Rowan Westenberg

ECS International  de factorij 24c, zwaag, Netherlands

June of 2022

Clarify & Afas api intergration

Contents

[Introduction 2](#_Toc106631107)

[DISCLAIMER 2](#_Toc106631108)

[Project Setup 2](#_Toc106631109)

[Potential Updates 5](#_Toc106631110)

[Sources 5](#_Toc106631111)

[Appendix 6](#_Toc106631112)

# Introduction

This document Will go over the implementation of the Afas Api into Cleo clarify Studio. For this document it is a prerequisite That you have basic knowledge of Cleo clarify Studio in addition of API’s and that you already know your API Token. In this documentation you will find certain text in curly brackets these are placeholders for the corresponding text applying to you for example {Company name} it will correspond to the company you work for and is using this implementation. As for the double quotes they are representing the literal Naming. This document uses Image references from “Appendix\_AfasApi\_Clarify\_intergration.docx” also mentioned later in this document under “Appendix”

DISCLAIMER

this document is completed as completely as possible to the knowledge of the documenting party at the time of documenting. The Software versions also might come into play and thus might certain functionalities be deprecated in future versions of clarify and the API from Afas. This documents serves as a base/guide line and not an endpoint of how to do the integration.

# Project Setup

For this part you need Source “1”

1. Create your project and give it an Explanatory/logical name
   1. Example: x4c.{CompanyName}.core
2. Add a Global variable to your project and give it an Explanatory/logical name. For reference see “ figure 1 global variables”
   1. Example:{ApiGV}
   2. Add the variable {Company Name}ApiBaseUrl with value
   3. Add the variable {company Name}ApiToken with the value that you get from AFAS and only put in the combination of letters & numbers and not the tags
      1. Yes: {AA00BB11CC22}
      2. No: <token><version>1</version><data>{AA00BB11CC22} </data></token>
   4. Add variables for the endpoint you want to use a handy tool you can use is source “1”
      1. Select the option that reads “Token Authentication”
      2. For the token in you got from AFAS That looks similar to the one below token. The token in the middle will most likely be different
         1. <token><version>1</version><data>{AA00BB11CC22}</data></token>
      3. Click the button that reads ”Login”
      4. Then you can scroll to your desired action and copy the link and remove the baseUrl
         1. What you get [https://89661.resttest.afas.online/profitrestservices/connectors/{your](https://89661.resttest.afas.online/profitrestservices/connectors/%7byour) endpoint}
         2. You should use /profitrestservices/connectors/{your Endpoint}
            1. Give the variable an explanatory name
      5. Example: AfasEndpoint{YourEndpoint}
      6. Value: /profitrestservices/connectors/{yourEndpoint}
3. Add the dependencies to your project for reference see “Figure 2 Dependencies”
   1. Add “x4c.connector.afas”
   2. Add “x4c.core”
   3. Add “x4c.settings”
4. Business process For reference see “Figure 3 Business process”
   1. Add a business process without any templates And give it an explanatory name
      1. Example: {CompanyName}{ApiUsage}
   2. In this you at least add the below standing tasks for a Request from the API
      1. Resolve global variable for the API token
         1. Create a variable in your business process for the token
            1. Example { V\_ApiToken}
         2. In properties Assign { V\_ApiToken} as the result
         3. In “properties” Tab give “string” the value of you Global variable in pointy brackets as a “string literal”
            1. Example: < x4c.{CompanyName}.core.{ ApiGV}.{Company Name}ApiToken>
      2. Resolve global variable for baseUrl
         1. Create a variable in your business process for the baseUrl
            1. Example { V\_ baseUrl }
         2. In properties Assign { V\_ baseUrl } as the result
         3. In “properties” Tab give “string” the value of you Global variable in pointy brackets as a “string literal”
            1. Example: < x4c.{TheNumberStore}.core.{ ApiGV}.   
               {Company Name}ApiBaseUrl >
      3. Resolve global variable for The service name
         1. Create a variable in your business process for the ServiceName
            1. Example c
         2. In properties Assign { V\_ ServiceName } as the result
         3. In “properties” Tab give “string” the value of you Global variable in pointy brackets as a “string literal”
            1. Example: < x4c.{TheNumberStore}.core.{ApiGV}. {AfasEndpointYourEndpoint}>
      4. String append
         1. “Result” as {V\_ServiceName}
         2. “String” as “stringliteral” with value { /profitrestservices}
         3. “Append” as {V\_ServiceName}
      5. Call on Afas afas Business
         1. Option 1: {x4c.connector.afas.bps.GetAfasGenericBPS}
            1. “p\_AfasApiServiceName” assigned with {V\_ ServiceName }
            2. “p\_ResponseBody” assigned with {V\_ResponseBody

Create the variable with an explanatory name

Example:{v\_ResponseBody}

* + - * 1. “p\_AfasApiToken” assigned with {V\_ApiToken}
        2. “p\_AfasApiBaseUrl” assigned with {V\_ baseUrl}
      1. Option 2: { x4c.connector.afas.bps.PostAfasGenericBPS}
         1. “p\_AfasApiServiceName” assigned with {V\_ ServiceName }
         2. “p\_ResponseBody” assigned with {V\_ResponseBody

Create the variable with an explanatory name

Example:{v\_ResponseBody}

* + - * 1. “p\_AfasApiToken” assigned with {V\_ApiToken}
        2. “p\_AfasApiBaseUrl” assigned with {V\_ baseUrl}
        3. “p\_RequestBody” assigned with {V\_RequestBody}

Create the variable with an explanatory name

Example: {V\_RequestBody}

* + - 1. Option 3(In Dev): {x4c.connector.afas.bps.DeleteAfasGenericBPS}
      2. Option 4(In Dev): {x4c.connector.afas.bps.PutAfasGenericBPS}
  1. INFO: The {V\_RequestBody} can be the output from a “ruleset” But the “Target” has to be JSON for reference see “figure Post Endpoint Fields”
     1. When you create this “ruleset” give it an explanatory/logical name
        1. Example: {source}To{your Endpoint name}
     2. You can make JSON schemas template with source “1”
        1. you scroll down to the get section Select the “Update connector” that you need.
        2. Click on the button that reads “POST”
        3. If possible Check ”Autonummering” and don’t fill in “VolgNummer”
           1. This is for auto incrementing Record Id’s
        4. Then fill in the rest of the fields as requested by Afas
        5. Click on the button that reads “Execute”
           1. If this is unsuccessful Look at the error code and try to resolve these
        6. If successful you will get a popup like “Figure 5 Post Successful JSON”
        7. You copy the “request body”
        8. You create a new “JSONschema” in ”x4c.connector.afas” under “schemas” with an explanatory/logical name
           1. Example: {YourEndpointName}Schema
           2. When choosing a method you choose Paste JSON
           3. And paste The previous copied ”request body”
           4. And give The schema an Explanatory/logical name
           5. If you want to send multiple at once you have to add Brackets Around fields for reference see “figure 6 sending multiple”
     3. Add {YourEndpoint}Schema to the {source}To{YourEndpointName} ”ruleset”
     4. As a “source” in this “ruleset” You can use anything as long as you do the mapping correctly.
     5. Add this “ruleset” to {CompanyName}{ApiUsage} before the call to   
        {x4c.connector.afas.bps.PostAfasGenericBPS}
     6. For the parameters of the “ruleset”
        1. “sourceNode” assigned with {V\_ResponseBody}
        2. “TargetNode” assigned with {V\_RequestBody}
        3. “Settings” TransformationSettings Literal {x4c.core.UndifinedTS}
        4. “UpdateNode” leave empty
        5. “SourceContextNode” Assigned with {V\_SourceContext
           1. Create the variable as a type off “storagenode”
        6. “targetContextNode” leave empty

1. Using the API
   1. “Bundle”/” deploy” x4c.{CompanyName}.core and if you added schemas {x4c.connector.afas} in the “admin console” under “Projects”
   2. Then run {CompanyName}{ApiUsage} in the “Admin console” Under “business Process”
   3. You can see the results in “Admin Console” under “Auditor”
      1. Look for the latest version that you tried with the same name as {CompanyName}{ApiUsage}
2. Extras
   1. You can automate this by using schedulers
   2. If you want to try if your end points work with the intended request body  
      You can use a tool called “Postman”
   3. If the posts works you can use Source “1” and scroll down to the “get” part with the end point you are trying to post to and see the inserted data there
   4. In {PostAfasWSC} In the “Default Request HTTP Headers” there is a static header With the name “Autorization” and with the value AfasToke{encodedApiKey}
      1. You should change this to the one corresponding to your ap
      2. This you can find with a task ”loginfo” in {x4c.connector.afas.bps.PostAfasGenericBPS} or {x4c.connector.afas.bps.GetAfasGenericBPS} After the “task” ”Encode storage Node” it default status is disabled

# Potential Updates

* Make it so the Authorization header can be dynamically changed by using the value’s created in the Post business process
* Finishing the functionality of deleting records dynamically
* Finishing the functionality of Putting/update records dynamically

# Sources

1. <https://89661.resttest.afas.online/profitrestservices>

# Appendix

Appendix\_AfasApi\_Clarify\_intergration