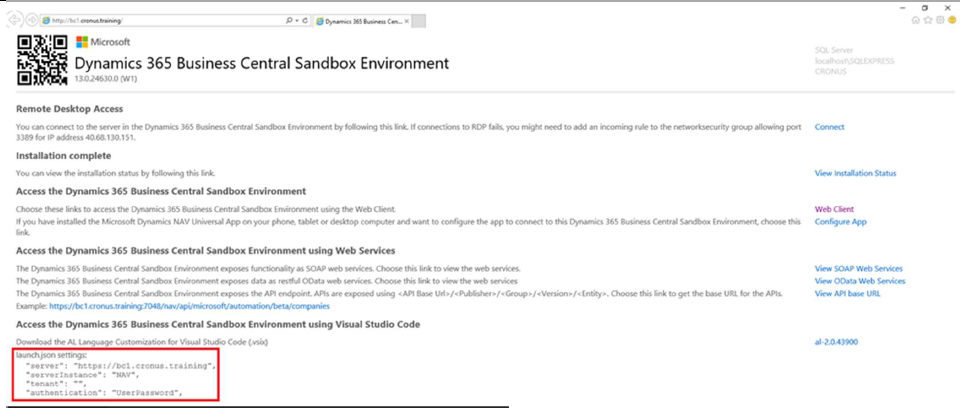
**Data exchange with AL language:**

1. **Creating new App:**

* Start VS Code
* Press “F1” (“Command Palette”)
* Enter “AL: Go!”
* Enter path (e.g. “…\Documents\Task3”)
* Select the latest target platform (4.0)
* In the dropdown list, pick “Your own server”
* When it asks you for credentials, don’t log in, just press ESC
* Set up the “launch.json”
* Make sure the settings are the same as the ones from your own landing page
* You can find those settings when you click “landing page” on the desktop of the VM.





Set up the “app.json”:

* Change “name” to “Task <Number of Task>”
* Change “publisher” to “<<your Name and Surname>>”
* Add feature “TranslationFile”



Download the Symbols:

* Use the Command Palette (F1) to search for “AL: Download symbols”

If necessary:

* Provide username and password from boarding pass



Delete “Hello world.al” file.

**Task 3.**

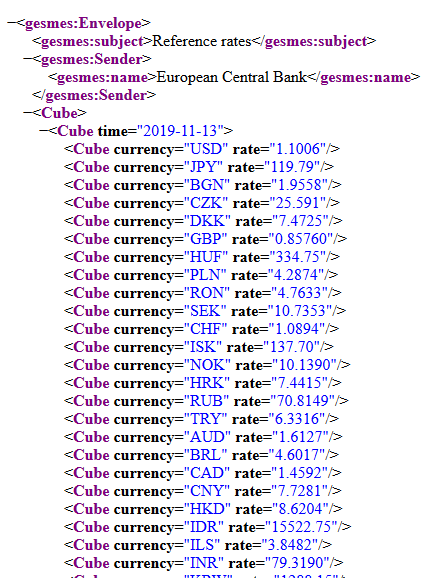
Importing European bank currency rates from XML

**Sources:**

1. ECB API: <http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml>
2. GitHub Repository with source code: github.com/abaludin/ImportCurrencyER

**Step 1**

Run API link at browser:



This response contains currencies rates to Euro for current date.

Open Currencies page in Business Central and create some new currency cards with codes like CAD, BGN, CZK, ISK, RUB, SGD

We have a situation when European company has customers/vendors with such currencies and needs to import actual rates for them.

**Step 2**

Create new management codeunit

* Create ImportCurrencyRates function with universal code to get API response (See Task 1 and Task2):

## Url := 'http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml';

## if not Client.Get(Url, Response) then

## Error(Text001\_Err, 'GET');

## Response.Content().ReadAs(ResponseText);

## if not Response.IsSuccessStatusCode() then

## Error(Text002\_Err, Response.HttpStatusCode(), ResponseText);

## 

**Step 4**

Create ParseXML function accepting ResponseText as a parameter.

First of all – remove all namespaces from xml to make parsing easier. Use “XML DOM Management” codeunit:

## ResponseText := XmlDom.RemoveNamespaces(ResponseText);

As you could see in response, all values stored as attributes, so we need to create one more function to extract Node Attribute value:

## local procedure GetAttributeValue(Node: XmlNode; AttributeName: Text): Text

## var

## Attribute: XmlAttribute;

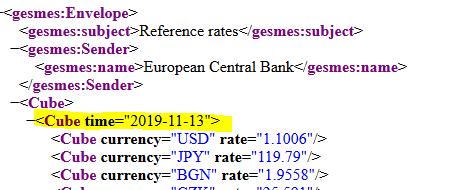
## begin

## If Node.AsXmlElement().Attributes().Get(AttributeName, Attribute) then

## exit(Attribute.Value());

## end;

Let’s try to extract rate date from “Cube time”. We need Evaluate(Date, XmlDate, 9) – to convert Xml Date format to normal date.



## XmlDocument.ReadFrom(ResponseText, XmlDoc);

## XmlDoc.SelectSingleNode('Envelope/Cube/Cube', Node);

## Evaluate(XmlDate, GetAttributeValue(Node, 'time'), 9);

Where

## XmlDoc: XmlDocument;

## Node: XmlNode;

## XmlDate: Date;

After that – we can extract currency nodes one by one:

## XmlDoc.SelectNodes('Envelope/Cube/Cube/Cube', XmlNodesList);

## foreach Node in XmlNodesList do

## If Currency.Get(GetAttributeValue(Node, 'currency')) then begin

## RateVariant := Rate;

## TypeHelper.Evaluate(RateVariant, GetAttributeValue(Node, 'rate'), '', '');

## InsertCurrencyRate(XmlDate, Currency.Code, RateVariant);

## end;

where

## TypeHelper: Codeunit "Type Helper";

## XmlNodesList: XmlNodeList;

## RateVariant: Variant;

## Rate: Decimal;

We use TypeHelper.Evaluate because of different localization standards for numbers – it could be incorrect numbers depends on , . and spaces.

**Step 5**

* Create last function “InsertCurrencyRate” for rates creation:

## local procedure InsertCurrencyRate(RateDate: date; codCurrency: code[10]; Rate: Decimal);

## var

## Currency: Record Currency;

## CurrencyExchangeRate: Record "Currency Exchange Rate";

## begin

## IF Currency.GET(codCurrency) then begin

## CurrencyExchangeRate."Currency Code" := codCurrency;

## CurrencyExchangeRate."Starting Date" := RateDate;

## CurrencyExchangeRate."Exchange Rate Amount" := 1;

## CurrencyExchangeRate."Relational Exch. Rate Amount" := Rate;

## CurrencyExchangeRate."Adjustment Exch. Rate Amount" := 1;

## CurrencyExchangeRate."Relational Adjmt Exch Rate Amt" := Rate;

## CurrencyExchangeRate."Fix Exchange Rate Amount" := CurrencyExchangeRate."Fix Exchange Rate Amount"::"Currency";

## IF NOT CurrencyExchangeRate.Insert() then

## CurrencyExchangeRate.Modify();

## end;

## end;

and place ImportCurrencyRates function to OnRun trigger of codeunit:

## trigger OnRun();

## begin

## ImportCurrencyRates();

## end;

Full codeunit code:

<https://github.com/ABaludin/ImportCurrencyER/blob/master/src/codeunit/Cod50135.AWR_ImportcurrencyER.al>

**Step 6**

Create Page extension for Currencies page to run Import codeunit:

## pageextension 50135 "AWR\_Currencies" extends "Currencies"

## {

## actions

## {

## addafter("Change Payment &Tolerance")

## {

## action("AWR\_ImportCurrencyRates")

## {

## ApplicationArea = All;

## Image = UpdateXML;

## Promoted = true;

## PromotedCategory = Process;

## Caption = 'Downloading exchange rates from the ECB';

## RunObject = codeunit "AWR\_Import currency ER";

## }

## }

## }

## }

Page extension:

<https://github.com/ABaludin/ImportCurrencyER/blob/master/src/pageextension/Pag-Ext50135.AWR_Currencies.al>

**Final Result:**

