APEXA

API End User Guide

Version 1.1

Purpose of the document

This process document is intended to walkthrough the APEXA API onboarding process

Version History

Version Number	Version Date	Author	Revised By	Description
1.0	09/05/2024	Clara Chijioke	Sahil Bhasin Chantel Munroe Sarah Sheppard	Initial version of document
1.1	01/04/2025	Clara Chijioke	Alysson Moulin	Added section on User Agent Header

Table of Contents

Introduction

The APEXA API is an outbound web service that allows our clients to pull advisor, shareholder, corporate and contract information from APEXA into their own back office. There are different inquiry methods (API call types), and each call type will return a specific output.

Types of API Calls

There are several API call types that can be made to APEXA.

Depending on the type of API call made, the output returned can either be:

- Generic – a high-level list of APEXA IDs with changes, within the specified timeframe or
- Specific – Full detailed results for specific advisors, shareholders, corporations or contracts.

ChangedProducerListing: Identify the Records with Changes

Typically, the first call most clients make is the "*ChangedProducerListing*" request. This is not a required call but is the easiest way to determine which contractors have had a change to their profiles or contracts within the specified timeframe.

To use this method, you must specify a Start Date/Time and an End Date/Time of **exactly 1 day** *(see sample request - Figure 1)*

This ensures that data is returned more efficiently and avoids the call timing out.

The output for this call is a list of APEXA IDs (advisor, shareholder, or corporation) for each producer record where changes occurred between the specified dates to either their profile or contracts. *(see sample response – Figure 2).*

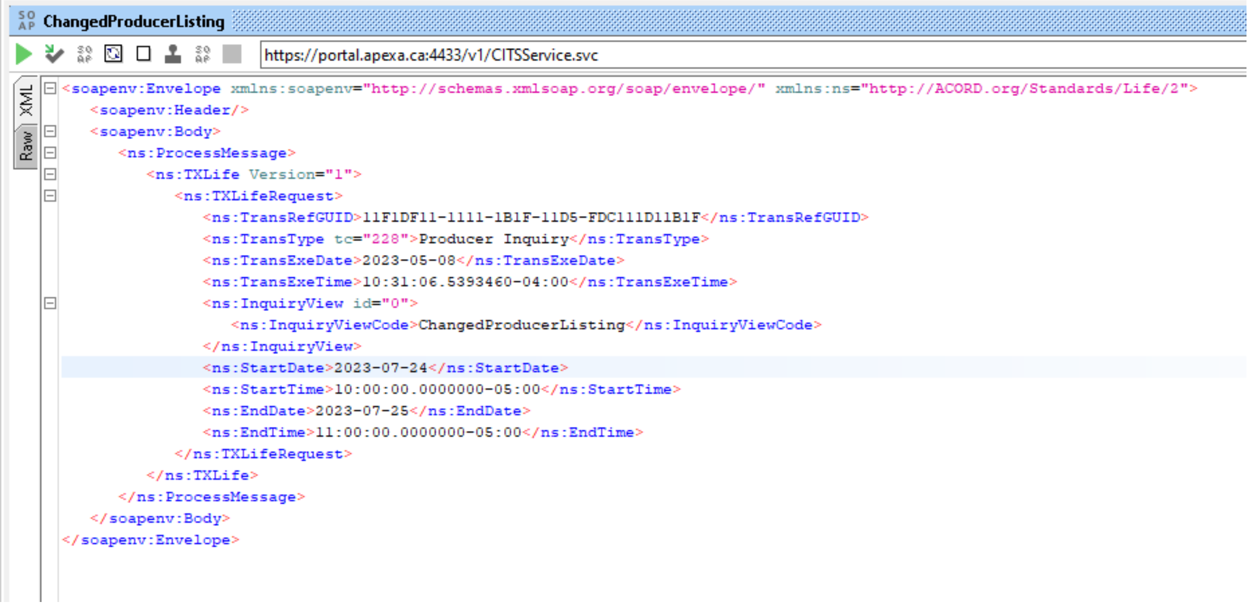
Note:
This is the only inquiry method that requires a date range to be included in the request. Providing a date range is not applicable for all other API inquiry methods.

The APEXA IDs returned by this first API call are used in a secondary API call to pull more detailed information.

Technical Documentation

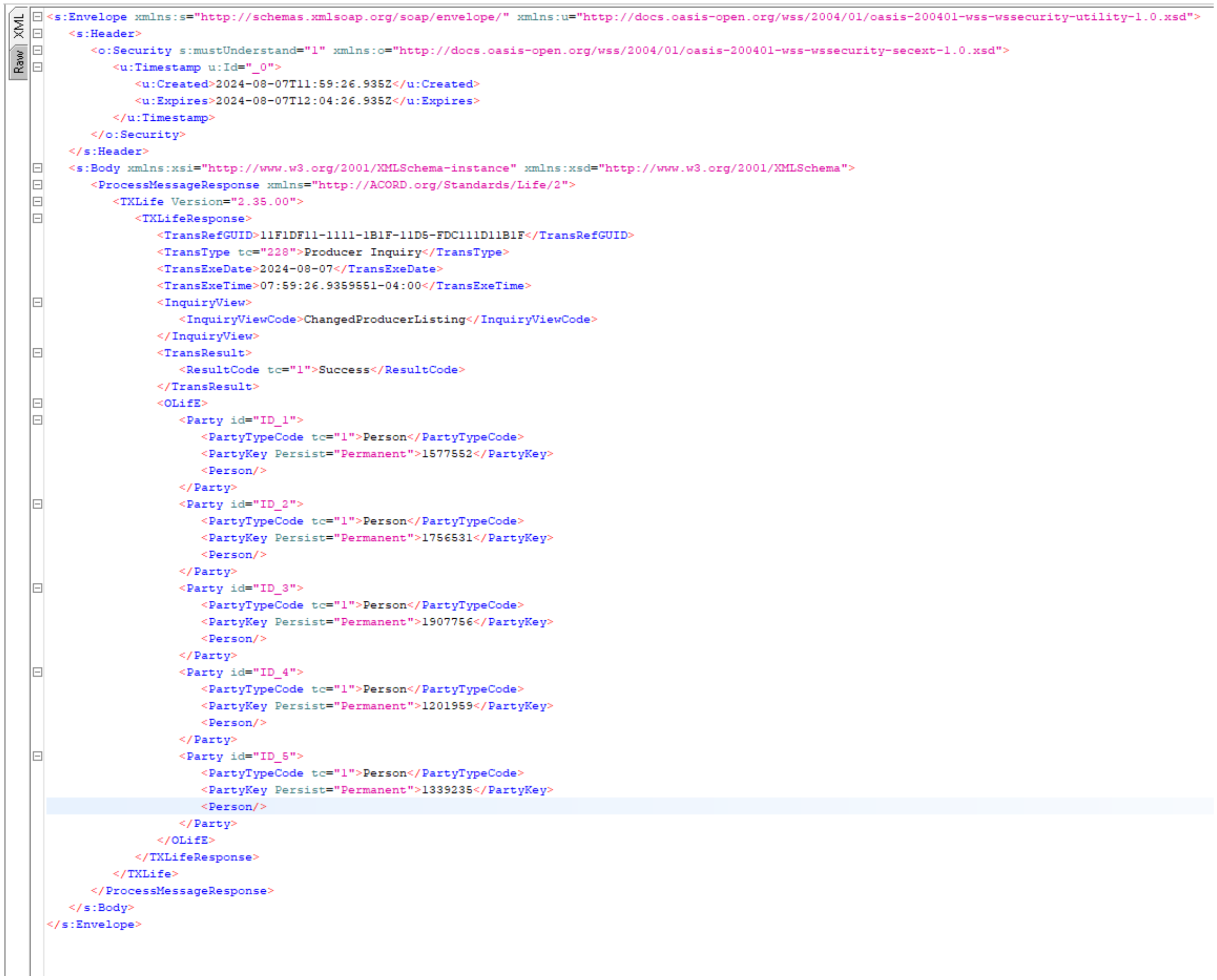
- Technical information related to the *ChangedProducerListing* API call can be found here:

Figure 1 ChangedProducerListing API Call Sample Request:



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http://ACORD.org/Standards/Life/2">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:ProcessMessage>
         <ns:TXLife Version="1">
            <ns:TXLifeRequest>
               <ns:TransRefGUID>11F1DF11-1111-1B1F-11D5-FDC111D11B1F</ns:TransRefGUID>
               <ns:TransType tc="228">Producer Inquiry</ns:TransType>
               <ns:TransExeDate>2023-05-08</ns:TransExeDate>
               <ns:TransExeTime>10:31:06.5393460-04:00</ns:TransExeTime>
               <ns:InquiryView id="0">
                  <ns:InquiryViewCode>ChangedProducerListing</ns:InquiryViewCode>
               </ns:InquiryView>
               <ns:StartDate>2023-07-24</ns:StartDate>
               <ns:StartTime>10:00:00.0000000-05:00</ns:StartTime>
               <ns:EndDate>2023-07-25</ns:EndDate>
               <ns:EndTime>11:00:00.0000000-05:00</ns:EndTime>
            </ns:TXLifeRequest>
         </ns:TXLife>
      </ns:ProcessMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 2 ChangedProducerListing API Call Sample Response:



```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2024-08-07T11:59:26.935Z</u:Created>
        <u:Expires>2024-08-07T12:04:26.935Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ProcessMessageResponse xmlns="http://ACORD.org/Standards/Life/2">
      <TXLife Version="2.35.00">
        <TXLifeResponse>
          <TransRefGUID>11F1DF11-1111-1B1F-11D5-FDC111D11B1F</TransRefGUID>
          <TransType tc="228">Producer Inquiry</TransType>
          <TransExeDate>2024-08-07</TransExeDate>
          <TransExeTime>07:59:26.9359551-04:00</TransExeTime>
          <InquiryView>
            <InquiryViewCode>ChangedProducerListing</InquiryViewCode>
          </InquiryView>
          <TransResult>
            <ResultCode tc="1">Success</ResultCode>
          </TransResult>
          <OLifE>
            <Party id="ID_1">
              <PartyTypeCode tc="1">Person</PartyTypeCode>
              <PartyKey Persist="Permanent">1577552</PartyKey>
              <Person/>
            </Party>
            <Party id="ID_2">
              <PartyTypeCode tc="1">Person</PartyTypeCode>
              <PartyKey Persist="Permanent">1756531</PartyKey>
              <Person/>
            </Party>
            <Party id="ID_3">
              <PartyTypeCode tc="1">Person</PartyTypeCode>
              <PartyKey Persist="Permanent">1907756</PartyKey>
              <Person/>
            </Party>
            <Party id="ID_4">
              <PartyTypeCode tc="1">Person</PartyTypeCode>
              <PartyKey Persist="Permanent">1201959</PartyKey>
              <Person/>
            </Party>
            <Party id="ID_5">
              <PartyTypeCode tc="1">Person</PartyTypeCode>
              <PartyKey Persist="Permanent">1339235</PartyKey>
              <Person/>
            </Party>
          </OLifE>
        </TXLifeResponse>
      </TXLife>
    </ProcessMessageResponse>
  </s:Body>
</s:Envelope>
```

Secondary Calls: Pulling Detailed Advisor, Shareholder, Corporation and Contract Information

There are two ways of pulling back detailed Advisor, Shareholder, Corporate and contract information:

- As a secondary call made after the "*ChangedProducerListing*" API request (most common)
- As an independent API call

Generally, the output of the *ChangedProducerListing* (list of APEXA IDs related to Advisors, Shareholders and Corporations who have had changes to their profile or contracts) is used to make secondary calls to receive more detailed information about the profiles or contracts that have had changes. However, if clients already have the Apexa ID and there are known updates, they can proceed to make the secondary API calls directly without first calling the *ChangedProducerListing*.

Depending on the type of CITS API call made, the two response types will be returned:

1. Detailed profile information and contract information
2. Detailed profile information only

For both response types returned, the API will provide all current information available on the profile at the time the call is made. There is no indicator that outlines the exact information (profile or contract details) that has changed.

As a secondary call made after the "ChangedProducerListing" API request

Once you have received the list of records that have changes (the response to the "Changed Producer Listing" request), you can take those IDs and pass them back to APEXA to obtain more detailed information.

As an independent API call

Clients can also proceed to make other API requests to pull back detailed Advisor, Shareholder, Corporate and contract information without having to make the "*ChangedProducerListing*" API request. This approach is used if clients already have the required parameter values (APEXA IDs).

Single or Multiple Records

In your second API call you can pass just a single APEXA ID, or you can pass a complete listing of APEXA IDs returned in the first call (Multiple Producers Request by ID).

The "Multiple Producers Request by ID" call encapsulates multiple calls or API inquiry methods for multiple contractor types (advisor, shareholder, or corporation) in the same request and receive the full record for each APEXA ID in return.

Most clients use the "Multiple Producers Request by ID" request as it allows the client to pass all the IDs, regardless of the type of contractor (advisor, shareholder, or corporation) at the same time.

The different CITS API calls that can be used are listed below:

- FullProducerWithAppointments
- ProducerWithAppointments
- ProducerAndRelated
- ProducerOnly

Note:

- Requests can be filtered to reduce volume of data.
  - For example: If you want to review only the profile information of a contractor (advisor, shareholder, or corporation) and contract and shareholder details are not required, it is recommended to make the *"ProducerOnly"* call, rather than making a "*FullProducerWithAppointments*" call that returns the profile information and includes contract and shareholder details in the response.

FullProducerWithAppointments:

Returns profile, shareholder & contract details for advisors & corporations depending on the APEXA ID passed.

- If an Advisor Apexa ID is passed, it returns the full Advisor profile, associated corporation(s), major shareholder details (shareholders with 20% ownership or more) and contract information.
- If a Corporation Apexa ID is passed, it returns the full Corporation profile, shareholder details, and contract information.

Technical Documentation

- To return information for a single record, clients can use the *FullProducerWithAppointment* call. Detailed information can be found here
- To return information for multiple records at one time, clients can use the *MultipleProducersByID* PI call. Detailed information can be found here
- Technical information on how to setup the *FullProducerWithAppointments* API call can be found here

ProducerWithAppointments:

Returns profile & contract details for advisors & corporations depending on the APEXA ID passed. **No shareholder information** is returned.

Technical Documentation

- Technical information on how to setup the *ProducerWithAppointments* API call can be found here

ProducerAndRelated:

Returns profile information for advisors and shareholder details for their associated corporations. **No contract information** is returned.

Technical Documentation

- Technical information on how to setup the *ProducerAndRelated* API call can be found here

ProducerOnly:

Returns profile details for advisors and corporations only depending on the APEXA ID passed. **No Shareholder or contract information is returned.**

Technical Documentation

- Technical information on how to setup the *ProducerOnly* API call can be found here

Non CITS Methods:

Non CITS API requests are made using a different format than the CITS API requests.

The CITS API is based on a specific standard called ACORD. Maintained by Cledis

The responses of the CITS API requests follow a very specific format, but the Non CITS requests are outside of this format, and do not have to conform to any specific standard.

Non CITS methods return APEXA IDs associated with an internal contractor ID, or a list of internal contractor IDs associated with APEXA IDs depending on the parameters provided.

Types of Non CITS API Methods:

- RequestContractorApexaIds
- RequestContractorInternalIds
- RquestContractorIdsBySellingcode

RequestContractorApexaIds (Internal ID):

Returns a list of APEXA IDs associated with the specified Internal ID. The internal ID is the unique identifier used by billable clients in their back-office systems to identify an advisor record.

Technical Documentation

- Technical information on how to setup the *RequestContractorApexaIds* API call can be found here

RequestContractorInternalIds (APEXA ID):

Returns a list of Internal IDs associated with the specified APEXA ID.

Technical Documentation

- Technical information on how to setup the *RequestContractorInternalIds* API call can be found here

RequestContractorApexaIdsBySellingCode (Selling Code):

Returns a list of APEXA IDs associated with the specified Selling Code.

Technical Documentation

- Technical information related to the *RequestContractorApexaIdsBySellingCode* API call can be found here

File Requests

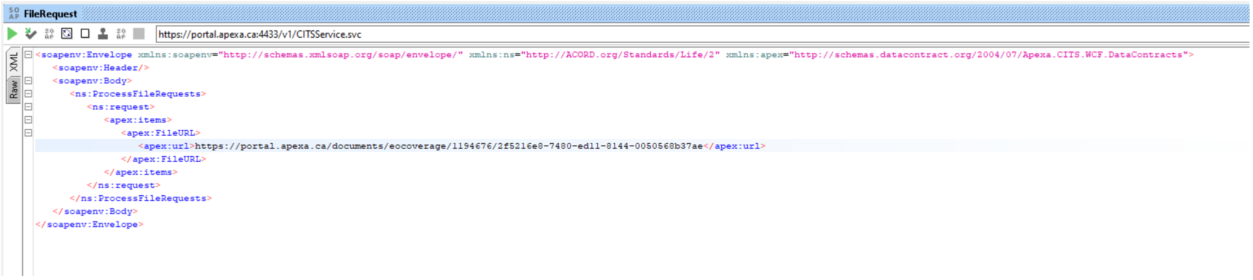
Files are attached to each response received as a hyperlink/URL in the response xml file. To download a pdf out of APEXA, you will need to use the “File Retrieval Request.

File request API calls are made with one or more file URLs received from the response of one of the API Inquiry Methods listed above.

The response will be a file name and byte array for each file returned.

This will return the pdf document *(see Sample Response below)*

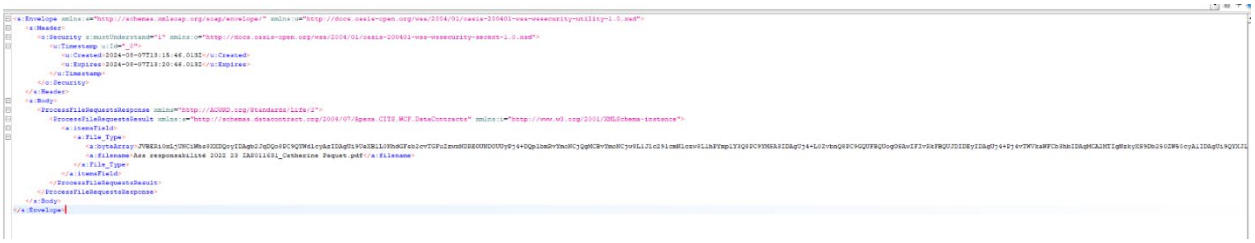
File Request API Call Sample Request



https://portal.apexa.ca:4433/v1/CITSService.svc

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http://ACORD.org/Standards/Life/2" xmlns:apex="http://schemas.datacontract.org/2004/07/Apexa.CITS.WCF.DataContracts">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:ProcessFileRequests>
         <ns:request>
            <apex:items>
               <apex:FileURL>
                  <apex:url>https://portal.apexa.ca/documents/eocoverage/1194676/2f5216e8-7480-ed11-8144-0050568b37ae</apex:url>
               </apex:FileURL>
            </apex:items>
         </ns:request>
      </ns:ProcessFileRequests>
   </soapenv:Body>
</soapenv:Envelope>
```

File Request API Call Sample Response



```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <s:Header>
      <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
         <u:Timestamp u:Id="_0">
            <u:Created>2024-08-07T13:15:46.013Z</u:Created>
            <u:Expires>2024-08-07T13:20:46.013Z</u:Expires>
         </u:Timestamp>
      </o:Security>
   </s:Header>
   <s:Body>
      <ProcessFileRequestsResponse xmlns="http://ACORD.org/Standards/Life/2">
         <ProcessFileRequestsResult xmlns:a="http://schemas.datacontract.org/2004/07/Apexa.CITS.WCF.DataContracts" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
            <a:itemsField>
               <a:File_Type>
                  <a:byteArray>JVBERi0xLjUNCiWhs8XXDQoyIDAgb2JqDQo8PC9QYWdlcyAzIDAgUi9UeXBlL0NhdGFsb2cvTGFuZzwmNDREOUNDOUUyPj4+DQplbmRvYmoNCjQgMCBvYmoNCjw8L1J1c291cmNlcyw8L1hPYmplY3Q8PC9YMSA3IDAgUj4+L0ZvbnQ8PC9GQUFBQUogOSAwIFIvRkFBQUJDIDEyIDAgUj4+Pj4vTWVkaWFCb3hbIDAgMCA2MTIgNzkyXS9Db250ZW50cyAxIDAgUi9QYXJ
                  <a:filename>Ass responsabilité 2022 23 IAS011631_Catherine Paquet.pdf</a:filename>
               </a:File_Type>
            </a:itemsField>
         </ProcessFileRequestsResult>
      </ProcessFileRequestsResponse>
   </s:Body>
</s:Envelope>
```

Applying Business Rules and Logic to Response

The responses to the various API calls will provide **everything** available for that advisor, corporation, or shareholder and contracts, depending on the call made. The response includes all details, not the changes

To recognize what has changed in a profile or on a contract, your organization will need to incorporate business logic in your back office. You may choose to store off the last response for each record and compare to determine if the change is material to your business.

In addition, your organization may only want to receive specific pieces of information for a profile or a contract (i.e. licence information) and so your organization may code your back-office system to only extract and load that key piece of information into your back office and ignore/omit the rest of the response.

Connecting to the API & Testing

API Credentials

The APEXA platform uses a special CITS feed account to allow your organization to access the API. The administrator of APEXA for your organization will need to create an account within the test (UAT) and Production environments.

To set up this account, you will need an accessible inbox. This address should be provided to your APEXA administrator. They will set up an account with the role type “CITS Feed” and an email will be sent to the inbox you have provided. Proceed with the set-up instructions – the username (email) and password will become the credentials you need to access the API in the respective environment.

As best practice we recommend creating a dedicated CITS user with this role.

New clients are to perform testing in UAT prior to setting up a CITS account in Production.

Creating a CITS API role:

- From the Profile Menu Tab click on >> Corporation >> Users >> Add
- Select a new user from your list of associated individuals or create a new employee.
- Fill in the required details and select CITS API as the " Role"
- Save
- If you add a new employee, they will be sent an invitation email to register. Then they can sign on using their email/password.

Note:

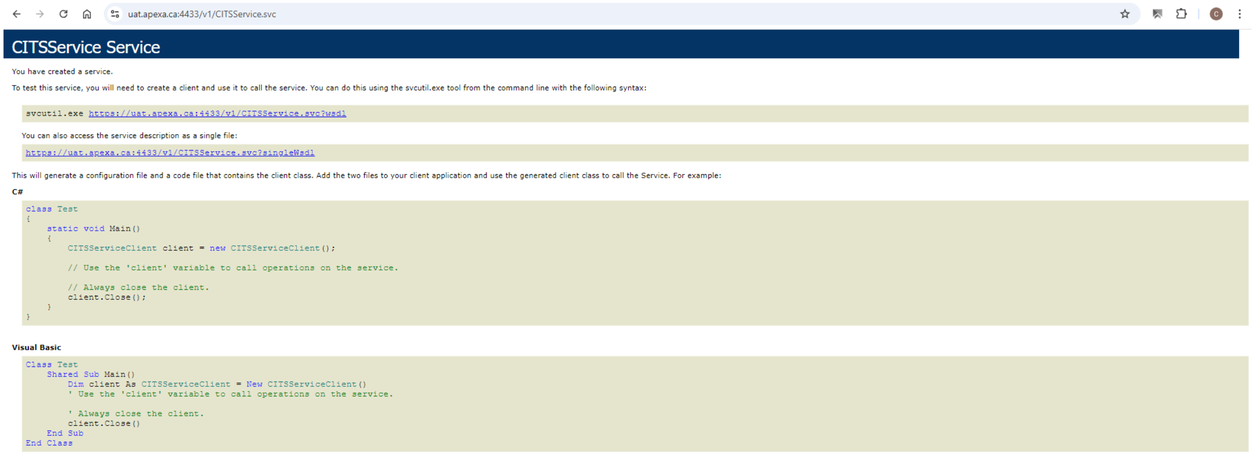
If you wish to use an existing employee as your CITS Administrator, you can edit the employee role and reassign them as a CITS Administrator.

API Portal Locations:

The UAT API is located at https://uportal.apexa.ca:4433/v1/CITSService.svc
The PRD API is located at https://portal.apexa.ca:4433/v1/CITSService.svc

Connecting to the API

Clients can click on the UAT & PRD portal locations above to confirm the CITS API service is online. The page below will be displayed if the CITS API service is available:



uat.apexa.ca:4433/v1/CITSService.svc

CITSService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe https://uat.apexa.ca:4433/v1/CITSService.svc?wsdl
```

You can also access the service description as a single file:

```
https://uat.apexa.ca:4433/v1/CITSService.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        CITSServiceClient client = new CITSServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As CITSServiceClient = New CITSServiceClient()
        ' Use the 'client' variable to call operations on the service.

        ' Always close the client.
        client.Close()
    End Sub
End Class
```

Once the availability of CITS is confirmed, clients can begin consuming the API from their back-end system.

Note: Your organization is responsible for building your backend system that consumes the API.

To get more information about the API, please refer to GitHub - ApexaCorp/cits-webservice. This project contains all the code you need to build your integration with APEXA's CITS API. The examples and code provided should be everything you need to jumpstart your development.

Using the API

API requests can either be made manually or automated.

Manual API Requests:

Clients can log into the API console at any time to make manual API requests using any of the API Inquiry Methods listed above.

Automated API Requests:

Clients can set up recurring rules to make API requests for any of the API Inquiry Methods listed above run at specific times.

The recommended occurrence for automated API calls is once daily, either early in the morning or after business hours.

Note:

To prevent clients from experiencing timeouts when making API requests, we have extended the **API call duration to 10 minutes** to allow users to receive a valid response for each call made.

We recommend mirroring the same call duration when setting up automated API calls on the client side.

Essential Information:

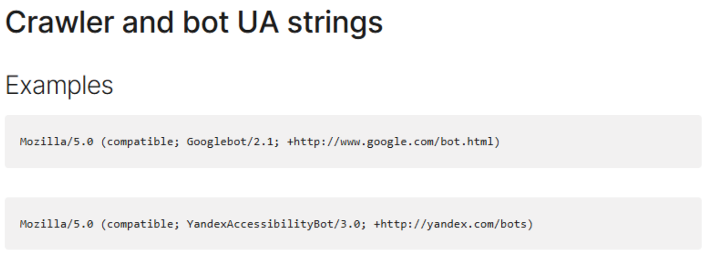
User-Agent Header:

To prevent clients from experiencing service failures when making API requests, it is **mandatory** to include the User-Agent header in your HTTP requests (API Calls).

The HTTP User-Agent request header is a characteristic string that lets servers and network peers identify the application, operating system, vendor, and/or version of the requesting user agent.

The User-Agent header is used for a variety of purposes, including:
Website optimization: Websites can use the User-Agent header to optimize content for different devices.
Analytics: Track and analyze what browsers, devices or operating systems are being used.
Security: Websites can use the User-Agent header to implement security features when a user logs in from a new device.
Content Adaptation: Websites can use the User-Agent header to adapt content to the device being used, such as a mobile phone, desktop, etc.
Debugging: Information from user-agent headers can be used to Trouble shoot issues related to specific browser types.

Example of a User-Agent Header:



Crawler and bot UA strings

Examples

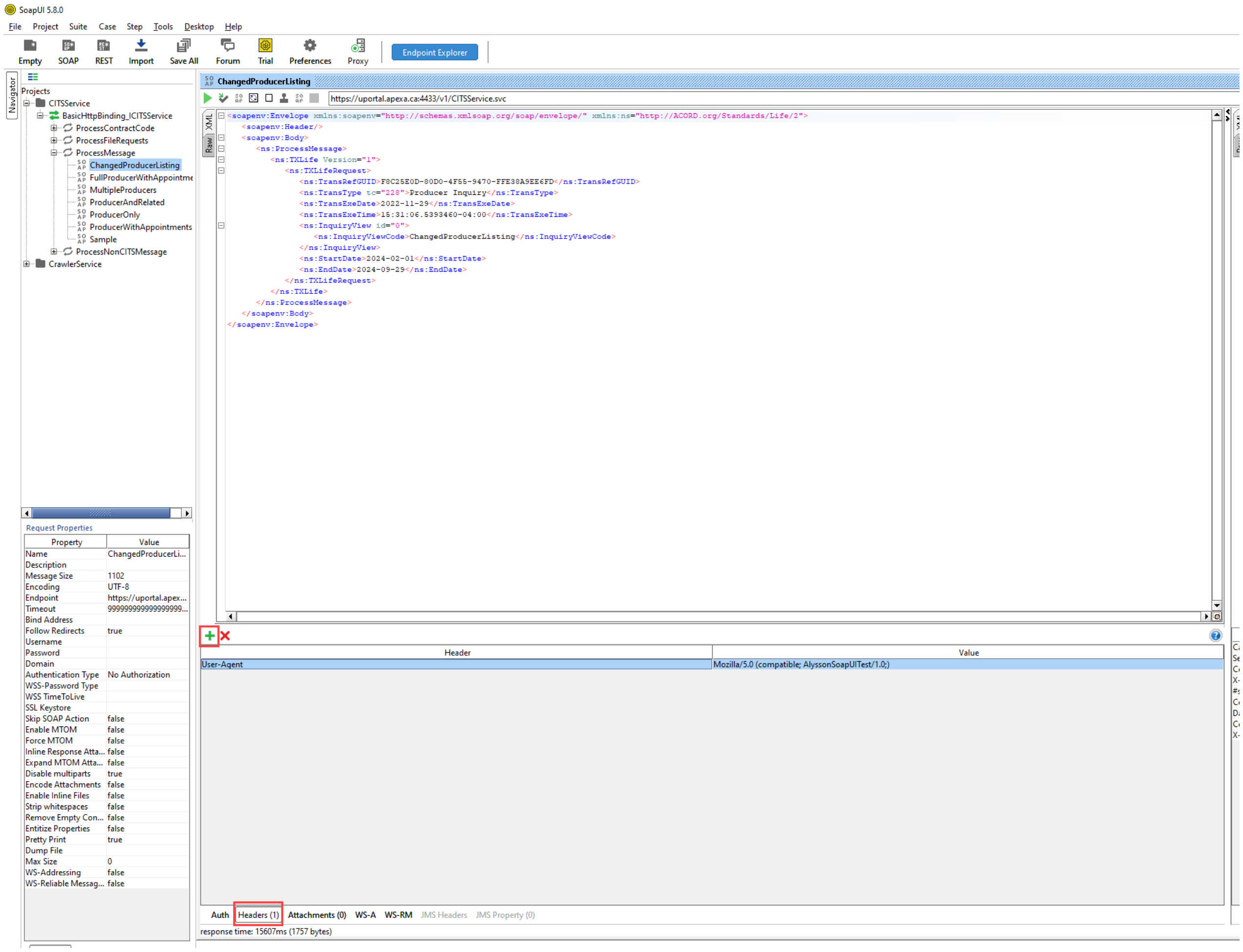
```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

```
Mozilla/5.0 (compatible; YandexAccessibilityBot/3.0; +http://yandex.com/bots)
```

Note: The example above is for system-to-system integration, clients are required to use unique User-Agent headers that identify their system.

How to Set Up a User-Agent Header

To set a user agent header, you can include an agent string in the HTTP header User-Agent. The minimum user agent string is the integration name and version. You can also include multiple values in a space-separated list.
Different technologies provide different ways to add the User-Agent header. Here is an example showing how this header can be added using SoapUI:



https://uportal.apexa.ca:4433/v1/CITSService.svc

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http://ACORD.org/Standards/Life/2">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:ProcessMessage>
         <ns:TXLife Version="1">
            <ns:TXLifeRequest>
               <ns:TransRefGUID>F8C25E0D-80D0-4F55-9470-FFE38A9EE6FD</ns:TransRefGUID>
               <ns:TransType tc="228">Producer Inquiry</ns:TransType>
               <ns:TransExeDate>2022-11-29</ns:TransExeDate>
               <ns:TransExeTime>15:31:06.5393460-04:00</ns:TransExeTime>
               <ns:InquiryView id="0">
                  <ns:InquiryViewCode>ChangedProducerListing</ns:InquiryViewCode>
               </ns:InquiryView>
               <ns:StartDate>2024-02-01</ns:StartDate>
               <ns:EndDate>2024-09-29</ns:EndDate>
            </ns:TXLifeRequest>
         </ns:TXLife>
      </ns:ProcessMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

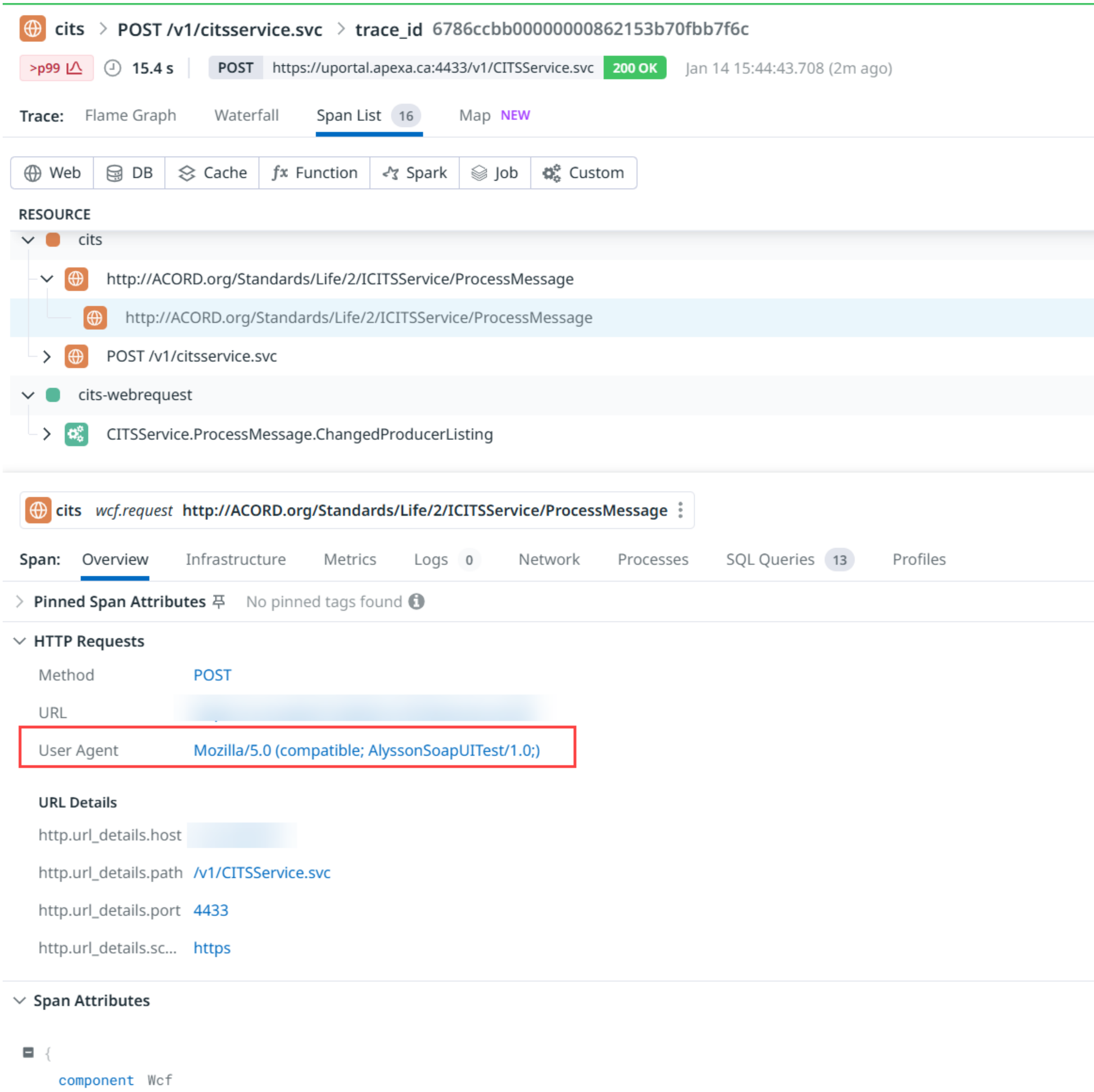
Request Properties

Property	Value
Name	ChangedProducerLi...
Description	
Message Size	1102
Encoding	UTF-8
Endpoint	https://uportal.apex...
Timeout	9999999999999999...
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization
WSS-Password Type	
WSS TimeToLive	
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false
Force MTOM	false
Inline Response Atta...	false
Expand MTOM Atta...	false
Disable multiparts	true
Encode Attachments	false
Enable Inline Files	false
Strip whitespaces	false
Remove Empty Con...	false
Entitize Properties	false
Pretty Print	true
Dump File	
Max Size	0
WS-Addressing	false
WS-Reliable Messag...	false

Header	Value
User-Agent	Mozilla/5.0 (compatible; AlyssonSoapUITest/1.0;)

response time: 15607ms (1757 bytes)

And here is how the information is captured and logged by Apexa:



cits > POST /v1/citsservice.svc > trace_id 6786ccbb00000000862153b70fbb7f6c

>p99 15.4 s POST https://uportal.apexa.ca:4433/v1/CITSService.svc 200 OK Jan 14 15:44:43.708 (2m ago)

Trace: Flame Graph Waterfall Span List 16 Map NEW

Web DB Cache Function Spark Job Custom

RESOURCE

- cits
  - http://ACORD.org/Standards/Life/2/ICITSService/ProcessMessage
    - http://ACORD.org/Standards/Life/2/ICITSService/ProcessMessage
  - POST /v1/citsservice.svc
- cits-webrequest
  - CITSService.ProcessMessage.ChangedProducerListing

cits *wcf.request* **http://ACORD.org/Standards/Life/2/ICITSService/ProcessMessage**

Span: Overview Infrastructure Metrics Logs 0 Network Processes SQL Queries 13 Profiles

Pinned Span Attributes No pinned tags found

HTTP Requests

Method	POST
URL	[redacted]
User Agent	Mozilla/5.0 (compatible; AlyssonSoapUITest/1.0;)

URL Details

http.url_details.host	[redacted]
http.url_details.path	/v1/CITSService.svc
http.url_details.port	4433
http.url_details.sc...	https

Span Attributes

```
{
    component Wcf
```

Note:

If a client inputs a User-Agent value that is on the blacklist of values known to be used by bots, the API call will be blocked.

Example of a crawler and bot User-Agent value:

```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

```
Mozilla/5.0 (compatible; YandexAccessibilityBot/3.0; +http://yandex.com/bots)
```

For more information on what a user-agent header is click here.

Any issues experienced when using the API should be escalated to APEXA through the APEXA HelpDesk Service.

Appendix A– Request Types

Request	Record Type	Link to API Call Sample
Changed Producer Listing	Returns Advisor & Corporation records with profile or contract changes for specified timeframe	link to API Call
Full Producer with Appointments	Returns all Advisor, Shareholder & Corporation records with Contracts (in any state)	link to API Call
Producer with Appointments	Returns all Advisor & Corporation records with Contracts (in any state)	link to API Call
Multiple Producers	Returns all (Advisors, Shareholders and Corporations)	link to API Call
Producer and Related	Returns all Advisors that are also shareholders and their organizations	link to API Call
Producer Only	Returns Advisor records only	link to API Call

Appendix B -Specification Documentation

This document outlines all the information that can be returned in secondary API calls.