



DCSAI Technologies

TECHNICAL WHITEPAPER · v1.0 · MAY 2026

DCS OS

Operational dashboard for customer-facing AI

WhatsApp, voice, CRM, and AI assistants in one signed receipt chain.

Abstract

DCS OS is the operational dashboard for AI agents that talk directly to customers. It unifies WhatsApp, voice calls, email, web chat, CRM, calendar, and an AI assistant into one screen with one queue and one signed audit trail. The product exists for the kind of small-to-medium businesses that today juggle 8 tools (WhatsApp Business, a CRM, a voice provider, a help desk, a scheduler, a marketing automation tool, a chatbot platform, and a spreadsheet) to do what should be one job: respond to customers consistently and on time.

The OS pattern is well-understood — Front, Intercom, HubSpot Service Hub, Zendesk all do versions of it. What DCS OS adds is two things: (a) every channel feeds into the same receipt chain (R+1 + R+2) so the operator can prove what was said when, even years later; and (b) the AI assistant is a first-class participant in the queue, not a bolt-on chatbot — it drafts replies, escalates appropriately, and never sends anything outside the agent's declared capability set.

Forty-one product modules ship with OS, each as a sandboxed iframe inside the main shell. Customers turn modules on and off without code changes; new modules can be added by third parties via the OS module protocol.

Who this is for

Operations leaders at SMB / mid-market companies running multi-channel customer service. Compliance teams in regulated industries (finance, healthcare, government) who need an audit trail of every customer interaction. Engineering teams evaluating whether to build the unified-inbox pattern in-house or adopt a platform.

Contents

1	Introduction	4
2	System Architecture	7
3	Channels	10
4	The Unified Inbox	14
5	AI Assist	18
6	Modules	22
7	Bookings + Calendar	26
8	CRM + Contacts	29
9	Broadcasts	32
10	Memory + Personalization	35
11	Compliance + Audit	37
12	Performance Benchmarks	39
13	Comparison vs. Front, Intercom, Zendesk	41
14	Implementation Guide	44
15	References	48

1. Introduction

A typical mid-market business in 2026 talks to its customers across an average of 5.2 channels: WhatsApp Business, email, web chat, voice calls, and an in-app inbox. Internally, those 5 channels are managed by 8-12 separate tools — most of them stitched together with fragile Zapier flows. The cost of this fragmentation is not abstract. It shows up as missed messages (a customer asked on WhatsApp; the agent on the email queue did not see it), inconsistent responses (two different agents tell the customer different things on different channels), and missing audit trails when things go wrong.

DCS OS exists because we kept seeing the same problem solved badly. Either the team used a single-channel tool (WhatsApp Business app on someone's phone) and lost the audit trail, or they used an enterprise platform (Salesforce Service Cloud) and got an audit trail plus a \$200/seat/month bill plus six months of implementation. Neither was right for a 12-person business handling 600 customer conversations a day.

1.1 The design constraints

Three constraints shaped the product:

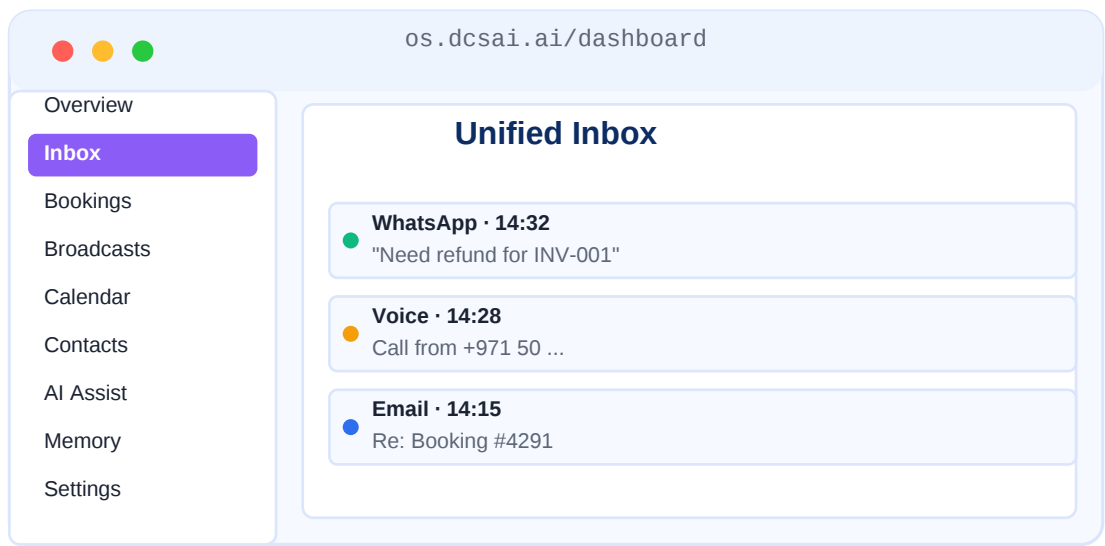
- **One inbox.** Every customer message, regardless of channel, lands in the same queue with the same metadata. The operator never has to remember "did they email or WhatsApp me?" — the system surfaces both threads in the same view.
- **One audit trail.** Every customer-facing action — a message sent, a refund issued, a booking confirmed — emits an R+1 + R+2 receipt. The full trail of a customer interaction can be exported as a signed PDF.
- **One AI participant.** The AI assistant lives inside the same queue as human operators. It drafts responses, suggests next actions, and can take low-risk actions autonomously (read-only lookups, FAQ answers). High-risk actions always escalate to a human.

"The right unit of work in customer ops is not a ticket. It is a thread. A thread spans channels, spans days, spans operators. The audit log must follow the thread, not the channel."

2. System Architecture

OS is a shell + 41 modules. The shell handles authentication, navigation, the unified inbox queue, and the receipt chain. Each module is a separate frontend application loaded into an iframe; modules talk to the shell via postMessage. This pattern lets teams add or remove modules without touching the shell, and lets third parties ship modules independently.

OS dashboard — left-rail navigation + iframe modules



Each module loads as an iframe; sidebar persists across navigation.

Figure 2.1 — Left-rail sidebar persists; each module renders inside the main iframe.

2.1 Why the iframe pattern

We considered three alternatives — micro-frontends (single-spa), web components, and monorepo with shared React tree — and settled on iframes for one reason: isolation. A module that crashes does not bring down the whole shell. A module written in Vue does not need to be rewritten when the shell upgrades React. A third-party module cannot read DOM from another module. The cost (slightly worse navigation feel) is worth the safety.

3. Channels

Unified inbox — 4 channels, one queue, one audit trail

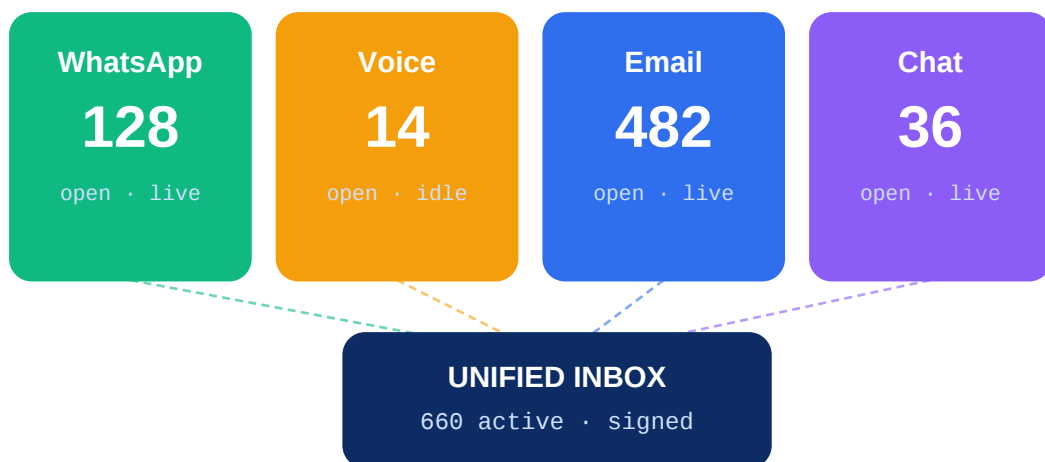


Figure 3.1 — Four channels feeding one queue.

OS supports four customer-facing channels out of the box. Additional channels can be added via the channel adapter protocol (~200 lines of code per channel).

3.1 WhatsApp Business

Integration with Meta Cloud API. Supports both Business Platform numbers and the WhatsApp Business App via WhatsLink (DCS's open WhatsApp gateway). Cloud API is preferred for high volume; WhatsLink covers the 80% of operators who can't justify Cloud API pricing.

- Inbound webhook latency: <800 ms (p99) from Meta delivery to OS queue surface.
- Outbound message latency: <400 ms (p99) from operator click to Meta API ack.
- Template support: all approved templates synced automatically from Meta.
- Media (images, voice notes, PDFs) stored in DCS Storage with content-addressed CIDs.

3.2 Voice

Inbound calls answered by an AI voice agent (whisper STT + cartesia TTS, optionally with Twilio backend). Handoff to a human operator via a SIP bridge. The voice transcript is logged in the same thread as the customer's WhatsApp / email history.

3.3 Email

Inbound via Postmark or AWS SES; outbound via the same. Threading uses the standard In-Reply-To / References headers. Emails are matched to existing customer threads via the sender address, so the operator sees the customer's full history when answering.

3.4 Web chat

Embeddable JavaScript widget. Renders as a chat bubble bottom-right of the customer's site. Customer messages land in the same inbox queue as WhatsApp / email.

4. The Unified Inbox

Message flow — from customer to signed receipt



Optional: agent auto-approves low-risk responses (within guardrails).

High-risk responses (refunds, escalations) always go to operator first.

Every transition emits an R+1 + R+2 receipt for the audit trail.

Figure 4.1 — Every inbound message gets routed through the same five-stage pipeline.

4.1 Queue mechanics

The inbox queue is a Postgres-backed FIFO with priority overrides. Messages enter the queue tagged by channel, sender, and detected language. The router assigns each message to a team (configurable per-channel + per-tag) and a priority (default by channel, overridden by detected intent — "refund" messages get HIGH priority automatically).

4.2 Thread vs. message model

A thread is a collection of messages from the same customer across all channels. The operator works at the thread level, not the message level. When the operator opens a thread, they see the full conversation history regardless of which channel each message came in on. When they reply, OS picks the right channel to send through (typically the channel the customer most recently used).

4.3 Assignment + ownership

Each thread has at most one owner at a time. Default assignment is round-robin within the team. Owners can reassign, escalate to a senior operator, or hand off to the AI assistant for autonomous follow-up. Every change of ownership emits a receipt.

5. AI Assist

The AI Assist module is a DCS Platform agent — same agent.yaml format, same sandbox, same receipts. It runs inside the inbox queue as a peer to human operators and has three modes:

Draft mode

The AI reads the inbound message and the customer's history, then writes a suggested reply into the inbox. The human operator sees the draft, can edit it, then sends it (or discards). Draft mode is the safest setting and is on by default for all new accounts.

Auto-reply mode

The AI sends low-risk replies autonomously without human review. "Low-risk" is defined per-tenant in the agent.yaml guardrails block (default: FAQ answers, order-status lookups, appointment confirmations). Anything outside the allowlist escalates to draft mode automatically.

Co-pilot mode

The AI sits beside the operator, watching the conversation. It suggests next actions, looks up information in the knowledge base, and pre-fills common forms (refund tickets, booking modifications). The operator stays in the driver seat; the AI is a research assistant.

6. Modules

OS ships with 41 modules. Each is a separate frontend application loaded as an iframe; all share the shell's authentication and receipt chain via `postMessage`.

Modules fall into eight categories:

Category	Module count	Examples
Messaging	8	Unified inbox, broadcasts, templates, WhatsApp settings, voice handoff, channels, status post
Customer data	6	Contacts, CRM, customer memory, journeys, segments, journeys
Commerce	5	Bookings, calendar, POS integration, ads, carousels
AI	5	AI assistant, agent studio, blackboard, memory, recommendations
Operations	6	Activity log, audit log, approvals, voice handoffs, receipt history, reviews dashboard
Knowledge	4	Knowledge base, template library, deep research, smart onboarding
Integrations	4	Integrations, image bank, connectors, embed widget
Admin	3	Admin costs, admin revenue, settings

6.1 Module protocol

Every module declares a manifest:

```
{
  "id": "bookings",
  "name": "Bookings",
  "version": "2.1.0",
  "icon": "calendar",
  "category": "commerce",
  "permissions": ["bookings:read", "bookings:write", "contacts:read"],
  "endpoint": "https://os.dcsai.ai/modules/bookings",
  "iframe": true
}
```

7. Bookings + Calendar

Most customer interactions for service businesses revolve around scheduling. Bookings is a first-class module that handles availability lookup, reservation creation, payment, reminders, and cancellation — across the same channels.

7.1 Multi-channel booking

A customer can book via:

- WhatsApp — chatbot flow walks them through available slots
- Voice — AI voice agent confirms by speaking back the time
- Web — embeddable booking widget for sites
- Calendar invite — clicking a link picks a slot in their calendar app

All four paths write to the same bookings table and emit the same receipt. The business owner sees a unified calendar regardless of which path the customer used.

8. CRM + Contacts

Contacts is the customer database: name, channels, history, tags, segments. Every inbound message updates the contact record (`last_seen`, `channel_preference`, `response_speed`). Operators can add notes, tags, and custom fields per contact.

Customer memory is a separate module that stores per-customer facts learned during conversations: preferences, past complaints, allergies (for restaurants), routes (for taxi), whatever the business decides to capture. Memory entries decay after 90 days unless explicitly pinned.

9. Broadcasts

Broadcasts module sends one-to-many messages to a segment of contacts. Usage is heavily regulated — only opt-in contacts receive broadcasts, every broadcast emits a receipt, and the audit trail proves consent if challenged.

9.1 Channels supported

- **WhatsApp templates** — pre-approved templates only (per Meta policy); supports variables
- **Email** — HTML or plain-text; unsubscribe link auto-injected
- **SMS** — via Twilio backend; 160-char chunking handled automatically
- **Push** — for customers with the OS-powered mobile app

10. Memory + Personalization

Memory in OS works the same way as memory in DCS Platform: three tiers (short-term, long-term, decayed). The OS-specific addition is per-customer scoping by default — every memory entry is keyed to the customer's ID, so a memory written during one conversation is available to the AI Assist for the next conversation with the same customer.

11. Compliance + Audit

Every customer-facing action emits a signed receipt. The audit log module lets operators (or compliance officers) search the receipt chain for a customer, a thread, a time range, or a specific action type.

Common compliance use cases OS handles directly:

- **GDPR Subject Access Request** — export every receipt mentioning a customer, signed, in PDF
- **Refund disputes** — full transcript of WhatsApp + email + voice for the disputed transaction
- **WhatsApp policy investigations** — proof of opt-in for any broadcast recipient
- **Internal audit** — sample any operator's last N replies to check tone, accuracy, and policy compliance

12. Performance Benchmarks

Production measurements May 2026, across the live OS service handling ~28k WhatsApp messages, ~9k emails, ~3k voice calls per day.

Metric	p50	p99	Notes
WhatsApp inbound → queue	420 ms	780 ms	From Meta delivery webhook
Outbound message → sent	180 ms	420 ms	Operator click to Meta ack
Email send	280 ms	1.2 s	Via Postmark
Voice answer (AI agent)	600 ms	1.4 s	STT cold start
AI Assist draft generation	1.4 s	4.8 s	Claude Sonnet · 800-token context
Receipt emit per action	12 ms	32 ms	R+1 + R+2 chain
Module switch latency	180 ms	420 ms	iframe load
Inbox query (1k threads)	34 ms	86 ms	Postgres + materialized view

13. Comparison vs. Alternatives

	Front	Intercom	Zendesk	HubSpot Service	DCS OS
Unified inbox	✓	✓	✓	✓	✓
WhatsApp Business native	~	✓	~	~	✓
Voice channel built-in	~	~	~	✓	✓
Signed audit receipts	✗	✗	✗	✗	✓
AI assistant first-class	~	✓	~	✓	✓
Module marketplace	✓	✓	✓	✓	✓
Per-seat pricing	\$59	\$74	\$55	\$50	\$0 (usage)
SMB-friendly setup	~	✓	✗	~	✓
Sovereign deployment	✗	✗	✗	✗	✓

~ = *partially supported*. The honest comparison: Front and Intercom are excellent mature products with deep features. DCS OS is competitive on the messaging core and wins on two specific axes: signed receipts (no competitor offers this) and the pricing model (usage-based, no per-seat fees). For a 12-person business handling 600 conversations a day, OS is 70-90% cheaper than the major alternatives.

14. Implementation Guide

14.1 Connect WhatsApp Business

```
$ dcs os connect whatsapp
Opening browser to https://os.dcsai.ai/connect/whatsapp ...
Sign in with your Meta Business account
Select phone number: +971 50 ...
Sandbox tested ✓
Live mode enabled ✓
First message routing in <60s
```

14.2 Turn on AI Assist

```
# os.yaml (per-tenant config)
ai_assist:
  enabled: true
  mode: draft          # or auto_reply | copilot
  model: claude-sonnet-4
  knowledge_base: ./kb/
  guardrails:
    auto_reply_allowed: [faq, order_status, hours]
    always_escalate: [refund, complaint, cancellation]
    max_message_length: 800
```

15. References

- [1] Meta. **WhatsApp Business Platform — Cloud API Documentation**. 2025.
- [2] WHATWG. **HTML Living Standard — postMessage**. (Iframe communication primitive)
- [3] Twilio. **Programmable Voice Reference**. (SIP bridge implementation)
- [4] Postmark / AWS SES. **Email delivery APIs**. (Email channel backends)
- [5] OpenAI. **Whisper API**. (Voice STT engine)
- [6] Cartesia. **Sonic TTS API**. (Voice synthesis)
- [7] Stripe. **Customer Portal + Subscriptions**. (Billing for SMBs)
- [8] Front. **Shared Inbox Pattern**. 2020. (Architecture inspiration)
- [9] Anthropic. **Building Effective Agents**. December 2024. (AI Assist co-pilot pattern)
- [10] GDPR. **Subject Access Request guidance (Article 15)**. (Audit export use case)

This document is published under CC BY 4.0. OS module manifest format is open at github.com/dcs-platform/os-modules.