DataObjectException

Lessons from a Cloud data migration journey with Dynamics CRM on-premises

Resolving a refresh token issue with Google API

Version 1.0, September 2019

DataObjectException

Thierry Sinassamy (M.Sc.)
Architect & Developer

https://www.yuzuai.ca/
https://github.com/thierry-sinassamy
https://www.linkedin.com/in/thierry-sinassamy-19467416/

Disclaimer

The white paper does not provide any confidential client data related to data migration from the Dynamics CRM “on-premises” to the DFP Publishers “Google Cloud”. But the white paper describes how I resolved the authentication issue (Google API OAuth2 provider).
The purpose is to describe the methodology and the orchestration used to troubleshoot and resolve the issue regarding the time constraints.

Abstract

The **Dynamics CRM (on-premises)** data migration failed following an update to Windows Server 2003 (security patches). Precisely, the **process of security token refresh of** Google Service (DFP Publishers) failed.

Resolving this authentication issue consisted of a 3-step approach:

- **Preliminary validation process**: Windows Service (listener) and data migration Servers (event viewer) logs.
- **Problem isolation process**: Windows Service application as an executable, Debugging of the .Net Library of Google API, Physical environment of the Windows Service and finally, the users and the domain controller.
- **Resolution process related to Windows Service physical environment**. It was necessary to develop a new service (as a listener) to transfer the Dynamics CRM data to the new physical environment (Windows Server 2008/2012) and in a new queue. Once the data is transferred, the service (listener) deployed in the new physical server will be able to retrieve data from the queue and will be able to migrate data to Google Cloud.

Contents

CHAPTER 1

INTRODUCTION

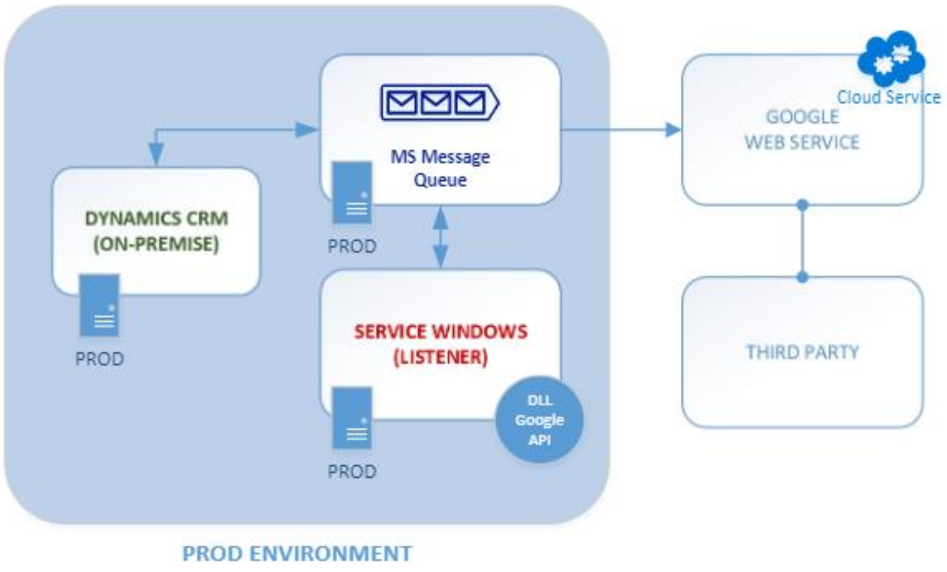
Dynamics CRM on-premises, Google Web Services, Google Premium and Data Migration

CONTEXT

Google API CRM and OAuth2 Provider Data Migration

Introduction

The architecture is composed of Dynamics CRM (On-Premises), Windows Services (Listener), Microsoft Message Queue, Google Premium (Google Cloud Service) and Third-Party (DFP Publishers).



Context

Accounts and contacts data had to be retrieved from CRM and placed in the Microsoft Message Queue (MSMQ) data storage directory. Data was exported in XML format and stored in a directory created for this purpose. The service (Windows Service - .NET application developed in C#, deployed in Windows Server 2003) behaved as a "Listener" in order to retrieve the data stored in the MSMQ directory.
Once retrieved, the "Listener" had to authenticate to Google Premium Web Services in order to import the data (CRM accounts and contacts) into the online service directories (Google Premium).

Once the data was extracted from the MSMQ directory to the Google Premium directories, the data was cleaned from the MSMQ directory.
However, in the event of an error or exception generated by Google's Web services, the data was stored in an MSMQ directory created for this type of situation.

Finally, the "Listener" had to establish communication with Google's Web services and for that purpose , a security token ("Token") had to be generated and the refresh of that "Token" was also generated during the transaction.
For further information, the Google API Web service uses the OAuth 2.0 "Framework" which specifies several types of authorization (Refresh Token, Client Credentials, etc.).

Background

Google API CRM and OAuth2 Provider Data Migration

CHAPTER 2

PROBLEM

Refreshing security token during the call of the Web Service is failing.

Refreshing the token during data migration

How does the authentication process work?

- **Step 1**: The "**Watcher**" retrieves or connects to Google's Web service without problem.
- **Step 2**: Once the service is retrieved, the Watcher executes a **CRUD** query (CREATE, READ, UPDATE, DELETE) i.e., runs a read on the DFP Publishers to determine if the newly entered contact already exists in DFP Publishers.
- **Step 3**: In this case, it will then execute a request of type "creation" of contact, otherwise it will be a request of type "update" of the contact.

Hence, the problem of refreshing the security token occurs **in all environments (DEV, TEST, PROD)** when reading the "Watcher" in DFP Publishers since authentication is required at this stage.

Impact of updating security patches in the data migration server

The upgrade of security "patches" in the data migration server (Windows Server 2003) had a negative impact on the authentication process of the Google Web service related to the refresh of the "Token" (security token).

Listener Log

The existing log file describes the problem as follows: "**Failed to refresh access token**". This is an "application" type error that is launched by the "3rd-Party" (DFP) used by Google's Web service to authenticate and execute "**CRUD**" type queries (CREATE, READ, UPDATE, DELETE). This error occurs precisely in the authentication provider of this "Third-Party" (DFP Publishers).

The security token could no longer be refreshed according to the contents of the error generated by the .NET library "**Google.Api.Ads.dll**":

Content of the stacktrace

```
« at Google.Api.Ads.Common.Lib.OAuth2ProviderForApplications.RefreshAccessTokenInOfflineMode () in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\OAuth2ProviderForApplications.cs:line 165
at Google.Api.Ads.Common.Lib.OAuth2ProviderForApplications.RefreshAccessToken() in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\OAuth2ProviderForApplications.cs:line 200
at Google.Api.Ads.Common.Lib.OAuth2ProviderBase.RefreshAccessTokenIfExpiring() in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\OAuth2ProviderBase.cs:line 249
at Google.Api.Ads.Common.Lib.OAuth2ProviderBase.GetAuthHeader() in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\OAuth2ProviderBase.cs:line 240
  at Google.Api.Ads.Dfp.Lib.DfpSoapClient.InitForCall(String methodName, Object[] parameters) in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\DfpSoapClient.cs:line 112
at Google.Api.Ads.Common.Lib.AdsSoapClient.MakeApiCall(String methodName, Object[] parameters) in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\AdsSoapClient.cs:line 211
At Google.Api.Ads.Common.Lib.AdsSoapClient.Invoke(String methodName, Object[] parameters) in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\Lib\AdsSoapClient.cs:line 127
at Google.Api.Ads.Dfp.v201408.CompanyService.getCompaniesByStatement(Statement filterStatement) in
  z:\Git\google3\third_party\dotnet_src\adsapi\compile\v201408\DfpApi.cs:line 7004
  at ...GooglePremium.Sync(BaseCompanie item, String cie, StreamWriter swFileTrace)
  in C:\Users\...\Documents\Visual Studio 2010\Projects\...\......cs:line 113 ».
```

Data Migration server and log

The **Event Viewer** of the Data Migration Server (Windows Server 2003) has an error in connection with the execution of the Listener and the **INFRA domain** in which the Listener was located:

Content of the event viewer

> “The Message Queuing service will not join the INFRA domain. An MSMQ Configuration (msmq) object exists in the new domain with an ID differing from the service ID. Please delete the MSMQ Configuration object in the new domain, restart the Message Queuing service, and log on again (Event ID = 2164)”

Related issues

There are also 3 points that should be considered in the "Troubleshooting" process of the problem: the lack of space in the server, the blocking of the server's "Firewall" and the anti-virus "McAfee".

CHAPTER 3

Preliminary solution validation process

With.NET library "**Google.Api.Ads.dll**", Refreshing the security token is a failure.

Preliminary Solution Validation Process

Listener: log file

- At first, the necessary information for the authentication process had to be regenerated from the tool "**OAuth2TokenGenerator**.exe[1]. I therefore regenerated the value of the "**refresh_token"**.

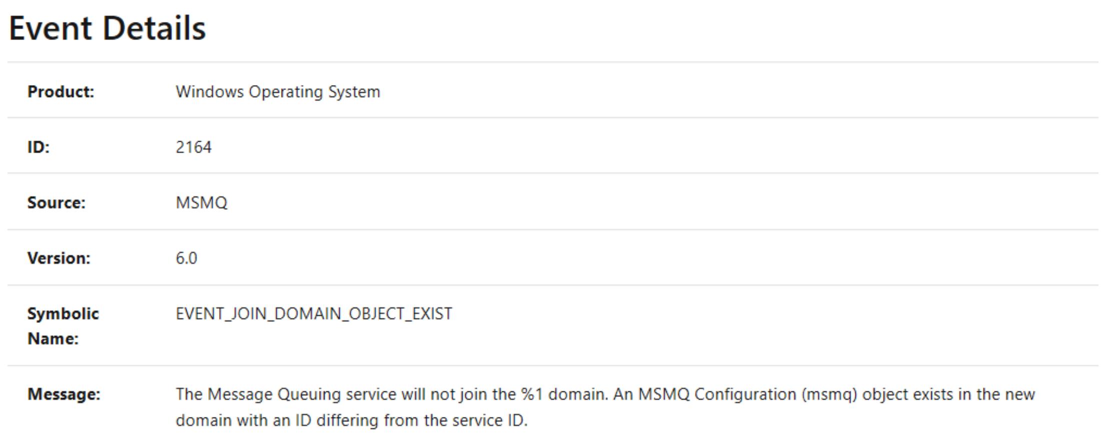
- Then, if the authentication settings had changed on the Google web services side, we ran a series of tests by extracting and adding the settings mentioned on the Google website. However, according to Google's website, the parameter that seemed to be missing was "**grant_type**" and by adding it, it didn't work.

Note: These parameters are added in the code of the "Third Party" used by Google's Web service and therefore all the parameters necessary for the process seemed to be already present.
The Third-Party code handles it in the "**RefreshAccessTokenInOfflineMode**()" method (line 163 of the **OAuth2ProviderForApplications**.cs file) that is, it fills it with the value of the "**refresh_token**".

Event viewer: data migration server

After much research, this error 2164 (associated with the "Watcher") led us to apply the solution proposed by Microsoft [2], but that did not work. However, error 2164 no longer appears.

> "Stale objects can prevent the MSMQ Service from operating properly. Deleting stale objects may solve this problem. However, deleting a computer object in Active Directory Domain Services (AD DS) can cause problems on the client computer.
>
> Before deleting the computer object, make sure that no services running on the client computer will be affected. In this case, deleting the Message Queuing Active Directory object will delete public queues on that computer.
>
> You must have the Active Directory services tools installed in Role Administration tools under Remote Server Administration. To perform these procedures, you must have membership in **Administrators**, or you must have been delegated the appropriate authority."



Event Details

Product:	Windows Operating System
ID:	2164
Source:	MSMQ
Version:	6.0
Symbolic Name:	EVENT_JOIN_DOMAIN_OBJECT_EXIST
Message:	The Message Queuing service will not join the %1 domain. An MSMQ Configuration (msmq) object exists in the new domain with an ID differing from the service ID.

Other issues

- The **lack of space** on the server's hard drive has led us to increase its space;
- The "**Firewall**" of the 2003 server was disabled;
- **McAfee** anti-virus software has been disabled.

Moreover, the server has been restarted repeatedly. However, the refresh token failed again.

Refreshing Issue

Regenerating the token failed again

Log

Logs display: “Failed to refresh access token”

Security Token

AdsOAuthException is thrown when the refresh token method calls the oken endpoint with the client_id, the client_secret and the refresh_token

Dynamic Link Library

Google.API.Ads.dll as a third-party
And stacktrace page 11

Validation

Validation of the logs (“listener”, data migration server, disk space, firewall, anti-virus)

Listener

As a windows Service, the .NET listener will get the data from the Queue and will call the Web Service

What’s next?

“After validating the logs, we really need to isolate the issue : authentication problem in a refresh process”…

CHAPTER 4

ISOLATION PROCESS

Start the first steps of isolation...and debug...

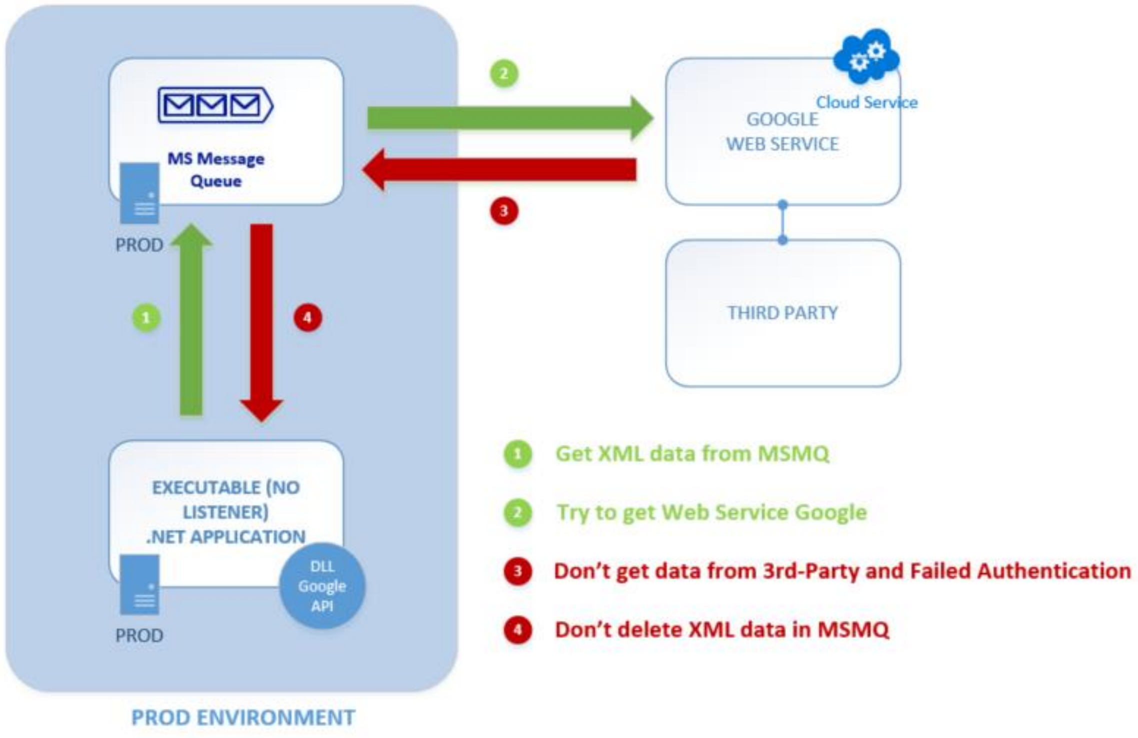
Authentication problem isolation process

Windows Service (.exe)

The purpose is to validate if the problem was directly related to the Windows service (listener). In the production environment (10.255.2...2006), two validation tests were carried out:

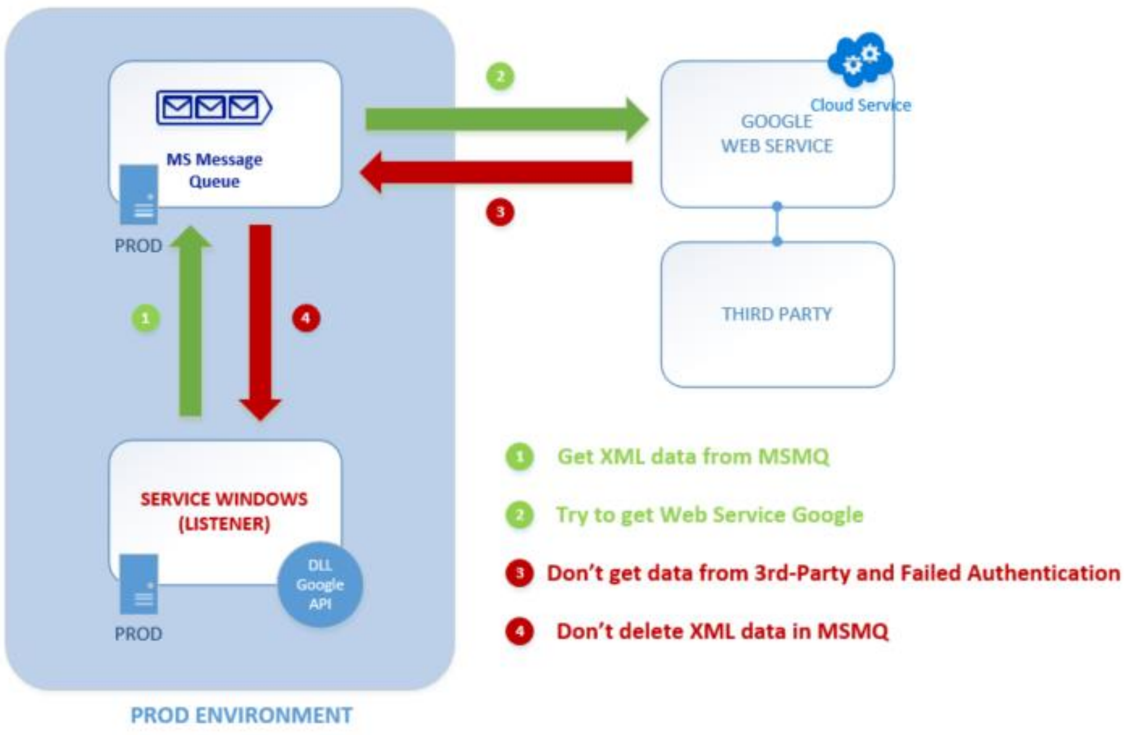
First isolation test

An application that was not of the "Windows Service" (Listener) type was deployed in this environment and run from the same environment. However, the validation test generates the same error as above "**Failed to refresh access token**".


MS Message Queue
PROD
Cloud Service
GOOGLE WEB SERVICE
THIRD PARTY
EXECUTABLE (NO LISTENER) .NET APPLICATION
DLL Google API
PROD
PROD ENVIRONMENT
1 Get XML data from MSMQ
2 Try to get Web Service Google
3 Don't get data from 3rd-Party and Failed Authentication
4 Don't delete XML data in MSMQ

Second isolation test

The application ("Windows Service"/Listener) was deployed in the same environment and run from the same environment. However, the second validation test gets the same error as above "**Failed to refresh access token**".


MS Message Queue
PROD
SERVICE WINDOWS (LISTENER)
DLL Google API
PROD
PROD ENVIRONMENT
GOOGLE WEB SERVICE
Cloud Service
THIRD PARTY
1 Get XML data from MSMQ
2 Try to get Web Service Google
3 Don't get data from 3rd-Party and Failed Authentication
4 Don't delete XML data in MSMQ

First isolation test and second isolation test…

Don't get the data from the third-party (Google) is directly linked to the failed authentication. First and second isolation tests are not enough, so what's next?

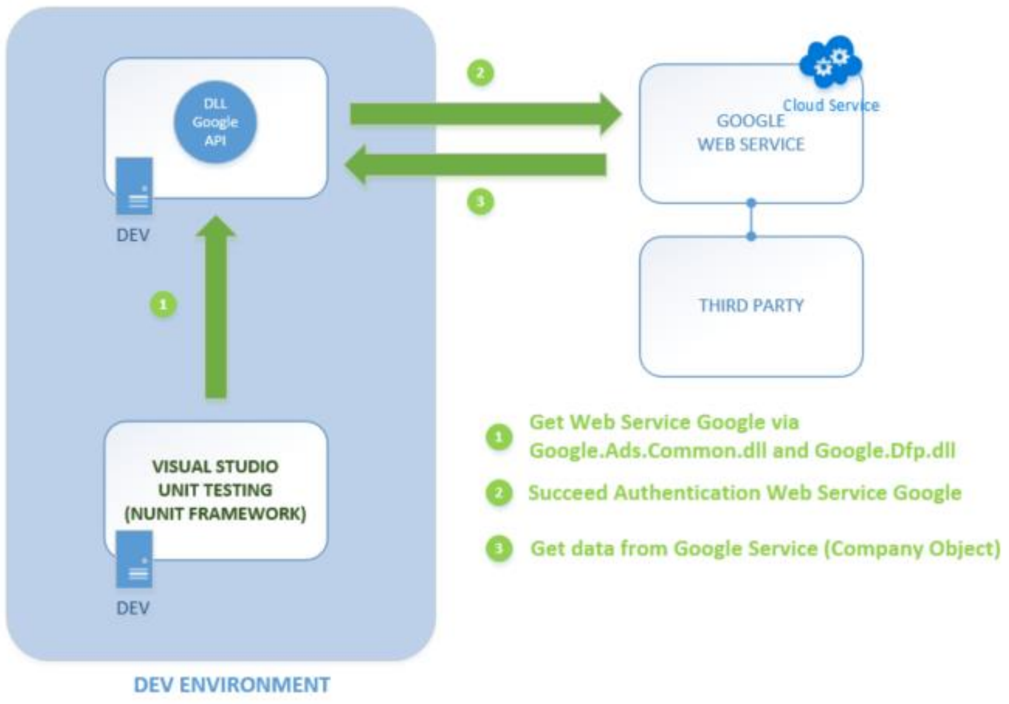
Third-Party and Google Web Services

The purpose is to know if the issue was directly related to the .NET libraries of the Google API. To do this, two validation methods were possible: **Unit Testing and Remote Debugging**.

Unit Testing

Unit tests were performed on the "Third-Party" used by the Google Web service to which the "Watcher" used when requesting DFP Publisher. The aim was to test the methods of the DLL "**Google.Dfp.dll**" in practice. In concrete terms, I have retrieved the code from the "GitHub" site and performed unit tests on this solution (Visual Studio).

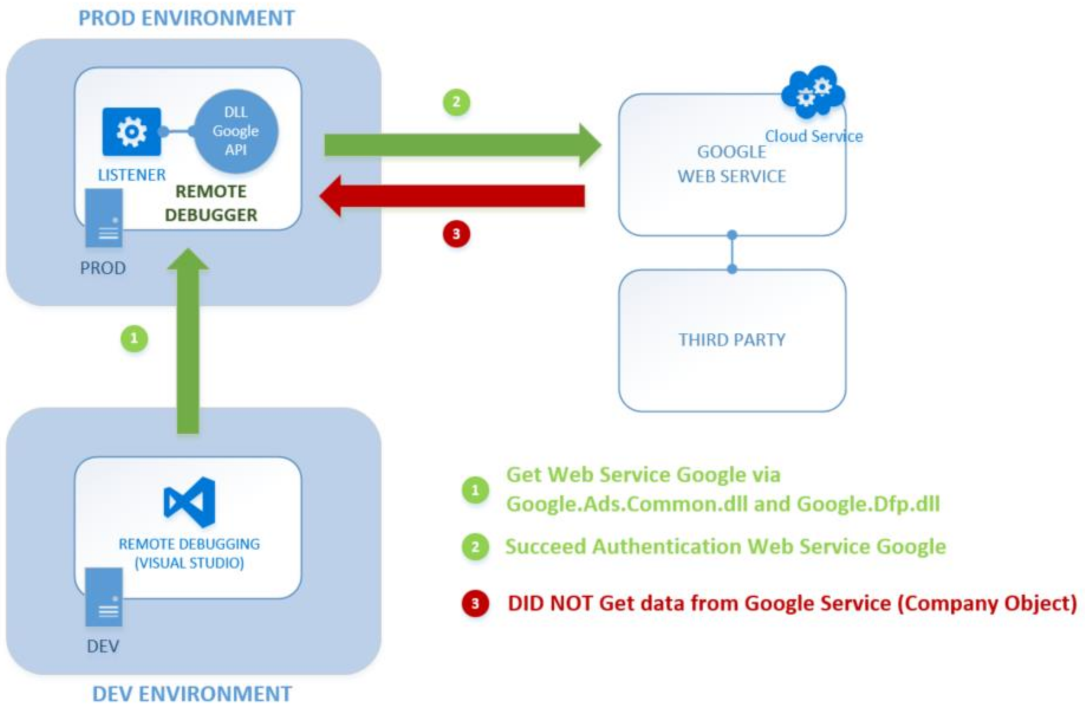
This test allowed me to test the classes and methods that were mentioned in the error (see the appendix page which presents the details of the exception). Also, the annex page presents a series of images showing the execution of unit tests (recovery of web services, initialization of services, generation of security token and refresh of security token).



DLL Google API
DEV
2
3
Cloud Service
GOOGLE WEB SERVICE
THIRD PARTY
1
VISUAL STUDIO UNIT TESTING (NUNIT FRAMEWORK)
DEV
DEV ENVIRONMENT
1 Get Web Service Google via Google.Ads.Common.dll and Google.Dfp.dll
2 Succeed Authentication Web Service Google
3 Get data from Google Service (Company Object)

Remote debugging

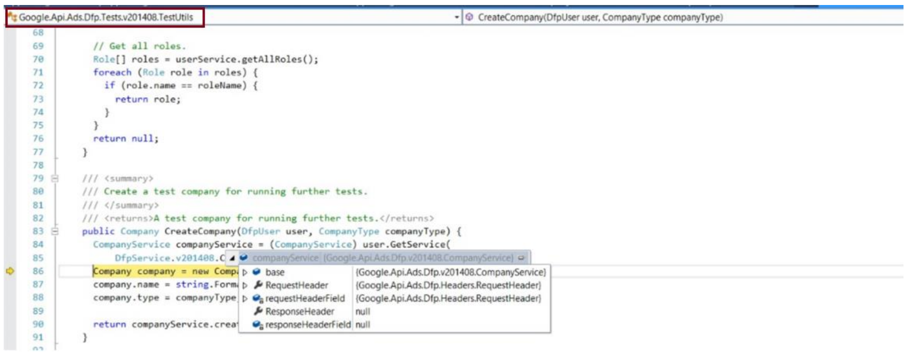
With the "Remote debugger" of Visual Studio 2010, I debugged the code of the "Watcher" to retrieve more information. However, I have only retrieved the information already mentioned.



PROD ENVIRONMENT
LISTENER
DLL Google API
REMOTE DEBUGGER
PROD
2
3
Cloud Service
GOOGLE WEB SERVICE
THIRD PARTY
1
REMOTE DEBUGGING (VISUAL STUDIO)
DEV
DEV ENVIRONMENT
1 Get Web Service Google via Google.Ads.Common.dll and Google.Dfp.dll
2 Succeed Authentication Web Service Google
3 DID NOT Get data from Google Service (Company Object)

Phases of testing and debugging process

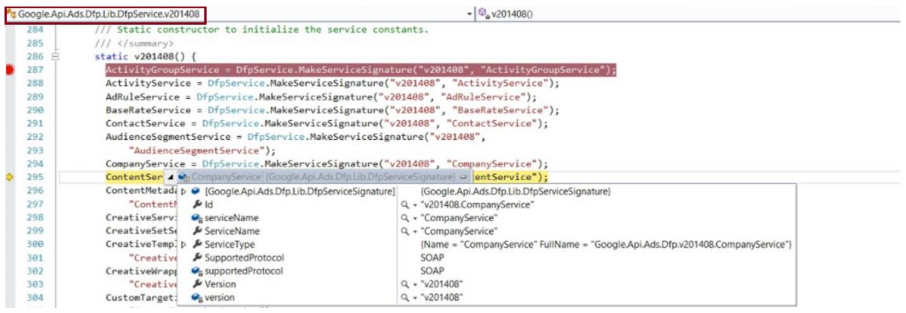
1- Get the Web services (DFP Publishers)



```
Google.Api.Ads.Dfp.Tests.v201408.TestUtils        CreateCompany(DfpUser user, CompanyType companyType)

      // Get all roles.
      Role[] roles = userService.getAllRoles();
      foreach (Role role in roles) {
        if (role.name == roleName) {
          return role;
        }
      }
      return null;
    }

    /// <summary>
    /// Create a test company for running further tests.
    /// </summary>
    /// <returns>A test company for running further tests.</returns>
    public Company CreateCompany(DfpUser user, CompanyType companyType) {
      CompanyService companyService = (CompanyService) user.GetService(
          DfpService.v201408.C
      Company company = new Comp
      company.name = string.Form
      company.type = companyType

      return companyService.creat
    }
```

```
companyService {Google.Api.Ads.Dfp.v201408.CompanyService}
  base                  {Google.Api.Ads.Dfp.v201408.CompanyService}
  RequestHeader         {Google.Api.Ads.Dfp.Headers.RequestHeader}
  requestHeaderField    {Google.Api.Ads.Dfp.Headers.RequestHeader}
  ResponseHeader        null
  responseHeaderField   null
```

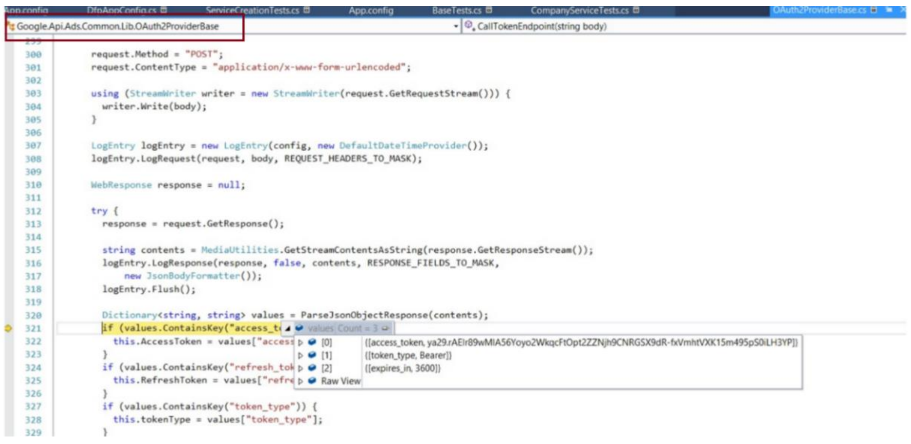
2- Initialize the services



```
Google.Api.Ads.Dfp.Lib.DfpService.v201408        v201408()

      /// Static constructor to initialize the service constants.
      /// </summary>
      static v201408() {
        ActivityGroupService = DfpService.MakeServiceSignature("v201408", "ActivityGroupService");
        ActivityService = DfpService.MakeServiceSignature("v201408", "ActivityService");
        AdRuleService = DfpService.MakeServiceSignature("v201408", "AdRuleService");
        BaseRateService = DfpService.MakeServiceSignature("v201408", "BaseRateService");
        ContactService = DfpService.MakeServiceSignature("v201408", "ContactService");
        AudienceSegmentService = DfpService.MakeServiceSignature("v201408",
            "AudienceSegmentService");
        CompanyService = DfpService.MakeServiceSignature("v201408", "CompanyService");
        ContentSer                                                  entService");
        ContentMetad
            "Content
        CreativeServ
        CreativeSetS
        CreativeTemp
            "Creativ
        CreativeWrap
            "Creativ
        CustomTarget
```

```
CompanyService {Google.Api.Ads.Dfp.Lib.DfpServiceSignature}
  [Google.Api.Ads.Dfp.Lib.DfpServiceSignature]   {Google.Api.Ads.Dfp.Lib.DfpServiceSignature}
  Id                     "v201408.CompanyService"
  serviceName            "CompanyService"
  ServiceName            "CompanyService"
  ServiceType            {Name = "CompanyService" FullName = "Google.Api.Ads.Dfp.v201408.CompanyService"}
  SupportedProtocol      SOAP
  supportedProtocol      SOAP
  Version                "v201408"
  version                "v201408"
```

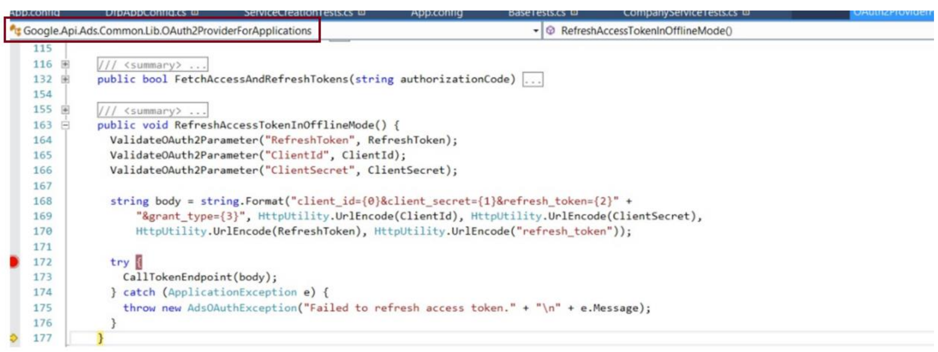
3- Generate the security token



```
App.config   DfpAppConfig.cs   ServiceCreationTests.cs   App.config   BaseTests.cs   CompanyServiceTests.cs   OAuth2ProviderBase.cs
Google.Api.Ads.Common.Lib.OAuth2ProviderBase        CallTokenEndpoint(string body)

      request.Method = "POST";
      request.ContentType = "application/x-www-form-urlencoded";

      using (StreamWriter writer = new StreamWriter(request.GetRequestStream())) {
        writer.Write(body);
      }

      LogEntry logEntry = new LogEntry(config, new DefaultDateTimeProvider());
      logEntry.LogRequest(request, body, REQUEST_HEADERS_TO_MASK);

      WebResponse response = null;

      try {
        response = request.GetResponse();

        string contents = MediaUtilities.GetStreamContentsAsString(response.GetResponseStream());
        logEntry.LogResponse(response, false, contents, RESPONSE_FIELDS_TO_MASK,
            new JsonBodyFormatter());
        logEntry.Flush();

        Dictionary<string, string> values = ParseJsonObjectResponse(contents);
        if (values.ContainsKey("access_t
          this.AccessToken = values["access
        }
        if (values.ContainsKey("refresh_tok
          this.RefreshToken = values["refre
        }
        if (values.ContainsKey("token_type")) {
          this.tokenType = values["token_type"];
        }
```

```
values Count = 3
  [0]        {[access_token, ya29.rAEIr89wMIA56Yoyo2WkqcFtOpt2ZZNjh9CNRGSX9dR-fxVmhtVXK15m495pS0iLH3YP]}
  [1]        {[token_type, Bearer]}
  [2]        {[expires_in, 3600]}
  Raw View
```

4- Refresh the security token



```
App.config   DfpAppConfig.cs   ServiceCreationTests.cs   App.config   BaseTests.cs   CompanyServiceTests.cs   OAuth2Provider
Google.Api.Ads.Common.Lib.OAuth2ProviderForApplications        RefreshAccessTokenInOfflineMode()

    /// <summary> ...
    public bool FetchAccessAndRefreshTokens(string authorizationCode) ...

    /// <summary> ...
    public void RefreshAccessTokenInOfflineMode() {
      ValidateOAuth2Parameter("RefreshToken", RefreshToken);
      ValidateOAuth2Parameter("ClientId", ClientId);
      ValidateOAuth2Parameter("ClientSecret", ClientSecret);

      string body = string.Format("client_id={0}&client_secret={1}&refresh_token={2}" +
          "&grant_type={3}", HttpUtility.UrlEncode(ClientId), HttpUtility.UrlEncode(ClientSecret),
          HttpUtility.UrlEncode(RefreshToken), HttpUtility.UrlEncode("refresh_token"));

      try {
        CallTokenEndpoint(body);
      } catch (ApplicationException e) {
        throw new AdsOAuthException("Failed to refresh access token." + "\n" + e.Message);
      }
    }
```

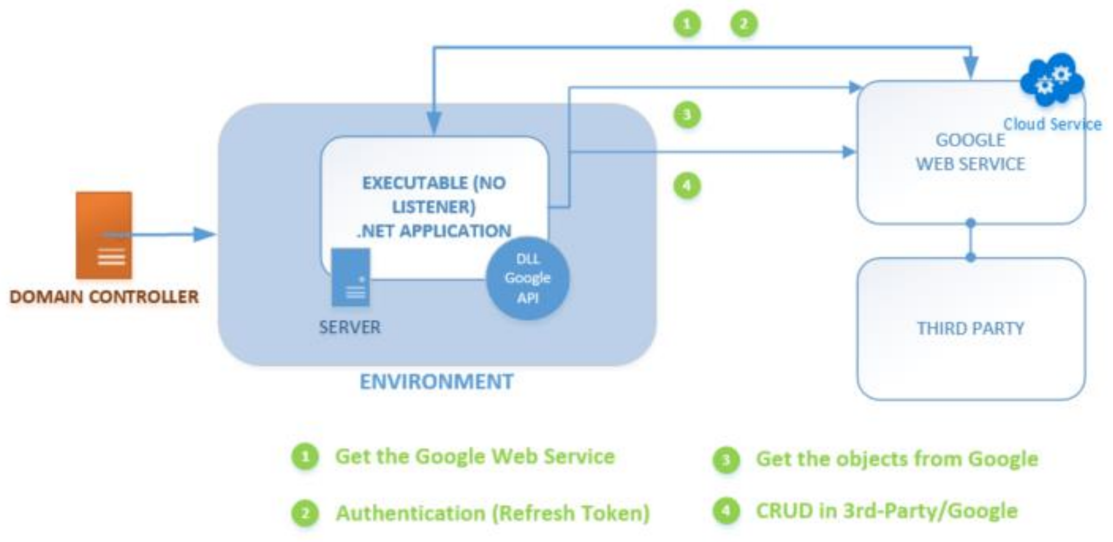
PHASES OF TESTING AND DEBUGGING IN DEV & PROD

SO WHAT'S HAPPENING AFTER THE STEP 3, ONCE THE TOKEN HAS BEEN GENERATED?

Phases	DEV/xxx	PROD
1- Get Web Services	V	V
2- Initialize Services	V	V
3- Generate Token	V	V
4- ReRefresh token	X	X

Physical environment of the Listener

The purpose is to run the Listener as a .NET executable in different Windows Server operating systems, considering the different Domain Controller which is a server responsible for network security and acts as a guardian for user authentication and authorization. The results were mixed, both positive and negative.


DOMAIN CONTROLLER
EXECUTABLE (NO LISTENER) .NET APPLICATION
DLL Google API
SERVER
ENVIRONMENT
GOOGLE WEB SERVICE
Cloud Service
THIRD PARTY
1 Get the Google Web Service
2 Authentication (Refresh Token)
3 Get the objects from Google
4 CRUD in 3rd-Party/Google

ENVIRONMENT	SERVER	DOMAIN CONTROLLER (DC)	RESULT
DEV/LOCAL	WINDOWS SERVER 2012 R2	N/A	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
DEV	WINDOWS SERVER 2008/2012	DC INFRA	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
TEST	WINDOWS SERVER 2008/2012	DC INFRA II	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
DEV	WINDOWS SERVER 2003	DC INFRA	AUTHENTICATION/REFRESH TOKEN IS A FAILURE
TEST	WINDOWS SERVER 2003	DC INFRA II	AUTHENTICATION/REFRESH TOKEN IA S FAILURE

Windows Server 2008/2012!

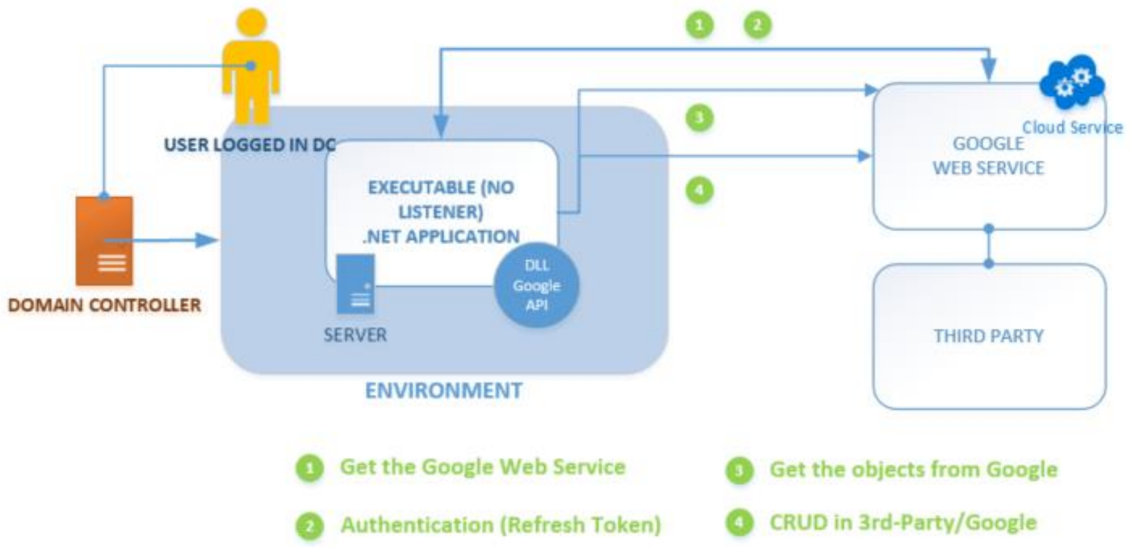
Windows 2008/2012R2 are the key to move forward.

The "pattern" occurred naturally, and the error was therefore related to the servers in which the latest security updates or patches were executed.

That said, in order to confirm our validations, it was necessary to determine whether the security token refresh problem was not also related to the logged-in user.

Logged-in user in servers and Domain Controller

As before, the goal is to run the Listener as a .NET executable in different Windows Server operating systems, considering the different Domain Controller systems. However, this time we take account of the user connected via the "Domain Controller".



USER LOGGED IN DC
DOMAIN CONTROLLER
EXECUTABLE (NO LISTENER) .NET APPLICATION
DLL Google API
SERVER
ENVIRONMENT
GOOGLE WEB SERVICE
Cloud Service
THIRD PARTY
1 Get the Google Web Service
2 Authentication (Refresh Token)
3 Get the objects from Google
4 CRUD in 3rd-Party/Google

USER LOGGED	ENVIRONMENT	SERVER	DOMAIN CONTROLLER (DC)	RESULT
Super Admin Réseau in DC, Admin in DC, Admin in Local Computer [3]	DEV/LOCAL	WINDOWS SERVER 2012 R2	N/A	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
	DEV	WINDOWS SERVER 2008/2012	DC INFRA	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
	TEST	WINDOWS SERVER 2008/2012	DC INFRA II	AUTHENTICATION/REFRESH TOKEN IS A SUCCESS
	DEV	WINDOWS SERVER 2003	DC INFRA	AUTHENTICATION/REFRESH TOKEN IS A FAILURE
	TEST	WINDOWS SERVER 2003	DC INFRA II	AUTHENTICATION/REFRESH TOKEN IS A FAILURE

Approaching the solution to implement!

Apparently, the process of refreshing the security token is a success in the other operation systems (Windows Server 2008/2012), so what’s next?

CHAPTER 5

RESOLUTION PROCESS

Provision a new Windows Services, a new listener ...

Resolution process: Provision a new listener in the Architecture

Following the numerous tests carried out in the 2003 and 2008/2012 servers, we established that a "pattern" seemed obvious and that it became undeniable that the "**Listener**" (as a Windows Service) should be moved to another server (2008/2012).

The idea was to create a **bridge between the two "Message Queue" servers in 2003 and 2008/2012** while maintaining the Dynamics CRM platform as it is and the current code of the "Listener".

Once the new listener (Listener A) has been developed and deployed in the environment (source), it will be able to transfer the data to the other listener (Listener B). The last one will be able to transfer the data to the Cloud (Google Web Service). Here's below the steps of the resolution process:

1 The CRM data is transferred in XML format to a private queue of the MSMQ of the 2003 server (Windows Server 2003).

2 The new bridge that is a Windows "**ListenerBridge**" service (Listener A) retrieves the message from the MSMQ server 2003.

3 The new "**ListenerBridge**" bridge transfers to the MSMQ private queue of another server (Windows Server 2008/2012).

4 Once the transfer is complete, the "**ListenerBridge**" clears the message sent from the MSMQ on the 2003 server and processes the other messages in the queue.

5 The old Windows service "MSMQ Listener" (Listener B), which is in the 2008/2012 server, retrieves the new message from the private queue of the same name.

6 The "MSMQ Listener" service connects to the Google[4] service and retrieves all objects (CompanyService, etc.).

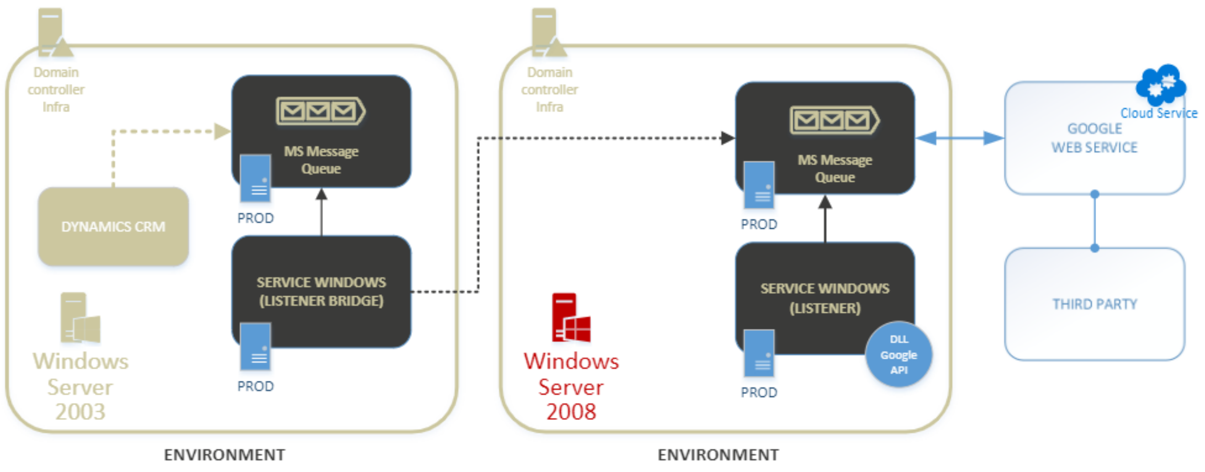
7 Once the specific service is retrieved (CompanyService, etc.), the Listener executes a “GET” request of type to validate if the contact and account mentioned in the message already exist in the directory "Cloud".

8

Once the validation of step 7 has been completed, the service creates or modifies the contact or account in Google's "Cloud" universe (POST type query).

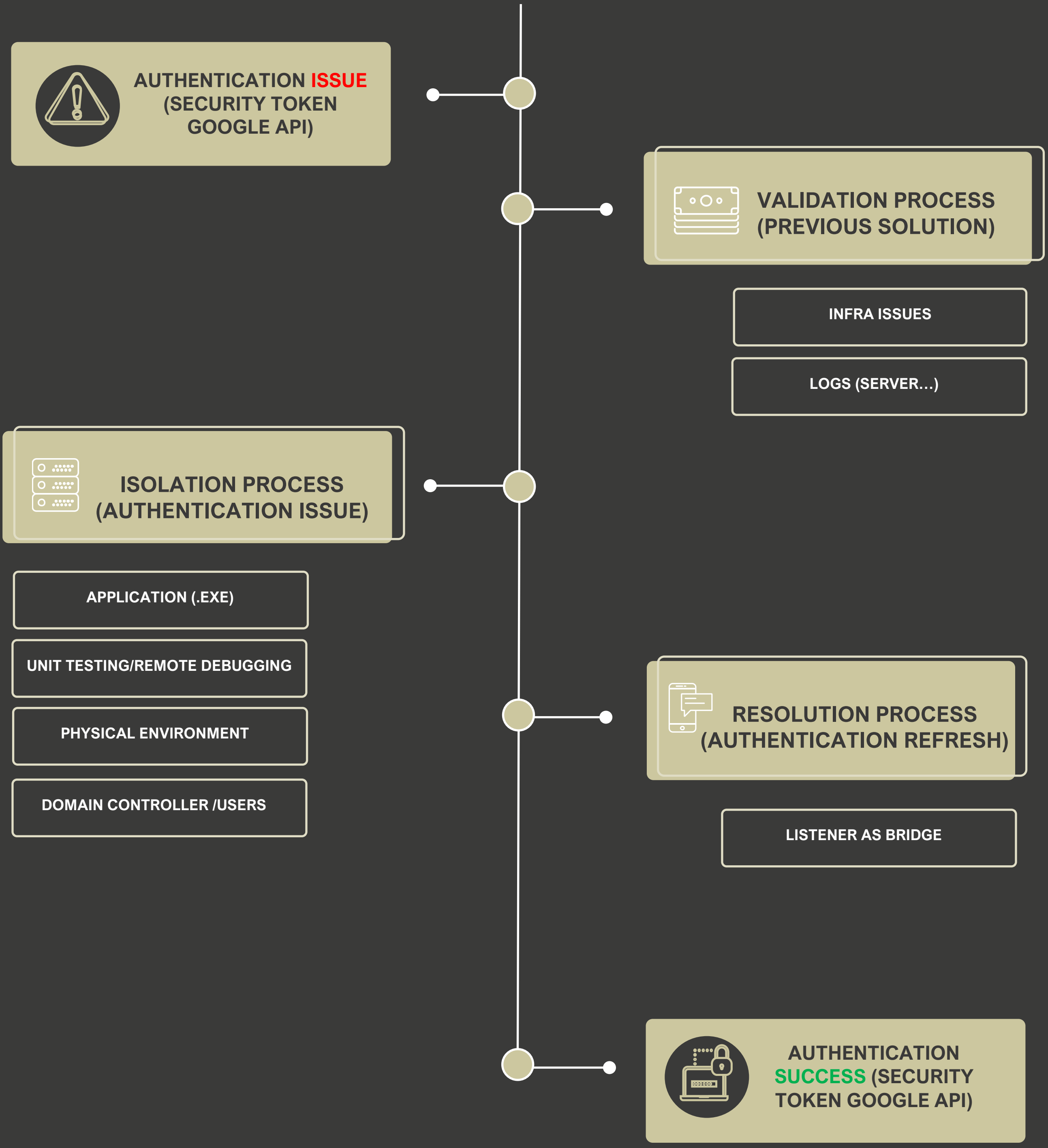
9

Then, the "Listener" deletes the message from the private queue and moves to the next message. If an error occurs in step 8, the message is transferred to a private queue created for that purpose.



CONCLUSION

Finally, the problem of refreshing the authentication token via the .NET library "**Google.API**" was solved in 3 linear steps: a process of validating preliminary solutions, a process of isolating components that play a role in refreshing the security token, and finally, the resolution process itself by creating a new "listener" as a "bridge" between CRM data and the Google Web Service directory.


AUTHENTICATION ISSUE (SECURITY TOKEN GOOGLE API)
VALIDATION PROCESS (PREVIOUS SOLUTION)
INFRA ISSUES
LOGS (SERVER...)
ISOLATION PROCESS (AUTHENTICATION ISSUE)
APPLICATION (.EXE)
UNIT TESTING/REMOTE DEBUGGING
PHYSICAL ENVIRONMENT
DOMAIN CONTROLLER /USERS
RESOLUTION PROCESS (AUTHENTICATION REFRESH)
LISTENER AS BRIDGE
AUTHENTICATION SUCCESS (SECURITY TOKEN GOOGLE API)

References

[1] The version of the tool is the same as the one presented in the Google site.

[2] https://technet.microsoft.com/en-us/library/cc773660%28v=ws.10%29.aspx

[3] This refers to an admin user who was placed in the administrator's group of the local server directly and who is not linked to a "DC" and therefore not to "Active Directory".

[4] https://www.google.com/apis/ads/publisher/ and https://www.google.com/apis/ads/publisher/v201408/CompanyService?wsdl

Legal Note

Glossary

Active Directory Domain Services (ADDS) - A directory service provides the methods for storing directory data and making this data available to network users and administrators.

Domain Controller - A server computer that responds to security authentication requests within a computer network domain. It is a server on a network that is responsible for allowing host access to domain resources.

Event Viewer - A component of Microsoft's Windows NT operating system that lets administrators and users view the event logs on a local or remote machine. Applications and operating-system components can use this centralized log service to report events that have taken place, such as a failure to start a component or to complete an action.

Google.Dfp.dll - This library provides you with functionalities to access the Google's Ad Manager API.

Google.API - Application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services.

MSMQ – Microsoft Message Queuing is a message queue implementation developed by Microsoft and deployed in its Windows Server operating systems since Windows NT 4 and Windows 95.

XML - Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

About Me and Philosophy

Thierry Sinassamy - I strongly advocate a methodogy of troubleshooting an issue no matter the technology we must face, even if I do not know the technology.

Contacts

DataObjectException Inc.
Thierry Sinassamy
1-514-883-6068
thierry.sinassamy@dataobjectexception.com
https://www.yuzuai.ca