Test LM Inference – Using Mid Server

```
var payload = {
   "model": "lmstudio-ai/gemma-2b-it-GGUF",
   "messages": [{
         "role": "system",
         "content": "You are an expert in writing haiku."
      },
      {
         "role": "user",
         "content": "write a haiku about the weather."
      }
   ],
   "temperature": 0,
   "max_tokens": -1,
   "stream": false
};

var request = new sn_ws.RESTMessageV2();
request.setHttpMethod('post');
request.setEndpoint('http://192.168.20.28:1234/v1/chat/completions');
request.setMIDServer("lin_mid");
request.setRequestHeader("Content-Type", "application/json");

request.setRequestBody(JSON.stringify(payload));

var response = request.executeAsync();
var responseBody = response.getBody();

var outText = JSON.parse(responseBody);

gs.print(outText.choices[0].message.content);
```

Test LM Inference – Using ngrok

```
var payload = {
   "model": "lmstudio-ai/gemma-2b-it-GGUF",
   "messages": [{
         "role": "system",
         "content": "You are an expert in writing haiku."
      },
      {
         "role": "user",
         "content": "write a haiku about the weather."
      }
```

```
    ],
    "temperature": 0,
    "max_tokens": -1,
    "stream": false
};

var request = new sn_ws.RESTMessageV2();
request.setHttpMethod('post');
request.setEndpoint('https://e1f7-2406-3400-31d-11a0-c856-df62-662b-941e.ngrok-
free.app/v1/chat/completions');
//request.setMIDServer("lin_mid");
request.setRequestHeader("Content-Type", "application/json");

request.setRequestBody(JSON.stringify(payload));

var response = request.executeAsync();
var responseBody = response.getBody();

var outText = JSON.parse(responseBody);

gs.print(outText.choices[0].message.content);
```

Sample Chat Transcript

```
[11:11] Virtual Agent: Please stand by while I connect you to a live agent.
[11:12] [PRIVATE] system: Edmund Blackadder has joined.
[11:12] Edmund Blackadder: Hi, this is Edmund from the Service Desk, how can I help?
[11:12] Baldrick: good morning 😃
[11:13] Baldrick: Could I please check a licence attached to my account?
I thought I had access to Visio, but I'm now being asked to have IT add the licence to my account?
[11:15] Edmund Blackadder: Give me a moment while I take a look at your account.
[11:15] Baldrick: thanks
[11:16] Edmund Blackadder: It seems that you do not have a Visio license.
[11:17] Baldrick: that's really weird because I was using it just a few months ago, before I went on leave. can you see if I had it previously and had it revoked?
[11:20] Edmund Blackadder: Based on your ticket history, I do not see a Visio license requested for you in the past.
[11:21] Edmund Blackadder: Would you know if this license was provided to you a while ago?
[11:22] [PRIVATE] system: User response will indicate if they want to continue the chat.
[11:22] Baldrick: tbh I'm not sure, it's just that I know I was using it in the project position I was in..
[11:23] Baldrick: oh wait - I can access it through the Common, but not through the application itself?
[11:23] Baldrick: is it meant to be like that?
[11:25] Edmund Blackadder: You may need a license to access the desktop version of Visio
[11:27] Baldrick: but I can continue to access through the common? Is it a lite version or something?
[11:30] Edmund Blackadder: Most likely
[11:30] Baldrick: alright, I'll go with that for now and if I need a more robust version I'll request it
[11:31] Baldrick: is there much of an issue getting a licence? I know they're protective of Adobe Pro licences, is this much the same?
```

```
[11:33] Edmund Blackadder: We do generally run low on Visio licenses constantly, but it doesn't hurt to put in the request for one.
[11:34] Edmund Blackadder: Here is a link to raise a request for the new visio license if you  if you need it: https://service-now.com
[11:34] Baldrick: thanks so much 😃
[11:35] Baldrick: if it turns out I do need a full licence, I'll put it through, but for now I'll see if the lite version can do everything I need it to
[11:35] Baldrick: thanks for your help, you can close this one now
[11:35] Edmund Blackadder: No worries, enjoy the rest of your day.
[11:40] [PRIVATE] system: User response will indicate if they want to continue the chat.
[11:40] [PRIVATE] system: Baldrick has ended the conversation.
[11:40] Virtual Agent: Thank you for using our support chat.
```

Code for Local LLM Inference

```
//Insert the chat transcript to the payload message
var chatString = current.getValue('u_chat_transcript');

var payload = {
    "model": "lmstudio-ai/gemma-2b-it-GGUF",
    "messages": [{
            "role": "system",
            "content": "You summarize service desk chat transcripts. You also analyse sentiments from the chat transcript."
        },
        {
            "role": "user",
            "content": "Summarize the chat enclosed within triple hashes in the following text. Please also highlight any important points, actions to take, and sentiments exhibited under separate headings  ###" + chatString + "###"
        }
    ],
    "temperature": 0,
    "max_tokens": -1,
    "stream": false
};

//Access LLM API Endpoint

var request = new sn_ws.RESTMessageV2();
request.setHttpMethod('post');
request.setEndpoint('http://192.168.20.28:1234/v1/chat/completions');
request.setMIDServer("lin_mid");
request.setRequestHeader("Content-Type", "application/json");
//request.setAuthenticationProfile('basic', 'efb1fce3833302105b1cf6d6feaad3e9');

request.setRequestBody(JSON.stringify(payload));

//var response = request.execute();
var response = request.executeAsync();

//Modified ECC Wait Time. Default 30 Sec. Also set sys property
glide.http.outbound.max_timeout.enabled=false
//var responseBody = response.waitForResponse(300);

var responseBody = response.getBody();
var httpResponseStatus = response.getStatusCode();
//Can validate server response status

var outText = JSON.parse(responseBody);
var chatSummary = outText.choices[0].message.content;

current.setValue('u_llm_chat_summary', chatSummary);
current.update();
action.setRedirectURL(current);
```

Code for OpenAI Inference

```
//Insert the chat transcript to the payload message
var chatString = current.getValue('u_chat_transcript');

var payload = {
   "model": "gpt-4o-mini",
   "messages": [{
         "role": "system",
         "content": "You summarize service desk chat transcripts. You also analyse
sentiments from the chat transcript."
      },
      {
         "role": "user",
            "content": "Summarize the chat enclosed within triple hashes in the
following text. Please also highlight any important points, actions to take, and
sentiments exhibited under separate headings ###" + chatString + "###"
        }
    ],
    "temperature": 0.7,
    "max_tokens": 400,
    "stream": false
};

var request = new sn_ws.RESTMessageV2();
request.setHttpMethod('post');

request.setEndpoint('https://api.openai.com/v1/chat/completions');
request.setRequestHeader("Content-Type", "application/json");

//Get OpenAI API Keys already stored in sys_auth_profile_basic table
request.setAuthenticationProfile('basic', 'efb1fce3833302105b1cf6d6feaad3e9');

request.setRequestBody(JSON.stringify(payload));

var response = request.execute();
//var response = request.executeAsync();

//Modified ECC Wait Time. Default 30 Sec. Also set sys property
glide.http.outbound.max_timeout.enabled=false
//var responseBody = response.waitForResponse(300);

var responseBody = response.getBody();
var httpResponseStatus = response.getStatusCode();


var outText = JSON.parse(responseBody);
var chatSummary = outText.choices[0].message.content;

current.setValue('u_llm_chat_summary', chatSummary);
current.update();
action.setRedirectURL(current);
```

Similarity Framework – API Test

```
var mlSolution =
sn_ml.SimilaritySolutionStore.get("ml_x_kitl2_global_global_similar_closed_inci
dents_model");

//gs.info(mlSolution);
//Example key-value pairs
var input = [{"short_description":"User is unable to send receive emails using
outlook",
            "description":"User can not send emails using outlook"
}];

// configure optional parameters
var options = {};
options.top_n = 3;
options.apply_threshold = true;

var results = mlSolution.getVersion(1).predict(input, options);
var obj = JSON.parse(results);
gs.print(results);

var x;
var y;
var predictionValues = []; //In this example, we will store results in an array

//Loop through each identifier
for(x in obj){
       //Loop through each result
       for(y in obj[x]){
              predictionValues.push(obj[x][y].predictedValue);
       }
}

gs.print(predictionValues);
gs.print(predictionValues.length);

var incSummaries="";

var incGr= new GlideRecord('incident');
```

```
   incGr.addQuery('sys_id','IN',predictionValues);
   incGr.query();
while(incGr.next()){

    incSummaries = incSummaries + "Resoution notes of incident#" + 
incGr.number+" : "+ incGr.getValue('close_notes')+ "\n\n";
}
gs.print(incSummaries);
```

Code - Similarity Framework – Update new incident

```
(function executeRule(current, previous /*null when async*/) {

var mlSolution = 
sn_ml.SimilaritySolutionStore.get("ml_x_kitl2_global_global_similar_closed_inci
dents_model");
var short_desc = current.short_description;
var desc = current.description;
//gs.info(mlSolution);
//Example key-value pairs
var input = [{"short_description":short_desc,
            "description":desc
}];

// configure optional parameters
var options = {};
options.top_n = 3;
options.apply_threshold = true;

var results = mlSolution.getVersion(1).predict(input, options);
var obj = JSON.parse(results);
//gs.info(results);

var x;
var y;
var predictionValues = []; //In this example, we will store results in an array

//Loop through each identifier
for(x in obj){
        //Loop through each result
        for(y in obj[x]){
                predictionValues.push(obj[x][y].predictedValue);
        }
}

gs.info(predictionValues);
gs.info(predictionValues.length);

var incSummaries="";

var incGr= new GlideRecord('incident');
    incGr.addQuery('sys_id','IN',predictionValues);
    incGr.query();
while(incGr.next()){

     incSummaries = incSummaries + "Resoution notes of incident#" +
incGr.number+" : "+ incGr.getValue('close_notes')+ "\n\n";
}

current.work_notes ="Similar incidents resolution notes: \n"+ incSummaries;
current.update();

var chatString = incSummaries;

var payload = {
    "model": "gpt-4o-mini",
    "messages": [
        {
            "role": "system",
            "content": "You summarize the IT Service Management Incident resolution
notes of multiple incidents and highlight common issues and resolutions"
        },
        {
            "role": "user",
            "content": "Summarize the IT Service Management incident resolution
notes enclosed within triple hashes in the following text. Please also highlight
common issues and resolutions. Any additional information which can be helpful
but not included in the resolution notes should be provided separately under a
different and appropriate header ###" + chatString + "###"
```

```
        }
    ],
    "temperature": 0.7,
    "max_tokens": 400,
    "stream": false
};

var request = new sn_ws.RESTMessageV2();
request.setHttpMethod('post');
request.setEndpoint('https://api.openai.com/v1/chat/completions');
request.setRequestHeader("Content-Type", "application/json");
request.setAuthenticationProfile('basic', 'efb1fce3833302105b1cf6d6feaad3e9');

request.setRequestBody(JSON.stringify(payload));
//gs.info(JSON.stringify(payload));

var response = request.execute();
//var response = request.executeAsync();
//var responseBody = response.waitForResponse(300);

var responseBody = response.getBody();

var httpResponseStatus = response.getStatusCode();


var outText = JSON.parse(responseBody);
var resNotes =  outText.choices[0].message.content;

current.work_notes ="Gen AI Summary \n"+ resNotes;
current.update();

})(current, previous);
```