AGCA - REST

- C#
- Java
- PHP
- RoR
- Python
- ColdFusion
- VB
- TSQL
- NodeJS

Address GeoCoder Canada C# Rest Snippet

```
 try
            {
                result = HttpGet(mainURL);
                //NULL ERROR || FATAL ERROR RETURNED -- TRY BACKUP
                if (result == null || (result.Error != null && result.
Error.Number == "3"))
                {
                    return HttpGet(backupURL);
                }
                else
                {
                    return result;
                }
            }
            catch (Exception)
            {   //ERROR IN MAIN URL - USING BACKUP
                return HttpGet(backupURL);
            }

    public static GCCResponse HttpGet(string requestUrl)
        {
            try
            {
                System.Diagnostics.Debug.Write(requestUrl);
                HttpWebRequest request = WebRequest.Create(requestUrl) as
HttpWebRequest;
                request.Timeout = WEB_SERVICE_REQUEST_TIMEOUT;//timeout
for get operation
                using (HttpWebResponse response = request.GetResponse() as
HttpWebResponse)
                {
                    if (response.StatusCode != HttpStatusCode.OK)
                        throw new Exception(String.Format(
                        "Server error (HTTP {0}: {1}).",
                        response.StatusCode,
                        response.StatusDescription));
                    //prrocessing response
                    DataContractJsonSerializer jsonSerializer = new
DataContractJsonSerializer(typeof(GCCResponse));
                    object objResponse = jsonSerializer.ReadObject
(response.GetResponseStream());
                    GCCResponse jsonResponse = objResponse as GCCResponse;
                    return jsonResponse;
                }
            }
            catch (Exception e)
            {
                throw e;
            }
        }
```

Address GeoCoder Canada Java Code Snippet

```
JSONObject results = RestClient(mainURL);
if (ErrorMessages != null || (results.has("Error") && results.getJSONObject
("Error").get("Number") == "3")) {
    // BACKUP
    results = RestClient(backupURL);
}


return results;
```

Address GeoCoder Canada PHP Code Snippet

```
$URL = "https://trial.serviceobjects.com/GCC/api.svc/xml/GetGeoLocation?
Address=".urlencode($Address)."&Municipality=".urlencode($Municipality)."
&Province=".urldecode($Province)."&PostalCode=".urldecode($PostalCode)."
&LicenseKey=".urlencode($LicenseKey);

//use backup url once given purchased license key
$backupURL = "https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=".urlencode($Address)."&Municipality=".urlencode
($Municipality)."&Province=".urldecode($Province)."&PostalCode=".urldecode
($PostalCode)."&LicenseKey=".urlencode($LicenseKey);

    // Get cURL resource
    $curl = curl_init();
    curl_setopt_array($curl, array(CURLOPT_RETURNTRANSFER => 1,
CURLOPT_URL => $URL, CURLOPT_USERAGENT => 'Service Objects Geocoder
Canada'));
    curl_setopt($curl, CURLOPT_TIMEOUT, 50); //timeout in seconds
    // Send the request & save response to $resp
    $resp = curl_exec($curl);

    // Close request to clear up some resources
    if($resp == false)
    {
        echo "IN back up block";
        curl_setopt_array($curl, array(CURLOPT_RETURNTRANSFER => 1,
CURLOPT_URL => $backupURL, CURLOPT_USERAGENT => 'Service Objects Geocoder
Canada'));
        curl_setopt($curl, CURLOPT_TIMEOUT, 50); //timeout in seconds
        // Send the request & save response to $resp
        $resp = curl_exec($curl);
        if($resp == false)
        {
            echo "<b> Both rest calls failed </b>";
            curl_close($curl);
            return;
        }


    }
```

Address GeoCoder Canada RoR Code Snippet

```
@request = Request.find(params[:id])

#This sets the default timeout for HTTParty get operation. This must be
set in order to use the gem
default_timeout = 10
```

```
address = @request.address
municipality = @request.municipality
province = @request.province
postalcode = @request.postalcode
licensekey = @request.licensekey

#Set Primary and Backup URLs as needed. Normalizes the URL to that it can
be passed to the web service.
primaryURL = URI.encode("https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=" + address + "&Municipality=" + municipality +
"&Province=" + province + "&PostalCode=" + postalcode + "&LicenseKey=" +
licensekey)
backupURL = URI.encode("https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation=" + address + "&Municipality=" + municipality +
"&Province=" + province + "&PostalCode=" + postalcode + "&LicenseKey=" +
licensekey)

#These are set to access the hash that is returned
@agcaresult ="Location"
@agcaerror = "Error"

  #Begins the call the RESTful web service
begin

  response = HTTParty.get(primaryURL, timeout: default_timeout)
  #processes the response to display to the screen

  #Passes the response returned from HTTParty and processes them depending
on the results
  processresults(response)


 rescue StandardError => e
      begin
      #uses the backupURl in the event that the service encountered an
error
      response = HTTParty.get(URI.encode(backupURL), timeout:
default_timeout)

    #processes the response returned from using the backupURL
      processresults(response)
    #If the backup url railed this will raise an error and display the
    #error message returned from the HTTParty gem.
      rescue StandardError => error
          @status = error.message
          @displaydata = {"Error" => "A Big Error Occured"}
      end

end
```

Address GeoCoder Canada Python Code Snippet

```
#Set the primary and backup URLs as necessary
primaryURL = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?'
backupURL = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?'

#The
 Requests package allows the user to format the path parameters like so
instead of having to manually insert them into the URL
inputs = {'Address': mAddress, 'Municipality': mMunicipality, 'Province':
mProvince, "PostalCode": mPostalCode, 'LicenseKey': mLicenseKey}
try:
    result = requests.get(primaryURL, params=inputs)
    #Parses the XML response from the service into a python dictionary type
    outputs = xmltodict.parse(result.content)
    #checks the output for Errors and displays the info accordingly
    if 'Error' in outputs['Location']:
        #loops through the response from the service and prints the values
to the screen.
        for key, value in outputs['Location']['Error'].iteritems():
            Label(swin.window, text=str(key) + " : " + str(value)).pack()
    else:
        #Removes unnecessary values that were parsed from the original xml
response
        outputs['Location'].pop("@xmlns:xsi", None)
        outputs['Location'].pop("@xmlns:xsd", None)
        outputs['Location'].pop("@xmlns", None)
        for key, value in outputs['Location'].iteritems():
            Label(swin.window, text=str(key) + " : " + str(value)).pack()

#Uses the backup URL in the event that the primary URL failed
except:
    try:
        result = requests.get(backupURL, params=inputs)
        #Parses the XML response from the service into a python dictionary
type
        outputs = xmltodict.parse(result.content)
        #checks the output for Errors and displays the info accordingly
        if 'Error' in outputs['Location']:
            #loops through the response from the service and prints the
values to the screen.
            for key, value in outputs['Location']['Error'].iteritems():
                Label(swin.window, text=str(key) + " : " + str(value)).
pack()
        else:
            #Removes unnecessary values that were parsed from the original
xml response
            outputs['Location'].pop("@xmlns:xsi", None)
            outputs['Location'].pop("@xmlns:xsd", None)
```

```
            outputs['Location'].pop("@xmlns", None)
            for key, value in outputs['Location'].iteritems():
                Label(swin.window, text=str(key) + " : " + str(value)).
pack()
```

Address GeoCoder Canada ColdFusion Code Snippet

```
<!--Makes Request to web service --->
<cfIf isDefined("form.Action") AND Action neq "" >
    <cftry>
        <cfset primaryURL = "https://trial.serviceobjects.com/GCC/api.svc
/xml/GetGeoLocation?
Address=#Address#&Municipality=#Municipality#&Province=#Province#&PostalCod
e=#PostalCode#&LicenseKey=#LicenseKey#">
        <cfhttp url="#primaryURL#" method="get" result="response">
        <cfset outputs = XmlParse(response.FileContent)>
        <cfcatch>
            <cftry>
                <cfset backupURL = "https://trial.serviceobjects.com/GCC
/api.svc/xml/GetGeoLocation?
Address=#Address#&City=#City#&State=#State#&PostalCode=#PostalCode#&License
Key=#LicenseKey#">
                <cfhttp url="#backupURL#" method="get" result="response">
                <cfset outputs = XmlParse(response.FileContent)
>
                <cfcatch >
                    <cfoutput >
                        The Following Error Occured: #response.StatusCode#
                    </cfoutput>
                </cfcatch>
            </cftry>
        </cfcatch>
    </cftry>
</cfif>
```

Address GeoCoder Canada VB Code Snippet

```
'encodes the URLs for the get Call. Set the primary and back urls as
necessary
Dim primaryurl As String = "https://trial.serviceobjects.com/GCC/api.svc
/xml/GetGeoLocation?Address=" + address + "&Municipality=" + municipality
+ "&Province=" + province + "&PostalCode=" + postalcode + "&LicenseKey=" +
licensekey
Dim backupurl As String = "https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=" + address + "&Municipality=" + municipality +
"&Province=" + province + "&PostalCode=" + postalcode + "&LicenseKey=" +
licensekey
Dim wsresponse As AGCAResponse.Location = httpGet(primaryurl)

'checks if a response was returned from the service, uses the backup url
if response is null or a fatal error occured.
If wsresponse Is Nothing OrElse (wsresponse.[Error] IsNot Nothing AndAlso
wsresponse.[Error].Number = "3") Then
    wsresponse = httpGet(backupurl)
End If
If wsresponse.[Error] IsNot Nothing Then
    ProcessErrorResponse(wsresponse.[Error])
Else
    ProcessSuccessfulResponse(wsresponse)

End If
```

Address GeoCoder Canada TSQL Code Snippet

```
IF @isLiveKey = 1
BEGIN
    SET @sUrl = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=' + @address + '&Municipality=' + @municipality +
'&Province=' + @province + '&PostalCode=' + @postalcode + '&LicenseKey=' +
@key
    EXEC sp_OACreate 'MSXML2.ServerXMLHttp', @obj OUT
    EXEC sp_OAMethod @obj, 'Open', NULL, 'Get', @sUrl, false
    EXEC sp_OAMethod @obj, 'send'
    EXEC sp_OAGetProperty @obj, 'responseText', @response OUT

    --Checks the Response for a fatal error or if null.
    IF @response IS NULL
    BEGIN
        SET @sBackupUrl = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=' + @address + '&Municipality=' + @municipality +
'&Province=' + @province + '&PostalCode=' + @postalcode + '&LicenseKey=' +
@key
        EXEC sp_OACreate 'MSXML2.ServerXMLHttp', @obj OUT
        EXEC sp_OAMethod @obj, 'Open', NULL, 'Get', @sBackupUrl, false
        EXEC sp_OAMethod @obj, 'send'
        EXEC sp_OAGetProperty @obj, 'responseText', @response OUT
    END
END
```

Address GeoCoder Canada NodeJS Code Snippet

```
var Address = 'Address';
var Municipality ='Municipality';
var Province ='Province';
var PostalCode ='PostalCode';
var LicenseKey ='LicenseKey';
//Set backup and primary URL as necessary
var primaryUrl = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=' + Address + '&Municipality=' + Municipality
+'&Province=' + Province +'&PostalCode=' + PostalCode +'&LicenseKey=' +
LicenseKey;
var backupUrl = 'https://trial.serviceobjects.com/GCC/api.svc/xml
/GetGeoLocation?Address=' + Address + '&Municipality=' + Municipality
+'&Province=' + Province +'&PostalCode=' + PostalCode +'&LicenseKey=' +
LicenseKey;

var req = http.get(primaryUrl, function(res) {
    res.setEncoding('utf8');
    res.on('data', function (results) {
        var parser = require('xml2js').Parser({explicitArray: false,
ignoreAttrs: true});
```

```
        parser.parseString(results, function (err, outputs) {
            if (outputs.Location.Error !=  null)
            {
                //Indicates a Fatal error has occured. If this happens,
the logic will then failover to the backup url
                if (outputs.Location.Error.Number == "4")
                {
                    var backupReq = http.get(backupUrl, function
(backupRes) {
                        backupRes.setEncoding('utf8');
                        backupRes.on('data', function (results) {
                            var parser = require('xml2js').Parser
({explicitArray: false,ignoreAttrs: true});
                            parser.parseString(results, function (err,
outputs) {
                                console.log("Backup Call Was Used.");
                                response.end(JSON.stringify(outputs ,
null, 3));
                                });
                            });
                        });
                }
                else
                {
                    //Will Display the JSON Formatted Error Response here
                    response.end(JSON.stringify(outputs, null, 3));
                    return;
                }
            }
            else
            {
                //Will Display the JSON Formatted Valid Response here
                response.end(JSON.stringify(outputs, null, 3));
                return;
            }
        });
    });
});
```