NLSQL 1.01 cloud solution for SAP
Installation/Configuration Guide

1. Before the test you should request test **API Key** by email denis@nlsql.com. Setup simple code script inside company IT ecosystem. Script are different based on your existing IT environment
    a. For Python environment

```
import requests
url = 'https://api.nlsql.com/sap'
headers = {'Authorization': 'Token API KEY', 'Content-Type': 'application/json'}
payload = {'message': 'What is payments of customer USCU_S01 in 2016 jan-sep'}
r = requests.post(url, headers=headers, json=payload)
print(r.json())
```

    b. For JavaScript environment

XHR

```
var xhr = new XMLHttpRequest();
xhr.addEventListener('load', function(e) {
  var response = e.target.responseText;
  console.log(response);
});
xhr.addEventListener('error', function(e) {
  console.error('Request errored with status', e.target.status);
});
xhr.open('POST', 'https://api.nlsql.com/sap');
xhr.setRequestHeader('Content-Type','application/json');
xhr.setRequestHeader('Authorization','Token API KEY');
var body = '';
body += '{\n';
body += '  "message": "Show me stock of MZ-TG-Y120 on warehouse 171A,  1710"\n';
body += '}\n';
xhr.send(body);
```

Fetch

```
const headers = new Headers();
headers.append('Content-Type', 'application/json');
headers.append('Authorization', 'Token API KEY');

const body = `{
  "message": "Show me stock of MZ-TG-Y120 on warehouse 171A,  1710"
}`;

const init = {
    method: 'POST',
    headers,
    body
};
```

```
fetch('https://api.nlsql.com/sap', init)
.then((response) => {
   return response.text(); // or .json() or .blob() ...
})
.then((text) => {
   console.log(text)
})
.catch((e) => {
   console.log(e)
});
```

c. For C# environment

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using RestSharp;
using RestSharp.Authenticators;

namespace ConsoleApp1
{
   class Program
   {
      static void Main(string[] args)
      {
         var client = new RestClient("https://api.nlsql.com/sap");
         var request = new RestRequest(Method.POST);
         request.AddHeader("Host", "api.nlsql.com");
         request.AddHeader("Content-Type", "application/json");
         request.AddHeader("Authorization", "Token API KEY");
         request.AddParameter("undefined", "{\r\n  \"message\": \"Show me stock of MZ-TG-Y120 on
warehouse 171A,  1710\"\r\n}", ParameterType.RequestBody);
         IRestResponse response = client.Execute(request);
         Console.WriteLine(response.Content);
         Console.ReadKey();
      }
   }
}
```

d. JAVA environment

```
URL url = new URL("https://api.nlsql.com/sap");
HttpsURLConnection con = (HttpsURLConnection) url.openConnection();
con.setRequestMethod("POST");
```

```
con.setRequestProperty("Content-Type", "application/json");
con.setRequestProperty("Authorization", "Token API KEY");

/* Payload support */
con.setDoOutput(true);
DataOutputStream out = new DataOutputStream(con.getOutputStream());
out.writeBytes("{\n");
out.writeBytes("  \"message\": \"Show me stock of MZ-TG-Y120 on warehouse 171A,  1710\"\n");
out.writeBytes("}\n");
out.flush();
out.close();

int status = con.getResponseCode();
BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer content = new StringBuffer();
while((inputLine = in.readLine()) != null) {
   content.append(inputLine);
}
in.close();
con.disconnect();
System.out.println("Response status: " + status);
System.out.println(content.toString());
```

e. For PHP environment

```
<?php

$request = new HttpRequest();
$request->setUrl('https://api.nlsql.com/sap');
$request->setMethod(HTTP_METH_POST);

$request->setHeaders(array(
  'Host' => 'api.nlsql.com',
  'Content-Type' => 'application/json',
  'Authorization' => 'Token API KEY'
));

$request->setBody('{
  "message": "Show me stock of MZ-TG-Y120 on warehouse 171A,  1710"
}');

try {
  $response = $request->send();

  echo $response->getBody();
} catch (HttpException $ex) {
  echo $ex;
```

}

2. Once one from abovementioned code is running successfully, that's mean you can send and receive requests. You can make a few tests and start customizing end user interface.
   End Users can send the requests with Natural Language using User Interface. Once end user provide Natural Language request, you should provide external API https://api.nlsql.com/sap with correspondant request and get the response. Based on response you are able to execute SQL and reply to end user inside your IT infrastructure.

3. API reply message types ("data_type"). NLSQL external API server has multiple types of message types:

   a. **Graph**,
   which means end user requested graphic interpretation of some information.
   In that case you should execute SQL("sql") you received from NLSQL API, get the data from your SAP database and based on received data provide user graph. There are multiple packages, which allows you to build the graphs. For more details you can refer to info@nlsql.com
   Natural Language request example: **Graph me payments for customer USCU_S01 in 2016?**

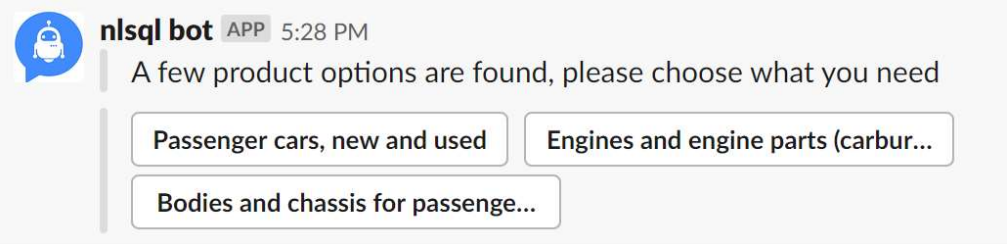
   b. Bubble,
   which means end user requested bubble graph interpretation of some information.
   In that case you should execute SQL("sql") you received from NLSQL API, get the data from your SAP database and based on received data provide user bubble graph. There are multiple packages, which allows you to build bubble graphs. For more details you can refer to info@nlsql.com
   Natural Language request example: **Show me bubble payments for customers USCU_S01, USCU_L10**

   c. **Message**,
   which means end user requested text reply for request in Natural Language. In that case you should execute SQL ("sql") you received from NLSQL API combine it with text message received ("message") from NLSQL API as well and reply to the end user inside your IT infrastructure.
   Natural Language request example: **Show me balance of customer USCU_L10**

   d. **YTD**,
   which means end user requested percentage comparison between 2 periods this year to date number and last year to date number. In that case you should take and execute 2 "sql1" and "sql2" scripts you received from NLSQL API and combine it with text message received ("message") from NLSQL API as well and reply to the end user inside your IT infrastructure. For better end user expierence it is recommended to use the following

formula YTD growth = (sql1/sql2-1)*100%
Natural Language request example: **YTD growth payments for customer USCU_S01**

e. **Buttons**,
which means the NLSQL systems detects a few similar options as per end user natural language request. In that case NLSQL system requires to ask end user additional question based on previous question. Instead of SQL in API response, system reply with dictionary {key1, value1; key2, value2, … ; keyN, valueN}, where key* is Button name and value* is post back response NLSQL need in order to proceed End User response further. Multiple buttons response possible. After end user press the button



the abovementioned script should reply with correspondent post back (value*) to external NLSQL API. Once done NLSQL system will get back with standard API response

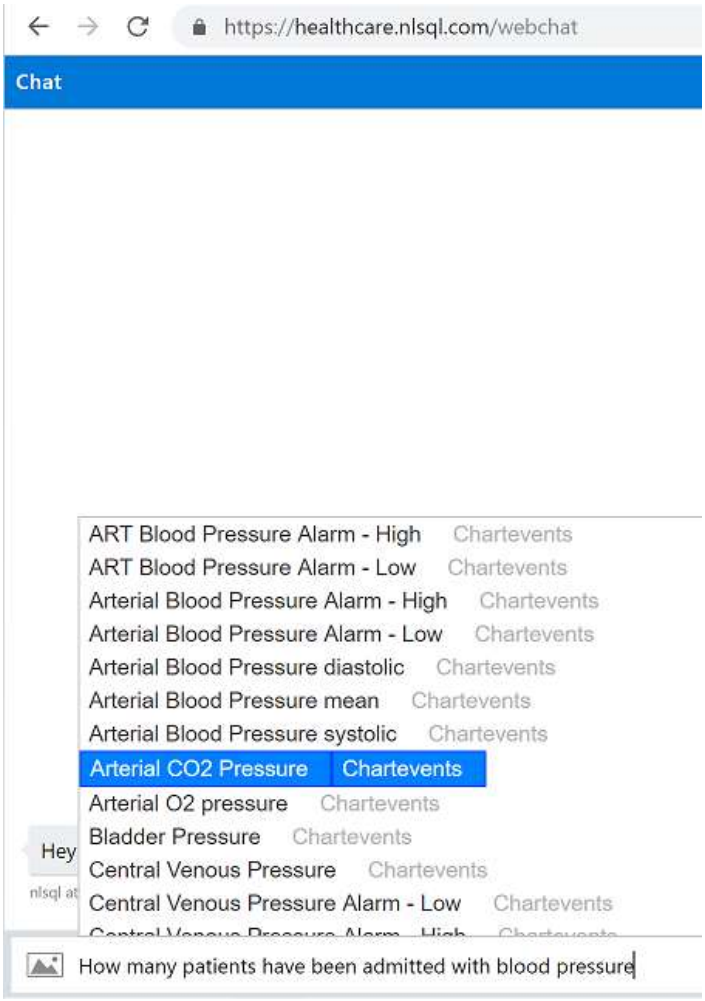
f. **Error,**
Error message type means something went wrong. Errors are the special cases, that is recorder and fixed by NLSQL support team. If NLSQL API replies request with Error type message, end user should receive friendly error message

g. **Custom message types**,
Custom message types is not included into standard functionality, as it should be developed by individual commercial terms. Custom message type included but not limited to Reports, Dashboards, Special Graphs, Waterfalls, Tables, Indicators, etc.

4. All user interfaces supported, which is included but not limited: Instant messenger for SAP, Mobile application, Skype for Business (Lync), Skype, Email, Web chat, Slack, Telegram, MS Teams, Cortana, Google Home, Alexa, Direct line Speech, etc. UI customization is vendor responsibility or it should be discussed separately with NLSQL customer success team with different commercial terms. Vendor is free to choose the most convenient Natural Language User Interface. External NLSQL API is compatible with all abovementioned user interfaces.

5. Customization requirements. In order to be properly customized and for use full features list, NLSQL required to get the following information from the Vendor:
   a. SAP product names and IDs
   b. SAP factory names and IDs
   c. SAP storage location names and IDs
   d. SAP customer names and IDs
   e. List of synonyms end users are used for, if any

6. Additional features. Depends on Natural Language User Interface choose by Vendor different additional features available.
   For Web Chat user interface Suggestions feature available:



   It is possible to implement smart user suggestions using Neural Network based on real user request history. Using suggestion is intuitive and most of end users get used to it as Google search are using the same logic.
   Additional features implementation is a subject of separate commercial terms negotiations and it is not included into default pricing scheme.

7. Architecture diagram of proposed cloud solution for SAP