TECHNICAL WHITE PAPER

Migrating Legacy IVRs to Conversational AI Architectures

A step-by-step technical guide to integrating modern NLP engines with existing SIP infrastructure using the Teligent P90/E platform

An engineering perspective from TELIGENT TELECOM

Rev A · 2026

Contents

1. Executive summary 3
2. Anatomy of a legacy IVR 4
3. The target conversational AI architecture 5
4. The P90/E as integration anchor 7
5. Step-by-step integration 8
6. End-to-end call flow 11
7. Latency and performance engineering 12
8. Migration strategy 13
9. Security, data, and compliance 14
10. About TELIGENT TELECOM 15
11. Next steps 16

1. Executive summary

Legacy interactive voice response (IVR) systems — DTMF menu trees driven by VoiceXML and static grammars — still front-end a large share of operator and enterprise call traffic. They are rigid, frustrating to callers, and expensive to change: every new option is a grammar edit and a redeploy. Conversational AI — streaming automatic speech recognition (ASR), a natural-language-understanding (NLU) or large-language-model (LLM) engine, a dialog manager, and streaming text-to-speech (TTS) — replaces the menu with natural dialogue, lifting containment and satisfaction.

The obstacle is rarely the AI; it is the integration. Operators cannot rip out the SIP core, the session border controllers (SBCs), the charging system, or the business back ends to adopt it. This paper is a technical guide to inserting a conversational AI layer into existing SIP infrastructure without disruption, using the Teligent P90/E platform as the integration anchor: the element that terminates the SIP leg, anchors the RTP media, streams audio to the AI services, orchestrates the dialogue, and integrates to charging and OSS/BSS — behind a stable interface that shields the network from the AI components running behind it.

It covers the reference architecture, the exact points of integration with SIP and RTP, how streaming ASR, NLU and TTS engines are connected, dialog orchestration, the latency budget that decides whether the experience feels natural, DTMF coexistence and agent handoff, and a phased migration that runs the new and old paths in parallel until the new one is proven.

Who should read this

Solutions architects, VAS and core-network engineers, and integration teams responsible for IVR modernization. The paper assumes working familiarity with SIP, RTP, and IP telephony.

2. Anatomy of a legacy IVR

Understanding what must be preserved — and what must be replaced — begins with the legacy design.

How a legacy IVR is built

A traditional IVR is a prompt-and-collect state machine. Call flows are authored in VoiceXML (VXML), sometimes orchestrated by CCXML, and execute against static speech grammars (SRGS) or, more commonly in telephony, DTMF grammars. The caller navigates nested menus — "press 1 for balance, press 2 for support" — and the system branches on the collected digit. Signalling reaches the IVR or media server over SIP (or legacy TDM), media flows as RTP, and DTMF is carried as RFC 4733 telephone-event packets within the RTP stream, or less ideally as SIP INFO.

Where it constrains the operator

The limitations are structural, not cosmetic:

- **Rigid interaction.** Callers must map their intent onto the designer's menu tree; anything not anticipated cannot be expressed.
- **High abandonment.** Deep menus and long prompts drive callers to zero-out to an agent, defeating the purpose of automation.
- **Slow to change.** Adding or reordering an option means editing VXML and grammars and redeploying — governed by a release cycle, not a business need.
- **No context or memory.** Each call starts from zero; the system cannot understand free speech, disambiguate, or carry context across turns.

What must be preserved

The parts worth keeping are the integration points, not the interaction model: the SIP entry point and routing, the media handling, the DTMF path (as a fallback), the charging integration, and the connectors to business systems. A well-designed migration retains these and replaces only the layer that decides what to say next.

> *The goal is not to replace the IVR's plumbing. It is to replace its brain — while leaving the plumbing, and the caller's number, untouched.*

3. The target conversational AI architecture

A conversational voice system is a real-time, full-duplex pipeline. Audio flows continuously in both directions; the system listens while it speaks (barge-in) and responds within a latency budget tight enough to feel like conversation. The reference architecture separates three planes — telephony signalling and media, dialog orchestration, and the AI and business services behind them.

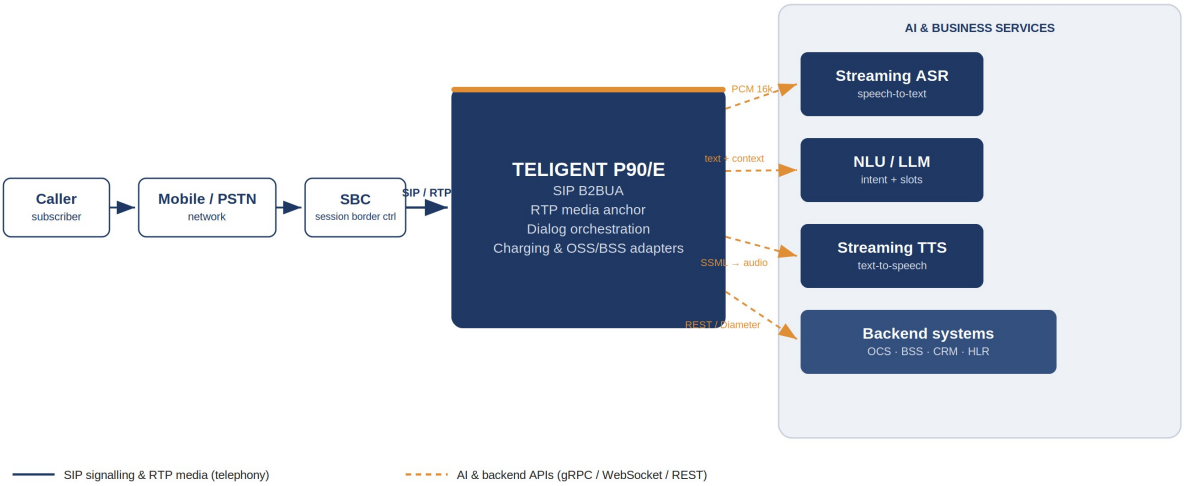


Figure 1. Conversational AI reference architecture. The P90/E anchors SIP and RTP toward the network and orchestrates the AI and business services behind a single, stable interface.

The runtime pipeline, per turn

Caller audio (RTP) is de-jittered and decoded to linear PCM and streamed to ASR, which returns partial and final transcripts with endpointing. The final (or stable interim) transcript, plus dialog context, goes to the NLU/LLM, which returns an intent and slots — or, in an LLM-driven design, the next utterance directly. The dialog manager applies policy: fill slots, call a business API, decide the next action. The response text is streamed to TTS, and the returned audio is played to the caller over RTP, interruptible by barge-in.

Integration interfaces

Each plane meets the platform at a defined interface. The table below summarizes them.

Interface	Protocol	Carries
Caller ↔ SBC ↔ P90/E	SIP; RTP / SRTP	call control; two-way voice media
DTMF input	RFC 4733 telephone-event	digit entry and fallback
P90/E ↔ ASR	gRPC / WebSocket, or MRCPv2	PCM audio out → transcripts in
P90/E ↔ NLU / LLM	REST / gRPC	text + context → intent, slots
P90/E ↔ TTS	gRPC / WebSocket, or MRCPv2	text (SSML) → synthesized audio
P90/E ↔ back end	REST / gRPC; Diameter, CAMEL	business data; real-time charging

Two ways to connect speech resources

Two integration patterns are established, and modern deployments often combine them. MRCPv2 (RFC 6787) is the standards-based protocol by which a media server controls ASR and TTS resources on a speech server, with SIP establishing the control channel and RTP carrying media — mature, interoperable, and well suited to a classic media-resource and speech-server split. Native streaming APIs use bidirectional gRPC or WebSocket streams to carry PCM to ASR and receive incremental transcripts, and to stream text to TTS and receive audio; this is how most modern AI engines expose real-time speech, and it minimizes first-response latency. A platform that speaks both — MRCP toward traditional speech servers and gRPC or WebSocket toward modern AI engines — maximizes choice and avoids lock-in.

4. The P90/E as integration anchor

The P90/E sits exactly where the integration needs an anchor: on the VAS media path, already SIP-integrated with the operator network and already connected to charging and provisioning. In this architecture it performs four roles at once.

- **SIP B2BUA.** It terminates the SIP dialog routed from the SBC, presenting a clean, controlled signalling boundary. The network sees a stable SIP endpoint regardless of which AI engines run behind it.
- **RTP media anchor.** It anchors and processes the media — jitter buffering, decoding and transcoding, RFC 4733 DTMF detection, and energy/VAD for barge-in — and taps the audio to ASR as linear PCM.
- **Dialog orchestrator.** It manages turn-taking, barge-in, timers, and the sequence of ASR → NLU → business logic → TTS calls, applying dialog policy and confidence thresholds.
- **Integration hub.** It invokes back-end APIs and applies charging — real-time via CAMEL and Diameter to the OCS, plus CDRs — reusing integrations the platform already holds.

Why anchor here rather than at the SBC or in the cloud

Anchoring at the P90/E decouples the volatile part of the system — fast-moving AI engines — from the stable part — the SIP core and charging. AI engines can be swapped, upgraded, or A/B-tested behind the platform without touching SBC routing or the charging integration. The platform's carrier-grade high availability and active-active geographic redundancy extend to the conversational service, and its existing provisioning and reporting apply to it. The result is a single, certifiable integration surface into the network instead of a new one for every AI vendor.

5. Step-by-step integration

The following ten steps take a live, DTMF-based IVR to a conversational service without an outage.

Step 1 — Discover and map

Inventory the SIP topology: SBCs, trunks, the dial-plan rules that route the service number, negotiated codecs, and the DTMF method in use. Catalogue the existing call flows and VXML, and extract the menu tree — each menu option is a candidate intent. Identify every back end the IVR calls. Fix target KPIs before touching anything: containment rate, end-to-end turn latency, ASR word-error rate (WER) per language, and intent-classification accuracy.

Step 2 — Position the P90/E in the SIP path

Route the service number's INVITE from the SBC to the P90/E over a SIP trunk; the P90/E acts as a B2BUA terminating the leg. Configure header handling and codec policy in the SDP offer/answer. Prefer a wideband codec where available — AMR-WB or Opus at 16 kHz — because ASR accuracy improves markedly over narrowband; fall back to G.711 with transcoding where the leg is narrowband.

Illustrative SDP answer — wideband preferred, narrowband and DTMF retained

```
m=audio 40000 RTP/AVP 96 0 101
a=rtpmap:96 AMR-WB/16000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
```

Step 3 — Anchor media and build the audio pipeline

With media anchored at the P90/E, de-jitter and decode the caller's RTP to 16 kHz linear PCM. Run voice-activity detection (VAD) for endpointing and barge-in. Preserve RFC 4733 telephone-event handling so DTMF remains available as a parallel input and fallback. Frame the audio — typically 20 ms — for streaming to ASR.

Step 4 — Connect streaming ASR

Open a bidirectional stream to the ASR engine (gRPC or WebSocket, or an MRCPv2 RECOGNIZE toward a speech server). Push PCM frames as they arrive and consume partial and final transcripts. Configure endpointing so a natural pause finalizes an utterance, and wire speech-onset events to barge-in so the platform stops TTS playback the instant the caller speaks. For African-language deployments, use models fine-tuned on the operator's own voice data for the target dialects; generic narrowband models are usually inadequate.

Illustrative streaming-ASR request configuration

```
StreamingRecognize.config {
  encoding:            LINEAR16
  sample_rate_hz:     16000
  language:           "ar-MA"      # Darija-tuned model
  interim_results:    true
  endpointing { silence_ms: 700 }
}
# then stream 20 ms PCM frames until end-of-speech
```

Step 5 — Integrate the NLU / LLM engine

Send the transcript and dialog context to the NLU or LLM over REST or gRPC. In a classic design the engine returns an intent and extracted slots with confidence scores; in an LLM-driven design it returns the next system utterance or a tool/function call. Define confidence thresholds — above threshold, proceed; below, reprompt or disambiguate — and keep prompts and few-shot examples versioned.

Illustrative NLU request and response

```
→ request
{ "session_id": "c1a2", "text": "check my data balance",
  "context": { "last_intent": null, "auth": "verified" } }

← response
{ "intent": "check_balance", "slots": { "product": "data" },
  "confidence": 0.94, "next_action": "call_ocs_balance" }
```

Step 6 — Orchestrate the dialogue

The dialog manager holds state: which slots are filled, what the caller is trying to do, and what to say next. A deterministic state machine is predictable and auditable; an LLM-driven policy is flexible and natural. In practice a hybrid works best — an LLM for understanding and generation, bounded by explicit state and guardrails for transactional steps such as authentication and payments, where determinism and compliance matter. The orchestrator calls business APIs at the right moments and assembles the response.

Step 7 — Connect streaming TTS

Stream the response text — with SSML for pronunciation, pauses, and prosody — to the TTS engine, and begin playing returned audio as soon as the first chunk arrives rather than waiting for full synthesis; this is the single biggest lever on perceived latency. Cache audio for static prompts such as greetings and disclaimers, and enable barge-in on every playback.

Step 8 — Wire back-end and charging integration

Connect the orchestrator to the systems the service needs — CRM, OCS, BSS, HLR — over REST or gRPC with proper authentication and timeouts. Apply charging through the platform's existing real-time (CAMEL/Diameter) and offline (CDR) integrations, so the conversational service is billed and reported like any other.

Step 9 — Provide fallback, DTMF coexistence, and handoff

Run DTMF as a parallel input path: a caller can always press a key, and the service can offer a DTMF menu when speech repeatedly fails. Define fallback policy — on low ASR or NLU confidence, or after N failed turns, degrade gracefully to directed dialogue, then DTMF, then a human. Escalate to an agent via SIP REFER (or a transfer through the SBC), passing collected context — identity, intent, transcript summary — through SIP headers or a companion API so the caller does not repeat themselves.

Step 10 — Test, tune, and cut over

Validate functionally, then under load. Measure the latency budget end-to-end (Section 7), and tune ASR (WER), NLU (intent accuracy), and endpointing on real traffic. Roll out as a canary — by number range or market — with the legacy IVR still reachable, and keep rollback available until KPIs hold.

6. End-to-end call flow

The sequence below traces a single call: SIP setup, an AI-generated greeting, and one full dialogue turn, ending in normal teardown or an agent handoff.

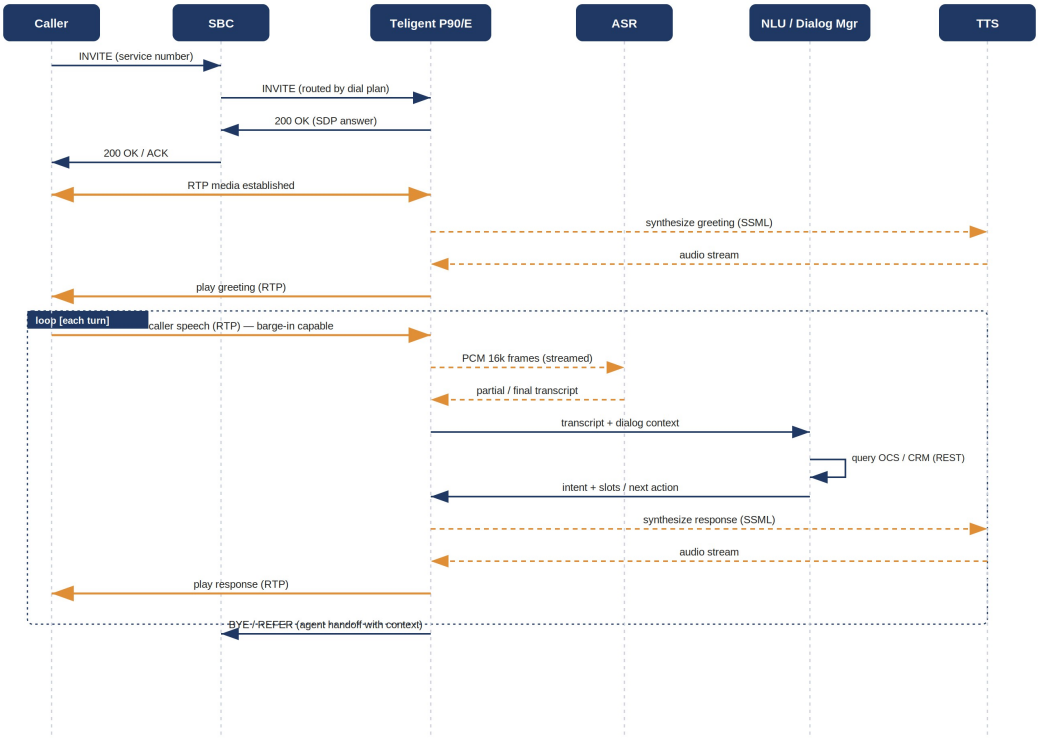


Figure 2. End-to-end call flow. SIP setup and media anchoring, an AI greeting, then a repeating dialogue turn (ASR → NLU/dialog → TTS), with barge-in throughout and REFER-based handoff on escalation.

The SBC routes the caller's INVITE to the P90/E, which answers and anchors media. The platform requests a greeting from TTS and plays it over RTP. When the caller speaks, the platform streams PCM to ASR, receives a transcript, and forwards it with context to the NLU/dialog manager, which resolves intent, queries the back end if needed, and returns the next action. The platform synthesizes the response via TTS and plays it — interruptible by barge-in — then loops for the next turn. On completion the platform sends BYE; on escalation it sends REFER to hand the call to an agent with context attached.

7. Latency and performance engineering

Whether the experience feels conversational is decided by latency. The target is a caller-perceived response within roughly 800 ms to 1 second of the caller finishing speaking; beyond that, the pause feels unnatural. Because the components run in sequence, their latencies add, and the only way to hit the budget is to stream at every stage and place real-time inference close to the media.

An illustrative per-turn latency budget

Stage	Typical contribution	Notes
Network / RTP + jitter buffer	20–60 ms	de-jitter adds a small fixed delay
ASR endpointing + final transcript	150–400 ms	dominated by end-of-speech detection; tune silence timeout
NLU / LLM inference	100–500 ms	depends on model size; stream generation
Dialog + back-end call	50–300 ms	a slow API is often the real culprit; cache, set timeouts
TTS time-to-first-audio	100–300 ms	stream; do not wait for full synthesis
Perceived total per turn	~0.5 – 1.2 s	lower end achievable with streaming + co-located inference

Design rules that follow from the budget

- **Stream everything.** ASR, NLU generation, and TTS should overlap rather than serialize, so processing happens while audio is still moving.
- **Play on first chunk.** Start TTS playback on the first returned audio chunk, and pre-synthesize and cache static prompts.
- **Co-locate real-time inference.** Place ASR, TTS, and barge-in detection close to the media — on operator infrastructure or a tightly coupled edge — rather than a distant cloud region; heavy non-real-time work (fine-tuning, analytics) can run in the cloud.
- **Interrupt fast.** Barge-in must stop playback within about 100–200 ms of detected speech, or the system will talk over the caller.

Capacity and availability: size concurrency by simultaneous calls, not calls per hour, since each call holds a media and ASR/TTS session for its full duration. The platform's active-active geographic redundancy carries over to the conversational service, with no single point of failure.

8. Migration strategy

No cutover should risk the live service. Deploy the conversational path alongside the legacy IVR and shift traffic gradually, treating each expansion as a gate passed on measured results.

- **Parallel run.** The P90/E fronts the service number; a share of calls — by number range, prefix, or percentage — is routed to the conversational path while the rest stay on the legacy IVR.
- **Canary and expand.** Start with one low-risk flow (for example, balance enquiry) in one market. Prove KPIs — containment, latency, WER, accuracy — then widen the flows and markets.
- **Fallback always on.** The legacy IVR and DTMF remain reachable throughout; any failure degrades to directed dialogue, DTMF, or an agent.
- **Gate on proof.** Each expansion is a gate passed on measured results, with rollback available. Legacy flows are retired only after the conversational path proves equivalence in production.

Migrate one intent, in one market, behind a canary. Expand on evidence. Keep the old path warm until the new one has earned the traffic.

9. Security, data, and compliance

A conversational voice service processes speech — among the most sensitive data an operator handles. Design for protection from the start.

- **Transport security.** SIP over TLS and media over SRTP on external legs; mutual TLS on the AI and back-end APIs.
- **Data minimization and residency.** Keep audio and transcripts only as long as needed, and process them in-country where data-residency rules require. Real-time inference on operator infrastructure keeps voice data on-net.
- **PII and consent.** Transcripts routinely contain personal and financial data; redact where possible, log deliberately, and govern use under the operator's consent regime.
- **Ownership.** Voice data and any models fine-tuned on it are strategic assets; retain ownership rather than ceding them to a third-party training pipeline.
- **Regulatory context.** Morocco's Law 09-08 on personal data, the broader West-African and ECOWAS data-protection frameworks, and emerging AI-governance rules increasingly constrain how voice data is processed. Designing for data sovereignty from the start avoids costly retrofits.

10. About TELIGENT TELECOM

TELIGENT TELECOM provides voice messaging and intelligent voice services to mobile operators across Africa and the Middle East. The P90/E is a carrier-grade, SIP-integrated media and service platform that anchors voice services in the operator network and integrates with charging and OSS/BSS.

As the integration anchor for conversational AI, the P90/E terminates SIP, anchors and processes media, orchestrates streaming ASR, NLU and TTS engines — over MRCPv2 or native gRPC/WebSocket — and applies charging and back-end integration, all behind carrier-grade high availability with active-active geographic redundancy. Our model is partnership: we integrate cleanly into the existing network, protect operational continuity through phased migration, and treat operator and subscriber data as the operator's own.

11. Next steps

The practical next step is a technical assessment: a review of the current IVR and SIP topology, the flows best suited to early migration, the speech and NLU engines and languages in scope, and the target latency budget. From that we co-design a phased integration and migration plan with defined KPIs and a canary rollout.

How to engage with us

- Request a technical assessment of your IVR and SIP infrastructure.
- Schedule an architecture workshop with your VAS, core-network, and AI teams.
- Scope a canary integration of one flow, in one market, on the P90/E.

C O N T A C T

To start the conversation, contact our team at the address below. We will respond within two working days to schedule an initial technical discussion.

Email: info@teligent.se
Phone: +46 8 520 66000
Web: www.teligent.se