QA5 SDK v. 5.5

Product Description & User Guide

Nemesysco, Ltd.

Version control

Issue	Date
Version 5.5	March 2009

6 Hagavish St. POB 8747 Sapir Industrial Park Netanya, Israel 42504 | Tel: +972-9-885-3864 | Fax: +972-9-885-3782 | www.nemesysco.com

Contents

1. Purpose of Document

This document provides a description of the features, capabilities and technical functionality of the **QA5 SDK, Version 5.5.**

1.1. What's new in QA5 SDK Version 5.5

	Change	Page
1.	Call Priority enhancements	
2.	LioNet engine modified & enhanced for better performance	13, 38
3.	New value: EmoValsArray(18)	31
4.	New function: ***nmsQA_getProfilerData***	35
5.	New function: ***nmsQA_getChangesFromProfiler***	35
6.	New function: ***nmsQA_InitalizeCallPriority***	35
7.	New Function: ***nmsQA_getCallPriority***	36
8.	New function: ***nmsQA_getMaxCallPriority***	36
9.	New Function: ***nmsQA_CollectAgentScoreData***	36
10.	New function: ***nmsQA_GetAgentScore***	36
11.	New function: ***nmslioNetForget***	42
12.	New function: ***nmslioWakeUp***	42
13.	New function: ***nmslioGenerateFullReport***	43
14.	New function: ***nmslioGetNetDetails***	43
15.	New function: ***nmslioGetCurrentStatus***	43
16.	New function: ***nmslioSetParmsNames***	43

1.2. Changes in QA5 SDK Version 5.5

	Change	Page
1.	Function modified: ***nmsConfigTestData***	29
2.	Function modified: ***nmslioNetGo***	33
3.	Function modified: ***nmsLioNetSave***	34
4.	Function removed: ***nmslioKillNet***	40
5.	Function modified: ***nmslioGetAllAnswers***	42
6.	Function modified: ***nmslioNetSleep***	42
7.	Function modified: ***nmslioTakeDeepSleep***	42
8.	Function removed: ***nmslioGetOwner***	43
9.	Function removed: ***nmslioGetlstBrdChk***	43

2. Product Description

2.1. General

- QA5 is a Software-Development-Kit (SDK) for Emotion Detection that utilizes Nemesysco's advanced voice analysis technology to detect and measure anger, stress and other relevant emotions that may arise in call center conversations. It is designed as a set of components that can be integrated into an existing call center solution to monitor agent-customer interactions and identify calls that may require special attention or intervention.
- In its online use, QA5 can analyze all ongoing calls in real-time and alert managers when problematic call events occur. In its offline use, QA5 can be used to identify patterns of interaction within calls and evaluate specific aspects of customer service quality, as well as provide insights about customers' preferences, likes and dislikes. The SDK also encompasses Nemesysco's LioNet™, an advanced learning system component which provides the ability to automatically tag archived conversations according to criteria defined by the call center's managers or policies (i.e., "Problematic," "Successful," etc.).
- Version 5.5. of QA5 adds 2 new functions into the core component, providing the new "Call Priority" and "Agent Priority" scores, both designed to assist your understanding of the call status both from the agent and the customer sides, indicating the priority of actions required.

2.2. Industry Rationale

- A company's most vital asset is its customers. Meeting customer needs and focusing on customer satisfaction helps ensure that your customers will continue to want to do business with you, as well as recommend your company to others. Even when monitoring the textual content of customer support calls, customers do not always explicitly state their preferences or feelings about their interactions with your company's CSR agents. Superior tools are needed to determine your customers' true emotions and reactions, quickly and efficiently.
- Integrating the QA5 emotion detection components into your call center solution gives you a significant competitive edge in today's market, where unsatisfied customers can and will easily take their business elsewhere.
- Using Nemesysco's Emotion Detection Platform, the QA5 SDK is able to detect and measure a wide variety of emotions and cognitive states including stress, anger, embarrassment, satisfaction, excitement, rationality and more. Each call center site may customize its data display according to criteria that is most relevant for its business or evaluation needs.
- By incorporating QA5 into your call center, supervision capabilities will be dramatically enhanced with real-time indicators, smart and efficient analysis of recordings databases, performance logs and reports of your agents' progress, and advanced tools to learn about customers' likes and dislikes.

2.3. Business Benefits

- **Save clients** - monitor conversations in real-time, and automatically detect when intervention is required.
- **Save time** - scan mass call logs/recordings and automatically flag calls that are classified as "interesting" or "urgent," according to pre-defined criteria.
- **Gather intelligence** – zoom in on any call to obtain feedback on your clients' preferences and reactions.
- **Improve CRM methods** – measure the effectiveness of your customer service operations and agent training.
- **Maximize agent productivity** – monitor agent performance to assure quality, prevent burn-out, and reward good performance.

2.4. Focus of Analysis

QA5 can analyze both ends of a customer service call. For example, the technology can be used to focus on the customer's side of the conversation, identifying specific emotion-related events that may take place during the call. When such events occur, the system alerts the manager in real-time, allowing him to monitor the call and determine whether an intervention is needed. Additionally, QA5 can be used to focus on the technique and effectiveness of the agent handling the call. Monitoring agent performance helps assure CS quality, prevent agent burn-out and churn, reward good performance, and maximize the efficiency and effectiveness of your customer service operations.

2.5. Product Highlights

- Scans all customer-agent interactions in real-time to detect problematic scenarios as they develop, and enables automatic alerts to CS managers.
- Performs efficient offline emotional data mining of archived calls, based on criteria defined by any CS manager, seamlessly, during normal work procedures.
- Measures different emotional parameters (such as averages of stress and anger) over any time period to assess overall CS site performance and generate automated report.
- Monitors any agent's performance and emotional development over any period of time.
- Identifies agents that are having a bad day instantly, allowing taking positive action.
- Applies an automated scoring system to reward agents' performance.

2.6. Technology

2.6.1. General

QA5 SDK employs **Nemesysco's ED (Emotion Detection) Platform** to detect and measure a wide range of emotions and cognitive states together with the LioNet™ learning engine, allowing you to define, train and detect unique self-defined emotional states to meet specific customer needs. Unique emotional states may include: "cancel order", "ready to buy" or any other

type of emotional situation that you deem important. This unique combination of automatic detection of a variety of emotions, together with the ability to train the system to detect any other emotion of interest, in real-time or from pre-recorded material, is what makes QA5 the most sophisticated, cutting edge voice analysis technology available today.
The ED Platform is based on Nemesysco's patented signal processing voice analysis technology (originally designed for security use), and a smart learning engine to classify voice segments and complete calls according to user specifications.

2.6.2. Audio Formats

The technology at the core of the QA5 SDK is able to analyze a variety of voice formats typically used in call center conversation recordings, both from a file or streaming directly from the IVR. These include 6/8/11 KHz sampling rates in 8/16 bit depth, as long as the data provided is in a single channel (mono), non-compressed PCM format.
While the system is able to handle two speakers within the same conversation, best results can be obtained when analyzing a stream/file of only one party in the conversation separated from other voices.

2.6.3. Integration

QA5 SDK contains the QA5_com Windows COM object (**NMSQA5COM.dll**) that contains all the functionality and objects required for development. To assist with product integration, 2 demonstration applications and VB6/DOT.NET source codes are provided, illustrating the different areas of functionality and capabilities supported by the QA5 SDK. This executable and code are also designed to function as the QA5 "System Training Manager" - enabling quick, efficient training and the customization of the QA5 SDK to meet your specific needs, if these are required for your system design and preferred architecture.
The QA5 Core component can process a large number of calls in real-time. However, exact performance is dependent on the type of hardware used in the existing system architecture.

3. Basic Definitions and QA5 Vocabulary

1. **LioNet™** – Nemesysco's Learning Engine.
2. **Voice Segment** – a segment of voice ranging in length from 0.3 seconds to 1, 2 or 3 seconds, according to user specifications and the system's internal logic. The segment is usually a logical speech block, containing a word or a part of sentence, however, this is not always the case and sometimes, sentences will be separated into different voice segments.
3. **Emotional Signature** – a string of numbers that summarizes the emotional activity detected in a specific voice segment by the core component. The Emotional Signature contains all of the relevant data that will be used by LioNet™ to produce its final analysis.
4. **Emotional Signature of a call** – a string of numbers that summarizes the emotional activity detected in a call by the core component. The Emotional Signature contains all of the relevant data that will be used by LioNet™ to produce a final conversation's "Priority Flag".
5. **Priority Flag** – user-defined flag for a call, determined by LioNet™ per call based on the Emotional Signature of the call, or used to train the LioNet™ according to end-user needs. Priority Flags may vary from "Normal Call", "Interesting Call", to "Immediate Attention" or even "Canceling Orders" calls, etc.
6. **Basic Parameters** – A set of proprietary 51 voice element parameters indicative of the various emotions based on Nemesysco's SENSE technology. The parameters themselves are not exposed through the SDK.
7. **Borders** – currently known Minimum/Maximum set of all basic parameters' ranges of activity in a predefined type of call.
8. **Envelopes** – currently known normal and acceptable internal set of all basic parameters' ranges of activity in a predefined type of call trained into the system. The Envelope is calculated as the Low-Median average and the High-Median average of each and every basic parameter.
9. **Calculated Emotions** – set of outputs defined in the EmoVals structure (see Source Code Structure), which are the result of mathematical interpolations of the Basic Parameters.
10. **BG Level** – Setting of the expected and normal background noise level. The core component uses this value as part of the voice stream segmentation logic.

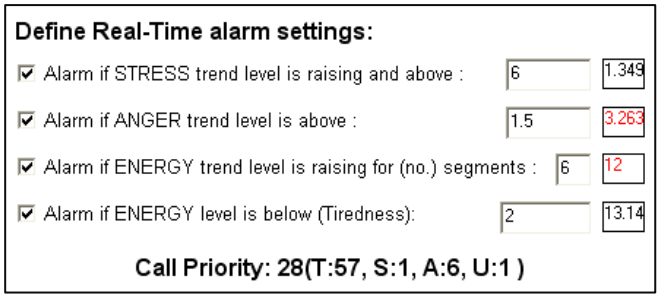
4. Designing a QA5 based system

The QA5 SDK can be used in many ways in your call center application. This section is designed to assist you with your new system design.
In principle, the QA5 features can be used in 4 ways:

1. Online/real-time analysis
2. Client emotional profiling to assist in sale approach
3. Offline/post processing analysis
4. Report generation

4.1. Online / Real-time implementation

The QA5 output's range of emotions is very extensive and changes rapidly. In most real-time applications, the needs are far more limited. The typical human operator limited capacity, additional duties and limited desktop space that must be shared with various other indicators of performance should be taken into consideration. In addition, due to the nature of psychological reactions and the QA5 limitations, it is important never to look at the single voice segment as a determining factor, but to examine several sequential segments and **establish the emotional development trend**.
Typically, the more relevant emotions and emotional states for online use are: energy, stressed, upset, angry and content. Concentration, SAF, Excitement and Confusion may also be used for different purposes like assisting sales.



Therefore, the QA5 can assist you with designing and implementing new business models for monitoring and analyzing calls using emotional data. Use the relevant emotional indicators gathered during or after the call and formulate them into your business process needs (i.e. unsatisfied customer, hesitant support person, people's reaction towards a sales pitch, etc.).

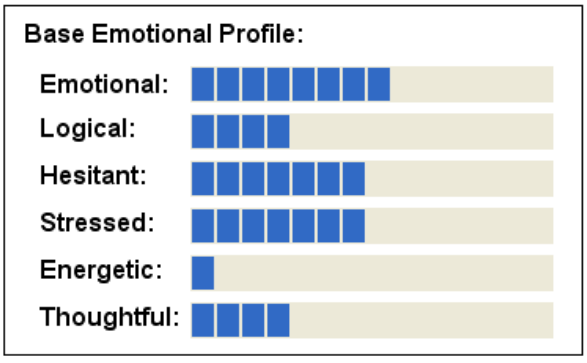
The QA5 (5.5) sample code provided with the QA5 SDK contains an example for one possible "Call Priority" Flag formula, as well as a new "Agent Priority" score. In this example, the online Call Priority flag is a single value scale, ranging from 0 to 100 (0 being a standard call and 100 reflects "Immediate attention needed").
The Agent Priority examines other emotional indicators to evaluate the performance of the CSR. The higher the score, the lower the quality of service is.

The business logic implemented in the Call Priority sample code assumes that repeating anger indications are to be dealt with in a much higher priority than indications of stress. Having said that, indications of stress should not be disregarded, and repeating indications of stress will also contribute to the call's priority increscent, as they may hold important information about customers' typical problems that should be reviewed at the management level.
In addition to the Agent Priority score, real time monitoring can also collect data about the general emotional state of the agent, his current stress and boredom level, and his customers' reaction to his responses.
In an optimal system design, a brief emotional history of any specific call should be displayed upon request, showing trends of emotional development within the call. This information should reveal enough details to the manager on site to decide whether he wishes to join the call or take some other immediate action. The details screen can also combine, for example, additional information about the customer and his past activities as registered in the call center database.

In addition, a flexible set of definitions (rule base) can be added to your system, defining which indications will be provided to the manager by the system, attracting his attention to a specific call.

> If you choose to incorporate real-time use of LioNet for the detection of specific key events, a business logic must be designed to take into account the method of training LioNet (in a continues fashion on site) and the way to present and handle events that are generated by LioNet.

4.2. Client Emotional profiling to assist in the sales approach



The customer's emotional profile may be an important indication to the agent handling the call, allowing him to adjust his support and sales approach to a more psychologically suited way for this type of personality. The emotional profile of the client is determined by the QA5 core after 5-6 voice segments and contains 6 emotional elements. These emotional elements can be used by the call center's professional team to design special instructions to handle different types of personalities.

The emotional profile should not be considered as a constant value per customer, and it is very likely that in different interactions and different days the emotional profile will change. QA5 actually monitors the changes in the emotional profile throughout the call to enable the detection of sharp changes that may be indicative of various significant events in the call.

4.3. Offline / Post processing analysis

Offline analysis of files provides the exact same data as the real-time analysis, but in this case, much more emotional data can be used and explored to provide better understanding about the customers, their likes and dislikes. In addition to this "segment level data", several conversation summary reports are generated and can be used for different purposes as described below:

4.3.1. Using the "Envelop and Borders" sub-system

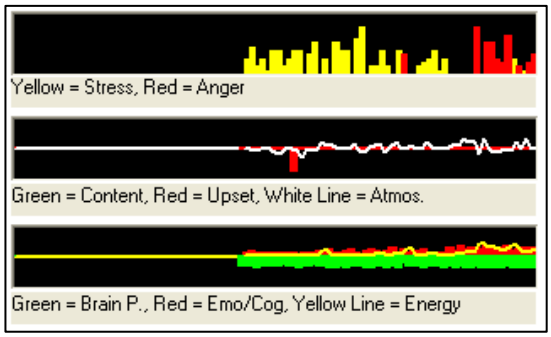
The very basic offline-based system design is using the "envelop and borders" sub-system. Using this sub-system, managers can, through their daily routine of listening to files and using a one-click operation, train the system to identify what they deem as "Normal call" that requires no further attention. If the call is determined by the manager as "Normal", the QA5 system should be instructed to simply "Update the known envelope and borders data file". From this point on, any conversation that produces an emotional activity range that falls within these levels is deemed "Normal" by the system, and anything else should be flagged in the calls database as worthy of further attention. The "envelop update" function is described in details in the demo application code.

4.3.2. Using LioNet on the conversation emotional signature

A more advanced utilization of the offline mode database marking can be done using the "conversation level LioNet" and specific feedback for each manager. While listening to calls, every manager may be looking at different aspects of the call, which are at times not easily verbally definable. For example, a manager responsible for training of products in the call center may be looking for calls in which the agent simply doesn't know what he is talking about, while the manager responsible for the customer service level is interested in the way the agent is handling the call. Using LioNet and the automatically generated conversation Emotional Signature string, the manager can feed into the system his own classification of the call, such as "Normal" (requires no further attention), "Interesting" or even "Urgent attention". It should be emphasized that LioNet can be trained to identify many other emotional states as may be needed to meet your system's definition. Training LioNet consists of a very simple operation; however an instance of LioNet should be created per manager and should be associated with a different LioNet data file.

Please note that Version 5.5 produces a different structure of the Emotional Signature that is not compatible with previous versions.

4.3.3. Graphical display of emotions in a conversation



Using the segment data generated during the analysis, it is recommended that your system's conversation player will present some type of graphical layout of various significant emotions, and some trends of development. That way, a manager can easily highlight the portions of the call where emotional reaction is detected to get to the bottom line faster.
See the Emotional Player in the QA5.5 SDK for more ideas of possible implementations.

4.3.4. Using additional analytics tools in conjunction with QA5

When coupled with word spotting or user identification, your system's capabilities can be further enhanced by providing, for example, emotional analysis of specific key-phrases like product names or extended emotional profile of different customers.

4.4. Report generation

Based on the various emotions captured over time, your system can be programmed to generate various types of reports. For example, a specific agent performance over a period of time (angry calls Vs. normal calls, average stress levels, average energy levels in different times of the day, etc.). Another important report may include the overall average per site or group of the Call Priority, Agent Priority, angry calls ratio or agents stress levels parameters.
In addition, averages of emotional values from different groups of agents handling different topics in the call center environment over different periods of time may reveal development of problems in different areas, for example, a sudden raise in the stress level of agents dealing with a certain product may indicate a mass detection of a failure in that product.

5. QA5 SDK Overview – Architecture and Components

QA5 SDK is a set of COM objects designed to quickly and efficiently perform complete analysis of emotions for quality assurance and speech analytics. Through this process, voice analysis becomes easy to comprehend and master.

QA5 SDK consists of two main components:

1. **QA5 Core:** serves two purposes:
    - License Server mode: issues operation licenses for other instances of the QA5 Core initiated for analysis (see B below). This instance is responsible for the communication with the HASP protection plug physically attached to the system, and receives the license details from it.
    - Analysis Mode: functions as the main analysis component for the call/voice channel. A new QA5 Core object must be used for each voice in each call (preferably on a separate thread); For example, one instance/thread is used for the operator and one for the customer calling. So, if your system is currently analyzing two calls, then four instances should be used (in addition to one instance for the License Server).
2. **LioNet™:**
    A powerful Learning Engine, designed for automated decision making based on continuous feedback input. This component does not require preparation or special configuration other than its creation, established by a single command argument. QA5 (5.5) SDK includes a newer version of the LioNet Engine which is faster and more robust than previous versions.
3. **Trend Analyzer:**
    A simple Com object designed to collect selected emotional data & calculate trend and rotating averages.

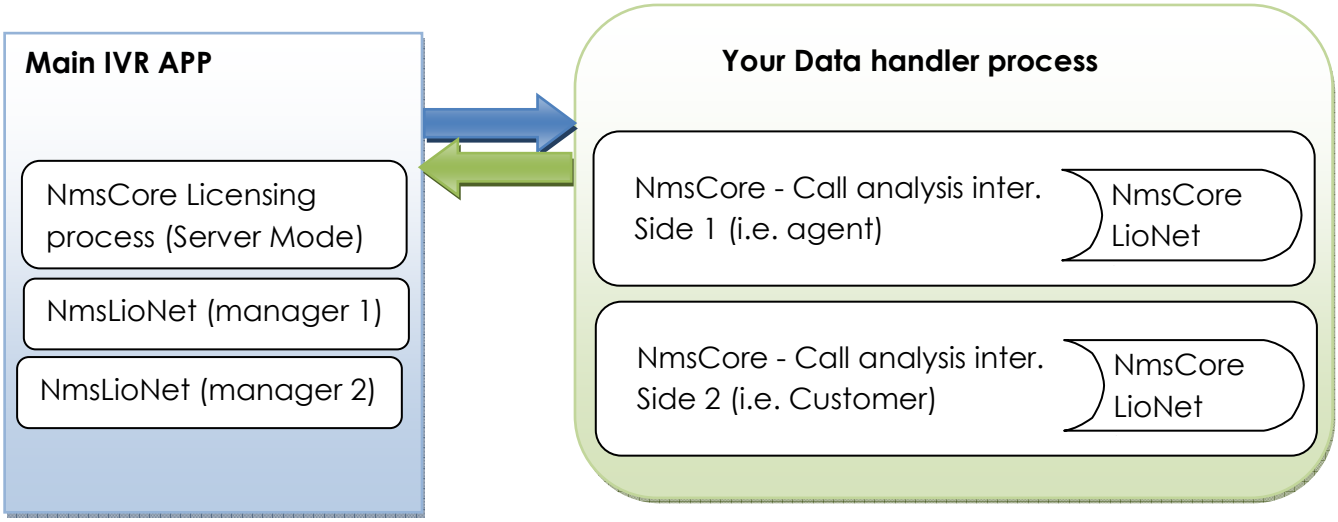
5.1. General System Architecture

The SDK is designed to interact with your main application through a data handler process that runs Nemesysco's emotion analysis engines.
The diagram shows the 2 parts of the system:

- **Main app**
    - Core - License Process interface (Server mode): in charge of providing the QA5 with license code for each call analysis interface
    - LioNet: Providing the final LioNet analysis in the "Conversation Level" for archiving and tagging purposes.
- **Data Handler Process**
    - Core - call analysis interface (Sides 1 & 2): receives the voice data stream from the phone system and in return provides

emotional analysis information for both sides of the conversation (i.e.: customer service & customer).

- LioNet (QA5 Core internal instances): are used by the analysis engines to detect the new trained emotions on the Segment Level.



6. Data Flow and Licensing System

6.1. Analysis process

The following is a common implementation of the QA5 online analytical process:

- Once started, the host application should create an instance of the QA5 Core in License server mode. This instance is a part of the Main application, and should not be used for voice analysis purposes.
    - In addition, the Main app. creates one or more global instances of the LioNet Engine and loads the knowledge data files relating to the conversation level Emotional Signatures for each defined manager.
    - **NOTE:** Only one license server can and should be used, and you should not run multiple instances of the server in different applications on the same PC. This may and potentially cause a serious and undetectable error that will lead to a crash.
- The License server will look for the Hasp Protection Plug and retrieve the license details. If the plug is missing an error message is generated.
- The system will now wait for call notifications.
- Once a call is received and connected to an agent, the system will initiate a **Call Handler process**, containing one or two (depending on your system design) new QA5 Core instances for analysis.
    - Each QA5 analysis instance must receive a unique activation license from the QA5 License server instance. For that, the Instance ID of each analysis instance is sent to the license server and a unique code is generated specifically for it. If the license is invalid, an error is generated.
    - The Call Handler process then streams the entire voice data of the conversation to the QA5 analysis instances, so that the 2 sides of the conversation are being separated into 2 voice channels that are streamed each to its own analysis instance.
      The voice buffers are typically streamed in buffers of 20 ms length, 6/8/11 KHz sampling rates, 8/16 bits of depth, PCM RAW format (Uncompressed!).
    - For every 20ms buffer, the analysis COM returns an indication:
        - **RC= 0** – Voice buffer is silent – No voice detected
        - **RC = 1** – Collecting voice for analysis
        - **RC = 2** – Voice buffer analyzed, emotion property bag is ready
        - **RC= -1** – The analysis of a voice segment failed or is suspected to be noise
        - When analysis is ready (RC= 2) the hosting call handler process will apply its business logic to determine system dependent actions. Other return codes may also be used in different UI and business logic needs.

- Once the conversation has ended the system will initiate the report generation instructions to the analysis COM objects and receive the final

conversation summaries, including the LioNet Emotional Signature data for the speaker throughout the call. Then, the analysis instances as well as the call handler are released.

- The reports information is passed to the Main hosting application for archiving and final Business Logic operation. The Main application will also analyze the Emotional Signature using its global LioNet instances for archiving and database operations.

6.2. Creating a License

- Each instance of the QA5 Core requires a unique license string obtained from the QA5 Core Server instance to function. The QA5 Server instance requires the QA5 Plug to be connected in order to operate properly. The plug should not be removed from the running PC as long as the system is running. Only the first instance of the created QA5 Core can and must be used as the QA5 Licensing Server.
- In order to create the server instance, define it in the highest level of your application, so it may be accessed from any part of your application, as in the following example (VB6):

```
Public nmsQA5LicServer As New nmsQA5core
```

Then, use the server initialization command when you run your application:

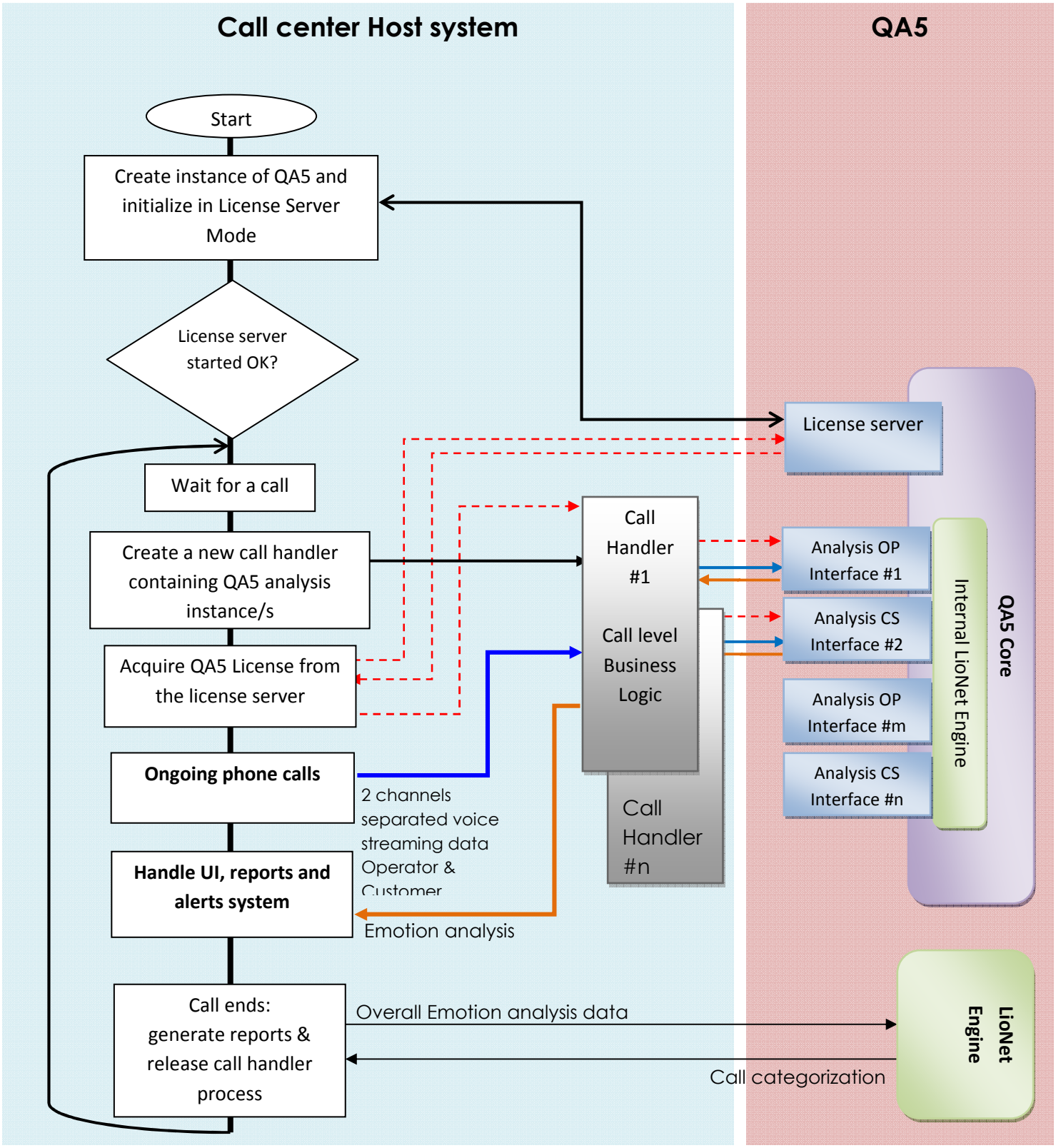
```
rci = nmsQA5LicServer.nmsSRV_INIT(DEF_OWNER)
```

  - Where rci is an integer return code
  - DEF_OWNER (string type) should be set as your company name prior to calling the function
- The function will normally return ***NMS_OK,*** indicating the server is up and running. However, it may also return a number of errors that must be tracked and handled (see List of Constants and the VB6 Example "QA5 Trainer" Source Code).
- Each of the analysis processes has its own internal Process ID, assigned when created, that must be passed as an argument to the License server. The function ***nmsSDKgetProcessId*** retrieves this internal number.
- **NOTE:** The first time you call the ***nmsCoreObj*** (i.e. for analysis purpose), it will also be created in the scope of this function. In order to obtain a license for a new analysis process, use the following function:

```
ActivationLicense = nmsQA5LicServer.nmsSRV_GetActivationLicense
(_nmsCoreObj.nmsSDKgetProcessId)
```

  - The function will return a string type argument, containing the activation license for the process, or a string formatted error code.
  - Refer to Constant Definitions and Source Code examples to learn about possible errors and how to handle them.

7. Analysis flow diagram



8. Call Configuration in the QA5 Core

8.1. Calibration Types

The calibration process is the initial phase performed by the ED Platform voice analysis technology to determine and detect the acceptable ranges of the Basic Parameters in the specific call and working environment. QA5 SDK has two built-in types of Calibrations:

- The original calibration (Type 1), which takes the first 10 segments for the calibration purpose, while producing partial results during the calibration phase.
- Shorter calibration process (Type 2) that uses only the first voice segment for this purpose.

For most QA5 systems, and in most instances, it is safe to use the Type 2 Calibration. For this reason, it is the default type in our sample application.
Normally, you would want to define the Calibration type as a constant, and use the same calibration type every time. It is advised not to change the Calibration type, even intermittently, as it will have an impact on the Emotional Signature creation and analysis.

8.2. Segment Length

QA5 SDK can be configured to segment the streaming voice data into Voice Segments of maximum length of 1 second, 2 seconds or 3 seconds. Note that the system may still produce shorter Voice Segments than the maximum according to its internal logic.
Shorter segments may produce more rapid data, however, with "sharper" changes, since the human voice is very sensitive to emotional brain activity, and that is a very fast, normally somewhat "spiky" process.
Longer segments will produce a more stable analysis, often eliminating the sharp changes and "generalizing" the emotions, however, depending on your needs, may prove to be too slow for your specific design.
It is recommended to use the 2 second voice segments to benefit from both rapid and comprehensive results that are typically more "humanly understandable".

8.3. Determining the Back Ground Noise (BG) Level

There is no way to predict in advance the optimal BG level for each site without testing some of its recordings, as the BG level also relates to the working environment itself. Therefore, it is advised to obtain several calls recorded in the destination environment, and by experimenting, determine the level that introduces a minimal number of noise segments, yet eliminating the minimal amount of real voice data.
In the QA5 Trainer, it is also recommended to view the wave window, and be certain you see a clean voice wave with a distinct beginning and end.

NOTE: Following this process, some calls will contain a higher noise level, as the caller may be in a noisier environment. It should also be noted that a crying baby, a barking dog or a passing car could temporarily affect segment analysis results, as well as (sometimes) the emotional state of the speaker.

9. Analysis Types and Analysis Sub-Systems

9.1. Calculated Emotions - Definitions and Limitations

SENSE7 technology produces a set of 12 normalized emotions and additional 5 indications of the tested party's emotional activity. These emotions are defined in the **EmoVals structure**:

Emotion	Emotion Type	Description	Ranges / Normal Level
Energy	Basic	Perhaps the most indicative parameter QA5 exposes. Conversation energy indicates if the speaker is sad or tired at low values (below 5), comfortable (5-9) or highly energetic (above 9). Values at the range of 0 to 1 may also indicate boredom.	Range is 0 to 50. Normally expected values are between 3 to 9.
Content	Calculated	Indicates how pleased or happy the tested party is, but due to the psychological nature of different conversation modes, in times may appear in an argument as well.	0 to 30, normal value is 0
Upset	Calculated	Indicates how unpleased or sad the tested party is.	0 to 30, normal value is 0
Angry	Calculated	Indicates how angry the tested party is. Please note that the system may accidently detect anger in different conversation moods.	0 to 30, normal value is 0
Stress	Basic	Indicates how nervous the tested party is.	0 to 30, normal value is 0
Embarrassment	Basic	Indicates how	0 to 30,

Emotion	Emotion Type	Description	Ranges / Normal Level
		uncomfortable the tested party is.	normal value is 0
Intensive Thinking	Basic	Indicates the tested party is thinking intensively while speaking.	0 to 30, normal value is below 3
Imagination Activity	Basic	Indicates that the tested party is either recalling information from his memory or visualizing something.	0 to 30, normal value is 0
Hesitation	Basic	Indicates how comfortable the tested party was when making the statement. Below 14, the subject is more comfortable than normal; above 17, the subject is regretting what he is saying.	0 to 30, normal range is 14-17
Uncertainty	Basic	Indicates how certain or uncertain the tested party is about what was said. Below 15, the subject is more certain; above 15, the subject is more uncertain.	0 to 30, normal value is **15**
Excitement	Basic	Indicates how positively or negatively excited the tested party is. Below 15, excitement is negative; above 15, indicates higher excitement or anger.	0 to 30, normal value is **15**
Concentration Level	Basic	Indicates how concentrated the tested party is.	0 to 30, normal value is 0
SAF	Calculated	Arousal factor, indicating deep and profound interest in the conversation topic (positive or negative!).	0 to 30, normal value is 0
Extreme State	Calculated	A value indicating how extreme the overall	0 to 30, normal

Emotion	Emotion Type	Description	Ranges / Normal Level
		emotional activity is according to a unique logic pre-programmed in the QA5 Core.	value is 0
Atmosphere	General	A general value indicating the emotional atmosphere during the conversation. Negative values normally indicate general uncomfortable feelings; positive values indicate positive feelings.*	(-100) to 100, normal range is (–10) to 10.
Brain Power	General	Overall summary of both emotional and cognitive processes in the brain. Used mainly for research purposes, but may also be useful assisting through research for your own needs and alert definitions.	0 – 6000 (both theoretical) Normal ranges are between 700 and 1000
EmoCog Ratio	General	Indicates rationality of the subject, i.e., to what extent the subject is talking based on emotions or logic. Above 100, the subject is more emotional; below 100, the subject is more logical.	1 to 5000 (both theoretical) Normal ranges are between 50 to 150
Aires	Trained Response	LioNet™ response (String Type) based on user-defined trained segments of emotions. If no emotions were trained, the Aires will return an empty string (“”).	String

* **Atmosphere and Happiness/Anger Conflict** – When people get angry and involved in an argument, or when “revenging”, they may respond (in the segment level) with sudden unexplained “happiness”, that may or may not appear together with the anger indication. Although this phenomenon can be easily explained psychologically, sometimes it is not immediately clear

when reviewing QA5 results. Consider this fact when designing your own system.

LioNet™ Segment Analysis

The LioNet™ response (String Type) is based on user-defined and taught "segment level" emotions. If no emotions were taught, the Aires will return an empty string ("").

You can train the LioNet™ to identify any number of new Boolean type emotions (whether they exist or not), to meet your needs and your system requirements. A User Defined Emotion may be simple like "boredom" or "interested", and may also be more state-indicative such as "Canceling Order" or "Ready to Buy".

Please note: Properly training LioNet™ an emotion may require 100 or more segment samples and training sessions, and will require proper testing procedures to ensure the accuracy of the specific emotion. Furthermore, the more new recognized emotions stored by LioNet™, the longer the time the system may take for processing its result.

Once you begin to train LioNet™ new emotions, you must also train LioNet™ to identify what a "Neutral" segment is (one containing no special emotions), since LioNet™ will only choose from the range of emotions it was taught.

9.2. Stress Analysis

The QA5 Core contains a sub-system for semi-clinical stress detection that can be used for monitoring the work-related stress level of your agents. This information can be used to automatically warn managers about a highly stressed employee that may simply "burn-out" or suffer serious stress-related symptoms if not treated. Stress analysis may also be used in selected systems to monitor the calling party's stress levels, i.e., call centers related to medical monitoring.

Stress levels are provided as a set of two values (high/low), and a final determination code is sent back as the return Code (see Constants Definitions for details).

Stress analysis is produced per voice channel (if configured upon initialization). High and low values indicate the normally high and low stress levels of the tested party. However, the low value is the more important variable, since it indicates how RELAXED the tested party can potentially become, and if low enough, the tested party can still reduce his own stress by himself.

A high "low" value will be deemed as any number above 45, a high "high" value is above 55.

Your system can be configured to monitor and log each agent's stress development over time. This can be used to generate periodical statistics

and reports, as well as issue immediate warnings to managers in extreme cases, so they may take affirmative action and assist the representative.

A typical stress report:

Stress Report: WARNING LEVEL: 0 - High/Low: 35/16

9.3. "Borders and Envelope" Analysis

The QA5 system can be configured to use the "Borders and Envelope" analysis sub-system, keeping a simple set of the Basic Parameter ranges and normal activity values in what you deem to be a "normal call profile". This can be used to match any new calls against these set of borders to detect any unique profile, which abnormally differs from the predefined borders.

9.3.1. Borders System - Internal Architecture

System data is stored in two levels:

"Borders" level: includes the highest and lowest values ever detected in a call (and were "accepted" as a part of the Borders and Envelope by the system architect or user).

"Envelope" level: tracks the highest and lowest MEDIAN values of "acceptable" calls, and is therefore the normally expected range of activity of the parameters.

Technically, you can manage any number of Borders files for any purpose and load them per analysis component immediately following its creation.

9.3.2. Borders Analysis per Segment

If configured to do so, QA5 Core will produce a return value indicating the overall deviation in the current segment from either loaded Borders or loaded internal and expected Envelope.

Clearly, we can expect to see greater differences from the internal Envelope than from the general Borders, something to be considered when designing your system and using this sub-system.

9.4. LioNet™ Call Analysis and Emotional Signature

The QA5 Core produces, on request, an "Emotional Signature" per call, summarizing the emotional activity in the call, and taking into account changes over time and other factors. The Emotional Signature is a long string containing a vector of numerical expressions and operational tags.

Once the Emotional Signature is created by the QA5 analysis core, it can be tested by the hosting MAIN LioNet's instance in order to receive the user-defined "Priority Flags" or "Conversation Analysis" if the call falls within the LioNet set rules. If a Priority Flag is raised – you can use it to create an immediate notification to the relevant manager or add it to the conversation log database for later use.

Your system should be programmed to keep the Emotional Signatures of the calls in a database, so the user can further train and fine-tune the system's analysis to suit his own requirements.
In fact, several instances of LioNet can be created by programming one for every manager, to further meet the specific needs of each manager, and fine-tuned to suit his exact needs. This being the case, the same Emotional Signature should be analyzed by all manager LioNet instances.
NOTE: In the above case (multiple managers), a different file name should be used for the different managers LioNet data files!

Please note: QA5 Version 5.5 produces a different structure of the Emotional Signature that is incompatible with older versions.

10. Call Profiler and QA5 Summary Reports

10.1. Call Profiler

The Call Profiler is a statistical report containing a summary of all the **output emotions** and emotional activity detected in the call, divided into ranges of low, medium and high.
Call Profiler provides a rough overview of the call analysis, and can quickly indicate what calls are worthy of further inspection.
Following is a typical Call Profile of a call in which very high levels of ANGER were detected. Note that the red bolded section is not originally colored in the report:

Content: Avrg Level=0.83, No Reaction=90.83%, High=5.00%, Mid.=1.67%, Low=2.50%
Upset: Avrg Level=0.00, No Reaction=100.00%, High=0.00%, Mid.=0.00%, Low=0.00%
Angry: Avrg Level=5.51, No Reaction=41.67%, High=41.67%, Mid.=12.50%, Low=4.17%
Stress: Avrg Level=1.52, No Reaction=60.83%, High=5.00%, Mid.=12.50%, Low=21.67%
Embarrassment: Avrg Level=3.33, No Reaction=70.83%, High=29.17%, Mid.=0.00%, Low=0.00%
Intensively thinking: Avrg Level=1.39, No Reaction=76.67%, High=7.50%, Mid.=5.00%, Low=10.83%
Imagination activity: Avrg Level=1.11, No Reaction=66.67%, High=3.33%, Mid.=8.33%, Low=21.67%
Hesitation: Avrg Level=16.31, No Reaction=0.00%, High=100.00%, Mid.=0.00%, Low=0.00%
Uncertainty: Avrg Level=8.48, Normal Reaction (Level15)=5.83%, High(>20)=58.33%, Mid.(10-20 | | 15)=8.33%, Low (<10)=27.50%
Excitement: Avrg Level=18.53, Normal Reaction (Level15)=0.00%, High(>20)=100.00%, Mid.(10-20 | | 15)=0.00%, Low (<10)=0.00%
Concentration level: Avrg Level=5.37, No Reaction=41.67%, High=29.17%, Mid.=10.00%, Low=19.17%
Atmosphere: Avrg Level=-3.99, Normal Reaction (Level -20 to 20)=99.17%, High(>20)=0.83%, Low (< -20)=0.00%

10.2. Summary Report

The QA5 summary report shows the development (changes) in selected, most relevant emotions and **basic parameters** over time, from the beginning to the end of the call, together with various average levels of a selected group of emotions and unique time domain indications.
Following is a typical QA5 Summary report:

QA5 data:

File contains 120 segments	Number of voice segments analyzed "OK"
EMO. Change Level: 252	The EMOTIONAL activity change from beginning to end. As you can see, the emotional level is higher at the end, indicating a more emotional state and contentment.
COG. Change Level: -113	The COGNITIVE activity change

	from beginning to end. As you can see, the cognitive level is lower at the end, indicating the tested party is MORE CERTAIN about what he said.
STRESS Change Level: -14	Stress was reduced at the end in comparison to the beginning.
CHL Change Level: 43	CHL is one of the main parameters for anger. The positive value here indicates anger rising through the call.
FMain Change Level: -15	Fmain is the basic parameter relating to concentration. The big drop of concentration here is an alarming sign.
Ant. Change Level: 3	Anticipation level change. A slight raise may indicate a still optimistic view.
Average EMO. Level: 741	The average emotional level (basic parameter scale, the higher, the more excited the tested party is).
Average COG. Level: 114	The average cognitive level (basic parameter scale). The higher the value, the more confused the person is, the more efforts he is investing in logically trying to solve his confusion.
Average STRESS Level: 11	The average stress level (basic parameter scale, the higher, the more stressed the tested party is).
Extreme STRESS Segments: 2	The number of segments where extreme stress was detected (indicating a serious problem or issue).
Extreme STRESS conversation portions: 0	The number of incidents where extreme stress was detected in few consecutive segments, indicating a seriously problematic issue.
Extreme EMOTION Segments: 47	The number of segments where extreme emotion was detected, indicating high levels of anger or excitement.

11. QA5 COM SDK Components and Functions

11.1. nmsQA5core Class

The nmsQA5core class can run in the two following modes of operation:
*(Changes from earlier versions are marked **New in Ver. 5.5**)*

11.1.1. nmsQA5core in Server Mode

(Note – this mode is not to perform analysis!)

***nmsSRV_INIT**(cOwner As String) As Integer*
This is the License Server initialization function, and must be the first instance of the QA5 Core COM created. It receives a String cOwner, used to mark your company as the owner of the data in a new LioNet™ instance if created inside the COM. It returns an Integer return-code indicating the completion of the task or an indication of the error which occurred during the task (see list of constant definitions and source code examples).

***nmsSRV_GetActivationLicense**(tProcessId As Long) As String*
This function is used for any new analysis instance of the QA5 Core. It receives the newly created analysis process internal Process ID (see below) and returns a String that is either the Activation Code for the process or a string formatted Error Return code.

***nmsSRV_GetOpDetails**(CallCounter As Long, sysID As String, PostsLicensed As Integer, runningProcesses as long) As Integer*
This function is used for maintenance and does not receive any input. Outputs are as follows:

Call Counter: indicates the number of calls processed by the system to date. In most QA5 systems, this number relates to tests conduct during the month.

System ID: a unique number identifying the QA5 SDK plug. This number is required for re-licensing and configuration of the system by a re-licensing code generated by Nemesysco.

PostsLicensed: The number of concurrent **posts** (instances) registered for this SDK kit.

RunningProcesses: The number of **currently running** instances of the core.

***nmsSRV_ResetCounter**(newLicenseCode As String) As Long*
This is a maintenance function, designed to remotely configure and re-license the system for paid conversations (if applicable) and reset the internal calls counter. It receives a code (String) generated per system/per re-licensing request by Nemesysco's support center, and returns a Long type return value indicating successful completion of the task or an error number.

11.1.2. nmsQA5core in Analysis Mode

The analysis mode functions are divided into several groups, designed for each analysis sub-system.

General Purpose and Main Functions

nmsSDKgetProcessId*() As Long*

Retrieves the Process ID of the analysis process as a Long (signed 4 byte) variable.

nmsInitCore*(ActCode As String, LioNetfile As String, OwnerID As String) As Long*

Initialize the Instance for analysis use. It receives the following:

ActCode: Unique activation code string generated by the licensing server instance.

LioNetFile: Name of LioNet™ data file to be used for segment analysis

OwnerID: String containing your company's name, used to mark a new LioNet™ data file if created inside the object.

This function returns a long variable indicating either successful completion or an error code.

nmsConfigTestData*(voiceSmpRate As Integer, BG_Level As Integer, LengthOfSegmentSec As Integer, CalibrationType As Integer) As Long*

Configure the core for the specific type of audio sampling rate and expected background noise level, as well as for the test type before beginning voice data streaming. This function cannot be used once data is already passed to this instance. This function receives the following inputs:

voiceSmpRate: Input stream voice sample rate, either 8Khz or 11Khz (8000, 11000 or 11025 are acceptable values)

BG_Level: Default optimal background noise level to be used - any positive number in the range of up to 32,000 is acceptable, however, in reality, this number should be approximately 1000-2000.

LengthOfSegmentSec: Your choice of analysis architecture with regard to the length of the segment in seconds (1, 2, and 3 are acceptable values)

CalibrationType: Your choice of calibration type and one of the 3 analysis modes:

0 for full calibration, 1 for short calibration in analysis mode 1;

2 for full calibration, 3 for short calibration in analysis mode 2;

*4 for full calibration, 5 for short calibration in analysis mode 3. (***New in Ver. 5.5***)*

Different systems may produce better results using different analysis mode depending on the used hardware and codecs. It is impossible to know in advance what the optimal analysis mode is without conducting some experiments, but the default setting (3) is typically the most suitable one.

This function returns a long variable indicating either successful completion or an error code.

nmsProcessBuffer*(InpBuf() As Integer, BufSize As Integer, EmoValsArray() As Integer, Aires As String, bStr As String, testbuf() As Integer, TestBufLeng As Long, BrderS As Double) As Long*

This is the main analysis function, receiving the streaming voice data (in buffers of preferably 20 milliseconds of voice), and performing internally the entire analysis process, starting from detection of silence and the segmentation of voice buffers for analysis, through the analysis of the basic parameters and the generation of the final outputs (set of Calculated Emotions).

This function receives the following inputs:

InpBuf() As Integer	Streaming buffer voice data, in a signed integer (16 bit) format. **Only MONO LINEAR PCM format is permitted**.
BufSize As Integer	Size of the buffer. Typically and preferably, this value will equal the number of voice samples in a 20 millisecond voice buffer, and depending on the sampling rate, i.e, in 8Khz, sampling it should be 160.
BrderS As Double	This is actually a unique input, as it is also a return value: If set to 0 (zero) prior to the process, it will force a "Segment Border Analysis" and test the segment against the "Normal Envelope". If set to 1 (one) prior to the process, it will force a "Segment Border Analysis" and test the segment against the "General Borders". In both cases, the **BrderS** return value will be the result of the border calculation. If set to any other number prior to the process, Borders analysis will not be performed.

The function will return the following Long type Return Code that can be:

(-1) NMS_PROCESS_ANALYSISFAILED	The Core could not complete the analysis operation, due to internal analysis conflict or an error.
(0) NMS_PROCESS_SILENCE	Silence was detected in the processed buffer. You can actually make use of the Core to detect long periods of silence.
(1) NMS_PROCESS_VOICEDETECTED	The Core detected voice in the streaming data and is now collecting voice data for the analysis of the voice segment, however, analysis is not yet available.
(2) NMS_PROCESS_ANALYSISREADY	The Core completed the voice segment collection, analyzed it, and the results of the segment analysis are now available in the following parameters: *EmoValsArray() As Integer* returns the values of the calculated emotions.

This array must be defined as a flexible array (dynamic allocation), as it will be redefined inside the Core. The elements values are as follows (in this version of the SDK):

EmoValsArray(0) EmoVals.angry
EmoValsArray(1) EmoVals.Atmos
EmoValsArray(2) EmoVals.concentration_level
EmoValsArray(3) EmoVals.embarrassment
EmoValsArray(4) EmoVals.excitement
EmoValsArray(5) EmoVals.hesitation
EmoValsArray(6) EmoVals.imagination_activity
EmoValsArray(7) EmoVals.intensive_thinking
EmoValsArray(8) EmoVals.content
EmoValsArray(9) EmoVals.saf
EmoValsArray(10) EmoVals.upset
EmoValsArray(11) EmoVals.extremeState
EmoValsArray(12) EmoVals.stress
EmoValsArray(13) EmoVals.uncertainty
EmoValsArray(14) EmoVals.Energy
EmoValsArray(15) EmoVals.BrainPower
EmoValsArray(16) EmoVals.EmoCogRatio
EmoValsArray(17) the highest amplitude detected in the speech signal
EmoValsArray(18) the calculated voice energy (Not emotional energy) detected in the speech signal. *(***New in Ver. 5.5***)*

Alres As String	Returns the internal LioNet™ "User-Defined" emotion analysis for the segment.
bStr As String	Returns a string containing all of the Basic Parameters. This string can be used to further train the internal "LioNet™" to classify the user-defined emotions.
testbuf() As Integer	Contains the actual complete voice segment that was analyzed
TestBufLeng As Long	The length (in samples) of the *testbuf* array.
BrderS As Double	Returns, if defined prior to the process (see above), the accumulative "distance" of the current set of Basic Parameters from the Envelope or Borders loaded to the system (see chapter 8.3, "Borders and Envelope" analysis sub-system above, and "Borders and Envelope" analysis sub-system functions, below).

11.1.3. "Borders and Envelope" Analysis Sub-System functions

nmsBordersSetConfig*(bActive As Boolean) As Integer*
Activate/de-activate the Borders check analysis system. (bActive as True/False respectively). It is important to realize that as any process, the borders sub-system consumes some system resources and to some degree, will slow the analysis process.

nmsBordersLoad*(BorderFileName As String) As Integer*
Loads a Borders and Envelope data file. If the file name is invalid or another error occurred, the system will return an integer (signed 2 bytes) WINDOWS or QA5 SDK error code.

nmsBordersCalculate*(resMap() As Double) As Integer*
Compares the current call data stored inside the analysis object with the loaded Border File data, using the nmsBordersLoad function. Use this function at the end of the call.

Following the analysis, the resMap array will contain a table of 11 X 51 Double Precision type parameters.
NOTE: resMap definition should be flexible (dynamic allocation), as the resMap array will be redefined in the Core.
The Y dimension (0 to 50) represents each of the Basic Parameters.

X Value	Description	X Value	Description
0	Minimum – lowest value detected in this call (low border)	5	Overall distance from the borders in %
1	Maximum value (high border) in this call	6	Distance from the low border in %
2	Average value in this call	7	Distance from the high border in %
3	Normalized Min. (low envelope) in this call	8	Overall distance from the internal envelope in %
4	Normalized Max (high envelope) in this call	9	Distance from the low envelope in %
		10	Distance from the high envelope in %

nmsBordersUpdateWithCurrent*(BorderFileName As String) As Integer*
Accepts and updates the loaded Border File with the current call data, and if the BorderFileName variable is not blank, the modified borders will be saved into the specified file. If you attempt to use this function before any voice

segment was analyzed or logged in the Borders sub-system, the function will generate the MS_SDKERROR_OPERATIONNOTALLOWED return code.

nmsBordersSave*(BorderFileName As String) As Integer*

Saves a copy of the currently active Border data to the specified file. If the file name is invalid or another error occurred, the system will return a WINDOWS or QA5 SDK error code.

11.1.4. QA5 Core Internal LioNet™ sub-system

It is important to separate between the QA5CORE LioNet calls and the separate LioNet object. While you can create external LioNet classes for different needs, the internal LioNet subsystem will only analyze the emotional responses of the QA5 Core.

nmslioNetCreate*(LioNetfile As String, nInputs As Long, OwnerID As String) As Long*

Resets the **internal** LioNet Data used for the Segment level analysis and creates a new LioNet data set using the following inputs:

LioNetfile is the destination file name for the newly created data set.

NInputs is the number of expected input data elements.

OwnerID is your company name, tagging the ownership of the information obtained in the LioNet file. The *OwnerId* can only be set in the creation of the LioNet File.

The function will return a standard WINDOWS error code if an error occurred during the operation (typically due to file names and saving the newly created file).

nmslioNetLoad *(lionetFilename As String) As Long*

Loads a LioNet Knowledge (data) file to the memory. If the file name is invalid or another error has occurred, the system will return a WINDOWS or QA5 SDK error code.

nmslioNetForget *(ForgetString As String) As Long*

Makes LioNet "forget" the specified ForgetString which must be exactly the same as the wrong training string previously used. If the system could not find the training string in its memory, nmslioNetForget will return a notification error code.

nmslioNetGo *(LionetString As String, LionetResponse As String, Dist As Double, Risk As Double, Prob As Double, OutR As Double, cResAccuracy as integer) As Long* ***(New in Ver. 5.5)***

This is the main operation function of the LioNet. It is used to process the LionetString data for both analysis and training. A correct format of a data string for analysis may appear as follows:
"1 2 13.5 4.4 20000 0" *(a string of numbers, separated by a space)*
Training strings (sessions) are in the same identical structure as the analysis string, with the addition of: " =<my_analysis>" (i.e., space and an equal sign immediately followed by the desired analysis) at the end of the string.
A correct format of a data string for training may be as follows:
"1 2 13.5 4.4 20000 0 =AnalysisA"
NOTE: The "answer" must not contain spaces or "=" marks of its own.
The analysis (LioNet determination) or training feedback will be returned in the LionetResponse string. The function will also return several other parameters (i.e., Dist, Risk, Prob and OutR) whose definitions exceed the scope of this document.
The new cResAccuracy will indicate in percentages how certain LioNet is with this specific answer.
The Return Code will indicate a successful completion or an error code.
NMS_LIOERROR will be returned when attempting to train or analyze a data string in an empty LioNet (not created or not loaded).

nmsLioNetSave*(ErrMsg As String) As Long* **(New in Ver. 5.5)**
Saves the LioNet data to the LioNet knowledge file.
The Return Code will indicate a successful completion or a windows standard error code.

11.1.5. QA5 general QA subsystem, Logging and "Conversation Emotional Signature" Functions:

nmsQA_ConfigUse*() As Long*
Configures the Core to collect data during the conversation to produce the Emotional Signature. This function requires no inputs, and will return a return code indicating successful completion or an error code.

nmsQA_Logdata*(SegmentID As Long, cStartPosSec As Long, cEndPosSec As Long) As Long*
This functions instructs the Core to log the details of the last-processed voice segment in the internal QA5 array, required for the creation of the Emotional Signature.

SegmentID	Serial number of the segment in the file, i.e., 1, 2, 3, etc…
cStartPosSec	Start time stamp point of the segment, calculated in your managing process controlling the call, described in tenths of a second from the beginning of the call. For example, 1020 is actually 10.20 seconds from the beginning of the file or call.

CEndPosSec	End time stamp point of the segment, calculated in your process controlling the call, described in tenths of a second.

Return codes of this function can be:
NMS_SDKERROR_OPERATIONNOTALLOWED or NMS_OK.

nmsQA_CreateSignature*(inNoSegments As Long, NoStresses As Long, m_ThisAVJstress As Double, m_ThisAVJemo As Double, TestRep As String) As String*

Receives the inNoSegments variable (containing the total number of segments detected in the call), and creates the conversation Emotional Signature in the return code (string), together with several other emotional indications, as follows:

NoStresse	Counter of highly stressed segments detected in the call.
M_ThisAVJstress	Average stress level detected in the call. This average is based on the Basic Stress Parameter, and ranges from 1 to 100. (High values are above 30!)
M_ThisAVJemo	Average emotional level detected in the call, based on the Basic Emotional Parameter, and ranges from 1 to 2000. (High values are above 900)
TestRep	String variable containing the entire QA5 Test Report and details.

NOTE: Other emotional summaries (as described in Section 8, Call Profiler and QA5 Summary Reports) can be extracted from this report using simple string manipulation functions.

New functions in Ver. 5.5:

nmsQA_getProfilerData(*emoLevel As Integer, logicalLevel As Integer, hasitantLevel As Integer, stressLevel As Integer, energeticLevel As Integer, thinkingLevel As Integer) As Integer*

This function can be used on both agent and customer sides to gather information on their current emotional profile as described by the 6 emotional parameters and captured over the first 6 voice samples of the call. The Emotional Profile can, for example, provide a better insight to the way a sales pitch may be taken.
All the emotional levels returned from this function are normalized to a scale between 0 to 100.
The function will return NMS_OK if enough data was captured to generate the profile, or NMS_FAIL if not enough data was captured yet. See the sample application to get more ideas of possible utilizations of this function.

nmsQA_getChangesFromProfiler(*emoLevel As Integer, logicalLevel As Integer, hasitantLevel As Integer, stressLevel As Integer, energeticLevel As Integer, thinkingLevel As Integer) As Integer*

Once the Emotional Profile was generated, use this function to gather information about the current emotional profile and changes thereof from the initial profile. All the emotional levels returned from this function are normalized to a scale between 0 to 100 as well.

nmsQA_InitalizeCallPriority*(cStressWarnLevel As Integer, cConstHighEnergy As Integer)*

Instructs the QA5 Core to prepare to log and process the collection of the Call Priority Score emotion related parameters. The Call Priority flag is a summary value designed mainly for real-time analysis to indicate how relevant it is for the managers to intervene. The Call Priority value will raise if unresolved stress, anger or upset are detected over time, and therefore it is more suitable for the analysis of the Customer side.

nmsQA_getCallPriority*() As Integer*

Processes the last voice segment data and returns the current Call Priority level.

To get the most accurate reading of the call priority this function should be called after each voice segment is detected. Call Priority values range from 0 to 100, the higher the more urgent intervention is required.

nmsQA_getMaxCallPriority*() As Integer*

Retrieves the highest Call Priority level detected in the call. Different utilizations can be made comparing this value to the final Call Priority level, indicative if problems detected during the call has been resolved or not, and to what extent.

nmsQA_CollectAgentScoreData*() As Long*

Instructs the QA5 Core to log and process the collection of the Agent Score related parameters in the last analyzed segment. The Agent Score is a summary function of several related emotional parameters that are most relevant for the agent's side performance analysis.

This function returns 0 during the first few voice segments, and then returns a value representing the changes in the relevant emotional parameters of the tested party's Emotional Profile. The higher the value is, the sharper the change is.

nmsQA_GetAgentScore*() As Integer*

Once the conversation is over, use this function to retrieve the final Agent's Score.

The higher the score is, the LESS acceptable was the Agent's performance. The Agent's Score ranges from 0 to a theoretical 100, but any value above 30 is worthy of your attention.

11.1.6. Stress Detection Sub-System:

nmsSD_ConfigStressCL *() As Integer*

Configures the Core and prepares to collect the Clinical Stress data during the conversation.
Return codes of this function can be
NMS_SDKERROR_OPERATIONNOTALLOWED or NMS_OK.

nmsSD_LogData *() As Integer*

Used following each voice segment analysis. Logs the Clinical Stress data per segment in the Core memory. This function returns NMS_OK, or a WINDOWS error code.

nmsSD_GenerateReport *(clStressLow As Long, clStressHigh As Long) As Integer*

Generates the Stress Report and Warning Level return value. The meaning of the values is described in Section 7.3, "Stress Analysis".
The return code is the warning level based on the stress levels, and is one of the following:

NMS_SD_NOSTRESS (0)	No clinical stress was detected. Note: you may find this indication in angry calls. This is not an analysis error; anger is a way the brain uses to dissolve stress.
NMS_SD_LOW (1)	Some low levels of stress were detected, but there is no need to be alerted.
NMS_SD_MID_TEMP_OK (2)	Although relatively high levels of stress were detected from time to time, there is still no reason to be alerted, as the tested party's clStressLow level is still considered low.
NMS_SD_HIGH_TEMP_OK (3)	Although relatively high levels of stress were detected often, there is still no reason to be alerted, as the tested party clStressLow level is still considered low.
NMS_SD_MID_WARN (4)	The system detected a relatively high level of stress from time to time, and the tested party does not seem to be able to relax properly. This is the time to take an affirmative action.
NMS_SD_HIGH_WARN (5)	The system detected a high ratio of highly stressful segments, and the tested party

	does not seem to be able to relax. This is a very high stress level and should be treated accordingly.
NMS_SD_DANGERLEVEL (6)	The system detected a dangerous level of stress that may soon begin to show symptoms if not already done so. This is the time to take action to relieve the stress or assist the tested party now.

Please note: This function monitors ONLY the Stress levels, and not any other emotion. It is possible that the tested parties will feel angry or extremely excited (which may also be dangerous for their health), and yet the system will return with the NO STRESS indication. Stress is a very specific emotion, and in fact, anger is the opposite psychological reaction to stress.

11.1.7. Call Profiler Sub-System:

nmsCollectProfiler*()* As Integer
Instructs the Profiler sub-system to add the last analyzed segment data to the profiler internal memory.
This function returns NMS_OK or NMS_SDKERROR in cases of data fault (very rare though!).

nmsCallProfiler*(SegCount As Long) As String*
Generates the Call Profiler Report.
SegCount returns the number of collected voice samples for profiling.
The return string is the Profiler Report. If no segments were collected, the report will be empty.

11.2. Using the Trend Analysis class

In most cases, there is little importance to the emotional reaction in one or few voice segments. Therefore, it is more important to understand and predict where a "conversation is heading" before raising an alarm to the manger, even if some segments show signs of negative emotions. In most cases, the agents are capable of solving the problems that may arise in a conversation by themselves, without further assistance, but once the entire conversation is taking a negative direction – the supervisor should be notified.
The trend analyzer is a class designed to ease the calculations of the emotional development trend or development by managing internally the history and solving the trend/rotating averages formula.
This class is provided for your convenience, but can be easily replaced by any other similar in nature function set of your choice.

11.3. nmsLioNetV6 Class (New version of LioNet in QA5.5)

11.3.1. The LioNet Principle

LioNet was designed to respond to a need for research assistance. When in its initial phase, no clear logic or any defined purpose is established. LioNet will receive a vector of numbers in various ranges, formatted as a space (" ") delimited string (to avoid confusion with languages in which a comma (",") is used as the decimal point).

The system uses "Basic Clusters" which are the information bins created each time the system learns something new, and a "Prime Cluster" for any new possible response. "Hidden Clusters" are created, merged and destroyed based on the accumulative knowledge and common criteria between "Basic Clusters". On top of these actions, LioNetV6 introduces an new type of "conclusion beans" called "Smart Clusters" which are in fact different mathematical formulas automatically created by LioNet, further assisting different types of uses. When going to "sleep", LioNet is attempting to fine tune its Smart Clusters and this process may take few seconds. While analyzing an input string, LioNet uses 13 different decision engines that can technically reach different decisions for the same input. Therefore, LioNet also keeps track of the decision engine that was most useful and accurate, and will use it for its final analysis.

The first 6 training sessions will be used by LioNet to establish its internal logic bases, calculate and normalize the input ranges. No analysis will be generated until the system is trained with at least 6 items. Therefore, although not a must, it is preferred to first train the system with various types of desired answers.

After this "incubation" phase, once LioNet receives any type of input (training or inquiry), it will immediately test it against its own knowledge base and try to reach a decision. If the input was designated for inquiry, LioNet will return its decision.

If the input was designated for training, LioNet will match its result with the training answer, and if found to be correct, LioNet will increase its level of "Confidence" in the internally selected decision engine. Overall accuracy reported by the system will be increased as well. If the System was wrong, it will create a new "Basic Cluster" that will be further used in the Sleeping Process and the conclusion-reaching phase, and its accuracy level will be temporarily reduced.

In any case, the input string will also be recorded in the internal "Long-term memory", and from time to time, during a "Deep Sleep" process, these memory records will be tested again in the system to ensure new data did not affect previously recorded good results.

Old versions users, please review this chapter carefully, as LioNet V6 interface has several changes and will require few code modifications.

11.3.2. Creating, Loading and Saving LioNet™ Networks

nmslioNetCreate*(ByVal NetName As String, ByVal NetInps As Long, ByVal Owner As String) As Long*

Resets the internal LioNet™ Data and creates a new LioNet data set using the following inputs:

NetName	Destination file name for the newly created data set.
NetInps	Number of expected input data elements.
Owner	Your company name, tagging the ownership of the information contained in the LioNet file. The *Owner* can only be set in the creation of the LioNet File.

The function will return a standard WINDOWS error code if an error occurred during the operation (typically due to file names and saving the newly created file).

nmslioLoadNet(*ByVal FileName As String, ByRef outMsg As String) As Long*

Loads a LioNet™ Knowledge (data) file to the memory. If the file name is invalid or another error occurred, the system will return a WINDOWS or QA5 SDK error code.

filename	Pointer to the file to be loaded.
OutMsg	Return value (String) containing a textual message in case of an error or the message "Net Loaded", if successful.

The DoSleep parameter from older versions is no longer being used.
The function will return NMS_OK or a standard Windows Error code if the operation failed.

nmslioSaveNet*(ByVal FileName As String, ByRef outMsg As String) As Long*

Saves the LioNet data to the file set in *fileName*.
outMsg returns a textual message in case of an error or "Net Saved OK" if the operation was successful. The DoSleep parameter from older versions is no longer being used.
The function will return NMS_OK (0) or a standard Windows Error code if the operation failed.

nmslioKillNet()

This call is no longer being supported and was removed from the interface in LioNet V6.

11.3.3. Training and Inquiring LioNet

nmslioNetGo*(ByVal inpString As String, ByRef OutResult As String, ByRef OutDist As Double, ByRef outProb As Double, ByRef outRisk As Double, ByRef OutOfRange As Double, ByRef cResAccuracy As Integer) As Long*

(Version updates: new output parameter and new return codes used in this function)

This is the main operation function of the LioNet. It is used to process the InpString data for both analysis and training.

A correct format of a data string for analysis is as follows:
"1 2 13.5 4.4 20000 0" (a string of numbers, separated by a space)
In this case, the LioNet response (answer) will be provided in the OutResult value (String type), and other information pertaining the query will be provided in the outDist ("Geometrical distance"), outProb (Answer probability), outRisk (The risk level associated with the answer), OutOfRange (indicating if the input string values are known to LioNet or if any parameter was extreme), and the cResAccuracy – indicating the expected accuracy of the specific type of answers based on the accumulative data LioNet has.

Training strings (sessions) are structured in the same identical structure as the analysis string, with the addition of: "=<my_analysis>" (i.e., space and an equal sign immediately followed by the desired analysis) at the end of the string.
Following is a correct format of a data string for training:
"1 2 13.5 4.4 20000 0 =AnalysisA"
(NOTE: "Answer" must not contain spaces (" ") or ("=") marks of its own).
In case of training, the system's feedback will be returned in the OutResult string. The other parameters (i.e., Dist, Risk, Prob and OutR) are not used in this case.
The Return Code will indicate one of the followings:

- **NMS_LIO_TRAIN_LEARN** – in case the system didn't know the answer and has processed it.
- **NMS_LIO_TRAIN_KNEW** – in case the system knew the answer prior to the training session.
- **NMS_LIO_TRAIN_REJECT** – in case LioNet does not agree with the training and rejects the training session. This case will only occur if a very similar data was already provided to the system with a different answer. **To force LioNet to accept the new definition, train it again on the same data string**.

NMS_LIOERROR will be returned when trying to train or analyze a data string in an empty LioNet™ (not created or not loaded).

LioNet V6 further utilizes this function call for other instructions to LioNet. Please see the section "Advanced uses" below.

Using "Tags" and "Basic Conditions" with nmslioNetGo function calls:

TAGS: LioNet keeps a record of trained responses so it would be possible to explain in certain situations why it reached a certain decision. A TAG may be used to instruct LioNet to keep some details attached to its Basic cluster (if created). A TAG can contain a specific name or any other information that

will help you identify the specific case. To add a TAG to your data string, simple add "/T'<MY tag>" to the inpString. (You can include spaces inside the tag, but remember to add the "'" sign at the beginning and the end of it.)

Basic Conditions are used when the same data may need to be processed differently according to some other identified situations, for example, voice input may be processed differently for men and women. In these cases, you can either create a different LioNet file, or simply add a Basic Condition to instruct LioNet to use only the data relevant to that Basic Condition. To add a basic condition to your inpString, use the following context to add the "/B" tag:
inpString = "/B<conditionName> " + inpString. (No spaces allowed in /B tags!)

A complete inpString making use of both Tags and Basic Conditions may look like this:
inpString = "/Bmale 100 12 13.5 14.7 0.003 12 /T'case x' =MyAnswer"

nmslioNetForget*(InpString As String) As Long* ***(New in Ver. 5.5)***
Makes LioNet forget the specified InpString, which must be exactly the same as the wrong training string previously used. If the system could not find the training string in its memory, an error will occur.

nmslioGetAllAnswers*(ByRef outRresults() As nmsLioResults) As String*
***(New in Ver. 5.5** - Replaces the old **nmslioGetAllResDet**(Details As String) function)*
This function returns the complete set of results from all the 13 internal decision engines for the last analysis.

nmslioNetSleep*() As String*
***(New in Ver. 5.5** - This function was modified: Changed return type and outputs)*
The sleep call will initiate an optimization process on the internal mathematical engine, and will cause it to look for the optimal function and associated ranges that will produce the most accurate results. This process cannot be interrupted until completed, and depending on the complexity of the LioNet data set, it may take some time to complete. The function will return a string type report summarizing the optimization process and its achievements.

nmslioTakeDeepSleep()
***(New in Ver. 5.5** - Replaces the older nmslioDeepSleep() function)*
Instructs LioNet to process again its conclusion database while it also re-analyzes data stored in its "Long Term Memory". Please note, the Deep Sleep process may take significant amount of time, but can be terminated quickly by using the "nmslioWakeUp" call, or by sending a new data string to nmslioNetGo function. No input or output is used or created in this call.

nmslioWakeUp() **(New in Ver. 5.5)**
Terminates the Deep Sleep process. No input or output is used or created in this call.

nmslioIsNetReady*(sNetFileName As String, sNetReport As String) As Single*
This function returns the following:

The internally determined accuracy level of the system, based on user feedback and last 100 training sessions. (NOTE: Therefore, it is important to "train" the system even when the system was accurate in its analysis!).
sNetFileName will return the current file name used for the LioNet™ database
sNetReport will return a short summary of the system's current state

nmslioGenerateFullReport(ToFileName As String) As Long **(New in Ver. 5.5)**
This function will create a full report detailing all the data gathered by LioNet and some its internal logic. The function will save the report to a designated file name by ToFileName variable, but in case the ToFileName variable is left empty (""), it will receive the output result that contains the whole report so it can be immediately accessible by your application.
The Long return code will be 0 or a standard windows error code.

nmslioGetNetDetails() As lioNetInfo **(New in Ver. 5.5)**
This function returns the net's static and statistical information using the lioNetInfo structure. (See the structures definitions below.

nmslioGetCurrentStatus() As String **(New in Ver. 5.5)**
This function returns a short string type summary containing basic information of the net's static and statistical information.

nmslioSetParmsNames(parmNamesString As String) **(New in Ver. 5.5)**
To simplify the report generation and understanding, you can instruct LioNet to identify the string of parameters by their actual "names", instead of the default "Parm1, Parm2…" names assigned to the parameters during the net creation.
parmNamesString is a SPACE delimited string containing the names to be used ("ParmX ParmY..")

nmslioGetOwner*() As String*
This function is no longer supported. Please see the function ***nmslioGetNetDetails*** for alternative method of obtaining the owner name.

nmslioGetIstBrdChk() *As Double*
This function is no longer supported.

12. Definitions, Structures and Constants

12.1. EmoVals Definition

Emovals structure contains all the segment analysis data. This structure is defined in the VB6 source code sample as follows:

```
Public Type EmotionRes
        content As Integer
        upset As Integer
        angry As Integer
        stress As Integer
        embarrassment As Integer
        intensive_thinking As Integer
        imagination_activity As Integer
        hesitation As Integer
        uncertainty As Integer
        excitement As Integer
        concentration_level As Integer
        extremeState As Integer
        saf As Integer
        Atmos As Integer
        Energy As Integer
        BrainPower As Integer
        EmoCogRatio As Integer
        bCheck As Long
        maxAmpVol As Integer
        VoiceEnergy As Integer      ' NEW

        Alres As String End Type
End Type
```

12.2. nmsLioResults structure

The nmsLioResults contains the whole analysis results and statistical information about the provided answer.

```
Public Type nmsLioResults
   Answer As String
   Power As Double
   Distance As Double
   Risk As Double
   Probability As Single
   ClusterID As Long
   Tag As String
   typeSpecific As String
End Type
```

12.3. LioNetInfo structure

The LioNetInfo structure contains all the LioNet internal definitions and analysis models statistical accuracy, as well as other variables that may be of importance in different times.

```
Public Type lioNetInfo
   NetName As String
   NetInps As Long
   NGain As Double
   NReject As Double
   NSpread As Integer
   NHistroy As Long
   NPower As Double
   NGood As Long
   NBad As Long
   rejectionCounter As Long
   UsedBasic As Long
   UsedHidden As Long
   UsedPrime As Long
   UsedSmart As Long
   LogScore As String
   UsingMethod() As Long
   FavorModel As Integer
   parmNames() As String
   Owner As String
   ID As String
   version As String
End Type
```

12.4. Constants and Return Codes

The following constants are defined in the VB6 sample code:

General purpose codes
Public Const NMS_OK = 0
Public Const NMS_FAILED = -1

Segment analysis return codes
Public Const NMS_PROCESS_SILENCE = 0
Public Const NMS_PROCESS_VOICEDETECTED = 1
Public Const NMS_PROCESS_ANALYSISREADY = 2

General SDK error codes
Public Const NMS_SDKERROR = -100
Public Const NMS_SDKERROR_FILENOTFOUND = -101
Public Const NMS_SDKERROR_PROTECTIONERROR = -102
Public Const NMS_SDKERROR_WAVESMPRATEWRONG = -103
Public Const NMS_SDKERROR_OPERATIONNOTALLOWED = -104
Public Const NMS_SDKERROR_UNSPECIFIED = -110

Licensing server warning and error codes
Public Const NMS_SDK_LICENSERENEWNEEDED = 1
Public Const NMS_SDKERROR_LICENSERENEWNEEDEDNOW = -105

Border Modification return code notice
Public Const NMS_SDK_BORDERSMODIFIED = 1
Public Const NMS_SDK_BORDERFILE_INIT = 1

LioNet Return and Error codes
Const NMS_LIO_TRAIN_LEARN = 0
Const NMS_LIO_TRAIN_KNEW = 1
Const NMS_LIO_TRAIN_REJECT = 2

Public Const NMS_LIOERROR = -200
Public Const NMS_LIOERROR_FILENOTFOUND = -201
Public Const NMS_LIOERROR_PROTECTIONERROR = -202
Public Const NMS_LIOERROR_FORGETFAILED1 = -203
Public Const NMS_LIOERROR_FORGETNOBASICFOUND = -204
Public Const NMS_LIOERROR_CANTMAKEFILE = -205
Public Const NMS_LIOERROR_UNSPECIFIED = -210
Public Const NMS_LIONET_CREATED_OK = 1

Stress levels and warnings
Public Const NMS_SD_NOSTRESS = 0
Public Const NMS_SD_LOW = 1

Public Const NMS_SD_MID_TEMP_OK = 2
Public Const NMS_SD_HIGH_TEMP_OK = 3
Public Const NMS_SD_MID_WARN = 4
Public Const NMS_SD_HIGH_WARN = 5
Public Const NMS_SD_DANGERLEVEL = 6