# “**PyCQ**” Beta version 1.0.0.0 Release Notes

**@bout**:

“PyCQ” is simple Windows voice chat application allow central (Admin) PC (so called ‘Caller’ application) connecting remote stations on Local Area Network (LAN) and send short voice messages / alerts in unicast, multicast or broadcast mode. PyCQ ‘Station’ application should be installed on remote computers in order maintaining communication with ‘Caller’ PC and play received messages / alerts.

**Requirements**:

HW: Any 64-bit PC with 4Gb RAM is sufficient for running PyCQ in normal manner (8Gb recommended), 70MB of free HD space for installation, minimal display resolution >= 640x480 pixels, sound card.

SW: Microsoft Windows 10 and higher editions required to run this SW (Win-7 is incompatible with this build, in order to install on Win-7 PC code must be reworked for using recent python version =<3.8 and PySide5 / PyQt5 libraries).

**What’s in this build?**

Specific functionalities were developed and implemented in this version:

*Communication:* UDP sockets for sending/listening to “caller advertise” packets, simple TCP application stack for maintaining signaling functionalities (including registration, connection restore and keep-alive procedures), simple HTTP Server serving voice messages to remote clients

*Audio:* applications automatically find and select preferred input/output audio HW based on specific PC configuration. Implemented setting audio stream volume, muting/unmuting non-application active audio streams during playback, audio recording and playback in various formats.

*Application logic*: Selection of single/multiple remote station for broadcast, message/alert types, message concurrency, message history and stations db.

*GUI*: introduced 2 applications – ‘Caller’ (windows main-form app.) and ‘Station’ (resident tray icon app.) ‘Station’ application initiates graphical popup followed by audio message/alarm prompt. Station popups are independent non-modal widgets running in separate processes that support multi-instancing and allows user to replay received messages and/or close each popup independently as well as sound playback implemented with threading and allows playing several concurrent streams in parallel. ‘Caller’ application window allows operator observe available connected stations list, select specific station/multiple stations for sending voice messages, sending predefined “system alerts” e.g., “Fire Alert”, “Intruder Alert”, “Missile Attack”, “Earthquake and Tsunami Alerts” in broadcast mode to all available stations with single button click. It also provides convenient recoding and sending voice message with single button push.

*Infrastructure*: Logging support – since PyCQ apps planned as standalone UI apps with no console the need of system log for debug was obvious. Custom logger class was developed using standard python logger libraries and running with both applications’ runtime, all debug print()s were disabled after finishing debugging the app in PyCharm IDE. Logging level can be adjusted by simply changing parameter in “constants” file as well as simple “log rotator” and clen-up procedures were implemented avoiding log and Message Store oversizing. System constants – most application parameters can be configured in ‘constants.py’ allowing better and faster rework and future implementations (see below):

*# ------------------------------------------------------------------------------------------------------*

*# constants.py - Main project constants (like logging level, file names, tcp\_port numbers etc.)*

*#*

*# Prerequisites: None*

*#*

*# Beta release: 10.07.2023 - MichaelZ*

*# ------------------------------------------------------------------------------------------------------*

*import logging*

*# Logging*

*LOGGING\_LEVEL = logging.DEBUG # Minimal desired logging level [default = DEBUG (maximal)]*

*LOG\_FILE = 'PyCQ.log' # Log file name*

*LOG\_MAX\_LINES = 1000 # Max number of log file lines after initial clean-up on application start*

*LOG\_ROTATOR\_COUNTER = 20 # Num of periodic register/keep-alive rounds before log is cleaned (1 round ~35-40 sec)*

*# Communications*

*HTTP\_PORT = 8080 # HTTP port (used by Caller HTTP server)*

*TCP\_PORT = 1234 # TCP port (used for Caller<->Station signaling)*

*UDP\_PORT = 50000 # UDP port (used by 'Announcer' module on caller and 'Listener' module on Station*

*MAGIC = 'py1234cq' # UDP 'magic' packet*

*ANNOUNCE\_INTERVAL = 5 # Caller announcement interval [default = 5 sec]*

*KEEP\_ALIVE\_INTERVAL = 30 # "Station Keep-Alive" interval used by Caller [default = 30 sec]*

*STATION\_REGISTER\_INTERVAL = 30 # "Periodic Register" interval used by Station [default = 30 sec]*

*# Audio*

*SAMPLERATE = 44100 # Audio recording/playback sample rate [default = 44100 Hz]*

*CHANNELS = 2 # number of audio playback channels [default = 2 (stereo)]*

*REC\_TIME = 5 # Voice Message recording length [default = 5 sec]*

*AUDIO\_TYPE = '.ogg' # Supported file types are: '.wav', '.flac' and '.ogg' [default = '.ogg']*

*OUTPUT\_VOLUME = -3.3 # Desired output audio device volume setting in dB (valid range -61.0 -0.0) default -3.3 (~80% of maximum)*

*PLAY\_C2A = True # Play "call to attention" preamble sound before playing new voice message [default = True]*

*C2A\_FILE = 'c2a.ogg' # "Call to attention" sound file*

*ALERT\_SOUND = 'emergency\_alarm.ogg' # sound file for system alerts*

*# Files and folders*

*MESSAGE\_STORE = 'MsgStore' # Filestore directory (in case of standalone ".exe" precompiled in file)*

*RESOURCE\_FOLDER = 'resources' # Resources directory (in case of standalone ".exe" precompiled in file)*

*HISTORY = 'history.csv' # message history database file (in case of standalone ".exe" precompiled in file)*

*HISTORY\_MAX\_ENTRIES = 100 # Max number of history entries after initial clean-up on Caller start*

*MESSAGE\_STORE\_MAX\_FILES = 100 # Max number of stored audio files after initial clean-up on Caller start*

*STATIONS = 'stations.csv'*

*# GUI settings*

*TRAY\_ICON = 'resources/icon.png' # tray icon file for Station tray application*

*TRANSPARENCY = 255 # Default value = 255 (non-transparent). May be changed through 'Change popup transparency' menu of tray icon*

*ENABLE\_CHANGE\_TRANSPARENCY = True # When False slider popup transparency control in station will be disabled and user will be forced default value*

*ENABLE\_CHANGE\_VOLUME = True # When False slider volume control in station will be disabled and user will be forced to OUTPUT\_VOLUME value*

*ENABLE\_EXIT = True # When False 'Exit' option in station disabled (user cannot exit application)*

**Added value features:**

Convenient user interface allows dragging station message popups across the screen / multiple screens. It also allows the station user to configure the audio volume and popups transparency as well as exit the application directly from its tray icon menu. In some cases, administrators will want restricting users from changing message volume, tricking popup transparency or exiting station app – in this case there are specific (true/false) parameters introduced in ‘constants.py’ that can be easily changed and entire application recompiled in a moment and selected options will be disabled in UI.

Caller application tabs can be placed in any order convenient to caller user. Caller application also allows replaying last X messages stored in history (the max number of messages before clen-up also can be configured in ‘constants.py’.

**Build Environment:**

Windows 10 x64 PC with PyCharm running python 3.11.4 ; install PySide6 and PyQt6; install pre-requisites libraries you can use requirements.txt provided in GitHub along the source code:

*altgraph==0.17.3*

*certifi==2023.5.7*

*cffi==1.15.1*

*charset-normalizer==3.2.0*

*comtypes==1.2.0*

*idna==3.4*

*numpy==1.25.1*

*pefile==2023.2.7*

*pipreqs==0.4.13*

*psutil==5.9.5*

*PyAudio==0.2.13*

*pycaw==20230407*

*pycparser==2.21*

*pyinstaller==5.13.0*

*pyinstaller-hooks-contrib==2023.5*

*PyQt6==6.5.1*

*PyQt6-Qt6==6.5.1*

*PyQt6-sip==13.5.1*

*PySide6==6.5.1.1*

*PySide6-Addons==6.5.1.1*

*PySide6-Essentials==6.5.1.1*

*pywin32==306*

*pywin32-ctypes==0.2.2*

*requests==2.31.0*

*shiboken6==6.5.1.1*

*sounddevice==0.4.6*

*soundfile==0.12.1*

*urllib3==2.0.3*

Use ‘Pyinstaller’ app to ‘freeze’ python code to standalone ‘.exe’ file. Specific ‘Pyinstaller’ and ‘Auto-py-to-exe’ configurations for Station and Caller can be found in GitHub along the code as well.

You can also use “Inno Setup” or “InstallShield” to compile windows installation file from this exe.

GitHub link: <https://github.com/mz-design/PyCQ>

**Testing:**

All application modules were unit-tested prior delivery. Performed system test with 5 PCs (caller + 4 stations) – personal environment limitation (I have only 5 inhouse…) 😊

Tested with Win-10 and Win-11 as well both ‘home’ and ‘Pro’ editions

**Known Issues:**

No specific issues were observed during short testing period

**Q&A:**

1. “It’s damn slow to startup…” – come on guys... it’s python, do you actually await performance of native C++ code??? And 67MB exe file includes ~800MB of compressed libraries inside 😊
2. Station tray icon not shown. – Windows 10 and 11 like “hiding” unused tray icons. You can unhide it in “windows taskbar settings”.
3. It takes time Caller “see” available stations. Basically, it can take up to ~30 sec after initial start of both applications. Caller Advertise UDP packets sent once per 5 sec, and ‘periodic registration’ and ‘periodic keep-alive’ parameters to 30 sec both by default (otherwise it will cause visible NW performance degradation in case of multiple stations and a lot of signaling)
4. Both PC are on same NW but not connected? Why? – Check if your PCs have single NW connection. When you have multiple NW adapters enabled in your PC it can cause routing problem on specific subnet. Disable second NW adapter or set ‘metric=1’ on IP connected to LAN.
5. Sometimes I see Station disconnects from Caller and does not appear in stations’ list? Specific can happen. Especially when connected with Wi-Fi router. WLAN able producing bursty traffic. This can happen basically on any Wi-Fi network of any kind, that can cause packet lost. In most situation connection will resume in single seconds, but you can try workaround it by increasing MAX\_RETRANS parameter on specific NW adapter level or decreasing Wi-Fi Access Point retransmission interval. Cable Gigabit Ethernet NW are not affected by this (at least within my tests)
6. Application crashed? – strange… for me it worked pretty stable 😊 - please open issue on GitHub. And if you will attach PyCQ.log file it will be even more than useful! Please provide as more information as you have – this will mainly help in reproducing and fixing it.
7. How do I configure pop-up transparency and sound volume on Station app? - Right click application tray icon to open its context menu.
8. Where is the message in ‘/MsgStore/’ of station after I closed the popup? - You can replay the message as much as you wish whilst the popup exists. Closing popup will inform the Caller the message was “read” and it will remain in his message history, but station will delete it – it’s “by design”.
9. When I re-start standalone ‘.exe’ file there are no messages in message history and MsgStore folder is empty? – Yes, this is the way the ‘frozen’ code works. Executable creates all data including /MsgStore/ and /resources/ folders as well as .csv files in Temporary folder when it started for this reason it can’t save the files from previous session. When you running native code in python project (i.e., PyCharm) you will continue working with same files and folder after restart.
10. **How do I install it?** - Use setup files it will install everything automatically, or just place ‘.exe’ file in folder you want to run it from. Important! – on first application run don’t forget to check both ‘v’ in Windows Defender Firewall popup that will appear. If you want starting "Station" app on system start, just copy shortcut to ‘station.exe’ to windows startup folder.
11. You need downloading and extract archives from <https://drive.google.com/drive/folders/13HQtb6TAIbM6Tfcey9NqujX9m3nG7wpP> . Archive files password is **mzdesign**.
12. Two options are available: standalone .exe files to be placed at any suitable folder on your drive or Windows installers to install specific application in "C:\Users\*user\_directory\*\AppData\Local\Programs". (The installers also provide option to install application for all Windows users).
13. “PyCQ\_Station\_Admin\_Setup.exe” will install the station application example precompiled with 2 "Administrative restrictions" in GUI: User cannot change the default volume (80%) and "Exit" button is disabled.
14. Please keep in mind station and caller applications sharing the same resource files and libraries, so you can run only one instance of application at time (i.e., specific PC may act as Caller or Station, but not both together!).
15. When running applications or installers first time there is a possibility 'Windows Defender' or other antivirus installed false alerting on specific malware: **Trojan:Win32/Wacatac.B!ml** - specific is false alert caused by improper handling of ‘Pyinstaller’ generated files by antivirus software ([pyinstaller/pyinstaller#5854](https://github.com/pyinstaller/pyinstaller/issues/5854)). You can be absolutely confident; all executables were thoroughly checked for malware of any kind as well as you welcome reviewing source code and asking ‘Pyinstaller’ staff 😊

**Some Screenshots:**











