Convio Web Services API Reference

Contents

Overview

Convio web services are designed with two primary goals in mind:

1. Give clients and partners the ability to implement data synchronization with external systems such as donor databases.
2. Provide external applications with flexible access to Convio data

Please read the topics under Getting Started first. The User Guide has more detailed information about building applications to implement data synchronization and retrieval.

See the API Reference for detailed information about web service operations and for a dictionary of supported data objects.

Getting Started

Understanding Convio Web Services

Convio Web Services provide clients and partners with a standard interface for building client applications that retrieve and synchronize data with the Convio server.

Convio Web Services (CWS) follow the SOAP standard, in which all messages between the client and server follow a particular XML format. In addition, all supported operations are described in a machine-readable document written in the Web Services Description Language (WSDL). The combination of SOAP and WSDL makes it easy to create Convio-aware applications with most popular development environments, including Java and .NET.

CWS is one component of Convio's Open API portfolio, and is intended primarily for implementing standalone applications such as data integration agents. Other components of the Open API portfolio follow the ReST style of operation, which is easier to work with in simpler browser-based applications.

Enabling Web Services

Contact Convio Support to have Convio Web Services enabled on your site.

Convio Support will activate CWS and provide you the endpoint URL for accessing the WSDL. They can also assist you with site configuration as described in the next section, *System Configuration*.

Once CWS is enabled and properly configured, you can use the *web services console* to verify that it is working correctly.

System Configuration

Convio Web Services API Configuration

To configure your Convio system for use with Convio Web Services:

1. Log in to Convio as a system administrator.
2. Go to Setup on the menu, and choose Site Options.
3. On the Site Options screen, click the Open API Configuration tab.

A dashboard screen will open showing the configuration status of the various Convio API options. Under the section heading Configure API Keys click on the link Edit API Keys. Enter a value in the Convio API Key field and click the Finish button.

Next, under the section heading Configure Convio Web Services click on the link Edit Web Service configuration. On this screen the system administrator must specify the IP range of servers allowed to access Convio Web Services. The administrator can also quickly generate a default partition of the Convio database for access through Convio Web Services, restricting access to data on only those constituents who are not site administrators.

Options on this page include:

- **Convio Web Services URI** -- Set this field if the Convio Web Services process runs on a separate system from the Convio Website. Otherwise, leave this field blank.
- **Convio Web Services IP White List** -- Designates servers which may access Convio Web Services using *CIDR* notation. To permit access from single IP address, use a 32-bit mask value (e.g. 192.168.1.100/32). To permit access from all addresses on a class-C subnet, use a 24-bit mask (e.g. 192.168.1.0/24).
- **Add Current IP Address to list** -- Optionally, use this button to automatically add the IP address of the system on which you are running the Convio Admin application to the White List.

- **Remove Selected IP Range** -- Use this button to remove servers from the IP White List by clicking the server's IP range in the list to highlight it, then clicking the button.
- **Create Default Partition** -- You must create a partition for Convio Web Services. Click the **Create** button to automatically create a default database partition that may be accessed via Convio Web Services. The default partition includes data for all constituents except for administrator accounts. If you prefer to use a custom partition, you must set up a custom partition by clicking **Data Management** on the main menu, selecting **Import/Export**, and choosing the **Partition Management** tab.

Note: IP restriction for Convio Web Services access includes the local host. If you wish to permit web services console access to CWS running on the same host, you may need to include the local host address (i.e. 127.0.0.1/32) in the IP white list.

Web Services Console

The web services console is a simple browser-faced interface for experimenting with Convio Web Services. The console allows you to send test messages to the server by completing a simple web form. The console displays both the request and response messages so you can see what is happening.

Each Convio site has its own console. To determine the URL of your site's console, follow these steps:

1. Look at the URL where you normally log in to Convio as a site administrator.
2. Identify your cluster by looking at the host name in the admin URL. If you log in to `secure2.convio.net`, you are on cluster 2; if you log in to `secure3.convio.net`, you are on cluster 3.
3. Identify your site short name by looking at the first part of the URL path after the host name. For example, if you normally log in to `https://secure3.convio.net/myorg/admin/AdminLogin` , then your site short name is `myorg` .
4. Open a new browser window and enter the URL of your site's console as `https://webservices.cluster[2 or 3].convio.net/1.0/[site short code]/console`

The console consists of a series of tabs, one for each supported operation. Before trying anything else, you must always log in first. You can then click on any other tab and try other operations.

Accessing the WSDL

Once web services have been enabled and configured, the next implementation step is to access the WSDL, which describes both the data schema as well as the full set of available operations. The WSDL is primarily intended for use by client applications that use the metadata to automatically generate interfaces to the web services. Since the WSDL is simply an XML document, it can also be viewed in a browser for verification purposes.

Each site has its own WSDL. You can access the WSDL for your site by following these steps:

1. Look at the URL where normally log in to Convio as a site administrator.
2. Identify your cluster by looking at the hostname in the admin URL. If you log in to `secure2.convio.net`, you are on cluster 2; if you log in to `secure3.convio.net`, you are on cluster 3.
3. Identify your site short name by looking at the first part of the URL path after the hostname. For example, if you normally log in to `https://secure3.convio.net/myorg/admin/AdminLogin` , then your site short name is `myorg` .
4. Open a new browser window and enter the URL of your site WSDL as `https://webservices.cluster[2 or 3].convio.net/1.0/[site short code]/wsdl`
5. Your browser should display a long XML document.

The WSDL for a particular site is unique in at least three ways:

1. The Constituent element in the WSDL schema reflects the current database configuration for a site. The WSDL only includes constituent fields that are enabled and visible to admins. This includes all custom fields.
2. The Donation element in the WSDL schema has any custom fields that have been added to donation forms.
3. The WSDL defines an "endpoint" URL for accessing the actual web services. Each site has its own endpoint URL.

Using the Partner WSDL

A generic form of the WSDL is also available for partners wishing to develop applications that may be used by multiple clients. The generic WSDL lists all standard constituent fields, regardless of whether they are enabled for a site or not. It also has an `any` element at the end of the Constituent type as a placeholder for custom fields.

The partner version of the WSDL can be obtained by appending the query string `?type=partner` to the URL, e.g. `http://webservices.cluster2.convio.net/1.0/mysite/wsdl?type=partner` .

It is important to note that there is no generic SOAP endpoint address associated with the partner WSDL. Applications using the partner WSDL will have to override the endpoint URL at runtime, specifying the appropriate site-specific URL.

soapUI Sample Project

You can use the soapUI web services test tool to test requests and responses through Convio Web Services. This sample project file can be loaded in soapUI to demonstrate several different types of request.

The sample project file covers the following requests: Login request, Query request, GetServerTime, Start Sync, Create request, Update request.

Setting up and Configuring soapUI

1. Download the project file: *Convio_Web_Services-soapui-project.xml*
2. Download soapUI from *http://www.soapui.org* There are both a Windows installer and a platform-independent binary available for download.
3. Run the Windows installer or unzip the binaries to set up soapUI on your system.
4. Start SoapUI.
5. From the **File** menu, select **Import Project**. In the dialog, browse to the location where you saved the project and select the file: "Convio_Web_Services-soapui-project.xml"
6. After the project is loaded, expand the project tree (click on the 'plus' sign next to "Convio_Web_Services").
7. Double click on the **SoapBinding**. This will load the "SoapBinding" window.
8. On the **Service Endpoints** tab set the **Endpoint** for your Convio Web Services server: (ex: http://cms200.convio.com:9090/1.0/bvt107)
   - Click the **Assign** button to pop up the "Assign Endpoint" dialog.
   - Select "All Requests and Test Requests" from the dropdown list.
   - Click **OK**.
9. On the **WSDL Content** tab set the WSDL location, by clicking the **Reload** button in the dialog's tool bar (the square button containing a "cycle" icon). In the resulting dialog, enter the **Definition URL**. This can be the URL of the WSDL on your CWS server (http://cms200.convio.com:9090/1.0/bvt107/wsdl) or the local or of a local saved copy of the WSDL (wsdl.xml).

Note: The current project file uses an endpoint of: http://cms200.convio.com:9090/1.0/bvt107 and a WSDL location: http://cms200.convio.com:9090/1.0/bvt107/wsdl. You must change these to the appropriate endpoint and WSDL URIs for your Convio Web Services server.

Running the tests

Once the Project file is loaded and the endpoint and WSDL definition are configured you can run the suites to illustrate each sample request. Each sample is broken into "suites" (a green check box next to the name defines each suite).

1. Double click on the suite name (ex: "Query Example")
2. Click the Green "Start" arrow at the top to execute all the commands in the suite.

3. The icons should turn green, and the progress bar at the top should be green and read "Finished." If any of the commands are colored red, there was a failure, you need to check the configuration or the status of the web services server.

After the commands have executed, double click on a request command (it should have a green icon now) to view the XML request and response message (for example: "Login Request"). The SOAP command request window will show the full request message on the left and the response message on the right.

Generating Client Code

A wide variety of development tools will automatically examine a WSDL and generate a corresponding programming interface that hides the mundane details of sending and receiving messages and converting data to and from XML format.

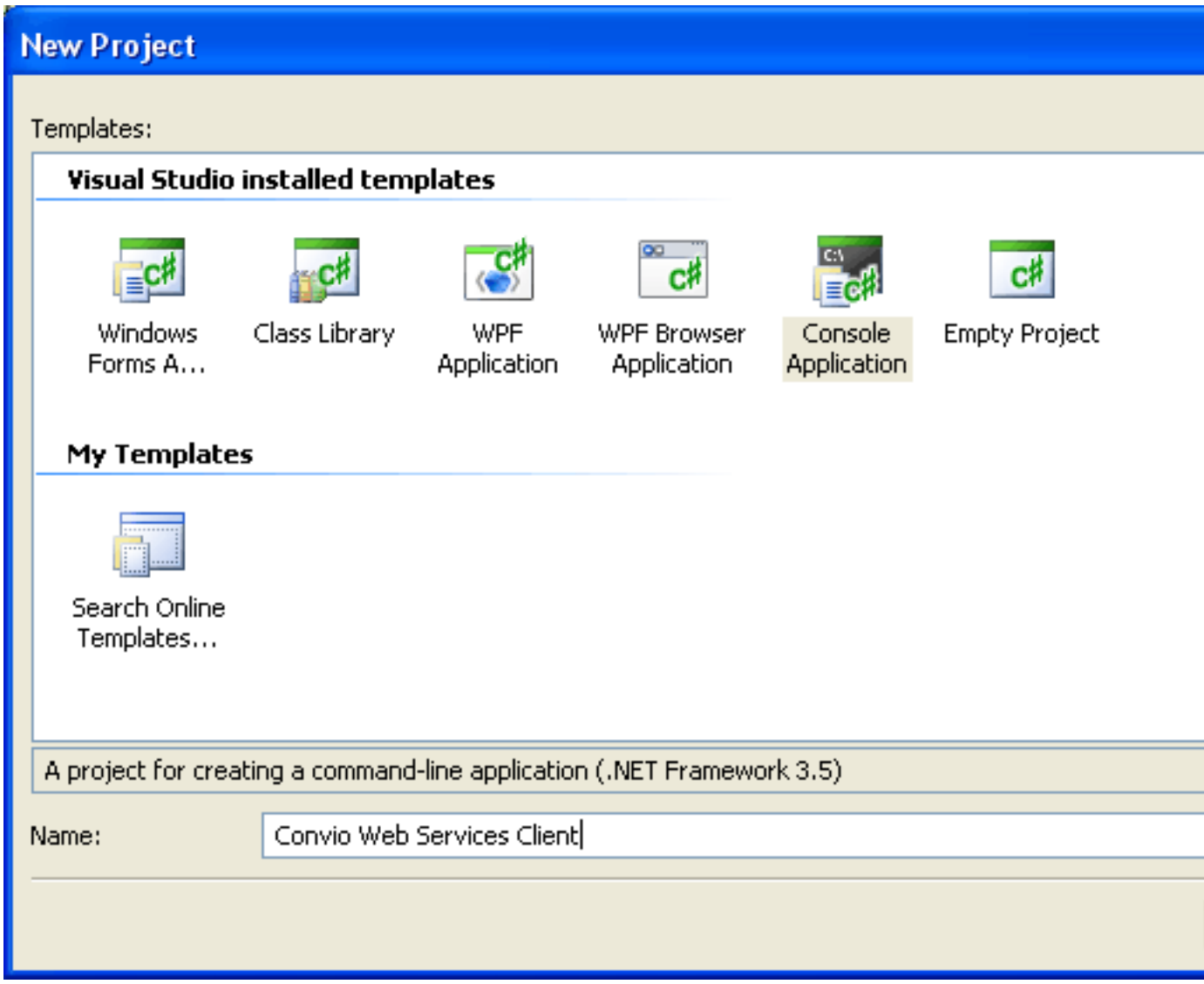
Convio Web Services is tested for compliance with the Basic Profile of the Web Services Interoperability Organization (WS-I). Most popular tools should therefore be able to consume the Convio WSDL and execute web services operations without compatibility problems.

This section contains snippets for generating clients using two of the most common platforms for web services client development, C# using Visual Studio, and Java using Apache/Axis. Convio has also developed more complete sample clients using these two platforms.

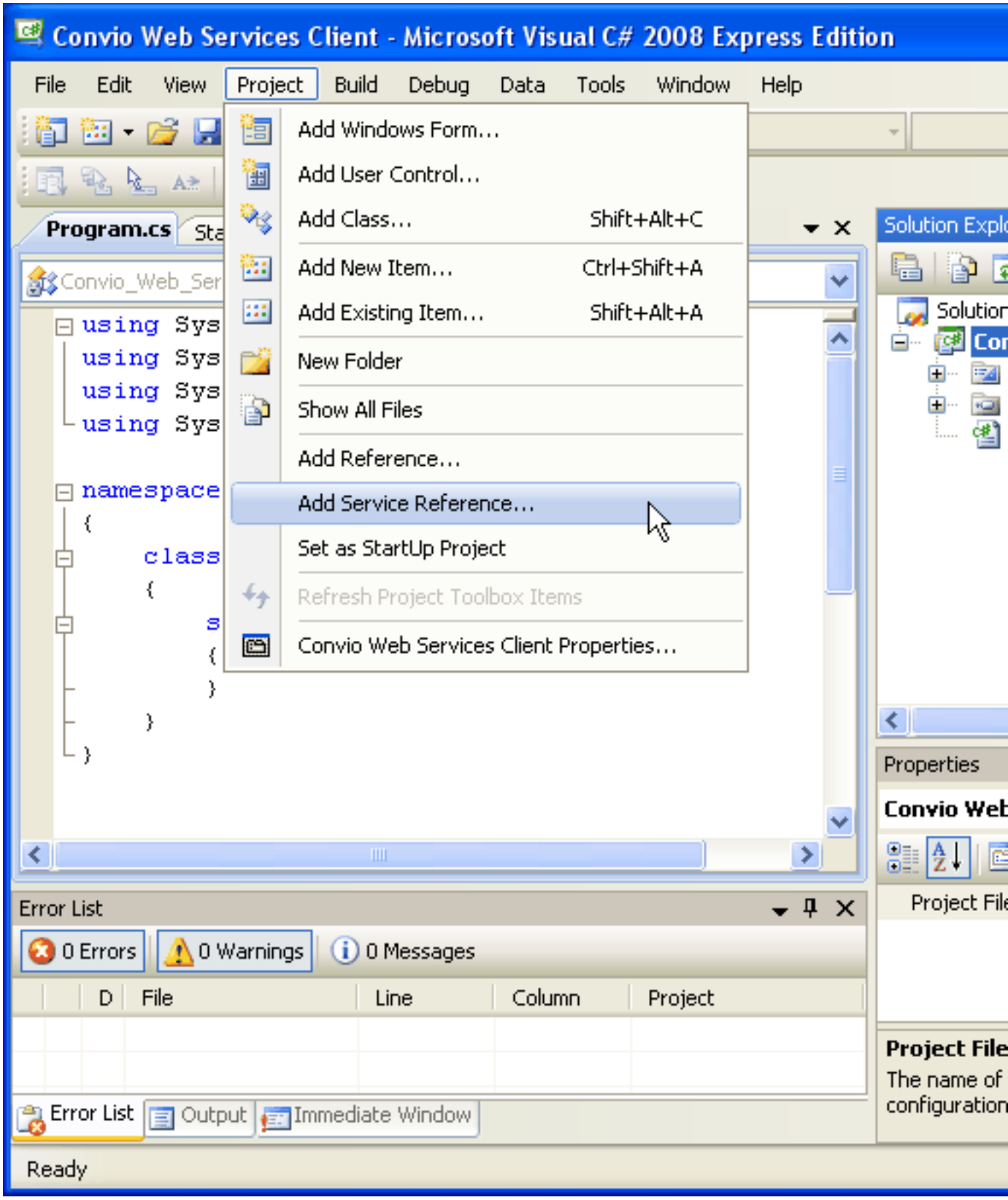
Getting Started with Visual Studio

You can use Microsoft Visual Studio to implement a client application that uses the Convio Web Services. This topic outlines the process for Visual C#.

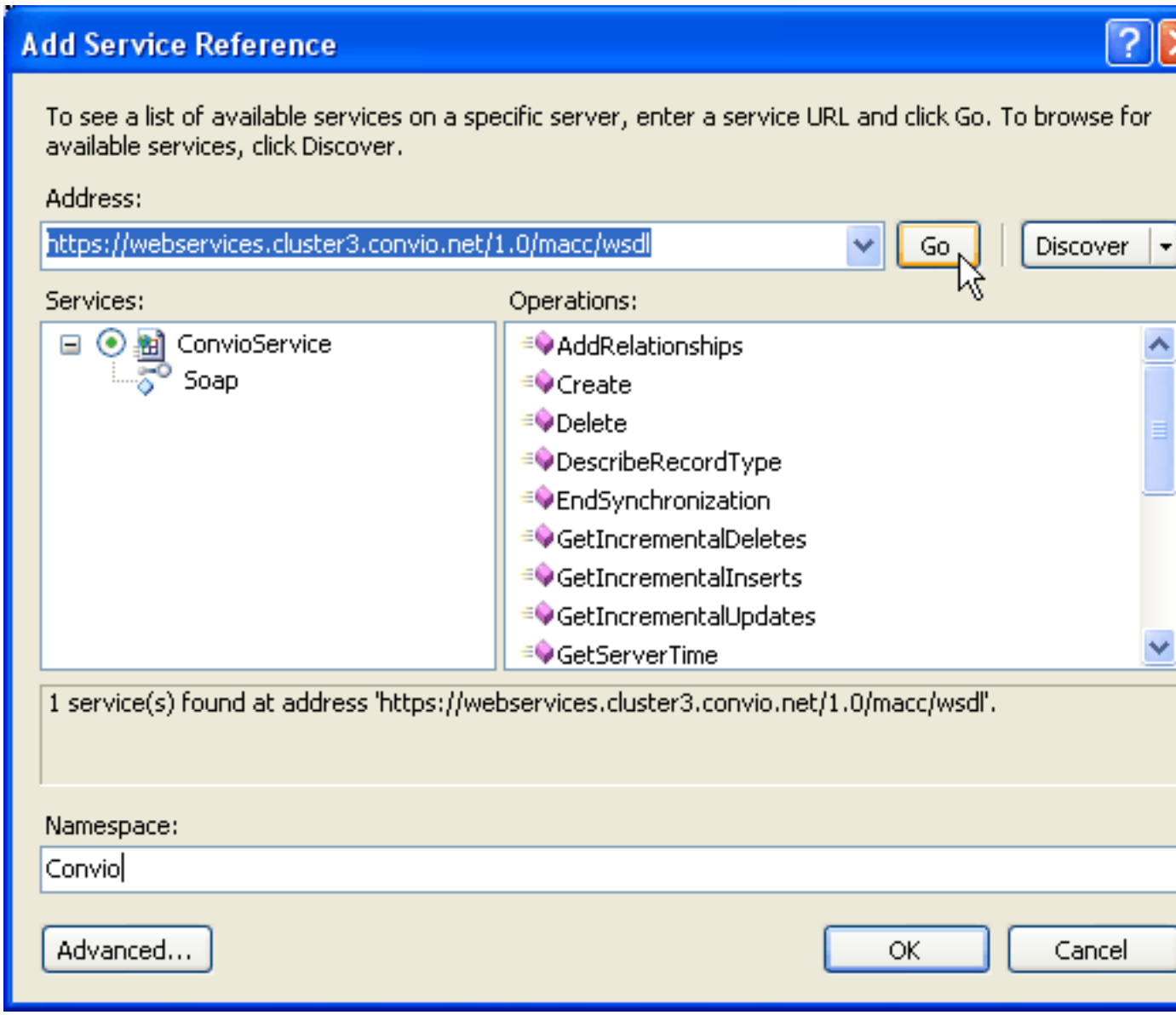
1. Open any edition of Microsoft Visual C# 2008. Note that you can implement the client using the Express Edition, which is available as a free download from Microsoft.
2. Choose File -> New Project from the application menu.
3. Choose the Console Application template in the New Project dialog and click OK to create the project:



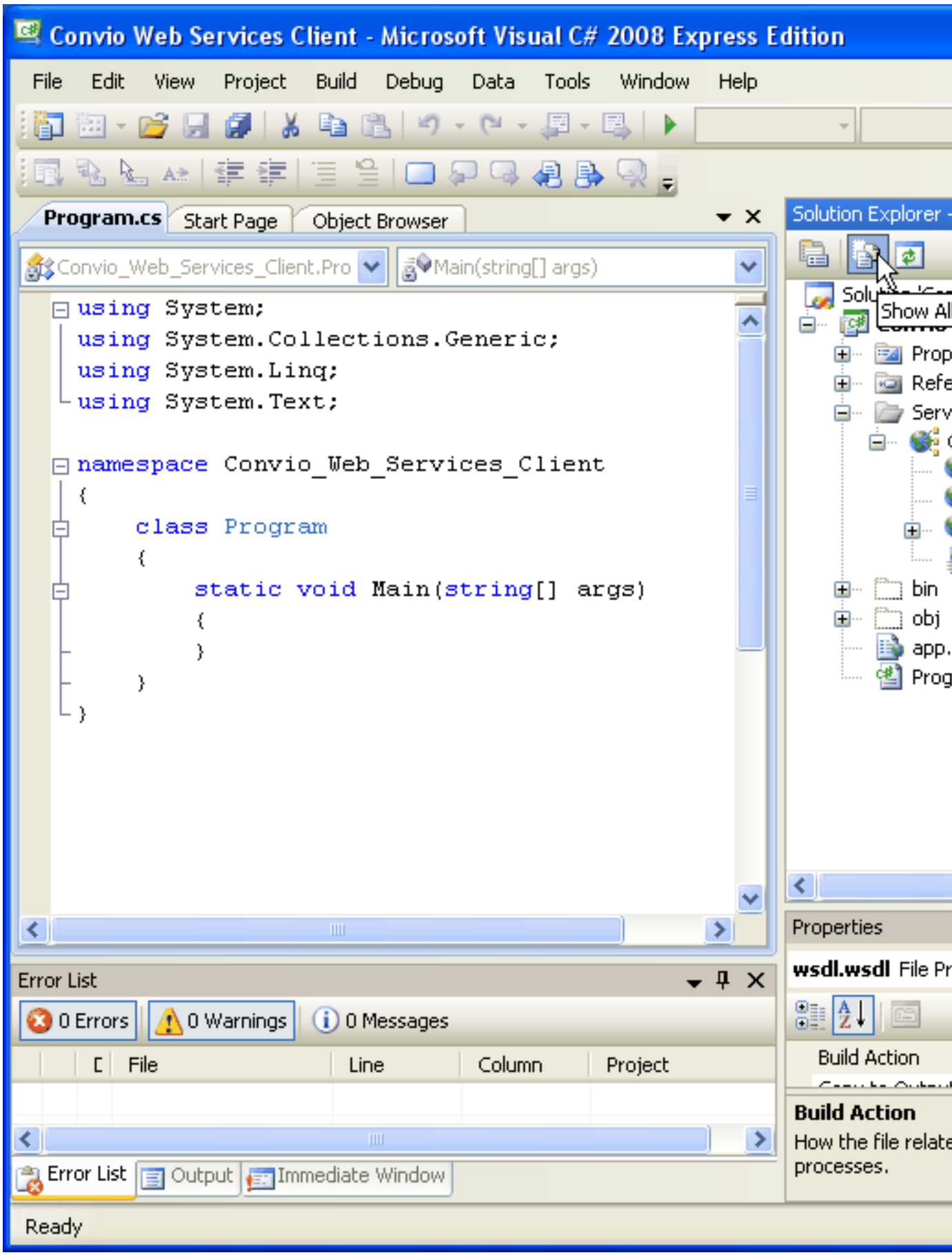
4. Choose Project-> Add Service Reference from the application menu:



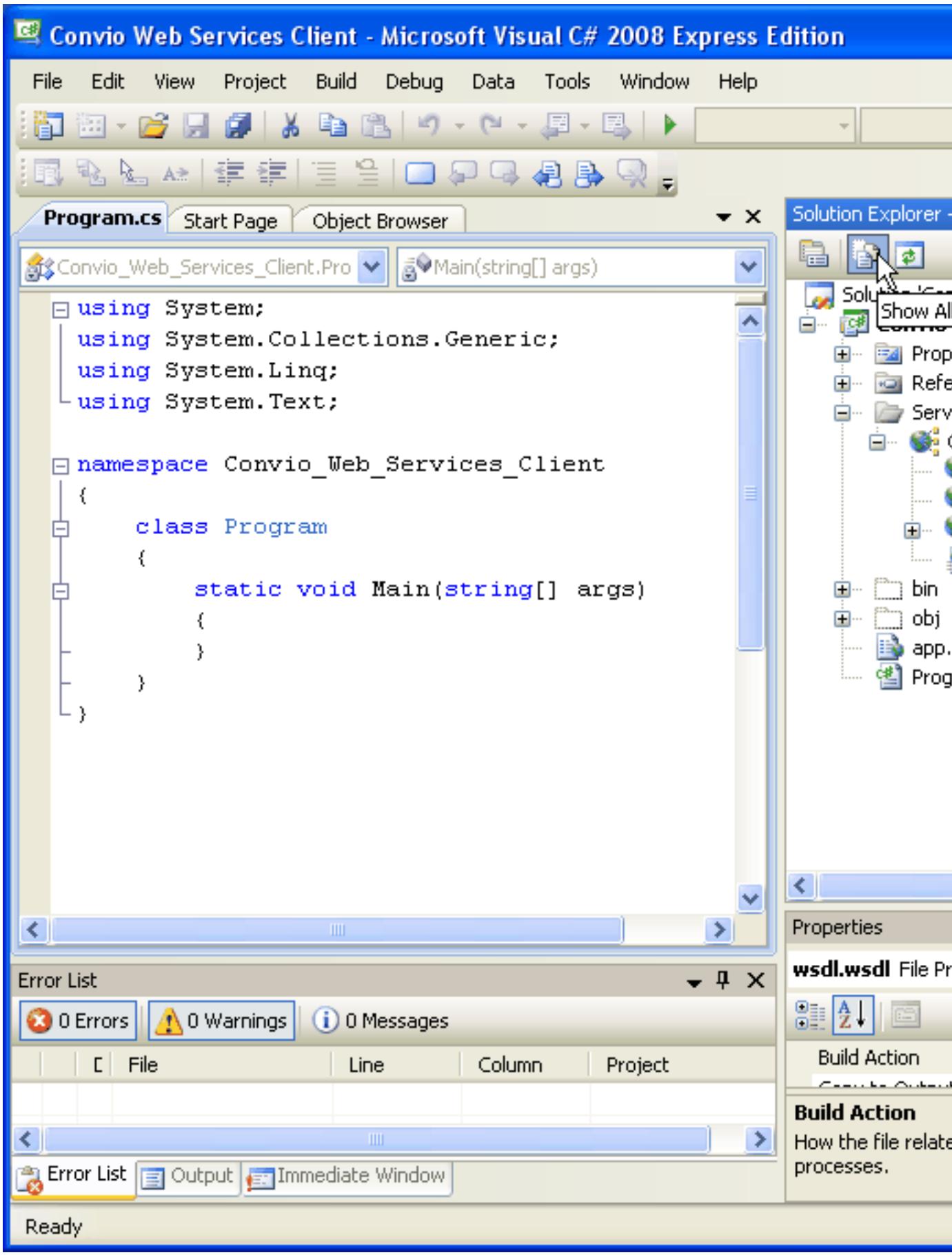
5. Enter the URL of your Convio WSDL in the Address field and click the Go button.
6. Enter Convio in the Namespace field:



7. Click OK to generate the classes representing the web service operations and data objects. The Convio service reference should appear in the Solution Explorer pane. To view the WSDL and check for problems, first click on the Show All Files button in the Solution Explorer toolbar:


Convio Web Services Client - Microsoft Visual C# 2008 Express Edition
File Edit View Project Build Debug Data Tools Window Help
Program.cs Start Page Object Browser
Convio_Web_Services_Client.Pro
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Convio_Web_Services_Client
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer
Show All
bin
obj
Properties
wsdl.wsdl File Pr
Build Action
Build Action
How the file relate
processes.
Error List
0 Errors
0 Warnings
0 Messages
File
Line
Column
Project
Error List
Output
Immediate Window
Ready

8. Then right-click on `wsdl.wsdl` and choose Open. Any errors or warnings associated with the WSDL will appear in the Error List pane. You may have to toggle display of the messages by clicking the buttons in the Error List toolbar:

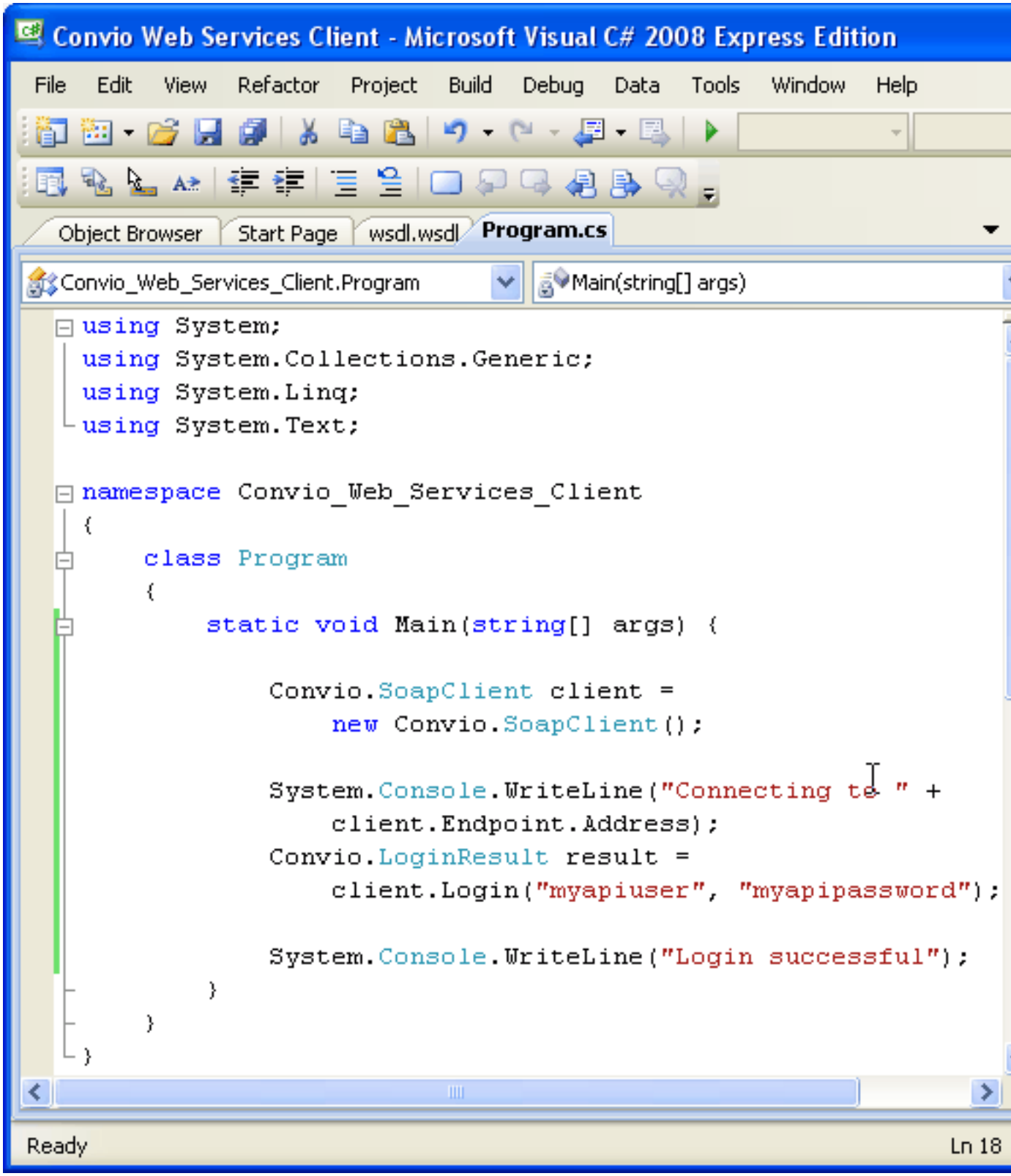

Convio Web Services Client - Microsoft Visual C# 2008 Express Edition
File Edit View Project Build Debug Data Tools Window Help
Program.cs Start Page Object Browser
Convio_Web_Services_Client.Pro
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace Convio_Web_Services_Client
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer
Show All
Properties
wsdl.wsdl File Pr
Build Action
Build Action
How the file relate
processes.
Error List
0 Errors
0 Warnings
0 Messages
File
Line
Column
Project
Error List
Output
Immediate Window
Ready

Note that warnings can be safely ignored.

9. Edit Program.cs and enter the following test program:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Convio.SoapClient client = new Convio.SoapClient();

            System.Console.WriteLine("Connecting to " +
 client.Endpoint.Address);

            Convio.LoginResult result = client.Login("myapiuser",
 "myapipassword");

            System.Console.WriteLine("Login successful");
        }
    }
}
```

The finished program should look like this:



10. Press F5 to run the test program.

Specifying the Endpoint URL at Runtime

Site-specific applications can simply use the static endpoint URL specified by the WSDL. Generic applications that can access multiple sites have to specify the endpoint URL at runtime. This can be done simply by overriding the `Endpoint.Address` property:

```
Convio.SoapClient client = new Convio.SoapClient();
client.Endpoint.Address = new EndpointAddress("http://
webservices.cluster2.convio.net/1.0/mysite");
```

Setting optional, nullable attributes

The Convio WSDL defines all fields as both nullable and optional (`minOccurs="0"`). This makes it easier to send and receive partial records, including only relevant fields, rather than requiring both the client and server to specify full records for every operation.

To distinguish between null elements that should be specified as `nil` and elements that should simply be omitted, the standard XML serializer generated by the .NET framework adds a special flag to each non-String attribute. This flag must be set to `true` for the attribute to be serialized as part of a SOAP message:

```
Convio.Constituent constituent = new Convio.Constituent();
constituent.ExternalYtdGiftAmountSpecified = true; // this flag must be set to
 serialize this attribute in the SOAP message.
constituent.ExternalYtdGiftAmount = 500.00;
```

Getting Started with Apache Axis

You can use Apache Axis2/Java to implement a client application that uses the Convio Web Services. This topic outlines the process.

1. Install a Java Development Kit, version 5 or newer. You should be able to execute `javac -version` from a shell or command prompt.
2. Install Apache Axis2, and Apache Ant. You should be able to execute `ant -version` and `WSDL2Java` from a shell or command prompt.
3. Execute the following to generate the client databindings using ADB. (Replace SITE with your Convio shortname, and ensure you have the correct cluster.) `WSDL2Java -uri http://webservices.cluster[2 or 3].convio.net/1.0/SITE/wsdl -p com.convio.webservice.client -d adb -ss -g -uw -s`
4. Copy the following test program into src/com/convio/webservice/client/DataSyncClient.java, changing SITE, cluster, myapiuser, and myapipassword to reflect your actual settings.

```
package com.convio.webservice.client;

import com.convio.soap.*;
import com.convio.webservice.client.*;

public class DataSyncClient {
    public static void main(String[] args) throws Exception {
        String endpointUrl = "http://webservices.cluster[2 or
 3].convio.net/1.0/SITE";
        ConvioServiceStub stub = new ConvioServiceStub(endpointUrl);

        System.out.println("Connecting to " + endpointUrl);

        LoginResult loginRes = stub.Login("myapiuser", "myapipassword");

        System.out.println("Login successful: " + loginRes.getSessionId());
    }
}
```

5. Build the client by typing `ant jar.client`.
6. Run the client by typing `axis2 -cp build\lib\ConvioService-test-client.jar com.convio.webservice.client.DataSyncClient`.

Using cURL to post SOAP messages

cURL is a popular library for submitting HTTP requests. It may be used from the command line or via bindings to various scripting languages such as PHP. SOAP client libraries such as nuSOAP use cURL as the underlying HTTP client.

By default, cURL checks the content length of a POST before making a request. If the content length is above a minimum size, it sends an initial, empty request with an 'Expect: 100-continue' header, to test whether the server will accept the full content. Upon receiving a 100 Continue response from the server, cURL will post the full content.

Convio does not support this behavior, so you have to configure cURL to disable it:

```
curl_setopt( $curl_handle, CURLOPT_HTTPHEADER, array('Expect:') );
```

Failure to disable this behavior will result in an error that the server closed the connection with data remaining.

C# Sample Client

This sample console application was written in C# using Visual Studio 2008 Express Edition. It demonstrates the following:

1. Developing with a Service Reference generated from the Convio WSDL.
2. Storing and using configuration properties in `app.config`.
3. Logging request and response messages to a file for debugging.
4. Calling the Login operation and retrieving the SessionId.
5. Starting a synchronization session.
6. Uploading constituent records from an XML data file.
7. Downloading constituent records to an XML data file.
8. Querying for cross-references and building a lookup table.
9. Downloading donation records to an XML data file, resolving the cross-reference for each record.

To try out the sample:

1. Download this *zip file* to your local computer and extract it.
2. Start Visual Studio and choose Open Project... from the File menu. Navigate to the extracted directory and open `DataSyncClient.sln`.
3. In the Solution Explorer pane, expand the Service References node.
4. Delete the existing 'Convio' service reference, and add a new one by right-clicking on Service Reference: 'Add Service Reference'.
5. In the Address field specify the URL corresponding to your site's WSDL.
6. Back in the Solution Explorer, double-click on `app.config` and edit the user name, password, and log file location.
7. Press F5 to run the sample in Debug mode.

Java Sample Client

This sample console application was written in Java using Apache's Axis2. It demonstrates the following:

1. Authentication
2. Query for available partitions
3. Start a synchronization session with and without an explicit time interval
4. Show catch of synchronization fault due to previously failed synchronization
5. Upload new constituents from a CSV file
6. Upload updated constituents from a CSV file

7. Remove interests and groups from a CSV file
8. Upload deleted constituents from a CSV file
9. Download new constituents
10. Download updated constituents
11. Download deleted constituents
12. Query for cross-reference values and build cross-reference table in memory
13. Download transactions
14. End synchronization session, saving session ID and info to a file

To try out the sample:

1. Download this *zip file* to your local computer and extract it.
2. Ensure you have a working JDK installed, version 5 or newer. (You should be able to execute `javac -version` from a shell or command prompt.)
3. Edit `DataSyncClient.properties` with a text editor of your choice, and point to your Convio Web Services URL with your credentials.
4. Execute `bin/ant run` (Unix) or `bin\ant run` (Windows) from a shell or command prompt in the javaDataSyncClient folder.

Next Steps

Once you have verified the basic ability to access the web services and implement a simple test program, you are ready to proceed with your implementation. Consult the User Guide for more details about using the web services to retrieve and synchronize data with Convio.

User Guide

Access and Authentication

Access to Convio Web Services is restricted in the following ways:

1. The server only accepts secure (SSL) connections.
2. The server only accepts connections from known IP addresses that are on a web services "white list."
3. The user name in a login request must be associated with an active constituent that has administrator status
4. The constituent associated with the user name must be a member of an administrator security group that has the "Use Convio APIs" permission.
5. The password in a login request must be valid for the constituent.

A successful login returns a `SessionId` token in the response. All other types of requests are authenticated by included a valid session token in the header of the request. For example:

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>e5d267b7c09382fd3c8b97aa75e61ad5ecc3e38d:JSESSIONID=abcu_AmzmlABt8amexGfs:100
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <GetServerTime xmlns='urn:soap.convio.com'>
    </GetServerTime>
  </soap:Body>
</soap:Envelope>
```

A session token is valid for up to 30 minutes of inactivity.

Convio Object Schema

Most Convio web service operations involve submitting or receiving data records from the Convio server. Common record types include Constituent, Campaign and Donation.

Each record type includes a particular set of fields. For example, a Constituent record has fields for name, home address and email, while a Donation record has fields for transaction type, billing address and amount.

Because many record types have a large number of fields, all data retrieval operations allow you to specify which fields you want the server to include in the response. This allows you to avoid transferring large amounts of data that you don't plan to consume.

The WSDL defines the full list of supported fields for each record type. Every field is marked as optional when the record is represented in XML format, to facilitate the transfer and processing of partial records containing only fields of interest.

See the reference section of this documentation for a full description of each record type.

Transaction

A transaction record represents any constituent interaction with the Convio system, either directly through a web site or indirectly through an API that involve a financial payment (i.e. donations, TeamRaiser registrations, product orders).

Each type of transaction record includes a different set of fields. For example, a product order may contain details about which items the constituent ordered, while a donation may include information about recurring or installment payments.

Convio Web Services only supports unidirectional synchronization of transaction records: when a constituent engages in a online activity through the Convio system, the web services can be used to download information about the activity into the client system. However, changes in the opposite direction are not supported: the client system cannot use the web services to create, update or delete transaction records in the Convio database.

Product Orders

Product orders have the most complex structure of all the financial transaction types. Each order is associated with purchaser and payment information, and can contain one or more shipments, and each shipment can contain one or more items. The resulting XML looks like this:

```
       <Record xsi:type="ens:ProductOrder">
       <ens:OrderId>100000064</ens:OrderId>
       <ens:Payment>
         <ens:Amount>1000.00</ens:Amount>
       </ens:Payment>
       <ens:Purchaser>
         <ens:ConsId>100000053</ens:ConsId>
         <ens:ConsName>
           <ens:FirstName>John</ens:FirstName>
           <ens:LastName>Doe</ens:LastName>
         </ens:ConsName>
       </ens:Purchaser>
       <ens:ShipTo>
         <ens:Item>
           <ens:ProductId>100000055</ens:ProductId>
           <!-- additional item attributes... -->
         </ens:Item>
       </ens:ShipTo>
       <ens:ShipTo>
         <ens:Item>
           <ens:ProductId>100000055</ens:ProductId>
           <!-- additional item attributes... -->
         </ens:Item>
       </ens:ShipTo>
     </Record>
```

It is important to note that each `Item` element represents a single unit, rather than a line item in a typical invoice. If someone places an order for 5 of the same product, then the corresponding record will contain 5 Item elements, rather than a single Item with a quantity of 5. In many cases this may seem repetitive, but it is necessary to accommodate the fact that each unit may have distinct shipping address, shipping option, fulfillment and tribute details.

In most cases client applications will want to summarize or "roll up" product orders into a flatter structure that reflects a particular product configuration. For example, say that your site typically only sells two products, and you just want to get a count of each product sold. The following C# code illustrates how you would summarize each order:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Configuration;

namespace DataSyncClient
{
    /**
```

```
    * Downloads new or updated donations to a local XML file.
    */
   class ProductOrderDownloader
   {
       private static String PRODUCT_ONE_ID = "1234";
       private static String PRODUCT_TWO_ID = "1235";

       private Convio.SoapClient client;
       private Convio.Session session;

       public ProductOrderDownloader(Convio.SoapClient client, Convio.Session
session)
       {
           this.client = client;
           this.session = session;
       }

       /**
        * Makes a series of requests to retrieve the full set of order of the
specified type.
        *
        * Sums line items by product ID
        */
       public void download(DownloadType type)
       {
           String typeLabel = type.ToString().ToLower();
           String partitionId =
ConfigurationSettings.AppSettings["partitionId"];

           String[] fields = { "OrderId", "Payment.Amount",
"Purchaser.ConsId",
                               "Purchaser.ConsName", "Purchaser.MemberId",
                               "ShipTo.Item.ProductId"};
           int page = 1;
           while (true)
           {
               Convio.Record[] records = (type == DownloadType.Inserts) ?
                   client.GetIncrementalInserts(session, partitionId,
"ProductOrder", page.ToString(), "100", fields) :
                   client.GetIncrementalUpdates(session, partitionId,
"ProductOrder", page.ToString(), "100", fields);
               if (records == null || records.Length == 0)
               {
                   if (page == 1)
                   {
                       System.Console.WriteLine("No Product Order " +
typeLabel + " to download");
                   }
                   break;
               }
               int start = (page - 1) * 100 + 1;
               System.Console.WriteLine("Downloading Product Order " +
typeLabel + " " +
                   start + " to " + (start + records.Length));

               foreach (Convio.Record record in records)
               {
                   Convio.ProductOrder order = (Convio.ProductOrder)record;

                   /**
                    * Each Item element represents a single unit.  Iterate
over all the elements
                    * to count how many were ordered of each product;
                    **/
```

```
                    int productOneCount = 0, productTwoCount = 0, otherCount =
0;

                    foreach (Convio.ShipTo ship in order.ShipTo)
                    {
                        foreach (Convio.Item item in ship.Item)
                        {
                            String productId = item.ProductId;
                            if (PRODUCT_ONE_ID.Equals(productId))
                            {
                                productOneCount++;
                            }
                            else if (PRODUCT_TWO_ID.Equals(productId))
                            {
                                productTwoCount++;
                            }
                            else
                            {
                                otherCount++;
                            }
                        }
                    }

                    // process the order, including product counts...
                }

                page++;
            }

        }
    }
}
```

Convio Query Language

Ad hoc data retrieval is one of the key target applications for Convio Web Services. Clients can use the *Query* operation to access a wide range of data from a Convio site.

See the *query syntax* topic for more details on writing queries.

See the *pagination* topic to learn how to retrieve large query results across multiple requests.

Convio Query Language

Grammar

The Convio Query Language supports a basic subset of the SQL SELECT grammar:

```
'select' field-list 'from' record-object ['where' criteria]

  field-list ::= '*' | FIELD [ ',' FIELD ]...

  criteria ::= criterion [('and' | 'or') criterion]...

  criterion ::=
  FIELD comparison-operator operand
  |  '(' criteria ')'

  comparison-operator ::= '<>' | '>=' | '>' | '<=' | '<' | '='
```

```
 operand ::= FIELD | STRING_LITERAL | NUMBER | DATETIME | DATE

 DIGIT ::= '0'..'9'

 TIMEZONE ::= ('Z' | 'z') | (('+' | '-') DIGIT DIGIT ':' DIGIT DIGIT)

 DATE ::= DIGIT DIGIT DIGIT DIGIT '-' DIGIT DIGIT '-' DIGIT DIGIT TIMEZONE

 DATETIME ::= DATE ('T' | 't') DIGIT DIGIT ':' DIGIT DIGIT ':' DIGIT DIGIT
TIMEZONE

 NUMBER ::= DIGIT... | DIGIT... ('.' DIGIT...) | '.' DIGIT...

 FIELD ::= ('a'..'z' | 'A'..'Z') ('a'..'z' | 'A'..'Z' | '0'..'9' | '_' |
'.')+

 QUOTED_CHARACTER ::= ( ~( '\'' | '\\' ) ) | '\\' ( ( '\'' | '\\' ) )

 STRING_LITERAL ::= '\'' ( QUOTED_CHARACTER )... '\''
```

The field and record type names in CQL do not have to map directly to schema column and table names. They may be aliases or be derived. For example, the SQL statement for a constituent query may involve a join to retrieve status information (such as whether a client is a donor).

The possible set of `record-object` types includes all sub-types of `Record` as defined by the Convio WSDL. You can use the `DescribeRecordType` operation to determine whether a particular record type supports the `Query` operation, as well as which fields may be used as criteria in the where clause of a CQL query.

The query parser will validate requested queries and convert them to an appropriate SQL SELECT statement for executing against the Convio database.

Date Criteria

Date criteria must be specified in ISO 8601 format with an appropriate time zone offset. For example:

```
 select UserName, ConsName from Constituent where ModifyDate >=
2009-05-04T16:45:29+00:00
```

Note that the date should not be quoted.

Query Examples

```

select UserName, ConsName.FirstName, ConsName.LastName from Constituent where
 ConsName.FirstName = 'John'

select ConsName from Constituent where GroupId = 456

select Id,Name from Group where TypeId = 789
```

Note that the query language supports grouping of related fields in the select clause. For example, clients can request first and last name specifically, or they just request the Name object and get back a nested element containing title, first, middle, last, etc.

Request Conventions

Query Conventions

Convio records can be quite large: the constituent record for an organization can have hundreds of fields depending on configuration, and only small subset may be needed for any particular sync or query.

In the schema for Convio Web Services, all fields are defined as optional (minOccurs="0"). This allows requests and responses to contain partial records (only a subset of the full record) and still be considered valid.

Likewise, by default, Query operations typically return a commonly-used subset of the fields for any given object. A request for a ConsName object returns only the most frequently used fields: FirstName and LastName. These two fields are "wildcard" fields for the ConsName object. You can find out which fields are included in wildcards by using the DescribeRecordType operation, which includes an 'IsWildcard' flag for each field.

For Query and GetIncrementalInserts, GetIncrementalUpdates, and GetIncrementalDeletes operations, nested fields are specified using dot notation, i.e. to return the constituent's middle name, specify 'ConsName.MiddleName' as a Field argument.

Update Conventions

Update operations follow a similar convention. Only fields specified in an Update request which contain values, or which specify the attribute `xsi:nil="true"` are updated. Fields missing from the request, or not containing a value, are not changed.

To delete the value of a given field on Update, you must specify the field in the Update request and explicitly set it null.

For example:

SOAP Request	Result
<MiddleName xsi:nil="true"/>	middle_name set to null
<MiddleName />	middle_name unchanged
<MiddleName>John</MiddleName>	middle_name changed to "John"

Pagination

Many Convio web service operations involve the upload or download of potentially large numbers of data records. For example, a middleware client may add several thousand new constituent records to Convio in a single synchronization session. A client may also download thousands of new members from Convio that signed up for an online campaign.

To lower the risk of network timeouts and to better balance server utilization, Convio limits the size of both requests and responses to a relatively small number of records. No more than fifty records may be uploaded (created, updated or deleted) in a single request. In the other direction, no more than two hundred records may be downloaded at a time.

Clients must break up data transfers into pieces that fit these constraints. For uploads, the client simply makes as many requests as necessary until all the records have been transferred. For example, if a client wants to add 170 new constituents to Convio, it needs to make 4 sequential requests (the first three containing 50 records each and the last one containing 20 records).

For downloads, the client needs to keep track of which records it has already downloaded. It does this by including a page number and page size in each request. The client starts each download by requesting page 1, and then increments the page number until no more records are returned.

Data Synchronization

Data synchronization is the process of ensuring that data is consistent in two or more different information systems. Convio Web Services are specifically designed to help organizations synchronize data in the Convio system with another system. In most cases the other system is a donor database that tracks "offline" activity by organization members and donors, conducted by mail, over the phone or in person.

Who Should Read This

Data Synchronization as supported by Convio Web Services is a structured batch process for exchanging data between Convio and an external database. CWS exposes functions that allow middleware developers to keep core

profile data about online constituents synchronized between the two systems and to export online transaction records from Convio to the external database.

This section is intended to provide software developers at Convio Partners and Clients the information they need to plan for, develop, and test data synchronization between Convio and an external database using CWS.

Client System

A client system represents a donor database or other repository that an organization wants to keep synchronized with the Convio database. Changes to a constituent record may be initiated in either the client system or in Convio. When a constituent record is created, modified or deleted in the client system, the client must notify Convio to update its database accordingly. Conversely, when a constituent joins or updates their online profile in Convio, the client system must be notified of the change.

Constituent Partition

A partition defines the set of constituents in the Convio database that are targeted for synchronization with a client system. In the most common scenario, you want to synchronize all constituents, but in some cases you may want to restrict the exchange of constituent information to a specific subset. For example, if you are synchronizing a donor database with Convio, you may only want to synchronize constituents who have a mailing address or who have made at least one donation.

The Convio system automatically tracks partition membership according to the criteria you specify. If you define a partition that includes all constituents, then the Convio system will add constituents immediately when they register. If you define a partition that includes only a subset of all constituents, the Convio system will check each time a constituent record is updated, and add or remove the constituent from the partition as appropriate.

Creating and managing partitions

Partition management is part of the Convio administration interface:

1. Choose **Import/Export** from the **Data Management** menu
2. Click on the **Partition Management** tab

This page lists available partitions and allows you to edit, copy and delete them, as well as manage their membership.

Determining available partitions

All of the Convio Web Service operations require a partition ID. Most sites will have a single partition whose ID you can set as a static property of your middleware implementation. You can use the *Web Services Console* to query for the ID of each available partition:

1. Log in to the console
2. Click on the Query tab.
3. Enter the following query:

```
select PartitionId,Name from Partition
```

4. Copy the relevant `PartitionId` from the query response

Middleware applications can also perform this query and obtain the partition ID at runtime if desired.

Middleware

In most cases the client system itself does not communicate with the Convio Web Services directly. Instead, a separate *middleware* application typically sits between the two systems. The middleware is responsible for extracting records from the client system, transforming the records into web service messages, and submitting the messages to the Convio server. The middleware is also responsible for receiving the response to each message and transforming it into a useful format for updating the client system.

Middleware applications come in two distinct flavors:

1. A dedicated *connector* developed specifically for managing the interaction between Convio Web Services and your particular donor database. Connectors must be implemented by a developer using a technology such as Java or .NET.
2. A generic integration tool that allows non-developers to define the synchronization process using a visual interface, without writing any code.

Understanding the CWS Schema

Nested Data Structures

Top-level schema types in Convio Web Services are all derived from the `Record` type. The primary top-level type is the `Constituent` record. Top-level `Record` types such as `Constituent`, `Donation` or `TeamRaiserGift` directly support various operations. Sub-types such as `Name`, `Address`, or `AnnualTransactionSummary`are only accessible through their parent type; they do not directly support operations.

Convio Web Services types are hierarchical, and in code, access to sub-elements is specified using dot notation. For example, `Constituent.Name` contains the fields `Constituent.Name.FirstName` and `Constituent.Name.LastName`. Likewise the `Patricipant` field of a `TeamRaiserRegistration` type record is a `Constituent` type; specify the participant's surname as: `TeamRaiserRegistration.Participant.Name.LastName`.

Configurable Constituent Fields

Like snowflakes, no two `Constituent` record structures are exactly alike. The fields of a `Consituent` record can be customized for individual clients, so some fields listed in the documentation may not be present in a specific client implementation, and custom fields may be included which are not necessarily documented. If you are unsure about the meaning or use of specific custom fields, you should contact the Client system administrator or Convio support for assistance.

Using the DescribeRecordType Operation

The `DescribeRecordType` operation allows you to inspect any CWS type in the WSDL to determine the properties of that type. This operation returns the list of the operations that the type supports and the list of fields in the type. `Field` properties include:

- `Name` - the name of the field
- `Writable` - whether or not the field may be modified on `Create` or `Update` operations
- `Custom` - whether or not the field name is an alias for a custom field in the Convio database
- `Nillable` - whether or not the field value may be Null
- `Multiple` - whether or not the field may return multiple instances
- `Type` - the field's data type (may be either a simple type or another CWS schema type)
- `MaxLength` - the maximum number of characters supported for text fields (longer values will be truncated)
- `IsCriterion` - whether or not the field may be used as search criteria in the "where" clause of a Query operation.
- `IsWildcard` - whether or not the field is returned to a query that does not specifically request it when the parent type is requested
- `Option` - a list of valid values for an enumeration

Wildcard and Nillable Fields

The number of fields in `Constituent` records can be quite large. Therefore, by default, CWS returns only a commonly-requested subset of fields by default if their parent type is requested in an operation. These fields are designated as "wildcard" fields, indicated by the`IsWildcard` property of each `Field` entry in the `DescribeRecordType` response for any given type. To return non-wildcard field values, simply specify the field in the request.

Likewise to optimize performance of `Update` operations, only those fields which contain values, or which specify the attribute `xsi:nil="true"` are updated. Some middleware frameworks may include empty field elements in a request if values are not specified for those fields. To guard against accidental data loss, CWS requires that you explicitly set a field to null to clear it on `Update`.

Attempting to set the value of a non-writable field, or clear a non-nillable field, will result in an error.

Bi-directional Synchronization, One-way Synchronization, and Query

The `Constituent` record is currently the only object in CWS that supports bi-directional synchronization (i.e. it supports `Create`, `Update` and `Delete` operations into the Convio database); other top-level schema types that expose transactions (e.g. `Donations`) support only one-way (download) synchronization. These support the synchronization operations `GetIncrementalInserts`, `GetIncrementalUpdates`, `GetIncrementalDeletes`, but not `Create`, `Update` and `Delete`. Other top-level types that provide metadata, summary information, or additional data about a `Constituent` or other synchronized type (e.g. `TeamRaiserParticipationType`) are accessed through `Query` operations. Such objects are not directly associated with events in time (i.e. a registration or donation) and so do not support the synchronization operations.

Mapping Fields between Convio and the External Database

Not all fields of a given Convio constituent record will correspond to fields in the external system database, and vice versa. The first step in defining any data synchronization is identifying the fields in each database which should be synchronized, and creating a formal mapping between them. You can do this in a spreadsheet, mapping each field of Convio data to its corresponding field in the external database and specifying whether the field is to be downloaded, uploaded, or both. This map is a key project artifact that should be kept current throughout the development and test process.

If you are using a middleware framework to develop the synchronization process, the framework will specify the map file format, likely in an XML schema.

Defining Policies

In addition to defining the synchronization data map, you will need to develop the synchronization policies, or business rules, that will allow the automatic synchronization to take appropriate action in response to synchronization errors, duplicate entries, or conflicts between the two databases.

Auditing and Logging

Understand what data you will need to log and audit both for the intitial synchronization and for the ongoing operation of your data sync process.

Your middleware should log, at minimum, each synchronization session ID, the start and end times for the session, the primary keys of all records uploaded and downloaded during the session, and each record operation status.

If practical, you should also consider logging all web service requests and responses between your middleware and Convio Web Services.

Convio Sync Audit Data Retention Policy

Convio Web Services operations generate an audit trail of records that were uploaded and downloaded as part of each sync session. Sync sessions may vary greatly in size, from zero records to millions (the latter would occur for a large site doing a complete sync of most or all of their constituent records).

Convio will retain this audit data for either the last 20 sync sessions, or for all sync sessions initiated in the last three months, whichever is greater. Convio will purge older sync session audit data on a nightly basis.

Synchronization Failure

You should determine in advance the process to be followed for different types of synchronization failure. Generally speaking, it should be safe to re-run a failed synchronization if the synchronization process guards against against duplicate entries. However,your process and policies should guard against skipping or reordering synchronization sessions, as out-of-sequence operations on the two databases can omit data or introduce collisions.

Your policy and systems should designate which database is the database of record, and allow for "ab initio" re-synchronization of data from the master database to the duplicate database in the event that coordination of database state between the two databases is ever lost due to a synchronization failure.

Notification

Your system may need to implement policies for notifying you or a system administrator, for example by email, in the event of certain failures.

Error reporting

Convio web service operations return two general types of error codes:

1. A **message error** indicates that the server failed to process the message in its entirety, either because the message itself is invalid or because of a server problem. The application returns message errors as SOAP faults.
2. A **record error** indicates a failure to process a particular constituent record as part of a generally valid bulk update or create request. The server returns record errors as part of the normal response to a bulk request. It is the client's responsibility to check the status code returned for each record and ensure that it processed successfully.

Message errors

Any request may return one of the following general message errors:

MalformedMessageFault	Malformed XML (wrong encoding, tags missing or unbalanced, etc.)
InvalidMessageFault	Message violates the WSDL specification (invalid SOAP message or missing, out-of-sequence or invalid elements in the message header or body)
SessionFault	Missing or invalid session ID in the SOAP header
UnsupportedOperationFault	Invalid SOAP operation (one not specified in the WSDL)
ServerFault	Unexpected server error occurred

Individual operations may return additional specific errors:

LoginFault	Invalid username or password in the `login` request

The server returns all message errors as standard SOAP faults:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <soap:faultcode>MALFORMED_MESSAGE</soap:faultcode>

      <soap:faultstring>SOAP header element is missing</soap:faultstring>
      <soap:detail>
        <MalformedMessageFault xmlns="urn:soap.convio.com">
          <line>2</line>
          <column>1</column>

        </MalformedMessageFault>
      </soap:detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Note that the `faultcode` element specifies the type of error, with the corresponding information in the `detail` element.

Record errors

Record error codes are consistent with those returned by the constituent ReST API (except for the one pertaining to the "post" method):

Code	Number	Description
OK	0	Record processed successfully
SERVER_ERROR	1	Unexpected server error
TOO_MANY_RECORDS	5	Exceeded the maximum number of records for an upload operation
REQUIRED_FIELDS_MISSING	10	Cannot add or update the record because one or more required fields are missing
MATCH_EXISTING	11	Cannot add the record because it matches an existing record
DUPLICATE_USER_NAME	12	Cannot add or update the record because user name matches an existing one
DUPLICATE_EMAIL	13	Cannot add or update the record because email matches an existing one
MATCH_AMBIGUOUS	15	Cannot update the record because multiple records match the search criteria
RECORD_NOT_FOUND	16	Cannot update the record because no matching records were found
INVALID_FIELDS	18	Cannot add or update the record because it includes invalid fields

The server returns record errors as part of the normal response to an operation:

```
<?xml version='1.0' encoding='UTF-8'?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
         <createResponse xmlns="urn:soap.convio.com">
            <Result>
               <StatusCode>MATCH_EXISTING</StatusCode>
               <Message>Create failed: one or more existing records found that
 match.</Message>
               <PrimaryEmail>augustus9@yahoo.com</PrimaryEmail>
            </Result>
            <Result>
               <StatusCode>MATCH_EXISTING</StatusCode>
               <Message>Create failed: one or more existing records found that
 match.</Message>
               <PrimaryEmail>elvis9@gmail.com</PrimaryEmail>
            </Result>
            <Result>
               <StatusCode>MATCH_EXISTING</StatusCode>
               <Message>Create failed: one or more existing records found that
 match.</Message>
               <PrimaryEmail>python9@gmail.com</PrimaryEmail>
            </Result>
         </createResponse>
      </soap:Body>
```

```
    </soap:Envelope>
```

Synchronization Collision

Synchronization collisions can occur when the same record is created, updated or deleted in both the Convio and client system within the same time window. For example, the following scenario may occur:

1. A regularly scheduled daily synchronization session occurs at 1 AM. Convio database and donor database are now completely in sync.
2. At 10 AM a constituent logs into the Convio system and updates the e-mail address that is part of their online profile.
3. At 11 AM a staff person at the organization's office logs into the donor database and updates the street address of the same constituent.
4. The next regularly scheduled daily synchronization occurs at 1 AM that night. The middleware application extracts the updated record from the donor database and sends it to Convio. Convio Web Services receives the update message, but detects that the same record was modified in the Convio database during the time window of the current synchronization session. To avoid any loss of data, the web service rejects the update and responds that a collision occurred.

When a collision like this occurs, the middleware application needs to report it for manual resolution. A collision resolution tool would ideally show the two related records (one from the client system and one from the Convio system) side by side, and allow a staff person to choose which fields to retain from each one. Convio Web Services provides a mechanism for forcing an update to the Convio database (overriding collision detection) once the manual resolution process is complete.

Collision scenarios

The following scenarios will result in a record-level collision response from the server:

Insert duplicate collision

In this scenario, the web services client attempts to insert a record that appears to be a duplicate of an existing record in Convio. The server examines several field values to detect a potential duplicate, and reports a different error code for each one:

Field	Error Code	Scenario
MEMBER_KEY	DUPLICATE_MEMBER_KEY	Client attempts to insert a constituent record with a member key that is associated with an existing constituent record.
EMAIL	DUPLICATE_EMAIL	Client attempts to insert a constituent record with an email that is associated with an existing constituent record.
USER_NAME	DUPLICATE_USER_NAME	Client attempts to insert a constituent record with an email that is associated with an existing constituent record.

A collision on CONS_ID is not possible on insert because the server automatically generates a new CONS_ID for each inserted record. The server simply ignores any CONS_ID that a client may include with records submitted for insertion.

A collision may occur with a constituent record that is in the "pending approval" state.

The time window for the active synchronization does **not** affect insert collisions, since member key, email and user name must all be unique across the entire site.

A conflict on insert is **not** possible if a deleted record exists with the same member key, email or user name. Uniqueness is only checked for non-removed constituents.

Update and delete collisions

The next general class of collisions is when both a client and the server or some other client update or remove (mark as deleted) a record within the same synchronization period. Consider this scenario:

1. John Smith signs in to the MYORG site and updates his email address at 3:00 PM on April 8, 2008.
2. A volunteer in the MYORG office updates John Smith's record in their offline donor database at 4:00 PM on the same day.
3. At 3 AM that night, MYORG's integration server starts a new synchronization covering the previous day, from 12:00 AM on April 8 to 12:00 AM on April 9.
4. The integration server submits an update request on John Smith's record.
5. Convio Web Services notices that John Smith's record has been modified by some other agent during the synchronization time frame.
6. Convio Web Services does not update the record, since that would overwrite John Smith's update of his email address.
7. Convio Web Services responds with a COLLISION error code.

Several permutations of this scenario are possible:

1. Two web service clients update or remove the same record within the same time frame.
2. A web service client updates a record and a Convio admin removes the same record in the same time frame.
3. A web service client removes a record and Convio updates the same record in the same time frame.

In any of these cases, Convio Web Services will refuse to modify the record because it would potentially overwrite the changes of another agent.

A web services client can force an update by setting the `Force` parameter to `true` in an update request.

Resolving Duplicate Registrations

Convio has an optional feature for detecting and manually resolving potential duplicate registrations (when the same person signs up twice, perhaps with a slightly different profile). This feature is enabled by setting the PREDUP_MATCH_ENABLED site attribute to TRUE under Setup -> Site Options -> Data Sync Properties in the Convio administrator interface.

When duplicate detection is enabled, the system checks each new registration for a potential match with existing constituent records. If a potential match is found, the registration is queued for manual resolution by an administrator, who must compare the new and existing registration and decide whether they are different. The interface for resolving registrations is located under Data Management -> Import/Export -> Resolve New Registrations.

Potential matches are not added to constituent partitions, and therefore are never included when downloading new constituent records using the GetIncrementalInserts web service operation. After a potential match is resolved as being distinct from any existing record, it is added to any relevant partitions and made available for download.

If you wish to resolve duplicates outside of the Convio system, simply ensure that the PREDUP_MATCH_ENABLED site attribute is set to FALSE. This will prevent the system from flagging any new registration as a potential duplicate, and ensure that all new registrations (including potential duplicates) are returned by the GetIncrementalInserts web service operation. In this case, it is the responsibility of the client application to remove resolved duplicates from the Convio system.

Synchronization Scheduling

There are two basic scheduling options for any data synchronization process:

1. Real-time synchronization involves immediate notification of other systems whenever a record is added, modified or deleted in one of the systems being synchronized. For example, with real-time synchronization the Convio system would send a message to all client systems immediately whenever a person registered, updated their profile or made a donation online.
2. Batch synchronization involves a periodic exchange of all records, modified or deleted during a specific time window. For example, with batch synchronization the middleware application might make a nightly request for all records added, updated or deleted in the Convio database during the previous day (midnight to midnight). As part of the same operation, the middleware application would also submit to Convio all constituent records added, updated or deleted in the client system during that time.

Real-time synchronization can be difficult to set up and is prone to reliability problems, because it requires that each participating system be able to send messages to all others. If one system goes down or becomes unavailable due to network issues, the entire synchronization process may be thrown off.

Because of the logistic difficulties of implementing real-time synchronization, Convio Web Services are designed for batch synchronization only. The middleware application orchestrates the entire synchronization operation, initiating all requests to both the Convio system and to all participating client systems. If the Convio or client system goes down or becomes unavailable for any reason, the middleware application can simply postpone or interrupt the synchronization for a particular time period without running the risk of data loss or corruption.

Synchronization Session

A synchronization session is a single batch procedure, involving the full exchange of new, updated and deleted records between a client system and the Convio database for a particular time window in the (typically recent) past.

A complete synchronization session may include the following:

- Upload
  - create new constituents in Convio to match records that were added to the client system
  - update existing constituents in Convio to match changes made to the corresponding records in the client system
  - remove constituents from Convio that were removed from the client system
- Download
  - retrieve new constituents that were added to the Convio database (typically by online registration) during the session time window.
  - retrieve updates to existing constituent records in the Convio database (only those that existed prior to the start of the session time window).
  - retrieve constituents that were removed from the Convio system during the session time window.
  - retrieve new transactions made by existing constituents during the session time window
  - retrieve transactions that were updated during the session time window (only that that existed prior to the start of the session time window).
  - retrieve transactions that were deleted during the session time window.

The middleware is responsible for orchestrating the entire synchronization session. It initiates a series of web service requests to Convio to authenticate, start the session, upload records, download records, and finally end the session.

Convio assigns a unique numeric identifier to each synchronization session.

Establishing a Synchronization Session

Log in

The first step in every conversation with Convio Web Services is to authenticate as a Convio administrator with appropriate permissions. See Site Configuration for information on setting up API Administrator accounts. The middleware does this by sending a Login message:

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Login xmlns="urn:soap.convio.com">
      <UserName>user</UserName>
      <Password>password</Password>
    </Login>
  </soap:Body>
</soap:Envelope>
```

Convio responds to valid login requests with a session token:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <LoginResponse xmlns="urn:soap.convio.com">
         <Result>

 <SessionId>e96ce3db61a80642510225a783540f6dcb6e702d:JSESSIONID=abcMZ5dmWgZeidZtyWZ3r:100
SessionId>
         </Result>
      </LoginResponse>
   </soap:Body>
</soap:Envelope>
```

The caller must include this session token in all subsequent requests. The token is valid for 30 minutes, after which the middleware must establish a new session and obtain a new session token.

Start a synchronization session

Once the caller has successfully authenticated, it can start a synchronization session by making a `StartSynchronization` request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Session xmlns="urn:soap.convio.com">
      <SessionId>80c3bf74d37e7bdd8d5b25cf320bbfb2065dd798:0</SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <StartSynchronization xmlns="urn:soap.convio.com">
      <PartitionId>123</PartitionId>
    </StartSynchronization>
  </soap:Body>
</soap:Envelope>
```

Convio responds with the time window that will be used to select records for subsequent synchronization calls. By default, the time window is between the last synchronization session request and the current one:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <StartSynchronizationResponse xmlns="urn:soap.convio.com">
         <Result>
            <PartitionId>123</PartitionId>
            <Start>2008-12-01T18:13:44Z</Start>
            <End>2008-12-01T18:13:44Z</End>
            <SyncId>118</SyncId>
         </Result>
      </StartSynchronizationResponse>
   </soap:Body>
</soap:Envelope>
```

The last call to the `StartSynchronization` method determines the active synchronization window for a particular administrator and partition. Only one synchronization session may be active for an administrator and partition combination at any time. Note that the system tracks synchronization windows separately for each administrator and partition, so that multiple offline systems can synchronize with the same partition as long as they authenticate using different administrator accounts.

The `SyncId` is a unique ID for each synchronization session. You can use it later to retrieve a log of records updated or downloaded as part of this synchronization.

Synchronizing Constituent Data

Upload Constituent Record Changes to Convio

For a two-way synchronization, the next step is to use the `Create`, `Update` and `Delete` operations to upload changes from the external database since the last synchronization to Convio, for example:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>c55d8f07700b4aa8401e2b9c4d4262efa6da30d8:JSESSIONID=abcKuVJBroiHpeNByWZ3r
ns3:SessionId>
      </ns3:Session>
   </soapenv:Header>
   <soapenv:Body>
      <ns3:Create xmlns:ns3="urn:soap.convio.com">
         <ns3:PartitionId>123</ns3:PartitionId>
         <ns3:Record xmlns:ns1="urn:object.soap.convio.com" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:type="ns1:Constituent">
            <ns1:ConsName>
               <ns1:FirstName>Albus</ns1:FirstName>
               <ns1:LastName>Dumbledore</ns1:LastName>
            </ns1:ConsName>
            <ns1:UserName>albus</ns1:UserName>
            <ns1:PrimaryEmail>albus@hogwarts.edu</ns1:PrimaryEmail>
            <ns1:HomeAddress>
               <ns1:Street1/>
               <ns1:City>Hogsmeade</ns1:City>
               <ns1:State>CA</ns1:State>
               <ns1:Zip>94702</ns1:Zip>
               <ns1:Country>USA</ns1:Country>
            </ns1:HomeAddress>
         </ns3:Record>
      </ns3:Create>
   </soapenv:Body>
</soapenv:Envelope>
```

Each request can include up to 50 constituent records. Convio responds with a Result element for each record:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <CreateResponse xmlns="urn:soap.convio.com"
 xmlns:ens="urn:object.soap.convio.com" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <Result>
            <ResultCode>OK</ResultCode>
            <Message>New user added.</Message>
            <Record xsi:type="ens:Constituent">
               <ens:ConsId>1001482</ens:ConsId>
               <ens:PrimaryEmail>albus@hogwarts.edu</ens:PrimaryEmail>
            </Record>
         </Result>
      </CreateResponse>
   </soap:Body>
```

```
</soap:Envelope>
```

The result of `Create` operation includes the constituent ID (`ConsId`), the unique primary key that Convio has assigned to each newly created constituent. The result of the `Update` and `Delete` operations includes the `ConsId` of the existing record that the system matched to the request, if any.

The middleware should check the `ResultCode` for each record and take appropriate action if an error or collision has occurred.

The requests and responses for the `Update` and `Delete` operations have the same structure.

Download Constituent Record Changes from Convio

The next step in the conversation is to download new, updated and deleted constituent records from Convio:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>3868cf8ebb49e046a024e18ad8420ea931fe8b64:JSESSIONID=abc_g9cmWx9L0_dZyWZ3r
ns3:SessionId>
      </ns3:Session>
   </soapenv:Header>
   <soapenv:Body>
      <ns3:GetIncrementalInserts xmlns:ns3="urn:soap.convio.com">
         <ns3:PartitionId>123</ns3:PartitionId>
         <ns3:RecordType>Constituent</ns3:RecordType>
         <ns3:Page>1</ns3:Page>
         <ns3:PageSize>3</ns3:PageSize>
         <ns3:Field>ConsId</ns3:Field>
         <ns3:Field>ConsName</ns3:Field>
         <ns3:Field>UserName</ns3:Field>
         <ns3:Field>PrimaryEmail</ns3:Field>
         <ns3:Field>HomeAddress</ns3:Field>
      </ns3:GetIncrementalInserts>
   </soapenv:Body>
</soapenv:Envelope>
```

Convio Web Services will respond to this request with a set of constituents updated during the synchronization window by agents other than the caller (it may be another admin or the constituent herself):

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <GetIncrementalInsertsResponse xmlns="urn:soap.convio.com"
 xmlns:ens="urn:object.soap.convio.com" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <Record xsi:type="ens:Constituent">
            <ens:ConsId>1001483</ens:ConsId>
            <ens:ConsName>
               <ens:FirstName>Harry</ens:FirstName>
               <ens:LastName>Potter</ens:LastName>
            </ens:ConsName>
            <ens:UserName>potter</ens:UserName>
            <ens:PrimaryEmail>potter@leakycauldron.com</ens:PrimaryEmail>
            <ens:HomeAddress>
               <ens:Street1>4 Privet Drive</ens:Street1>
               <ens:City>Little Whinging</ens:City>
               <ens:State>CA</ens:State>
```

```
               <ens:Zip>94705</ens:Zip>
               <ens:Country>USA</ens:Country>
            </ens:HomeAddress>
         </Record>
      </GetIncrementalInsertsResponse>
   </soap:Body>
</soap:Envelope>
```

Note that the system limits each request to a maximum of 200 records. To download larger record sets, the client must make multiple requests, incrementing the `Page` parameter each time. The basic logic that a client must implement to download a full set of records is as follows:

1. Make an initial request to `GetIncrementalInserts`, `GetIncrementalUpdates` or `GetIncrementalDeletes`. Specify `Page` 1 and a `PageSize` of up to 200.
2. Process the records in the response, counting them as you go.
3. If the number of records in the response equals the requested `PageSize`, then make another request, this time specifying `Page = 2` and always using the same `PageSize`.
4. Continue until the number of records in the response is less than the `PageSize`. Note that the record count in a response may be zero if the size of the record set is an even multiple of the requested `PageSize`.

Note: All three methods, `GetIncrementalInserts`, `GetIncrementalUpdates`, and `GetIncrementalDeletes` must be called in order to retrieve all of the Convio records that may have changed during the synchronization window.

`GetIncrementalUpdates` returns the list of previously-synchronized records that have been updated on the Convio system during the synchronization window that were created before the start of the synchronization window. Records created after the start of the synchronization window and later modified would be downloaded as new, using `GetIncrementalInserts`, rather than as updated from the standpoint of the client. This method returns only updates to constituent records that have been assigned a key on the external system. This method retroactively includes any previously updated records if their key was assigned during the current sync window. Processing for this method on the middleware should handle the possibility of collisions.

`GetIncrementalInserts` returns the list of records added to the Convio Web Services partition during the synchronization window. Processing of this response should add or merge these records into the external database according to your policy on detecting and handling duplicates. Also, it is good practice for each record added to update the `MemberId` field of the record in the Convio database with the unique primary key value assigned to that record by the external database system, for example:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:soap.convio.com">
  <soapenv:Header>
 <urn:Session><urn:SessionId>3868cf8ebb49e046a024e18ad8420ea931fe8b64:JSESSIONID=abc_g9cr
urn:SessionId></urn:Session>
  </soapenv:Header>
  <soapenv:Body>
    <urn:Update>
    <urn:PartitionId>123</urn:PartitionId>
      <urn:Record xsi:type="ns3:Constituent"
        xmlns:ns3="urn:object.soap.convio.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ns3:ConsId>1001123</ns3:ConsId>
        <ns3:MemberId>98675</ns3:MemberId>
      </urn:Record>
      <urn:Record xsi:type="ns3:Constituent"
        xmlns:ns3="urn:object.soap.convio.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ns3:ConsId>1001124</ns3:ConsId>
        <ns3:MemberId>98676</ns3:MemberId>
      </urn:Record>
```

```
      <urn:Record xsi:type="ns3:Constituent"
        xmlns:ns3="urn:object.soap.convio.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ns3:ConsId>1001125</ns3:ConsId>
        <ns3:MemberId>98678</ns3:MemberId>
      </urn:Record>
    </urn:Update>
  </soapenv:Body>
</soapenv:Envelope>
```

This will make it easier to maintain synchronization between Convio and the external database in subsequent syncs.

Note: If Convio duplicate record detection is enabled, new constituent registrations may not be immediately visible through GetIncrementalInserts. Instead, the system first queues them for duplicate record checking. An automatic batch processes this queue periodically, inserting unique records, merging duplicates with existing constituent records, and flagging questionable duplicates to be checked manually by an administrator. Only after a record has been processed and either inserted as a new record or merged with an existing record will the data appear in a GetIncrementalInserts or GetIncrementalUpdates response.

`GetIncrementalDeletes` returns the list of records removed from the Convio Web Services partition during the synchronization window.

Note: If a record was both added to and deleted from the Convio partition during the synchronization window, only the deletion event, and not the insertion event, will be returned.

Synchronizing Summary Transaction History

The Convio constituent record includes two sets of fields that summarize the constituent's online and offline transaction history. The values of these fields are commonly used for personalizing fundraising appeals and other content.

The first set of fields summarizes transactions that the constituent has made through all Convio modules, including Donations, TeamRaiser, E-Commerce, etc. These fields are maintained exclusively by Convio and are **read-only**. They can be downloaded as part of the synchronization process using the `GetIncrementalInserts` and `GetIncrementalUpdates` operations. They should **not** ever be uploaded as part of a `Create` or `Update` operation.

The WSDL defines two types of child elements for the online summary transaction history:

1. `TransactionSummary` contains a brief record of a single transaction. The constituent record includes this summary for the first, largest and last transactions made by the constituent. Each transaction summary includes:
   a. Amount
   b. Campaign
   c. Date
   d. Donation type
   e. Payment method
   f. Tender type
2. `AnnualTransactionSummary` contains two fields: the total amount and the number of transactions made in a particular year. The constituent record includes an annual summary for the current and previous years.

Convio automatically updates the online transaction summary fields as appropriate when the constituent performs a transaction. The system updates the last modified date on the constituent record when it updates any of these fields, so the record will be included in the `GetIncrementalUpdates` operation.

Synchronizing Offline Transaction Summary history

The second set of fields summarizes the constituent's transaction history based on an external source, such as an offline donor database. These fields are updatable manually through the administrator interface or automatically via web services. They can be downloaded as part of the synchronization process using the `GetIncrementalInserts` and `GetIncrementalUpdates` operations. They can also be uploaded as part of a `Create` or `Update` operation.

The constituent record has five pairs of fields for external transaction history, each of which has a corresponding field in the WSDL:

1. First gift date and amount
2. Last gift date and amount
3. Largest gift date and amount
4. Year-to-date amount and count
5. Lifetime amount and count

Note that maintenance of these fields is entirely up to the client or middleware. The field values can reflect online as well as offline activity, or follow any other sets rules defined by the client.

Querying for Additional Information

The synchronization logic implemented by the middleware may require additional information that is not provided by the upload or download responses. Convio Web Services also provides a Query operation for this purpose.

A `Query` request must contain a `QueryString` in the general form:

```
select Field1 [, Field2, Field3, ...] from RecordType [where
Criteria]
```

`RecordType` is the name of a top-level element in the Convio WSDL that extends the `Record` base type, such as `Constituent` or `Donation`. The field list must be a subset of element names of fields that the WSDL defines for that type. To specify fields that are sub-fields of an embedded complex type, use dot notation, e.g. `ConsName.FirstName` or `HomeAddress.City`.

The query `Criteria` supports basic Boolean comparison expressions. See *Convio Query Language* for reference to the grammar.

A typical query request looks like this:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>e96ce3db61a80642510225a783540f6dcb6e702d:JSESSIONID=abcMZ5dmWgZeidZtyWZ3
ns3:SessionId>
      </ns3:Session>
   </soapenv:Header>
   <soapenv:Body>
      <ns3:Query xmlns:ns3="urn:soap.convio.com">
         <ns3:QueryString>select ConsId from Constituent where ConsId =
 1001482</ns3:QueryString>
         <ns3:Page>1</ns3:Page>
         <ns3:PageSize>3</ns3:PageSize>
      </ns3:Query>
   </soapenv:Body>
</soapenv:Envelope>
```

The structure of the response is identical to that of the GetIncrementalInserts, GetIncrementalUpdates, and GetIncrementalDeletes operations:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <QueryResponse xmlns="urn:soap.convio.com"
 xmlns:ens="urn:object.soap.convio.com" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <Record xsi:type="ens:Constituent">
            <ens:ConsId>1001482</ens:ConsId>
         </Record>
      </QueryResponse>
   </soap:Body>
</soap:Envelope>
```

Synchronizing Transaction Data

Convio Web Services provides download synchronization of transaction data for the following transaction types from Convio to the external system:

- Action Alert Responses
- Donations
- Personal Fundraising Gifts
- Product Orders
- TeamRaiser Gifts
- TeamRaiser Registrations
- Ticket Purchases
- Refunds

As with Constituent data, transaction data is accessed within an active synchronization session. Unlike Constituent data, transaction data supports download synchronization only; data cannot be uploaded from the external system to Convio. Additional data associated with transactions may be retrieved using the `Query` or `Find` operations.

Synchronizing Action Alert Responses

The `ActionAlertResponse` type supports the `GetIncrementalInserts` operation to return the list of Action Alerts taken by constituents during the synchronization window. The Response records contain `AlertId` fields which can be used with the `Query` or `Find` operation to retrieve information about the Action Alert.

Synchronizing Gifts and Purchases

The `Donation`, `PersonalFundraisingGift`, `ProductOrder`, `TeamRaiserGift`, `TeamRaiserRegistration`, and `TicketPurchase` transaction types support `GetIncrementalInserts`, `GetIncrementalUpdates`, and `GetIncrementalDeletes`.

Associated types can be retrieved using `Query` or `Find` operations to retrieve additional information about these transactions.

Types associated with `Donation` records include `DonationCampaign`, `DonationForm`, and `DonationLevel`.

The `PersonalFundraisingCampaign` type is associated with `PersonalFundraisingGift` records.

`Shipment`, `Item` and `Product` types are be associated with `ProductOrder` transactions. See *Product Orders* for additional information.

Types associated with `TeamRaiserGift` and `TeamRaiserRegistration` records include `TeamRasierEvent`, `TeamRaiserCompany`, `TeamRaiserParticipationType` and `TeamRaiserTeam`.

`TicketPurchase` records are associated with `CalendarEvent` types.

`Discount` records may be associated with `Payment` transactions applied to any of these transaction types.

Synchronizing Refunds

Refunds for Convio financial transactions have two primary characteristics:

1. **Online vs. offline** Online refunds are processed by Convio, by issuing a credit through the Convio payment gateway back to the same card used to make the donation. Offline refunds are recorded in Convio but processed by external means, such as sending a check to the donor in the mail.

2. **Partial vs. full** A partial refund reduces the amount of the original donation. A partial refund updates the amount in the original donation record. In contrast, a full refund deletes the original donation record entirely. A copy of the deleted donation record is saved for auditing purposes.

All four combinations of online vs. offline and partial vs. full refunds are available to client administrators using the donation management console. Administrators may issue multiple partial refunds for a single donation. They may also refund the full remaining amount after issuing one or more partial refunds. The system creates a new "refund transaction" for each online refund event; as described below, the amount of all offline refunds can be inferred by comparing the current amount with the original amount.

Partial refunds

The `GetIncrementalInserts` and `GetIncrementalUpdates` operations may include financial transactions that have been partially refunded. The `GetIncrementalInserts` operation may include transactions that were both created and partially refunded during the active synchronization window. The `GetIncrementalUpdates` operation may include transactions that were created prior to the start of the active synchronization window, and then partially refunded during the window.

If a partial refund has occurred, then the `OriginalAmount` field will show how much the donor originally paid, while the `Amount` field shows the current amount after all online and offline partial refunds have been applied. Donations that have not been partially refunded have a value of zero for `OriginalAmount` field.

Full refunds

The `GetIncrementalDeletes` operation includes financial transactions that have been fully refunded. The `Amount` field indicates shows how much was refunded. Note that this may be less than the value of the original transaction if a partial refund was issued first.

Retrieving refund details

Financial transaction records downloaded with the `GetIncrementalInserts` , `GetIncrementalUpdates` and `GetIncrementalDeletes` operations only provide a "before and after" view of refund information. A financial transaction record itself does not include details about the date, amount and other processing details for each refund (either partial or full) that administrators may have applied.

To obtain additional details about refunds that occurred during the synchronization window, you can use the `GetIncrementalInserts` operation to retrieve records of type `Refund` . Each `Refund` record provides complete processing details for a single partial or full refund, and includes an `OriginalTransactionId` field so that you relate the refund back to a donation or other financial transaction. A single financial transaction may be associated with multiple refunds. Note that `Refund` records are only available for online refunds at this time; you can infer an offline refund if the sum of all online refunds does not add up to the difference between the `Amount` and the `OriginalAmount` on a financial transaction.

It is also possible to use the `Query` operation to retrieve the full set of refund records for a particular financial transaction, regardless of date:

```
 select RefundId, Payment from Refund where OriginalTransactionId = [insert a
transaction ID here]
```

When using this approach, care must be taken to avoid recognizing the same refund multiple times. In most cases it will be more convenient to use `GetIncrementalInserts` to only retrieve the refunds that have occurred during the active synchronization window.

Retrieving Settled Financial Transactions

The following type of query may be used to retrieve the list of transactions that settled (were cleared for payment) within a particular time period:

```
 select
   TransactionId, Payment.Amount, Payment.BillingName,
Payment.BillingAddress, ...
 from
```

```
    Donation
  where
    Payment.SettlementDate >= 2009-05-01
  and
    Payment.SettlementDate < 2009-06-01
  and
    Payment.SettlementStatus = "SETTLED"
```

This may be useful for reconciling online donations with a bank statement.

Ending the Synchronization Session

The final step after exchanging data is to call the `EndSynchronization` operation. This marks the synchronization session as complete and causes the system to increment the synchronization window the next time that a client starts a synchronization session.

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>3868cf8ebb49e046a024e18ad8420ea931fe8b64:JSESSIONID=abc_g9cmWx9L0_dZyWZ3r
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:EndSynchronization xmlns:ns3="urn:soap.convio.com">
      <ns3:PartitionId>123</ns3:PartitionId>
    </ns3:EndSynchronization>
  </soapenv:Body>
</soapenv:Envelope>
```

Convio responds by confirming the time window and ID of the active synchronization:

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EndSynchronizationResponse xmlns="urn:soap.convio.com">
      <Result>
        <PartitionId>10000104</PartitionId>
        <Start>2008-12-01T18:13:44Z</Start>
        <End>2008-12-01T18:13:47Z</End>
        <SyncId>118</SyncId>
      </Result>
    </EndSynchronizationResponse>
  </soap:Body>
</soap:Envelope>
```

Note: Your middleware should record each session's `SyncId`. It can be used to retrieve a log of the records updated and downloaded during the session at a later date.

Testing and Data Validation

Creating a Test Data Partition on Convio

It is strongly recommended that you create a test Partition on Convio for use during the development and testing of your data synchronization process, to ensure that test data is kept separate from "live" data and can be re-initialized to

a know state for the purpose of re-running a test. A Partition is a subset of your Convio Constituent database which Convio Web Services may access. To create a Partition on Convio:

1. Log in to Convio as a site administrator.
2. Click **Data Management** on the main menu of the Administrator Home Page.
3. From the **Data Management** menu, select **Query**.
4. **Create a new Query** that defines the common, unique characteristic of your test records.
5. From the **Data Management** menu, select **Import/Export**.
6. Choose the **Partition Management** tab.
7. **Create a new Partition** based on the **Query** you created above.

Note: You cannot use Queries based on Group membership as the basis for creating a Partition.

Creating a Test Partition in the External Database

For the same reasons you should use a "test" database partition in Convio, the external database used for development and test should contain only test data, not "live" data. How this is accomplished will depend upon the external system; consult the system administrator, or documentation to determine how to create a test partition in the external system, or contact the system vendor for support.

Creating Test Transactions on Convio

If you will synchronize transaction information with the external database, you also need to create test transaction data for the transaction types you are interested in using the constituents in your test partition. You will need to enter these transactions through the Convio Online Marketing system by either documented manual test steps or a scripted process.

Debugging Tools

Your middleware synchronization platform or development environment may provide tools to help you develop test and debug the synchronization process. Be sure to take full advantage of these. Best practices include:

- Use the most sensitive error level and warning level settings practical in your compiler or development environment.
- Include debug assertions to test any assumptions in your code or scripts.
- Set breakpoints and step through your code or scripts in the debugger to inspect the code at runtime.
- Disable any global error-trapping features of your middleware or IDE.
- Enable verbose logging of all SOAP requests and responses in the middleware or IDE to more easily diagnose failures.
- Use the Convio Web Services Console to monitor, log and view incoming SOAP requests and responses on the Convio server side.

The Data Validation Cycle: Test, Fix, Repeat

Development, testing, and data validation is typically a cycle, rather than a sequence of events. During this cycle you will run a number of test synchronizations, carefully inspecting the data on both systems, and making adjustments to the field mapping or fixing bugs in the data synchronization process as necessary, until the process runs successfully and reliably.

Things to look for when verifying the results of synchronization include:

- Incorrect field mappings
- Invalid values
- Truncated values
- Missing records
- Un-handled synchronization errors
- Boundary conditions (for example, minimum and maximum data values, high transaction volume).

- Exception conditions (for example, missing required fields in one or the other system).

The Logging and Monitoring features available through the Convio Web Services Console are helpful when debugging a synchronization session.

Implementation

The data synchronization process may affect valuable data and impact systems that are critical to the ongoing mission of your organization. Therefore its implementation should be planned in advance, any risks identified and mitigation plans in place.

The Implementation Plan

A typical implementation plan might include:

- The roles and responsibilities of the implementation team.
- A backup and recovery strategy for any mission-critical data and a redundancy or recovery strategy for systems in the event of critical data loss or system failure during implementation.
- Carefully-reviewed and tested step-by-step processes detailing any necessary software installation, and any operating system, database, or network configuration.
- Verification steps in the process for auditing the results of installation, configuration, or sync operation.
- A data verification / data scrubbing process following the initial sync.
- Any other identified risks and their mitigation plans.
- Standard operating procedures for ensuring the continued operation of the system once deployed.
- Security and privacy policies.
- System Monitoring and Alerting policy.
- Logging policies.
- Auditing policies.

API Reference

Operations

AddRelationships

Adds Center, Group, or Interest relationships to one or more Constituent records.

Usage

This operation is a specialized version of the *Update* operation. It is provided as a logical complement to the *RemoveRelationships* operation.

Specify one or more CenterId, GroupId, and/or InterestId values to associate with the Constituent record when calling this operation.

Request Parameters

PartitionId

`nonNegativeInteger`

The system-generated unique identifier of a constituent *partition*

Force

`boolean`

Specify `true` to override *synchronization collisions*. Default is false.

Record

`Record`

A value is required

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Response Parameters

The response may contain zero or more response elements

ResultCode

`RecordResultCode`

`OK` if the operation was successful on the corresponding element in the request, or an appropriate error code otherwise

Message

`string`

A short readable status message associated with the result code

Record

`Record`

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*

- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
 <?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Header>
    <ns3:Session xmlns:ns3='urn:soap.convio.com' soapenv:mustUnderstand='0'>

 <ns3:SessionId>c55d8f07700b4aa8401e2b9c4d4262efa6da30d8:JSESSIONID=abcKuVJBroiHpeNByWZ3r
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:AddRelationships xmlns:ns3='urn:soap.convio.com'>
      <ns3:PartitionId>123</ns3:PartitionId>
      <ns3:Record xmlns:ns1='urn:object.soap.convio.com'
      xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:type='ns1:Constituent'>
        <ns1:PrimaryEmail>albus@hogwarts.edu</ns1:PrimaryEmail>
        <ns1:CenterId>1001</ns1:CenterId>
        <ns1:CenterId>1002</ns1:CenterId>
        <ns1:InterestId>1001</ns1:InterestId>
        <ns1:InterestId>1002</ns1:InterestId>
        <ns1:InterestId>1003</ns1:InterestId>
        <ns1:InterestId>1005</ns1:InterestId>
        <ns1:InterestId>1007</ns1:InterestId>
        <ns1:InterestId>1011</ns1:InterestId>
        <ns1:GroupId>1010</ns1:GroupId>
        <ns1:GroupId>1011</ns1:GroupId>
        <ns1:GroupId>1015</ns1:GroupId>
      </ns3:Record>
    </ns3:AddRelationships>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <AddRelationshipsResponse xmlns="urn:soap.convio.com"
      xmlns:ens="urn:object.soap.convio.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Result>
            <ResultCode>OK</ResultCode>
            <Message>User updated.</Message>
            <Record xsi:type="ens:Constituent">
               <ens:ConsId>1001082</ens:ConsId>
               <ens:MemberId xsi:nil="true"/>
               <ens:PrimaryEmail>albus@hogwarts.edu</ens:PrimaryEmail>
            </Record>
         </Result>
      </AddRelationshipsResponse>
   </soap:Body>
</soap:Envelope>
```

Create

Adds one or more constituent records to the Convio system

Usage

Adds one or more constituent records to Convio.

Note that even if a constituent is created in the context of synchronizing with a particular partition, the constituent will not be added to the partition unless it meets the membership criteria (i.e. has a mailing address).

Returns a separate result element for each record in the request, including the unique Convio ID (ConsId) assigned to the record. It is recommended that clients store the ConsId associated with each record in their offline system(s).

Accepts up to 50 records per request. There is no limit to the number of requests that may be submitted within a single synchronization.

Request Parameters

PartitionId

`nonNegativeInteger`

The system-generated unique identifier of a constituent *partition*

Record

`Record`

A value is required

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Response Parameters

The response may contain zero or more response elements

ResultCode

`RecordResultCode`

`OK` if the operation was successful on the corresponding element in the request, or an appropriate error code otherwise

Message

`string`

A short readable status message associated with the result code

Record

`Record`

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Header>
    <ns3:Session xmlns:ns3='urn:soap.convio.com' soapenv:mustUnderstand='0'>

 <ns3:SessionId>c55d8f07700b4aa8401e2b9c4d4262efa6da30d8:JSESSIONID=abcKuVJBroiHpeNByWZ3r
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:Create xmlns:ns3='urn:soap.convio.com'>
      <ns3:PartitionId>123</ns3:PartitionId>
      <ns3:Record xmlns:ns1='urn:object.soap.convio.com'
      xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:type='ns1:Constituent'>
        <ns1:ConsName>
          <ns1:FirstName>Albus</ns1:FirstName>
          <ns1:LastName>Dumbledore</ns1:LastName>
        </ns1:ConsName>
        <ns1:UserName>albus</ns1:UserName>
        <ns1:PrimaryEmail>albus@hogwarts.edu</ns1:PrimaryEmail>
        <ns1:HomeAddress>
          <ns1:Street1/>
          <ns1:City>Hogsmeade</ns1:City>
          <ns1:State>CA</ns1:State>
          <ns1:Zip>94702</ns1:Zip>
          <ns1:Country>USA</ns1:Country>
        </ns1:HomeAddress>
      </ns3:Record>
    </ns3:Create>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <CreateResponse xmlns='urn:soap.convio.com'
    xmlns:ens='urn:object.soap.convio.com'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
      <Result>
        <ResultCode>OK</ResultCode>
        <Message>New user added.</Message>
        <Record xsi:type='ens:Constituent'>
          <ens:ConsId>1001482</ens:ConsId>
          <ens:PrimaryEmail>albus@hogwarts.edu</ens:PrimaryEmail>
        </Record>
      </Result>
    </CreateResponse>
  </soap:Body>
</soap:Envelope>
```

Delete

Removes one or more constituents from the Convio system.

Usage

Use this operation to delete one or more Constituent records. Records to be deleted must be uniquely identified by ConsId, MemberId, or PrimaryEmail.

Request Parameters

PartitionId

`nonNegativeInteger`

The system-generated unique identifier of a constituent *partition*

Force

`boolean`

Specify `true` to override *synchronization collisions*. Default is false.

Record

`Record`

A value is required

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Response Parameters

The response may contain zero or more response elements

ResultCode

`RecordResultCode`

`OK` if the operation was successful on the corresponding element in the request, or an appropriate error code otherwise

Message

`string`

A short readable status message associated with the result code

Record

`Record`

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Session xmlns="urn:soap.convio.com">

 <SessionId>4eaace35dae004a4f7d90a8dcae205b1ad5f5756:JSESSIONID=abc5Zdz6hzoWW4FfPx9ds:100
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <Delete xmlns="urn:soap.convio.com">
      <PartitionId>10000103</PartitionId>
```

```
      <Constituent>
        <ConsId>1001201</ConsId>
      </Constituent>
    </Delete>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <DeleteResponse xmlns="urn:soap.convio.com"
 xmlns:ens="urn:object.soap.convio.com" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <Result>
        <ResultCode>OK</ResultCode>
        <Message>User deleted.</Message>
        <Record xsi:type="ens:Constituent">
          <ens:ConsId>1001201</ens:ConsId>
          <ens:PrimaryEmail>karl@convio.com</ens:PrimaryEmail>
        </Record>
      </Result>
    </DeleteResponse>
  </soap:Body>
</soap:Envelope>
```

DescribeRecordType

Provides extended metadata about a Record type

Usage

This operation provides extended metadata about a Record type defined in the Convio WSDL.

The SupportedOperations elements list the operations supported for the specified record type. For example, the *Constituent* record type supports synchronization in both directions as well as query; the *Donation* record type supports only "download" synchronization from Convio to remote systems as well as query; and the *DonationCampaign* record type supports only query.

The Field elements provide information about the type's fields. Field properties may include:

Element	Description
Name	The name of the field
Writable	Whether or not the field may be modified on Create or Update operations
Custom	Whether or not the field name is an alias for a custom field in the Convio database
Nillable	Whether or not the field value may be Null
Multiple	Whether or not the field may return multiple instances
Type	The field's data type (may be either a simple type or another CWS schema type)

Element	Description
MaxLength	The maximum number of characters supported for text fields (longer values will be truncated)
IsCriterion	Whether or not the field may be used as search criteria in the "where" clause of a Query operation
IsWildcard	Whether or not the field is returned to a query that does not specifically request it when the parent type is requested
Option	One of a list of valid values for an enumeration type

Request Parameters

RecordType

`normalizedString`

A value is required

The name of the record type for which information is requested

ParentType

`normalizedString`

The name of the record's parent type, if the record type is a sub-type

ParentField

`normalizedString`

The field name of the type in the parent record

Response Parameters

Name

`string`

The name of the requested record type

SupportedOperations

`SupportedOperations`

The list of operations supported for the record type

Field

`Field`

Multiple values allowed

The names of one or more fields to include in the response

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>9b35f8351f300a7c11a5ce232c9ce82ed6240aa1:JSESSIONID=abc0IZk5Fzfq8gdboKT4r:10
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <DescribeRecordType xmlns='urn:soap.convio.com'>
      <RecordType>Constituent</RecordType>
    </DescribeRecordType>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <DescribeRecordTypeResponse xmlns="urn:soap.convio.com">
      <Result>
        <Name>Constituent</Name>
        <SupportedOperations>
          <Create>true</Create>
          <Update>true</Update>
          <Delete>true</Delete>
          <GetIncrementalInserts>true</GetIncrementalInserts>
          <GetIncrementalUpdates>true</GetIncrementalUpdates>
          <GetIncrementalDeletes>true</GetIncrementalDeletes>
          <Query>true</Query>
        </SupportedOperations>
        <Field>
          <Name>ConsId</Name>
          <Writable>false</Writable>
          <Custom>false</Custom>
          <Nillable>true</Nillable>
          <Multiple>false</Multiple>
          <Type>nonNegativeInteger</Type>
          <IsCriterion>true</IsCriterion>
        </Field>
        <Field>
          <Name>SiteId</Name>
          <Writable>false</Writable>
          <Custom>false</Custom>
          <Nillable>true</Nillable>
          <Multiple>false</Multiple>
          <Type>nonNegativeInteger</Type>
          <IsCriterion>false</IsCriterion>
        </Field>
        <Field>

          . . .

      </Result>
    </DescribeRecordTypeResponse>
  </soap:Body>
</soap:Envelope>
```

EndSynchronization

Ends a synchronization session between Convio and a remote system.

Usage

Ending a synchronization indicates that the remote system has successfully exchanged all records with Convio, and that it is safe to move the default time window for the next synchronization, starting with the start time of the synchronization that has just ended and ending with the current time.

Returns a *SynchronizationFault* if there is no active synchronization session for the specified partition and currently authenticated user.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

Response Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

Start

`dateTime`

A value is required

The start time of the current synchronization window

End

`dateTime`

A value is required

The end time of the current synchronization window

SyncId

`nonNegativeInteger`

A value is required

The system-generated unique identifier for the synchronization session

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*
- *SynchronizationFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?&gt;
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
```

```
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>9b35f8351f300a7c11a5ce232c9ce82ed6240aa1:JSESSIONID=abc0IZk5Fzfq8gdboKT4r:10
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <EndSynchronization xmlns='urn:soap.convio.com'>
      <PartitionId>10000103</PartitionId>
    </EndSynchronization>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <EndSynchronizationResponse xmlns="urn:soap.convio.com">
      <Result>
        <PartitionId>10000103</PartitionId>
        <Start>2008-12-11T00:00:00Z</Start>
        <End>2008-12-12T00:00:00Z</End>
        <SyncId>183</SyncId>
      </Result>
    </EndSynchronizationResponse>
  </soap:Body>
</soap:Envelope>
```

Find

Locates one or more records of the specified type by primary key and returns the requested fields.

Usage

This operation is a specific type of *Query* for easily locating one or more records by Id (primary key).

Specify the RecordType to return.

Specify one or more Id values. Requesting multiple Id values will return one record for each valid value.

Specify one or more fields to return for each record in Field elements.

Request Parameters

RecordType

`string`

A value is required

The name of a synchronizable record type.

Field

`string`

A value is required

Multiple values allowed

The field or fields to return in the result set

Id

`nonNegativeInteger`

A value is required

Multiple values allowed

One or more instances of IDs (primary key values) for the requested record type

Response Parameters

The response may contain zero or more response elements

Record

`Record`

Multiple values allowed

The record or records returned by the query

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:soap.convio.com">
  <soapenv:Header>
    <urn:Session>
      <urn:SessionId>

 eb00584da96a24e3d78d70e70fb19b0b0c688851:JServSessionIdr005=1bzmoymcx1.app8007:1001021:3
      </urn:SessionId>
    </urn:Session>
  </soapenv:Header>
  <soapenv:Body>
    <urn:Find>
      <urn:RecordType>Constituent</urn:RecordType>
      <urn:Id>1001081</urn:Id>
      <urn:Field>ConsName</urn:Field>
      <urn:Field>PrimaryEmail</urn:Field>
      <urn:Field>HomeAddress</urn:Field>
    </urn:Find>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <FindResponse xmlns="urn:soap.convio.com"
      xmlns:ens="urn:object.soap.convio.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Record xsi:type="ens:Constituent">
            <ens:ConsName>
               <ens:FirstName>Jim</ens:FirstName>
```

```
               <ens:LastName>CenterOne</ens:LastName>
            </ens:ConsName>
            <ens:PrimaryEmail>devnull+c1@convio.com</ens:PrimaryEmail>
            <ens:HomeAddress>
               <ens:Street1 xsi:nil="true"/>
               <ens:Street2 xsi:nil="true"/>
               <ens:City xsi:nil="true"/>
               <ens:State xsi:nil="true"/>
               <ens:Zip xsi:nil="true"/>
            </ens:HomeAddress>
         </Record>
      </FindResponse>
   </soap:Body>
</soap:Envelope>
```

GetIncrementalDeletes

Returns all records of a particular type that were deleted during the active synchronization period

Usage

Use this operation to retrieve the following types of information:

- Constituents that were inactivated in the Convio system
- Financial transactions that were voided or fully refunded

Downloads records of a particular type that were removed from the partition during the active synchronization period.

Note that a record may be removed from the partition without actually being deleted. This may happen if the constituent associated with the record no longer meets the criteria for belonging to the partition. For example, say a partition is limited to constituents with a mailing address. If a constituent updates her profile and deletes a previously entered mailing address, then her constituent record will be removed from the partition, even though her record is still active.

If a constituent record is removed, associated financial transactions are implicitly removed from the partition as well, but they are not included in this operation. Only transactions that are explicitly deleted are removed.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

Page

`positiveInteger`

A value is required

The index of the page of results to retrieve.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Field

`string`

A value is required

Multiple values allowed

The names of one or more fields to include in the response

Response Parameters

The response may contain zero or more response elements

Record

`Record`

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>87b595b8e3c82928155dc5f424849a00f32e761d:JServSessionIdr005=0jqh43iip1.app80
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <GetIncrementalDeletes xmlns='urn:soap.convio.com'>
      <PartitionId>1001</PartitionId>
      <RecordType>Constituent</RecordType>
      <Page>1</Page>
      <PageSize>100</PageSize>
      <Field>ConsId</Field>
      <Field>MemberId</Field>
    </GetIncrementalDeletes>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalDeletesResponse xmlns="urn:soap.convio.com"
     xmlns:ens="urn:object.soap.convio.com"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
      <Record xsi:type="ens:Constituent">
        <ens:ConsId>1001201</ens:ConsId>
       <ens:MemberId xsi:nil="true" />
      </Record>
      <Record xsi:type="ens:Constituent">
        <ens:ConsId>1001065</ens:ConsId>
       <ens:MemberId>A-100223</ens:MemberId>
       </Record>
    </GetIncrementalDeletesResponse>
  </soap:Body>
</soap:Envelope>
```

GetIncrementalDeletesCount

Returns the count of deleted records for a particular record type and synchronization session

Usage

This operation provides the total number of records that may be returned by *GetIncrementalDeletes* for the specified RecordType in a synchronization session.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Response Parameters

RecordCount

`nonNegativeInteger`

A value is required

The total number of records in the result set for the call (may be greater than the PageSize or the maximum number that can be returned by one call).

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:soap.convio.com">
  <soapenv:Header>
    <urn:Session>
      <urn:SessionId>

 eb00584da96a24e3d78d70e70fb19b0b0c688851:JServSessionIdr005=1bzmoymcx1.app8007:1001021:3
      </urn:SessionId>
    </urn:Session>
  </soapenv:Header>
  <soapenv:Body>
    <urn:GetIncrementalDeletesCount>
      <urn:PartitionId>1001</urn:PartitionId>
      <urn:RecordType>Constituent</urn:RecordType>
      <urn:PageSize>100</urn:PageSize>
    </urn:GetIncrementalDeletesCount>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalDeletesCountResponse xmlns="urn:soap.convio.com">
      <Result>
        <RecordCount>2</RecordCount>
      </Result>
    </GetIncrementalDeletesCountResponse>
  </soap:Body>
</soap:Envelope>
```

GetIncrementalInserts

Returns all records of a particular type that were added to the Convio system during the active synchronization period.

Usage

Downloads records of a particular type that were added to a partition during the active synchronization period.

Note that record selection is based on when a record is added to the specified partition, not when the record was actually created. For a constituent record, there are several events that can trigger the addition of the record to a partition:

1. The record may represent a completely new registration that immediately meets the selection criteria for the partition.
2. An existing constituent may update her online profile (i.e. by adding a mailing address) such that her record now meets the selection criteria for the partition.
3. An existing constituent may engage in some interaction (i.e. by making a donation) that now qualifies her for the partition.

Note: If Convio duplicate record detection is enabled, new registrations are first added to the duplicate checking queue, and are moved into the partition as the queue is processed.

Constituents added or modified by the currently authenticated user are excluded from the results selected for this operation, although they may become downloadable in future synchronizations if modified by another user.

Financial transaction may be added to the partition in one of two ways:

1. The transaction was executed during the active synchronization period by a constituent who belongs to the partition. The constituent may have been added to the partition during the current synchronization period or before.
2. The transaction was executed prior to the start of the active synchronization period by an existing constituent who has just been added to the partition. Even though the transaction is older, they are now considered to be part of the partition since the constituent has become a member of the partition.

Pagination is handled in the same fashion as the Query operation.

The requested set of fields may include any subset of the fields that the Convio WSDL defines for the specified record type. The fields in the request must be listed in the same order as they are defined in the WSDL. You may use dot notation to specify selected subfields within composite element, such as ConsName.FirstName.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

Page

`positiveInteger`

A value is required

The index of the page of results to retrieve.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Field

`string`

A value is required

Multiple values allowed

The names of one or more fields to include in the response

Response Parameters

The response may contain zero or more response elements

Record

`Record`

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*

- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>3868cf8ebb49e046a024e18ad8420ea931fe8b64:JSESSIONID=abc_g9cmWx9L0_dZyWZ3
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:GetIncrementalInserts xmlns:ns3="urn:soap.convio.com">
      <ns3:PartitionId>123</ns3:PartitionId>
      <ns3:RecordType>Constituent</ns3:RecordType>
      <ns3:Page>1</ns3:Page>
      <ns3:PageSize>3</ns3:PageSize>
      <ns3:Field>ConsId</ns3:Field>
      <ns3:Field>ConsName</ns3:Field>
      <ns3:Field>UserName</ns3:Field>
      <ns3:Field>PrimaryEmail</ns3:Field>
      <ns3:Field>HomeAddress</ns3:Field>
    </ns3:GetIncrementalInserts>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalInsertsResponse xmlns="urn:soap.convio.com"
    xmlns:ens="urn:object.soap.convio.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Record xsi:type="ens:Constituent">
        <ens:ConsId>1001483</ens:ConsId>
        <ens:ConsName>
          <ens:FirstName>Harry</ens:FirstName>
          <ens:LastName>Potter</ens:LastName>
        </ens:ConsName>
        <ens:UserName>potter</ens:UserName>
        <ens:PrimaryEmail>potter@leakycauldron.com</ens:PrimaryEmail>
        <ens:HomeAddress>
          <ens:Street1>4 Privet Drive</ens:Street1>
          <ens:City>Little Whinging</ens:City>
          <ens:State>CA</ens:State>
          <ens:Zip>94705</ens:Zip>
          <ens:Country>USA</ens:Country>
        </ens:HomeAddress>
      </Record>
    </GetIncrementalInsertsResponse>
  </soap:Body>
</soap:Envelope>
```

GetIncrementalInsertsCount

Returns the count of inserted records for a particular record type and synchronization session

Usage

This operation provides the total number of records that may be returned by *GetIncrementalInserts* for the specified RecordType in a synchronization session.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Response Parameters

RecordCount

`nonNegativeInteger`

A value is required

The total number of records in the result set for the call (may be greater than the PageSize or the maximum number that can be returned by one call).

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:soap.convio.com">
  <soapenv:Header>
    <urn:Session>
      <urn:SessionId>

 eb00584da96a24e3d78d70e70fb19b0b0c688851:JServSessionIdr005=1bzmoymcx1.app8007:1001021:3
      </urn:SessionId>
    </urn:Session>
  </soapenv:Header>
  <soapenv:Body>
```

```
    <urn:GetIncrementalInsertsCount>
      <urn:PartitionId>1001</urn:PartitionId>
      <urn:RecordType>Constituent</urn:RecordType>
      <urn:PageSize>100</urn:PageSize>
    </urn:GetIncrementalInsertsCount>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalInsertsCountResponse xmlns="urn:soap.convio.com">
      <Result>
        <RecordCount>2</RecordCount>
      </Result>
    </GetIncrementalInsertsCountResponse>
  </soap:Body>
</soap:Envelope>
```

GetIncrementalUpdates

Retrieves pre-existing records of a particular type that were updated during the active synchronization session.

Usage

Downloads records of a particular type that were updated during the active synchronization period.

The result set only includes records that were part of the specified partition during the synchronization period. Records that were both added and subsequently updated during the synchronization period are treated as "inserts" and may be downloaded via the *GetIncrementalInsertss* operation. Records that were removed during the synchronization period are treated as "deletes" even if they were previously added and/or updated within the same period. Deletes may be downloaded via the *GetIncrementalDeletes* operation.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

Page

`positiveInteger`

A value is required

The index of the page of results to retrieve.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Field

`string`

A value is required

Multiple values allowed

The names of one or more fields to include in the response

Response Parameters

The response may contain zero or more response elements

Record

`Record`

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>3868cf8ebb49e046a024e18ad8420ea931fe8b64:JSESSIONID=abc_g9cmWx9L0_dZyWZ3r
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:GetIncrementalUpdates xmlns:ns3="urn:soap.convio.com">
      <ns3:PartitionId>123</ns3:PartitionId>
      <ns3:RecordType>Constituent</ns3:RecordType>
      <ns3:Page>1</ns3:Page>
      <ns3:PageSize>3</ns3:PageSize>
      <ns3:Field>ConsId</ns3:Field>
      <ns3:Field>ConsName</ns3:Field>
      <ns3:Field>UserName</ns3:Field>
      <ns3:Field>PrimaryEmail</ns3:Field>
      <ns3:Field>HomeAddress</ns3:Field>
    </ns3:GetIncrementalUpdates>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalUpdatesResponse xmlns="urn:soap.convio.com"
    xmlns:ens="urn:object.soap.convio.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <Record xsi:type="ens:Constituent">
        <ens:ConsId>1001483</ens:ConsId>
        <ens:ConsName>
          <ens:FirstName>Harry</ens:FirstName>
          <ens:LastName>Potter</ens:LastName>
        </ens:ConsName>
        <ens:UserName>potter</ens:UserName>
        <ens:PrimaryEmail>potter@leakycauldron.com</ens:PrimaryEmail>
        <ens:HomeAddress>
          <ens:Street1>4 Privet Drive</ens:Street1>
          <ens:City>Little Whinging</ens:City>
          <ens:State>CA</ens:State>
          <ens:Zip>94705</ens:Zip>
          <ens:Country>USA</ens:Country>
        </ens:HomeAddress>
      </Record>
    </GetIncrementalUpdatesResponse>
  </soap:Body>
</soap:Envelope>
```

GetIncrementalUpdatesCount

Returns the count of updated records for a particular record type and synchronization session

Usage

This operation provides the total number of records that may be returned by *GetIncrementalUpdates* for the specified RecordType in a synchronization session.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

RecordType

`string`

A value is required

The name of a synchronizable record type.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Response Parameters

RecordCount

`nonNegativeInteger`

A value is required

The total number of records in the result set for the call (may be greater than the PageSize or the maximum number that can be returned by one call).

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:urn="urn:soap.convio.com">
  <soapenv:Header>
    <urn:Session>
      <urn:SessionId>

 eb00584da96a24e3d78d70e70fb19b0b0c688851:JServSessionIdr005=1bzmoymcx1.app8007:1001021:3
      </urn:SessionId>
    </urn:Session>
  </soapenv:Header>
  <soapenv:Body>
    <urn:GetIncrementalUpdatesCount>
      <urn:PartitionId>1001</urn:PartitionId>
      <urn:RecordType>Constituent</urn:RecordType>
      <urn:PageSize>100</urn:PageSize>
    </urn:GetIncrementalUpdatesCount>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetIncrementalUpdatesCountResponse xmlns="urn:soap.convio.com">
      <Result>
        <RecordCount>2</RecordCount>
      </Result>
    </GetIncrementalUpdatesCountResponse>
  </soap:Body>
</soap:Envelope>
```

GetServerTime

Usage

Returns the current time according the system clock on the Convio database server.

All record creation and modification timestamps are determined by the system clock on the Convio database server. Clients may call this operation to verify that their middleware clock is consistent with Convio. Discrepancies of more than 30 seconds may produce misleading results when performing a synchronization.

Request Parameters

No parameters

Response Parameters

Time

`dateTime`

A value is required

The current date and time on the server

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetServerTime xmlns="urn:soap.convio.com">
    </GetServerTime>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetServerTimeResponse xmlns="urn:soap.convio.com">
      <Result>
        <Time>2008-12-12T17:19:58Z</Time>
      </Result>
    </GetServerTimeResponse>
  </soap:Body>
</soap:Envelope>
```

GetSynchronizationLog

Returns a list of synchronization sessions that overlap with the specified time frame

Usage

This operation lists the synchronization sessions that occurred during a particular time period. It may be useful for diagnostics and reporting.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

Start

`dateTime`

Start of the date range for returning sessions. Default is the date of first synchronization session.

End

`dateTime`

End of the date range for returning sessions. Default is the date of the last synchronization session.

Response Parameters

The response may contain zero or more response elements

PartitionId

`string`

The system-generated unique identifier of a constituent *partition*

Created

`dateTime`

Creation date for the log

Start

`dateTime`

Start of the date range for returning sessions. Default is the date of first synchronization session.

End

`dateTime`

End of the date range for returning sessions. Default is the date of the last synchronization session.

Completed

`boolean`

Flag indicating when the session was completed successfully

SyncId

`nonNegativeInteger`

The system-generated unique identifier for the synchronization session

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>de34cc6560b3615d06cf2820e454f2d8503776f9:JServSessionIdr005=ef0913cjj1.app80
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <GetSynchronizationLog xmlns='urn:soap.convio.com'>
      <PartitionId>10000103</PartitionId>
```

```
      <Start>2009-5-1T00:00:00+0000</Start>
      <End>2009-5-10T00:00:00+0000</End>
    </GetSynchronizationLog>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSynchronizationLogResponse xmlns="urn:soap.convio.com">
      <Synchronization>
        <PartitionId>10000103</PartitionId>
        <Start>2009-04-20T23:36:20Z</Start>
        <End>2009-05-03T14:47:46Z</End>
        <Completed>false</Completed>
        <SyncId>242</SyncId>
      </Synchronization>
    </GetSynchronizationLogResponse>
  </soap:Body>
</soap:Envelope>
```

GetSynchronizationRequestLog

Returns a log of the the operations performed and record counts returned for a particular synchronization session.

Usage

Given the SyncId of a previous synchronization session, this operation returns a list of the synchronization requests logged during the session and the counts of the records returned.

Request Parameters

SyncId

`string`

A value is required

The system-generated unique identifier for the synchronization session

Response Parameters

The response may contain zero or more response elements

Operation

`normalizedString`

RecordType

`normalizedString`

The name of a synchronizable record type.

RecordCount

`nonNegativeInteger`

The total number of records in the result set for the call (may be greater than the PageSize or the maximum number that can be returned by one call).

RequestTime

`dateTime`

SyncId

`nonNegativeInteger`

The system-generated unique identifier for the synchronization session

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>87b595b8e3c82928155dc5f424849a00f32e761d:JServSessionIdr005=0jqh43iip1.app80
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <GetSynchronizationRequestLog xmlns='urn:soap.convio.com'>
      <SyncId>82</SyncId>
    </GetSynchronizationRequestLog>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSynchronizationRequestLogResponse xmlns="urn:soap.convio.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ens="urn:object.soap.convio.com">
      <RecordRequest xsi:type="RecordQueryRequest">
        <Operation>GetIncrementalInserts</Operation>
        <RecordType>Constituent</RecordType>
        <RecordCount>43</RecordCount>
        <RequestTime>2010-03-09T18:10:32Z</RequestTime>
        <SyncId>82</SyncId>
        <Page>1</Page>
        <PageSize>100</PageSize>
      </RecordRequest>
      <RecordRequest xsi:type="RecordQueryRequest">
        <Operation>GetIncrementalUpdates</Operation>
        <RecordType>Constituent</RecordType>
        <RecordCount>9</RecordCount>
        <RequestTime>2010-03-09T18:12:09Z</RequestTime>
        <SyncId>82</SyncId>
        <Page>1</Page>
        <PageSize>100</PageSize>
```

```
      </RecordRequest>
      <RecordRequest xsi:type="RecordQueryRequest">
        <Operation>GetIncrementalDeletes</Operation>
        <RecordType>Constituent</RecordType>
        <RecordCount>2</RecordCount>
        <RequestTime>2010-03-09T18:12:12Z</RequestTime>
        <SyncId>82</SyncId>
        <Page>1</Page>
        <PageSize>100</PageSize>
      </RecordRequest>
    </GetSynchronizationRequestLogResponse>
  </soap:Body>
</soap:Envelope>
```

Login

Authenticates the credentials of a Convio API Administrator account that is authorized to access Web Services

Usage

Authenticates and authorizes a web service client using the specified user name and password.

A valid username and password for accessing Convio web services must meet the following criteria:

1. The username is associated with a active constituent that has administrator status
2. The password is valid for the constituent
3. The constituent is a member of an admin security group that has the "Use Convio APIs" permission

Returns a *LoginFault* if any of these criteria are not met.

Returns a SessionId session token upon successful login. Clients must include this token in the header of subsequent requests within the same session. The token is valid for up to 30 minutes of inactivity, after which the client must call this operation again to acquire a new token.

Request Parameters

UserName

`string`

A value is required

The User Name for an API Administrator Account

Password

`string`

A value is required

The Password for the API Administrator Account

Response Parameters

SessionId

`string`

A token for use in subsequent requests within the same login session

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *LoginFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Body>
    <Login xmlns='urn:soap.convio.com'>
      <UserName>apiuser-test</UserName>
      <Password>convio</Password>
    </Login>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LoginResponse xmlns="urn:soap.convio.com">
      <Result>

 <SessionId>1c79d882d01b4e24dc5cfcc9e462138391342c23:JSESSIONID=abcwDjW3UwpfAe_yIlS4r:100
SessionId>
      </Result>
    </LoginResponse>
  </soap:Body>
</soap:Envelope>
```

Query

Usage

Returns a filtered set of records of a particular type from the Convio database.

A Query request must include a query string in the general form:

```
  select Field1 [, Field2, Field3, ...] from RecordType [where Criteria]
```

The RecordType is the name of an element in the Convio WSDL that extends the *Record* base type, such as *Constituent* or *Donation*. The field list should be a list of element names selected from the WSDL schema definition for that type.

Not all record types defined in the WSDL support the Query operation. You can use the *DescribeRecordType* operation to check this.

The `[where Criteria]` clause may include one or more boolean phrases that reference fields in the requested record type, similar to the format of a simple WHERE clause in an SQL query.

Not all fields may be used in the criteria clause. Once again, you can use the *DescribeRecordType* operation to verify which fields may be used.

As with all operations, a valid SessionId session token returned by the *Login* operation is required in the header of the request.

The response consists of zero or more records of the requested type.A single response contains a maximum of 200 records. To ensure that you have retrieved all records for a particular query, your client needs to implement the following logic:

1. Make an initial request, optionally specifying a PageSize if you want to retrieve fewer than 200 records at a time.
2. If the number of records in the response is equal to the requested page size, submit a second request with the same query string and page size as the initial request, along with Page parameter of 2.
3. Continue making requests, incrementing the page each time, until the number of records in the response is fewer than the requested page size.

Only two specific usages of the Query operation are supported at this time:

- selecting all records from a reference table for performing lookups in the middleware, such as:

```
select CampaignId, Title from DonationCampaign
```

- looking up additional information about a specific record by primary key, such as:

```
select ConsId, UserName, HomeAddress from Constituent where ConsId = 123
```

Wildcards are supported in the select clause in in two ways. An asterisk by itself returns a subset of top-level fields for the selected records:

```
select * from Constituent where ConsId = 123
```

An asterisk appended to a nested element returns a subset of fields within that element:

```
select Donor.* from Donation
```

Note that the asterisk is optional in the latter case. This query returns the same result:

```
select Donor from Donation
```

A wildcard query does not return all fields in an element. Instead, it only returns a set of the most commonly used fields. You can use the field-level metadata returned by the *DescribeRecordType* operation to determine which fields are part of the wildcard set for a particular element.

Note: Timestamp fields in the Convio database are stored with millisecond precision. The Query operation returns values using one-second precision, which may cause unexpected results. For example, the query: `select * from Refund where Payment.Modifydate > 2009-10-21T01:10:16Z` may return a record with the apparently-equal value `2009-10-21T01:10:16Z` because the actual stored value includes a milliseconds component not shown.

Request Parameters

QueryString

`string`

A value is required

The *Query* to execute

Page

`positiveInteger`

A value is required

The index of the page of results to retrieve.

PageSize

`positiveInteger`

A value is required

The number of records to return per page

Response Parameters

The response may contain zero or more response elements

Record

`Record`

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*
- *InvalidQueryFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
      <ns3:Session xmlns:ns3="urn:soap.convio.com" soapenv:mustUnderstand="0">

 <ns3:SessionId>e96ce3db61a80642510225a783540f6dcb6e702d:JSESSIONID=abcMZ5dmWgZeidZtyWZ3r
ns3:SessionId>
      </ns3:Session>
   </soapenv:Header>
   <soapenv:Body>
      <ns3:Query xmlns:ns3="urn:soap.convio.com">
         <ns3:QueryString>select ConsId from Constituent where ConsId =
 1001482</ns3:QueryString>
         <ns3:Page>1</ns3:Page>
         <ns3:PageSize>3</ns3:PageSize>
      </ns3:Query>
   </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <QueryResponse xmlns="urn:soap.convio.com"
      xmlns:ens="urn:object.soap.convio.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Record xsi:type="ens:Constituent">
            <ens:ConsId>1001482</ens:ConsId>
         </Record>
      </QueryResponse>
   </soap:Body>
</soap:Envelope>
```

RemoveRelationships

Removes Center, Group, or Interest relationships from one or more constituent records.

Usage

This operation is a specialized version of the *Update* operation. It is a logical complement to the *AddRelationships* operation.

Specify the CenterId, GroupId, and/or InterestId values to remove from the Constituent record when calling this operation.

Note: When you remove a CenterId, the Center Members Group will also be automatically removed from the Constituent record. However, other Groups or Interests associated with that Center will not. Query the Group and Interest record where SecurityCategoryId is equal to the CenterId of the Center record to obtain the complete list of Groups or Interests associated with a Center.

Request Parameters

PartitionId

`nonNegativeInteger`

The system-generated unique identifier of a constituent *partition*

Force

`boolean`

Specify `true` to override *synchronization collisions*. Default is false.

Record

`Record`

A value is required

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Response Parameters

The response may contain zero or more response elements

ResultCode

`RecordResultCode`

`OK` if the operation was successful on the corresponding element in the request, or an appropriate error code otherwise

Message

`string`

A short readable status message associated with the result code

Record

`Record`

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*

- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
 <?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Header>
    <ns3:Session xmlns:ns3='urn:soap.convio.com' soapenv:mustUnderstand='0'>

 <ns3:SessionId>c55d8f07700b4aa8401e2b9c4d4262efa6da30d8:JSESSIONID=abcKuVJBroiHpeNByWZ3r
ns3:SessionId>
    </ns3:Session>
  </soapenv:Header>
  <soapenv:Body>
    <ns3:RemoveRelationships xmlns:ns3='urn:soap.convio.com'>
      <ns3:PartitionId>123</ns3:PartitionId>
      <ns3:Record xmlns:ns1='urn:object.soap.convio.com'
       xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
       xsi:type='ns1:Constituent'>
        <ns1:PrimaryEmail>albus@hogwarts.edu</ns1:PrimaryEmail>
        <ns1:CenterId>1001</ns1:CenterId>
        <ns1:CenterId>1002</ns1:CenterId>
        <ns1:InterestId>1001</ns1:InterestId>
        <ns1:InterestId>1002</ns1:InterestId>
        <ns1:InterestId>1003</ns1:InterestId>
        <ns1:InterestId>1005</ns1:InterestId>
        <ns1:InterestId>1007</ns1:InterestId>
        <ns1:InterestId>1011</ns1:InterestId>
        <ns1:GroupId>1010</ns1:GroupId>
        <ns1:GroupId>1011</ns1:GroupId>
        <ns1:GroupId>1015</ns1:GroupId>
      </ns3:Record>
    </ns3:RemoveRelationships>
  </soapenv:Body>
</soapenv:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <RemoveRelationshipsResponse xmlns="urn:soap.convio.com"
      xmlns:ens="urn:object.soap.convio.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Result>
            <ResultCode>OK</ResultCode>
            <Message>User updated.</Message>
            <Record xsi:type="ens:Constituent">
               <ens:ConsId>1001082</ens:ConsId>
               <ens:MemberId xsi:nil="true"/>
               <ens:PrimaryEmail>albus@hogwarts.edu</ens:PrimaryEmail>
            </Record>
         </Result>
      </RemoveRelationshipsResponse>
   </soap:Body>
</soap:Envelope>
```

StartSynchronization

Starts a synchronization session

Usage

Initiates a synchronization between Convio and a remote system.

Allows the caller to specify an optional time period for the synchronization. If a start time is not specified, the system defaults to the start time of the last successful synchronization. If an end time is not specified, the system defaults to the current time.

Requires a partition ID that specifies which set of constituents to synchronize. Partitions are managed through the Convio administration interface. You can use the *Query* operation to retrieve a list of available partition IDs:

```
select PartitionId, Name from Partition
```

Synchronizations are associated with both a partition ID and the currently authenticated user. Each user can only have one active synchronization per partition at a time.

Returns a SyncId that uniquely identifies this synchronization for auditing purposes. This ID can be used to retrieve an audit log of the synchronization for up to 90 days after it occurs (See *GetSynchronizationRequestLog*).

Throws a *SynchronizationFault* if there is an existing active synchronization for the same user and partition that has not been explicitly ended by a call to the *EndSynchronization* operation. There are three possible reasons for this:

1. Another sync session is currently in progress.
2. The client is not performing the expected sequence of operations by calling EndSynchronization at the end of each sync session
3. An error happened during the last sync session that prevents the client from ending the session successfully.

To force the system to start a new session even if a previous session is already active, set the Force parameter to `true`. In this case the last active synchronization will be left open for auditing purposes. Note that it is not advisable to use the Force parameter unless you have determined why the active sync already exists.

Request Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

Force

`boolean`

Specify `true` to override *synchronization collisions*. Default is false.

Start

`dateTime`

The start time for the current synchronization window

End

`dateTime`

The end time for the current synchronization window

Response Parameters

PartitionId

`nonNegativeInteger`

A value is required

The system-generated unique identifier of a constituent *partition*

Start

`dateTime`

A value is required

The start time for the current synchronization window

End

`dateTime`

A value is required

The end time for the current synchronization window

SyncId

`nonNegativeInteger`

A value is required

The system-generated unique identifier for the synchronization session

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*
- *SynchronizationFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8' ?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>

 <SessionId>9b35f8351f300a7c11a5ce232c9ce82ed6240aa1:JSESSIONID=abc0IZk5Fzfq8gdboKT4r:100
SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <StartSynchronization xmlns='urn:soap.convio.com'>
      <PartitionId>10000103</PartitionId>
      <Start>2008-12-11T00:00:00+0000</Start>
      <End>2008-12-12T00:00:00+0000</End>
    </StartSynchronization>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <StartSynchronizationResponse xmlns="urn:soap.convio.com">
```

```
      <Result>
        <PartitionId>10000103</PartitionId>
        <Start>2008-12-11T00:00:00Z</Start>
        <End>2008-12-12T00:00:00Z</End>
        <SyncId>183</SyncId>
      </Result>
    </StartSynchronizationResponse>
  </soap:Body>
</soap:Envelope>
```

Update

Updates one or more constituent records

Usage

The Update operation attempts to update a record with the values supplied in the request. Only fields which contain values or which specify the attribute `xsi:nil="true"` are updated. Fields missing from the request, or not containing a value, are not changed. Records to be updated must be uniquely identified by ConsId or by a unique MemberId or PrimaryEmail .

One of ConsId, MemberId, or PrimaryEmail must be specified in order to locate the record to update.

The order of precedence of the search terms is (a) ConsId (b) MemberId, then (c) PrimaryEmail. If a record is not found (RECORD_NOT_FOUND) or if multiple records are returned (UPDATE_MATCH_AMBIGUOUS) by any specified search term evaluated in this order, the method returns an error.

If a request specifies more than one search parameter, then the system will attempt to update the values of lower-precedence parameters using the values provided. This may result in an error if a duplicate value exists in the database for a specified MemberId (MATCH_EXISTING) or PrimaryEmail (DUPLICATE_EMAIL).

To delete the value of a given field on `Update` , you must specify the field in the `Update` request and explicitly set it null.

For example:

SOAP Request	Result
<MiddleName xsi:nil="true"/>	middle_name set to null
<MiddleName />	middle_name unchanged
<MiddleName>John</MiddleName>	middle_name changed to "John"

Note: Fields which describe one-to-many relationships between the Constituent record and Center, Group or Interest records are not modified using `Update` . Use instead the `AddRelationships` and `RemoveRelationships` operations to change these associations.

Request Parameters

PartitionId

`nonNegativeInteger`

The system-generated unique identifier of a constituent *partition*

Force

`boolean`

Specify `true` to override *synchronization collisions*. Default is false.

Record

Record

A value is required

Multiple values allowed

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Response Parameters

The response may contain zero or more response elements

ResultCode

RecordResultCode

`OK` if the operation was successful on the corresponding element in the request, or an appropriate error code otherwise

Message

`string`

A short readable status message associated with the result code

Record

Record

A constituent record containing at least `ConsId`, `MemberId` and `PrimaryEmail` for identification purposes.

Faults

- *ServiceUnavailableFault*
- *ServerFault*
- *MalformedMessageFault*
- *InvalidMessageFault*
- *SessionFault*

Example SOAP request

```
<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'>
  <soap:Header>
    <Session xmlns='urn:soap.convio.com'>
      <SessionId>

 87b595b8e3c82928155dc5f424849a00f32e761d:JServSessionIdr005=0jqh43iip1.app8007:1001021:3
      </SessionId>
    </Session>
  </soap:Header>
  <soap:Body>
    <Update xmlns='urn:soap.convio.com'>
      <PartitionId>1001</PartitionId>
      <Constituent>
        <ConsId>1001081</ConsId>
        <ConsName>
          <MiddleName>Williamson</MiddleName>
        </ConsName>
      </Constituent>
    </Update>
  </soap:Body>
</soap:Envelope>
```

Example SOAP response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <UpdateResponse xmlns="urn:soap.convio.com"
         xmlns:ens="urn:object.soap.convio.com"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Result>
            <ResultCode>OK</ResultCode>
            <Message>User updated.</Message>
            <Record xsi:type="ens:Constituent">
               <ens:ConsId>1001081</ens:ConsId>
               <ens:MemberId xsi:nil="true"/>
               <ens:PrimaryEmail>jwcenter@convio.com</ens:PrimaryEmail>
            </Record>
         </Result>
      </UpdateResponse>
   </soap:Body>
</soap:Envelope>
```

Types

ActionAlert

Fields

AlertId

`nonNegativeInteger`

A system-generated unique identifier for the Alert

ActionAlertName

`string`

Name that identifies the Alert to constituents

ActionAlertInternalName

`string`

An internally-visible name or alias for the Alert

ActionAlertDescription

`string`

Description provided by the administrator when the Alert record was created

ActionAlertStatus

`AlertStatusType`

Active or Disabled

ActionAlertType

`string`

Action Alert, Call Alert, or Vote Alert

AllowChoice

`boolean`

Whether or not the user may specify to whom the response to this Alert is sent

AllowMultipleResponse

`boolean`

Whether or not the user may respond more than once

AllowPrinting

`boolean`

Whether or not to allow printing of the alert

DeliveryMethod

`string`

Email or Fax

ResponseCount

`nonNegativeInteger`

Number of activists (constituents) who responded to this Alert

StateList

`string`

Determines whether a State list is displayed to user on the Alert Survey Page

SurveyId

`nonNegativeInteger`

A system-generated unique identifier for the alert response Survey Page

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

ActionAlertResponse

Fields

InteractionId

`nonNegativeInteger`

System-generated unique identifier of the Interaction record associated with this Response

SiteId

`nonNegativeInteger`

References the system-generated value identifying the Convio client site for the Response record

AlertId

`nonNegativeInteger`

System-generated unique identifier of the Alert that generated this Response

AlertResponseId

`nonNegativeInteger`

System-generated unique identifier of the ActionAlertResponse record

SubmitDate

`dateTime`

Date this Response was sent

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Activist

`Constituent`

Responder (constituent)

InteractionSource

`InteractionSource`

Interaction source information for this Response

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

Find

Query

ActivityStream

Fields

SiteId

`nonNegativeInteger`

InteractionId

`nonNegativeInteger`

InteractionType

`InteractionType`

ReferenceId

`nonNegativeInteger`

ObjectId

`nonNegativeInteger`

ObjectTitle

`string`

ObjectCreationDate

`long`

ConsId

`nonNegativeInteger`

CreationDate

`long`

SecurityCategoryId

`nonNegativeInteger`

Subject

`string`

ActorFirstName

`string`

ActorLastName

`string`

Supported Operations

No operations directly supported.

Address

Fields

Street1

`string`

Street number and name

Street2

`string`

Additional address information

Street3

`string`

Additional address information

City

`string`

Name of the city

State

`string`

Standard abbreviation of the state or province. One of:

- None

- AK
- AL
- AR
- AZ
- CA
- CO
- CT
- DC
- DE
- FL
- GA
- HI
- IA
- ID
- IL
- IN
- KS
- KY
- LA
- MA
- MD
- ME
- MI
- MN
- MO
- MS
- MT
- NC
- ND
- NE
- NH
- NJ
- NM
- NV
- NY
- OH
- OK
- OR
- PA
- RI
- SC
- SD
- TN
- TX
- UT
- VA
- VT
- WA
- WI
- WV
- WY
- AS

- FM
- GU
- MH
- MP
- PR
- PW
- VI
- AA
- AE
- AP
- AB
- BC
- MB
- NB
- NL
- NS
- NT
- NU
- ON
- PE
- QC
- SK
- YT

Zip

`string`

ZIP code or postal code

Country

`string`

Name of the country. One of:

- Afghanistan
- Albania
- Algeria
- American Samoa
- Andorra
- Angola
- Anguilla
- Antarctica
- Antigua and Barbuda
- Argentina
- Armenia
- Aruba
- Australia
- Austria
- Azerbaijan
- Bahamas
- Bahrain
- Bangladesh
- Barbados
- Belarus
- Belgium

- Belize
- Benin
- Bermuda
- Bhutan
- Bolivia
- Bosnia-Herzegovina
- Botswana
- Bouvet Island
- Brazil
- British Indian Ocean Territory
- Brunei
- Bulgaria
- Burkina Faso
- Burundi
- Cambodia
- Cameroon
- Canada
- Cape Verde
- Cayman Islands
- Central African Republic
- Chad
- Chile
- China
- Christmas Island
- Cocos (Keeling) Islands
- Colombia
- Comoros
- Congo
- Cook Islands
- Costa Rica
- Croatia
- Cuba
- Cyprus
- Czech Republic
- Denmark
- Djibouti
- Dominica
- Dominican Republic
- East Timor
- Ecuador
- Egypt
- El Salvador
- Equatorial Guinea
- Eritrea
- Estonia
- Ethiopia
- Falkland Islands
- Faroe Islands
- Fiji
- Finland
- Former Czechoslovakia
- Former USSR

- France
- French Guyana
- French Southern Territories
- Gabon
- Gambia
- Georgia
- Germany
- Ghana
- Gibraltar
- Great Britain
- Greece
- Greenland
- Grenada
- Guadeloupe (French)
- Guam (USA)
- Guatemala
- Guinea
- Guinea-Bissau
- Guyana
- Haiti
- Heard and McDonald Islands
- Holy See (Vatican City State)
- Honduras
- Hong Kong
- Hungary
- Iceland
- India
- Indonesia
- Iran
- Iraq
- Ireland
- Israel
- Italy
- Ivory Coast (Cote D'Ivoire)
- Jamaica
- Japan
- Jordan
- Kazakhstan
- Kenya
- Kiribati
- Kuwait
- Kyrgyz Republic (Kyrgyzstan)
- Laos
- Latvia
- Lebanon
- Lesotho
- Liberia
- Libya
- Liechtenstein
- Lithuania
- Luxembourg
- Macau

- Macedonia
- Madagascar
- Malawi
- Malaysia
- Maldives
- Mali
- Malta
- Marshall Islands
- Martinique (French)
- Mauritania
- Mauritius
- Mayotte
- Mexico
- Micronesia
- Moldavia
- Monaco
- Mongolia
- Montenegro
- Montserrat
- Morocco
- Mozambique
- Myanmar
- Namibia
- Nauru
- Nepal
- Netherlands
- Netherlands Antilles
- New Caledonia (French)
- New Zealand
- Nicaragua
- Niger
- Nigeria
- Niue
- Norfolk Island
- Northern Mariana Islands
- North Korea
- Norway
- Oman
- Pakistan
- Palau
- Panama
- Papua New Guinea
- Paraguay
- Peru
- Philippines
- Pitcairn Island
- Poland
- Polynesia (French)
- Portugal
- Puerto Rico
- Qatar
- Reunion (French)

- Romania
- Russian Federation
- Rwanda
- Saint Helena
- Saint Kitts & Nevis Anguilla
- Saint Lucia
- Saint Pierre and Miquelon
- Saint Tome (Sao Tome) and Principe
- Saint Vincent & Grenadines
- Samoa
- San Marino
- Saudi Arabia
- Senegal
- Serbia
- Seychelles
- S. Georgia & S. Sandwich Isls.
- Sierra Leone
- Singapore
- Slovak Republic
- Slovenia
- Solomon Islands
- Somalia
- South Africa
- South Korea
- Spain
- Sri Lanka
- Sudan
- Suriname
- Svalbard and Jan Mayen Islands
- Swaziland
- Sweden
- Switzerland
- Syria
- Tadjikistan
- Taiwan
- Tanzania
- Thailand
- Togo
- Tokelau
- Tonga
- Trinidad and Tobago
- Tunisia
- Turkey
- Turkmenistan
- Turks and Caicos Islands
- Tuvalu
- Uganda
- Ukraine
- United Arab Emirates
- United Kingdom
- United States
- Uruguay

- USA Minor Outlying Islands
- Uzbekistan
- Vanuatu
- Venezuela
- Vietnam
- Virgin Islands (British)
- Virgin Islands (USA)
- Wallis and Futuna Islands
- Western Sahara
- Yemen
- Zaire
- Zambia
- Zimbabwe

Supported Operations
No operations directly supported.

AnnualTransactionSummary

Fields
Year

`integer`

The four-digit year to which this summary applies

TransAmount

`decimal`

The total monetary amount for the period

TransCount

`integer`

The total number of transactions for the period

Supported Operations
No operations directly supported.

ApplicationSurveyResponse

Fields
SurveyId

`nonNegativeInteger`

References the system-generated value identifying the Survey to which the Survey Response applies

QuestionId

`nonNegativeInteger`

References the system-generated value identifying the Question

AnswerId

`nonNegativeInteger`

A system-generated value identifying the ApplicationSurveyResponse record

ConsId

`nonNegativeInteger`

References the system-generated value identifying the Constituent who submitted this Survey Response

Value

`string`

The response to the survey question

SubmitDate

`dateTime`

The timestamp on the Survey Response

QuestionKey

`string`

A key by which this response may be cross-referenced to other instances of the same question

Supported Operations

Find

Query

CalendarEvent

Fields

EventId

`nonNegativeInteger`

A system-generated unique identifier for the Event

Name

`string`

A descriptive name assigned to the Calendar Event by the system administrator when it was created

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Description

`string`

Description of the event

Location

`string`

Identifies the name of the place where the event will occur

CreationDate

`long`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`long`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

StartDate

`long`

The event start date

StartTime

`nonNegativeInteger`

The event start time

EndDate

`long`

The event end date

EndTime

`integer`

The event end time

EventMode

`nonNegativeInteger`

The type of event: 0 = Standard, 1 = Recurring, 2 = Multi-Day, 3 = Multi-Part, 4 = Sub-Event

AccountingCode

`string`

The accounting code for the event

Supported Operations

Find

Query

CalendarEventRSVP

Fields

RSVPId

`nonNegativeInteger`

A system-generated unique identifier for the RSVP

EventId

`nonNegativeInteger`

The unique ID of the Event

NumberOfAttendees

`nonNegativeInteger`

Number attending with respondent

OriginalNumberOfAttendees

`nonNegativeInteger`

Number expected or invited with respondent

ShowInGuestList

`boolean`

Indicates whether to show the respondent in the event guest list

IsOfflineRegistration

`boolean`

Indicates whether the registration occurred online or offline

LineItemAmount

`decimal`

Total amount of the event ticket price

LineItemDeductibleAmount

`decimal`

Tax deductible amount of the event ticket price

CreationDate

`dateTime`

Timestamp indicating when the record was created

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

Attendee

`Constituent`

The Constituent Record of the Attendee

Payment

`Payment`

Event admission payment

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

Center

Fields

Id

`integer`

System-generated unique identifier for the record

SecurityCategoryId

`integer`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

MemberGroupId

`integer`

The ID for the group defining center members

AdminGroupId

`integer`

The ID for the administrative security group defining center administrators

Name

`string`

Name of the center (internal)

PublicName

`string`

Name of the center (public)

Description

`string`

Description of the center

Status

`string`

Center status: may be be 'active', 'inactive', or 'disabled'

HomePage

`string`

URL of the center's home page

NotFoundPage

`string`

URL of the 404 - Page Not Found page for the center web site

AffiliateCode

`string`

Client-defined affiliate code for the center

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Supported Operations

Find

Query

CenterMember

A record of constituent, organization member or contact's membership to a Center.

A center member record represents membership of a constituent (in Convio database) to a center.

DataSync Web Services supports query and update operations on CenterMember records in Convio database. A CenterMember record cannot be added to or deleted from Convio database using a DataSync Web Services operation on a CenterMember record. The only information about a CenterMember record that can be modified or updated using a DataSync Web Servicse operation on a CenterMember record is the constituent's email opt-in preference for the center. A constituent's email opt-in preference for a center is tracked via the OptIn field in a CenterMember record. A constituent's email opt-in preference for a center can only be set to 'False' using a DataSync Web Service operation on a CenterMember record.

When you change the constituent's email opt-in preference for the center using DataSync Web Services operation, that constituent's interest opt-in preferences for the center are not changed. As a result, if the constituent is opted into email for the center again, their interest opt-in's preferences for the center are again effective.

If you want to run any of the following operations:

1. Add a Constituent to a Center
2. Remove a Constituent from Center
3. Change Interest Opt-in preference for a Center Member

please use the appropriate DataSync Web Services operation on a Constituent record. (Refer to the documentation.)

Fields

CenterId

`nonNegativeInteger`

Unique identifier of the Center Record.

ConsId

`nonNegativeInteger`

Unique identifier of the Constituent Record.

OptIn

`boolean`

Boolean flag indicating constituent's email opt-in preference for the center. This field takes value of either 'True' or 'False'.

OptInDate

`dateTime`

Date and time when constituent's email OptIn flag was last set to 'True'.

OptInBy

`nonNegativeInteger`

Identifier of the constituent who changed the email OptIn flag for this record to 'True'.

OptOutDate

`dateTime`

Date and time when constituent's email OptIn flag was last set to 'False'.

OptOutBy

`nonNegativeInteger`

Identifier of the constituent who changed the email OptIn flag for this record to 'False'.

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

Supported Operations

Update

Find

Query

Config

Fields

Value

`string`

ConfigId

`string`

Supported Operations

No operations directly supported.

Constituent

A constituent, contact or organization member

A constituent record represents a member, donor, subscriber or other type of individual whose information is tracked in the Convio database. A constituent record may include fields such as:

1. name
2. phone number(s)
3. home address
4. political districts (i.e. senator(s), representative)
5. employment information (i.e. name of company, work address, work phone)
6. educational background (i.e. name of college, year of graduation)
7. relationships to other constituents (i.e. spouse, friend, boss)
8. donation summary (i.e. total amount donated last year)
9. interests (i.e. baby seals, global warming)
10. group affiliations designated by the organization (i.e. Major Donors)
11. custom fields defined by the organization (i.e. Political Party Affiliation)

Some fields in the Convio constituent record are populated by the system and may not be modified by the client. For example, the system determines the congressional district automatically based on the constituent's street address. Other Constituent fields which are read-only via Web Services API are:

- AffiliateAssignment
- MatchedDonor
- EmailOptInStatus
- EmailStatus
- HardBounceCount
- SoftBounceCount
- EngagementFactor1-10

Most fields in the constituent record have a single value. However, a constituent may belong to any number of groups and have any number of interests.

DataSync Web Services supports bidirectional synchronization of constituent records: when a record is added, updated or deleted in the client system, the web services can be used to make the same change in the Convio database.

Conversely, when a record is added, updated or deleted in the Convio database (i.e. due to an online registration or profile change), the web services can be used to retrieve the change so that it can be made in the client system as well.

Fields

ConsId

`nonNegativeInteger`

The unique Convio identifier for constituent records.

SiteId

`nonNegativeInteger`

The unique Convio identifier for your site. Will have the same value for all records.

AdminStatus

`AdministratorStatus`

Indicates whether this Constituent is an Administrator in the Convio system

ConsName

`Name`

Constituent's name

UserName

`normalizedString`

Constituent's user name, supplied when logging into the Convio system

CatId

`normalizedString`

Constituent's security category

MemberId

`normalizedString`

Unique ID assigned to the constituent by the client's registration system (client-defined). This field may be specified when a new record is created, or may be updated in an existing record. In the latter case, you must also provide a ConsId so the Constituent in question can be uniquely identified.

AffiliateCode

`normalizedString`

AffiliateAssignment

`AffiliateAssignment`

ExchangeId

`normalizedString`

OriginSrcCode

`normalizedString`

Source code for the Constituent record

OriginSubsrcCode

`normalizedString`

Sub-source code for the Constituent record

Active

`ActiveStatus`

Indicates whether the Constituent is registered user who can log in to the site

ActiveDetail

ActiveDetail

Flag providing additional detail on the Constituent's status; One of the following: Undefined, Unknown to Contact, Known to Contact, Accepted by Contact, Registered Contact, New Prospect, Interested Prospect, Removed by Administrator, Removed by Contact, Removed as Duplicate

Origin

Origin

Convio feature through which Constituent initiated registration

OriginRefId

`integer`

DonorStatus

DonorStatus

Whether this Constituent has donated money to the site: Never, Undefined, Indirect, Direct, Unknown, Non-donor

MatchedDonor

`boolean`

Indicates whether any gifts made by this person are matched by an employer

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`integer`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`integer`

Constituent who last modified the record

LastLoginDate

`dateTime`

Date Constituent last logged in

UserPassword

`normalizedString`

Hexadecimal representation of Constituent's encrypted password, supplied when logging into the Convio system

LastPasswordChangeDate

`dateTime`

Date the Constituent last changed his or her login password

PasswordType

PasswordType

The method used to generate the password: Undefined, DES (in the Convio database, DES encrypted), LDAP (in an external database), Convio SHA (SHA-256 encrypted), Convio RSA

ReminderHint

`normalizedString`

Constituent's login password reminder hint

RegOptInNonce

`decimal`

Cookie

`normalizedString`

CookieSet

`boolean`

UploadChecksum

`dateTime`

Optionally stores a checksum for a set of fields when uploading Constituent data. Note the actual maximum value of this column is the same as a signed 64 bit number.

DownloadChecksum

`dateTime`

Optionally stores a checksum for a set of fields when downloading Constituent data. Note the actual maximum value of this column is the same as a signed 64 bit number.

DatasyncDate

`dateTime`

HomePhoneImportId

`normalizedString`

WorkPhoneImportId

`normalizedString`

MobilePhoneImportId

`normalizedString`

FaxPhoneImportId

`normalizedString`

PrimaryEmailImportId

`normalizedString`

SecondaryEmailImportId

`normalizedString`

HomeAddrImportId

`normalizedString`

WorkAddrImportId

`normalizedString`

OtherAddrImportId

`normalizedString`

EmployerAddrImportId

`normalizedString`

PrimaryBusinessOrgImportId

`normalizedString`

PrimaryBusinessRelLinkId

`normalizedString`

SoImportId

`normalizedString`

IndividualRelLink

`normalizedString`

ConsGender

`normalizedString`

Constituent's gender

BirthDate

`dateTime`

Constituent's date of birth

PrimaryEmail

`normalizedString`

Constituent's primary email address

AcceptsEmail

`boolean`

Indicates whether the Constituent will accept email from the site

PreferredEmailFormat

`EmailFormat`

Setting that determines if a participant has selected to receive emails that are HTML-formatted or in plain text

EmailOptOutNonce

`decimal`

EmailOptOutDate

`dateTime`

Timestamp that shows when a participant disabled the email Opt-In

EmailOptOutMethod

`OptOutMethod`

EmailOptOutRefId

`integer`

EmailStatus

`EmailStatus`

May be one of: Unknown, Good, Bad (soft bounce), Bad (hard bounce); set by the Email system based on results of Email Campaigns. Changes to this field by the system do not cause the record to appear in GetIncrementalUpdates.

HardBounceCount

`integer`

Count of the number of emails sent back as undeliverable because of an invalid email address

SoftBounceCount

`integer`

Total number of system emails returned to this user due to temporary mail server errors, such as Inbox Full. Emails returned due to soft bounce are re-tried by the system for two days. If the error condition persists, the Constituent's EmailStatus is updated and this count is incremented.

HomeAddress

`Address`

Constituent's home mailing address

StdHomeAddress

`Address`

The Constituent's home address, used by the Advocacy application as the Constituent's legal place of residence when determining representation districts

StdHomeAddressStatus

`AddressStatus`

One of: 0=Unknown (as entered), 1=Validated

StdHomeAddressModifyDate

`dateTime`

When the Constituent's home address record was last modified

WorkAddress

`Address`

Constituent's work address

OtherAddress

`Address`

Alternate mailing address for the Constituent

PreferredAddress

`AddressType`

Constituent's preferred mailing address

AcceptsPostalMail

`boolean`

Indicates whether the Constituent will accept postal mail

CongDistId

`normalizedString`

Identifier used by Advocacy for the U.S. House of Representatives district corresponding to the zip code of the Constituent's place of legal residence (StdHomeAddress)

CongDistIdOverride

`normalizedString`

The U.S. House of Representatives district ID for the Constituent manually entered (overrides CongDistId)

StateSenateDistId

`normalizedString`

Identifier used by Advocacy for the state Senate district corresponding to the zip code of the Constituent's place of legal residence (StdHomeAddress)

StateSenateDistIdOverride

`normalizedString`

The state Senate district ID for the Constituent manually entered (overrides StateSenateDistId)

StateHouseDistId

`normalizedString`

Identifier used by Advocacy for the state House of Representatives district corresponding to the zip code of the Constituent's place of legal residence (StdHomeAddress)

StateHouseDistIdOverride

`normalizedString`

The state House of Representatives district ID for the Constituent manually entered (overrides StateHouseDistId)

HomeCounty

`normalizedString`

Constituent's home county

HomeCountyOverride

`normalizedString`

Constituent's home county (manually entered, overrides HomeCounty

HomePhone

`normalizedString`

Constituent's home telephone number

PreferredPhone

`PhoneType`

Constituent's preferred phone number

EmployerAddress

`Address`

Business address of Constituent's employer.

Employer

`normalizedString`

Name of Constituent's employer

ConsOccupation

`normalizedString`

Constituent's occupation

ConsHighSchoolName

`normalizedString`

Constituent's secondary school name

ConsHighSchoolClass

`integer`

Constituent's secondary school graduating class year

ConsSchool1

`PostSecondarySchool`

Constituent's college or university (up to 6)

ConsSchool2

PostSecondarySchool

Constituent's college or university (up to 6)

ConsSchool3

PostSecondarySchool

Constituent's college or university (up to 6)

ConsSchool4

PostSecondarySchool

Constituent's college or university (up to 6)

ConsSchool5

PostSecondarySchool

Constituent's college or university (up to 6)

ConsSchool6

PostSecondarySchool

Constituent's college or university (up to 6)

SoName

Name

Name of the Constituent's spouse or partner

Relationship1

Relationship

A person related to the Constituent (up to 8)

Relationship2

Relationship

A person related to the Constituent (up to 8)

Relationship3

Relationship

A person related to the Constituent (up to 8)

Relationship4

Relationship

A person related to the Constituent (up to 8)

Relationship5

Relationship

A person related to the Constituent (up to 8)

Relationship6

Relationship

A person related to the Constituent (up to 8)

Relationship7

Relationship

A person related to the Constituent (up to 8)

Relationship8

Relationship

A person related to the Constituent (up to 8)

FirstTransaction

TransactionSummary

LastTransaction

TransactionSummary

A transaction summary of this Constituent's most recent monetary transaction

LargestTransaction

TransactionSummary

A transaction summary of this Constituent's largest monetary transaction

PrevYearTotal

AnnualTransactionSummary

Constituent's total donations during the previous calendar year

CurrYearTotal

AnnualTransactionSummary

LifetimeTotalTransAmount

`decimal`

Total monetary amount of transactions (donations, etc.) completed since the Constituent record was created

LifetimeTotalTransCount

`integer`

Total number of transactions completed since the Constituent record was created

TransRecencyPercentile

`decimal`

Percentile comparing how recently this constituent has donated compared to all other constituents

TransFrequencyPercentile

`decimal`

Percentile comparing how often this constituent donates, compared to all other constituents

TransMonValPercentile

`decimal`

Percentile comparing how much money this constituent has donated compared to all other constituents

ExternalFirstGiftDate

`dateTime`

ExternalFirstGiftAmount

`decimal`

ExternalLastGiftDate

`dateTime`

ExternalLastGiftAmount

`decimal`

ExternalLargestGiftDate
`dateTime`

ExternalLargestGiftAmount
`decimal`

ExternalYtdGiftAmount
`decimal`

ExternalYtdGiftCount
`integer`

ExternalLifetimeGiftAmount
`decimal`

ExternalLifetimeGiftCount
`integer`

EngagementFactor1
`decimal`

EngagementFactor2
`decimal`

EngagementFactor3
`decimal`

EngagementFactor4
`decimal`

EngagementFactor5
`decimal`

EngagementFactor6
`decimal`

EngagementFactor7
`decimal`

EngagementFactor8
`decimal`

EngagementFactor9
`decimal`

EngagementFactor10
`decimal`

PrevYearAlertResponseCount
`integer`

Number of alerts taken by the user in the previous year

CurrYearAlertResponseCount
`integer`

The number of Advocacy alerts the Constituent has responded to during the current calendar year

LifetimeAlertResponseCount

`integer`

Total responses to Advocacy alerts the Constituent has made since first interacting with this site

GroupId

`nonNegativeInteger`

Multiple values allowed

References the system-generated value identifying the primary Group to which this Constituent is assigned

InterestId

`nonNegativeInteger`

Multiple values allowed

References the Interest Category to which this Constituent belongs

CenterId

`nonNegativeInteger`

Multiple values allowed

The Center with which this constituent is associated (only used when Multi-Center support is installed)

Supported Operations

Create

Update

Delete

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

Find

Query

CrossReferenceType

Fields

SiteId

`nonNegativeInteger`

XrefTypeId

`nonNegativeInteger`

XrefType

`string`

Supported Operations

Find

Query

CrossReferenceValue

Fields

SiteId

`nonNegativeInteger`

XrefTypeId

`nonNegativeInteger`

XrefTypeName

`string`

ConvioKey

`string`

ForeignKey

`string`

ForeignKey2

`string`

ForeignKey3

`string`

ForeignKey4

`string`

ForeignKey5

`string`

ForeignKey6

`string`

ForeignKey7

`string`

ForeignKey8

`string`

ForeignKey9

`string`

ForeignKey10

`string`

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

DesignatedTransaction

Fields

TransactionId

`nonNegativeInteger`

DesignationTypeId

`nonNegativeInteger`

Unique ID of the DesignationType record associated with this transaction

DesigneeId

`nonNegativeInteger`

Unique ID of the Designee record associated with this transaction (not a write-in)

Amount

`decimal`

Monetary amount of the transaction

AgencyName

`string`

Name of the designee for a write-in designation

AgencyContactInfo

`string`

Contact information for a write-in designation

Payment

`Payment`

Payment record for the gift

Donor

`Constituent`

Constituent record of the donor

Supported Operations

Find

Query

Designee

Fields

DesigneeId

`nonNegativeInteger`

System-generated unique identifier for the record

DesignationTypeId

`nonNegativeInteger`

Unique ID of the DesignationType record associated with this designee

Name

`string`

Name of the designee (internal)

PublicName

`string`

Name of the designee (public)

AdminCategory

`integer`

Unique identifier of the Administrative Security Category for this designee

Status

`string`

Designee status, may be one of: 'archived', 'active', 'write_in'

AccountingCode

`string`

Client-defined accounting code for this designee (internal)

ThresholdAmount

`decimal`

Designations above this amount trigger email notification and/or group membership as defined in EmailNotification and GroupId fields

EmailNotification

`string`

Email recipients to be notified whenever a designation exceeds ThresholdAmount

GroupId

`nonNegativeInteger`

Group to which constituents will be assigned if their designation exceeds ThresholdAmount

InterestId

`nonNegativeInteger`

Unique ID of an Interest Category associated with this designee

SortOrder

`nonNegativeInteger`

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Supported Operations

Find

Query

Discount

Fields

Total

`decimal`

DiscountIds

`string`

Supported Operations

No operations directly supported.

Donation

Fields

TransactionId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

NewDonorStatus

`NewDonorStatus`

TaxDeductibleValue

`decimal`

The monetary amount that the donor may deduct for tax purposes

FairMarketValue

`decimal`

The fair-market monetary value of a non-cash donation or donation in kind, if applicable, for tax purposes

AdditionalAmount

`decimal`

An additional amount a donor chooses to give above and beyond the preset donation level

CampaignId

`nonNegativeInteger`

References the system-generated value identifying the Campaign which generated the Donation

DonationLevelId

`nonNegativeInteger`

References the system-generated value identifying the predefined Donation Level chosen by the donor

FormId

`integer`

References the system-generated value identifying the predefined Donation Form through which the donation was entered

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Payment

`Payment`

Payment information for the Donation

Donor

`Constituent`

The record of the Constituent credited with the Donation

InteractionSource

`InteractionSource`

Interaction source information for this Donation

MatchingGift

`MatchingGift`

Employer matching gift information for this Donation, if applicable

Recognition

`Recognition`

Recognition information for this Donation

RecurringPayment

`RecurringPayment`

Recurring payment information for this Donation

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

DonationCampaign

Fields

CampaignId

`nonNegativeInteger`

A system-generated unique identifier for the Campaign

Title

`string`

A descriptive name assigned to the Donation Campaign by the system administrator when it was created

Description

`string`

Description of the campaign

Type

`string`

Unique ID of a CampaignType record associated with this campaign

IsPublished

`boolean`

Campaign status: true if published

IsArchived

`boolean`

Campaign status: true if archived

Goal

`decimal`

Campaign fundraising goal

MaxDonorEnteredAmount

`decimal`

The maximum amount allowed on a donation form for this campaign

CurrentlyRaisedAmount

`decimal`

Current total donations to this campaign

LastGiftAmount

`decimal`

Amount of the most recent gift

LastGiftDate

`dateTime`

Date of the most recent gift

FundraisingGroupId

`nonNegativeInteger`

Unique ID of the Group for donors to this campaign

LargeGiftGroupId

`nonNegativeInteger`

Unique ID of the Group for donors to this campaign whose gifts exceed a threshold amount

AdministratorId

`nonNegativeInteger`

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

DonationForm

Fields

FormId

`nonNegativeInteger`

A system-generated unique identifier for the Donation Form

CampaignId

`nonNegativeInteger`

References the system-generated value identifying the Campaign to which this Donation Form applies

Title

`string`

A descriptive name assigned to the Donation Form by the system administrator when it was created

Description

`string`

Description of this donation form

IsPublished

`boolean`

Form status: true if published

IsArchived

`boolean`

Form status: true if archived

IsOffline

`boolean`

Rrue if offline donations can be recorded through this form

PublishDate

`dateTime`

Publication date of the form

UnpublishDate

`dateTime`

Expiration date of the form

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

AdminSecurityCategoryId

`nonNegativeInteger`

Unique ID of the Administrative Security Category associated with this donation form

UserSecurityCategoryId

`nonNegativeInteger`

Unique ID of the User Security Category associated with this donation form

Supported Operations

Find

Query

DonationLevel

Fields

DonationLevelId

`nonNegativeInteger`

A system-generated unique identifier for the Donation Level

FormId

`integer`

References the system-generated value identifying the Donation Form for this Donation Level

Name

`string`

A descriptive name assigned to the Donation Level by the system administrator when it was created

Description

`string`

Description of the Donation Level

Type

`DonationLevelType`

The type of Donation Level record: Fundraising, Recurring, or Both.

ValueOfGoods

`decimal`

Dollar value, in decimal format, if the value of the goods associated with the level

PromotionCode

`string`

A code that registrants can enter to receive promotional discounts or options

GroupId

`nonNegativeInteger`

References the system-generated value identifying the Group to which donors at this Donation Level are assigned

Value

`decimal`

The monetary value of the recommended gift associated with this Donation Level

DiscountCode

`nonNegativeInteger`

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

Group

Fields

Id

`integer`

A system-generated unique identifier for this Group

Name

`string`

A descriptive name assigned to the Group by the system administrator when it was created

TypeId

`integer`

References the system-generated value identifying the Group Type for this Group

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

GroupType

Fields

Id

`nonNegativeInteger`

A system-generated unique identifier for this Group Type

Name

`string`

A descriptive name assigned to the GroupType by the system administrator when it was created

Supported Operations

Find

Query

InteractionSource

Fields

CodeType

`SourceCodeType`

The type of interaction source:

- 0 = Other (passed via origin override)
- 1 = Link with source code value
- 2 = Tell-A-Friend
- 3 = E-mail (original
- 4 = E-mail (forwarded)
- 5 = eCard
- 6 = Referer
- 7 = Personal Outreach

Code

`string`

The primary code designating a point of interaction with a constituent:

- is the origin value for type 0
- is 0 for type 1 (only the text fields are used)
- points to a specific mail user ID for type 2
- points to a specific mail message ID for types 3-4
- points to eCard campaign ID for type 5
- is 0 for type 6
- is the outreach goal id for type 7

CodeText

`string`

A printable version of source code:

- For type 0, it is the same as the number
- For type 1, it is the value passed in as the s_src argument

- For type 2, it is the forwarding user's name
- For types 3 and 4, it is the name of the message
- For type 5, it is the name of the eCard campaign
- For type 6, it is the hostname of the referer

SubCode

`string`

A sub-code designating a point of interaction with a constituent:

- is the origin_ref_id value for type 0
- is 0 (only the text fields are used) for type 1
- is 0 for type 2
- points to a specific redirectorId for types 3-4
- points to a stationery layout (card type) for type 5
- is 0 for type 6
- is the outreach component id for type 7

SubCodeText

`string`

A printable version of the sub-source:

- For type 0, it is the same as the number
- For type 1, it is the value passed in as the s_subsrc argument
- For type 2, it is the blank
- For types 3 and 4, it is the URL of the link
- For type 5, it is the name of the Stationery Layout
- For type 6, it is the path of the referer unless it is a known search engine, in which case it will be the search terms

Supported Operations

No operations directly supported.

Interest

Fields

Id

`integer`

A system-generated unique identifier for the Interest category

Name

`string`

A descriptive name assigned to the Interest category by the system administrator when it was created

Description

`string`

Descrtiption of the Interest category

ParentId

`nonNegativeInteger`

References a parent Interest category in a family of related Interests, if applicable

IsEmail

`ZeroOrOneBoolean`

Email interests can be associated with email campaigns for each interest giving constituents the ability to opt-in or out of the specific types of communications they want

IsWebsite

`ZeroOrOneBoolean`

Website Interests are associated with news articles, photo albums, calendar events, TeamRaisers, and surveys. Administrators can use them to customize what content specific constituents see based on their interest categories. The system can also associate web interests with Products and use the association for cross-selling.

IsDefault

ZeroOrOneBoolean

If true, indicates this is a default interest for all constituents; that is, all constituents will have this interest selected by default, though they may override the selection.

OptInGroupId

nonNegativeInteger

References the system-generated value identifying the Opt-In Group for email associated with this Interest

OptOutGroupId

nonNegativeInteger

References the system-generated value identifying the Opt-Out group for email associated with this Interest

SecurityCategoryId

nonNegativeInteger

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

InterestEvent

Fields

EventId

integer

A system-generated unique identifier for the transaction

InterestId

integer

References the system-generated value identifying the Interest category associated with this Interest Event

ConsId

integer

References the system-generated value identifying the Constituent associated with this InterestEvent (for "opt-in" or "opt-out" type events)

EventType

InterestEventType

One of: "opt-in", "opt-out", or "totals"

CreationDate

dateTime

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

For "totals" type records, references the system-generated value identifying the Administrator who created the record

Supported Operations

Find

Query

Item

Fields

ItemId

`nonNegativeInteger`

A system-generated unique identifier for the Item

ProductId

`nonNegativeInteger`

References the system-generated value identifying the Product associated with this Item

Price

`nonNegativeInteger`

Price of the Item

ShippingCharge

`decimal`

Shipping charge for the Item

DiscountAmount

`decimal`

Monetary amount subtracted from the order total when a purchaser applies a discount code

ExpressShipping

`boolean`

Indicates whether this Item qualifies for express shipping

IsPremium

`boolean`

Indicates whether this Item is a premium for a Donation Level

ExternalIdentifier

`string`

SkuNumber

`string`

Item's designated Stock Keeping Unit number

TributeType

`string`

Text describing an Item sent in response to a tribute gift, such as "In Honor of," "In Memory of," "In Celebration of," "In Tribute to," or "In Recognition of"

TributeName1

`string`

Name of first honoree of a tribute gift Item

TributeName2

`string`

Name of second honoree of a tribute gift Item

TributeName3

`string`

Name of third honoree of a tribute gift Item

TributeMessage1

`string`

Text of a tribute message associated with this item

TributeMessage2

`string`

Text of a tribute message associated with this item

TributeClosing

`string`

Phrase to precede the signature to express the regard for the person in whose honor a gift is being made

TributeSignature1

`string`

Signature text for the first donor

TributeSignature2

`string`

Signature text for the first donor

TributeSignature3

`string`

Signature text for the third donor

Taxable

`boolean`

Indicates whether sales tax applies to this item

FairMarketValue

`decimal`

The fair market value of this item for tax purposes

TaxDeductibleValue

`decimal`

Monetary amount that the donor may deduct for tax purposes

Config

`Config`

Multiple values allowed

Item configuration

Supported Operations

No operations directly supported.

MatchingGift

Fields

EmployerName

`string`

Name of employer that will match a donor's gift

Supported Operations

No operations directly supported.

Name

Fields

Title

`string`

A person's form of polite address, such as Mr. or Mrs.

FirstName

`string`

A person's first name

MiddleName

`string`

A person's middle name or initial

LastName

`string`

A person's surname

Suffix

`string`

A qualifier that follows a person's surname, if applicable, such as Jr., Sr. or III

ProfSuffix

`string`

A person's professional title, if applicable, such as M.D. or Ph.D.

Supported Operations

No operations directly supported.

Partition

Fields

SiteId

`nonNegativeInteger`

References the system-generated value identifying the Convio client site for the database Partition

PartitionId

`nonNegativeInteger`

A system-generated unique identifier for the database Partition

Name

`string`

The name of the database Partition

Supported Operations

Find

Query

Payment

Fields

Amount

`decimal`

Monetary amount of the Payment

PaymentDate

`dateTime`

Date of the Payment

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

CreationMethod

`CreationMethod`

One of the following values that determines how this transaction was processed: 0 if entered in the database as an offline transaction (for example, an offline gift or registration fee), 1 if processed online, 2 if from a recurring pledge

BillingName

`Name`

Name on the credit card used to make the online transaction, for the bank account withdrawal, or entered by the administrator when recording or processing the transaction offline

BillingAddress

`Address`

Billing address of the party making the Payment

ConfirmationCode

`string`

Code supplied by the financial transaction processor (credit card center) to the system confirming the transaction

ConvioConfirmationCode

`string`

System-generated transaction confirmation code provided to the donor confirming the transaction

ProcessorTransactionId

`string`

Transaction ID supplied to the system by the financial institution (credit card center or bank)

ProcessorRefNumber

`string`

Processor-supplied reference number for the payment transaction

TenderType

`TenderType`

Indicates the method of payment

CreditCardType

`CreditCardType`

Indicates the type of credit card (for credit card payment)

CreditCardNumber

`string`

The card number (for credit card payment)

CreditCardExpDate

`string`

The expiration date (for credit card payments)

BatchSettlementNumber

`string`

The batch settlement number received from the financial institution when the payment was settled

SettlementDate

`dateTime`

Date on which this transaction was settled

SettlementStatus

`SettlementStatus`

The step in the payment and settlement process that this transaction is in: Settled, Unsettled, Submitted, ACH only

MerchantAccountId

`nonNegativeInteger`

The system-generated unique identifier of the Merchant Account associated with this Payment

CheckNumber

`string`

Check number for payment by check

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

RoutingNumber

`string`

ACH Routing number

TransitNumber

`string`

ACH transit number (Canada)

BankNumber

`string`

ACH bank number (Canada)

AccountType

`string`

ACH account type: CHECKING (default), SAVINGS

AccountNumber

`string`

ACH (debit) account number

OriginationType

`OriginationType`

Payment origination: entered by a consitituent or administrator online, posted via an internal or external API call, or entered by an administrator in CRM.

CustomString1

`string`

CustomString2

`string`

CustomString3

`string`

CustomString4

`string`

Discount

`Discount`

Supported Operations

No operations directly supported.

PersonalFundraisingCampaign

Fields

CampaignId

`nonNegativeInteger`

A system-generated unique identifier for the Personal Fundraising Campaign

Title

`string`

A descriptive name assigned to the Personal Fundraising Campaign by the organizer

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

PersonalFundraisingChampion

Fields

RegistrationId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

CampaignId

`nonNegativeInteger`

>References the system-generated value identifying the Campaign associated with the transaction

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Payment

`Payment`

Payment information for this transaction

Participant

`Constituent`

The constituent record of the participant

InteractionSource

`InteractionSource`

Interaction source information for the transaction.

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

Find

Query

PersonalFundraisingGift

Fields

GiftId

`nonNegativeInteger`

A system-generated unique identifier for the gift

CampaignId

`nonNegativeInteger`

References the system-generated value identifying the Personal Fundraising Campaign which generated this Gift

TransactionId

`nonNegativeInteger`

References the unique identifier for the ValuedTransaction record associated with this gift

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Payment

`Payment`

Payment information for this transaction

Donor

`Constituent`

The constituent record of the donor

InteractionSource

`InteractionSource`

Interaction source information for the transaction

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

PostSecondarySchool

Fields

Name

`string`

Name of the college or university

Type

`string`

Type of institution

ClassYear

`integer`

Graduating class year as recognized by the institution

Degree

`string`

Degree awarded

Major

`string`

Major course of study

Conferred

`boolean`

Indicates whether a degree was conferred

PreferredClass

`integer`

Preferred graduating class year as indicated by the constituent

Minor

`string`

Minor course of study

Concentration

`string`

Academic area of concentration

Department

`string`

Name of department or college

Supported Operations

No operations directly supported.

Product

Fields

ProductId

`nonNegativeInteger`

A system-generated unique identifier for the Product

Title

`string`

A descriptive name assigned to the Product by the system administrator when it was created

ShortDescription

`string`

The short description of the Product assigned by the system administrator when it was created

LongDescription

`string`

The full description of the Product

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

ProductOrder

Fields

SiteId

`nonNegativeInteger`

References the system-generated value identifying the eCommerce Store from which the Product was ordered

OrderId

`nonNegativeInteger`

A system-generated unique identifier for the Order

TransactionId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

PurchaseAmount

`decimal`

The monetary amount charged for the purchase of products

TaxDeductibleValue

`decimal`

The monetary amount that the person performing the transaction can deduct for tax purposes

ShippingCharge

`decimal`

The monetary amount of the charge for shipping the order

AdditionalDonation

`decimal`

The monetary amount of any additional donation included with this order, above and beyond the base price of the products

DiscountAmount

`decimal`

The monetary amount subtracted from the product order price when the purchaser applies a discount code

DiscountCode

`string`

The discount code, if any, supplied by the purchaser when placing this order

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Payment

`Payment`

Payment information for this transaction

Purchaser

`Constituent`

The constituent record of the purchaser

InteractionSource

`InteractionSource`

Interaction source information for the transaction

ShipTo

`ShipTo`

Multiple values allowed

Shipping address information for the order

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

Recognition

Fields

IsAnonymous

`ZeroOrOneBoolean`

Indicates whether the donor is to be identified in the recognition

Type

`RecognitionType`

Type of recognition

JointName

`string`

Full name of the person recognized as a donor who is not directly making the gift payment (the joint donor)

JointTitle

`string`

Form of address prefixed to name of the joint donor

JointFirstName

`string`

First name of the joint donor

JointMiddleName

`string`

Middle initial or name of the joint donor

JointLastName

`string`

Last name of the joint donor

JointSuffix

`string`

If applicable, a name qualifier or professional title such as Jr., Sr., M.D., or Ph.D. to follow the surname

TributeTitle

`string`

Form of polite address to precede the first name, such as Mr., Mrs. or Dr., of the person in whose honor the gift is made

TributeFirstName

`string`

First name of the person in whose honor a gift is being made

TributeLastName

`string`

Last name of the person in whose honor a gift is being made

TributeMiddleInitial

`string`

Middle initial or the person in whose honor a gift is being made

TributeClosing

`string`

Phrase selected to precede the signature to express regard for the person in whose honor a gift is being made

TributeSignature

`string`

Signature of the sender's message

TributeMessage

`string`

Text of the sender's message

SenderName

`Name`

Name of individual or organizational representative sending the recognition

SenderAddress

`Address`

Email address of individual or organizational representative whose email address will be displayed as the sender's

SenderPhone

`string`

Sender's telepone number

SenderEmail

`string`

Sender's email address

IsAmountSent

`ZeroOrOneBoolean`

Indicates whether the amount of the donation is to be included in the recognition message

Supported Operations

No operations directly supported.

Record

Fields

This type has no fields.

RecurringPayment

Fields

Type

`RecurringPaymentType`

Type of recurring payment

OriginalTransactionId

`nonNegativeInteger`

Transaction ID of the original payment transaction in a series of recurring payments

InitialAmount

`decimal`

Amount of the first payment of the series

RecurringAmount

`decimal`

Amount of subsequent payments in the series

Frequency

`nonNegativeInteger`

The intervening time between recurring payments

Duration

`nonNegativeInteger`

The range of time across which recurring payments will be made

NumberOfPayments

`nonNegativeInteger`

The number of payments made

NumberOfFailedPayments

`nonNegativeInteger`

Number of payments not collected from a series of recurring payments

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

No operations directly supported.

Refund

Fields

RefundId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

RefundAdminId

`nonNegativeInteger`

References the system-generated value identifying the Administrator who created the Refund record

OriginalTransactionId

`nonNegativeInteger`

References the system-generated value identifying the original transaction to which the Refund applies

Payment

`Payment`

Payment information for this transaction

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

Find

Query

Relationship

Fields

FirstName

`string`

First name of Constituent's relative

LastName

`string`

Last name of the Constituent's relative

Type

`string`

Relationship between the Constituent and this relative

Gender

`string`

Gender of Constituent's relative

BirthDate

`dateTime`

Birth date of Constituent's relative

Alumnus

`boolean`

Indicates whether the Constituent's relative is an alumnus if this field is used

Supported Operations

No operations directly supported.

ShipTo

Fields

Number

`nonNegativeInteger`

System-generated unique identifier for the Ship-To record

Title

`string`

Addressee's title of polite address, such as Mr. or Mrs.

FirstName

`string`

First name of the addressee

MiddleName

`string`

Middle name of the addressee

LastName

`string`

Last name of the addressee

Email

`string`

Addressee-provided email address

Street1

`string`

Street number and name

Street2

`string`

Additional address information

City

`string`

Name of the city

Stateprov

`string`

Name or abbreviation of the state or province

Zip

`string`

Zip code or postal code

Country

`string`

Name of the country

GiftNote

`string`

Text of a note to include with the gift

GiftSignature

`string`

Text of the signature for the gift note

ExpressShipping

`boolean`

Indicates whether to use express shipping

PhoneNumber

`string`

Telephone number of recipient

Item

`Item`

Multiple values allowed

References the Item shipped

Supported Operations

No operations directly supported.

Survey

Fields

SurveyId

`nonNegativeInteger`

A system-generated unique identifier for the Survey

Type

`SurveyType`

1 = Standard Survey, 2 = SurveyComponent Survey (Event, FriendRaiser)

Title

`string`

A descriptive name assigned to the Survey by the system administrator when it was created

StartDate

`dateTime`

The target date for the Survey to be published

StopDate

`dateTime`

The date on which the Survey will expire

ResponseCount

`nonNegativeInteger`

The total number of responses to the Survey

MultipleResponsesAllowed

`ZeroOrOneBoolean`

Determines whether a Constituent may respond to the survey more than once

SecurityCategoryId

`nonNegativeInteger`

References the system-generated value identifying a Security Category controlling which constituents can see the Survey

GroupId

`nonNegativeInteger`

References the system-generated value identifying a Group to which responders to this survey will be assigned

Question

`SurveyQuestion`

Multiple values allowed

The question posed by the survey

Supported Operations

Find

Query

SurveyAnswer

Fields

QuestionId

`nonNegativeInteger`

References the system-generated value identifying the Survey Question which prompted this Answer

Value

`string`

The text of the survey Answer

Supported Operations

No operations directly supported.

SurveyQuestion

Fields

QuestionId

`nonNegativeInteger`

A system-generated unique identifier for the Survey Question

SortOrder

`nonNegativeInteger`

Ordinal indicating this Question's position in the Survey

Type

`string`

Defines the type of responses to select or enter for a question (note that Captions have no responses associated with them):

- Caption
- Constituent Registration Info
- Date Value
- Hidden Interest Update
- Hidden Text Update
- Hidden True/False Update
- Interest Categories
- Multiple Choice Combo Box
- Multiple Choice Multiple Response
- Multiple Choice Single Response (Drop Down)
- Multiple Choice Single Response (Radio)
- Numeric Value Rating Scale
- Short Text Value (40 characters)
- Text Value
- True/False
- Unlimited Text Value
- Yes/No
- CAPTCHA

Text

`string`

The text of the Survey Question

AnswerRequired

`ZeroOrOneBoolean`

Indicates whether the question must be answered to successfully submit a Survey response

Option

SurveyQuestionOption

Multiple values allowed

List of possible responses to a multiple-choice question

Supported Operations

No operations directly supported.

SurveyQuestionOption

Fields

Value

`string`

The value returned in the survey answer if this option is selected

Label

`string`

The text of the option displayed to the user

SortOrder

`integer`

Ordinal indicating this Option's position in the list of possible responses

Supported Operations

No operations directly supported.

SurveyResponse

Fields

InteractionId

`nonNegativeInteger`

References the system-generated value identifying an Interaction type to be logged for Survey Responses

SurveyId

`nonNegativeInteger`

Multiple values allowed

References the system-generated value identifying the Survey to which the Survey Response applies

ConsId

`nonNegativeInteger`

Multiple values allowed

References the system-generated value identifying the Constituent who submitted this Survey Response

SubmitDate

`dateTime`

Multiple values allowed

The timestamp on the Survey Response

Answer

SurveyAnswer

Multiple values allowed

The response submitted

Supported Operations

Find

Query

TeamRaiserClassification

Fields

ClassificationId

`nonNegativeInteger`

System-generated unique identifier for the record

Name

`string`

Name of the classification

Type

`nonNegativeInteger`

Type of classification

Supported Operations

Find

Query

TeamRaiserCompany

Fields

CompanyId

`nonNegativeInteger`

System-generated unique identifier for the Company record

EventId

`nonNegativeInteger`

References the unique ID of the TeamRaiser Event associated with the company

Name

`string`

Name of the company

DisplayName

`string`

Display name of the company

IsOnlySponsor

`boolean`

Whether or not the company is the sole sponsor of this TeamRaiser event

IsNational

`boolean`

True for a National company participation

IsRegional

`boolean`

True for a Regional company participation

CoordinatorConsId

`integer`

Constiutent ID of the company coordinator

PrimaryContactConsId

`integer`

Constiutent ID of the primary company contact

Goal

`decimal`

Corporate fundraising goal amount

SponsorshipAskDate

`date`

Date on which sponsorship was requested

SponsorshipProjectedCloseDate

`date`

Date on which sponsorship is expected to close

SponsorAmountCommitted

`decimal`

SponsorAmountPaid

`decimal`

SponsorAskAmount

`decimal`

Requested sponsorship amount

SponsorContactEmail

`string`

Sponsor contact email

SponsorContactName

`string`

Sponsor contact name

SponsorContactPhone

`string`

Sponsor contact phone number

SponsorGiftType

`SponsorGiftType`

Sponsor gift type

SponsorNotes

`string`

Notes

SponsorPaymentType

`SponsorPaymentType`

Payment type

SponsorStaffName

`string`

Staff name

SponsorStatus

`SponsorStatus`

Status

SponsorVolunteerManagerName

`string`

Name of the company volunteer manager

SponsorshipType

`SponsorshipType`

Company sponsorship type

FiscalYear

`nonNegativeInteger`

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

TeamRaiserEvent

Fields

EventId

`nonNegativeInteger`

A system-generated unique identifier for the Event

CampaignId

`nonNegativeInteger`

References the unique ID of the Campaign associated with this TeamRaiser event

Title

`string`

A descriptive name assigned to the TeamRaiser Event by the organizer

Name

`string`

Name of the event

Type

`token`

Status

`TeamRaiserEventStatus`

EventStatus

FormId

`integer`

References the system-generated value identifying a Donation Form associated with this TeamRaiser Event

PreviousEventId

`integer`

Links this event with a previous event to provide returning participants with access to their donation history and to make comparison reports available.

FiscalYear

`nonNegativeInteger`

Fiscal year of the event.

IsBlueprint

`boolean`

True if this event is a blueprint for new events.

ClassificationId1

`nonNegativeInteger`

One of a sequence of up to three unique IDs of associated TeamRaiserEventClassification records

ClassificationId2

`nonNegativeInteger`

One of a sequence of up to three unique IDs of associated TeamRaiserEventClassification records

ClassificationId3

`nonNegativeInteger`

One of a sequence of up to three unique IDs of associated TeamRaiserEventClassification records

SurveyId

`nonNegativeInteger`

References the system-generated value identifying a Survey associated with this TeamRaiser Event

TaxId

`string`

Federal tax ID for the event

EventDate

`dateTime`

Date of the event.

PublishDate

`dateTime`

Publication date of the event

UnpublishDate

`dateTime`

Event expiration date

AdminConsId

`integer`

ChairpersonConsId

`integer`

References the Constituent ID of the event chairperson

ChairpersonCompanyId

`integer`

References the unique ID of the event chairperson's company

Goal

`decimal`

Funraising goal of the event.

SuggestedParticipantGoal

`decimal`

Suggested fundraising goal for each event participant

SuggestedTeamGoal

`decimal`

Suggested fundraising goal for each team

EventAddress

`Address`

Address of the event.

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

LocationName

`string`

Event location name or venue

LocalOfficeAttn

`string`

Attention line for local office mailing address

LocalOfficeAddress

`Address`

Local office mailing address

Supported Operations

Find

Query

TeamRaiserGift

Fields

GiftId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

EventId

`nonNegativeInteger`

The system-generated unique identifier for the predefined TeamRaiser Event associated with the Gift

TeamId

`nonNegativeInteger`

Unique ID of the TeamRaiserTeam record associated with the gift

ParticipantConsId

`nonNegativeInteger`

Constituent ID of the TeamRaiser Participant associated with the gift

Type

`TeamRaiserGiftType`

ProxyType

`ProxyType`

Proxy type

TransactionId

`nonNegativeInteger`

Unique ID of the ValuedTransaction record associated with the gift

TotalAmount

`decimal`

Amount of gift including any additional amount

GiftAmount

`decimal`

Base gift amount

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Payment

`Payment`

Payment information for the TeamRaiser Gift

Donor

`Constituent`

The record for the Constituent credited with the TeamRaiser Gift

InteractionSource

`InteractionSource`

Interaction source information for the transaction

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

TeamRaiserParticipationType

Fields

ParticipationId

`nonNegativeInteger`

A system-generated unique identifier for the record

EventId

`nonNegativeInteger`

A system-generated unique identifier for the associated Event

Name

`string`

Name of the Participation Type

PromotionCode

`string`

Promotion code

Fee

`decimal`

Participation fee

MinimumGoal

`decimal`

Minimum fundraising goal for this Participation Type

TaxDeductibleValue

`decimal`

Tax deductible value

WaiverRequired

`boolean`

True if a waiver of liability is required for this Participation Type

IsActive

`boolean`

True if this Participation Type is active

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Supported Operations

Find

Query

TeamRaiserRegistration

Fields

RegistrationId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

EventId

`nonNegativeInteger`

References the system-generated value identifying the TeamRaiser Event for this Registration

TeamId

`integer`

A system-generated unique identifier for the registered Team

ParticipationId

`nonNegativeInteger`

A system-generated unique identifier for the associated Participation Type record

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

AdditionalGift

`decimal`

Monetary amount that a participant can enter in the Additional Gift field during registration if this option is enabled

DiscountType

`string`

Label that identifies what type of discount the participant applied to their registration fee

Discount

`decimal`

Monetary amount that was subtracted from registration fee because the participant applied a discount code

TotalFeesPaid

`decimal`

Total fees collected through all registration transactions

UpsellTotal

`decimal`

Total monetary value paid by participants for additional option purchases they made while registering for this TeamRaiser

Type

`RegistrationType`

Online if participant completes and submits a registration form online Offline if an administrator registers the participant or the participant submits a printed paper form, other values reserved

ParentRegistrationId

`nonNegativeInteger`

If this is a secondary Registration, the unique identifier of the primary Registration

IsAnonymous

`boolean`

Indicates whether the participation is anonymous (does not appear in public lists or searches).

ScreenName

`string`

The participant's screen name.

Payment

Payment

Payment information for the TeamRaiser Registration

Participant

Constituent

The constituent record for the registered participant

InteractionSource

InteractionSource

Interaction source information for the transaction

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

TeamRaiserTeam

Fields

TeamId

nonNegativeInteger

A system-generated unique identifier for the Team

EventId

nonNegativeInteger

A system-generated unique identifier for the associated Event

CompanyId

nonNegativeInteger

Unique ID of the TeamRasierCompany record associated with this team

CaptainConsId

nonNegativeInteger

Constituent ID of the team captain

Name

string

Name of the team

Description

string

Description of the team

Division

string

Name of the division that identifies a group of teams competing against each other in this TeamRaiser Event

Type

`token`

Classic, an ordinary team for a TeamRaiser Event (null is equivalent); Memorial, a memorial fund for Tributes; Tribute, a tribute fund for Tributes; Personal, a personal fund for Tributes

Goal

`decimal`

Fundraising goal for the team as a whole

MemberGoal

`nonNegativeInteger`

Fundraising goal for each team member

IsHonoraryTeam

`boolean`

True if this is an honorary team that participants are assigned to

IsPrivate

`boolean`

True if the Team Captain has set a password that people must know in order to join the team

CreationDate

`dateTime`

Timestamp indicating when the record was created

CreatedBy

`nonNegativeInteger`

Constituent who created the record

ModifyDate

`dateTime`

Timestamp indicating when the record was last changed

LastChangeBy

`nonNegativeInteger`

Constituent who last modified the record

Supported Operations

Find

Query

TicketPurchase

Fields

TransactionId

`nonNegativeInteger`

A system-generated unique identifier for the transaction

EventId

`nonNegativeInteger`

References the system-generated value identifying the Event for this transaction

SecurityCategoryId

`nonNegativeInteger`

References a system-generated value identifying the Security Category, controlling which constituents can see the record

Payment

`Payment`

Payment information for this transaction

Purchaser

`Constituent`

The constituent record of the purchaser

InteractionSource

`InteractionSource`

Interaction source information for the transaction

Supported Operations

GetIncrementalInserts

GetIncrementalInsertsCount

GetIncrementalUpdates

GetIncrementalUpdatesCount

GetIncrementalDeletes

GetIncrementalDeletesCount

TransactionSummary

Fields

Date

`dateTime`

Date of the transaction summarized in this record

Amount

`decimal`

Monetary amount of the transaction summarized in this record

TenderType

`TenderType`

Tender type for the transaction summarized in this record

PayMethod

`PaymentMethod`

Payment method of the transaction summarized in this record

CampaignId

`nonNegativeInteger`

References the system-generated value identifying the Campaign which generated the transaction summarized in this record

DonationType

`string`

One of: 1 = Donation, 2 = Pledge, 3 = Recurring Gift, 4 = Ticket, 5 = Pending, 6 = Pending Accepted, 7 = Pending Rejected, 8 = Pending Accepted Failed, 9 = Pending Rejected Failed, 10 = Donation Accepted Failed, 11 = Donation Rejected Failed, 12 = Settled, 13 = Rejected, 20 = TeamRaiser Registration, 21 = Tribute Registration, 25 = eCommerce, 30 = Refund

Supported Operations

No operations directly supported.

Enumerations

ActiveDetail

Fields

UNDEFINED

UNKNOWN_TO_CONTACT

KNOWN_TO_CONTACT

ACCEPTED_BY_CONTACT

REGISTERED_CONTACT

NEW_PROSPECT

INTERESTED_PROSPECT

REMOVED_BY_ADMINISTRATOR

REMOVED_BY_CONTACT

REMOVED_AS_DUPLICATE

ActiveStatus

Fields

AWAITING_APPROVAL

ACTIVE

REMOVED

APPROVAL_DENIED

OPT_IN_PENDING

PROSPECT

AddressStatus

Fields

UNKNOWN

CORRECTED

AddressType

Fields

HOME

WORK

OTHER

AdministratorStatus

Indicates whether or not constituent is an administrator

Controlled in the Convio admin interface

Fields

NOT_AN_ADMINISTRATOR

Constituent has not been made eligible to perform administrator role

ADMINISTRATOR

Constituent has been made eligible to perform administrator role

AffiliateAssignment

Fields

AUTOMATIC

MANUAL

AlertStatusType

Fields

INVALID

ACTIVE

INACTIVE

DELETED

DATE_DRIVEN

ARCHIVED

CreationMethod

Fields

OFFLINE

ONLINE

SUSTAINING

CreditCardType

Fields

NONE

VISA

MASTERCARD

AMERICAN_EXPRESS

DISCOVER

DINERS_CLUB

SWITCH

SOLO

ACH

PAYPAL

AMAZON

GOOGLE

BILLMELATER

DonationLevelType

Fields

FUNDRAISER

RECURRING

BOTH

DonationType

Fields

OFFLINE

SETTLED_DONATION

PLEDGE_PAYMENT

RECURRING_GIFT

TICKETED_EVENT

DONATION_PENDING_REVIEW

DONATION_ACCEPT_SETTLE_PENDING

DONATION_REJECTED_SETTLE_PENDING

DONATION_ACCEPTED_SETTLE_FAILED

DONATION_REJECTED_SETTLE_FAILED

DONATION_ACCEPTED_SETTLE_REFUSED

DONATION_REJECTED_SETTLE_REFUSED

DONATION_ACCEPTED_SETTLED

DONATION_REJECTED_SETTLED

TEAM_RAISER_REGISTRATION

TRIBUTES_REGISTRATION

E_COMMERCE_PURCHASE

REFUND

FAILED_RECURRING_DONATION_DECLINED

FAILED_RECURRING_DONATION_FRAUD

DonorStatus

Fields
UNDEFINED

NON_DONOR

INDIRECT_DONOR

JOINT_DONOR

DONOR

UNKNOWN

EmailFormat

Fields
UNKNOWN

HTML

PLAIN_TEXT

EmailStatus

Fields
UNKNOWN

GOOD

BAD_SOFT_BOUNCE

BAD_HARD_BOUNCE

InteractionType

Fields
DEFAULT

DONATION

PLEDGE

PAYMENT

RSVP

INVITATION

TICKETPURCHASE

REDIRECT

MAIL_SENT

MAIL_BOUNCE

MAIL_MERGE

PHONE_CALL

PERSONAL_VISIT

SURVEY_SUBMITTED

SURVEY_INVITATION

EMAIL_DECLINED

PHONE_LIST

CAMPAIGN_VISIT

CAMPAIGN_APPEAL

LOGIN_SUCCESS

LOGIN_FAILURE

PLEDGE_DELETED

ACTION_ALERT_SENT

ACTION_ALERT_INVITATION

PAYMENT_ACCEPTED

PAYMENT_REJECTED

PAYMENT_IGNORED

REGISTRATION

STORIES

F2F_REGISTRATION_ID

TELL_A_FRIEND

PAGE_VIEW

REWARD_REDEEMED

RECURRING_PLEDGE_PROCESSED

RECURRING_PLEDGE_CANCELLED

ECOMMERCE_PURCHASE

CONS_RECORD_CHANGE

LOGIN_SUCCESS_ROLLUP

LOGIN_FAILURE_ROLLUP

MAIL_SENT_ROLLUP

DIRECTORY_SEARCH

CAREER_CENTER_SEARCH

DIRECTORY_OPTIN

TRANSACTION_ADJUST

BLIND_EMAIL_SENT

BLIND_EMAIL_RECIEVED

RECEIPT_REQUEST

SURVEY_MODIFIED

AUCTION_DONATED

AUCTION_BID

AUCTION_WON

AUCTION_BOUGHT

TRIBUTES_REGISTRATION_ID

RECURRING_PLEDGE_SKIP_PAYMENT

RECURRING_PLEDGE_CHANGE_BILLING_INFO

ECOMMERCE_REFUND

F2F_REGISTRATION_REFUND_ID

ECOMMERCE_ECARD_SENT

TRANSACTION_REFUND

MEMBERSHIP_ADJUSTMENT

ECOMMERCE_BILL_ME_LATER

GETTOGETHER_HOST_CREATE_EVENT

GETTOGETHER_HOST_CHANGE_EVENT

GETTOGETHER_ATTENDEE_REGISTRATION

GETTOGETHER_ATTENDEE_CHANGE_REGISTRATION

F2F_DELETE_REGISTRATION_ID

RECEIPT_FULL_REFUND

RECEIPT_PARTIAL_REFUND

RECEIPT_REPLACEMENT

ECOMMERCE_DOWNLOAD_PICKED_UP

CAPTAIN_PROMOTE

VOLUNTEER_REGISTRATION

VOLUNTEER_REGISTRATION_DELETE

VOLUNTEER_PROFILE_UPDATE

F2F_REG_FEES_PAID_OFFLINE

InterestEventType

Fields

OPT_IN

OPT_OUT

TOTALS

NewDonorStatus

Fields

NEW_CONSTITUENT

NEW_DONOR

EXISTING_DONOR

OptOutMethod

Fields

UNKNOWN

UNSUBSCRIBE_LINK

CONSTITUENT_UPDATE

ADMINISTRATOR_UPDATE

DATA_SYNC

Origin

Fields

SELF_REGISTRATION

ADMINISTRATOR_REGISTRATION

PROSPECT_LIST

DATA_SYNC

QUICK_REGISTRATION

DONATION

JOINT_DONATION

TRIBUTE

MEMORIAL

GIFT_NOTIFICATION

TICKET_PURCHASE

EVENT_GUEST

ADVOCACY_RESPONSE

TELL_A_FRIEND

TEAM_RAISER_DONOR

UNKNOWN_REGISTRATION

UNKNOWN_AUTO_REGISTRATION

NEWS_CONTRIBUTION

TEAM_RAISER_PARTICIPANT

TEAM_RAISER_FRIEND

FORWARDED_EMAIL

SURVEY_PARTICIPANT

PRE_REGISTRATION

E_COMMERCE_PURCHASE

LDAP

RSVP

EMAIL_UPLOAD

SENT_E_CARD

TEAM_RAISER_COMPANY_CONTACT

TRIBUTES_CHAMPION

TRIBUTES_FUND

AUCTION_UPLOAD

WIDGETS

TEAM_RAISER_FAMILY_PARTICIPANT

PERSONAL_EVENTS_HOST

PERSONAL_EVENTS_ATTENDEE

PERSONAL_EVENTS_CONTACT

IMPORT

OriginationType

Fields

ONLINE

API_INTERNAL

API_EXTERNAL

CRM

PasswordType

Fields

UNDEFINED

CONVIO

LDAP

PaymentMethod

Fields
OFFLINE

ONLINE

PhoneType

Fields
HOME

WORK

MOBILE

ProxyType

Fields
NONE

OFFLINE

TEAMRAISER_PARTICIPANT

TEAMRAISER_EVENT

TEAMRAISER_TEAM

PERSONAL_FUNDRAISING_CHAMPION

PERSONAL_FUNDRAISING_TEAM

RecognitionType

Fields
NONE

MEMORIAL

TRIBUTE

RecurringPaymentType

Fields
SUSTAINING

INSTALLMENT

RegistrationType

Fields
ONLINE

OFFLINE

REG_TYPE_2

REG_TYPE_3

REG_TYPE_4

REG_TYPE_5

REG_TYPE_6

REG_TYPE_7

REG_TYPE_8

REG_TYPE_9

SettlementStatus

Fields

UNSETTLED

SUBMITTED

SETTLED

SourceCodeType

Fields

OTHER

WEB_LINK

TELL_A_FRIEND

EMAIL

FORWARDED_EMAIL

ECARD

REFERER

PERSONAL_OUTREACH

SponsorGiftType

Fields

NONE

CASH

VOLUNTEERS

OTHER

SponsorPaymentType

Fields

NONE

CASH

CHECK

CREDIT

IN_KIND

OTHER

SponsorStatus

Fields
NONE

ASK

COMMITTED

CONTRACT

PAID

PROSPECT

DECLINED

SponsorshipType

Fields
NONE

NATIONAL

REGIONAL

LOCAL

OTHER

SurveyType

Fields
STANDARD

COMPONENT

TeamRaiserEventStatus

Fields
UNPUBLISHED

ACCEPT_REGISTRATION

ACCEPT_ALL

ACCEPT_GIFT

CLOSED

ARCHIVED

DELETED

TeamRaiserGiftType

Fields

ONLINE

OFFLINE_CONFIRMED

OFFLINE_UNCONFIRMED

OFFLINE_REJECTED

TenderType

Fields

UNKNOWN

CREDIT_CARD

CHECK

CASH

ACH

ZeroOrOneBoolean

Fields

false

true

RecordResultCode

Fields

OK

UNEXPECTED

INVALID_API_KEY

TOO_MANY_RECORDS

REQ_FIELDS_MISSING

MATCH_EXISTING

DUPLICATE_USER_NAME

DUPLICATE_EMAIL

CREATE_NOT_ALLOWED_FOR_CALLER

UPDATE_MATCH_AMBIGUOUS

RECORD_NOT_FOUND

INVALID_FIELD

COLLISION

READ_ONLY_FIELD

ATTRIBUTE_FAILED_VALIDATION

INVALID_RECORD_TYPE

DELETE_NOT_ALLOWED

Faults

InvalidMessageFault

Fault Parameters

Parent

`string`

Element

`string`

InvalidParameterFault

Fault Parameters

Parent

`string`

Element

`string`

Value

`string`

InvalidQueryFault

Fault Parameters

Query

`string`

Token

`normalizedString`

Position

`integer`

LoginFault

Indicates a failure to authenticate

Returned by the server if the user name is invalid, the password is invalid or the user does not have the required permissions to access web services.

Fault Parameters

UserName

`normalizedString`

The submitted user name.

MalformedMessageFault

Fault Parameters

Line

`int`

Column

`int`

RequestFault

Fault Parameters

ResultCode

`int`

Message

`string`

Date

`dateTime`

RequestRateLimitFault

Indicates that the hourly or daily request rate limit has been exceeded

There are two request rate limits set for Convio Web Services requests: by default, each site is permitted 10,000 requests per hour and 100,000 requests per day. Exceeding these limits will result in a RequestRateLimit fault. These rate limits can be adjusted by Convio per site, as needed. These rates are adequate to query 2 million records per hour at 200 records per response.

Fault Parameters

No parameters

ServerFault

Fault Parameters

Uuid

`normalizedString`

ServiceUnavailableFault

Fault Parameters

No parameters

SessionFault

Fault Parameters

SessionId

`normalizedString`

SynchronizationFault

Fault Parameters

PartitionName

`string`

PartitionId

`nonNegativeInteger`

UnsupportedOperationFault

Fault Parameters

Operation

`normalizedString`

Reason

`normalizedString`