User Manual
POLQA OEM Library

Version 3.3

20 March 2019

1 Contents

2 Preface

The OPTICOM POLQA OEM library is the latest out of a family of products for speech quality assessment. It allows you to implement POLQA fully conforming to the ITU –T P.863 standard with an optimum time to market.

Libraries for various measurement algorithms are available from OPTICOM. All libraries are optimized for best performance, speed and accuracy. Using these libraries will save you months of development work and guarantee a successful implementation.

Since OPTICOM is the leading technology provider for voice, audio and video quality tools we feel obliged to push the development further. We will provide more algorithms in the future, as well as improved versions of the existing tools.

2.1 Abbreviations

This manual uses the following abbreviations:

ACR	Absolute Category Rating
AMR	Adaptive Multi Rate
API	Application Programming Interface
DCR	Degradation Category Rating
DLL	Dynamic Link Library
dBOV	dB Overload
ERP	Ear Reference Point
EVRC	Enhanced Variable Rate Codec
FR	Full Reference
GSM	Global System for Mobile Communications
IRS	Intermediate Reference System
ITU	International Telecommunication Union
LQ	Listening Quality
MOS	Mean Opinion Score
MOS-LQO	Mean Opinion Score Listening Quality Objective
MOS-TQO	Mean Opinion Score Talking Quality Objective
NR	No Reference

NMR	Noise to Mask ratio
MUSHRA	Multiple Stimuli with Hidden Reference and Anchor
PESQ	Perceptual Evaluation of Speech Quality
PEAQ	Perceptual Evaluation of Audio Quality
PSQM	Perceptual Speech Quality Measure
POLQA	Perceptual Objective Listening Quality Assessment
PSTN	Public Switched Telephone Network
RMSE	Root Mean Squared Error
SDG	Subjective Diff Grade
SNR	Signal to Noise Ratio
SPL	Sound Pressure Level
WAV	Waveform Audio File Format

2.2 Organization of this Manual

In Chapter 4 some terms which are widely used throughout this manual are explained. While most readers are probably familiar with these, there common usage is however not precise enough for using these terms in this manual without giving our interpretation of the meaning. In Chapter 5 we start with an introduction to subjective testing. Since subjective testing is what we want to simulate by perceptual measurement, it is very important to have a sound understanding of several peculiarities of subjective tests. This chapter focuses on the most important points and is by no means a replacement for reading the according standards documents and ITU handbooks. In Chapter 7 we try to give a very broad overview on the various perceptual objective metrics that are commonly used today. Both Chapters 5 and 7 go far beyond the scope of POLQA, but this helps to identify what POLQA is intended for, what can be done with it and where it's principle limits are. Chapter 8 is just a short summary of the previous chapters. Chapter 9 is where we finally start with a detailed discussion of the POLQA algorithm and the results obtained by the OPTICOM POLQA OEM Library. The practical usage of the POLQA OEM Library is explained from Chapter 10 onwards.

3 Legal Notes

This Software is licensed, not sold. Any commercial use of this software is subject to the execution of a POLQA OEM license agreement with OPTICOM, based on terms and conditions to be agreed.

The following statement(s) shall be incorporated by **LICENSEE** into the product as defined under the POLQA License Agreement:

Further statements shall be incorporated to

- prohibit additional copying of the POLQA software in whole or in part, other than is essential for the proper operation of the POLQA software or for normal security back-up purposes;
- prevent the End-User from modifying, translating, reverse-engineering or decompiling the POLQA software except to the extent permitted by law;
- require that the acknowledgement of the rights in the POLQA software shall not be removed from the POLQA software or any installation of it;

4 Principles and Definitions

This chapter contains some general definitions of terms which are used throughout this manual. Most readers will be familiar with these terms, but since some of them are often used synonymously although they describe different aspects, we prefer to define our usage here.

4.1 FR/NR, Active/Passive, Intrusive/Non-intrusive

The terms "intrusive", "Active" and "full reference" measurement as well as their opposites are very often used synonymously. However, they all have their distinct meaning and the ITU is currently working on the according clarification.

Full Reference / No Reference (FR/NR) describe the type of the used measurement algorithm. An FR algorithm will require access to the undistorted reference signal as well as to the degraded signal. An NR algorithm in turn needs access to the degraded signal only. A typical FR algorithm is e.g. PESQ. A typical NR measure is P.563. NR methods are also often referred to as "single ended" measures.

Intrusive and Non-intrusive in turn describes the way in which the data acquisition is performed. Intrusive means that a known signal is injected into the system under test, while non-intrusive means that measurements are performed with whatever content can be detected on e.g. a network. Please note, that FR measurement algorithms can be used non-intrusively as well, if the reference signal is recorded at the input of the system under test, while the degraded signal is recorded at the output of the system under test.

Last but not least, the terms **Active and Passive** describe the way in which the test system establishes the connection to the system under test in order to perform the measurement. A network sniffer for example is a passive system, since it simply spies on the network and measures with whatever data comes along. The same would be true for a voice quality test system which is performing tests on calls that exist already on the line. An active voice quality tester for example would explicitly setup a connection to another party (e.g. dial a number) before starting the data acquisition.

Please note, that most combinations of these three pairs are possible, but not all combinations make sense.

4.2 Perceptual / Non-Perceptual

Perceptual metrics like e.g. POLQA, PESQ or P.563 assess certain effects similar as human beings would do. Typically, such measurement algorithms try to model the human perception. Perceptual modeling is not only used for the assessment of quality. Other famous perceptual algorithms are e.g. MP3 or AAC which use perceptual models for the compression of music. Non-perceptual metrics are general physical or technical measures like e.g. levels or SNR.

4.3 Subjective / Objective

The terms subjective and objective are used to distinguish between the assessment by human beings (subjective) and a technical model of human perception (e.g. a computer program) on the other side (objective). The persons scoring e.g. voice quality in a subjective test are often called subjects, while the algorithm performing a similar assessment in the objective domain is frequently called an objective model or method.

4.4 MOS Scales

In all fields of subjective testing and perceptual measurement **M**ean **O**pinion **S**core (MOS) scales are widely used to report the results. MOS scales are defined e.g. in the ITU recommendations P.800, BS.1116, BS.1534 and others. A MOS scale simply defines a well specified range of numbers that describe the quality as it is assessed by human subjects. A general misconception is however, that these scales are something absolute and context free. In reality, quite the opposite is the case, as explained in the following.

4.5 Context Dependency of MOS Scales

It is generally not possible to reproduce subjective test results with absolute accuracy. This is not only true for two subjective tests conducted by different labs, but also for identical tests conducted by the same lab. The cause for this lies in the multitude of factors that affect our human perception, mostly our expectation and other cognitive effects. An easy to understand example is, if the range of qualities presented in two different experiments covers a different range of degradations. One example could be a first experiment containing 90% very high-quality samples and only 10% poor quality samples and a second experiment containing 10% very high quality and 90% poor quality samples. A small common set of identical test conditions should be contained in both experiments. In this case the scores for the common set achieved in the first experiment will most likely be worse than those achieved in the second experiment, although the subjects were given exactly the same instructions and thought that they were using the same MOS scale for their judgment of quality.

The reason for this effect is that the range of qualities in a subjective experiment also defines the scale applied by the subjects.

> There is no such thing like an absolute MOS scale. MOS values are only valid within the context of the experiment that was conducted to generate them.

Naturally, an objective quality measure does not show such a behavior, since it is context free by

The need therefore arises, to compensate for systematic differences between subjective experiments, before comparing subjective and objective scores. One method to do this is to apply a third order, monotonic polynomial to the objective scores which minimizes the root mean square error (RMSE) or maximizes the correlation between the two data sets. Doing this in the exact and correct way requires some special tools which are not easily available. For the general case however, it is sufficient to draw a chart in MS Excel, add a third order regression line and read the correlation given for the regression line. The biggest risk in this case is that the regression line is not monotonic, but this can be checked visually.

4.6 Voice Quality: Listening Quality, Talking Quality and Conversational Quality

When assessing the quality of telecommunication scenarios, there are different aspects which have to be taken into consideration. So far, the quality was commonly assessed by the quality which a user perceives when he listens to the voice signal. This aspect, known as **listening quality** is a very obvious view onto quality as such and is also a prominent part in the assessment of overall quality. Yet, when looking at the overall quality more closely, one will find that this sole feature will not allow a view onto the whole picture. The reason for this is the following:

Telecommunication systems as opposed to e.g. broadcasting systems are designed for the users to interact with each other, namely to carry out conversations (e.g. a telephone call). So, when looking at the quality only from a listening quality point of view, one reduces a telecommunication scenario to a broadcast-like scenario, in which only one party is uttering speech and the other one is listening. This is of course not reflecting the reality. One therefore needs to also assess further aspects which become important in a conversation and assign a quality index to them. These aspects are the following:

Talking quality: the quality as perceived while talking oneself. The importance of this aspect can be subjectively observed when having a conversation with a peer who is using a mobile or a hands-free phone: The echo which most commonly occurs when such equipment is used can be extremely disruptive, and can even lead to such a high irritation of the talker that the whole conversation has to be abandoned.

Interactional quality: the aspect of how "fluent" the conversation can take place. This is mainly affected by the aspect of delay in the transmission system. Long delays cause interruptions as two parties have difficulties to synchronize the cadence of their respective questions and answers and will tend to overlap.

The overall quality of the conversation (the **conversational quality**) must therefore be seen as a function of the listening quality, the talking quality and the interactional quality. Addressing only a single aspect of this, though giving important insights on its own, will generally not cover the whole picture.

4.7 Classification of MOS Values (P.800.1)

ITU-T Recommendation P.800.1 defines a terminology for MOS values which describes how the MOS value was created and in which context it has to be seen. This terminology consists of suffixes which are added to the term "MOS". The suffix starts with a two-character abbreviation for the listening situation (LQ=listening only, TQ=listening while talking, CQ=conversation), a one-character indicator describing whether the value was derived from a subjective experiment ("S"), and objective measurement ("O") or an estimation using a planning tool like e.g. the E-model ("E"). The final letter ("y" in Table 4-1), describes the context of the experiment, and can have any of the following values:

N for MOS scores obtained for narrowband (300-3400 Hz) voice relative to a narrow-band high quality reference. This is applicable for instance to narrow-band only subjective tests or to P.862.1 scores.

M for MOS scores obtained for narrowband or wideband voice relative to a wideband high-quality reference in a mixed bandwidths context. This is applicable for instance to mixed bandwidths subjective tests.

W for MOS scores obtained for wideband (50-7000 Hz) voice relative to a wideband high-quality reference. This is applicable for instance to wideband only subjective tests or to P.862.2 scores.

S for MOS scores obtained for audio signals up to super-wideband (20-14000 Hz) relative to a super-wideband high-quality reference. This is applicable for instance to super-wideband only subjective tests or to ITU-T P.863 scores.

F for MOS scores obtained for audio signals up to fullband (10-20000 Hz) relative to a fullband high-quality reference.

	Listening-only	Conversational	Talking
Subjective	MOS-LQSy	MOS-CQSy	MOS-TQSy
Objective	MOS-LQOy	MOS-CQOy	MOS-TQOy
Estimated	MOS-LQEy	MOS-CQEy	MOS-TQEy

Table 4-1 MOS terminology according to P.800.1

Certainly, this is not sufficient to describe the entire context within which a MOS value is valid, but at least some of the most confusing details are clarified.

4.8 Quality

4.8.1 Definition

Many definitions of quality have been published in the past, but the one that we like most is:

"Quality is the judgment of the difference between what we perceive and what we expect." [JEK000]

This statement clearly defines the limits of subjective experiments and objective methods. The subjective experiments are always biased by the expectation of the subjects. This is unavoidable and may already be caused by simple things, like e.g. the shape of the handset used for listening. Subjects will expect more distortions from a handset which looks like a mobile than from a handset which looks like a traditional ISDN phone. Objective methods on the other side naturally have no expectation at all. They are not sensitive to the context and they will produce the same score for the same degradations, independent of the shape of the phone.

Which quality statement is correct, the subjective or the objective one is a philosophical question. For engineers and technicians, the answer is given by the objective of the experiment and must be decided on a case by case basis. However, it very is important, that the person performing the assessment is aware of this difference.

4.8.2 What is good Quality?

As outlined in Section 4.8.1, the perceived quality is highly depending on the expectation of the subjects. In real life, this expectation is depending on a multitude of factors, which cannot be foreseen here. One such factor could for example be the price of a service. For a free service the expectation of the users may be low and thus the relatively poor (objective) quality may be sufficient. For an expensive premium service however, the same persons will most likely expect much higher objective quality. The answer to the question of what we define as good quality is therefore not a simple one and definitely not one to be answered by an R&D department and even less by a measurement tool vendor.

One possible scenario for technical teams to find the right answer could be to make recordings of different objective quality and play these to the responsible persons from e.g. marketing. It is then up to the marketing department to take a decision of what quality is really required. Of course, they should have listening examples of competitive products available too. Once the marketing department has made that decision, the technical teams can define the objective quality of the finally chosen file versions as the target quality for the service. In any case one should be warned to use the verbal meanings for the scores of the MOS scale as the reference. These are definitely just anchor points for subjects in the subjective test and also subject to the user's expectation. Also, one must never expect to achieve the ideal results published for a specific codec in real life. Typically, these scores were determined for an isolated codec without a real network.

4.8.3 How can Quality be related to Non-Perceptual Values?

Very often people try to relate a poor MOS score to other, non-perceptual values like e.g. delay, packet loss or attenuation. This may sometimes work, but generally it is not possible, since human perception is working in entirely different dimensions than those rather technical parameters. Very often different artefacts in a test sample emphasise or mask each other. By looking at only one parameter it is impossible to model this behaviour.

5 Subjective Experiments vs. Measurements

Assessments based on human perception can be made in two fundamentally different ways. One possibility is to perform subjective experiments in which a panel of viewers or listeners (=subjects) has to assess e.g. voice quality of some hundred prerecorded voice files. The other method consists of measurements that lead to ideally the same results. Those measurements are often also referred to as objective experiments. Both methods have their pros and cons and it is not possible to completely replace one method by the other. For the assessment of listening quality several subjective and objective methods are standardized. Before performing measurements, it is important to have at least some basic understanding of the subjective test methods since all objective methods are designed to predict the results of specific subjective experiments. Consequently, without knowing the simulated subjective experiment, it is not possible to correctly interpret the measurement results.

Subjective assessments are often regarded as the most accurate method to predict quality based on human perception. The disadvantage is however, that conducting such experiments is very expensive and time consuming. Also, the results are only valid within the context of one experiment which widely controls the expectation of the subjects. No method is more accurate in comparing e.g. two codecs, but both codecs must be compared in the same experiment. Splitting experiments across different labs is sometimes done, but this requires enormous subjective know how and the tests conducted by the different labs must be very well balanced to avoid such effects as mentioned in Section 4.4. Also, the reproducibility of the results is strongly depending on the day to day performance of the subjects as well as the number of subjects used in the experiment. Subjective experiments are widely used today in order to evaluate and select codecs or to assess the accuracy of objective measurement methods. It is generally impossible or at least not practical to use subjective experiments in the field, daily in the lab or for e.g. drive tests.

Objective measurements are certainly not as accurate as subjective experiments, but they are easy and cheap to apply. Very often they are the only feasible way to assess quality. The biggest advantage of objective methods is however, that they always calculate the same result from the same input data, independent of the day, location or anything else. Objective methods are absolutely context free and it is therefore possible to analyze a codec at high bitrates in lab A, at low bitrates in lab B and then to combine the data. The objective results will be valid even though the experiments were completely unbalanced. It is also possible to conduct measurements in noisy environment and in the field. Objective methods can be applied with virtually no limitations. One should however never trust the results blindly. Objective results are only valid if the objective method has been validated for the type of expected impairments and they may be of limited accuracy in some applications.

5.1 Voice versus Audio Quality Assessment

While the terms of **voice** and **audio quality** are sometimes used synonymously, they describe two distinct aspects of quality, which must not be confused.

Even though, voice is of course some sort of audio signal, the term **voice quality** is used, when referring to the user perceived quality in the context of telecommunication scenarios. This includes technical equipment such as telephone sets as well as components such as voice codecs. It can also cover complex communication systems such as a whole network with various segments (PSTN, wireless, IP-based….) including the terminal equipment. The key point is, that the main focus should be communication.

On the other hand, the term **audio quality** refers to the user perceived quality of wideband audio, e.g. music as used for CDs. This of course also includes the effects (impairments) of signal processing such as codecs (for instance MP3) or technical equipment such as mixing desks, yet the user's expectation in this case is very different from a telephony context. The focus here is on entertainment and pleasure.

This difference is directly reflected in the subjective and objective methods which are used to assess the quality. If one refers to listening-only quality for telephony-band quality, a subject would be asked the following question:

"How good does the voice sample sound?"

Thus the user will have to give an absolute category referring to his own experience. This is possible, because subjects are used to handle telecommunication equipment, and therefore know what quality they can expect in such a scenario.

On the other hand, assessing quality in a wide-band audio quality context looks for the difference of an audio sample before and after the processing so that a subject will be asked

"How different do the samples sound?"

It can be seen that these two questions will lead to completely different results, which is the foundation for different assessment methods (both subjectively and objectively) which must not be confused. The following sections will further detail on this.

6 Subjective Voice and Audio Quality Assessment in Theory

6.1 Subjective Voice Quality Assessment

6.1.1 Room and Equipment Specifications

The requirements with regard to the necessary equipment for conducting subjective listening tests differ significantly between the individual recommendations. Generally wide band audio experiments require much better equipment than voice tests. While voice tests can be made using average quality headphones or good handsets, wideband audio tests demand for very expensive professional grade equipment like e.g. diffuse field equalized headphones. Also, for voice tests the room specifications are usually easy to meet, but special sound labs are required in the case of audio tests.

WARNING: Never assess voice or audio signals in a noisy environment or using consumer grade PC speakers. The minimum required is a decent sound card, good headphones and a quiet room! Please note, that these are the minimum requirements for an informal small test or prescreening. Formal subjective tests demand for much better equipment.

6.1.2 Instructions

When performing subjective tests, special care has to be taken with regard to the instructions given to the subjects. Different instructions may lead to completely different results. Guidance and example instructions can be found in the ITU documents of the individual subjective test recommendation. The instructions given also define part of the context of the subjective experiment.

6.1.3 Subjective Listening Quality Testing

All methods described in this chapter were developed for the purpose of voice quality assessment. All these models assume that the subjects in the test have a good understanding of how undistorted voices sound due to their daily experience. The subjective experiments do therefore not include a comparison between a reference and a test file. Instead only the test file is scored by the subjects. In order to cope with personal preferences of the subjects for the voice of one or the other talker, usually at least two different male and two different female talkers are used per experiment. The scores for all four talkers are then averaged and form the so-called condition MOS, which is the final result of subjective voice quality tests.

6.1.3.1 ITU-T P.800

Historically being related to the assessment of telephone connections, useful methods for testing telephone band voice signals were first standardized within the ITU-T. Recommendation P.800 defines the Absolute Category Rating (ACR) test method which has been used for the assessment of voice codecs since 1993. Within the ACR test method, the ITU five grade impairment scale is applied (see Table 6-1). Because of the telecommunication environment, testing is done without a comparison to an undistorted reference. This copes with a typical situation of a phone call, where the listener has no access to a comparison with a reference, for example the original voice of the other party. However, the listening test according to P.800 could be regarded as a comparison between a test signal and a reference "in the mind" of the listener. This is since the typical listener is very familiar with the natural sound of a human voice in his own language.

The central question which the test persons have to answer is:

“How good does the voice sample sound?”

For each voice sample five possible answers exist, which are given by Table 6-1. The subjects have to score each sample using the "Impairment" description, but usually the numerical "Grades" will be reported.

For comparison reasons, and in order to be able to merge the results of different individuals, it is necessary to adjust the listeners' opinions to an absolute scale. For this purpose, predefined examples with well defined noise insertions of fixed modulated noise reference units (MNRU, [ITUT810]) are presented at the beginning of a test. Each sample represents an example distortion corresponding to the ITU-T version of the five grade listening quality scale.

Quality	Grade
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

Table 6-1 the ITU-T 5-point listening quality scale

Based on these test conditions a population of typically 20 to 50 test subjects will be presented with an identical series of speech fragments. Each fragment has a duration of typically around 8s. Every subject will be asked to score each sample by applying the impairment scale.

In addition to the ACR method and the above mentioned quality scale, P.800 also recommends a number of other subjective test methods for different purposes (e.g. conversational quality), as well as some different quality scales, like DCR for specialized purposes. Since these methods and scales have either no direct objective counterpart, or they are rarely used in conjunction with objective tests, they are not described here.

6.1.3.2 ITU-T P.835

P.835 has been developed to subjectively assess the listening quality of voice under noisy conditions. The idea is to help the subjects in the test by asking them to score the background noise, the active speech parts and the overall quality all separately, but within the same test. Typically, each sample is divided into three subsamples. After playing the first subsample, the subjects have to score the quality of the active speech signal, after the second subsample the quality of the background signal and after the third subsample the overall voice quality. For the first two subsamples scoring scales are used which have no direct counterpart in the objective world. The overall quality however is scored using the P.800 ACR scale given in Table 6-1. The duration of the subsamples in P.835 is defined as 1s leading noise, followed by 2s speech and finally 1s noise again. The very short speech period makes it difficult to use speech samples from P.835 tests for objective measurements. Some studies were performed that showed a very high correlation between P.800 ACR tests and the overall rating in P.835. For the overall rating the question asked to the subjects is also:

"How good does the voice sample sound?"

When comparing P.835 scores with those derived from P.800 tests one must act carefully since both recommendations have slightly different, though partly overlapping scopes. For a vast range of tests, both methods will yield similar results. P.835 is however designed to also assess signals in the presence of very strong background noise. The level of the background noise may reach levels at which the intelligibility of the voice signal may already be significantly impaired. Such conditions are outside the scope of P.800.

6.1.4 Subjective Talking Quality Testing

Currently there are no methods standardised for the isolated assessment of talking quality. Nevertheless such experiments are performed. The applied method is very similar to P.800 tests, but differing from those, the subject does not listen to a pre-recorded file. Instead, each subject is talking into a phone line and judging the perceived echo. It is important to note, that due to the different spectral content and temporal structure of the voices from different speakers, it is critical to form a Mean Opinion Score from the results. Strictly speaking, each individual vote forms one test condition in the case of talking quality tests. Due to this limited possibility to average results, the accuracy of the test is limited. To leverage this a bit it is advisable to perform talking quality tests by using expert listeners, which is also in contrast to P.800.

Very often today talking quality is assessed as one result of a full conversational test. Since in conversational tests the subjects have to fulfil a more difficult task and have to answer questions on other quality dimensions as well, the possibility of biased results can't be completely excluded.

6.1.5 Subjective Conversational Quality Testing

Conversational tests are the ones that are closest to reality of telephony systems. But they are also the most difficult ones to perform since they involve more than one subject in one test round. Also, since no pre-recorded material can be used, these tests are extremely difficult to control. It is for example well accepted today that the heat of the conversation (the number of changes of talking direction per time) combined with long delays may have a significant impact on the result. The heat of the conversation is however difficult to control since it highly depends on the temper of the subjects having the conversation.

6.1.5.1 ITU-T P.805 (P.CONV)

P.805 is an ITU recommendation for the assessment of conversational quality. The principle is easy. Two subjects are sitting indifferent rooms and have a conversation on a predefined topic. The topic ("task") can be anything from ordering a Pizza to counting numbers. It refers to the general methods described in P.800 (which is extremely vague with regard to conversational tests) and tries to better specify such methods. However, P.805 is still not really precise in so far as it allows more or less everything. Its value lies in the fact that it is an excellent list, covering most of the important consideration required for the successful conduction of conversation opinion tests. In addition, it gives examples for test scenarios, example tasks for the subjects as well as some ideas for questions that the subjects will be asked to score. P.805 tests typically give answers to questions regarding the conversational quality, talking quality and listening quality. Nevertheless, other the subjects may be asked completely different questions too, like e.g. "How comfortable did you feel?".

6.2 Subjective Audio Quality Assessment

Unlike for human voices, the test subjects do not have a general knowledge of how music sounds, since music can by definition contain any kind of sounds. In order to properly assess music, it is therefore required to compare the signal which is to be tested to a known reference signal. All audio tests which are standardized today follow this approach. This is very clear in BS.1116, but also present in BS.1534, though less obvious.

6.2.1 ITU-R BS.1116

The ITU has also recommended test procedures to assess wide band audio codecs on the basis of subjective tests. For this reason, the test method focuses on the comparison of the coded/decoded signal to the unprocessed original reference. The relevant recommendation is known as BS.1116, titled "Methods for the Subjective Assessment of small Impairments in Audio Systems including Multichannel Sound Systems" [ITUR1116]. It was issued by the ITU-R in 1994 and was updated in 1997.

The test method which is recommended by BS.1116 is referred to as "double-blind triple-stimulus with hidden reference". It is extremely sensitive and allows for the accurate detection of small impairments. The grading scale used should be treated as continuous with "anchors" derived from the ITU-R five-grade impairment scale according to ITU-R BS.562 [ITUR562]. It is depicted in Table 6-2.

Impairment	Grade	SDG
Imperceptible	5	0.0
Perceptible, but not annoying	4	-1.0
Slightly annoying	3	-2.0
Annoying	2	-3.0
Very annoying	1	-4.0

Table 6-2 The ITU-R 5-point impairment scale

Distortions as used in this context mean audible differences between the reference and the test signal. Consequently, the question answered by subjects in this case is:

"How different do the samples sound?"

The analysis of the results from a subjective listening test is in general based on the Subjective Difference Grade (SDG) defined as:

$$SDG = Grade_{\text{Signal Under Test}} - Grade_{\text{Reference Signal}}$$

Provided that the listener correctly assigns the hidden reference signal, the SDG values will range from 0 to –4, where 0 corresponds to an imperceptible impairment and –4 to an impairment judged as very annoying. The assignment of the SDG scale is shown in the last column in Table 6-2.

In contrast to the listening test according to ITU-T P.800, an explicit comparison between the test signal and a reference signal is needed in the case of BS.1116, since the listener never knows how the original signal sounds.

This method was applied in a variety of international verification tests in the past. However, it should be kept in mind that because of the scope of ITU-R it can be applied to small impairments only, which means a practical limitation to almost "transparent" studio quality. Another issue, which has been discussed among experts is the fact that it recommends to use the scale at a resolution of one decimal place, resulting in 41 (!) discrete steps. There are

indications that for some subjects this is too much of a choice, and in addition the meaning of the impairment anchors is interpreted differently [KARJ85].

6.2.2 ITU-R BS.1534 (MUSHRA)

The European Broadcasting Union EBU has proposed an advanced listening test procedure known as "MUSHRA". MUSHRA stands for "Multiple Stimulus with Hidden Reference Anchors". This method targets at testing significantly impaired audio signals, such as those derived at very low bit rates. This method has been adopted as ITU-R Recommendation BS. 1534 in 2001 and is widely used in subjective testing.

The principle is to always asses a set of samples, e.g. 10, at the same time. Amongst these are three anchors (the reference signal, the 7kHz low pass filtered reference signal and the 3.5kHz low pass filtered reference signal). Additional anchors, representing different dimensions of quality, are allowed, but optional and rarely used. The subject is allowed to freely switch between the samples. The switching must be instantaneous (without delay) and seamless. The task for the subject is to answer the question:

"How good do the samples sound relative to each other?"

This question has to be answered for all samples except for the full bandwidth reference signal, which is the only signal whose identity is known to the subjects. In order to answer this question, the subject has to adjust sliders on the screen which use a pseudo-continuous scale from 0 to 100, which uses the same anchor points as the ITU-R five-point scale shown above. A typical screen for one trial is shown in Figure 6-1.

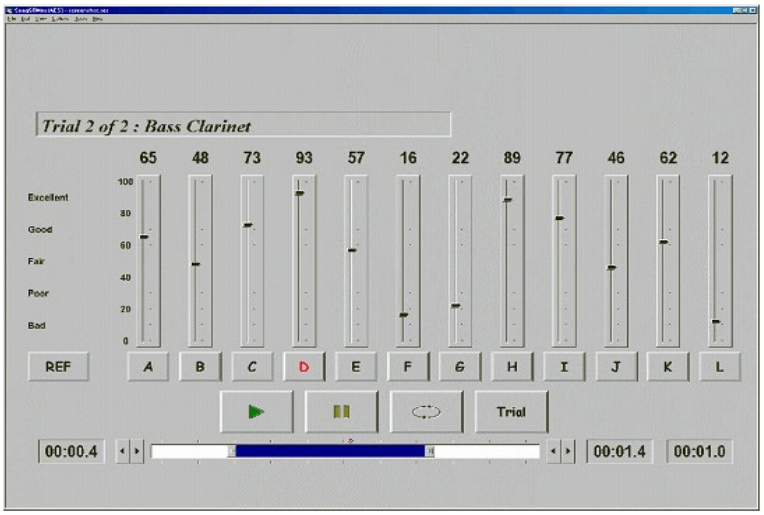


Figure 6-1 Example MUSHRA trial screen

After having scored one such trial, the next trial will be presented to the subject for scoring. Due to the anchors it is theoretically possible to combine the results from the different trials. In practice however, this highly depends on the selection of the anchors and using the low pass filtered signals only as anchors, is certainly not enough since they represent only one dimension out of a multitude of possible degradations and not necessarily define the scale used by the subjects.

7 Objective Voice and Audio Quality Assessment in Theory

During all the years in the development of compression schemes assessing the quality was a pending issue. Consequently, the idea of substituting the subjective tests by objective, computer-based methods has been an ongoing focus of research and development. Early work motivated from the development in voice coding was reported in [KARJ85]. Since then several methods were introduced.

7.1 FR vs. NR Methods

The first generation of algorithms was focused on so called full-reference models (often also incorrectly referred to as intrusive algorithms) which are based on the comparison of an undistorted reference sample with a distorted test sample. Later research has led to no-reference models in some fields, which are solely based on the assessment of the test sample without having access to the undistorted reference. While such models have proven to work reliably e.g. when assessing voice transmission quality [ITUT563] in other fields such as audio or video quality assessment they can only provide reliable outputs under very limited conditions. The reason for this is that each computational model needs to have some a-priori knowledge about the signal under test (see below). In case of voice quality assessment such knowledge can be modeled, as it is known that the signal under test is voice and therefore has certain properties. Research has defined quite well which sounds can be produced by the human voice and which ones cannot. For other types of signals such as music or video this is not as clearly defined. Certain aspects which could be found in a “degraded” signal could be artifacts caused by a system under test as well as artistic features which are inserted into the signal intentionally.

7.2 FR Methods for Telephony Bandwidth Voice Signals

Generally, it can be said that full-reference methods are the current state of the art, delivering the most reliable results at the cost of needing to have access to an undistorted reference sample. The common structure of these algorithms is depicted in Figure 7-1. The process of human perception is modeled by employing a difference measurement technique which compares both, a reference signal (i.e. the "input" signal to a codec) and a test signal (i.e. the "output" signal of the codec). First, the algorithms process an ear model for the reference and the test signal, in order to calculate an estimate for the audible signal components. The result can be imagined as the "internal representation inside the human auditory system”. The comparison of the internal representations of the reference, and the test signal leads to an estimate of the *audible difference*. In order to derive an overall quality figure, this information, which is a function of time, must be processed accordingly, like the human brain of a subject would do in a listening test. The respective part of processing within an algorithm is referred to as *cognitive modeling*. In the end, a total quality figure will be derived, which can be compared to a MOS ("Mean Opinion Score”) resulting from a listening test.

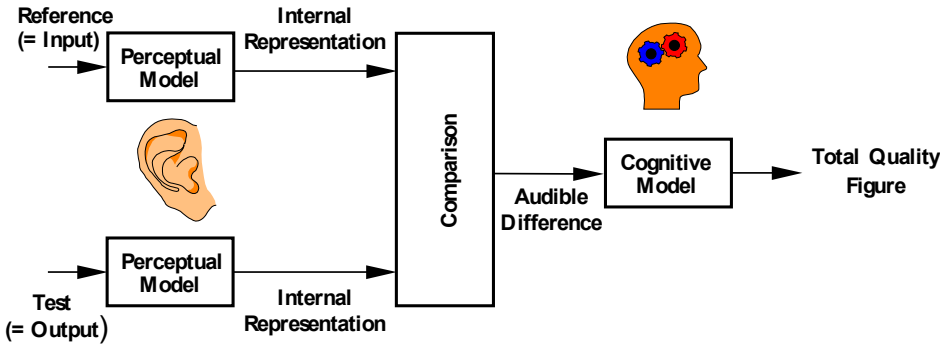


Figure 7-1: The underlying concept for perceptual measurement

The evaluation of the internal representation is often related to an estimate of the masked threshold. This estimate is based on data found in a number of psychoacoustic experiments, such as those conducted by Zwicker [ZWIC67,

ZWIC82]. Most of these experiments model certain isolated effects of the human auditory system. One way to design a perceptual measurement algorithm is to generalize these models and apply them to complex audio signals. This was for example the approach outlined in the NMR Algorithm in 1987 [BRAN87, BRAN89, BRAN92, GILC96, HERR92a, HERR92b, KEYH93, KEYH96, KEYH98, SEIT89]. Similar approaches were used for PAQM and PSQM [BEER95, BEER92, BEER94].

7.2.1 Listening Quality Measurement with ITU-T P.861 (PSQM)

Within the telecommunication sector of the ITU, in 1996 study group 12 finalized recommendation P.861 [ITUT861] for the objective analysis of voice codecs. This was the first ITU-T recommended objective quality measure. After a wide-ranging comparison of proposed methods, the group opted for the PSQM algorithm. PSQM correlated up to 98 percent with the scores of subjective listening tests. In the following years there have been several enhancements to this algorithm and finally in 2001 this standard was succeeded by the successful ITU-T Recommendation P.862, widely known as PESQ. The question answered by P.861 is:

"How good does the voice sample sound?"

7.2.2 Listening Quality Measurement with ITU-T P.862 (PESQ)

Driven by the demand for a verified test procedure for VoIP, an expert group within ITU-T SG12 has been working on an improved voice quality model. After a competitive phase, the new model PESQ has been devised. PESQ stands for "Perceptual Evaluation of Speech Quality". PESQ combines a further refinement of PSQM and a measure which was known as PAMS. Extensive tests showed PESQ's superior performance especially for VoIP applications. In February 2001 PESQ was accepted as the ITU-T Rec. P.862 [ITUT862]. Since then PESQ has proven to be the most reliable algorithm for objective voice quality assessment, and has reached a strong market penetration. From 2001 up to now different amendments to the standard have been introduced as follows:

- Mapping function for the PESQ score to the MOS scale (MOS-LQO) and PESQ conformance testing [ITU862.1]
- Introduction of wideband voice assessment with an appropriate mapping function to the MOS scale [ITUT862.2]
- Application guide for PESQ [ITUT862.3]

The question answered by P.862 is:

"How good does the voice sample sound?"

7.2.3 Listening Quality Measurement with ITU-T P.863 (POLQA)

POLQA is the next-generation voice quality testing technology for fixed, mobile and IP based networks. POLQA has been selected to form the new ITU-T voice quality testing standard, P.863, and will be used with HD Voice, 3G,4G/LTE and 5G. In February 2011 it was consented by ITU-T Study Group 12. Revised versions were approved in 2014 and 2018.

The POLQA standard was developed between 2006-2010 by leading experts in a competition carried out by Study Group 12 of ITU-T, in order to define a technology update for PESQ/P.862. POLQA will offer a new level of benchmarking capability to determine the voice quality of mobile network services. The POLQA perceptual measurement algorithm is a joint development of OPTICOM, SwissQual and TNO, protected by copyright and patents and available under license as software for various platforms.

The former industry standard PESQ/P.862.x was originally released in the year 2000. But since then, and as shown in the time line diagram in Figure 7-2 below, certain key technologies like speech codecs and 3G/4G evolution had been introduced. While PESQ has proven to be very robust, an updated measurement technology was desirable to overcome recently discovered limitations under specific circumstances. POLQA provides significantly improved benchmark accuracy for 3G, and strong support for testing of most recent technologies such as Unified Communications, Next Gen Networks and 4G/LTE. Besides classical band-limited telephony POLQA also supports HD Voice, i.e. 'wideband' and 'super-wideband' telephony up to 14.5 kHz audio bandwidth. The subsequent maintenance updates of POLQA included minor improvements but also significant extensions, like fullband measurements with an audio bandwidth of up to 20kHz, which keep it fit for 5G and other yet to come applications.

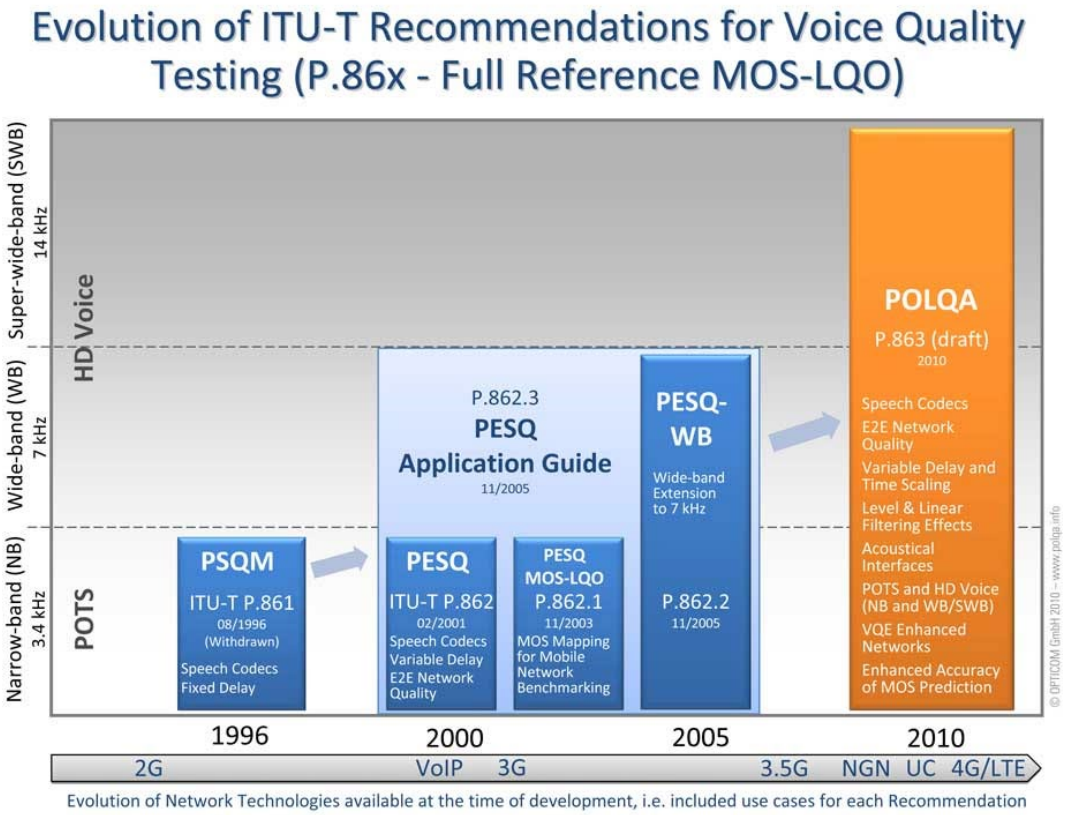


Figure 7-2, Evolution of perceptual voice quality measurement standards.

For most applications the change to POLQA is optional, though it will lead to improved accuracy in most cases. For the following applications however, POLQA is clearly preferable:

- Direct comparison of EVRC with AMR type codecs
- Fully validated for EVS at all bandwidth modes
- Narrowband, and super-wideband / fullband measurements (incl. wideband)
- Acoustic interfaces on the receiving side
- Terminal assessment
- Severe linear filtering
- Very high background noise levels

- Time scaling (playout speed variations)

7.2.3.1 Versions of P.863

The initial version of the ITU recommendation was V1.0. This version included a few bugs which were corrected in Version 1.1 in November 2011. A major revision of the recommendation took place in 2014 and resulted in POLQA 2015 / V2.4, while in 2018 POLQA V3 was released as a major update and significant extension of the scope of POLQA. All versions produce slightly differing results on the same data, but the average across a larger number of file pairs is very similar 8as long as these are within the scope of the respective POLQA version).

7.2.3.2 Changes in POLQA 2015 / V2.4

POLQA V1.1 was found to have three minor issues, which had very little effect in most real measurement scenarios, but which nevertheless may have caused problems if users were unaware of standard best practices for the use of POLQA. These issues were,

- **Length dependency** of the results in narrowband (NB) mode if the signal duration exceeded approximately 10 s in narrowband mode
- **Transparency problems:** Too often a comparison of the reference signal with itself resulted in MOS scores below the optimum (known as "the transparency problem")
- **Shift performance:** Small shifts of the starting point of the degraded signal may have had unexpectedly large impacts on the measured MOS

Due to time constraints it was not feasible to tackle all three matters for this update. Where the shift performance is critical there already exists a pragmatic solution, by using the High Accuracy mode (HA-Mode), therefore this matter was deferred for possible future consideration, while the other two issues were solved with this update. In addition, the update to V2.4 also provides several other improvements due to general bug fixes and optimization, which resulted in a general better performance and higher stability of the algorithm.

A white Paper which highlights the most important changes is available from www.polqa.info .

7.2.3.3 Changes in POLQA 2018 / V3.0

V3 formed the most significant update so far, as it replaces the former super-wideband mode with a fullband mode, which extended the measurement bandwidth beyond the frequency range currently used by codecs in the field. In addition, the perceptual model was drastically simplified and at the same time the accuracy of the model could be further increased. Also, problems of former versions with the extensions of very short pauses during speech segments as well as with level variations were eliminated, which was important for OTT and VoLTE applications. V3 is fully 5G ready and replaces all previous versions.

7.2.4 Talking Quality Measurement with ITU-T G.131

The ITU has released a recommendation which specifically discusses the annoyance of echo as perceived by users. This Recommendation [ITU131] is not only used in planning networks and transmission systems but is also employed to assess network performance with respect to the user's perception of echo. In this Recommendation graphs are given, which directly relate echo parameters such as the echo level and delay to the expected user behavior (whether he will accept this echo or will be disturbed by its effect) which have been based on subjective test series. G.131 is however not a perceptual measure. It uses tones or chirps to measure the echo return loss (ERL) and the delay of the echo. A curve is presented in G.131 which specifies which ERL is acceptable for users at specific echo delays. The given curve has been developed for single echoes only. It's applicability to signals with multiple echoes is unknown.

7.2.5 Talking Quality Measurement PESQM / PESQ-TQ

While the most well-known voice quality metrics such as P.863 (POLQA), P.862 (PESQ) or P.563 focus on the effect of user perceived listening quality, they are not suitable to assess the overall user perceived quality of a phone call which has to consider also effects such as the user's perception of his own voice (the so-called talking quality),

which is strongly influenced by echo and effects of interactivity which can be impaired e.g. by delay. In order to allow for objective assessment OPTICOM is actively working within different ITU-T expert groups to pursue the standardization of similar metrics as PESQ for these types of impairments. For talking quality different approaches have been proposed to the ITU study groups already, known e.g. as the Perceptual Echo and Sidetone Quality Measurement (PESQM) [BEER00] and its most recent enhancement by OPTICOM, PESQ-TQ.

7.3 NR Methods for Telephony Bandwidth Voice Signals

7.3.1 Listening Quality Measurement with ITU-T P.563

While PSQM and PESQ are full-reference models which require the user to have access to both an undistorted reference signal and the distorted test signal, the ITU-T Study Group 12 has in 2004 approved a standard for a no-reference voice quality measure after carrying out a competition between different proponents. This standard incorporates OPTICOM's approach to a no-reference model with those of two other companies working on this field. In a collaboration of OPTICOM and its partners, a model could be designed which leads to the highest correlation with subjective user perception, which was fulfilling the strict requirements of the ITU [ITUT563]. It should be noted that only this combined model is the ITU-T approved no-reference model even so there are metrics which are marketed by claims such as "very close to the results of P.563".

While single-sided approaches, such as P.563 have the advantage that they do not need a reference signal and are therefore qualified for passive monitoring, this property comes at the cost of a strong talker dependency. Full-reference models can partially compensate for the effects different voices have on the perception of quality, as they can directly compare the distorted signal with its reference, while no-reference models do not have the option to do so. This introduces the difficulty that for single assessments, no-reference measurements show quite a large variability. Only by carrying out a large number of tests with various different speakers, the results of objective no-reference models will have similar accuracy as full-reference models or subjective tests. The question answered by P.563 is:

"How good does the voice sample sound?"

7.4 FR Methods for Wideband Audio Signals

7.4.1 ITU-R BS.1387 (PEAQ)

Within the study period 1994 – 1998 the ITU-R had established task group 10/4 with the scope to recommend an objective, perception based model to evaluate the quality of wide band audio codecs. After collecting a set of proposals, including the most popular ideas such as NMR, PAQM, PERCEVAL, POM and others, the group of model proponents opted for a joint collaboration to derive an improved model. In 1998, two versions of this new model were presented: A "Basic" version, featuring a low complexity approach, and an "Advanced" version for higher accuracy at the trade off of higher complexity. After thorough verification, the model was recommended as a measure for the perceived audio quality ("PEAQ") under recommendation BS.1387 in late 1998. The question answered by BS.1387 is:

"How different do the samples sound?"

8 Summary of Subjective and Objective Test-Methods

The standards ITU-T P.863, and ITU-R BS.1387, today represent the state-of-the-art technique for the objective evaluation of the perceived voice and audio quality. It should be noted, however, that both techniques were derived from modeling the corresponding subjective experiment by an algorithm-based approach. Thus, it is essential to understand the scope of the modeled subjective experiment when trying to interpret the calculated results. ITU-R BS.1378 is based on a BS.1116 listening test whereas ITU-T P.863 is based on a P.800 Listening test.

9 Objective Listening Quality Assessment with POLQA

9.1 POLQA Measurement

When PSQM was standardized as P.861, the scope of the standard was at that time the assessment of state-of-the-art codecs as they were mainly used for mobile transmission, like GSM. VoIP was not yet a topic at this time. The requirements for measurement equipment have changed dramatically since then. As a consequence, the ITU introduced the P.862 (PESQ) standard in order to cope with the new demands arising from modern networks like VoIP. With these networks the measurement algorithm has to deal with much higher distortions as with GSM codecs, but maybe the most eminent factor is that the delay between the reference and the test signal is not constant anymore, instead it is time varying. However, development of network transmission schemes and codecs is an ongoing process. Additional requirements arise for modern measurement schemes. To stay abreast of these changes the new ITU standard P.863/POLQA was developed. POLQA can handle effects caused by new voice services like stretching and compression of speech signals in the time domain. POLQA improves the quality prediction for new and old codecs and allows the direct comparison of AMR and EVRC codecs. POLQA combines an excellent psychoacoustic and cognitive model with a new time alignment algorithm that perfectly handles varying delays.

9.2 Description of the POLQA Algorithm

A general overview on the algorithm is shown in Figure 9-1. The inputs to the algorithm are two waveforms represented by two data vectors containing 16 bit PCM samples. The first vector contains the samples of the (undistorted) reference signal, whereas the second vector contains the samples of the degraded signal. The POLQA algorithm consists of a temporal alignment block, a sample rate estimator of a sample rate converter, which is used to compensate for differences in the sample rate of the input signals, and the actual core model, which performs the MOS calculation. In a first step, the delay between the two input signals is determined and the sample rate of the two signals relative to each other is estimated. The sample rate estimation is based on the delay information calculated by the temporal alignment. If the sample rate differs by more than approximately 1%, the signal with the higher sample rate is down sampled. After each step, the results are stored together with an average delay reliability indicator, which is a measure for the quality of the delay estimation. The result from the resampling step, which yielded the highest overall reliability, is finally chosen. Once the correct delay is determined and the sample rate differences have been compensated, the signals and the delay information are passed on to the core model, which calculates the perceptibility as well as the annoyance of the distortions and maps them to a MOS scale.

A much more detailed and comprehensive description of the algorithm can be found in ITU-T P.863 [ITUT863]. The next few sections are only intended to give an overview on the basics of POLQA's internal structure.

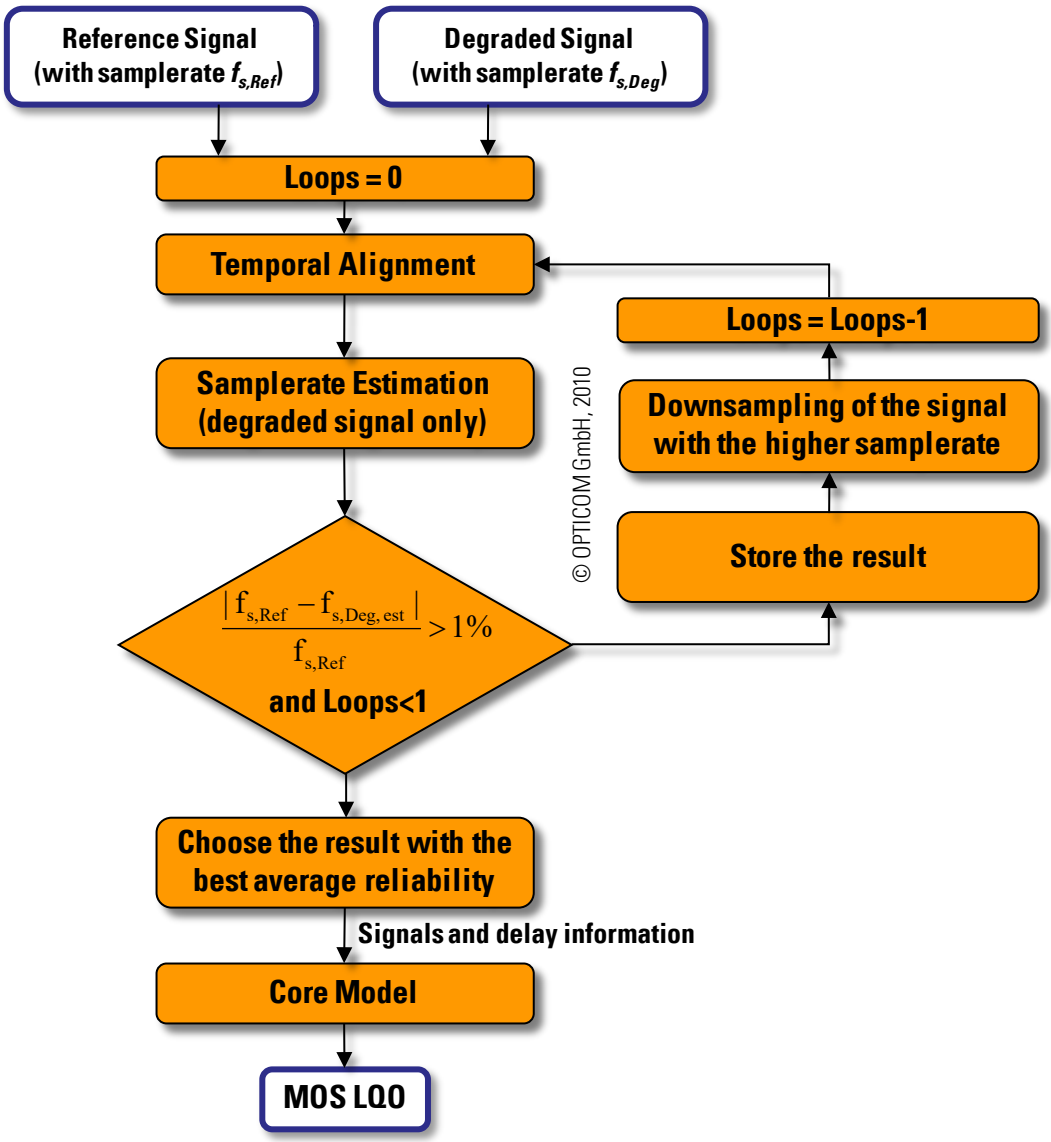


Figure 9-1, general overview of POLQA

9.2.1 The Core Model

An overview of the core model is given in Figure 9-3. The main element here is the perceptual model which is calculated four times using different parameters in order to cope with different major distortion types. Those distortion types can be split into additive distortions and subtracted distortions. For both types a further distinction is made between very strong and weaker effects. The inputs to the perceptual models are waveforms and the delay information. The output is the "Disturbance Density", which is a measure for the perceptibility of distortions in the signals. The perceptual model for the main branch also produces indicators for Frequency distortions, Noise and Reverberation distortions. A subsequent switch which is triggered by a detector for very strong distortions reduces the four Disturbance Density values down to two, one for added and one for subtracted distortions. So far, the Disturbance Density is an indicator for the perceptibility of distortions only and cognitive effects are not yet taken into account. Cognitive aspects are however important when human beings are asked to score the quality of what they can perceive. Essentially, they convert the perceptibility measure Disturbance Density into an annoyance measure. This conversion is performed by correcting the Disturbance Density values for situations with:

- Significant level variations
- Many frame repetitions
- Strong timbre
- Spectral flatness
- Noise switching during speech pauses
- Many delay variations
- Strong variations of the Disturbance Density over time
- Strong variations of the loudness of the signals

Two further indicators, one for spectral flatness ad one for level variations are also calculated in this step.

So far, all operations were performed on frames with a duration of approximately 32 and 43ms duration (depending on the sample rate and using an overlap of 50%, resp. 75% in V3) and for each Bark bands separately. In a final step all indicators are integrated over time and frequency in order to compute the final MOS LQO value. POLQA version 3 does not distinguish between "Big Distortions" and other distortions anymore and thus calculates the core perceptual model only twice, once for all distortions and once for additive distortions. Figure 9-2 refers to POLQA V2 and V1.

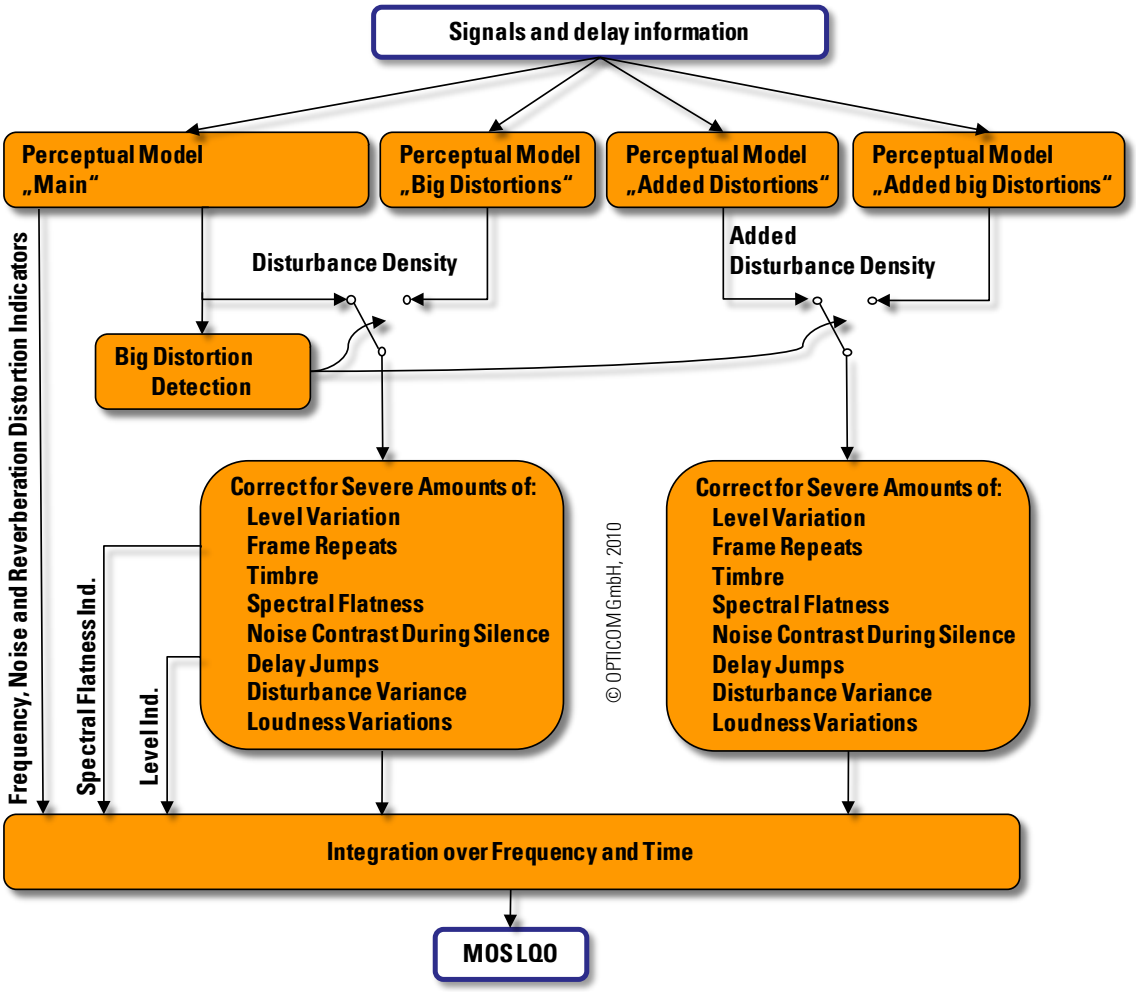


Figure 9-2, overview of the POLQA core model. Calculation of final disturbance densities from the four different variants of the internal representations (two in POLQA V3).

9.2.1.1 The Perceptual Model

The key concept behind the perceptual model (Figure 9-3) is "Idealisation". The idea behind this is, that POLQA is supposed to simulate ACR tests. In an ACR test however, subjects have no comparison to the actual reference signal when they score a speech signal. Instead, it is assumed that subjects have an understanding of what an ideal signal sounds like and they use this as their own reference. Consequently, if they are asked to score a reference signal which is not absolutely perfect (e.g. it has the wrong volume or contains too much timbre, noise or reverberation), it will be scored worse than perfect. In its idealization step POLQA therefore corrects small imperfections of the reference signals in order to derive the same ideal reference for the comparison to the degraded signal as human subjects would use in their minds. Similar to the idealization of the reference signal, some distortions present in the degraded signal which are hardly perceptible in an ACR test will be partially compensated (e.g. small pitch shifts, linear frequency distortions).

The perceptual model starts with scaling the reference signal to an ideal average active speech level of approximately -26dBov. No such scaling is performed on the degraded signal. It is assumed that any deviation of the level of the degraded signal from the ideal -26dBov is to be scored as a degradation of the signal.

Next, the spectra of both signals are computed using an FFT with 50% (75% in POLQA V3) overlapping frames with a duration of between 32ms and 43ms duration (depending on the sample rate). Subsequently small pitch shifts of the degraded signal will be eliminated ("Frequency Dewarping"). Now, the spectra will be transformed to a psychoacoustically motivated pitch scale, by combining individual spectral lines (FFT bins) to so-called critical bands. The pitch scale used is similar to the Bark scale with an average resolution of 0.3 Bark per band. The result is the "Pitch Power Density". At this stage the first three distortion indicators for frequency response distortions, additive noise and room reverberations are calculated.

After this, the excitation of each band is derived. This includes the modeling of masking effects in the frequency as well as in the temporal domain. The result is for each frame of each signal a head-internal representation which indicates roughly how loud each frequency component would be perceived.

Now, a further idealization step of the reference signal takes place by filtering out excessive timbre and low-level stationary noise. At the same time, linear frequency distortions and stationary noise are partially removed from the degraded signal.

A subtraction of the idealized excitations finally leads to the "Distortion Density", which is measure for the audibility of distortions.

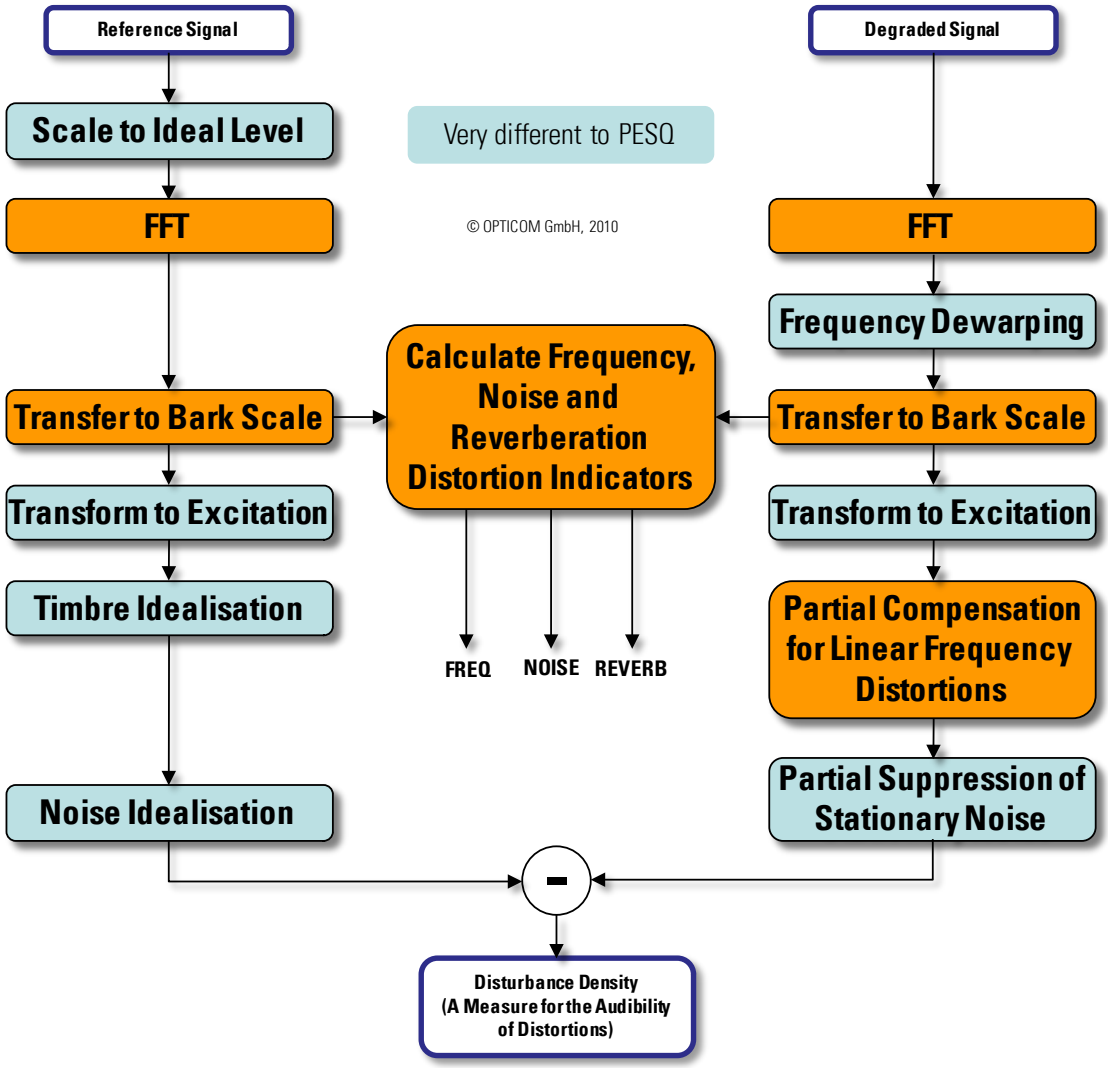


Figure 9-3, overview of the POLQA perceptual model, which performs the calculation of the Disturbance Density. Four different variants this model are calculated, each focussed on a specific set of distortions ("Main", "Big Distortions", "Added Distortions" and "Added big Distortions"). POLQA V3 calculates only two of these ("Main" and "Added Distortions").

9.3 Operating modes of POLQA

OPTICOM's POLQA implementation offers two operating modes, which allow the assessment of fullband and narrowband speech signals. In previous versions of POLQA only super-wideband signals were supported. The rest of this section is valid for POLQA V3, for V2 and before replace fullband by super-wideband.

By choosing the fullband mode a flat input filter and a different mapping function to the MOS scale than narrow band speech is used. Consequently, in fullband mode, bandwidth limitations are considered as degradations and scored accordingly. The listening quality is modeled as perceived by a human listeners using a diffuse field equalized headphone with diotic (a mono signal is presented to both ears) presentation. If POLQA is used in fullband mode, then the usage of fullband or super-wideband reference signals is mandatory, fullband references are strongly recommended.

In narrowband mode the received speech signal is compared to a narrowband reference signal. Consequently, normal telephone bandwidth limitations are not considered as severe degradations and are scored much less than in super-wideband mode. The listening quality is modeled as perceived by a human listener using a loosely coupled IRS type handset at one ear.

Note: When interpreting POLQA results it is important to know which version of the algorithm was used. Although the narrowband and the supre-wideband/fullband version of POLQA will generate MOS scores on the same five-point scale, it is strictly forbidden to directly compare results obtained in super-wideband/fullband mode with results obtained in narrowband mode!

If super-wideband or fullband networks have to be compared to narrowband networks, then the fullband version of POLQA has to be used in both cases. It is also not allowed to use reference signals with bandwidth <14 kHz in fullband mode (strictly speaking, the bandwidth should be up to 24 kHz, but the spectral content above 14kHz is absolutely negligible for clean speech signals).

The ITU Rec. P.863 strictly requires a sample rate of 48kHz for fullband measurements. For narrowband applications, 8, 16 and 48kHz sample rate are supported. As an extension to this, the OPTICOM POLQA Library provides the capability to resample the input signals to these sample rates if required.

9.3.1 Fullband vs. super-wideband mode

Previous versions of POLQA supported super-wideband operation only, Since POLQA V3, fullband operation is the default. Since clean reference signals typically contain no frequency components above 14 kHz, which would be relevant in an ACR test, SWB reference signals can still be used with POLQA V3. The big difference is however, that degradations may very well exceed 14 kHz bandwidth and POLQA V3 will detect these, while previous versions did not!

9.4 Source Material

It is important that test signals for use with POLQA are representative of the real signals carried by communications networks. Networks may treat voice and silence differently and coding algorithms are often highly optimized for voice – and so may give meaningless results if they are tested with signals that do not contain the key temporal and spectral properties of voice. Further pre-processing is often necessary to take account of filtering in the send path of a handset, and to ensure that power levels are set to an appropriate range.

9.4.1 Recommended Number of Used Speech Samples

There is one simple rule: The more speech samples are used the better will the characterisation of the system under test be. If natural speech recordings are used, the guidelines given in clause 7 of P.830 [ITUT830] should be

followed, and it is recommended that a minimum of two male talkers and two female talkers be used for each testing condition. If talker dependency is to be tested as a factor in its own right, it is recommended that more talkers be used: Eight male, eight female and eight children.

Of course, this is not always feasible and also not always required. A full characterisation is definitely required when terminals shall be assessed. If only the transmission through a network without any transcoding shall be tested, then it may be enough to use one speech sample. Since transcoding is often unavoidable (even worse, one can often not predict if it will happen), a compromise is needed for e.g. drive testing. In drive testing there is also the requirement to keep the measurement period short since it correlates directly with the possible geographical resolution of the test result. The pragmatic approach to this is to use short speech samples with two or four utterances spoken by different speakers. Before using such signals, they should however be validated to make sure that the maximum scores are achievable with a specific sample.

Note: OPTICOM provides a set of speech samples which have been tested for their suitability to be used for POLQA measurements. These speech samples are available in different formats and languages. Distribution of these files is via download from the OPTICOM License Management Server.

The speech samples provided by OPTICOM include some mixed talker samples and have all been tested to be suitable for valid POLQA measurements.

9.4.2 Choice of Source Material

Reference speech or reference signal is the original speech signal without any degradation. This should be recorded and stored in conformance with ITU-T Recommendation P.830 [11]. In the case of an acoustical sending path, this signal is used for feeding the artificial mouth. This speech signal is used by the POLQA algorithm as a reference against which the effects of the system under test are revealed. Examples of such speech files are to be found in ITU-T Recommendation P.501 [6].

POLQA was tested with human speech material. For a consistent speech quality prediction, active speech parts and speech pauses are required in the speech sample. It is recommended to use typical spoken sentences with typical syllable and word structures. It is not recommended using single word samples only (i.e. counting).

The reference signals should be leveled to -26 dBov rel. to OVL. Other signal levels will be accepted by POLQA to and internally equalized; however different levels than -26 dBov could cause amplitude clipping or lower SNR.

In order to achieve optimum results, it is essential to keep noise during speech pauses very low. Based on empiric studies, -85dBov (white noise) are ideal. Higher noise levels may lead to non-transparent reference signals, which means that the reference signal compared to itself will score lower than 4.75/4.8 (SWB/FB) or 4.5 (NB). For details on transparency see section 9.5.1.1.

The use of artificial speech signals and concatenated real speech test signals is recommended only if they represent the temporal structure (including silent intervals). Artificial speech test signals can be prepared in several ways. A concatenated real speech test signal may be constructed by concatenating short fragments (e.g. one second) of real speech while retaining a representative structure of speech and silence. Alternatively, a phonetic approach may be used to produce a minimally redundant artificial speech signal which is representative of both the temporal and phonetic structure of a large corpus of natural speech. Test signals should be representative of both male and female talkers. It should however be noted that for artificial speech a direct subjective assessment is not possible as subjects are not able to determine the quality correctly when artificial speech samples as described above are used. Therefore, the use of real speech samples is preferred to also allow subjective assessment of the test data. No tests were performed yet for POLQA using artificial speech.

9.4.3 Temporal structure and duration of source material

Test signals should include speech bursts separated by silent periods of at least 1s but not more than 2s to be representative of natural pauses in speech. As a guide, 1s to 3s is a typical duration of a speech burst, although this varies considerably between languages. A minimum of 3s of active speech should be included in each file. A sufficiently long silent period should be included between speech bursts as certain types of voice activity detectors are sensitive only to silent periods that are longer than 200ms. For best results at least two speech bursts separated by a one pause are required since POLQA assesses the quality of the silent periods between the active speech sections only and not during the leading or trailing silence. If only one utterance is used, the result will still be correct for a continuous stream of speech, but not for a realistic conversation. For cases with no degradations during speech pause, this may however be accurate enough.

Most of the experiments used in calibrating and validating POLQA contained pairs of sentences separated by silence, totaling 8s in duration; in some cases, three or four sentences were used, with slightly longer recordings (up to 12s). Recordings made for use with POLQA should be of similar length and structure.

Thus, if a condition is to be tested over a long period, it is most appropriate to make a number of separate recordings of around 8s to 20s of speech and process each file separately with POLQA. This has additional benefits: if the same original recording is used in every case, time variations in the quality of the condition will be very apparent; alternatively, several different talkers and/or source recordings can be used, allowing more accurate measurement of talker or material dependence in the condition.

Note that the non-linear averaging process in POLQA means that the average score over a set of files will usually not equal the score of a single concatenated version of the same set of files.

WARNING: If long sequences are used, POLQA **V1.1** results are known to be biased be the sample length. This is a known issue and **has been fixed in V2.4 of P.863**.

NOTE: The OPTICOM POLQA OEM Library can handle input signals of up to 2 minutes duration. For longer input signals an error code will be returned. For signals exceeding app. 30s duration a warning will be returned in the POLQA result structure. Memory restrictions of the machine used for processing may limit this further.

9.4.4 Required Filtering, Level Calibration and Sample Rates

9.4.4.1 Signal Levels and Filtering in Super-wideband Mode

In its fullband operational mode POLQA **always** requires a fullband reference signal. It has to be provided in mono and in a sampling frequency that allows the full spectral range of fullband. The signal has to be pre-filtered by a 50 to 24'000 Hz band-pass filter. Alternativly, a super-wideband filter (50 to 14'000 Hz, according to ITU-T Rec. G.191, 14 kHz may be applied).

The only sample rate supported by P.863/POLQA in fullband mode is 48 kHz. OPTICOM's POLQA library however provides an option to automatically up sample signals with lower sample rates to the required 48 kHz. Please note, the minimum sample rate for the reference signal is 32 kHz. Otherwise the requirements regarding the spectral content of up to at least 14000 Hz cannot be met.

For all signals to be used in fullband operational mode, a digital level of -26 dBov (obtained in accordance with ITU-T Rec. P.56) corresponds to the nominal presentation level (73 dB (A) SPL in case of **diotic** presentation). The

actual presentation level can be directly derived from the P.56 level of the degraded signal (e.g., a level of -34 dBov corresponds to a presentation level 8 dB below the nominal level).

The nominal level for POLQA reference signals in fullband operational mode is therefore -26 dBov.

As far as degraded signals are concerned, potential differences between the recording level and the presentation level are allowed and are part of the test condition. These level differences have to be restricted to a range of +5 dB to -20 dB relative to the nominal level which is again -26dBov (this results in a range from -21 dBov to -46 dBov).

9.4.4.2 Signal Levels and Filtering in Narrowband Mode

In narrowband operational mode, the chosen sampling frequency has to allow the presentation of the full spectral range of narrow-band telephony (50 to 3800 Hz). The reference signal must not be limited in bandwidth below 3.8 kHz. The reference signal used by POLQA must not be IRS pre-filtered. This clean reference signal should be used for POLQA. Before this signal is sent through the system under test an IRSsend filter should be applied. This filter forms part of the channel under test.

A sampling rate of 8 kHz is preferred for narrow-band operational mode, but 48 kHz is also supported. OPTICOM's POLQA library however provides an option to automatically resample signals to 8 kHz if they are not presented at that sample-rate. The OPTICOM POLQA Library does also support 16 kHz, but the this not part of the standard. Please note, that signals provided at 16 kHz sample rate will also not be resampled if the optional automatic sample rate conversion is switched on.

The narrowband operational mode of the POLQA algorithm pre-assumes that a digital level of -26 dBov corresponds to a presentation level of 79 dB(A) SPL at the ERP (ear reference point) in **monotic** presentation. Deviations from this nominal level are taken into account as non-optimal presentation level.

9.4.4.3 Optional Automatic Level Alignment

Since the level of a speech signal is a very important factor in human perception as well as in modern telecommunication networks, POLQA takes them into account when calculating the MOS score. For this to work however, some very strict requirements on measurement systems regarding the control of record and playback levels have to be met. In most cases it will be required to either amplify or attenuate the signals in the analog domain, or to use AD/DA converters with a resolution >= 24Bits if the level control shall be performed digitally.

For systems and applications where the signal levels cannot be controlled, the OPICOM POLQA library provides an optional automatic level alignment. This level alignment will scale both input signals to an average active speech level of approximately -26 dBov. Please note that this is only a workaround and cannot replace correct tuning of the hardware. This feature is enabled by setting a flag during the initialization of the Library.

9.4.4.4 Sample Rates

The ITU Rec. P.863 specifies very few sample rates at which POLQA can operate (8 and 48 kHz for NB and 48 kHz only for FB measurements). **Since POLQA 2015 / V2.4** an automatic sample rate conversion is part of the model. This conversion accepts all sample rates between 8 and 48 kHz and will resample the input signals depending on the selected mode (NB or SWB). 8 kHz input signals will never be resampled, all others will be sampled to 48 kHz in SWB mode and to 8 kHz in NB mode. We call this the *model-internal resampling*. This model-internal resampling requires that reference and degraded signal are sampled at the same rate. The model-internal resampling is available starting from POLQA 2015 / V2.4 only and it cannot be switched off.

To ease the application and implementation of **POLQA V1.1**, the OPTICOM POLQA Library can automatically resample the input signals the correct required sample rate if the input signals are provided with any non-standard sample rate. This *external resampling* is enabled by setting a flag during the initialization of the Library. The resulting sample rate used for the processing depends on the selected mode (NB/SWB) and the sample rate in

which the signals are presented. In NB mode the behavior is such that if both signals are of the same sample rate and this sample rate is 8, 16 or 48 kHz, no resampling is performed. In all other cases, both signals will be resampled to 8 kHz. In SWB mode all signals will be resampled to 48 kHz. Both signals may be sampled at different rates. This feature is available for all versions of POLQA.

NOTE: The model-internal resampling and the external resampling do not necessarily result in the same sample rate! Differences exist for 8 kHz / NB mode as well as 48 kHz /NB mode. Also, the model-internal resampling will resample 16 kHz to either 8 or 48 kHz, while the external resampling will not change it.

If V2.4 is selected **and** the external sample rate conversion is active **and** both input signals have the same sample rate **and** this sample rate is different from 16 kHz, then the signals will be resampled as required before the actual POLQA algorithm is started. In this case, the model internal conversion will have no effect.

9.5 Important Differences between PESQ and POLQA

This chapter focuses on the main differences between PESQ and POLQA which have a direct impact on the application and the interpretation of the measurement results. This section is directed to experienced users which have dealt with PESQ in the past and are now investigating the usage of POLQA.

9.5.1 Differences between the POLQA and PESQ Core Models

9.5.1.1 Transparency - or why Comparing a Signal to Itself gives Unexpected Results

In PESQ, a comparison of a file with itself will always result in a perfect MOS-LQO. For POLQA this is not necessarily the case.

The background to this lies in the behavior of listeners in a subjective experiment. Listeners do also not score all reference signals as perfect. Especially, timbre in voices or background noise during the silent intervals is critical for listeners. If any of those exceeds a certain amount, then listeners will score the signal below perfect. In order to cope with this effect, POLQA performs an idealization of the reference signal before it is compared to the not idealized degraded signal.

Due to this asymmetry in the processing of the reference and the degraded signal, the MOS measured when comparing the reference signal to itself may be slightly smaller than expected, but this is in line with subjective experiments, where subjects as well make the comparison to an ideal which exists in their minds only (there is no direct comparison of two files in the common ACR method). This principle of idealization is also one of the reasons for the improved accuracy of POLQA compared to PESQ.

One way to avoid confusing results which might result from this behavior is to use selected reference signals which *sound* perfect. Such signals can often be constructed from existing reference files by filtering them slightly. For samples consisting of concatenated voices this is however difficult due to the timbre variations between the individual voices. Another essential factor for transparency is the noise level during silent intervals. This should ideally be in the range of -85dBov and have a white spectrum. Comparing an ideal reference signal with itself will then lead to the expected maximum MOS.

9.5.1.2 Effects of Level

PESQ is almost completely insensitive to level differences between the reference and the degraded signal. Also, the perceptual effect of level variations is very often under predicted by PESQ. Both factors are corrected in POLQA, which scores these effects in the same way as they are perceived in subjective listening only tests. Consequently, some rules have to be followed in order to obtain correct results:

- POLQA strictly assumes that **the reference signals** used have an active speech level of -26dBov (measured acc. to P.56) and are presented at a nominal level of 73dB SPL at the reference point of **both** ears in super- wideband mode (diotic presentation!) and 79dB SPL at the reference point of **one** ear in narrowband mode (monotic presentation!).

- In super-wideband / fullband mode, the **degraded signals** used for the validation of POLQA had an active speech level in the range between -21dBov to -46dBov for which POLQA worked reliably. A level of -26dBov would represent the ideal normal listening level. Further investigations in the future may extend the operating range.

- In narrowband mode, the **degraded signals** should be scaled towards 26dBov. Further investigations on the behavior of POLQA for severe deviations from this level in narrowband mode have not yet been performed.

Please note that the frequently used tool "Cool Edit Pro" / "Adobe Audition" does not measure according to P.56 and that the resulting levels may vary by up to approximately 3dB.

Under all circumstances the correct scaling between digital levels and sound pressure levels as described for the reference signals above should be maintained.

9.5.1.3 Narrowband – Wideband – Fullband

For PESQ, there are two operating modes, narrowband (NB, 300-3400Hz) and wideband (100-7000Hz). For POLQA, also two modes exist, but they cover narrowband and fullband (SWB, 50-24000 Hz). This means, there is an overlap between the NB modes, but no direct correspondence exists for the WB respectively FB modes.

For NB, the results obtained by either algorithm are usually in a similar range, but they will hardly ever be exactly the same. Comparing individual results from PESQ measurements to results obtained by POLQA should clearly be avoided and certainly those comparisons make no sense at all for applications for which PESQ wasn't designed or fails.

PESQ's WB mode is completely replaced by the SWB/FB mode of POLQA. The major consequence of this is that in order to correctly apply POLQA for WB scenarios, SWB/FB reference files must be used. The achieved scores will then be on a slightly different range of the MOS scale as they used to be for PESQ in WB mode. This is caused by the different usage of the five-point scale which is applied in all modes and which must now accommodate for scores given to NB, WB and SWB/FB conditions in the same experiment. Nevertheless, care was taken to place the scores obtained by POLQA for typical WB conditions in a *similar* range as those obtained by PESQ WB for the same conditions. If a more exact match of the scales is required, then a transformation of one of the scales must be used. The maximum score which can be achieved by using POLQA to compare a SWB/FB reference signal to its WB equivalent is around 4.5, the same as with PESQ WB.

Using WB reference signals with POLQA is technically possible, but a scale transformation will definitely be required. **Please note, that using WB or NB reference signals in SWB/FB mode are not part of P.863 and should be avoided.**

In order to perform a direct comparison between PESQ and POLQA results, a scale transformation of the POLQA scores is required in order to compensate for the different context (WB vs. FB). Such a transformation is usually performed by applying a monotonic third order polynomial fit to one of the dataset.

Compared to PESQ, POLQA always shows a much higher accuracy for wideband scenarios, no matter, which reference signals are used.

9.5.1.4 Range of MOS values

Table 1 shows the limits of the MOS scales used by the PESQ (P.862.1/2) and POLQA, depending on the operating mode of the algorithm. Note that there is no overlap for WB and SWB mode. In SWB mode, POLQA will typically score an excellent WB signal compared to a SWB reference signal at around 4.5 as well. In the same way, a perfect NB signal compared to the corresponding SWB reference signal will be scored at around 4.0.

Mode	P.862.1/2 MOSmin	P.862.1/2 MOSmax	POLQA MOSmin	POLQA MOSmax
NB	1	4.5	1	4.5
WB	1	4.5	-	-
SWB	-	-	1	4.75
FB	-	-	1	4.8

Table 3, limits of the used MOS scales per mode.

9.5.1.5 Full Scale Context

One consideration that led to the design of PESQ was to predict the results of either plain NB or plain WB experiments. In NB mode, POLQA does exactly the same. In SWB/FB however, POLQA is trained to predict full scale experiments. These are experiments which contain a balanced amount of NB, WB, SWB and FB conditions. The effect of this is that for the first time a measure can be used to directly compare all those data. On the other side, a direct comparison with plain NB or WB experiments will show a skew in the scales which are used and will require a first or better third order regression to be applied to the MOS scores before they can be compared (in most cases the Excel "Trendline" will do a sufficient job).

9.5.1.6 Effects of Linear Frequency Distortions

PESQ was very insensitive to linear frequency distortions. They only had an impact if they were extreme. This was caused by the equalization built into PESQ which compensated such distortions almost completely. Especially for handset testing this meant a significant drawback. In POLQA those issues have been corrected and the weighting of such distortions is now in line with human perception.

9.5.1.7 Requirements for Recordings

While PESQ was very tolerant towards incorrect recording levels (and thus often wrong in such cases…), POLQA requires fairly exact control of the levels as mentioned above. If the ratio between the digital level and the sound pressure level is unknown, then it is better to scale both signals to -26 dBov, rather than performing a measurement with a "by chance" recording of only 40 dBov. Note, that care has to be taken when performing such scaling operations on digital signals with an insufficient bit resolution.

On the other side, PESQ was very sensitive to clock differences between the D/A and A/D converters involved in the transmission. This often led to too pessimistic results. POLQA is however designed to handle such cases and it will easily compensate for such clock differences and other time scaling effects in most cases. Of course, this does not mean that test equipment should now use cheaper converters/sound cards….

Another issue could rise from POLQA being a super-wideband speech quality measure. To fully exploit this, the recording equipment used must be capable of recording signals at least 32 kHz sample rate with very high quality.

9.5.2 New Application Areas

Compared to PESQ, POLQA does not only provide significantly enhanced accuracy. It also extends the application range for which perceptual measurements can be applied now. Many applications for which PESQ was unsuitable are now within the scope of POLQA.

9.5.2.1 Acoustical Measurements

While PESQ could only be used to assess acoustical recordings if you knew exactly what you were doing, this is different for POLQA. POLQA comes with full support for acoustical measurements. Those acoustical measurements include the correct weighting of reverberations as well linear and non-linear filtering characteristics caused by handsets and transducers. Also, super-wideband noise superimposed on a narrowband speech signal was part of the POLQA validation. Significant background noise at the listener side however has not yet been validated.

9.5.2.2 Time Scaling

Time scaling is in principle a variation of the playout speed of the signal. The causes for time scaling can be intentional, e.g. to compensate for packet losses or to avoid jitter buffer over-or underruns. This intentional time scaling comes in two flavors, either with or without pitch correction. In addition, time scaling effects can also be introduced unintentionally, due to poor playback or due to recording devices with inaccurate sample rates. At sample rates above 16 kHz it also becomes important to synchronize the sample clocks of the recording and the playing device. Using PESQ, the only chance was to avoid time scaling at all. POLQA instead handles all described effects without difficulties and can be safely applied in such situations. However, it is still recommended to synchronize sample clocks higher than 16 kHz in order to improve the stability of measurement results. Below 16 kHz, it is generally enough to use sufficiently accurate sample clocks, but the higher the sample rate gets, the more important becomes the synchronization of those clocks. The actual maximum amount of time scaling that can be handled by POLQA is signal dependent, but tests showed that up to 5% are typically handled well, even in the presence of a significant amount of other distortions, like e.g. background noise.

9.5.2.3 High Background Noise

For signals with very high background noises PESQ simply failed since its temporal alignment could not handle such signals. POLQA instead has been tested with very high background noise levels, and in many cases, it even worked at an SNR below 0dB. A fixed lower limit for which SNR POLQA still works can't be given since this is again highly signal dependent.

9.5.2.4 Direct Comparisons between EVRC and AMR type Codecs

PESQ is known to score EVRC and AMR type codecs with a very small, but systematic offset. It is unknown if PESQ over predicts AMR or under predicts EVRC and the effect is generally so small that it is negligible in field measurements, but it is annoying if a direct comparison between those two codec types shall be made. To avoid such a bias was one of the main targets for POLQA and this was successfully achieved.

9.5.2.5 Validation for EVS Codec

POLQA V2.4 was already known to perform excellent for EVS coded signals and the accuracy was further improved by POLQA V3 which is fully validated for such use.

9.6 Perceptual Results Obtained from POLQA

This chapter is intended to give a broad overview of the measured parameters.

9.6.1 MOS-LQO

The most eminent result of POLQA is the MOS-LQO. It directly expresses the voice quality on the MOS scale. This score is defined by ITU-T Recommendation P.863 which uses a MOS-like scale ranging from 0 (worst) up to 4.5 (best) for the narrowband mode and 0 (worst) up to 4.75 (best) in super-wideband mode or 4.8 in fullband mode.

9.6.2 MOS-LQO per Frame

Strictly speaking, it is not possible to specify an exact MOS for very short periods, like a POLQA frame, since all temporal masking effects etc. are void in this case and the resulting value can take into account spectral effects only. Therefore, the values provided by POLQA cannot be used to predict the overall MOS based on these only and also the scale is not identical to the scale of the overall MOS-LQO. The scale of the MOS per frame values is similar to the overall MOS-LQO, but not identical and not bounded to [1;4.8]. However, despite the inherent inaccuracy from a perceptual point of view, such an approximation of the MOS can be very helpful in order to identify critical sections of speech samples.

9.6.3 G.107 R-Factor / I_e Value

The POLQA library also provides a mapping of the MOS-LQO score to the scale used by G.107 (e-model). The resulting parameter is equivalent to an I_e – Value. Many people also refer to it as an R-factor. The scale ranges from 0 (bad) up to 100 (best). All values below 60 indicate unacceptable quality. The mapping between R Scale and MOS is implemented as defined in G.107.

Note: This result is valid for narrowband measurements only. Currently there is no such conversion standardized for super-wideband or fullband measurements.

9.6.4 Disturbance Density

The disturbance density is an indicator of how loud a signal degradation is perceived in a specific frequency range and at a specific time. The amplitude is in Sone and the frequency scale uses 1/3 Bark bands. For each frame a disturbance density vector is calculated. The disturbance density is a perceptual measure and takes frequency as well as temporal masking into account.

9.7 Non-Perceptual Results Helpful for Cause Analysis

Apart from the perceptual KPIs mentioned above, many other values are reported by the listening quality test module. Most of them are not directly related to listening quality, but they give further insight in the functionality and characteristics of a system.

9.7.1 Delay/Latency

As soon as a signal is processed by any piece of equipment, it will be slightly delayed. The resulting **delay** is also frequently called **latency**. While delay does not affect the listening quality, it is an important factor when assessing telecommunication systems because its value has a major impact on the interactivity of a conversation. The longer the delay is, the more discipline is required from both parties involved in a conversation. Delays larger than approx. 300ms are generally unacceptable. While the delay for the old POTS is usually in the range of a few milliseconds, it is typically around 150ms for VoIP systems, sometimes even much longer.

9.7.2 Delay Jitter

Delay jitter in the voice signal may have various reasons. The most frequent cause for this is the dynamic adaptation of the jitter buffers built into modern VoIP equipment. The purpose of these buffers is to assemble a

continuous voice stream out of the RTP (=voice) packets which arrive in bursts with non-deterministic timing. The longer these buffers are, the more packet jitter they may compensate for, but the overall latency of the speech signal is then also increasing. On the other side, if the jitter buffer is shorter, the latency is shorter too, but the danger of packet loss is significantly higher. The optimum length of the jitter buffer is depending on the network itself and the load on the network. In order to optimize the latency of VoIP equipment, adaptive algorithms are used to automatically adjust the size of the jitter buffer to whatever is required by the network.

Please note that in this context delay and jitter are both referring to the speech signals. They must not be confused with the packet delay and jitter in IP networks even so they are of course resulting from these effects. There might be a significant network jitter in a system under test which will be translated into a large overall delay of the voice without any jitter because a sufficiently large jitter buffer is available in the receiving terminal.

OPTICOM's POLQA implementation offers you information not only about the overall delay but also about the jitter in terms of the minimum and maximum delay which occurs in the test sample. The positive delay jitter is the difference between the maximum delay and the average delay, while the jitter in the negative direction is the difference between the average delay and the minimum delay.

The delay measurement in POLQA is based on the comparison of the two input files. The accuracy of the delay measurement is therefore only limited by the accuracy with which the recording of the degraded file was synchronized to the playback of the reference file. Ideally, recording and playback start simultaneously. While this can be easily accomplished using one soundcard in one test system, it becomes quite difficult across different interfaces or even machine boundaries.

OPTICOM's POLQA OEM Library returns two different vectors, each containing the delay of each sample of the degraded signal, but using different signals to determine the delay. It is important to understand the difference between the two. One vector calculates the delay which is valid for the input signals of POLQA (with 8 or 48 kHz nominal sample rate). The second vector contains values which are related to the input signals after correction of small sample rate differences. Consequently, if the actual sample rate of the two signals differs by e.g. 2%, then this will add an additional delay of 20ms per second signal length in the first, input signal related vector, while this is typically almost completely compensated in the second vector.

The number of values is the same in both vectors. The sample which is associated with each value in these vectors is the sample with the same index from the degraded signal after the correction of small sample rate differences. Together with the aligned waveforms, which are also generated by the POLQA OEM Library, it is possible to construct charts which show the aligned signals together with the real delay of the input signals.

Min, max and average delay are always related to the input signals and **not** to the signals after the compensation of small sample rate differences.

9.7.3 Attenuation

Especially all analog equipment modifies the level of the speech signal. A high attenuation generally leads to a worse perception of voice. In contrast to PESQ, POLQA **does** weight this as degradation of the signal. Knowing the value of the attenuation is also important for optimizing the overall system design. Attention should be paid to signals which show either a negative attenuation, or attenuations larger than approximately 10dB. In the first case, the signal was amplified instead of attenuated. This may eventually lead to level clipping during the transmission. In the second case, the quantization noise may become an important source of degradation, if low level analog signals are converted to the digital domain and are subsequently amplified in the digital domain. Depending on the test setup, both cases may be ok and intended, but this has to be decided on a case by case basis.

In order to calculate the attenuation, POLQA computes P.56 like active speech levels of the reference as well as the degraded signal in dB. The level of the degraded signal minus the level of the reference signal is then used as the attenuation. Please note that this excludes the pause intervals and that this procedure is different from the one used in OPTICOM's PESQ OEM Library.

9.7.4 Level and Background Noise Measurements

In transmission systems it is frequently important to know the exact levels of the signals. Especially for VoIP systems and voice activity detection (VAD) it becomes also important to know the signal level during the silent intervals as well as during active speech. It is important, that the received background noise does not exceed a certain limit. Levels can be measured in dB if you want to relate the level directly to a sound pressure or electrical level, or as loudness levels. Since the latter takes human perception into account it should be used in preference to the levels in dB. With the OPTICOM POLQA implementation both options are available. Please note, that the loudness as presented by POLQA is directly taken from the intermediate processing steps of POLQA and differs from other loudness definitions as e.g. the Zwicker Loudness. This may change in future versions of the OPTICOM POLQA implementation. All levels are derived using a P.56-like filtering. In order to discriminate between active speech and pause, information from POLQA's temporal alignment is used.

9.7.5 Signal to Noise Ratio (SNR)

OPTICOM's POLQA OEM DLL also calculates the SNR for the reference and the degraded signal independently. The noise as well as the signal level is calculated by the VAD which is part of POLQA's temporal alignment algorithm. Please note that this is completely different from the way in which the SNR is calculated in OPTICOM's PESQ OEM DLL. The result obtained by PESQ indicates the SNR between the reference and the degraded signal, while in POLQA the SNR is calculated for each of the two signals separately based on that signal only (without relation to the other signal).

9.7.6 Active Speech Ratio (ASR)

ASR is calculated by the POLQA OEM DLL based on the information calculated by the Voice Activity Detection (VAD) which is part of the temporal alignment. The ASR defines the ratio between active speech and the overall signal length.

9.7.7 Pitch

POLQA bases some of its analysis on the pitch of the reference and the degraded signal. The pitch frequency of the signals is calculated by analysis of the subharmonics [BEER89] in the signals and subsequently averaging the pitch frequency of individual voiced signal sections. The pitch is given in Hz and separately for the reference and the degraded signal.

9.7.8 Short Term Spectra

The POLQA OEM DLL also calculates the short-term spectrum of each frame. The spectra are available either as line spectra, using a linear Hz scale, or in so-called critical bands on a psycho acoustically motivated pitch scale, using 1/3 bark bands as resolution. The amplitude of the line spectra is the power of the signal at a given frequency in dB. In case of the Bark spectra, the amplitude is the power density of the signal at a given Bark band in dB.

9.8 Reporting and Averaging of Results

Whenever possible, POLQA results should be reported **after** averaging some data points. As a rule of thumb averaging should at least be performed over sentences spoken by two male and two female talkers. The averaging should preferably happen in the MOS domain, although a concatenated speech sample using different voices also provides accurate results. It must however be noted that averaging the MOS LQO of four files leads to slightly

different results than concatenating these four files and performing one measurement. The type of averaging should thus be reported (averaging in the MOS domain or in the signal domain).

Nevertheless, there are some applications that do not allow for such averaging. These include scenarios like drive testing and most tests run on live networks with non-deterministic behaviour. In these cases, no averaging can be performed and per file results must be reported.

If averaging was possible, at least the average, maximum and minimum MOS values should be reported along with the number of samples used for calculating the average. In addition, the 95% confidence intervals should also be mentioned.

If, due to the nature of the experiment, averaging across files was not allowed, the report should contain more elaborated statistics on the MOS values and ideally the distribution of the scores in a graphical form.

9.9 Accuracy of POLQA Results (P.863)

One very common mistake when interpreting POLQA results is to overestimate the accuracy of the results. When presenting MOS-LQO values with a resolution of three decimals, one should always be aware, that the subjects in a listening test have zero decimals available for their votes. Without any further uncertainties and other subjective factors this would result in a theoretical maximum resolution of 0.03 points on the MOS scale if 30 subjects were scoring the same speech sample. In reality, the accuracy of a subjective test will be much worse since very often the subjects disagree on the correct vote. POLQA is however trained on such data and the resulting errors will propagate. The absolute accuracy of the POLQA results depends on the application as well as the number of measurements which are averaged. For a single measurement the Prediction Error will usually be much better than 0.3 MOS.

9.10 Limitations of POLQA

Static delay differences can't be perceived in a listening only experiment. Delay is therefore, although measured by POLQA, not taken into account as a degradation of the signal. Delay variations that happen during active speech will however be weighted properly.

If the delay changes occur during active speech and this leads to significantly extended pauses during the active parts of the speech signal, but e.g. between two phonemes, then this may be over weighted by POLQA and the resulting MOS may be lower than expected. This is currently under research.

9.11 How to Assess "Signal Enhancers"

Signal enhancers or VEDs (Voice Enhancement Devices) are pieces of equipment that try to make the processed signal sound better than the original signal. Examples are e.g. noise reduction systems. If you take the input signal of the enhancer as the reference and the output signal as the test signal of any perceptual measure, the result will usually be the opposite of what you would expect. In general, the enhanced signal will be graded down more, the better your enhancer works. This is due to the fact, that perception-based measurement algorithms assume that any audible difference between the two input signals is a distortion, and by definition the "enhanced" signal will sound different than the unprocessed signal.

In order to come around this, we recommend a setup as shown in Figure 9-4. Here a clean signal is chosen as the reference file and afterwards the signal is distorted artificially, which results in signal D. Subsequently the distorted signal D is sent to the enhancer. The output of the enhancer will be E, the enhanced signal.

If voice quality is assessed now, one should choose the clean reference R and the enhanced signal E as the input signals of POLQA, which will result in MOS_E. The value of MOS_E now indicates how similar the enhanced signal sounds to the clean reference.

Going one step further, one can calculate the gain achieved by the enhancer by relating the final MOS derived this way, to the MOS achieved by comparing the clean reference (R) to the distorted reference (D), resulting in MOS_D. A comparison of MOS_E and MOS_D is then a good measure for the effectiveness f the VED.

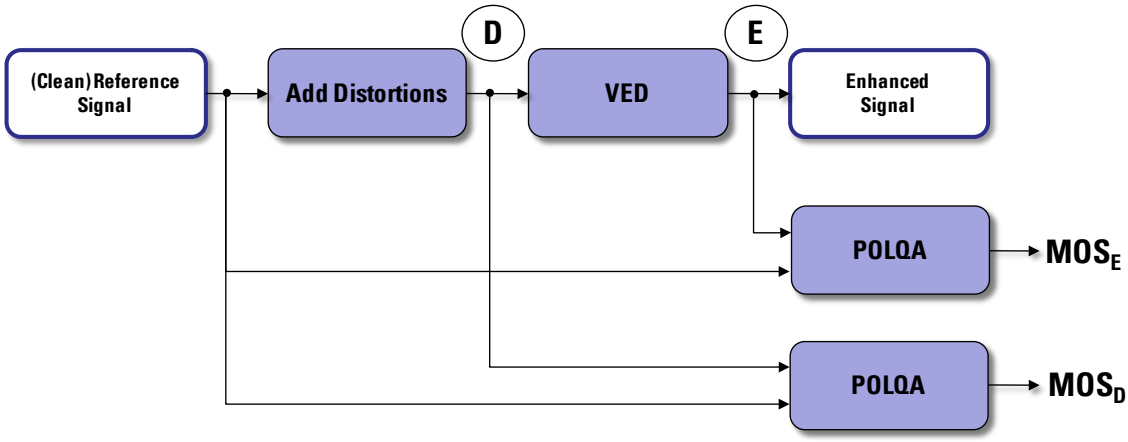


Figure 9-4 Measurement setup for the assessment of Voice Enhancement Devices (VEDs)

9.12 How to deal with Comfort Noise Insertion

Many modern telephony systems use noise substitution or comfort noise insertion. Especially the second case may lead to worse measurement results than expected when reference signals are used that contain digital zero during the silent intervals. POLQA will compare this digital silence to the comfort noise and detect the comfort noise as a distortion.

One can partially compensate for this issue by avoiding digital zero in the silent periods of the reference signals and using a low-level noise floor instead. The level of the noise floor should be in the same range as the expected comfort noise level is. The exact value is best determined in practical experiments.

9.13 Considerations Regarding the Data Acquisition for POLQA

There are two major issues which have to be taken into account when designing a test system that shall perform data acquisition for POLQA. The first issue is related to the accuracy of the clocks used for A/D or D/A converters and the second issue is related to the synchronization of the moments when playing the speech files starts on one side of the connection and recording starts on the other side.

9.13.1 Synchronisation of the Sample Clocks

If the playing and recording side uses different clocks to generate the sample rate, then this appears to POLQA like a variable delay of the system under test. Although POLQA is excellent in handling larger delay variations that occur from time to time as well as drifting delays, we strongly advise using professional soundcard equipment as it is used in studio environment as well. For sample rates higher than 8kHz it is also strongly recommended to synchronize the clocks of the soundcards.

9.13.2 Synchronisation of Play and Record

Besides the quality scores, the OPTICOM POLQA OEM Library also provides delay measurements between the sender and the receiver. This measurement is based on the comparison of the two signals provided to POLQA. It is obvious, that the accuracy of the delay measurement therefore depends on the accuracy of the synchronization of

playing and recording the speech samples. If the transmission started too late, then the delay will be measured larger than it is and if playing the sample started too early, the beginning of the signal will be cut off in the recorded file and the measured delay will therefore be too short.

9.13.3 Control of Playback and Recording Levels

POLQA imposes some very strict constraints regarding the signal levels. Please refer to chapter 9.4.4 for details on this.

9.14 POLQA High Accuracy Mode

In this mode of operation, the POLQA evaluation is repeated n times with the same file pair. Between each repetition, the degraded signal is shifted by s samples. Shifting is performed by removing samples from the start of the degraded file. The resulting n MOS values are then aggregated by first eliminating all MOS values which are at least 1.0 MOS below the best found value. The average of the remaining values is calculated and reported as the MOS-LQO value. Experiments showed, that n=4 and s=0.0025*sample rate are a very good compromise, where the exact value of s is not really important. Larger values of n do of course lead to more stable results, but the biggest step is between n=1 and n=2. Values larger than n=4 lead to very small further improvements which do not justify the additional POLQA calculations.

The red bars in Figure 9-5 below illustrate the effect on the distribution of the residual errors. In order to validate the High Accuracy Mode, it has been applied 64 times to each of the ~45000 samples from the POLQA pool. For each repetition a different start point in the degraded signal was used. It can be seen that there are virtually no values with differences exceeding 0.15 MOS for this worst-case analysis. For comparison reason, the same analysis has been performed without the High Accuracy Mode and the values are shown in the same chart (blue bars). Of course, we also investigated the influence of the High accuracy Mode on the general performance of POLQA. If this mode is applied, the resulting rmse* values for all POLQA pool databases are almost identical, on average they even improve slightly.

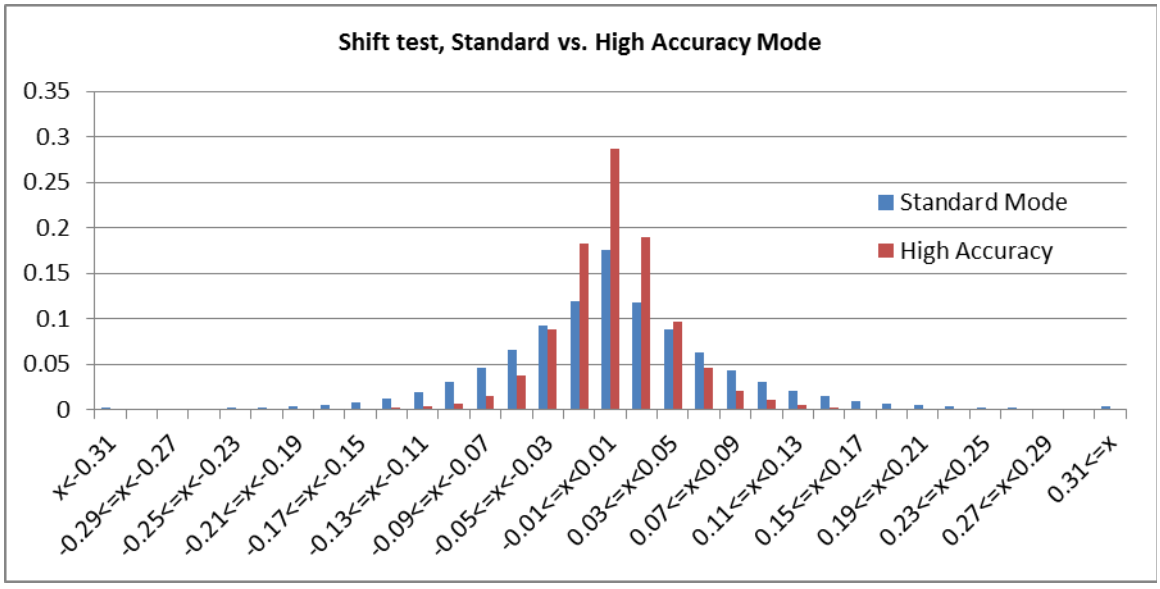


Figure 9-5, comparison of POLQA V2.4 High Accuracy Mode with standard mode (64 shifts per file pair), the basic shift effect is much smaller with POLQA 3 and the resulting distribution is much sharper.

10 The OPTICOM POLQA Library

Using OPTICOM's POLQA library, it becomes a rather simple task to implement POLQA in an OEM product. Since the library is built on the original sources as they were used for the standardization of POLQA, you are also sure that your final product will be fully conforming to the P.863 standard.

The package which you received consists of the following components:

10.1 Windows version (this version is available as evaluation version for download):

- The OEM POLQA DLL
- User credentials and access information for OPTICOM's license management server (LMS)
- An executable demo program
- The demo program as source code, including a ready to compile MS Visual C++ project workspace.
- Some helper modules

10.2 Linux Version (this version is available as evaluation version for download):

- The OEM POLQA lib
- User credentials and access information for OPTICOM's license management server (LMS)
- An executable demo program
- The demo program as source code, including a ready to compile gcc makefile
- Some helper modules

10.3 Android Version (this version is available as evaluation version for download):

- The OEM POLQA dynamic library (file libPolqaOemJava.so)
- User credentials and access information for OPTICOM's license management server (LMS)
- An installation package (file PolqaOemAndroid.zip), that contains a demo program PolqaOemAndroid-release-X.X.apk.
- The demo program as source code, including a ready to compile Android Studio project.
- A helper module (file libIoWrapperDll.so)

10.4 Java Version for Windows or Linux (this version is available as evaluation version for download):

- The OEM POLQA dynamic library
- Windows: PolqaOemJava.dll and PolqaOem.dll
- Linux: libPolqaOemJava.so and libPolqaOem.so
- User credentials and access information for OPTICOM's license management server (LMS)
- A Java jar package (file PolqaOemJava.jar), that contains an executable demo program
- The demo program as source code, including a ready to compile Eclipse project
- A helper module
- Windows: IoWrapperDll.dll
- Linux: libIoWrapperDll.so

10.5 Additional Features of the OPTICOM POLQA DLL

P.863 supports only 8 and 48 kHz sample rate and both signals must be available at the same sample rate. OPTICOM's DLL supports mixed sample rates and automatically converts the input sample rates as required. See section 9.4.4.4.

P.863 is very level sensitive. This is desired for some applications, but causes problems in situations where no exact control over the signal levels is available or where results shall be compared to legacy PESQ measurements. OPTICOM's P.863 implementation therefore offers the possibility of an optional automatic level alignment. See section 9.4.4.3.

The only measurement result standardized by P.863 is the MOS LQO. OPTICOM's POLQA DLL instead provides a wealth of additional KPIs (see 10.18.3.7 for details).

10.6 Correspondence to ITU Standard Versions

10.6.1.1 Versions of P.863

The initial version of the ITU recommendation was V1.0. This version included a few bugs which were corrected in Version 1.1 in November 2011. Both versions produce slightly differing results. Please note that the versioning of the ITU recommendation is different from the version number of the OPTICOM POLQA Library. Starting from the POLQA library version 1.6, the ITU model 1.1 is implemented, and starting from V1.18 both, V1.1 as well as 2015 / V2.4 are implemented. Information on the differences between those versions can be found in section 7.2.3.1. Further breaking changes were introduced with Library V3.2, which, besides numerous minor improvements, introduced fullband mode and an entirely new concept of license management.

10.6.1.2 Mapping to MOS Scale

All commercially available versions of the POLQA Library starting from library Version 1.3 include the mapping to the MOS scale as defined in the P.863 Implementer's Guide from November 2011.

10.7 Supported Platforms

Currently the following operating systems are supported:

- Windows 7 (deprecated)
- Windows 8, 8.1, 10
- Linux
- Android 4.4 up to 8.1 Systems with ARM CPUs
- Java on Windows or Linux

NOTE: The installation of the OPTICOM POLQA OEM Library on virtual machines is **not** supported!

10.8 Software Download

The POLQA OEM Library in all its versions is distributed through OPTICOM's license management server. When ordering a license or an eval version of POLQA you have received user credentials and instructions on how to use them. Please log on to your account and download the latest version as a ZIP archive. The same user credentials will later be required to activate your license.

10.9 Hard- and Software Requirements

10.9.1 Windows, Linux and Java

Currently the POLQA Library supports only INTEL CPUs released after 2008. No problems have been reported from customers running the OPTICOM POLQA library on AMD processors so far, but OPTICOM does not guarantee proper functionality on any CPUs other than INTEL. No special requirements regarding RAM or CPU speed exist. Any modern PC will work.

Using Visual Studio 2005 for linking the POLQA OEM DLL with applications may result in a linker error (e.g. LNK1103). In this case, please make sure to install the following hotfix from Microsoft:

http://support.microsoft.com/kb/949009/en

Redistribution of the DLL will require either the installation of the Visual Studio 2013 runtime library or at least a copy of vcomp120.dll in the path or in the same folder where the DLL resides. The VC runtime can be downloaded from OPTICOM's License Management Server.

10.9.2 Android

The POLQA OEM library has been tested on following mobile phones:

- Samsung Galaxy S5 SM-G901F
- Samsung Galaxy J7 SM-J730F/DS

- Samsung Galaxy S6 SM-G925F
- Samsung Galaxy S7 SM-G930F
- Samsung Galaxy S8 SM-G950F
- Samsung Galaxy S9 SM-G960F

10.9.3 Linux

The POLQA OEM library has been tested on following versions of Linux. Many other platforms work out of the box, but Linux version specific support for other versions than those listed below may be available at extra cost only.

- CentOS 6, 32 bit and 64 bit
- Debian 7, 32 bit and 64 bit
- OpenSuse 12.3, 32 bit and 64 bit
- Ubuntu 12.04, 32 bit and 64 bit

10.10 License Types and License Management

Generally, each copy of the POLQA OEM Library requires a license in order to work. This license is retrieved from OPTICOM's license management server (LMS). All licenses are bound to the device for which they were activated. For a quick start this is typically performed by running the demo program delivered with the POLQA OEM Library with special commandline arguments as explained below.

The exact procedure is explained in detail in separate documentation.

The following license types exist.

- Evaluation license
- Demo Licenses
- Developer Licenses
- Per seat license
- Self-generated licenses

Evaluation licenses are time limited licenses which will expire after a fixed time. Once expired, they cannot be reused.

Demo licenses are time limited licenses which will expire after a fixed time and will fall back into the pool after they expire. They can be reactivated for either the same or another device.

Developer licenses are time limited licenses which will expire after a time which is predetermined by the user at the time of the license generation and will fall back into the pool after they expire. They can be reactivated for either the same or another device.

The per-seat license uses a license key which is bound to a specific machine. You may use as many licenses as are allocated to your license pool on the LMS.

Self-generated licenses are available for customers with a large volume agreement and allow for an unlimited number of licenses to be generated on the LMS by the licensee. For licensees with a small volume contract, OPTICOM will add licenses to the customer specific pool as ordered.

Please consult your license agreement for detailed information!

10.10.1 License Dependent Processing Speed Limitation

As explained in your License Agreement, the type of your license and the licensed number of effective channels will allow you to process a certain amount of speech per time only. If this amount is exceeded, the POLQA OEM Library will suspend processing and automatically continue as soon as a processing budget becomes available again. On Class AA, A, B, or C type systems you should never experience this since the system throughput is limited by the hardware interfaces. If an unlimited, large volume key is installed on your system, then there is also no limitation. For other systems you should purchase additional effective channel licenses if the speed limitation becomes an important issue for you.

Please note that before Version 1.9 of the POLQA OEM Library it was your own responsibility to maintain the proper license count and that you would have had to implement the proper protection scheme. Starting from V1.9 you can rely on the OEM Library to do that job for you as long as you are not using an unlimited large volume key. Large volume customers are still responsible to adhere to the proper license count. However, if they wish, they can at no additional cost request special license keys from OPTICOM in order to leave the speed limitation to the POLQA OEM Library.

The current implementation averages the amount of speech processed during a sliding window of ten minutes. For each licensed effective channel ten minutes speech may be processed during this averaging window. If one channel is licensed and the processing of ten minutes speech takes six minutes only, the POLQA OEM library will pause processing for four minutes after processing the speech signals. During this wait time, any request to process more speech will return an error code when calling the function *PolqaLibRun()*. The required time to wait until requests will succeed again can be queried by calling the function *PolqaLibGetSecondsToWait()*.

10.10.2 Reducing the Processing Time Overhead for the License Management

While the processing time required to validate license keys from license files is very marginal, there is significant time required to detect and evaluate hardware dongles. The overhead for dongles is in the range of few seconds. However, checking for a dongle if there is no dongle attached to the system may take up to two minutes. Typically, this only happens if you have no valid license file installed **and** no valid dongle attached, a situation in which the POLQA calculation would fail anyway. If you want to avoid this, you may call PolqaLibInit() with special parameters that disable the search for hardware dongles (see 10.18.3.1).

10.11 Getting Started on Windows Systems

Note: Microsoft stopped supporting Windows XP and Vista. The OPTICOM POLQA OEM Library has not been tested on Windows XP and Windows Vista. Support for Windows 7 will expire in 2019.

Note: All licenses are bound to a specific device. Make sure that you are performing the registration and

activation step below on exactly the device on which you want to use POLQA!

Note: The description here assumes you are working with a new license. If you want to continue to use an existing POLQA license, omit the step with device registration and license activation!

Please note that this chapter applies to the native version of POLQA only. If you want to install the Java version instead please refer to chapter 10.14.

Installation of the library is very easy. You have to follow two simple steps:

- Copy the contents of the downloaded ZIP archive to any folder of your hard disk. In the following we will call this folder the **POLQA OEM folder**.
- Starting from V1.12 it is required to either install the Visual C++ runtime library (vcredist), or to copy the DLL vcopm120.dll into the directory where the executable is located. Starting from the library version 1.18, both are delivered with the OPTICOM POLQA OEM Library for Windows.
- Make sure your internet connection is working, open a command prompt in the POLQA OEM folder and type (replace the <…> by your user credentials):

  ```
  PolqaOemDemo -Register “<username>” “<password>” “<pool>” –Activate
  ```

  Note: Quotes are required only in case that username, password or pool name contain spaces.

Unless this resulted in any error messages, the machine / device on which you executed this line is now registered at the LMS, added to the pool <pool> and a license was activated. The address of the LMS as well as the type of license to be activated is currently hard-coded in the demo program. The demo program is provided in source code, so both can be adapted to your needs.

10.12 Getting Started on Linux Systems

Note: All licenses are bound to a specific device. Make sure that you are performing the registration and activation step below on exactly the device on which you want to use POLQA!

Note: The description here assumes you are working with a new license. If you want to continue to use an existing POLQA license, omit the step with device registration and license activation!

Please note that this chapter applies to the native version of POLQA only. If you want to install the Java version instead please refer to chapter 10.14.

Installation of the library is very easy. You have to follow three simple steps:

- Start by copying the entire contents of the downloaded ZIP archive to any folder of your hard disk.
- Make sure your internet connection is working, open a command prompt in the folder say PolqaOem64-Linux and type (replace the <…> by your user credentials) with sudo.

  ```
  $ sudo ./PolqaOemDemo64 -Register “<username>” “<password>” “<pool>” –
  Activate
  ```

Unless this resulted in any error messages, the machine / device on which you executed this line is now registered at the LMS, added to the pool <pool> and a license was activated. The address of the LMS as well as the type of license to be activated is currently hard-coded in the demo program. The demo program is provided in source code, so both can be adapted to your needs.

10.13 Getting Started on Android Systems

The OPTICOM POLQA OEM package contains the following Android installation files in the folder "PolqaOemAndroid/install/":

- PolqaOemAndroid-release-X.X.apk
- RefNB.wav and TestNB.wav
- RefSWB.wav and TestSWB.wav

Installation of the library and the demo program can be done by following these steps:

- Make the following settings on your Android device:
- Enable the "Developer options" by tapping seven times on the "Build number", which can be found in "Settings->About device->Software info->Build number" (Samsung Galaxy S7) or in "Settings->About device->Build info" (Samsung Galaxy S6).
- Enable the "USB debugging" option in "Settings->Developer options->USB debugging".
- Enable the "Unknown sources" option in "Settings->Lock screen and security->Unknown Sources".
- The demo program needs an Android device with so called external storage. Depending on the device this can be an extra flash chip on the PCB or a micro SD card. In the latter case a micro SD card needs to be provided by the user. Note: for the Samsung Galaxy S5, S6, S7, S8 and S9 the external storage is an extra flash chip on the PCB so for these devices no micro SD card needs to be provided.

The next items describe how to install the POLQA demo application on your Android device:

- Download Android platform-tools package for Win/Linux from https://developer.android.com/studio/releaes/platform-tools.
- Add the location of the „platform-tools" directory to your system PATH. This lets you run Android Debug Bridge (adb) without needing to supply the full path to the tools directory.
- If you develop on Windows you probably need to install an USB driver specific to your device. For Google Nexus devices follow the instructions given on http://developer.android.com/sdk/win-usb.html. For other devices follow the instructions on https://developer.android.com/studio/run/oem-usb.html.
- If you develop on Linux no device specific USB driver needs to be installed. Instead you need to create an udev rules file (/etc/udev/rules.d/51-android.rules). To create an udev rules file follow the steps given on https://developer.android.com/studio/run/device.html.

- Open a command prompt on your PC and change the working directory to "PolqaOemAndroid/install/". Enter the following command:
- adb install –r PolqaOemAndroid-release-X.X.apk (Installs the POLQA library and the demo program. Please note: you can uninstall the package with the command "adb uninstall de.opticom.polqaoem").
- Now start the application "PolqaOemAndroid" on your Android device ("Application Menu->PolqaOemAndroid"). On startup the application creates the folder /sdcard/opticom/polqa/ if it does not exist and copies the reference and degraded wave files RefNB.wav, TestNB.wav, RefSWB.wav and TestSWB.wav into it.

Note: All licenses are bound to a specific device. Make sure that you are performing the registration and activation step below on exactly the device on which you want to use POLQA!

Note: The description here assumes you are working with a new license. If you want to continue to use an existing POLQA license, omit the step with device registration and license activation!

To download a valid license file, please make sure your device is connected to internet, start "PolqaOemAndroid" app via launcher, press the menu button and select the item "Get License online" (see Figure 10-1c (4)) and enter username, password and license pool info provided by OPTICOM as shown in the Figure 10.1d. Select type of License allocated in the pool and POLQA ITU version (Please note the online license file generated for POLQA ITU – Version 3.0 maintains backward compatibility with ITU-Version 2.4 and not in vice versa) Click on "Get License File" button to register the device and retrieve the license file for the device. The downloaded License file shall be available in the folder "/sdcard/opticom/polqa/"along with server and client certificates (Please refrain from tampering/replacing/moving the certificates on the device).

10.14 Installation of the Java version on Windows or Linux

NOTE: This section is deprecated and will be updated with the final release of POLQA V3!

Installation of Java version on Window and Linux is work in progress. Please skip this section.

Please note that this chapter applies to the Java version of POLQA only. If you want to install the native version instead please refer to chapter 10.11 (Windows) or 10.12 (Linux).

Start by copying the entire contents of the OPTICOM POLQA OEM CD to any folder of your hard disk. In the following we will call this folder the **POLQA OEM folder**.

If your POLQA library is protected by a software key the last step is to copy your license key file "PolqaLicenseFile.txt" to the POLQA OEM folder. You may be able to use the license file from a different directory in your own product later on. How to accomplish this will be explained later. Our demo program however expects the file to be in the POLQA OEM folder.

If you do not yet have a valid license file or a dongle, please start the demo program with the command line option –Mac (i.e. java –jar PolqaOemJava.jar -Mac) and send the generated file "LicenseInfoFile.txt" to OPTICOM in order to obtain a license for your system.

10.15 The Demo Program (Windows, Linux)

10.15.1 Running the Demo Program

The first quick test to verify the installation should be to run the demo program.

In order to do so, on windows systems open a DOS window, go to the POLAQ OEM folder and type:

PolqaOemDemo -Ref RefNB.wav -Test TestNB.wav –LC NB

On Linux systems open a shell and type

$ sudo ./PolqaOemDemo -Ref RefNB.wav -Test TestNB.wav –LC NB

Please note that the –Ref, –Test, -LC switches are case sensitive. Depending on the version which you are using, there may be additional options possible. All existing options will be listed if you start PolqOemDemo without any parameters. The resulting output of the above example should look as below (the most up to date version may differ slightly):

```
[c:]PolqaOemDemo -Ref RefNB.wav -Test TestNB.wav –LC NB

 POLQA Library Version: 1.600

 POLQA   P863 Version : 1.100

 POLQA          Reference File: RefNB.wav
 POLQA    Nr Samples Reference: 50956 (6.370s)
 POLQA             Sample Rate: 8000

 POLQA          Degraded  File: TestNB.wav
 POLQA     Nr Samples Degraded: 51040 (6.380s)
 POLQA             Sample Rate: 8000

 POLQA         Processing Mode: Narrowband

 Warning: For P.863 conformance specifiy -DisableLevelAlignment
             and -DisableSRConv

 POLQA RESULTS
 POLQA                       MOS-LQO: 3.5965
 POLQA                   G107 Rating: 69.988

 POLQA                    MIN  Delay: 4.250ms
 POLQA                    AVG  Delay: 11.762ms
 POLQA                    MAX  Delay: 19.250ms
 POLQA                   Attenuation: 0.720dB

 POLQA               Level Reference: -27.331dBov
 POLQA                Level Degraded: -29.285dBov
 POLQA  Active Speech Lev. Reference: -26.934dBov
 POLQA   Active Speech Lev. Degraded: -27.654dBov
 POLQA         Pause Level Reference: -67.491dBov
 POLQA          Pause Level Degraded: -75.214dBov
 POLQA                 SNR Reference: 40.930dB
 POLQA                  SNR Degraded: 40.836dB
 POLQA Active Speech Ratio Reference: 0.681
 POLQA  Active Speech Ratio Degraded: 0.681

 Pass through data:
```

Note: The exact output may vary with the version of the POLQA Library which you are using!

The demo program can evaluate POLQA for wave files containing either 16bit linear or G.711 (A-law, mu-law) coded speech. The only command line parameters required to start the program are the name of the reference file which contains the unprocessed speech signal, the name of the test file which contains the degraded speech signal and the measurement type.

PolqaOemDemo.exe prints all results calculated by the POLQA OEM library to the screen. The only exceptions are the delay vs. time data and all other vector or matrix results.

The available results are described in detail in Chapter 10.18.

10.15.2 Compiling the Demo Program

10.15.2.1 Windows

For your convenience we provide the complete Visual C++ workspace of the Demo Program including the source code. To open the workspace simply double click on the file PolqaOEMDemo.vcproj. The program consists of the single file PolqaOEMDemo.cpp plus a utility library Iolib\IOLib.lib and of course the POLQA OEM library in the subdirectory PolqaLib.

The iolib.lib handles reading of the wave files as well as processing command line arguments. It is provided on an as is basis, but you may use it in your own programs if you like.

PolqaOEMDemo.cpp contains the main function which first processes the command line arguments, then reads the speech data from wave files into two data vectors, before it calculates POLQA using the OPTICOM POLQA OEM Library.

Please keep in mind that the demo program is for demonstration purpose only and may be changed at any time and without notice. If you want to, you can use the demo program as a starting point for your own application. However, there is no warranty, support or whatsoever for the Demo program available from OPTICOM. Also, no fitness for any purpose despite demoing the OPTICOM POLQA OEM DLL is guaranteed.

Note: There is one warning issued while generating the debug build. This is normal and expected since the POLQA DLL uses the release startup code and the application the debug version. You can safely ignore this warning.

10.15.2.2 Linux

For your convenience we provide the complete gcc makefile of the Demo Program including the source code. The program consists of the single file PolqaOEMDemo.cpp plus a utility library Iolib\libIOLib.a and of course the POLQA OEM library in the subdirectory PolqaLib.

The libiolib.a handles reading of the wave files as well as processing command line arguments. It is provided on an as is basis, but you may use it in your own programs if you like.

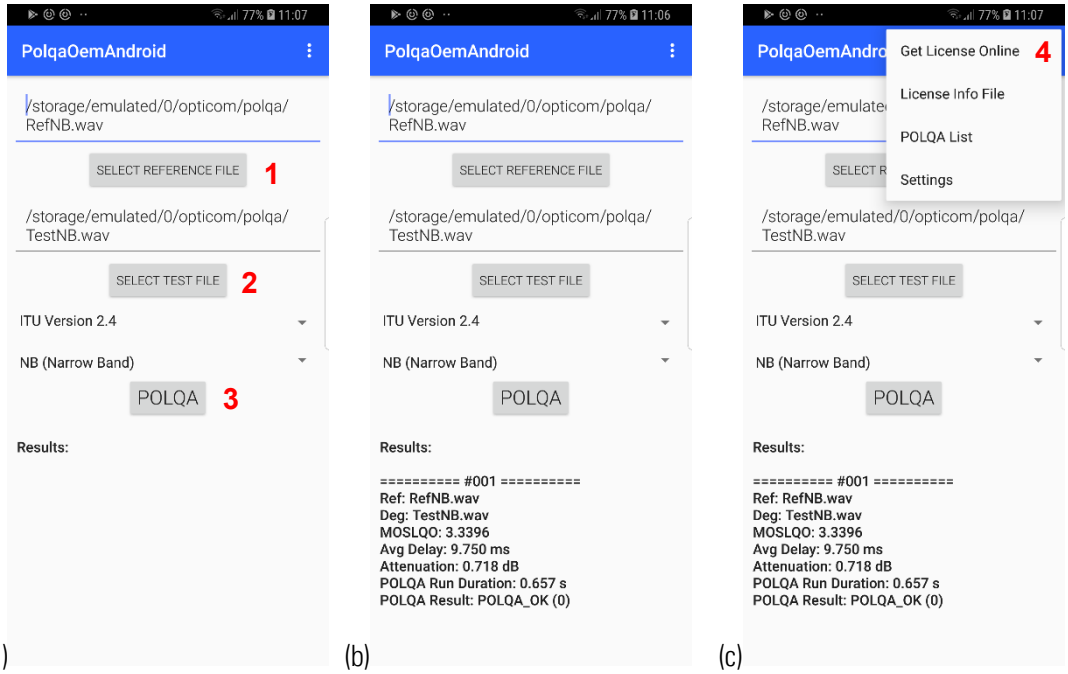
PolqaOEMDemo.cpp contains the main function which first processes the command line arguments, then reads the speech data from wave files into two data vectors, before it calculates POLQA using the OPTICOM POLQA OEM Library.

Please keep in mind that the demo program is for demonstration purpose only and may change at any time without notice.

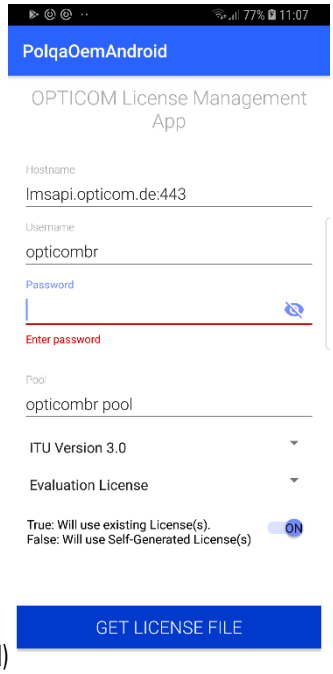
10.16 The Demo Program (Android)

10.16.1 Running the Demo Program

The first quick test to verify the installation should be to run the demo program. For this purpose open the application menu and launch the program PolqaOemAndroid (see Figure 10-1a).



(a) (b) (c)



(d)

Figure 10-1: Android Demo Program PolqaOemAndroid on a Samsung Galaxy S5 (a) before and (b) after POLQA calculation

Press the "Select Reference File" button (1) and select the reference file "RefNB.wav". Press the "Select Test File" button (2) and select the degraded file "TestNB.wav". Then start the POLQA calculation by pressing the "POLQA" button (3). After the POLQA calculation finished, Figure 10-1b appears which displays the results.

If you encountered any problems at this stage you should check the following:

Is an SD card inserted? The license file and the wave files RefNB.wav and TestNB.wav must be installed on the SD card in the folder "/sdcard/opticom/polqa/"

Is there a valid license file "OpticomLicenseFile.txt" in the folder "/sdcard/opticom/polqa/" on your Android phone? If not please read chapter 10.13 for further instructions.

Is your license still valid? You can check the expiration date by opening the license file in a text editor. Although the expiration date is coded in the license key, it is also mentioned as clear text.

Are the wave files RefNB.wav and TestNB.wav installed in the folder "/sdcard/opticom/polqa/" on your Android phone?

The demo program can evaluate POLQA for wave files containing either 16bit linear or G.711 (A-law, mu-law) coded speech.

The demo program PolqaOemAndroid prints some results calculated by the POLQA OEM library to the screen. The available results are described in detail in chapter 10.18. Please note that the DLL provides far more results than those that are shown in our demo program.

10.16.2 Compiling the Demo Program

In order to compile the demo program PolqaOemAndroid, follow these steps:

Open the folder PolqaOemAndroid/ of your OPTICOM POLQA package with Android Studio ("Open an existing Android Studio project").

Rebuild your project.

The program consists of the project PolqaOemAndroid plus a utility library libloWrapperDll.so (binary version only, no source code) and of course the POLQA OEM library (libPolqaOemJava.so, binary version only, no source code).

The libloWrapperDll.so handles reading of the wave files. It is provided on an as is basis, but you may use it in your own programs if you like.

The file app/src/main/java/de/opticom/polqaoem/PolqaOemAndroid.java contains the Android Activity with the graphical user interface. The files app/src/main/java/de/opticom/polqa/PolqaWrapper.java and app/src/main/java/de/opticom/polqa/PolqaResultData.java contain Java wrappers for the native POLQA OEM library.

Please keep in mind that the demo program is for demonstration purpose only and may be changed at any time and without notice. If you want to, you can use the demo program as a starting point for your own application. However, there is no warranty, support or whatsoever for the Demo program available from OPTICOM. Also, no fitness for any purpose despite demoing the OPTICOM POLQA OEM DLL is guaranteed.

10.17 The Demo Program (Java on Windows or Linux)

NOTE: This section is deprecated and will be updated with the final release of the Java version of POLQA V3!

10.17.1 Running the Demo Program

The first quick test to verify the installation should be to run the demo program.

In order to do so, open a command prompt, go to the POLAQ OEM folder and type:

```
java –jar PolqaOemJava.jar -Ref RefNB.wav -Test TestNB.wav –LC NB
```

Please note that the –Ref, –Test, -LC switches are case sensitive. The resulting output should look as below (the most up to date version may differ slightly):

```
$ java -jar PolqaOemJava.jar -Ref RefNB.wav -Test TestNB.wav -LC NB

POLQA Library Version: 1.400

POLQA   P863 Version : 1.000

POLQA          Reference File: RefNB.wav
POLQA    Nr Samples Reference: 66468 (8.309s)
POLQA              Sample Rate: 8000

POLQA          Degraded  File: TestNB.wav
POLQA     Nr Samples Degraded: 68868 (8.609s)
POLQA              Sample Rate: 8000

POLQA         Processing Mode: Narrowband

POLQA RESULTS
POLQA                        MOS-LQO: 1.1901
POLQA                   G107 Rating: 17.774
```

```
POLQA                    MIN  Delay: 326.750ms
POLQA                    AVG  Delay: 462.445ms
POLQA                    MAX  Delay: 650.750ms
POLQA                  Attenuation: -1.773dB

POLQA               Level Reference: -30.425dBov
POLQA                Level Degraded: -28.513dBov
POLQA  Active Speech Lev. Reference: -27.090dBov
POLQA   Active Speech Lev. Degraded: -25.318dBov
POLQA         Pause Level Reference: -69.877dBov
POLQA          Pause Level Degraded: -65.116dBov
POLQA                 SNR Reference: 38.594dB
POLQA                  SNR Degraded: 38.745dB
POLQA Active Speech Ratio Reference: 0.449
POLQA  Active Speech Ratio Degraded: 0.461

Pass through data:
```

Note: The exact output may vary with the version of the POLQA Library which you are using!

If you encountered any problems at this stage you should check the following:

Windows version:

Is the license file in the current folder? The file name must be PolqaLicenseFile.txt.

Is your license still valid? You can check the expiration date by opening the license file in a text editor. Although the expiration date is coded in the license key, it is also mentioned as clear text.

Are IoWrapperDll.dll, PoqaOem.dll and PolqaOemJava.dll available in the current folder?

Are you running the program on an INTEL CPU?

Linux version:

Is the license file in the current folder? The file name must be PolqaLicenseFile.txt.

Is your license still valid? You can check the expiration date by opening the license file in a text editor. Although the expiration date is coded in the license key, it is also mentioned as clear text.

Are libIoWrapperDll.so, libPolqaOemJava.so and libPolqaOem.so available in the current folder?

Is the current folder in the search path of shared libraries? If not, execute the following command before starting the demo program:

```
export LD_LIBRARY_PATH=.
```

Are you running the program on an INTEL CPU?

The demo program can evaluate POLQA for wave files containing either 16bit linear or G.711 (A-law, mu-law) coded speech. The only command line parameters required to start the program are the name of the reference file which contains the unprocessed speech signal, the name of the test file which contains the degraded speech signal and the measurement type.

The demo program prints all results calculated by the POLQA OEM library to the screen. The only exceptions are the delay vs. time data and all other vector or matrix results.

The available results are described in detail in Chapter 10.18.

10.17.2 Compiling the Demo Program

In order to compile the demo program PolqaOemJava, follow these steps:

Copy the folder \PolqaOemJava\ of your OPTICOM POLQA package into your Eclipse workspace. Import these files with Eclipse (File->Import->Existing Projects into Workspace).

Clean your project.

Windows version:

The program consists of the project PolqaOemJava plus a utility library IoWrapperDll.dll (binary version only, no source code) and of course the POLQA OEM library (PolqaOemJava.dll and PolqaOem.dll, binary version only, no source code).

The IoWrapperDll.dll handles reading of the wave files. It is provided on an as is basis, but you may use it in your own programs if you like.

Linux version:

The program consists of the project PolqaOemJava plus a utility library libIoWrapperDll.so (binary version only, no source code) and of course the POLQA OEM library (libPolqaOemJava.so and libPolqaOem.so, binary version only, no source code).

The libIoWrapperDll.so handles reading of the wave files. It is provided on an as is basis, but you may use it in your own programs if you like.

The file src\PolqaOemJava.java contains the command line demo program. The files src\de\opticom\polqa\PolqaWrapper.java and src\de\opticom\polq\PolqaResultData.java contain Java wrappers for the native POLQA OEM library.

Please keep in mind that the demo program is for demonstration purpose only and may be changed at any time and without notice. If you want to, you can use the demo program as a starting point for your own application. However, there is no warranty, support or whatsoever for the Demo program available from OPTICOM. Also, no fitness for any purpose despite demoing the OPTICOM POLQA OEM DLL is guaranteed.

10.18 The POLQA Library

10.18.1 Contents

The POLQA OEM DLL for Windows consists of the following files:

PolqaDllInterface.h	Standard C header file which defines the API. This file must be included in all C/C++ files that want to perform calls to the POLQA OEM API. Since it is a standard C file you must use *extern "C"* if you want to include it in C++ sources.
PolqaOem.lib	This file must be linked to your own code. It defines the exported functions of the DLL.
PolqaOem.dll	This file contains the POLQA library. It must be either in the working directory of your application, or in your system root directory.

The POLQA OEM Lib for Linux consists of the following files:

PolqaDllInterface.h	Standard C header file which defines the API. This file must be included in all C/C++ files that want to perform calls to the POLQA OEM API. Since it is a standard C file you must use *extern "C"* if you want to include it in C++ sources.
LibPolqalib.a	This file must be linked to your own code. It contains the POLQA OEM lib.

The POLQA OEM library for Android consists of the following files:

libPolqaOemJava.so in the folders app/src/main/jniLibs/arm64-v8a/ and app/src/main/jniLibs/armeabi-v7a/	These files contain the POLQA library plus native wrapper functions, which can be called from Java.
PolqaWrapper.java and PolqaResultData.java in the folder app/src/main/java/de/opticom/polqa/	Java wrapper classes, which invoke the native wrapper functions in libPolqaOemJava.so.

The POLQA OEM library for Java consists of the following files:

Windows: PolqaOem.dll and PolqaOemJava.dll Linux: libPolqaOem.so and libPolqaOemJava.so	These files contain the POLQA library plus native wrapper functions, which can be called from Java.
PolqaWrapper.java and PolqaResultData.java in the folder src\de\opticom\polqa\	Java wrapper classes, which invoke the native wrapper functions in PolqaOemJava.dll and PolqaOem.dll (Windows) or libPolqaOemJava.so and libPolqaOem.so (Linux) respectively.

10.18.2 Explanation of the API

The POLQA OEM Library contains two groups of functions. The first group contains functions required to process speech files and to report the results. The second group is responsible for license management. This second group is the same for all OPTICOM products and therefore explained in detail in a separate document.

10.18.2.1 C/C++ API

The API consists of five functions and is very straight forward to use. There is one function for initializing the library (*PolqaLibInit*), one to calculate POLQA (*PolqaLibRun*) and one to free all resources again (*PolqaLibFree*). A handle of type *POLQA_HANDLE* must be passed between all three functions. This handle must be initialized to NULL before calling *PolqaLibInit*. Between the calls to *PolqaLibRun* and *PolqaLibFree* you can access the results. A pointer to the result structure of type *POLQA_RESULT_DATA* can be obtained by calling *PolqaLibGetResultsPointer*. Please note that after calling *PolqaLibFree* this pointer is not valid anymore. Except for *PolqaLibGetResultsPointer* all functions return an error code which is part of the enum *POLQA_ERRORCODE*.

The following code snippet demonstrates well how few lines of code are required in order to use the POLQA DLL in your own application. It assumes that the two data vectors are filled with the reference and the degraded signal and that the sample rate is known. Please note that POLQA is currently standardized for 8kHz, 16kHz and 48kHz sample rate only. For the sake of readability, the initialization as well as any error checking has been removed from the example. The full version including file IO etc. can be found in the file PolqaOEMDemo.cpp which is part of this SDK.

```
#include "PolqaDllInterface.h"
{
   ... fill the data vectors pfRefData and pfDegData with data and set their size in
   iNrRefSamples and iNrDegSamples and the respectice sample rates in
   ulRefSampleRate and ulDegSampleRate

   long PolqaErrorcode;
   POLQA_DLL_HANDLE PolqaHandle=0;
   INSTDATA         InstanceData=0;

   /* Optional: Enable use of dongles for POLQA 2.4 and earlier */
   PolqaLibEnableDongleLegacy(&InstanceData);


   /* Init library */
   PolqaErrorcode=PolqaLibInit(&PolqaHandle, InstanceData, 0,
                               POLQA_LC_STANDARD_IRS);

   /* Calculate POLQA */
   PolqaErrorcode=PolqaLibRun(PolqaHandle,iNrRefSamples, pfRefData, ulRefSampleRate,
                           iNrDegSamples, pfDegData, ulDegSampleRate);


   /* Get and print the results */
   pPolqaRes = PolqaLibGetResultsPointer(PolqaHandle);
   if(pPolqaRes != NULL)
   {
        printf("\n POLQA MOS: %f  ", pPolqaRes->mfMOSLQO);
        ... Print other results here
   }

   /* Close library */
   PolqaErrorcode=PolqaLibFree(&PolqaHandle);
}
```

A simple code sequence to perform device registration and license activation can be implemented as shown below (the code is for windows, on Linux use char instead of wchar_t for strings):

```
#include "PolqaDllInterface.h"

// LMS account specific
wchar_t  OEMDeviceID[MAX_STRING_LEN]   = L"My test device name";
wchar_t  Hostname[MAX_STRING_LEN]      = L"lms.opticom.de";
wchar_t  Password[MAX_STRING_LEN]      = L"MyPassword";
wchar_t  UserName[MAX_STRING_LEN]      = L"MyUserName";
wchar_t  Pool[MAX_STRING_LEN]          = L"YourLicensePool";
wchar_t  LicensePath[MAX_STRING_LEN]   = L"";        // Keep it empty…
// License specific
wchar_t  LicenseType[MAX_STRING_LEN]   = L"Evaluation License 7 days";
wchar_t  LicenseClass[MAX_STRING_LEN]  = L"C";
wchar_t  Version[MAX_STRING_LEN]       = L"3";
int      EffectiveChannels             = 2;
bool     UseExisting = true;       // If true, use licenses already existing in
                                   // the pool, otherwise, generate a new license
                                   // ("self-generated license")
INSTDATA  InstanceData=0;

printf("Registering device...\n");
PolqaErrorcode = PolqaLibRegisterDevice(UserName, Password, Pool, OEMDeviceID,
                                        Hostname);
if (POLQA_OK == PolqaErrorcode)
{
      printf("Activating license...\n");
      PolqaErrorcode = PolqaLibActivateLicense(&InstanceData, LicenseType,
                                               LicenseClass, EffectiveChannels,
                                              Version, UseExisting, Hostname);
      if (POLQA_OK == PolqaErrorcode)
      {
          printf("License activation successful\n");
      }
      else printf(" PolqaLibActivateLicense() failed: %d\n", PolqaErrorcode);
}
else printf(" PolqaLibRegisterDevice() failed: %d\n", PolqaErrorcode);
```

10.18.2.2 Java API (Java or Android)

The API consists of the two Java classes PolqaWrapper and PolqaResultData and is very similar to the C/C++ API (see chapter 10.18.2.1). There is one function for initializing the library (PolqaWrapper.PolqaLibInit()), one to calculate POLQA (PolqaWrapper.PolqaLibRun()) and one to free all resources again (PolqaWrapper.PolqaLibFree()). A handle to the native POLQA library is hidden as a private member of PolqaWrapper. Between the calls to PolqaWrapper.PolqaLibRun() and PolqaWrapper.PolqaLibFree() you can access the results. An object of the result class PolqaResultData can be filled by calling PolqaWrapper.PolqaLibGetResult(). All functions return an error code of type integer.

The following code snippet demonstrates well how few lines of code are required in order to use the POLQA Java API in your own application. It assumes that the two data vectors are filled with the reference and the degraded signal and that the sample rate is known. Please note that POLQA is currently standardized for 8kHz, 16kHz and 48kHz sample rate only. For the sake of readability the initialization as well as any error checking has been removed from the example. Please note also that the third parameter of PolqaWrapper.PolqaLibInit() must be a TelephonyManager object on Android and null on the Java virtual machine.

```
{
   PolqaWrapper polqaWrapper = new PolqaWrapper();
   int result;

   ... fill the data vectors pfRefData and pfDegData with data

   ... Set ulSampleRate to the sample rate used.

   // Init library
   Object tm = getSystemService(TELEPHONY_SERVICE); // on Android
   Object tm = null; // on the Java virtual machine
   result = polqaWrapper.PolqaLibInit(
            "/sdcard/opticom/polqa/PolqaLicenseFile.txt",
            PolqaWrapper.POLQA_LC_STANDARD_IRS,
            tm);

   // Calculate POLQA
   result = polqaWrapper.PolqaLibRun(
            pfRefData, 0, pfRefData.length, ulSampleRate,
            pfDegData, 0, pfDegData.length, ulSampleRate);

   // Get and print the results
   PolqaResultData resultData = new PolqaResultData();
   result = polqaWrapper.PolqaLibGetResult(resultData);
   if(result == PolqaWrapper.POLQA_OK) {
        System.out.format("POLQA MOS: %f\n", resultData.mfMOSLQO);
        ... Print other results here
   }

   // Close library
   polqaWrapper.PolqaLibFree();
}
```

10.18.3 The API Functions in Detail

10.18.3.1 PolqaLibEnableDongleLegacy ()

```
// C/C++
POLQA_ERRORCODE PolqaLibEnableDongleLegacy (INSTDATA* pInstanceData)
```

Explanation:

This function enables support for Dongles to be used to run POLQA 2.4 and earlier. It will have no effect on POLQA 3. Call `PolqaLibEnableDongleLegacy()` before any other POLQA API function. The `INSTDATA` structure instance to which `pInstanceData` points, must be passed to all other used API functions too. It will become invalid after calling `PolqaLibFree()`.

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

Parameters:

pInstanceData	Pointer to an INSTDATA structure.

10.18.3.2 PolqaLibInit()

```
// C/C++
POLQA_ERRORCODE PolqaLibInitUTF8 ( POLQA_HANDLE *PolqaHandle,
                                   INSTDATA InstanceData,
                                   const char  *RegFilename,
                                   unsigned long ulMode)

POLQA_ERRORCODE PolqaLibInitW ( POLQA_HANDLE *PolqaHandle,
                                INSTDATA InstanceData,
                                const wchar_t  *RegFilename,
                                unsigned long ulMode)

POLQA_ERRORCODE PolqaLibInit( … )


// Java
public class PolqaWrapper {
   ...
   int PolqaLibInit(String RegFilename,
                    int ulMode,
                    Object TelephonyManager) {
        ...
   }
   ...
}
```

Explanation:

This function initializes the library and must be called prior to calling any other function from the POLQA API. Depending on the compiler settings, the generic version PolqaLibInit(…) will be mapped to either the UTF8 version PolqaLibInitUTF8(…) or the wide character version PolqaLibInitW(…). The UTF8 or W version may also be called directly.

Note: For ulMode at least POLQA_LC_STANDARD_IRS or POLQA_LC_SWIDE_H must be specified. For all other flags default values can be used. Currently the default for those is:

POLQA_RLEVEL1 | POLQA_RLEVEL2 | POLQA_RLEVEL3

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

Parameters:

PolqaHandle	C/C++ only: Pointer to a variable of type POLQA_HANDLE. This variable must have been initialized to NULL before passing it to PolqaLibInit. In Java this handle is hidden as a private class member of the PolqaWrapper object.
InstanceData	An optional INSTDATA structure as used by the license management routines. May be 0.

RegFilename

String specifying the location of all license specific files. Default is to pass zero and let the library chose a default location.

In order to use old POLQA V2 and V1 license files, this string must contain the full path and file name of the PolqaLicenseFile. This file contains your personal registration key. You can specify any file here, but be sure that the specified file contains your license key. Otherwise the initialization of the library will fail.

TelephonyManager

On Android a TelephonyManager object must be provided, which you can get with a call to Activity.getSystemService(TELEPHONY_SERVICE). On the Java virtual machine null must be provided.

ulMode

0, or a combination of the following flags (Note that only one of POLQA_LC_STANDARD_IRS and POLQA_LC_SWIDE_H may be specified!:

POLQA_LC_STANDARD_IRS	Use narrowband mode
POLQA_LC_SWIDE_H	Use SWB mode (POLQA V1.x and 2.x only) or fullband mode (POLQA V3)
POLQA_RLEVEL1	Calculate a minimum set of results only (fastest).
POLQA_RLEVEL2	Calculate some more results (marginally slower).
POLQA_RLEVEL3	Calculate all results (slower).
POLQA_LEVEL_ALIGN	Automatically align the levels of the reference and the degraded signal. This switches POLQA_RLEVEL2 on too. **Note:** This is non-standard!
POLQA_AUTO_SR_CONVERSION_OFF	Switch off the automatic sample rate conversion for input data with unsuitable sample rates. **Note:** This must be set for conformance with P.863!
POLQA_CHECK_PROCESSING_TIME	Return the processing time in the result structure. **Note 1:** Approximately 2 seconds additional initialization time are required in PolqaLibInit(). **Note 2:** no checks for counter overflows are performed, time

	measurement results are therefore not valid for very long sequences.
POLQA_HA_MODE	Enables the POLQA High Accuracy mode, which averages MOS values from four single POLQA runs on slightly shifted degraded signals (for details see chapter 9.15 and [ITUTG12Cont84], chapter 4.2.1 Proposed High Accuracy Mode).
POLQA_HA_OMP	Enables that the four POLQA runs for the POLQA High Accuracy mode are computed on multiple cores on a multi core CPU. **Note:** this flag is not implemented on Android.
POLQA_V1_1	Calculate results according to P.863 V1.1 (mutually exclusive with other POLQA_Vx_x values)
POLQA_V2_4	Calculate results according to P.863 V2.4 (mutually exclusive with other POLQA_Vx_x values)
POLQA_V3_0	Calculate results according to P.863 V3.0 (mutually exclusive with other POLQA_Vx_x values)

10.18.3.3 PolqaLibRun()

```
// C/C++
POLQA_ERRORCODE PolqaLibRun(POLQA_HANDLE PolqaHandle,
                            int mRefSamples,
                            float *RefData,
                            unsigned long ulRefSampleRate,
                            int mDegSamples,
                            float *DegData,
                            unsigned long ulDegSampleRate )


// Java
public class PolqaWrapper {
   ...
   int PolqaLibRun(float[] RefData, int mRefOffset, int mRefSamples,
                   int ulRefSampleRate,
                   float[] DegData, int mDegOffset, int mDegSamples,
                   int ulDegSampleRate) {
        ...
   }
   ...
}
```

Explanation:

This function performs the actual calculation of the POLQA results. If ulRefSampleRate differs from ulDegSampleRate, both signals will be resampled to 8 kHz (NB mode) or 48 kHz (FB mode). In NB mode all signals which are not at 8, 16 or 48 kHz sample rate will be resampled to 8 kHz. In FB mode all signals will be resampled to 48 kHz. Input signals should be scaled to the range of -32767 to +32767, similar to converting 16 bit linear PCM samples to int values.

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

If POLQA_LIMIT_EXCEEDED is returned, then the amount of processed speech exceeds the the allowed amount defined by the licensed number of channels. In this case all further requests will fail for a certain time. This amount of time can be queried by calling PolqaLibGetSecondsToWait().

If POLQA_LIMIT_INTERNAL_ERROR was returned, the system was not able to properly detect the number of actually used effective channels. In this case the log file must be deleted. Ten minutes after this, request will succeed again. The log file is %PROGRAMDATA%\polqastats.raw on windows machines and /tmp/polqastats.raw on Linux machines. Typically this error would occur when a manipulation of the log file was detected, so please, don't mess around with it!

Parameters:

POLQA_HANDLE PolqaHandle	C/C++ only: The handle which must have been initialized by PolqaLibInit() before.
int mRefSamples	The number of samples contained in the vector with the reference signal.
int mRefOffset	Java only: How many samples to ignore at the beginning of the reference signal.
float *RefData	Data vector with the reference speech signal.
unsigned long ulRefSampleRate	Sample rate of the reference speech signal in Hz.
int mDegSamples	The number of samples contained in the vector with the degraded signal.
int mDegOffset	Java only: How many samples to ignore at the beginning of the degraded signal.
float * DegData	Data vector with the degraded speech signal.
unsigned long ulDegSampleRate	Sample rate of the degraded speech signal in Hz

10.18.3.4 PolqaLibGetResultsPointer()

```
// C/C++
POLQA_RESULT_DATA *PolqaLibGetResultsPointer(POLQA_HANDLE PolqaHandle)


// Java
public class PolqaWrapper {
   ...
   int PolqaLibGetResult(PolqaResultData result) {
        ...
   }
   ...
}
```

Explanation:

After completion of PolqaLibRun() use this function to obtain a pointer to the POLA results (C/C++) or copy the POLQA result values to a PolqaResultDate object (Java). Please note that the pointer returned is only valid until PolqaLibFree() has been called (C/C++).

Returns:

A pointer to the POLQA Result structure (C/C++).

Parameters:

POLQA_HANDLE PolqaHandle	C/C++ only: The handle which must have been initialized by PolqaLibInit() before.

10.18.3.5 PolqaLibFree()

```
// C/C++
POLQA_ERRORCODE PolqaLibFree(POLQA_HANDLE *PolqaHandle)


// Java
public class PolqaWrapper {
   ...
   int PolqaLibFree() {
        ...
   }
   ...
}
```

Explanation:

Frees all resources allocated by the library and invalidates all previous results.

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

Parameters:

POLQA_HANDLE PolqaHandle	C/C++ only: The handle which must have been initialized by PolqaLibInit() before.

10.18.3.6 PolqaLibGetSecondsToWait ()

```
// C/C++
double PolqaLibGetSecondsToWait(POLQA_HANDLE PolqaHandle)


// Java
public class PolqaWrapper {
   ...
   double PolqaLibGetSecondsToWait () {
        ...
   }
   ...
}
```

Explanation:

Returns the number of seconds until requests to PolqaLibRun() will succeed again after a the actually used effective channels exceeded the licensed number of effective channels. This function is optional, but you may want to use it if PolqaLibRun() returned POLQA_LIMIT_EXCEEDED.

Returns:

The required wait time in seconds.

Parameters:

POLQA_HANDLE PolqaHandle	C/C++ only: The handle which must have been initialized by PolqaLibInit() before.

10.18.3.7 PolqaLibGetDelayHistogram()

```
// C/C++
PolqaLibGetDelayHistogram(POLQA_DLL_HANDLE PolqaHandle, int FirstSample,
        int NumSamples, float* pFirstBin, float* pBinWidthExt,
        int* pPeakIndexExt, int* pNumBinsExt, int** ppHistogram);

// Java
public class PolqaDelayHistogram
{
   public float pFirstBin;
   public float pBinWidthExt;
   public int pPeakIndexExt;
   public int pNumBinsExt;
   public int[] ppHistogram;
   // ...
}

public class PolqaWrapper {
   ...
   public int PolqaLibGetDelayHistogram(
                  int FirstSample,
                  int NumSamples,
                  PolqaDelayHistogram histogram)
        ...
   }
   ...
}
```

Explanation:

Calculate a histogram of the per sample delays. The histogram indicates how often each delay occured. The histogram will be allocated and destroyed by the DLL. It lives until PolqaLibFree() is called, or PolqaLibGetDelayHistogram() is called again. The histogram is calculated over a section of the reference signal. This section starts at sample FirstSample of the reference signal and is NumSamples long. The resolution of the histogram (=BinWidth) will be 1 ms and it ranges from pPOLQARes->mfMinDelay to pPOLQARes->mfMaxDelay. If the difference between pPOLQARes->mfMaxDelay and pPOLQARes->mfMinDelay exceeds 10 s, the upper limit will be corrected accordingly. The length of the histogram is thus variable and depends on the delay range as well as the resolution.

All delays are expressed in milliseconds.

Note: PolqaLibRun() must be called at least once before calling PolqaLibGetDelayHistogram().

Returns:

Any of the codes defined by the enum `POLQA_ERRORCODE`.

Parameters:

POLQA_HANDLE PolqaHandle	C/C++ only: The handle which must have been initialized by PolqaLibInit() before.
int FirstSample	Specifies the start of the section over which the histogram shall be calculated. FirstSample is the index of the sample of the reference signal where the section starts. If FirstSample is before the algorithm internal start sample, it will be corrected accordingly.
int NumSamples	Specifies the length of the section over which the histogram shall be calculated in samples. If the specified section ends after the algorithm internal stop point, it will be limited accordingly.
float* FirstBin	The delay corresponding to the first bin
float* BinWidthExt	The width of each bin in ms
int* PeakIndexExt	Index of the bin with the highest probability
int* NumBinsExt	Number of bins in the histogram
int* Histogram	A pointer to the calculated histogram

Example:

In order to find the most frequent delay over the entire input signal, you may use the following code after your call to PolqaLibRun():

```
{
   …
   float FirstBin;
   float BinWidth;
   int PeakIndex;
   int NumBins;
   int* Histogram;
   PolqaErrorcode = PolqaLibGetDelayHistogram(PolqaHandle, 0,
            pPolqaRes->mulElementsInRefAlignedTimeBuffer,
            &FirstBin, &BinWidth, &PeakIndex, &NumBins,
            &Histogram);
   // Get the most frequent delay in milliseconds:
   float MostLikelyDelay = FirstBin + BinWidth * PeakIndex;
}
```

10.18.3.9 PolqaLibRegisterDevice()

```
// C/C++
POLQA_ERRORCODE PolqaLibRegisterDeviceUTF8(
                      INSTDATA*   pInstanceData,
                      const char* Username,
                      const char* Password,
                      const char* Pool,
                      const char* OEMDeviceID,
                      const char* Hostname,
                      const char* Licensefolder=0);

POLQA_ERRORCODE PolqaLibRegisterDeviceW(
                      INSTDATA*   pInstanceData,
                      const wchar_t* Username,
                      const wchar_t* Password,
                      const wchar_t* Pool,
                      const wchar_t* OEMDeviceID,
                      const wchar_t* Hostname,
                      const wchar_t* Licensefolder=0);

POLQA_ERRORCODE PolqaLibRegisterDevice(…)

// Java
public class PolqaWrapper {
   ...

   public int PolqaLibRegisterDevice(
                    String Username,
                    String Password,
                    String pool,
                    String OEMDeviceID,/*Build.MODEL+Build.MANUFACTURER*/
                    String Hostname,
                    Object telephonyManager)
        ...
   }
   ...
}
```

Explanation:

This function registers the current device/machine with the license management server and associates it with a pool of licenses. It must be called prior to activating a license.

Depending on the conmpiler settings, the generic version PolqaLibInit(…) will be mapped to either the UTF8 version PolqaLibInitUTF8(…) or the wide character version PolqaLibInitW(…). The UTF8 or W version may also be called directly.

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

Parameters:

pInstanceData	Pointer to an INSTDATA structure.
Username	Username for access to the LMS
Password	password associated to the username
Pool	Name of the license pool for which the device shall be registered, provide String as "name pool"
OEMDeviceID	Optional text which can be used to identify devices in reports generated by the LMS. May be any arbitrary text.
Hostname	Optional specification of the LMS hostname. Use 0 to choose the default.
Licensefolder	String specifying the location of all license specific files. Default is to pass zero and let the library chose a default location.
TelephonyManager	On Android a TelephonyManager object must be provided, which you can get with a call to Activity.getSystemService(TELEPHONY_SERVICE). On the Java virtual machine null must be provided.

10.18.3.10 PolqaLibActivateLicense ()

```
// C/C++
POLQA_ERRORCODE PolqaLibActivateLicenseUTF8(
                      INSTDATA*   pInstanceData,
                      const char* LicenseType,
                      const char* LicenseClass,
                      const char* LicenseCategory,
                      int         EffectiveChannels,
                      const char* Version,
                      bool        UseExisting,
                      const char* Hostname,
                      const char* Licensefolder=0)

POLQA_ERRORCODE PolqaLibActivateLicenseW(
                      INSTDATA*      pInstanceData,
                      const wchar_t* LicenseType,
                      const wchar_t* LicenseClass,
                      const wchar_t* LicenseCategory,
                      int            EffectiveChannels,
                      const wchar_t* Version,
                      bool           UseExisting,
                      const wchar_t* Hostname,
                      const wchar_t* Licensefolder=0)

POLQA_ERRORCODE PolqaLibActivateLicense(…)


// Java
public class PolqaWrapper {
   ...

   public int PolqaLibActivateLicense(
                    String LicenseType,
                    String Version,
                    String Hostname)
        ...
   }
   ...
}
```

Explanation:

This function downloads a license from the LMS. The device / machine must already be registered for a specific license pool. A license exactly matching the parameters specified in the call to PolqaLibActivateLicense() must exist in the license pool for which the device is registered, otherwise an error code will be returned.

Depending on the compiler settings, the generic version PolqaLibActivateLicense (…) will be mapped to either the UTF8 version PolqaLibActivateLicenseUTF8(…) or the wide character version PolqaLibActivateLicenseW(…). The UTF8 or W version may also be called directly.

Returns:

Any of the values from the POLQA_ERRORCODE enum. A value different from POLQA_OK indicates an error.

Parameters:

pInstanceData	Pointer to an INSTDATA structure.
LicenseType	The license type, exactly as specified in the GUI of the LMS, e.g. "Evaluation License 7 days"
LicenseClass	Allways "1"
LicenseCategory	Category of the license which shall be used. Must be any of "A", "B", "C" or "D" and depends on license agreement as well as desired application. For Android is always “A”
EffectiveChannels	The number of ECs for which this license should be valid. For Android is always “1”
Version	POLQA ITU Version for which this license shall be valid ("3" or "2"). Higher version values will also execute lower versions.
UseExisting	If true, get a free license which already exists in the pool, otherwise generate a new one ("self generated license").
Hostname	Hostname of the license management server. If empty, a default will be used (usually lms.opticom.de).
Licensefolder	String specifying the location of all license specific files. Default is to pass zero and let the library chose a default location.

10.18.3.11 PolqaCreateLicenseKeyInfo()

```
// C/C++
POLQA_ERRORCODE PolqaCreateLicenseKeyInfo(
                      const char *LicenseInfoFilename,
                      const char *LicenseFileName );


// Java
public class PolqaWrapper {
   ...
   int PolqaCreateLicenseKeyInfo( String LicenseInfoFilename,
                                  String LicenseFileName,
                                  Object TelephonyManager) {
        ...
   }
   ...
}
```

Explanation:

This function is currently not supported!

Returns:

Any of the codes defined by the enum `POLQA_ERRORCODE`.

Parameters:

const char *LicenseInfoFilename	Name of the generated file.
const char *LicenseFileName	Name of the license file. May be set to NULL. If you append the string "|DisableDongle" to the variable LicenseFileName then POLQA does not try to access an attached external USB dongle. This would speed up the execution of PolqaCreateLicenseKeyInfo().
Object TelephonyManager	On Android a TelephonyManager object must be provided, which you can get with a call to Activity.getSystemService(TELEPHONY_SERVICE). On the Java virtual machine null must be provided.

10.18.3.12 POLQA Result Data Structure

POLQA_RESULT_DATA - Results Returned by POLQA

This structure holds all the results as they are derived by the POLQA Library. It is defined as follows:

```
typedef struct
{
   float mfVersion;
   float mfP863Version;

   unsigned long   mulMode;
   unsigned long mulSampleRate;

   float mfMOSLQO;
   float mfMinDelay;
   float mfMaxDelay;
   float mfAvgDelay;
   float mfEModelRValue;

   float mfPitchReference;
   float mfPitchDegraded;

   float mfEstimatedSampleRate;
   int miResamplingApplied;

   /*-------------------------------------------------------------*/
   /* Values below are computed only if POLQA_RLEVEL2 was specified!
   */

   float mfLevelReference;
   float mfLevelDegraded;

   float mfP56ActiveSpeechLevelRefdB
   float mfP56ActiveSpeechLevelDegdB
   float mfP56PauseLevelRefdB;
   float mfP56PauseLevelDegdB;

   float mfAttenuation;
   float mfSnrReference;
   float mfSnrDegraded;


   float mfActiveSpeechRatioRef;
   float mfActiveSpeechRatioDeg;

   /* VAD Info */
   unsigned long mulElementsInVADVectors;
   int *mpiVADSpeechStartRef;
   int *mpiVADSpeechStartDeg;
   int *mpiVADSpeechLengthRef;
   int *mpiVADSpeechLengthDeg;

   /*-------------------------------------------------------------*/
   /* Values below are computed only if POLQA_RLEVEL3 was specified!
   */

   unsigned long mulElementsInDelayBuffer;
   float *mpfDelayVsTimeBuffer;
```

```
   unsigned long mulElementsInRefAlignedTimeBuffer;
   float *mpfRefAlignedTimeBuffer;
   unsigned long mulElementsInDegAlignedTimeBuffer;
   float *mpfDegAlignedTimeBuffer;

   unsigned long mulMOSPerFrameBufferSize;
   float *mfpMOSPerFrame;

   unsigned long mulFrameSize;
   unsigned long mulNrFramesInSpectrogram;
   unsigned long mulNrLinesInSpectra;
   float** mppfSpectrumRef;
   float** mppfSpectrumDeg;
   unsigned long mulNrBarkBands;
   float** mppfLoudnessDensityRef;
   float** mppfLoudnessDensityDeg;
   float** mppfNoiseLoudnessGraph;

   unsigned long mulWarningSatus;

   double dDeltaTime;

} POLQA_DLL_RESULT_DATA;
```

float mfVersion	Ten times the first two digits of the version id of the POLQA OEM lib which created this structure. E.g. 10 = V1.0.x
float mfP863Version	Ten times the first two digits of the version id of the ITU refernce code to which the processing was conforming. E.g. 10 = V1.0.x
float mfMOSLQO	POLQA score according to P.863
unsigned long mulElementsInDelayBuffer	Number of samples in the delay vs. frames buffer.
float *mpfDelayVsTimeBuffer	Delay vs. samples buffer. Buffer containing the delay in ms of each sample of the reference signal.
float mfMinDelay	The minimum delay between the reference and the test signal in ms
float mfAvgDelay	The average delay between the reference and the test signal in ms
float mfMaxDelay	The maximum delay between the reference and the test signal in ms
float mfEModelRValue	The POLQA score mapped as a G.107 (E-Model) Ie value. **Note:** This value is valid for narrowband measurements only.
float mfPitchReference	The average pitch frequency of the reference signal.
float mfPitchDegraded	The average pitch frequency of the degraded signal.
float mfEstimatedSampleRate	The sample rate of the degraded signal as measured by POLQA
int miResamplingApplied	If the sample rates of the reference and the degraded signal differ by more than 0.5%, POLQA will downsample the signal with the higher sample rate. If this happens, miResamplingApplied will be set to 1.

float mfLevelReference	The Level of the reference signal in dB (averaged over the entire signal)
float mfLevelDegraded	The Level of the degraded signal in dB (averaged over the entire signal)
float mfP56ActiveSpeechLevelRefdB	The active speech level of the reference signal in dBov, measured similar to P.56.
float mfP56ActiveSpeechLevelDegdB	The active speech level of the degraded signal in dBov, measured similar to P.56.
float mfP56PauseLevelRefdB	The silence level of the reference signal in dBov, measured similar to P.56.
float mfP56PauseLevelDegdB	The silence level of the degraded signal in dBov, measured similar to P.56.
float mfSnrReference	The SNR in dB of the reference signal
float mfSnrDegraded	The SNR in dB of the degraded signal
float mfAttenuation	The attenuation between the two input signals in dB
float mfActiveSpeechRatioRef	The active Speech Ratio (ASR) of the reference signal. ASR is the ratio of the length of the active speech signal parts and the total signal length [0..1].
float mfActiveSpeechRatioDeg	The active Speech Ratio (ASR) of the degraded signal. ASR is the ratio of the length of the active speech signal parts and the total signal length [0..1].
unsigned long mulElementsInVADVectors	Number of active speech sections in the input signals. This is identical to the number of valid elements in the mpiVADSpeechXXX vectors. Note that section *j* of the reference signal always corresponds to section *j* of the degraded signal.
int *mpiVADSpeechStartRef	For each active speech section of the reference signal the index of the first sample.
int *mpiVADSpeechStartDeg	For each active speech section of the degraded signal the index of the first sample.
int *mpiVADSpeechLengthRef	For each active speech section of the reference signal the length of the section in samples.
int *mpiVADSpeechLengthDeg	For each active speech section of the degraded signal the length of the section in samples.
unsigned long mulElementsInDelayBufferInp	The number of valid elements in the vector mpfDelayVsTimeBufferInp and mpfDelayVsTimeBufferRes
float *mpfDelayVsTimeBufferInp	Float vector which contains for each frame of the degraded signal the offset to the associated signal section in the reference signal in samples. Related to the input signals without correction of small sample rate differences.
float *mpfDelayVsTimeBufferRes	Float vector which contains for each frame of the degraded signal the offset to the associated signal section in the reference signal in samples. Related to the input signals including the correction of small sample rate differences.
unsigned long mulElementsInRefAlignedTimeBuffer	The number of valid elements in the vector mpfRefAlignedTimeBuffer
float *mpfRefAlignedTimeBuffer	The sample-wise reference signal after the temporal alignment and potential resampling and level alignment.
unsigned long mulElementsInDegAlignedTimeBuffer	The number of valid elements in the vector mpfDegAlignedTimeBuffer

float *mpfDegAlignedTimeBuffer	The sample wise reference signal after the temporal alignment and potential resampling. If POLQA_LEVEL_ALIGN was specified, the signal is also level aligned.
unsigned long mulMOSPerFrameBufferSize	The number of valid elements in the vector mfpMOSPerFrame
float *mfpMOSPerFrame	MOS-LQO for each frame of the signals. Please note that this is an approximated value only and that it is not limited to the interval [1; 4.75], which means the value can be negative.
unsigned long mulFrameSize	The size of each processing frame in samples. Frames overlap by 50%.
unsigned long mulNrFramesInSpectrogram	The number of frames in the following arrays mppfSpectrumXXX, mppfLoudnessDensityXXX and mppfNoiseLoudness.
unsigned long mulNrLinesInSpectra	The number of lines for each frame of the spectra in mppfSpectrumXXX
float** mppfSpectrumRef	The spectrum of the reference signal with the dimensions mppfSpectrumRef[mulNrFramesInSpectrogram][mulNrLinesInSpectra]
float** mppfSpectrumDeg	The spectrum of the degraded signal with the dimensions mppfSpectrumDeg[mulNrFramesInSpectrogram][mulNrLinesInSpectra]
unsigned long mulNrBarkBands	The number of (1/3) bark bands stored in the arrays mppfLoudnessDensityXXX and mppfNoiseLoudness
float** mppfLoudnessDensityRef	The loudness density of the reference signal with the dimensions mppfLoudnessDensityRef [mulNrFramesInSpectrogram][mulNrBarkBands]
float** mppfLoudnessDensityDeg	The loudness density of the degraded signal with the dimensions mppfLoudnessDensityDeg [mulNrFramesInSpectrogram][mulNrBarkBands]
float** mppfNoiseLoudnessGraph	The measured noise loudness with the dimensions mppfNoiseLoudnessGraph [mulNrFramesInSpectrogram][mulNrBarkBands]
unsigned long mulWarningStatus	Bitmap for Warnings
double dDeltaTime;	Processing time of the PolqaRun function. Requires POLQA_CHECK_PROCESSING_TIME to be set.

The **mpfDelayVsTimeBufferInp** can be used to plot the delay of the two signals vs. time. An example code snippet that prints the values of this vector can be found below:

```
{
   unsigned int i

   printf("\n Delay History Size %lu \n",
        pPolqaRes->mulElementsInDelayBuffer);

   printf("0 \t %f\t",pPolqaRes->mpfDelayVsTimeBuffer[0]);

   for(i=1;i<pPolqaRes->mulElementsInDelayBuffer;i++)
   {
        printf("%f\t",pPolqaRes->mpfDelayVsTimeBuffer[i]);
        if(i%10==0)printf("\n%lu\t",i);
   }
}
```

The **mulWarningStatus** contains information about potential problems related to the usability of the input files, especially their conformance with the requirements of the ITU-T recommendation P.863. The information is stored as a bitmap with following elements:

Define	Value	Meaning
POLQA_NO_WARNINGS	0x00	No Warnings
POLQA_WARNING_REFERENCE_TOO_LONG	0x01	The reference signal was longer than allowed by P.863
POLQA_WARNING_DEGRADED_TOO_LONG	0x02	The degraded signal was longer than allowed by P.863
POLQA_WARNING_REFERENCE_LEVEL_HIGH	0x04	The level of the reference signal was higher than allowed by P.863
POLQA_WARNING_DEGRADED_LEVEL_HIGH	0x08	The level of the degraded signal was higher than allowed by P.863
POLQA_WARNING_REFERENCE_LEVEL_LOW	0x10	The level of the reference signal was lower than allowed by P.863
POLQA_WARNING_DEGRADED_LEVEL_LOW	0x20	The level of the degraded signal was lower than allowed by P.863

10.18.3.12.1 Scaling of the Spectra, Loudness and Noise Loudness

In order to convert the value of a single frequency line to dB_{fs} use the formula:

dBfs = 10.0 * log10(max(pPolqaResult->mppfSpectrumRef[i][j], 1)) - 109.00171885936986;

10.18.3.13 POLQA Error Codes (enum POLQA_ERRORCODE)

```
typedef enum
{
   POLQA_OK=0,                          // Everything is ok
   POLQA_MEMORY_ALLOCATION_FAILED=1,    // Out of memory
   POLQA_REGISTRATION_FAILED=2,         // Invalid license key
   POLQA_NO_LICENSE=3,                  // Failure during init.
   POLQA_CALCULATION_FAILED=4,          // Failure during POLQA
                                        // calculation
   POLQA_CREATE_LICENSE_INFO_FAILED=5,      // Failure creating the
                                        // license info file
   POLQA_INPUT_SIGNALS_TOO_LONG=6,      // The length of the input files
                                        // exceeds the maximum allowed
                                        // number of samples
  POLQA_INPUT_SIGNALS_TOO_LONG=7,       // the length of the input files
                                        // is less than the minimum
                                        // required number of samples
   POLQA_SAMPLE_RATE_NOT_SUPPORTED=8, // a sample rate different
                                        // from 8, 16kHz, 48kHz
                                        // has been specified
   POLQA_WRONG_HANDLE=9,                // Invalid handle passed to the
                                        // API functions
   POLQA_FILE_OPEN_FAILED=10,           // Failed to open a file
   POLQA_RESULT_LEVEL_TOO_LOW=11,       // Results required which were
                                        // not calculated. See result
                                        // level in PolqaLibInit().
   POLQA_LIMIT_EXCEEDED=12,             // The licensed effective channel
                                        // count is exceeded.
   POLQA_LIMIT_INTERNAL_ERROR=13,       // A problem with the determination
                                        // of the used effective channel count
                                        // was detected (see PolqaLibRun()).
   POLQA_COMMAND_LINE_FAILED,           // Invalid command specified in the
                                        // PolqaOEM executable
   POLQA_LAST_ERROR=14

   POLQA_OK=0,                          // Everything is ok
   POLQA_MEMORY_ALLOCATION_FAILED=1,    // Out of memory
   POLQA_REGISTRATION_FAILED=2,         // Invalid license key
   POLQA_NO_LICENSE=3,                  // Failure during init.
   POLQA_CALCULATION_FAILED=4,          // Failure during POLQA
                                        // calculation
   POLQA_CREATE_LICENSE_INFO_FAILED=5,      // Failure creating the
                                        // license info file
   POLQA_INPUT_SIGNALS_TOO_LONG=6,      // The length of the input files
                                        // exceeds the maximum allowed
                                        // number of samples
  POLQA_INPUT_SIGNALS_TOO_LONG=7,       // the length of the input files
                                        // is less than the minimum
                                        // required number of samples
   POLQA_SAMPLE_RATE_NOT_SUPPORTED=8, // a sample rate different
                                        // from 8, 16kHz, 48kHz
                                        // has been specified
   POLQA_WRONG_HANDLE=9,                // Invalid handle passed to the
                                        // API functions
   POLQA_FILE_OPEN_FAILED=10,           // Failed to open a file
   POLQA_RESULT_LEVEL_TOO_LOW=11,       // Results required which were
                                        // not calculated. See result
                                        // level in PolqaLibInit().
```

```
  POLQA_LIMIT_EXCEEDED=12,          // The licensed effective channel
                                    // count is exceeded.
  POLQA_LIMIT_INTERNAL_ERROR=13,    // A problem with the determination
                                    // of the used effective channel count
                                    // was detected (see PolqaLibRun()).
  POLQA_COMMAND_LINE_FAILED=14,     // Invalid command specified in the
                                    // PolqaOEMDemo executable

  POLQA_ACCESS_VIOLATION=15,
  POLQA_EXCEPTION=16,
  POLQA_FP_EXCEPTION=17,
  POLQA_MAX_UPSAMPLING_EXCEEDED=18,
  POLQA_REF_LEVEL_TOO_LOW=19,
  POLQA_DEG_LEVEL_TOO_LOW=20,
  POLQA_LICFOLDER_REQUIRED=21,
  POLQA_LICENSE_RENEWAL_REQUIRED=22,

  // Device / LMS communication (device registration or activation)
  REG_NO_MATCHING_LICENSE=101,      // No free license of requested type
                                    // available on LMS
  REG_LICENSE_EXPIRED=102,          // The device has an eval associated
                                    // on the LMS, which is already
                                    // expired
  REG_INVALID_LICENSING_PARAMETERS=103,    // Invalid parameters were
                                           // specified when registering a
                                           // device or activating a
                                           // license
  REG_DEVICE_REGISTRATION_FAILED=104,
  REG_LICENSE_ACTIVATION_FAILED=105,
  REG_CANNOT_CONTACT_SERVER=106,    // The LMS did not respond under
                                    // the specified URL
  REG_WRONG_CA=107,                 // The root CA(s) specified in
                                    // the root CA bundle are
                                    // invalid
  REG_INVALID_CERTIFICATE=108,      // The client uses a certificate
                                    // which is not registered on
                                    // the LMS (or no certificate)
  REG_CANT_SAVE_FILE=109,
  REG_REQUEST_TIMEOUT=110,          // The request timed out (server
                                    // busy).

  // HTTP response codes mapped to POLQA errors
  REG_ErrBadRequest=150,
  REG_ErrForbidden=151,
  REG_ErrNotFound=152,
  REG_ErrMethodNotAllowed=153,
  REG_ErrNotAcceptable=154,
  REG_ErrInternalServerError=155,
  REG_HTTP_UNKNOWN_RC=156,
  REG_PAGE_NOT_FOUND=157,

  // Local errors (checking the local license file)
  REG_NO_VALID_LICENSE=200,         // The local licensefile is
                                    // invalid

  POLQA_LAST_ERROR = 201
}POLQA_ERRORCODE;
```

10.18.4 Using POLQA in your own Programs

10.18.4.1 C/C++ API

If you want to use POLQA in your own programs, all you have to do is:

Link PolqaOem.lib (libPolqalib.a on Linux) with your own code

Copy the files PolqaOem.DLL (Windows only) and PolqaLicenseFile.txt to the proper locations

Include the header file PolqaInterface.h in your own sources.

Please note that the POLQA OEM Library is written in ANSI C and not in C++. This means that you have to use the "extern" keyword if you want to include the header file into a C++ source file. The syntax would look as below:

```
extern "C"
{
   include "PolqaInterface.h"
}
```

10.18.4.2 Java API (Android)

If you want to use POLQA in your own programs, all you have to do is:

Copy the "PolqaOemAndroid/app/src/main/jniLibs/" folder, which contains arm64-v8a/libPolqaOemJava.so and armeabi-v7a/libPolqaOemJava.so, to your own Android project.

Copy the files PolqaWrapper.java, PolqaResultData.java and PolqaDelayHistogram.java from the PolqaOemAndroid/app/src/main/java/de/opticom/polqa folder into your own Android project.

10.18.4.3 Java API (Windows or Linux)

If you want to use POLQA in your own programs, all you have to do is:

Copy the files PolqaOem.dll and PolqaOemJava.dll (Windows) or libPolqaOem.so and libPolqaOemJava.so (Linux) respectively to the proper locations

Copy the files src\de\opticom\polqa\PolqaWrapper.java and src\de\opticom\polqa\PolqaResultData.java into your own Android or Java project.

10.18.5 Summary of the Results

10.18.5.1 MOS for narrow-band signals according to P.863 (MOS-LQO)

Indicator	Unit	Value Range	Description	Variable Name
Overall MOS-LQO (P.863)	MOS-LQO	1 – 4.5	MOS-LQO according to P.863 representing the overall quality (narrowband mode only)	mfMOSLQO

10.18.5.2 MOS for super-wideband / fullband signals according to P.863 (MOS-LQO)

Indicator	Unit	Value Range	Description	Variable Name
Overall MOS-LQO (P.863)	MOS-LQO	1 – 4.75 (V1.x and 2.x), 1 – 4.8 (V3.x)	MOS-LQO according to P.863 representing the overall quality (super-wideband / fullband mode only)	mfMOSLQO

10.18.5.3 Transmission Distortions and Delay

Indicator	Unit	Value Range	Description	Variable Name
Minimum Transmission Delay	ms	$\|\lambda\| < \infty$	Shortest occurring delay between reference and test signal. **ideal** value < 300ms	mfMinDelay
Maximum Transmission Delay	ms	$\|\lambda\| < \infty$	Largest occurring delay between reference and test signal. **ideal** value < 300ms	mfMaxDelay
Average Transmission Delay	ms	$\|\lambda\| < \infty$	Average delay between reference and test signal. **ideal** value < 300ms	mfAvgDelay
Attenuation of Test Signal	dB	[-100; 100] dB	Level difference between reference and test signal **ideal** value between [0; 10]dB **typical** value between [-6; 17]dB	mfAttenuation

10.18.5.4 Information on the Reference Signal

Indicator	Unit	Value Range	Description	Variable Name
Level of Reference Signal	dBov	[–100;0] dBov	Level of the reference signal (entire signal, speech plus pause) **ideal** value is-26 dBov **typical** range [-35; -17] dBov (avg. -30 dBov)	mfLevelReference
Active Speech Level (ASL)	dBov	[–100;0] dBov	Average level of the active speech sections of the reference signal measured similar to P.56 **ideal** value is-26 dBov **typical** range [-35; -17] dBov (avg. -30 dBov)	mfP65ActiveSpeechLevelRefdB
Pause Level	dBov	[–100;0] dBov	Average level of the pause sections of the reference signal measured similar to P.56. This value should be very low. **ideal** value is >45 dBov	mfP65ActiveSpeechLevelRefdB
Active Speech Ratio (ASR)	n.a.	[0;1]	The ratio between the length of active speech parts and the total signal length	mfActiveSpeechRatioRef,
Signal to Noise Ratio (SNR)	dBov	[100;10] dBov	The ratio between the noise level and the signal level of the reference signal. Ideally this number is very high. **ideal** value >45 dB	mfSnrReference

10.18.5.5 Information on the Degraded Signal

Indicator	Unit	Value Range	Description	Variable Name
Level of Test Signal	dBov	[–100;0] dBov	Level of the test signal **ideal** value is -26dBov **typical** range [-40; -20] dBov (avg. -33 dBov)	mfLevelDegraded, mfP65ActiveSpeechLevelDegdB
Active Speech Level (ASL)	dBov	[–100;0] dBov	Average level of the active speech sections of the degraded signal measured similar to P.56 **ideal** value is-26 dBov **typical** range [-35; -17] dBov (avg. -30 dBov)	mfP65ActiveSpeechLevelDegdB
Pause Level	dBov	[–100;0] dBov	Average level of the pause sections of the degraded signal measured similar to P.56. This value should be very low. **ideal** value is >45 dBov	mfP65ActiveSpeechLevelDegdB
Active Speech Ratio (ASR)	n.a.	[0;1]	The ratio between the length of active speech parts and the total signal length	mfActiveSpeechRatioDeg
Signal to Noise Ratio (SNR)	dBov	[100;10] dBov	The ratio between the noise level and the signal level of the degraded signal. Ideally this number is very high. **ideal** value >45 dB	mfSnrDegraded

Please note that the values of the degraded signal should correspond to the values found in the reference signal, for example the signal length and the speech activity measure.

10.18.5.6 VAD Information

Indicator	Unit	Value Range	Description	Variable Name
Voice Activity Information	Sample index	0..signal length in samples	Information on where in the signals sections with active speech start and end.	mulElementsInVADVectors, mpiVADSpeechStartRef, mpiVADSpeechStartDeg, mpiVADSpeechLengthRef, mpiVADSpeechLengthDeg

10.18.6 Limits for Input Signals

There are certain limits for input signals which must be met in order to perform measurements which meet the requirements of P.863. If these limits are violated, the accuracy of the results may be limited. If those limits are exceeded, the POLQA OEM Library will return the following warning flags in the result structure:

Criterion	Threshold NB	Threshold SWB	Flag
Duration of ref signal too long	> 30 s	>30 s	POLQA_WARNING_REFERENCE_TOO_LONG
Duration of deg signal too long	> 30 s	>30 s	POLQA_WARNING_DEGRADED_TOO_LONG
Leading silence of signals is too short	0.2 s	0.2 s	POLQA_WARNING_REFERENCE_LEADING_SILENCE_TOO_SHORT
Active speech level of ref signal too high	> -23 dB	>-23 dB	POLQA_WARNING_REFERENCE_LEVEL_HIGH
Active speech level of ref signal too low	< -29 dB	<-29 dB	POLQA_WARNING_REFERENCE_LEVEL_LOW
Active speech level of deg signal too high	> -23 dB	>-20 dB	POLQA_WARNING_DEGRADED_LEVEL_HIGH
Active speech level of deg signal too low	< -29 dB	<-46 dB	POLQA_WARNING_DEGRADED_LEVEL_LOW

Apart from the above-mentioned limits, there are also certain limits which have to be met under all circumstances since POLQA will simply not work if they are violated. In such cases, the POLQA OEM Library will return an error code. The table below lists those limits.

Criterion	Threshold NB	Threshold SWB	Error Code
Duration of ref signal too long	> 120 s	> 120 s	POLQA_INPUT_SIGNALS_TOO_LONG
Duration of deg signal too long	> 120 s	> 120 s	POLQA_INPUT_SIGNALS_TOO_LONG
Duration of ref signal too short	< 1 s	< 1 s	POLQA_INPUT_SIGNALS_TOO_SHORT
Duration of deg signal too short	< 1 s	< 1 s	POLQA_INPUT_SIGNALS_TOO_SHORT

10.18.7 Performance

The OPTICOM POLQA library is optimized for speed. The average performance for narrowband is more than **12 times faster than real-time** on an INTEL Core i5 machine with 2.27GHz under Windows 7. Depending on the input data, performance varies significantly. The processing times measured contain the entire time spent in the POLQA OEM DLL. The time required for opening and reading the wave files is excluded. Table 10-1 gives an overview of the performance of the library for different sampling rates. The numbers are valid for P.863 V2.4 / 2015. V1.1 is even faster. Table 10-2 shows the equivalent for the Android version of the library. These numbers are still for POLQA V1.1.

	Intel Core i5 2.27GHz (M340)	**INTEL Core i7, 3.4GHz,(i7 4770)**
NB	12.5	21.1
SWB	5.6	11

Table 10-1: Performance of the OPTICOM POLQA OEM Library V1.18. The numbers indicate how often a 10 s long file can be processed in 10 s.

	Sony Ericsson Xperia X10i, Android 2.3.3	Google Nexus One, Android 2.3.6	Samsung Galaxy S2 GT-I9100, Android 2.3.3
fs=8kHz (NB)	1.05	1.05	2.22
fs=48kHz (SWB)	0.37	0.38	0.78

Table 10-2: Performance of the OPTICOM POLQA OEM Library V1.6 on some Android phones. The numbers indicate how often a 10 s long file can be processed in 10 s. **These numbers are still for V1.1.** V2015 / 2.4 is considerably slower! This table will be updated as soon as new benchmark results become available.

All Performance numbers are given in processing time depending on the speech file duration, measured over some speech files. E.g. 10 means execution is ten times faster than real-time, 0.5 means processing takes twice as long as the duration of the speech files is. Only the pure processor times are measured, excluding program start and file IO. Please note that the processing time of POLQA is very dependent on the degradations in the file and may vary by up to 100%. The numbers above are averages which include such cases.

11 Conformance to ITU-T P.863

The OPTICOM POLQA Library is tested for conformance with ITU-T Rec. P.863 and on all supported platforms 100% conforming to the standard. However, the OPTICOM POLQA Library provides some advanced features which are often required and helpful in practical applications, but which must be disabled when running the conformance test. The features in question are the automatic level alignment and the automatic sample rate correction, which must both be switched off.

The automatic level alignment is a feature which is only meant for cases where the user has no control over the signal levels during the data acquisition. The automatic level alignment should only be used if signal level differences shall not be counted as degradations.

The automatic sample rate correction will resample all input signals to either 8 or 48 kHz sample rate, depending on the operational mode of POLQA. P.863 does not support any other sample rate. However, the conformance test which is part of P.863 requires processing of some input files with 16 kHz sample rate. This is a contradiction within the ITU recommendation itself and already noted by ITU-T Study Group 12. Changing it does require a revision of the recommendation and this will not happen very soon. For the time being, automatic sample rate conversion must therefore be switched off when running the conformance test. Note that the automatic sample rate correction is independent from the POLQA internal resampling of time scaled signals. The internal resampling shall not and can't be switched off.

The following chapters explain how to run the OPTICOM POLQA Library in ITU conforming mode.

11.1 POLQA DLL

When calling PolqaLibInit() (see 10.18.3.1), make sure that the parameter ulMode has

- POLQA_AUTO_SR_CONVERSION_OFF set and
- POLQA_LEVEL_ALIGN **not** set and
- POLQA_HA_MODE **not** set.

11.2 POLQA Demo Program

When calling the demo program from the command line specify the following switches in addition to your other needs:

-DisableLevelAlignment -DisableSRConv

12 Release Notes

12.1 Breaking Changes

12.1.1 Release 3.1

- New license protection scheme
- Some API functions have been modified in order to support the new license protection scheme.
- Additional error codes
- Dongle support is disabled by default and not available for POLQA 3. See 10.18.3.1 for details on how to enable legacy dongle support for POLQA 2.4 and earlier versions.

12.2 Important Changes

12.2.1 Release 3.3

- Fixed multithreading behavior
- Added some error codes

12.2.2 Important Changes in Release 3.1

- POLQA V3
- New license protection scheme

12.2.3 Important Changes in Release 1.29

- Bug fixes.

12.2.4 Important Changes in Release 1.27

- This is an Android only release.
- Change: converted the project files from Eclipse to Android Studio.
- New: added support for platform arm64-v8a.
- Change: removed support for platform armeabi.
- The supported platforms are now arm64-v8a and armeabi-v7a.
- Bug fixes.

12.2.5 Important Changes in Release 1.22

- A bug was fixed which caused a crash under Linux if reference and test signal contained no speech.
- Following a new Implementor's Guide for ITU-T P.863 the (POLQA-internal) reverb indicator is now switched off in SWB mode. This leads to more stable and consistent results, especially in the high quality range and for AMR-WB. **This change is important and impacts some measurement results, which were outliers before!**
- Fix for input files longer than 40 s.
- The licensing module now supports up to 128 different unique IDs (e.g. MAC addresses) and a not hard limited number of related IP addresses.
- Some minor bug fixes.

12.2.6 Important Changes in Release 1.18

- Results according to P.863 2011 / V1.1 as well as P.863 2015 / V2.4. Which version is used can be defined by setting a flag when calling PolqaLibInit(). This has important impact on the sample rates. Make sure you read section 9.4.4.4!
- Now two delay vectors are returned in the result structure. One is related to the sample rate corrected signals and one related to the input signals. The old vector mpfDelayVsTimeBuffer has been renamed to mpfDelayVsTimeBufferRes. The new one is called mpfDelayVsTimeBufferInp. Make sure you read section 9.7.1 and understand the difference between the two!
- Min, max and average delay are now related to the input signals and not to the sample rate corrected signals.
- General bug fixes
- Android demo app is using multiple cores if multiple input file pairs are specified.
- Windows: The library is now compiled with Visual Studio 2013. Using older versions of Visual Studio to link the DLL with applications may require a hotfix from Microsoft. See section 10.8.1 for details. Also, redistribution of the DLL now requires one DLL from the Visual Studio 2013 runtime library (vcomp120.dll).
- Support for Windows XP is deprecated.
- Support for Windows 2000 is terminated.

12.2.7 Important Changes in Release 1.12

- New: Implementation of the POLQA High Accuracy mode, which averages MOS values from four single POLQA runs on slightly shifted degraded signals (for details see chapter 9.15 and [ITUTG12Cont84], chapter 4.2.1 Proposed High Accuracy Mode).
- New (API): New flag POLQA_HA_MODE for the mode parameter of the PolqaLibInit() function, which enables the POLQA High Accuracy mode. New flag POLQA_HA_OMP, also for the mode parameter, which enables that the four POLQA runs are computed on multiple cores on a multi core CPU. Please note that the POLQA_HA_OMP flag is not implemented on Android.
- New (Demo program): New command line option –EnableHaMode, which turns on the POLQA High Accuracy mode.
- Change: Improved sample rate conversion. In case the samplerates of reference and degraded signal differ, the improved sample rate conversion leads to different MOS values.

12.2.8 Important Changes in Release 1.11

- Fix: POLQA for Android was not working on a Samsung Galaxy S4. The creation of a License Info File failed and the POLQA calculation failed with the error message REGISTRATION_FAILED.

12.2.9 Important Changes in Release 1.10

- New: If you had started POLQA on Windows with an expired license, it took about two minutes until PolqaLibInit() finished and returned the error code POLQA_REGISTRATION_FAILED. The reason was that POLQA tried to access an external USB license dongle which takes long. Now the user can disable the access to an USB dongle completely which speeds up the mentioned use case tremendously. See 10.18.3.1 and 10.18.3.6.
- Fix: In V1.9, the results mfMinDelay and mfMaxDelay were swapped. This also leads to errors in the delay histogram calculation.

12.2.10 Important Changes in Release 1.9

- New: automatic monitoring of the number of licensed effective channels and the amount of speech processed per time (see 10.9.1).
- Change: More accurate determination of speech level during pauses
- Change: All delay information during speech pauses is ignored in the min max and average delay calculation as well as the delay histogram data.
- Fix: In older versions it was possible to request a higher result level (e.g. 3) without including the lower levels. This has been changed to automatically include all levels below the requested level as well.
- Change: POLQA does now also support older CPU types again. This increases the size of the DLL, but eliminates many compatibility issues.

12.2.11 Important Changes in Release 1.7 (this is a highly recommended major update)

- New: Windows 64 bit version available
- New: Linux 64 bit version available
- Fix: Fixed crash for long input signals.
- New: The result structure now includes warnings if signals violate P.863 requirements
- Change: Improved internal error handling
- Change: Error Code list updated (POLQA_ERRORCODE)
- Fix: Handling of short files
- Change: Update of the IO library (used in demo executable only)
- Various small internal bug fixes

12.2.12 Important Changes in Release 1.6 (this is a highly recommended major update)

Please note that the names of the binary files have been aligned with the names of our PESQ library in this version!

- New: Update to ITU Version 1.1 (Nov. 2011). Note: Results obtained by this version may differ slightly from results obtained by ITU Version V1.0 (which is implemented in previous library versions).
- New: VAD information included in the result structure
- New: Optional calculation of delay histograms
- New: Code is re-entrant now
- Change: Lower memory usage
- Change: length of result vectors is now consistent
- Change: Improved performance regarding processing speed
- New (demo program): New switch –DisableSRConv to disable the automatic correction of sample rates which do not conform to p.863. This flag must be set to pass the ITU conformance test!
- Change (demo program): Renamed –LevelOff option to -DisableLevelAlignment. This flag must be set to pass the ITU conformance test!

12.2.13 Important Changes in Release 1.4

- Android version only: PolqaWrapper.PolqaLibGetResult() now returns all result values (corresponding to POLQA_RLEVEL1, POLQA_RLEVEL2 and POLQA_RLEVEL3). Previous versions returned result values corresponding to POLQA_RLEVEL1 and POLQA_RLEVEL2 only.

12.2.14 Important Changes in Release 1.3

- New MOS scale. The absolute values are now much closer to the values achieved by PESQ and subjective tests.
- Bug fix in P.56-like level measurements.
- Further optimisation of the processing time

12.2.15 Important Changes in Release 1.1

- Fix for bug in automatic level alignment

12.2.16 Important Changes in Release 1.0

- Stable API
- Many new results
- Full conformance with ITU P.863 (under AAP)
- Significantly enhanced speed and stability

12.2.17 Important Changes in Release 0.9

- Initial release.

13 Known Issues

- POLQA Scores varying with small static delays

  It is known that POLQA scores may vary unexpectedly with small static delay changes (i.e. inserting a small amount of silence before the degraded signal starts). This is a shortcoming of the ITU recommendation and not a problem of the POLQA OEM Library. To overcome this, it is recommended to perform several (>4) measurements for each condition and select the best result. This problem has been recognized and the POLQA coalition has significantly improved this with every update of the recommendation.

14 The Future Roadmap

Of course, we are permanently improving our POLQA library, as well as we are developing new algorithms for different purposes in the field of perceptual voice, audio and video quality measurement.

15 References

[BEER89] J. G. BEERENDS, Pitches of simultaneous complex tones, Chapter 5: A stochastic subharmonic pitch model, Ph.D. dissertation, Technical University of Eindhoven, April 1989.

[BEER92] BEERENDS J. G., STEMERDINK J. A., *A perceptual audio quality measure based on a psychoacoustic sound representation*, J. Audio Eng. Soc., Vol. 40, No. 12, pp. 963-987, 1992

[BEER94] BEERENDS J. G., STEMERDINK J. A., *A perceptual speech quality measure based on a psychoacoustic sound representation*, J. Audio Eng. Soc., Vol. 42, No. 3, pp. 115-123, 1994

[BEER95] BEERENDS J. G., Measuring the Quality of Speech and Music Codecs, an Integrated Psychoacoustic Approach, 98th AES Convention, Paris 1995, Preprint #3945

[BRAN87] BRANDENBURG K., *Evaluation of Quality for Audio Encoding at low Bit Rates*, 82nd AES Convention, London 1987, Preprint #2433

[BRAN89] BRANDENBURG K., Ein Beitrag zu den Verfahren und der Qualitätsbeurteilung für hochwertige Musikcodierung, Ph.D. Thesis, Erlangen 1989

[BRAN97] BRANDENBURG K., BOSI M., *Overview of MPEG Audio: Current and Future Standards for Low-Bit-Rate Audio Coding*, J. Audio Eng. Soc., Vol. 45, No. 1/2, pp. 4-21, 1997

[BRAN92] BRANDENBURG K., SPORER Th.*: 'NMR' and 'masking flag': Evaluation of Quality using Perceptual Criteria*, Proc. of the 11th International AES Conference on Audio Test and Measurement, Portland 1992, pp. 169-179

[COLO95] COLOMES C., LEVER M., RAULT J.B., DEHERY Y.F., *A perceptual model applied to audio bit-rate reduction*, J. Audio Eng. Soc., Vol. 43, pp. 233-240, 1995

[ETSI96] ETSI Technical Report ETR 250, Transmission and Multiplexing (TM); Speech communication quality from mouth to ear for 3,1 kHz handset telephony across networks, ETSI 1996

[FLAN72] FLANAGEN J. L., *Speech Analysis, Synthesis and Perception,* Springer, Berlin - Heidelberg - New York, 1972

[GILC96] GILCHRIST N., GREWIN Ch. (Editors), *Collected Papers on Digital Audio Bitrate Reduction,* AES Special Publication, AES 1996

[HERR92a] HERRE J., EBERLEIN E., SCHOTT H., BRANDENBURG K., *Advanced Audio Measurement System using Psychoacoustic Properties*, 92th AES Convention, Vienna 1992, Preprint #3332

[HERR92b] HERRE J., EBERLEIN E., SCHOTT H., SCHMIDMER Ch., *Analysis Tool for Realtime Measurements using Perceptual Criteria*, Proc. of the 11th International AES Conference on Audio Test and Measurement, Portland 1992, pp.180-190

[ISO97] ISO/IEC/JTC1/SC29/WG11 Draft Document N1557, *Evaluation Methods and procedures for MPEG-4 tests*, 1997

[ITUR96] ITU-R Doc. 10-4/14, *Report on the fourth meeting of TG 10/4 (Stockholm)*, Chairman's Report, 1996

[ITUT96a] ITU-T Contribution COM12-74-E, Review of Validation Tests for Objective Speech Quality Measures, March 1996

[ITUR562] ITU-R Recommendation BS.562-3, *Subjective assessment of sound quality*

[ITUR1116] ITU-R Recommendation BS.1116-1, Methods for the Subjective Assessment of small Impairments in Audio Systems including Multichannel Sound Systems, 1997

[ITUR1387] ITU-R Recommendation BS.1387, Method for Objective Measurements of Perceived Audio, 1998

[ITUT501] ITU-T Recommendation P.501, http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-P.501Test signals for use in telephonometry Annex B, 2004

[ITUT800] ITU-T Recommendation P.800, Methods for subjective determination of transmission quality, 1996

[ITUT810] ITU-T Recommendation P.810, *Modulated Noise Reference Unit (MNRU),* 1996

[ITUT830] ITU-T Recommendation P.830, Subjective Performance Assessment of Telephone-Band and Wideband Digital Codecs, 1996

[ITUT861] ITU-T Recommendation P.861, Objective Quality measurement of telephone-band (300 - 3400 Hz) speech codecs, 1996

[ITUT862] ITU-T Recommendation P.862, PESQ an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs, February 2001

[ITUT862.2] ITU-T Recommendation P.862.2, Wideband extension to P.862 for the assessment of wideband telephone networks and speech codecs, Oct. 2005

[ITUT862.3] ITU-T Recommendation P.862.3, Application Guide for Objective Quality Measurement Based on Recommendation P.862, Oct. 2005

[ITUT863] Draft ITU-T Recommendation P.863, Perceptual Objective Listening Quality Assessment (POLQA) , November 2010

[ITUTG12Cont84] ITU-T, Study Group 12, Contribution 84: Effects of static delay and analysis window positioning on POLQA MOS scores, December 2013

[ITUTSupp23] Supplement 23 to the ITU-T P-series Recommendations, *ITU-T coded-speech database*,1998

[JEKO00] Jekosch U., *Sprache hören und beurteilen*, Habilitationsschrift am Fachbereich 3 der Literatur und Sprachwissenschaften der Universität/Gesamthochschule Essen, 2000

[KARJ85] KARJALEINEN M., A New Auditory Model for the Evaluation of Sound Quality of Audio Systems, Proc. of the ICASSP 1985, pp. 608-611

[KAST07] Co-Editors of P.PCQ., *Initial Draft Recommendation P.CQO-L*, ITU-T Contribution TD 49, 2007

[KEYH93] KEYHL M., HERRE J., SCHMIDMER Ch., *NMR Measurements of Consumer Recording Devices Which Use Low Bit-Rate Audio Coding*, 94th AES Convention, Berlin 1993, Preprint #3616

[KEYH96] KEYHL M., HERRE J., SCHMIDMER Ch., *NMR Measurements on Multiple Generations Audio Coding*, 96th AES Convention, Amsterdam, 1994, Preprint #3803

[KEYH98a] KEYHL M., SCHMIDMER Ch., HERRE J., HILPERT J., Maintaining Sound Quality - Experiences and Constraints of Perceptual Measurements in Today´s and Future Network", 98th AES Convention, Paris, 1995, Preprint #3946

[KEYH98b] KEYHL M., SCHMIDMER Ch., SPORER Th., PETERSON R., Quality Assurance Tests of MPEG Encoders for a Digital Broadcasting System (Part 2) - Minimizing Subjective Test Efforts by Perceptual Measurements, 104th AES Convention, Amsterdam, 1998, Preprint #4753 (P16-7)

[PAIL92] PAILLARD B., MABILLEAU P., MORISETTE S., SOUMAGNE J., *PERCEVAL: Perceptual evaluation of the quality of audio signals*, J. Audio Eng. Soc., Vol. 40, 21-31, 1992

[0OPTI97] PA&SQM PC-Software, Version 6.0, *User Manual*, OPTICOM 1997

[PRAC98] PRACHT St., *Voice Quality*, COMMUNICATE, November 1998, p. 43-46

[SEIT89] SEITZER D., BRANDENBURG K., KAPUST R., EBERLEIN E., GERHÄUSER H., KRÄGELOH S., SCHOTT H.: *DSP based real time implementation of an advanced analysis tool for audio channels*, Proc. ICASSP'89, pages 2057-2060, 1989

[SPOR96] SPORER Th., Evaluating Small Impairments with the Mean Opinion Scale - Reliable or Just a Guess?, 101st AES Convention 1996, Preprint #4396 (E-1)

[SPOR95a] SPORER Th., BRANDENBURG K., *Constraints of Filter Banks Used for Perceptual Measurement,* J. Audio Eng. Soc., Vol. 43, No. 3, 1995 March, pp. 107-116

[SPOR95b] SPORER Th., GBUR U., HERRE J., KAPUST R., *Evaluating a Measurement System*, J. Audio Eng. Soc., Vol. 43, No. 5, 1995 May, pp. 353-363

[SPOR97] SPORER Th., Objective Audio Signal Evaluation - Applied Psychoacoustics for Modeling the Perceived Quality of Digital Audio, 103rd AES Convention, New York, 1997 Preprint #4512

[SPOR98] SPORER Th., KEYHL M., SCHMIDMER Ch., PETERSON R., Quality assurance tests of MPEG encoders for a digital broadcasting system (Part 1) - A method for subjective assessments of very-low bit-rate audio, 104th AES Convention, Amsterdam, 1998

[TERH79] TERHARDT E., *Calculating Virtual Pitch*, Hearing Research, Vol. 1, 1979, p. 155-182

[THIE96] THIEDE Th., KABOT E., *A New Perceptual Quality for Bit Rate Reduced Audio*, 100th AES Convention, Copenhagen, 1996, Preprint #4280

[ZWIC67] ZWICKER E., FELDTKELLER R., *Das Ohr als Nachrichtenempfänger*, Hirzel-Verlag, Stuttgart, 1967

[ZWIC82] ZWICKER E., *Psychoakustik*, Springer-Verlag, Berlin - Heidelberg - New York, 1982

16 Contact Information

OPTICOM

Dipl.-Ing. M. Keyhl GmbH

Nägelsbachstrasse 38

D - 91052 Erlangen

GERMANY

Phone: +49 (0) 91 31 - 5 30 20 - 0

Fax: +49 (0) 91 31 - 5 30 20 - 20

EMail: info@opticom.de

Webseite: http://www.opticom.de

Further information:

http://www.polqa.info

http://www.opticom.de