## API desktop client

TestAPI is C# desktop based application that is using ivrlla.net.dll and other Sense libraries to
describe how external clients can use API. Ivrlla.net.dll and ivrlla.dll are placed in the API folder in
the project.

Application consists of three sections:

1.Create call request section
2.Live grid section and
3.Database calls and playback section

With the Create call button you can carry out requests for creating the call via API.

From live grid we can receive alertingEx, Connected and Disconnected events from API. Depending if
the event is alerting, connected or disconnected we take some action so we can be sure that something is
changed in live grid.

In the playback section we use direct access to the database from DbUtils.dll. We can see all finished
calls in the list by clicking on 'Refresh calls' and we can take playback action: Export, playback or
playback via Sense Player.

## Connect with the API

For creating external clients that will user Sense API, we need to import our ivrlla.net.dll library that
using ivrlla.dll c++ library and need to be copied in the working directory of

ivrlla.net.dll.
Example of using the API in the code:
.

```
public IvrllaClass api = null
api = new IvrllaClass()
api.set_alertingExCb(AlertingExHandler)
api.set_connectedCb(ConnectedHandler)


api.set_disconnectedCb(DisconnectedHandler)
api.set_loginResponseCb(Login)
api.connectToServer("127.0.0.1", "9803", "", "")
```

Now initialise new IvrllaClass object. Then set callbacks that you want to listen on events. Register the set_loginResponseCb to get responses that the API client is connected with the API server.

Use the connectToServer to connect API client with API server.
Before connecting to the API server ensure that the SenseServer.exe application is up and running or RetellService service from Windows services. After connecting with the API server, use the callback events and request through the API.

## Exporting calls

On selecting call from the call list and clicking on export call button we are exporting call via API. This is
done by sending xml request on socket server with '127.0.0.1' and port 9802. Xml request looks like this:

```
<?xml version="1.0" encoding="us-ascii"?>
<ArchiveCommand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<deleteCallsOlderThen>0</deleteCallsOlderThen>
<scheduleDate>0</scheduleDate>
<password />
<requestId>859d18bbe3ef49979ed28d7f65d522ce</requestId>
<archiveOpType>AOP_EXPORT</archiveOpType>
```

<archiveQueryType>AQ_NONE</archiveQueryType>
<dbArchiveType>DAR_NONE</dbArchiveType>
<restoreType>RST_NONE</restoreType>
<dateFrom>0</dateFrom>
<dateTo>0</dateTo>
<path>C:\</path>
<volumeBaseName>20160909120240</volumeBaseName>
<maxVolumeSize>0</maxVolumeSize>
<addToExistingVolume>false</addToExistingVolume>
<mediaType>MD_NONE</mediaType>
<callid>NCS0000de3eEAB0D35C8A5F4BF989A525F17CC27A15</callid>
<scheduleRepeatType>SCH_NONE</scheduleRepeatType>
<

An Important element in this xml string is the SqlQuery, path, CallID and volumeBaseName. ArchiveOpType must be AOP_EXPORT.

In the TestAPI application we have helper class callback that directly implements the API.

Example:

```
Playback playback = new Playback("admin", Sense17*, "127.0.0.1", 9802)
playback. ExportCall(lstCalls.SelectedItem.ToString(), @"C:\")
playback.FileReceived.WaitOne(5 * 1000)
```

In the playback class we use ManualResetEvent to give responses on the desktop client that show the operation is finished.

Playback call

For playback call use the helper Playback.cs class in TestAPI project.

```
Playback playback = new Playback("admin", Sense17*, "127.0.0.1", 9800,
RequestType.REQ_UPLOAD_WMA)

playback.PlaybackCall(lstCalls.SelectedItem.ToString());
```

Arguments of the methods are: username, password, server address, port and request codec type.
This codec type means in which codec the call will be playbacked.

PlaybackCall method in Playback.cs class is using Socket connection to connect with our Server API.
On connecting the socket we send a request with a specific format and expect reponses. The responses
can come as one or more messages. The first response string is the response code (0 -success), the second response string is the size of the file in bytes and third one is the file bytes.

Example for response
0; 16342; datadatadata
0 - success
16342 bytes of filesize
datadatadata- file contents

Playback Sense Player
On clicking the Playback Sense Player button on the client, start the process with specific arguments given in the example:

```
bool exportCalls = true;
Process process = new Process();
process.StartInfo.WorkingDirectory = Path.Combine(playerWorkingDirectory);
process.StartInfo.FileName = GlobalParams.SENSE_PLAYER_FILENAME;
process.StartInfo.Arguments = String.Format("u{0} c{1} b{2} s{3} \"d{4}\" e{5}",
Username,
callid, 1, "FinishedCallsPlayer", "Description", exportCalls);
process.Start();
```

We need the Call ID that we are trying to playback, the working directory of Sense Player application,
usually at 'C:\Program Files (x86)\Retell\Sense Calls\Bin\Player\' and exportCalls flag. This flag is
used to export calls from the Sense Player