```lua
-------------------------------------------------------------------------------
-- TwilioRestClient.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Access the Twilio Rest API within a Corona aplication.
-- License: just friggin use it. Shoot me an email for any questions.
-------------------------------------------------------------------------------

local mime = require "mime"
local json = require("json")
local Util = require "Twilio.Util"

local TwilioRestClient = setmetatable({}, nil)
TwilioRestClient.__index = TwilioRestClient

-------------------------------------------------------------------------------
-- create() - public
-- @param acc_sid - type(string) - Your account SID.
-- @param acc_token - type(string) - Your account token.
-- @param [base_url] - type(string) - Optionally define base Twilio URL
-- @param api_version - type(string) - Optionally define API version.
-- @return - Your new TwilioRestClient instance.
-------------------------------------------------------------------------------
function TwilioRestClient.create(acc_sid, acc_token, base_url, api_version)
   assert(acc_sid and type(acc_sid) == "string",
          acc_token and type(acc_token) == "string",
          base_url == nil or type(base_url) == "string",
          api_version == nil or type(api_version) == "string",
        "Rest.create(acc_sid string, acc_token string[, "..
           "base_url string[, api_version string]]")

   base_url = base_url or "https://api.twilio.com"
   api_version = api_version or "2010-04-01"

   local instance = setmetatable({}, TwilioRestClient)
   instance.sid = acc_sid
   instance.token = acc_token
   instance.base = base_url
   instance.api_version = api_version

   return instance
end

-------------------------------------------------------------------------------
-- buildURL() - private
-- @param client - type(TwilioRestClient) - instance of your Twilio client.
-- @param arg - type(string) - any additional arguments will append to the
--              request URL.
-- @return - type(string) - Twilio http request URL, which in only valid if
--                          all args are Twilio valid.
-------------------------------------------------------------------------------
local function buildURL(client, ...)
    local out = client.base .. "/" .. client.api_version .. "/Accounts/" ..
client.sid
    for i,v in ipairs(arg) do
        out = out .. "/" .. tostring(v)
    end
    out = out .. ".json"
    return out
end

-------------------------------------------------------------------------------
-- formatPhoneNumber() - private
-- @param number - type(string) - phone number in Twilio format 11234567890
-- @return - type(string) - number with + in front if it doesn't have it.
-------------------------------------------------------------------------------
local function formatPhoneNumber(number)
    -- use '%2B' for +
    local s,e = string.find(number,"+")
    if not s then
        number = "+" .. number
    end
    return number
end



-------------------------------------------------------------------------------
-- buildURI() - private
-- @param vars - type(table) - table containing body uri data. For example,
--                             passing {Url=blah.com, To=123456789} returns
--                             'Url=blah.com&To=123456789'
-- @return - type(string)
-------------------------------------------------------------------------------
--Vars is a table containing body uri data ie passing
--{Url=blah.com, To=123456789} returns Url=blah.com&To=123456789
-- Used by post function
local function buildURI(vars)
    local out = ""
    local first = true
    for k,v in pairs(vars) do
        if first then
```

```
            first = false
        end
        --Prevent tables or other unwanted types
        if Util.typechk(v,"string","number") and
           Util.typechk(k,"string","number") then
            if k == "To" or k == "From" then v = formatPhoneNumber(v) end
            out = out .. (not first and "&" or "") .. tostring(k) .. "=" ..
tostring(v)
        end
    end
    return out
end

-------------------------------------------------------------------------------
-- network_middle_man() - private
-- @param e - type(table) - event recieved from Corona network.
-- @param listener - type(function) - user defined network listener
--  Used internally by TwilioRestClient to format data from server and handle
--  errors. Formats Twilio data for you :D
-------------------------------------------------------------------------------
local function network_middle_man(e, listener)
    --Convert json response into Lua table. Magic!
    e.response = json.decode(e.response)

    --Combine Twilio response errors (ie bad request by user)
    -- and HTTP errors as a general success status
    -- TODO: is this bad practice?
    e.success = (e.status < 300 and e.status > 199) and not e.isError
                and not e.response.code

    if not e.success then
        e.message = e.response.message
    end

    --Send formatted event data to user defined listener
    listener(e)
end


-------------------------------------------------------------------------------
-- post() - private
-- @param vars - type(table) - table containing body uri data. For example,
--                              passing {Url=blah.com, To=123456789} returns
--                              'Url=blah.com&To=123456789'
-- @return - type(string)
```

```lua
--------------------------------------------------------------------------------
local function post(client, vars, listener)
    local t = vars.Type
    vars.Type = nil
    local body = buildURI(vars)
    local url = buildURL(client,t)

    local headers = {}
    headers["Content-Type"] = "application/x-www-form-urlencoded"
    headers["Authorization"] =  "Basic "..mime.b64(client.sid..":"..client.token)

    local params = {}
    params.headers = headers
    params.body = body

    network.request(url, "POST", listener, params)
end

--------------------------------------------------------------------------------
-- get() - private
-- @param vars - type(table) - table containing body uri data. Get requests can
--                             optionally have an instance SID in which case no
--                             no body data is sent to Twilio except the ISid.
-- @return - type(string)
--------------------------------------------------------------------------------
--GET requests can optionally have an instance sid, in which
--case no body data is sent to twilio
function get(client, vars, listener)
   local t = vars.Type
   local ISid = vars.InstanceSid
   vars.Type = nil
   vars.InstanceSid = nil
   local url = buildURL(client, t, ISid)
   --If InstanceSid is provided, REST doesn't need any properties
   local body = ISid and "" or buildURI(vars)

   local headers = {}
   headers["Content-Type"] = "application/x-www-form-urlencoded"
   headers["Authorization"] =  "Basic "..mime.b64(client.sid..":"..client.token)

   local params ={headers = headers, body = body}

   network.request(url, "GET", listener, params)
end
```

```lua
-------------------------------------------------------------------------------
-- Request() - public
-- @param vars - type(table) - [key,val] pairs associated with a Twilio request
--                special key 'Type' defines what kind of request you're making.
-- @param method - type(string) - either "POST" or "GET".
-- @param listener - type(function) - function that accepts the http request
--                    events asyncronously.
-- @return the return value comes to the user defined listener(event) function
--          event.success specifies if the request was valid. Refer to
--          event.message for failure information. event.response is a lua table
--          containing your data returned from Twilio. Refer to
--          https://www.twilio.com/docs/api/rest for Twilio response infos.
-------------------------------------------------------------------------------
function TwilioRestClient:request(vars, method, listener)
    assert( type(vars) == 'table',
            method == "POST" or method == "GET",
            type(listener) == 'function',
            "TwilioRestClient:request(vars, method)" )
    vars = (vars and Util.DeepCopy(vars)) or {} --deep copy so I can modify in place
    local net_listener = function(e) network_middle_man(e,listener) end
    if method == "POST" then
        return pcall(function() post(self, vars, net_listener) end)
    elseif method == "GET" then
        return pcall(function() get(self, vars, net_listener) end)
    end
end

return TwilioRestClient
```