Technical Feasibility of Audio Capture on macOS and Real-Time Transcription Options

Executive Summary

This research report evaluates the technical feasibility of building a meeting assistant application for macOS that can record system audio and microphone during meetings, transcribe in real-time, and send to OpenAI API for summarization. The application needs to work in a locked-down corporate environment with Microsoft SSO and Intune security, and should be compatible with multiple meeting platforms (Teams, Google Meet, WebEx, GoToMeeting).

Key findings: - **Audio Capture**: BlackHole is a viable open-source virtual audio device for capturing system audio, with Loopback being a premium alternative better suited for corporate environments. - **Simultaneous Capture**: Combining system audio and microphone input requires creating aggregate devices or using specialized APIs like AVAudioEngine and ScreenCaptureKit. - **Real-Time Transcription**: While Whisper AI offers high accuracy, it has limitations for real-time use. Commercial alternatives like Deepgram and AssemblyAI provide better streaming capabilities. - **Corporate Compatibility**: Deployment in Intune-managed environments requires careful consideration of system extension approvals, notarization, and privacy permissions.

1. Virtual Audio Devices for Capturing System Audio on macOS

1.1 BlackHole

BlackHole is a modern, open-source virtual audio driver designed as a replacement for Soundflower, compatible with macOS 10.10 and above, including Apple Silicon Macs.

Key Features: - Open-source (GPL-3.0 license) - Supports multiple channel configurations (2, 16, 64, 128, 256) - Zero additional latency - High sample rates up to 768kHz - Compatible with Intel and Apple Silicon Macs - Can be integrated into aggregate devices via macOS Audio MIDI Setup

Limitations: - Requires manual setup and configuration - Needs system extension approval in corporate environments - No GUI for configuration (relies on macOS Audio MIDI Setup)

1.2 Loopback

Loopback by Rogue Amoeba is a premium commercial solution with extensive features and a user-friendly interface.

Key Features: - Intuitive GUI for creating complex routing configurations - Supports up to 64 channels per virtual device - Can create multiple virtual devices with different configurations - Captures audio from multiple sources, including hidden and system sources - Commercial license with dedicated support - Regular updates and compatibility with latest macOS versions

Limitations: - Commercial product with licensing costs - Still requires system extension approval in corporate environments

1.3 Other Alternatives

Soundflower: - Classic open-source virtual audio driver - Development discontinued, compatibility issues with newer macOS versions - Not recommended for new deployments, especially on Apple Silicon Macs

VB-Cable: - Free alternative with simple setup - Limited features compared to BlackHole and Loopback - Less widely used in macOS environments

1.4 Recommendation for Corporate Environment

For a corporate environment with Intune management: - **Loopback** is recommended for its reliability, support, and user-friendly interface - **BlackHole** is a viable free alternative if cost is a concern and technical setup is acceptable

2. Methods to Simultaneously Capture System Audio and Microphone Input

2.1 Core Challenges

- macOS does not natively allow applications to record internal system audio
- Ensuring synchronization between system audio and microphone input
- Managing latency and drift between different audio sources
- Handling echo cancellation when capturing both output and input

2.2 Technical Approaches

2.2.1 Multi-Output Device Method

1. Create a multi-output device in macOS Audio MIDI Setup that combines:
   - System speakers/headphones
   - Virtual audio device (BlackHole/Loopback)
2. Set the system output to this multi-output device
3. Capture both the virtual device (for system audio) and microphone as separate inputs
4. Mix and synchronize the streams in software

Pros: - Works with any application without modification - No code required for basic setup

Cons: - Manual configuration required - Potential synchronization issues - No built-in echo cancellation

2.2.2 AVAudioEngine Approach

AVAudioEngine provides a powerful framework for audio processing and routing:

1. Connect the microphone to an `AVAudioInputNode`
2. Capture system audio via virtual audio device
3. Use `AVAudioEngine`'s `synchronizationClock` to align streams
4. Process, mix, and record both streams

Code Example (Conceptual):

```
let engine = AVAudioEngine()
let micInput = engine.inputNode
let systemAudioInput = engine.inputNode // From virtual device
let mixer = engine.mainMixerNode

engine.connect(micInput, to: mixer, format: micInput.outputFormat(forBus: 0))
engine.connect(systemAudioInput, to: mixer, format: systemAudioInput.outputFormat(forBus: 0

// Start recording
try engine.start()
```

Pros: - Precise control over audio processing - Better synchronization capabilities - Potential for echo cancellation using `kAudioUnitSubType_VoiceProcessingIO`

Cons: - Requires custom code implementation - More complex setup

2.2.3 ScreenCaptureKit Method (macOS 13+)

ScreenCaptureKit is Apple's newer framework for screen and audio capture:

1. Use `SCShareableContent` to identify capturable content
2. Create `SCContentFilter` to specify what to capture
3. Configure `SCStreamConfiguration` with audio enabled
4. Start capture session and process audio buffers

Pros: - Native API for capturing screen and audio - Provides synchronized capture - Officially supported by Apple

Cons: - Requires macOS 13 or later - Needs Screen Recording permission - More complex implementation

2.3 Synchronization Strategy

For proper synchronization of system audio and microphone:

1. Capture both streams with accurate timestamps
2. Use a common synchronization clock
3. Implement buffer alignment and drift correction
4. Process audio offline for precise synchronization if needed

3. Real-Time Transcription Options

3.1 Whisper AI

OpenAI's Whisper is an open-source speech recognition model with high accuracy and multilingual support.

Key Features: - Open-source and customizable - Supports 95+ languages - High accuracy, especially in noisy environments - Translation capabilities

Limitations for Real-Time Use: - Not designed for streaming by default - Higher latency compared to commercial alternatives - Requires significant computational resources - Custom implementation needed for real-time processing

Adaptation for Real-Time: - Segment audio into small chunks (e.g., 5-10 seconds) - Process chunks in parallel or sequentially - Implement custom streaming pipeline - Consider running on GPU for faster processing

3.2 Commercial Alternatives

3.2.1 Deepgram Deepgram offers enterprise-grade speech recognition optimized for real-time streaming.

Key Features: - Native streaming support via WebSocket APIs - Ultra-low latency (~300ms) - High accuracy with custom model training - Speaker diarization and word timestamps - On-premises deployment options

Pricing: - Pay-per-use model based on audio hours - Enterprise plans available

3.2.2 AssemblyAI AssemblyAI provides a comprehensive API with advanced features.

Key Features: - Streaming API with low latency - 99+ languages supported - Sentiment analysis and summarization - Speaker diarization and content moderation - Simple REST API integration

Pricing: - Pay-per-use model - Free tier available for development

3.2.3 Cloud Provider Solutions **Google Speech-to-Text:** - Low latency (~250ms) - 125+ languages supported - Noise robustness - Integration with Google Cloud

Microsoft Azure Speech SDK: - Low latency (<250ms) - 85+ languages supported - Custom speech models - Integration with Azure ecosystem

3.3 Latency Comparison

Solution	Native Streaming	Typical Latency	Deployment Options
Whisper AI	No (adaptable)	High (~seconds)	Local, Open-source
Deepgram	Yes	<300ms	Cloud, On-premises

Solution	Native Streaming	Typical Latency	Deployment Options
AssemblyAI	Yes	<1s	Cloud
Google Speech	Yes	~250ms	Cloud
Azure Speech	Yes	<250ms	Cloud, Hybrid

3.4 Recommendation

For a corporate meeting assistant application: - **Deepgram** is recommended for its low latency and on-premises options - **Azure Speech SDK** is a good alternative if already using Microsoft ecosystem - **Whisper AI** could be used for offline processing or as a fallback

4. Compatibility with Locked-Down Corporate Environments

4.1 Intune Management Considerations

Microsoft Intune manages macOS devices in corporate environments through: - Configuration profiles for system settings - App deployment and management - Security policies and compliance

4.2 System Extensions and Permissions

4.2.1 System Extensions Virtual audio devices like BlackHole and Loopback require system extensions, which need approval in Intune-managed environments:

- **System Extensions Policy:** Administrators can create policies in Intune to pre-approve extensions based on team identifiers
- **Transition from Kernel Extensions:** Apple is phasing out kernel extensions in favor of system extensions, which run in user space for enhanced security

Intune Configuration: - Use the "Extensions" profile type or Settings Catalog - Specify allowed team IDs and extensions - Consider blocking user overrides for consistent deployment

4.2.2 Notarization Requirements Apple requires software to be notarized for security: - All third-party extensions should be signed with a Developer ID - Extensions must be submitted to Apple's notarization service - System extensions must be properly signed and notarized before deployment

4.2.3 Privacy Permissions The application will require several privacy permissions:

- **Microphone Access:** Required for capturing user's microphone
- **Screen Recording:** Needed if using ScreenCaptureKit for system audio capture

- **Automation:** May be needed for controlling meeting applications

Managing Permissions via Intune: - Deploy privacy preferences profiles (PPPC profiles) - Pre-approve access for approved apps or extensions - Use TCC (Transparency, Consent, and Control) settings

4.3 Deployment Strategies

For deploying in a locked-down corporate environment:

1. **Package the Application:**
   - Sign with Developer ID
   - Notarize with Apple
   - Create installer package (.pkg)
2. **Configure Intune Policies:**
   - Create system extension approval policy
   - Deploy privacy preferences profile
   - Set up app configuration policy
3. **User Experience Considerations:**
   - Minimize permission prompts through pre-approval
   - Provide clear documentation for any required user actions
   - Consider SSO integration for authentication

5. Implementation Approaches

5.1 Standalone Application

Advantages: - Full control over the audio capture pipeline - Direct access to system APIs - Better performance and reliability

Technical Stack: - **Frontend:** Electron with JavaScript/TypeScript - **Backend:** Node.js with native modules or Swift/Objective-C bridges - **Audio Capture:** AVAudioEngine or BlackHole/Loopback with aggregate devices - **Transcription:** WebSocket connection to Deepgram/AssemblyAI or local Whisper implementation

Development Considerations: - Requires code signing and notarization - Needs system extension approval for virtual audio devices - Must request privacy permissions (microphone, screen recording)

5.2 Browser Extension

Advantages: - Easier deployment in corporate environments - Works across different browsers - Potentially fewer permission issues

Technical Stack: - **Frontend:** JavaScript/TypeScript with Chrome/Edge extension APIs - **Audio Capture:** `chrome.tabCapture` API or content scripts - **Transcription:** WebSocket connection to cloud transcription service - **Native Bridge:** Native messaging host for system-level functionality

Limitations: - Restricted access to system audio (browser tab audio only) - Cannot capture audio from native applications (Teams desktop app) - Limited by browser extension APIs

5.3 Hybrid Approach

A hybrid approach combines a lightweight native application with browser integration:

1. Native app handles audio routing and system integration
2. Browser extension captures in-browser meeting audio
3. Native messaging protocol connects the two components
4. Transcription processing can happen in either component

Advantages: - Leverages strengths of both approaches - More flexible deployment options - Better compatibility across meeting platforms

6. macOS-Specific Limitations and Permissions

6.1 System Audio Capture Limitations

- No direct API for system audio capture
- Requires virtual audio device or ScreenCaptureKit
- Audio routing can be disrupted by system updates or user changes

6.2 Required Permissions

Permission	Purpose	Approval Method
Microphone	Capture user's voice	Standard permission prompt or PPPC profile
Screen Recording	Capture screen and system audio	Permission prompt or PPPC profile
Accessibility	Automation of meeting apps	Permission prompt or PPPC profile
Full Disk Access	Access to protected files	Permission prompt or PPPC profile

6.3 Notarization and Gatekeeper

- All applications must be notarized by Apple
- Gatekeeper verifies notarization before allowing installation
- Corporate environments may have additional security requirements

6.4 Apple Silicon Considerations

- Kernel extensions are not supported on Apple Silicon
- System extensions must be compatible with arm64 architecture

- Performance benefits for on-device transcription with Apple Neural Engine

7. Conclusion and Recommendations

7.1 Technical Feasibility Assessment

Building a meeting assistant application for macOS that captures system audio and microphone for real-time transcription is technically feasible, but with several considerations:

- **Audio Capture:** Viable through virtual audio devices or newer APIs
- **Transcription:** Real-time capabilities available through commercial APIs
- **Corporate Deployment:** Requires careful planning for permissions and approvals

7.2 Recommended Approach

1. **Audio Capture:**
    - Use Loopback for corporate environments (premium, reliable)
    - BlackHole as an alternative for cost-sensitive deployments
    - Implement AVAudioEngine for processing and synchronization
2. **Transcription:**
    - Deepgram for real-time, low-latency transcription
    - Consider Azure Speech if already in Microsoft ecosystem
    - Whisper AI for offline processing or as a fallback
3. **Implementation:**
    - Standalone Electron application with native modules
    - Proper signing, notarization, and packaging
    - Intune deployment with pre-approved permissions
4. **Corporate Compatibility:**
    - Work with IT administrators to approve system extensions
    - Deploy privacy preferences profiles via Intune
    - Document installation and permission requirements

7.3 Next Steps

1. Develop a proof-of-concept to validate audio capture approach
2. Test transcription services for accuracy and latency
3. Create deployment packages and Intune policies
4. Conduct testing in a simulated corporate environment
5. Refine the solution based on feedback and performance

By addressing these technical considerations and following the recommended approach, it is possible to build a robust meeting assistant application that works effectively in a locked-down corporate environment while providing valuable real-time transcription and summarization capabilities.