

EngagemoreCRM Software Technical Report*

John Wooten Ph.D.

March 11, 2020

Version: 1.3

Abstract

The technical description of a database and web application for tracking and managing EngagemoreCRM customers and their payments. The Thrivecart application is used to allow customers to sign up for various subscription options. Their signups result in a webhook notification being sent to the thrivecart.php webhook. This processes their action and creates or modifies the customers EngagemoreCRM account.

*Software developed under contract with EngagemoreCRM

Contents

| | | |
|---|---------------------------|----|
| 1 | Track Changes | 4 |
| 2 | Introduction | 5 |
| 3 | Approach | 5 |
| 4 | Database Tables | 6 |
| 5 | Flow Chart | 7 |
| | Appendices | 8 |
| A | config.ini.php | 8 |
| B | mysql_common.php | 9 |
| C | thrivecart_api.php | 17 |
| D | thrivecart.php | 18 |
| E | utilities.php | 23 |
| F | add_account.php | 25 |
| G | change_account_status.php | 27 |
| H | post_api_url.php | 28 |
| I | upgrade_account.php | 29 |
| J | git-info.php | 31 |

List of Figures

| | | |
|---|----------------------|---|
| 1 | Flow Chart | 7 |
|---|----------------------|---|

List of Tables

| | | |
|---|----------------------------|---|
| 1 | Table of Changes | 4 |
| 2 | users table. | 6 |
| 3 | logs table. | 7 |

1 Track Changes

| <i>Date</i> | <i>Editor</i> | <i>Comment</i> | <i>Version</i> |
|-------------|---------------|------------------------------|----------------|
| 191125 | J. Wooten | Initial Draft | 1.0 |
| 200109 | J. Wooten | Expand on Operation | 1.1 |
| 200214 | J. Wooten | Update Table Definitions | 1.2 |
| 200311 | J. Wooten | Update user table definition | 1.3 |

Table 1: Table of Changes

2 Introduction

Customers who desire to utilize the **EngagemoreCRM**¹ services, are directed from a webpage where the description of the EngagemoreCRM site is found along with a comparison of various offerings to a **Thrivecart**² site where they enter their subscription information including their *email* and their credit card information, which Thrivecart manages. Upon completion of their signup, a webhook notification is sent to the webhook **thrivecart.php** on a server used by EngagemoreCRM. That webhook, updates the customer information stored in a mysql database called **user_db/users**, which keeps the customers *email* and after adding the user to EngagemoreCRM, their *engagemoreId* and the status, either *active* or *in-active*. A web-app is available to examine the users database table, and also the logs database table. The logs database table **user_db/logs** contains the logs of each transaction received from Thrivecart.

3 Approach

In order to process thrivecart events that are related to EngagemoreCRM users, we must connect the *email* which is sent by Thrivecart when a user is added to the EngagemoreCRM id that occurs when a new user is created within EngagemoreCRM. This allows us to manage future Thrivecart notifications about cancellation of their subscription and possible upgrades in their subscription, as well as use various API's and Zapier³ notifications to add contacts to a users account.

¹<https://secure.engagemorecrm.com/Login.aspx>

²<https://thrivecart.com/signin/>

³<https://platform.zapier.com/quickstart/introduction>

4 Database Tables

There are two database tables, users, and logs that are used by the webhook. Their structures are:

| Name | Type | Null | Default | Extra |
|--------------|--------------|------|---------|----------------|
| id | int(11) | No | None | Auto Increment |
| added | datetime | No | None | |
| email | varchar(128) | No | None | |
| engagemoreid | varchar(64) | No | None | |
| orderid | int(11) | No | None | |
| invoiceid | int(11) | No | None | |
| status | varchar(256) | No | None | |
| product | varchar(255) | No | None | |

Table 2: users table.

and:

| Name | Type | Null | Default | Extra |
|--------------|----------------|------|---------|---------------|
| id | int(11) | No | None | AutoIncrement |
| received | datetime | No | None | |
| email | varchar(128) | No | None | |
| request_json | varchar(16000) | No | None | |
| status | varchar(128) | No | None | |
| commit_hash | varchar(64) | Yes | NULL | None |
| branch | varchar(64) | Yes | NULL | None |

Table 3: logs table.

5 Flow Chart

The following chart shows the events and the actions that happen in response to them.

To Be Provided

Figure 1: Flow Chart

Appendices

In the directory patti/webhook on the web server there several files that comprise the thrivecart webhook.

A config.ini.php

The config.ini.php page provides the externalized constants so that if the server address changes, it can be changed in one place. Passwords are maintained here also, and thus this file is **NOT** checked into the github⁴ repository but are passed in by copying from the Trello⁵ site where they are specified.

```
<?php
$config['RAILS_ENV']='development';
$config['RAILS_USER']='woo';
$config['RAILS_USERID']='jwooten37830@mac.com';
$config['RAILS_PASSWORD']='random1';
$config['MAIL_HOST']='localhost:3000';
$config['EMAIL_FROM']=$config['RAILS_USERID'];
$config['SENDMAIL_SERVER']='smtp.gmail.com';
$config['SENDMAIL_PORT']='587';
$config['SENDMAIL_USERNAME']='wooten.666@gmail.com';
$config['SENDMAIL_PASSWORD']='passwd';
$config['ISSUES_SERVER']='https://github.com/woo37830/blog/issues';
$config['PATTI_ISSUES_SERVER']='https://github.com/woo37830/patti_rails/issues';
$config['PATTI_DOCUMENTS']='https://jwooten37830.com/patti/webhook/documentation/';
// $config['PATTI_DATABASE_SERVER']='jwooten37830.com';
$config['PATTI_DATABASE_SERVER']='44.231.61.194';
$config['PATTI_DATABASE_USER']='woo';
$config['PATTI_DATABASE_PASSWORD']='enaqbz1';
$config['PATTI_DATABASE']='users_db';
$config['PATTI_LOG_TABLE']='logs';
$config['PATTI_USERS_TABLE']='users';
$config['NOTES_SERVER']='https://jwooten37830.com/wiki/projects/blogdiscussions/';
$config['MSG_USER']='4K9vV0InIxP5znCa7d';
```

⁴<https://github.com>

⁵<http://thrivecart.com>


```
$config['MSG_PASSWORD']='ie6n85dF826iYe5npA';
$config['THRIVECART_SECRET']='IEYDASLZ8FR7';
?>
```

B mysql_common.php

The file, mysql.comon.php, contains common mysql functions that all of the other files often share. Such things as connect, logging to the logs database(see Table 3) getStatus, etc. are here.

```
<?php // mysql_common.php

// COMMON CODE AVAILABLE TO ALL OUR webhook scripts

// SET ERROR REPORTING SO WE CAN DEBUG OUR SCRIPTS EASILY
error_reporting(E_ALL);
date_default_timezone_set('America/New_York');

function connect($db) {
    require 'config.ini.php';
    $db_host = $config['PATTI_DATABASE_SERVER']; // PROBABLY THIS IS OK
    $db_name = $db; // GET THESE FROM YOUR HOSTING COMPANY
    $db_user = $config['PATTI_DATABASE_USER'];
    $db_word = $config['PATTI_DATABASE_PASSWORD'];

    // OPEN A CONNECTION TO THE DATA BASE SERVER AND SELECT THE DB
    $mysqli = new mysqli($db_host, $db_user, $db_word, $db_name);

    // DID THE CONNECT/SELECT WORK OR FAIL?
    if ($mysqli->connect_errno)
    {
        // die("mysql connect error: $mysqli->connect_error");
        $err
        = "CONNECT FAIL: "
        . $mysqli->connect_errno
        . ' '
    }
}
```

```

        . $mysqli->connect_error
        ;
        trigger_error($err, E_USER_ERROR);
    }
    return $mysqli;
}

function logit($user, $json, $my_status)
{
    require 'config.ini.php';

    $dbase = $config['PATTI_DATABASE'];
    if( $conn = connect($dbase) )
    {
        $rev = exec('git rev-parse --short HEAD');
        $branch = exec('git rev-parse --abbrev-ref HEAD');

        $datetime = date_create()->format('Y-m-d H:i:s');
        $table = $config['PATTI_LOG_TABLE'];
        $sql = "INSERT INTO $table
        ( received
        , email
        , request_json
        , status
        , commit_hash
        , branch

        ) VALUES
        ( '$datetime'
        , '$user'
        , '$json'
        , '$my_status'
        , '$rev'
        , '$branch'
        )";

        if (!$res = $conn->query($sql))
        {

```

```

        $err
        = "QUERY FAIL: "
        . $sql
        . ' ERRNO: '
        . $conn->errno
        . ' ERROR: '
        . $conn->error
        ;
        mysqli_close($conn);
        trigger_error($err, E_USER_ERROR);
    }
    else
    {
        mysqli_close($conn);
        return;
    }
}

}

function addUser($user, $engagemoreid, $productid, $invoiceid, $orderid)
{
    require 'config.ini.php';

    $dbase = $config['PATTI_DATABASE'];
    if( $conn = connect($dbase) )
    {
        $datetime = date_create()->format('Y-m-d H:i:s');
        $table = $config['PATTI_USERS_TABLE'];
        $sql = "INSERT INTO $table
        ( added
        , email
        , engagemoreid
        , product
        , invoiceid
        , orderid
        ) VALUES
        ( '$datetime'
        , '$user'

```

```

        , '$engagemoreid'
        , '$productid'
        , '$invoiceid'
        , '$orderid'
    )";

    if (!$res = $conn->query($sql))
    {
        $err
        = "QUERY FAIL: "
        . $sql
        . ' ERRNO: '
        . $conn->errno
        . ' ERROR: '
        . $conn->error
        ;
        mysqli_close($conn);
        trigger_error($err, E_USER_ERROR);
    }
    else
    {
        mysqli_close($conn);
        return;
    }
}

function updateAccountStatus($accountid, $new_status)
{ // Set the status in the users table to show it is inactive for the $accountid.
    require 'config.ini.php';

    $table = $config['PATTI_USERS_TABLE'];

    $sql = " UPDATE $table SET status='" . $new_status . "' WHERE engagemoreid = " .
    $status = 'Failed';
    $dbase = $config['PATTI_DATABASE'];

    if( $conn = connect($dbase) )

```

```

{
    if ($conn->query($sql))
    {
        $status = 'Succeeded';
    }
    else
    {
        logit($accountid,"","Error updating record: " . $conn->mysqli_error);
    }
    mysqli_close($conn);
}
return $status;
}

function getStatusFor( $accountid ) {
    require 'config.ini.php';

    $dbase = $config['PATTI_DATABASE'];

    if( $conn = connect($dbase) )
    {
        $datetime = date_create()->format('Y-m-d H:i:s');
        $table = $config['PATTI_USERS_TABLE'];

        $query = "SELECT status FROM $table WHERE engagemoreid = " . $accountid;
        $results_array = array();
        $result = $conn->query($query);
        while( $row = $result->fetch_assoc() ) {
            $results_array[] = $row;
        }
        if( !empty( $results_array[0] ) )
        {
            $value = $results_array[0]['status'];
        }
        $result -> close();
        $conn->close();
        return $value;
    }
}

```

```

        return 'inactive';
    }

function getAccountId($email)
{ // Get the Engagemore(AllClients) engagemoreid from the users database
  // given the email or the thrivecart id
  require 'config.ini.php';

  $value = -1;
  $dbase = $config['PATTI_DATABASE'];

  if( $conn = connect($dbase) )
  {
    $datetime = date_create()->format('Y-m-d H:i:s');
    $table = $config['PATTI_USERS_TABLE'];

    $query = "SELECT engagemoreid FROM $table WHERE email = '$email' ";
    $results_array = array();
    $result = $conn->query($query);
    while( $row = $result->fetch_assoc() ) {
        $results_array[] = $row;
    }
    if( !empty( $results_array[0] ) )
    {
        $value = (int)$results_array[0]['engagemoreid'];
    }
    $result -> close();
    $conn->close();
  }
  return $value;
}

function getProductFor( $email ) {
  $value = -1;
  require 'config.ini.php';

  $dbase = $config['PATTI_DATABASE'];

  if( $conn = connect($dbase) )

```

```

{
    $datetime = date_create()->format('Y-m-d H:i:s');
    $table = $config['PATTI_USERS_TABLE'];

    $query = "SELECT product FROM $table WHERE email = '$email' ";

    $results_array = array();
    $result = $conn->query($query);
    while( $row = $result->fetch_assoc() ) {
        $results_array[] = $row;
    }
    if( !empty( $results_array[0] ) )
    {
        $value = $results_array[0]['product'];
    }
    $result -> close();
    $conn->close();
}
return $value;
}

function updateProduct($accountid, $new_product)
{ // Set the status in the users table to show it is inactive for the $accountid.
    require 'config.ini.php';

    $table = $config['PATTI_USERS_TABLE'];

    $sql = " UPDATE $table SET product = '" . $new_product . "' WHERE engagemoreid = $status = 'Failed';
    $dbase = $config['PATTI_DATABASE'];

    if( $conn = connect($dbase) )
    {
        if ($conn->query($sql))
        {
            $status = 'Succeeded';
        }
        else

```

```

        {
            $status = "FAILED: " . mysqli_error($conn);
        }
        mysqli_close($conn);
    }
    return $status;
}

function addOrderAndInvoiceIds($accountid, $orderid, $invoiceid)
{ // Set the status in the users table to show it is inactive for the $accountid.
    require 'config.ini.php';

    $table = $config['PATTI_USERS_TABLE'];

    $sql = " UPDATE $table SET orderid = '" . $orderid . "', invoiceid = '" . $invoiceid . "'";
    $status = false;
    $dbase = $config['PATTI_DATABASE'];

    if( $conn = connect($dbase) )
    {
        if ($conn->query($sql))
        {
            $status = true;
        }
        else
        {
            echo "FAILED: " . mysqli_error($conn) . "\n";
        }
        mysqli_close($conn);
    }
    return $status;
}

function getEmailForInvoiceId( $invoiceid ) {
    require 'config.ini.php';

    $dbase = $config['PATTI_DATABASE'];
    $value = 'Unknown';

```



```

if( $conn = connect($dbase) )
{
    $datetime = date_create()->format('Y-m-d H:i:s');
    $table = $config['PATTI_USERS_TABLE'];

    $query = "SELECT email FROM $table WHERE invoiceid = " . $invoiceid;
    $results_array = array();
    $result = $conn->query($query);
    while( $row = $result->fetch_assoc() ) {
        $results_array[] = $row;
    }
    if( !empty( $results_array[0] ) )
    {
        $value = $results_array[0]['email'];
    }
    $result -> close();
    $conn->close();
    return $value;
}
return $value;
}
?>

```

C thrivecart_api.php

This page contains the logic to communicate the data received from the Thrivecart notification system to the EngagemoreCRM system. The data and the specific EngagemoreCRM function will cause such actions as adding a user, changing their status from active to inactive, and changing the group in EngagemoreCRM to which they belong.

```

<?php // thrivecart_api.php
// Expects a $data array containing data required for the particular AllClients ap
// Also, the $url for the call
// It will return a $results_xml structure if there is not an error.
//

```

```

require 'post_api_url.php';

function thrivecart_api($url, $data) {

/**
 * Insert the account and get the response as XML string:
 *
 * <?xml version="1.0"?>
 * <results>
 *   <message>Success</message>
 *   <contactid>15631</contactid>
 * </results>
 *
 * @var string $contacts_xml_string
 */
$result_xml_string = post_api_url($url, $data);

/**
 * SimpleXML will create an object representation of the XML API response. If
 * the XML is invalid, simplexml_load_string will return false.
 *
 * @var SimpleXMLElement $results_xml
 */
$results_xml = simplexml_load_string($result_xml_string);

return $results_xml;
}
?>

```

D thrivecart.php

This is the php page that is added to the Thrivecart site to receive the notifications when actions are taken in Thrivecart such as someone signing up, cancelling, etc. It contains a switch statement that uses the event provided by Thrivecart to select the proper EngagemoreCRM API to perform the desired EngagemoreCRM action. All notifications result in a logging event,

and if the event is one supported by the webhook, then the proper API is invoked and the results are logged. If a user is added, modified, or cancelled, the status is added or modified in the users(see Table 2) data table.

```
<?php
//
require 'config.ini.php';
require 'thrivecart_api.php';
require 'mysql_common.php';
require 'add_account.php';
require 'change_account_status.php';
require 'upgrade_account.php';
require 'utilities.php';
/**
 * AllClients Account ID and API Key.
 */
$account_id    = $config['MSG_USER'];
$api_key       = $config['MSG_PASSWORD'];
$api_endpoint  = 'https://secure.engagemorecrm.com/api/2/';

$events = array('order.success', 'order.subscription_payment', 'order.subscription_cancelled');
$affiliate_events = array('affiliate.commission_refund', 'affiliate.commission_earned');

$email_limits = array("product-9" => 5000, "product-12" => 5000, "product-13" => 10000,
                      "product-14" => 10000, "product-15" => 10000, "product-16" => 10000,
                      "product-17" => 10000);
echo "<html><head></head><body><h1>OK</h1></body></html>";
$json_data = json_encode($_REQUEST);

/**
 * The API endpoint and time zone.
 */
// Verify the webhook origin by checking for the Webhook Key value you defined in
/*if( empty( $_REQUEST['thrivecart_secret'] ) || $_REQUEST['thrivecart_secret'] != $secret ) {
    logit("INVALID", $json_data, "No key supplied");
    die('Invalid request, no key supplied');
}
}
```

```

*/
$email = 'Undefined';
if( isset( $_REQUEST['event'] ) ) {
    $event = $_REQUEST['event'];
    if( isset($_REQUEST['customer'] ) ) {
        $email = $_REQUEST['customer']['email'];
    }
}
// Message seems to be from ThriveCart so log it.
// Look for the order.success webhook event. Make sure the response is complete be
if( empty( $event ) ) {
    logit("INVALID", $json_data, "No event provided");
    die('No event provided');
}
switch( $event ) {
    case 'order.success':
        echo "Received order.success<br />" . $email . " - " . $json_data . "<br />";
        handleOrderSuccess($email, $api_endpoint, $account_id, $api_key, $json_data);
        break;
    case 'order.subscription_payment':
        logit($email, $json_data, "order.subscription_payment");
        echo "Received order.subscription_payment<br />" . $email . " - " . $json_data;
        break;
    case 'order.subscription_cancelled':
        $result = change_account_status($api_endpoint,$account_id, $api_key, $email,0);
        logit($email,$json_data, "Subscription_cancelled, result: $result");
        echo "Received order.subscription_cancelled. result = $result<br />" . $email;
        break;
    case 'order.refund':
        echo "Received order.refund<br />" . $email . " - " . $json_data . "<br />";
        logit($email, $json_data, "order.refund");
        break;
    case 'affiliate.commission_refund':
        echo "Received affiliate.commission_refund<br />" . $email . " - " . $json_data;
        logit($email, $json_data, "affiliate.commission_refund");
        break;
    case 'affiliate.commission_earned':
        echo "Received affiliate.commission_earned<br />" . $email . " - " . $json_data;

```

```

        logit($email, $json_data, "affiliate.commission_earned");
        break;
    case 'affiliate.commission_payout':
        echo "Received affiliate.commission_payout<br />" . $email . " - " . $json_data;
        logit($email, $json_data, "affiliate.commission_payout");
        break;
    default:
        echo "Invalid event - $event<br />" . $email . " - " . $json_data . "<br />";
        logit($email, $json_data, "Invalid event- $event");
        die();
}

//echo "Received event: $event with email: $email</br/>";
echo "<br /><hr />";
require 'git-info.php';
die('All Done');

```

```

function handleOrderSuccess($email, $api_endpoint, $account_id, $api_key, $json_data) {
    //echo "Check if account_exists for: $email <br />";
    //echo "json_data : " . $json_data . "<br />";
    $product = getProductId($_REQUEST);
    require 'product_data.php';
    if( array_key_exists($product, $products) ) { // Here is where we check that we

        $group_name = getProductName($product, $email, $json_data);

        if( account_exists($email) ) {
            echo "It does!<br />";
            if( account_isInactive($email) )
            {
                // reactivate account
                reactivate_account($email, $api_endpoint, $account_id, $api_key);
            }
            else
            {
                // account is active
                if( product_isTheSame($email, $product) )

```

```

{
    // It is a payment and just let it go.
    echo "Payment received for product: $product<br />";
    logit( $email, $json_data, "Payment was received for product: $product")
}
else
{
    // different product, then change the group for the account
    $engagemoreacct = (int)change_account_group($email, $api_endpoint, $account_id,
    $group_name, $product);
    if( $engagemoreacct != -1 ) {
        echo "Changed subscription to product: $product<br />";
        logit($email, $json_data, "SUCCESS: Changed product to $product");
        //adjust_email_limits($api_endpoint, $account_id, $api_key, $engagemoreacct)
    }
}
}
}
else { // account does not exist
/**
 * The contact information to insert.
 *
 * Information will be added to your AllClients contacts!
 */
$account = array(
    'password' => 'engage123',
);
$message = " with productid: $product";
$invoiceId = getInvoiceId();
$orderId = getOrderId();
$engagemoreacct = (int)add_account($api_endpoint, $account_id, $api_key, $product);
if( $engagemoreacct != -1 ) {
    if( $product == "product-15") { // One month free for Impact product
        $message = " - One month free/$99 mo. for product $product";
    }
    if( $product == "product-16") { // 2 months free and discounted rate
        $message = " - Special $990/yr. for $690/yr. product $product";
    }
}
}

```

```

        if( $product == "product-17") { // discounted rate
            $message = " - Special $99/mo. for $69/mo. product $product";
        }
        logit($email, $json_data, "SUCCESS: Added to account: $group_name, $mess

        //adjust_email_limits($api_endpoint, $account_id, $api_key, $engagemoreacco
    } // end not invalid engagemoreid, so it was created.
    } // end account does not exist - create it
} // end valid product
else {
    logit($email, $json_data, "NOT TRACKED: $product is not tracked by this webhoo
    echo "<br /><h3>NOT TRACKED: $product is not tracked by this webhook.</h3>";
    echo "<br />for " . $email . ", " . $json_data. "<br />";
}
}
}

```

?>

E utilities.php

A set of functions used in tests and in the main thrivecart.php application.

```

<?php

/**
 * A set of functions used in tests and in the main thrivecart.php application
 */

function account_exists($thrivecartid) {
    //return $value['account_exists'];
    $acct_id = getAccountId( $thrivecartid );
    return $acct_id != -1;
}

function account_isInactive($thrivecartid) {

```

```

    $id = getAccountId($thrivecartid);
    $saved_status = getStatusFor($id);
    // return $value['account_isInactive'];
    return $saved_status == 'inactive';
}

function reactivate_account($thrivecartid, $api_endpoint, $account_id, $api_key) {
    $accountid = (int)getAccountId( $thrivecartid );
    if( $accountid != -1 ) {
        change_account_status($api_endpoint,$account_id, $api_key, $thrivecartid,1);
    } else {
        logit($thrivecartid, "", "FAILURE in reactivate: Did not find email for $thrivecartid");
    }
}

function change_account_group($thrivecartid, $api_endpoint, $account_id, $api_key, $productid) {
    $accountid = (int)getAccountId( $thrivecartid );
    if( $accountid != -1 ) {
        upgrade_account($api_endpoint, $account_id, $api_key, $accountid,
            $group_name, $productid, $thrivecartid);
    } else {
        logit($thrivecartid, "", "FAILURE in change_account_group: Did not find email for $thrivecartid");
    }
    return $accountid;
}

function product_isTheSame($thrivecartid, $product) {
    $saved_product = getProductFor( $thrivecartid );
    return $product == $saved_product;
}

function getProductId() {
    $pmf = (int)$_REQUEST['base_product'];
    $product = "product-$pmf";
    return $product;
}

function getInvoiceId() {
    return $_REQUEST['invoice_id'];
}

```



```

}

function getOrderId() {
    return $_REQUEST['order_id'];
}

function getProductName($product, $email, $json_data) {

    require 'product_data.php';

    if( array_key_exists($product, $products) ) { // Here is where we check that we
        return $products[$product];
    }
    logit($email, $json_data, "Invalid product: $product");
    die("Invalid product: $product");
}
?>

```

F add_account.php

The file, add_account.php, handles the addition of the user who has just signed up through Thrivecart to the EngagemoreCRM system. Depending upon which product they selected, the proper EngagemoreCRM group is selected and the data is added to the users table and marked as 'active'.

```

<?php

function add_account($api_endpoint, $account_id, $api_key, $account,
    $group_name, $email, $product, $invoiceid, $orderid, $json_data) {
/**
 * Specify URL and form fields for AddContact API function.
 */

$url = $api_endpoint . 'AddAccount.aspx';

$data = array(
    'apiusername' => $account_id,
    'apipassword' => $api_key,

```

```

'email' => $email,
'password' => $account['password'],
'group' => $group_name,
'invoiceid' => $invoiceid,
'orderid' => $orderid,
);
$results_xml = thrivecart_api($url, $data); // returns simplexml_load_string object
if ($results_xml === false) {
logit($email, "", "FAILURE: Error parsing XML");
return -1;
}
/**
 * If an API error has occurred, the results object will contain a child 'error'
 * SimpleXMLElement parsed from the error response:
 *
 * <?xml version="1.0"?>
 * <results>
 *   <error>Authentication failed</error>
 * </results>
 */

if (isset($results_xml->error)) {
logit($email,$json_data, "FAILURE: $results_xml->error" );
echo "Failure: " . $results_xml->error . "<br />";
return -1;
}
/**
 * If no error was returned, the AddContact results object will contain a
 * 'contactid' child SimpleXMLElement, which can be cast to an integer.
 */
$account_id = (int)$results_xml->accountid;

// Here I write the account information using addUser in mysql_common.php
addUser($email, $account_id, $product, $invoiceid, $orderid);
return $account_id;
}
?>

```

G change_account_status.php

Depending upon certain activities, the account status might be changed from 'active' to 'inactive'. This file provides the update function for that and communicates to EngagemoreCRM by either suspending or activating an account.

```
<?php
/**
 * thrivecartid is the same as $email
 */
function change_account_status($api_endpoint, $account_id, $api_key, $thrivecartid) {
    $url = $api_endpoint . 'SetAccountStatus.aspx';
    // Get the account id from thrivecart, and then look up
    // the accountid for AllClients and use it.

    /**
     * If no error was returned, the AddContact results object will contain a
     * 'contactid' child SimpleXMLElement, which can be cast to an integer.
     */
    $accountid = (int) getAccountId($thrivecartid);
    if( $accountid == -1 )
    {
        return 'Failed to find accountid for ' . $thrivecartid;
    }
    $url = $api_endpoint . 'SetAccountStatus.aspx';
    $data = array(
        'apiusername' => $account_id,
        'apipassword' => $api_key,
        'accountid' => $accountid,
        'status' => (int)$new_status,
    );
    $result_xml_string = post_api_url($url, $data);
    $results_xml = simplexml_load_string($result_xml_string);

    if (isset($results_xml->error)) {
        return 'Failed with error ' . $results_xml->error;
    }
}
```

```
// Here I write the account information using addUser in mysql_common.php
$status = "inactive";
if( $new_status == 0 ) {
    $status = updateAccountStatus($accountid, 'inactive');
} else {
    $status = updateAccountStatus($accountid, 'active');
}
return $status;
}
?>
```

H post_api_url.php

This file was provided as a model by EngagemoreCRM to add users, change their status, etc. by the arguments provided.

```
<?php
/**
 * Post data to URL with cURL and return result XML string.
 *
 * Outputs cURL error and exits on failure.
 *
 * @param string $url
 * @param array $data
 *
 * @return string
 */
function post_api_url($url, array $data = array()) {
    global $nl;

    // Initialize a new cURL resource and set the URL.
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);

    // Form data must be transformed from an array into a query string.
    $data_query = http_build_query($data);

    // Set request type to POST and set the data to post.
```

```

curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data_query);

// Set cURL to error on failed response codes from AllClients server,
// such as 404 and 500.
curl_setopt($ch, CURLOPT_FAILONERROR, true);

// Set cURL option to return the response from the AllClients server,
// otherwise $output below will just return true/false.
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

// Post data to API.
$output = curl_exec($ch);

// Exit on cURL error.
if ($output === false) {
// It is important to close the cURL session after curl_error()
curl_close($ch);
    http_response_code(400);
    exit;
}

// Close the cURL session
curl_close($ch);

// Return response
return $output;
}
?>

```

I upgrade_account.php

If the user decides to change their subscription, they can upgrade or downgrade their experiences in EngagemoreCRM. This file handles those changes.

```
<?php
```

```
function upgrade_account($api_endpoint, $account_id, $api_key, $account,
```

```

    $group_name, $productid, $email) {
/**
 * Specify URL and form fields for AddContact API function.
 */
$url = $api_endpoint . 'SetAccountGroup.aspx';
$data = array(
'apiusername' => $account_id,
'apipassword'  => $api_key,
'accountid'    => $account, // the engagemore id
'group'        => $group_name,
);
$json_data = json_encode($_REQUEST);

$results_xml = thrivecart_api($url, $data); // returns simplexml_load_string object
if ($results_xml === false) {

    logit($email, $json_data, "failure parsing xml: ".$results_xml);
    http_response_code(400);
    exit;
}
if (isset($results_xml->message)) {
    logit($email,$json_data,"Upgrade result: $results_xml->message");
}
/**
 * If an API error has occurred, the results object will contain a child 'error'
 * SimpleXMLElement parsed from the error response:
 *
 * <?xml version="1.0"?>
 * <results>
 *     <error>Authentication failed</error>
 * </results>
 */

if (isset($results_xml->error)) {
    logit($email,$json_data, "Failure: " . $results_xml->error );
    http_response_code(400);
    exit;
}

```

```

    // Here I write the account information using updateProduct in mysql_common.php
    updateProduct($account, $productid);
    logit($email, $json_data, "Success upgraded account to: " . $productid);
    return $account;
}
?>

```

J git-info.php

This file is provided to show, in certain cases the revision, branch, and database that is being used for particular transactions.

```

<?php
    require 'config.ini.php';
    $last = exec('git log -1 --date=format:@"%Y/%m/%d" --format="%ad"');
    $rev = exec('git rev-parse --short HEAD');
    $branch = exec('git rev-parse --abbrev-ref HEAD');
    $server = $config['PATTI_DATABASE_SERVER'];
    echo "<center>Last Update: $last &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Commit: $rev &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Bra
?>

```