letter, etc.)
- Set the amount of this gift as the New Amount (19.95)
- Set the record_type to G unless the adjustment took the new amount to $0.00 in which case you set this gift to have record_type of C (or J if the original gift is a split gift)
- Set the @alink field to the gift_id number of the Original gift
- **Please Note**: If the original gift included additional field data (e.g.; financial field data or User Defined Field data), the Resulting gift should be set with the same values

Example retrieved results:

```
<result>

<record>

This is the ORIGINAL gift:

<record>
```

```
<field name="donor_id" id="donor_id" value="640"/>

<field name="gift_id" id="gift_id" value="12885"/>

<field name="record_type" id="record_type" value="A"/>

<field name="gift_date" id="gift_date" value="1/10/2018"/>

<field name="amount" id="amount" value="29.95"/>

<field name="created_date" id="created_date" value="1/10/2018"/>

<field name="created_by" id="created_by" value="My App Name"/>

<field name="modified_date" id="modified_date" value="4/5/2018"/>

<field name="modified_by" id="modified_by" value="My App Name"/>

<field name="gift_narrative" id="gift_narrative" value="xxx"/>

<field name="alink" id="alink" value=""/>

<field name="glink" id="glink" value=""/>

</record>
```

This is the ADJUSTMENT gift:

```
<record>

<field name="donor_id" id="donor_id" value="640"/>

<field name="gift_id" id="gift_id" value="12891"/>

<field name="record_type" id="record_type" value="H"/>

<field name="gift_date" id="gift_date" value="4/5/2018"/>

<field name="amount" id="amount" value="-10"/>

<field name="created_date" id="created_date" value="4/5/2018"/>

<field name="created_by" id="created_by" value=" My App Name"/>

<field name="modified_date" id="modified_date" value=""/>

<field name="modified_by" id="modified_by" value=""/>

<field name="gift_narrative" id="gift_narrative" value="xxx"/>

<field name="alink" id="alink" value=""/>

<field name="glink" id="glink" value="12885"/>

</record>
```

This is the RESULTING gift:

```
<record>

<field name="donor_id" id="donor_id" value="640"/>

<field name="gift_id" id="gift_id" value="12890"/>

<field name="record_type" id="record_type" value="G"/>

<field name="gift_date" id="gift_date" value="1/10/2018"/>

<field name="amount" id="amount" value="19.95"/>

<field name="created_date" id="created_date" value="4/5/2018"/>
```

```
<field name="created_by" id="created_by" value=" My App Name"/>

<field name="modified_date" id="modified_date" value=""/>

<field name="modified_by" id="modified_by" value=""/>

<field name="gift_narrative" id="gift_narrative" value="xxx This gift was adjusted by
-$10.00 from $29.95 to new amount $19.95"/>

<field name="alink" id="alink" value="12885"/>

<field name="glink" id="glink" value=""/>

</record>

</result>
```

The result of your gift adjustment will look like this in the user interface:

Created_date and modified_date fields

Although these fields are not accessible directly via the API, these fields are set in the following way:

@created_by / @modified_by

- When you set the @user_id field,
  - this value is placed in the @created_by if it is a new entry (e.g.; a new donor or gift)
  - this value is placed in the @modified_by you are using a predefined procedure (e.g.; dp_savedonor, dp_savegift) to modify an existing entry

@created_date / @modified_date

- If you are creating a new entry, the @created_date is set to {today}
- If you are modifying an existing entry, the @modified_date is set to {today}

Please note: Date fields are set as 'MM/DD/YYYY' only. Setting time values is not supported.

Tributes API Calls

Here's a quick primer on what each of the tribute related API calls does:

API Call	Function
Dp_tribAnon_MyTribSummary	This API call retrieves a list of all existing tributes. There is a switch in the call to exclude inactive tributes if desired.
Dp_tribAnon_Search	Allows you to search for tributes by name
Dp_tribAnon_Create	Use this API call to create a new tribute
Dp_tribAnon_AssocTribsToGift	Associates a specified gift to an individual tribute or to a comma separated list of tributes
dp_tribAnon_SaveTribRecipient	Use this call to specify the donor ID of the person who will be notified of a tribute gift. This API call causes the tribute box for the specified tribute to appear in the tribute section of the gift owners gift screen.
Dp_tribNotif_Save	This API call completes the notification process and adds a notification gift to the person being notified.

Unicode Support

DonorPerfect systems configured for Unicode text field handling store text data in UCS-2 / UTF-16 character encoding format.

If your API based application needs to send Unicode characters - e.g.; for foreign language text strings that involve special characters (e.g.; Vietnamese, Polish, Chinese, etc.) to DonorPerfect, there are two things you will need to do:

1. Contact your DonorPerfect account representative and request that your DonorPerfect system be set up to support Unicode text field handling.

2. Once the above has been completed, you can access any text fields that may include foreign language characters by inserting the capital letter 'N' after the = sign and before the quoted text string in your text_field_name='text string' API calls.

```
Example 1 :  ... text_field=N'Phúc cho những người nghèo về tinh thần, vì họ
             là vương quốc thiên đàng.'

Example 2 (using named parameters syntax):

             ... @text_field=N'Phúc cho những người nghèo về tinh thần, vì họ
             là vương quốc thiên đàng.'
```

IMPORTANT NOTES:

1.   You CAN use the modified syntax (FIELD_NAME=N'foreign language text here') to send non-Unicode text to DonorPerfect systems that are not set up for Unicode field support.

2.   This modified syntax works nicely with the common API Predefined Procedure calls e.g.; dp_savedonor, dp_savegift, dp_savecontact, dp_saveother, etc.

3.   The modified =N'xxx' syntax does not currently support the sending of Unicode text in the dp_save_udf_xml API call. The API call still works BUT the text that gets received will just show question marks where any Unicode characters were in the original text.

      •   So, 3rd party applications that need to write Unicode text to any User Defined Fields (UDFs), can **not** send Unicode text with the modified =N'xxx' syntax using the dp_save_udf_xml API call

      •   There is a temporary work-around for this. Please contact the DonorPerfect API Desk if you need more info on this workaround.

6. Support and Implementation Services

How to contact DPO API Support:

>Mark Warren – XML API Consultant
>
>Email: mwarren@softerware.com (Fastest path for resolution.)
>
>Phone: 1-855-896-5100
>
>- Normal Working Hours: Monday - Friday, 8:30 a.m. to 5:00 p.m. EST.
>
>- The acknowledgement response time for API support calls is not guaranteed and is independent of any other support guidelines. Calls are answered in the order they are received and under normal circumstances, it's reasonable to expect a response within one business day. For emergencies, send an additional email to support@donorperfect.com.

Your purchase of the API/XML service allows you access to our support staff and we can offer the following type of services at no additional charge:

- Assistance with API Call syntax.

- Recommendations or suggestions to accomplish your task.

- Error message explanations

- Restoring API service due to our own outages.

Provision of the DPO API by SofterWare includes:

- The DPO API toolkit (commands)

- The DPO API documentation

- API support as described above

API Support does not include assistance or consulting services on applications developed by the client organization or an authorized third party integrator.   These remain the sole responsibility of the client organization.

As a result, additional chargeable services not covered above include:

- Creating, Reviewing or Testing API calls on behalf of the client.

- Debugging API Code created by the client or the client's authorized third party.

- Creating or editing website code to utilize the DPO API.

- Creating or editing any customer specific API documentation

Please contact your Account Manager for any additional questions regarding our support for the DPO API.