Network Working Group
Internet-Draft
Intended status: Informational
Expires: 30 June 2026

T. Rocha
Independent Researcher

December 2025

When Independent Constraints Collapse: Why Real-Time Systems Need a Missing Control Layer

draft-rocha-independent-constraints-collapse-00

Abstract

Real-time communication systems evolved by solving discrete problems independently: session establishment, transport, media handling, security, analytics, and compliance were addressed as separate concerns. For two decades, the resulting architectural fragmentation was tolerable. That tolerance has ended. This document argues that the convergence of Zero Trust security requirements, AI-driven computational demands, accessibility mandates, data residency enforcement, and next-generation transport capabilities now exceeds the coordination capacity of fragmented architectures. The failure is not incremental but fundamental: real-time systems lack a control layer capable of maintaining continuous session identity across orchestration, policy enforcement, and computation. This architectural gap was not anticipated by existing standards or platform designs. Its recognition requires reduction rather than expansion, collapsing fragmented contexts into unified session governance. This document examines the historical evolution that produced this gap, analyzes why current approaches fail under convergent pressures, and articulates the architectural constraint necessary for future systems.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 2026.

Table of Contents

1. Introduction

Service disruptions occur regularly across major collaboration platforms, affecting millions of users [TEAMS-OUTAGE]. As real-time interactions accumulate auxiliary services such as transcription, recording, and AI processing, systems rely on cross-service coordination to preserve coherent behavior. When policy or configuration changes propagate unevenly across service boundaries, recovery can be difficult even when core transport remains healthy. These incidents demonstrate that large-scale real-time communication platforms fail for reasons beyond transport, including configuration errors and control-plane complexity.

Real-time communication systems have reached an inflection point. Pressures that evolved independently (Zero Trust security models, artificial intelligence integration, accessibility requirements, data residency enforcement, and transport advances) now collide in ways that exceed the coordination capacity of existing designs. The prevailing response has been additive: more orchestration layers, additional policy engines, auxiliary processing pipelines. This once worked. It no longer does.

This document argues that the failure is architectural rather than

operational. Real-time systems lack a control layer capable of maintaining a single, continuous session identity across orchestration, policy enforcement, and computation. This absence was not anticipated by prior designs, not because engineers failed to recognize the problem, but because the problem did not exist until independent constraints converged.

What follows is neither a protocol proposal nor a platform specification. It is an architectural analysis: an examination of how real-time systems evolved, why that evolution now fails, and what fundamental constraint must be satisfied for future systems to function under converged pressures.

2. Historical Evolution and Architectural Foundations

2.1. Circuit-Switched Origins and Inherited Assumptions

Real-time communication inherited its conceptual model from circuit-switched telephony [TELCO-SIG]. A telephone call was singular: two endpoints, one connection, fixed identity. Policy, to the extent it existed, was established at call setup and remained static until teardown. The session simply existed.

When packet networks replaced physical circuits, this abstraction persisted. Sessions became signaling dialogs rather than dedicated circuits, but the mental model remained narrow [INET-TELE]. The Internet Engineering Task Force's Session Initiation Protocol (SIP) preserved the call-centric paradigm: INVITE-ACK-BYE captured session lifecycle, but governance during the session itself remained minimal [RFC3261].

This worked because nothing demanded more. Transport was constrained, auxiliary services were limited, and policy was largely static. Session establishment was the hard problem. Session governance was not.

2.2. The SIP Era: Stateless Intermediaries and the First Missed Juncture

SIP's introduction of stateless intermediaries and horizontal scalability represented significant architectural advances [RFC3261]. Proxy servers could route signaling without maintaining session state, enabling failure tolerance and elastic scale. These were necessary innovations for Internet-scale deployment.

They also shifted responsibility away from the session itself. Control moved to the network edge. Session identity became contextual rather than continuous, reconstructed from signaling exchanges rather than preserved as a first-class entity.

In retrospect, this was the first point where session continuity could have been elevated from signaling detail to architectural control surface. It was not. Author review of IETF standards development did not identify debate in working group discussions, alternative proposals in Internet-Drafts, or white papers exploring continuous session identity as an architectural requirement. The omission is notable not as oversight but as evidence: there was no forcing function to question the assumption that sessions were artifacts to establish rather than entities to govern.

2.3. WebRTC and the Second Missed Juncture

Web Real-Time Communication (WebRTC) introduced peer-to-peer connections, dynamic renegotiation, and elastic media handling at global scale [WEBRTC]. Unlike SIP's server-mediated model, WebRTC

enabled direct browser-to-browser communication with sophisticated media control. Sessions became elastic: tracks could be added or removed, transceivers reconfigured, connections renegotiated, all without ending the session.

Elasticity exposed a choice: preserve a single session identity through change, or create new sessions when complexity appeared. The industry chose the latter.

Parallel PeerConnections became standard practice for handling multiple media types, separate data channels, or auxiliary processing [WEBRTC-IMPL]. A single video call might involve three or more PeerConnection objects: one for primary audio/video, another for screen sharing, a third for data exchange. From the user's perspective, these constituted one interaction. From the architecture's perspective, they were independent sessions requiring external coordination.

Again, there was no articulation of session singularity as a control requirement. The WebRTC specifications document renegotiation mechanisms and connection management but do not define requirements for maintaining unified session identity across orchestration boundaries [WEBRTC]. Performance was cheap enough, complexity manageable enough, that fragmentation remained tolerable.

2.4. Platform Unification and the Fragmentation Strategy

Modern platforms unified user experience by hiding architectural fragmentation behind product layers. Internally, sessions multiplied. Sidecar services emerged: recording operated out-of-band, transcription ran in parallel pipelines, analytics consumed mirrored streams, compliance systems captured independent copies. Each auxiliary function maintained its own session context, coordinated through external mechanisms.

Functionality improved dramatically. Architectural coherence did not. Fragmentation became not just acceptable but standard practice, the solution rather than a compromise.

This strategy succeeded because operational benefits outweighed coordination costs. Modularity simplified debugging: failures isolated to specific services. Independent scaling allowed targeted resource allocation: AI pipelines could scale differently than media transport. Risk management improved: experimental features could deploy without endangering core functionality [RELEASE-IT]. For fifteen years, these advantages justified the architecture. They no longer do.

2.5. Prior Approaches and Their Limits

Existing work on session management falls into distinct categories, none of which establish single-session orchestration as an architectural requirement:

Standards-based session control includes SIP [RFC3261], WebRTC specifications [WEBRTC], Interactive Connectivity Establishment (ICE) [RFC8445], and Secure Real-time Transport Protocol (SRTP) [RFC3711]. These optimize component-level concerns (signaling, peer negotiation, network traversal, media security) without defining whole-session governance. Sessions remain negotiated artifacts rather than governed entities.

Session mobility frameworks address continuity across network handoffs or device migration. 3GPP's 5G specifications [TS-23.501] [TS-23.502] define session continuity for cellular networks; SIP session timers maintain signaling liveness [RFC4028]. These treat

continuity as a transport-layer concern, ensuring connections survive network changes. They do not govern orchestration, policy enforcement, or computational routing within the session itself.

Platform-specific implementations may use unified session management internally, but this remains proprietary implementation detail rather than disclosed architectural requirement. Platforms like Zoom, Microsoft Teams, and Google Meet have evolved sophisticated orchestration systems, but these are product implementations, not constraint specifications that other systems must satisfy.

The gap is not capability but requirement. Existing systems can implement continuous session identity. Standards and platforms do not mandate it as the architectural foundation for orchestration across policy, computation, and auxiliary services. This distinction matters: a constraint-based architecture differs fundamentally from an implementation choice.

3. The Convergence: Why Current Architectures Fail

3.1. Independent Pressures That No Longer Remain Independent

Several forces evolved on separate trajectories. None were designed to interact. They now collide in ways that fragmented architectures cannot accommodate.

3.1.1. Zero Trust Security

Traditional security models assumed stable posture during interaction: authenticate at session establishment, maintain trust until teardown. Zero Trust architectures require continuous verification and dynamic policy adjustment during the session [ZERO-TRUST] [NIST-ZT].

In fragmented architectures, this creates multiplicative enforcement complexity. Each auxiliary service (transcription, recording, AI processing) operates in its own session context with separate policy state. A mid-session policy change (user joins from restricted jurisdiction, classification level changes, consent is revoked) must propagate across all fragments. Policy consistency becomes an inference problem rather than an enforcement guarantee.

External coordination cannot solve this. The fundamental issue is that policy enforcement requires knowing what constitutes "the session": which participants, which data flows, which computational processes are in-scope for a given policy decision. Fragmentation makes this boundary indefinite.

3.1.2. The AI Efficiency Paradox

Artificial intelligence adds extraordinary capability to real-time systems: live transcription, translation, content moderation, summarization, meeting assistance. It also multiplies computational, power, and coordination costs in ways that parallel architectures amplify [AI-ENERGY] [AI-CARBON].

Consider a 50-participant meeting with AI features:

Fragmented approach:

- 50 peer connections (base topology)
- 50 parallel transcription pipelines (one per audio stream)
- 50 translation engines (assuming multi-language support)
- Independent recording streams
- Coordination overhead: O(n*m) where n = participants, m = auxiliary services

- Result: 2,500+ discrete coordination points

Unified approach:
- 1 session identity
- Dynamic routing to shared processing pools
- Unified policy enforcement across all computational resources
- Coordination overhead: O(n + m)
- Result: approximately 100 coordination points

As participant count scales or auxiliary services multiply (add accessibility features, compliance monitoring, quality analysis, content moderation), fragmented coordination grows geometrically while unified coordination grows linearly [DIST-ALGO] [DIST-COMP].

In practice, AI features commonly run as service-side pipelines that process audio separately from the primary media path and introduce their own scaling limits and regional constraints, increasing operational cost and coordination burden [CHIME-TRANSCRIBE]. The efficiency paradox manifests when each AI feature requires duplicate infrastructure across fragmented sessions.

3.1.3. Accessibility as Native Requirement

Accessibility has transitioned from post-hoc feature to regulatory mandate. The European Accessibility Act comes into effect on 28 June 2025; accessibility obligations for information and communication technology are enforced through procurement and regulatory regimes [EAA] [SECTION-508]. Post-processing approaches are often insufficient for real-time interactions in practice.

Live captions, sign language interpretation, audio description, and assistive modalities require real-time negotiation within the interaction itself. In fragmented architectures, accessibility becomes another parallel pipeline, one more sidecar service with its own session context, policy enforcement, and failure modes.

The regulatory requirement is not that accessibility features exist, but that they function as part of the interaction with the same reliability, security, and privacy guarantees. Fragmentation undermines this by making accessibility an auxiliary concern rather than a native capability.

3.1.4. Data Residency as Enforceable Constraint

Data residency has evolved from declarative policy to enforceable mandate. The EU’s GDPR, China’s Data Security Law, and various sector-specific regulations now require not inference but proof of data locality [GDPR] [CHINA-DSL].

In fragmented architectures where auxiliary services operate independently, data residency becomes a coordination problem: ensuring that media, transcripts, recordings, AI processing, and analytics all respect the same geographical constraints requires external verification across session fragments. Detached pipelines turn compliance from deterministic enforcement into probabilistic inference.

When a participant from a restricted jurisdiction joins mid-session, policy must propagate immediately across all processing: media routing changes, AI features may need to disable, recording may need to segregate streams. In a fragmented architecture, this requires synchronized updates across independent contexts. Eventual consistency is not sufficient.

Regulation requires deterministic enforcement.

3.1.5. Transport Saturation Without Control Abstraction

Network performance continues to improve across cellular, Wi-Fi, and satellite systems. As latency falls and capacity rises, transport ceases to be the limiting factor, and coordination becomes the bottleneck.

Faster transport does not resolve coordination failures. It accelerates them. Lower latency makes coordination overhead more visible. Higher bandwidth enables more parallel streams, multiplying the coordination burden. Next-generation transport amplifies architectural weakness rather than masking it.

The gap is not transport capability but control abstraction. We have the pipes but lack the valve system.

3.2. Why Incremental Fixes Fail

The prevailing response to these pressures has been additive: deploy new orchestration layers, stack additional policy engines, attach auxiliary processing services. This approach succeeded when pressures evolved independently. It fails when they converge.

The mathematical reality is unfavorable. Existing orchestration strategies apply linear control logic to systems whose complexity now scales exponentially with participants, policies, modalities, and auxiliary computation [LAMPORT] [HERLIHY]. Each addition multiplies coordination across identities, enforcement jurisdictions, and processing boundaries.

More bandwidth does not resolve this mismatch. It amplifies failures. More compute worsens the efficiency paradox by enabling even more parallel pipelines. Post-hoc governance satisfies neither regulators (who require real-time enforcement) nor operators (who cannot verify policy consistency across fragments).

Resource masking (deploying more power, more compute, more infrastructure) delays failure but does not prevent it. As orchestration, policy, and computation scale together, coordination cost grows faster than resource capacity. Fragmentation reasserts itself once linear resource growth encounters exponential coordination complexity.

The result is not gradual degradation but architectural exhaustion. Systems begin failing not because any single component breaks, but because coordination across fragments becomes intractable.

3.3. The Fragmentation Trade-off Reversal

Fragmentation persisted for two decades because it offered genuine operational advantages. Modularity simplified debugging by isolating failures to specific services. Independent scaling allowed targeted resource allocation. AI pipelines could scale differently than media transport. Risk management improved: experimental features could deploy without endangering core functionality. These benefits were real, and they justified the architecture [RELEASE-IT] [PARNAS] [MICROSERVICES].

That calculus has reversed. The operational simplicity that made fragmentation attractive now costs more than it saves. When every new capability requires another parallel pipeline, when policy changes must synchronize across dozens of session contexts, when regulatory enforcement demands deterministic control rather than eventual consistency, the coordination burden exceeds the modularity benefit.

This reversal is evident in operational patterns. Operational experience and incident reporting from platform operators suggest that coordination complexity across auxiliary services increasingly contributes to production incidents. The architecture optimized for operational simplicity has become operationally complex.

4. The Missing Constraint

4.1. Why the Path Was Always Open

Existing systems never forbade unified session identity. Standards did not prohibit it. Platforms could have implemented it. The absence was not technical impossibility but lack of requirement.

Fragmentation was an optimization for a different set of pressures. When auxiliary services were limited, when policy was largely static, when computational demands were modest, fragmentation simplified operations without imposing unsustainable coordination costs.

Reduction was never considered necessary.

Leaving a path open is not the same as recognizing its importance. Author review of the IETF RFC repository and Internet-Draft archive did not identify a document that defines continuous session identity as an architectural requirement for orchestration across policy, computation, and auxiliary services. This was not oversight. It was absence of necessity. The pressures that would make this requirement apparent had not yet converged.

4.2. Reduction as Architectural Necessity

The path forward is subtractive, not additive. It requires collapsing fragmented contexts into unified session governance, treating the session as a control surface rather than a negotiated artifact.

This is not nostalgia for simpler systems. It is architectural realism about complexity limits. The current approach (adding coordination layers atop fragmented sessions) scales cost faster than capability. Reduction inverts this: establish the session as the coordination primitive, then build auxiliary services within that unified context rather than around it.

The distinction is fundamental. In fragmented architectures, orchestration coordinates between sessions. In unified architectures, orchestration operates within a session. The former requires external synchronization across independent contexts. The latter requires internal routing within a governed entity.

4.3. The Control Layer Requirement

What is absent is not a protocol, platform, or feature. It is a control layer responsible for preserving continuous session identity across orchestration, policy enforcement, and computational routing.

This layer must satisfy three constraints:

Session continuity: Maintain a single, persistent session identity through all orchestration changes (participant additions/removals, policy updates, media renegotiations, auxiliary service routing) without fragmenting into multiple contexts requiring external coordination.

Transport agnosticism: Operate independently of underlying

transport mechanisms (WebRTC, SIP, proprietary protocols) such that the constraint applies regardless of how media and data are actually moved.

Computational transparency: Route computational resources (AI processing, recording, transcription, analytics) within the session's policy boundary without creating separate session contexts for each auxiliary service.

Any approach that violates these constraints reintroduces fragmentation under a different name. The layer must govern what occurs within an interaction rather than around it. Its purpose is coherence, not optimization.

In this document, this architectural requirement is referred to as Single-Session Orchestration and Adaptive Routing (SSOAR). The terminology is secondary to the constraint it reflects: that real-time systems must maintain unified session governance as an architectural foundation rather than an implementation option.

No specific mechanism is prescribed. No implementation is implied. Only the boundary condition is defined: preserve session continuity across all orchestration, policy, and computational requirements.

5. Implications and Conclusion

5.1. Why This Emerges Now

This architectural requirement could not have been articulated earlier. The pressures were not aligned. Fragmentation's coordination costs remained below its operational benefits. Zero Trust operated at network perimeters, AI was peripheral to core functionality, accessibility was post-hoc, data residency was declarative, and transport limitations masked coordination inefficiency.

The requirement emerges now because independent constraints collapsed simultaneously. Each pressure evolved to the point where it could no longer be satisfied through external coordination.

Their convergence exceeded the architectural tolerance of fragmented systems.

This is how foundational shifts occur, not through deliberate design but through pressure convergence that makes previously optional constraints necessary.

5.2. What Recognition Enables

Once this control layer requirement is acknowledged, previously intractable problems become architecturally solvable:

Zero Trust becomes enforceable during interaction: Policy operates on a single session boundary rather than being inferred across fragments. Mid-session policy changes propagate through internal routing rather than external synchronization.

AI becomes governable and efficient: Computational resources route within unified session context. Coordination overhead scales linearly with services rather than geometrically with service-participant products. The efficiency paradox resolves.

Accessibility becomes native: Assistive modalities participate as first-class capabilities within the session rather than as parallel pipelines. Regulatory requirements for real-time

accommodation become architecturally natural.

Data residency becomes deterministic: Geographical policy enforcement operates on session-level routing rather than being verified across independent contexts. Compliance transitions from inference to control.

Transport advances regain meaning: Network performance improvements translate to application capability rather than being consumed by coordination overhead. Control abstraction catches up to transport capability.

None of this requires technological invention. It requires architectural recognition: that the missing layer is not optional anymore.

5.3. The Path From Recognition to Realization

Recognition is necessary but not sufficient. Realizing this architectural requirement faces substantial challenges:

Migration economics: Existing systems represent billions in deployed infrastructure. Transition paths must preserve operational continuity while enabling architectural evolution.

Backward compatibility: New systems must interoperate with fragmented architectures during transition. The constraint layer cannot demand immediate, universal adoption.

Standards development: Establishing single-session orchestration as an architectural requirement likely requires standards coordination, either through new specifications or extensions to existing frameworks (WebRTC, SIP, 3GPP).

Platform adoption incentives: First-mover costs must align with competitive advantages. Platforms adopting unified session governance must see operational or market benefits that offset migration complexity.

These challenges are substantial but not insurmountable. The forcing function is not theoretical elegance but operational necessity. As pressures continue converging, systems that cannot satisfy the unified session requirement will face increasing incidents, regulatory violations, and competitive disadvantage.

5.4. Conclusion

Real-time communication systems evolved by solving fragmentation through composition. That strategy succeeded for two decades. Its success has ended.

The failure is not incremental but architectural. Independently evolved pressures (Zero Trust, AI integration, accessibility mandates, data residency enforcement, transport advances) now exceed the coordination capacity of fragmented designs. The missing element is not capability but constraint: a control layer that maintains continuous session identity across orchestration, policy, and computation.

This requirement was not anticipated because it was not necessary until independent constraints converged. Recognition of this gap is the first step. Realization requires departure from the additive logic that has governed real-time systems since the SIP era.

The limits of current architecture are now clear. The path forward, once understood, becomes unavoidable. What remains is the

transition from recognition to implementation, from acknowledging the missing constraint to establishing it as architectural foundation.

6. Security Considerations

This document presents an architectural analysis rather than a protocol specification. It does not define new protocols, message formats, or mechanisms.

The analysis has security implications. Fragmented session identity across auxiliary services can complicate policy enforcement and auditability, particularly under Zero Trust models that require continuous verification and mid-session policy change.

If a future architecture realizes unified session governance, the control-plane would represent a high-value target. Considerations may include authentication, authorization, integrity protection for policy updates, protections against session identity misuse, and isolation of auxiliary services within the session boundary. The shift from fragmented to unified session governance can change threat models, and future work may address specific security mechanisms appropriate to particular realizations.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[WEBRTC] Bergkvist, A., Burnett, D., Jennings, C., Aboba, B., Brandstetter, T., and J. Bruaroey, "WebRTC 1.0: Real-Time Communication Between Browsers", W3C Recommendation, January 2021, <https://www.w3.org/TR/webrtc/>.

8.2. Informative References

[ZENODO] Rocha, T., "When Independent Constraints Collapse: Why Real-Time Systems Need a Missing Control Layer", Zenodo, 2025, https://doi.org/10.5281/zenodo.18001608.

[AI-CARBON] Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L., Rothchild, D., So, D., Texier, M., and J. Dean, "The

Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink", IEEE Computer, July 2022.

[AI-ENERGY] Strubell, E., Ganesh, A., and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP", Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 3645-3650, 2019.

[CHIME-TRANSCRIBE] Amazon Web Services, "Using Amazon Chime SDK live transcription", AWS Documentation, <https://docs.aws.amazon.com/chime-sdk/latest/dg/meeting-transcription.html>.

[CHINA-DSL] "Data Security Law of the People’s Republic of China", effective September 2021.

[DIST-ALGO] Lynch, N., "Distributed Algorithms", Morgan Kaufmann, 1996.

[DIST-COMP] Attiya, H. and J. Welch, "Distributed Computing: Fundamentals, Simulations, and Advanced Topics", 2nd Edition, Wiley, 2004.

[EAA] "Directive (EU) 2019/882 on the accessibility requirements for products and services", European Accessibility Act, effective 28 June 2025, <https://www.accessible.eu/european-accessibility-act/>.

[GDPR] "Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation)", Articles 44-49.

[HERLIHY] Herlihy, M. and N. Shavit, "The Art of Multiprocessor Programming", Morgan Kaufmann, 2008.

[INET-TELE] Schulzrinne, H. and J. Rosenberg, "Internet Telephony: Architecture and Protocols", IEEE Network, Vol. 17, pp. 18-23, 1999.

[LAMPORT] Lamport, L., "The Part-Time Parliament", ACM Transactions on Computer Systems, May 1998.

[MICROSERVICES] Fowler, M., "Microservices", 2014, <https://martinfowler.com/articles/microservices.html>.

[NIST-ZT] Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture", NIST Special Publication 800-207, August 2020.

[PARNAS] Parnas, D., "On the Criteria To Be Used in Decomposing Systems into Modules", Communications of the ACM, December 1972.

[RELEASE-IT] Nygard, M., "Release It!: Design and Deploy Production-Ready Software", 2nd Edition, Pragmatic Bookshelf, 2018.

[RFC4028] Donovan, S., "Session Timers in the Session Initiation

Protocol (SIP)", RFC 4028, DOI 10.17487/RFC4028, April 2005, <https://www.rfc-editor.org/info/rfc4028>.

[SECTION-508]
"Section 508 Standards for Information and Communication Technology", 36 CFR Part 1194.

[TEAMS-OUTAGE]
Gatlan, S., "Microsoft Teams outage caused by broken Azure load balancing rule", BleepingComputer, September 2023, <https://www.bleepingcomputer.com/news/microsoft/microsoft-teams-outage-caused-by-broken-azure-load-balancing-rule/>.

[TELCO-SIG]
van Bosse, J. and F. Devetak, "Signaling in Telecommunication Networks", 2nd Edition, Wiley, 2006.

[TS-23.501]
3GPP, "System architecture for the 5G System (5GS)", 3GPP TS 23.501, Section 5.6 (Session Management).

[TS-23.502]
3GPP, "Procedures for the 5G System (5GS)", 3GPP TS 23.502, Section 4.3 (Session Continuity).

[WEBRTC-IMPL]
Dutton, S., "Getting Started with WebRTC", Google Developers, 2014; Lachapelle, S. and B. McDonnell, "WebRTC Insights", Dialogic, 2020.

[ZERO-TRUST]
Kindervag, J., "Build Security Into Your Network’s DNA: The Zero Trust Network Architecture", Forrester Research, 2010.

Author’s Address

Thomas Rocha III
Independent Researcher

Email: tom.rocha.iii@gmail.com