

OPEN AUTO

Conversational Consumer Retail Automotive (CCRA)

Reference Implementation v1.0

Executive Summary

AI-referred visits to US retail sites rose 393% year over year in Q1 2026 (Adobe). 63% of in-market US car shoppers expect to use AI on their next purchase (Cox Automotive). Despite the massive uptake of AI by consumers, only one major automaker, BMW, has an app in ChatGPT, launched in July 2026 (BMW Group), and none appears in Claude's connector directory. There is a significant gap to address.

Open Auto is an independent standards initiative focused on the use of generative AI within the retail automotive industry that is addressing that gap head on. Its first technical artifact is a reference implementation for Conversational Consumer Retail Automotive (CCRA). The reference implementation provides a basis to configure, build, test, package, and deploy a branded MCP App suitable for use within ChatGPT. It establishes standard surfaces and practices and is a general accelerator on bringing conversational UI to retail automotive consumers. What sets it apart is that its standard is expressed not only as data models and tool definitions, but as a body of automated tests that define how a conversational app should behave.

In this white paper, we explain the need for new standards in an AI-powered world, note some aspects of the retail automotive industry that shape the Open Auto approach, detail why MCP Apps are the most direct way to reach AI-enabled automotive consumers, and discuss in detail the CCRA reference implementation. We conclude with a discussion of future Open Auto work, a suggestion of next readings, and a list of ways to get involved. The audience for this document is strategic, digital, product, and engineering staff at OEMs, dealer networks, dealers, and vendors servicing the retail automotive industry that are interested in realizing the promise of AI in retail automotive through collaboration.

Standards in an AI World

In a world of autonomous agents capable of inference and reasoning, is there a need for standards at all? There is, but not the kind the industry is used to. Traditional standards fix the syntax of messages exchanged between deterministic programs: field names, formats, and sequences that every party must match exactly. Agents are good at translating syntax, so that part of the problem shrinks. What agents struggle to infer is context, boundaries, and accountability. Those are decisions that businesses, regulators, and customers make together, and they are exactly what a new kind of standard must capture.

Previous approaches to software standards focused on stating the set of possible data states and the field of allowed transitions between states. This is useful and feasible for applications that strictly control and limit the possible user journeys. This level of control is not possible with conversational interfaces. Conversational users are free to say anything at any time. That is not to say that an application with a conversational interface must be prepared for chaos. Users typically operate with the norms of an understood business domain. The challenge for a new standard is to usefully state how a system should behave in response to a wider and less precisely stated set of inputs – to set a behavioral contract.

Open Auto expresses that contract as a body of realistic conversations paired with measures of correct behavior, run as automated tests. For AI-native systems, a working reference implementation with extensive quality automation layers is the most precise standard available.

The fundamental building blocks of standards to date are still relevant. To be meaningful and useful, data must have an expressible shape. Agentic systems cannot effectively reason about data that has no pattern or rules. AI-powered applications need lists of values with allowed types that combine into objects. Similarly, AI-powered services need to advertise their capabilities and how clients can expect to use them.

The retail automotive industry has no shortage of standards. The good news is that the coming transition does not mean we have to throw away and replace what came before. The next generation of standards will be built directly on top of the existing landscape to enable the conversational and agentic interfaces of the future while making use of established data shapes and service surfaces.

Retail Automotive

Within this paper, “retail automotive” is inclusive of everything consumer-facing in the cycle of finding, buying, and owning a car. Vehicles are among the most significant purchases that most people make. Consumers have a unique relationship with the vehicles and vehicle brands. The industry has distinctive characteristics that have influenced Open Auto’s approach.

Actors

Original Equipment Manufacturers (OEMs)

Consumer automotive OEMs include some of the world’s most recognized companies. Ford, Toyota, Mitsubishi, and Volkswagen have spent decades building incredibly complex global business operations while investing heavily in cultivating distinctly recognizable brands. Brand is a foundational element of the value brought by OEMs. They typically control national and international marketing and are generally the first and most persistent point of contact for retail automotive shoppers. Despite their size and complexity, OEMs have many similarities, making it entirely tractable to model general consumer-facing business practice in standardized software.

Dealers

The relationship between OEMs and dealers varies substantially throughout the world. The US is notably distinct in that every US state has its own automotive franchise laws. Dealerships typically have their own digital presence for establishing local credibility and displaying inventory. Service and parts contribute a large share of dealer gross profit, often more than new-vehicle sales. Dealers value transmission of high-context leads from OEM digital touchpoints to dealer digital touchpoints and the smooth movement of users from OEM-maintained national web and mobile apps to dealer service centers. Standardization is directly in the critical profit path of dealers.

Software Vendors

Automotive software vendors come in many shapes and sizes. Some provide highly bespoke solutions that address only defined market sectors. Others provide broad platforms that can fulfill almost all of the complex needs of a dealership. All software vendors understand that standardization is a route to wider and easier adoption by more customers.

Software Customizers

Many OEMs and some dealerships have a deep relationship with a partner providing platform customization and bespoke software development services. Customizers bring the most value when they can focus on what truly differentiates their client or best supports the client's strategic aims. Standardization enables customizers to spend a greater proportion of their time where their clients feel the most value.

Constraints

Enterprise Security is a Must

After homes, vehicles are one of the most expensive things that most people ever buy. A great deal of personal information changes hands in the process of buying or leasing a new car. OEMs and dealers must be confident that they can trust their vendors, partners, and software systems support the level of trust that they must maintain with their customers. The high level of security demanded by retail automotive is as much about maintaining a reputation of trust and safety with consumers as it is about mitigating business risk. The reference implementation reflects this: browsing requires no personal data, sign-in requests only narrow scopes, and no appointment is created or changed without the customer's confirmation. Deploying MCP Apps requires exposing a service to the public internet and appropriate measures are taken to secure this potential point of vulnerability.

Complex Global Operations

National OEM corporations are typically owned by a global corporation which in turn is owned by a multi-brand conglomerate. Dealerships may be independent or may be part of much larger dealer groups. Ownership of platforms and technology decisions varies between organizations. Open Auto designs standard solutions with this diversity of structure and management in mind.

MCP Apps

Model Context Protocol (MCP) is the most widely adopted standard for adding knowledge and capabilities to host AIs like ChatGPT and Claude. MCP alone extends only the mostly text-based conversational user experience. The MCP Apps standard enables inclusion of HTML content alongside the conversational features offered by MCP tools. This allows apps to present interfaces that would be difficult to navigate using only formatted text and simple images. Complex data visualizations are one

example. MCP Apps create an opportunity to bring forward visual brand elements as well, a consideration of critical importance in the retail automotive industry.

MCP Apps are supported by the leading host AIs. Both ChatGPT and Claude have an official directory enabling easy discovery and installation of MCP Apps. Inclusion in these directories requires review, which adds a layer of vetting for the security, quality, and value of listed apps. ChatGPT reaches 1.2 billion people every week and now connects to more than 4,000 apps (OpenAI, via TechCrunch). Claude's directory lists over 950 connectors used by millions of people every day (Anthropic).

MCP Apps present some novel challenges. They depend on the reasoning and inference services of the model used by the user's host AI to function, meaning that the way the application functions is not entirely within control of the developer and application behavior will change as models continue to evolve. This lack of control is analogous to web applications and web browsers. The user experiences a web app through a browser. Different browsers have different features and render content differently. The web app developer cannot control the user's selection of browser and how browser behavior changes as new versions roll out. For a standard, this means the behavioral contract must be checked continuously as host models change, not verified once at launch.

Just as the software industry worked in collaboration to reduce the ongoing pain of browser compatibility, Open Auto offers the retail automotive industry a way to work together to solve universal problems more quickly and at lower cost.

The Reference Implementation

The reference implementation acts as both a standard and source code suitable for creating branded demos and experiments or providing an accelerated starting point for a production build.

As a Standard

The reference implementation expresses Open Auto's behavioral contract through four cooperating layers: which tools exist, how each tool is defined, how widgets are used, and the tests that confirm the whole system behaves as expected.

MCP Tool Design

MCP tools are analogous to endpoints in a REST API design. They are the list of capabilities that an MCP App makes available to the host AI. The more specific an MCP tool is, the more likely it is to provide a good user experience. The more tools that are available to an agent, the more likely it is to use the available tools inefficiently. This tension is a heavy constraint on MCP tool design. Open Auto's MCP tool decomposition adds the additional constraint of real-world implementation patterns within retail automotive. The MCP tool selection intentionally enables "mixed" implementation of the tool

surface through composition from various sources. An OEM could build their MCP App to the Open Auto standard by composing custom tool implementations alongside tool implementations from vendors whose tools follow the Open Auto definitions.

Table 1 in the appendix lists the tools in the reference implementation. Fifteen are available to the assistant. Five more can be called only from widgets, and the three tools that change appointments are all in that group, so no appointment is created, changed, or cancelled without the customer completing a step in the widget. Browsing tools need no sign-in; tools that touch the customer's own data request only the scope they need.

UI Resource Design

MCP Apps may associate UI resources with tools, enabling display of interactive web content widgets that can evolve alongside the ongoing conversational interactions. Choosing when and how to use widgets is a balancing act between informing and confusing the user. A new car hunter working within the text-heavy environment of ChatGPT may find it difficult to incorporate highly visual elements into the overall experience. Open Auto has arrived at an optimized approach to widget use through user testing and feedback, establishing a standard based on data rather than opinion or convention.

Tests as Specification

The reference implementation states its expectations as a body of realistic conversations, each paired with measures of a correct outcome: which tools the assistant should call, what it should and should not say, and how it should respond. The measures cover logical correctness as well as less precisely defined qualities such as politeness, concision, and staying on brand. The suite can be curated and extended, and it runs during development, as part of the build and deployment process, and as ongoing monitoring when host models change.

The tests come in three layers:

- **Unit tests** check tools and data handling without a model.
- **Evaluations** run scripted conversations against a model several times and score the results, so the model's unpredictability is part of the test.
- **Live tests** drive the deployed app inside ChatGPT itself.

The evaluations and live tests use the Sunpeak testing library. For live tests, it automates deploying and testing an MCP App directly within ChatGPT. This goes a step beyond testing simulated client behavior through API access to a model, in the same way that automating a real browser goes beyond simulating one when testing a web application. It shows exactly how the solution will appear to customers and whether that falls within the boundaries of correct, allowable behavior.

As Source Code

The reference implementation intentionally uses the widely supported combination of TypeScript and React and a selection of commonly used open source libraries. The source and all dependencies are offered without restrictive licensing to use and adapt. Testing relies on the Sunpeak library.

Figure 1 shows how the code is organized. Widgets run inside the host AI and reach the server through the MCP Apps bridge. The server has a thin HTTP layer and an MCP layer that registers tools, widget resources, and server instructions. Every tool reads and writes data through the shared, STAR-derived data model, and data adapters connect that model to mock data, the appointment store, or real services. Tests and build tooling surround the whole.

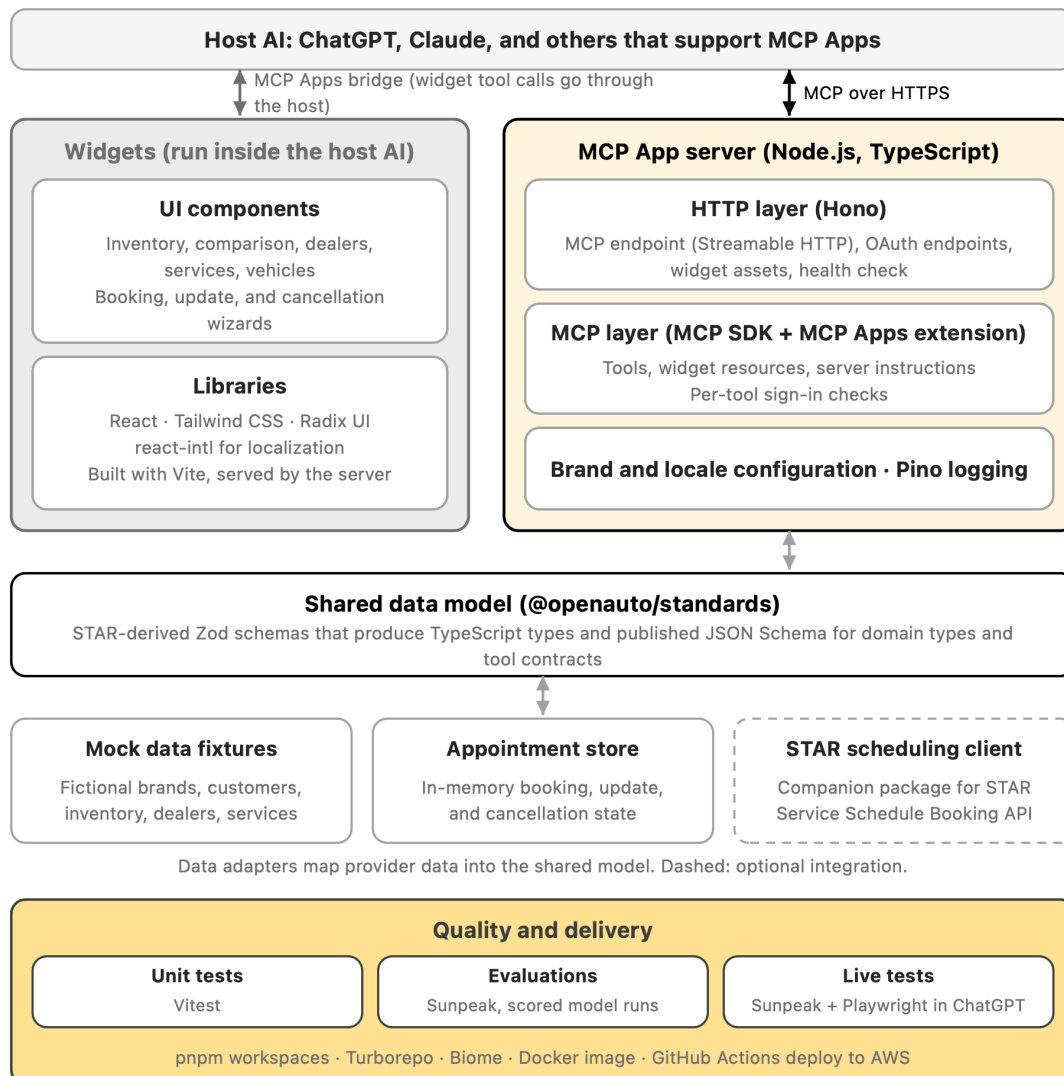


Figure 1. Software architecture

System Architecture

Out of the Box

The system architecture of the deployed reference implementation is as simple as it can be. In its default configuration, all data is provided through mock service integrations. Figure 2 shows the reference implementation as delivered. Everything runs in a single process: the tools, widgets, and configuration, a shared data model derived from STAR, a built-in demonstration sign-in server, and bundled mock data. No real identity provider or external system is involved, so a demo brand can be configured and deployed without any integration work.

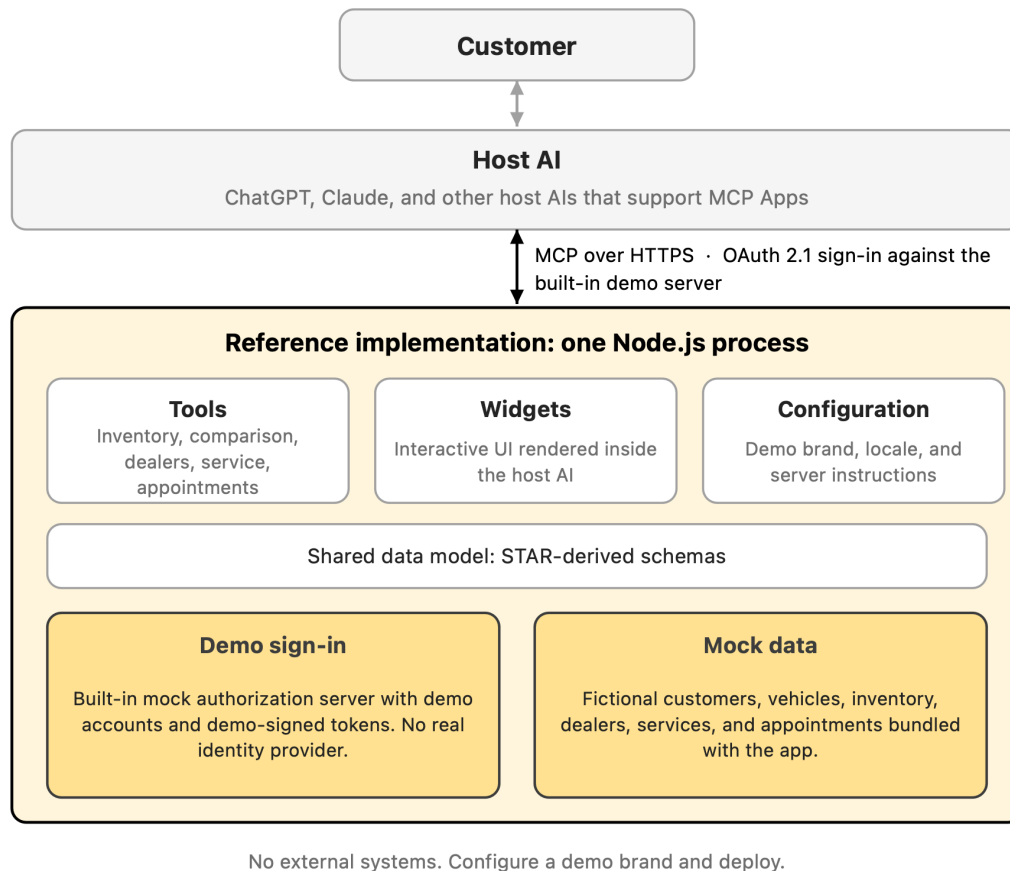


Figure 2. Out-of-the-box architecture

Path to Production

The reference implementation serves as a standard blueprint for a customized, production-ready deployment. Such a deployment will compose a mixture of vendor-provided and custom tool definitions to create a fully integrated compatible deployment.

Figure 3 shows a production deployment. Mock data and the demonstration sign-in server are gone. Customers sign in through the brand's own identity provider, and the app combines custom tools built for the brand with compatible vendor tools. Behind those tools sit the brand's own OEM and dealer

systems and vendor services, some of which implement STAR standards such as the STAR Service Schedule Booking API. The shared data model keeps the pieces interchangeable.

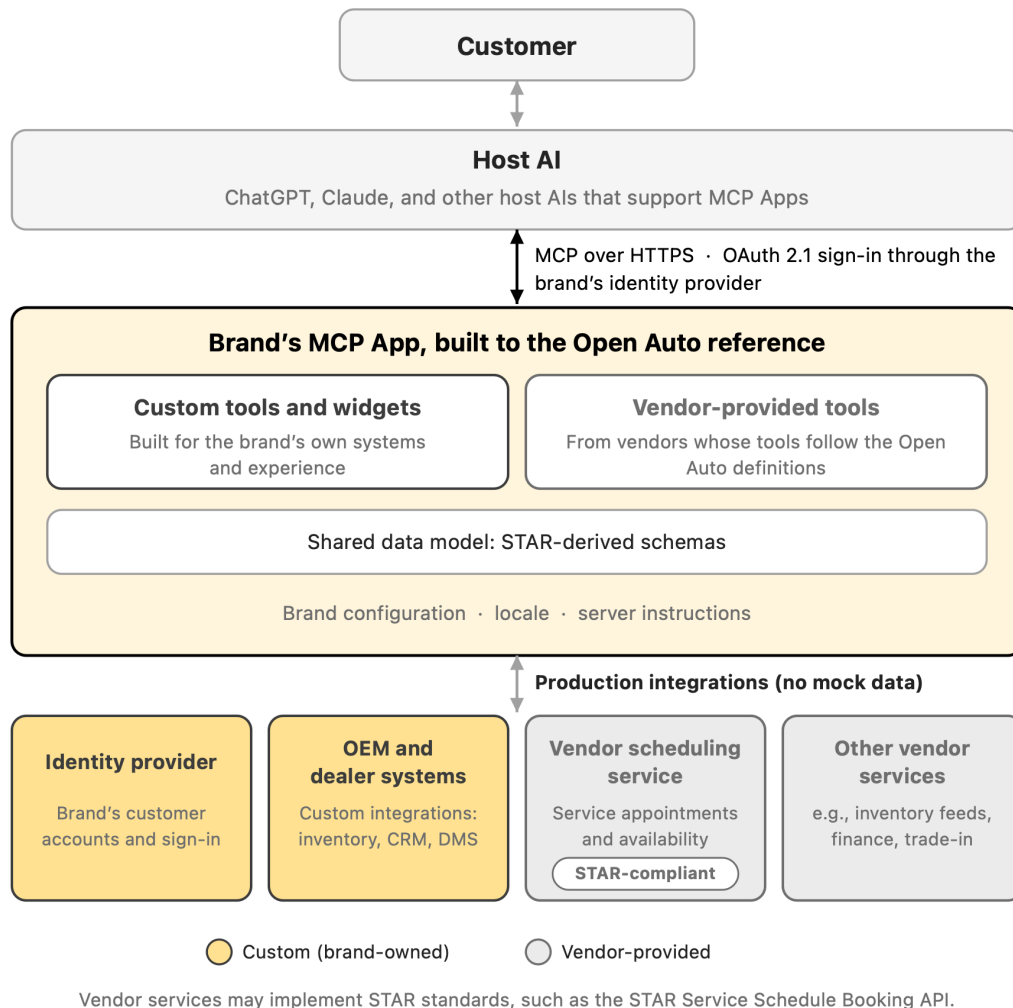


Figure 3. Production-ready architecture

Features

Configurable Branding

The brand name and identity of a deployment can be managed through configuration. This enables rapid deployment of branded instances as well as easier long term maintenance of branding details.

Customizable Data Mocks

The base configuration uses only mocked data interfaces. The data used by these mocks can be easily evolved or replaced to meet brand needs, for experiments, or as part of the long term evolution of an offering.

STAR Interfaces

The reference implementation's data model is derived from the STAR Automotive Domain Model and the STAR Service Schedule Booking API v2.0.0, and it keeps STAR field names. A companion scheduling package includes a client for STAR-compliant service scheduling APIs. This enables rapid connection of STAR compliant systems into the solution, building directly on previous investments in standardization.

OAuth Login

The reference implementation follows the MCP authorization specification, using OAuth 2.1 with PKCE and dynamic client registration. Customers can browse models, inventory, and dealers without signing in. Sign-in is requested only when a tool needs the customer's own data, such as their profile, vehicles, or appointments, and access is limited to narrow scopes. The included authorization server is for demonstration; a production deployment connects to the brand's own identity provider.

Common Workflows

A selection of common user workflows that demonstrate different user types and objectives have been chosen and implemented. These are a foundational set that will be built upon to eventually include all standard user interactions.

Service booking shows how the reference implementation keeps a free-form conversation within an allowed set of steps (Figure 4). When a customer asks to book service, the assistant opens a booking wizard instead of collecting details in chat. The wizard moves through six fixed steps, and the appointment is written by a single call, after the customer reviews and confirms. Rescheduling and cancellation follow the same pattern.

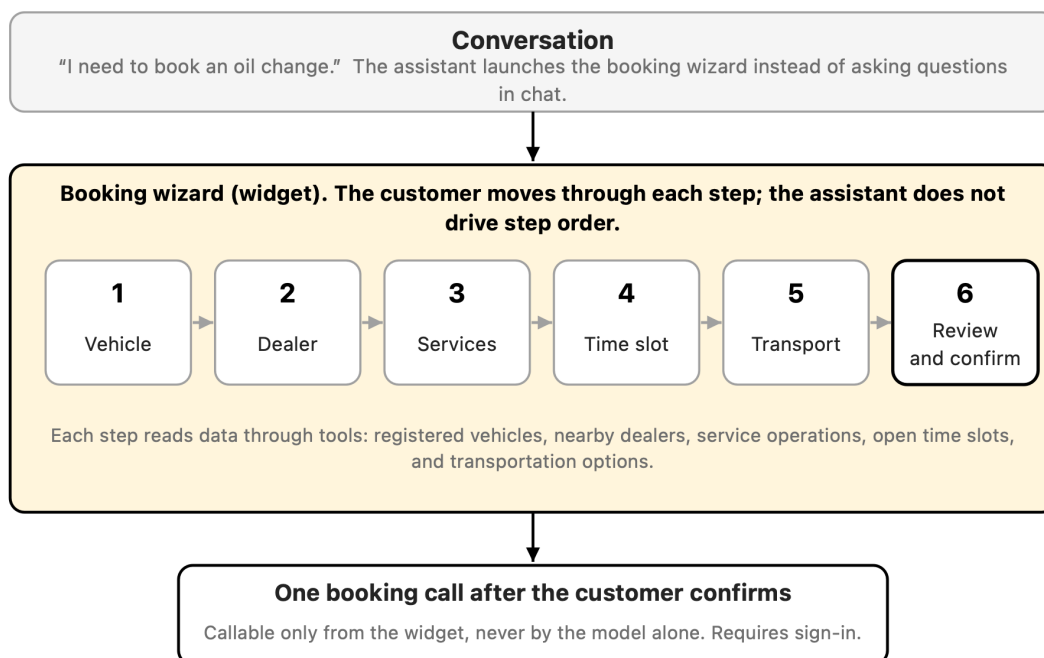


Figure 4. Service booking flow

Multi-Workflow Pathways

The reference implementation supports user journeys that combine aspects of established workflows. The need to support such workflows is a big challenge, but the value of doing it well is also large. With conversational UIs, users are no longer confined to preset “flows” and can explore as they see fit. Enabling this independence without creating confusion is a critical goal. For example, a customer can ask how their current vehicle compares with the latest version of the same model: the assistant looks up the vehicles they own, finds matching trims, and shows a side-by-side comparison, and the customer can go on to book service for their current car in the same conversation.

AWS Deployment

The reference implementation provides hooks for simple deployment into AWS for sharing over the public internet. AWS was chosen as a familiar and widely supported public cloud. With light customization, the reference implementation could easily be deployed onto other clouds or owned/on-premises servers.

Adoption Value

Customers are moving steadily from browsers and apps to conversational clients. Every part of the retail automotive industry that wants to stay in front of those customers will need to meet them there.

Working from the CCRA allows OEMs, dealers, vendors, and their development partners to spend less time solving low value common problems in favor of more time solving high value problems that are specific to their products and business practices.

Adoption of standard surfaces and approaches enables easier integration with other standard-compliant systems and reduces lock-in.

What It Means for Each Participant

- **OEMs.** Establish a branded presence in the host AIs where customers already shop, built on a shared foundation rather than a one-off integration.
- **Dealers.** Receive service bookings and shopping interest from AI conversations through standard surfaces, with the details the customer confirmed intact.
- **Software vendors.** Build tools once against a common definition and offer them to any compatible OEM or dealer deployment.
- **Development partners.** Start from working, tested code and focus effort on what differentiates each client.

Questions to Ask Your Vendors

Whether you build, buy, or partner, these questions help assess any conversational retail solution:

- Which host AIs does the solution support, and how is its behavior tested in each one as host models change?
- Is the data model built on STAR or another published standard, and can it be mapped field by field?
- Which actions can the AI take on its own, and which require the customer's explicit confirmation?
- What can a customer do without signing in, and which data scopes are requested when they do?
- How are answers grounded in your systems' data rather than the model's general knowledge?
- Can tools from different vendors be combined in one deployment?
- How are brand identity and language configured, and which changes require new code?
- What automated tests ship with the solution, and can your team run them?

Future Work

Work on the CCRA reference implementation continues on several paths:

- **Workflows.** There are many more common and important consumer retail automotive workflows to support. These include scheduling test drives, sharing credit-related data, deal negotiation, offer, and purchase. These follow the Open Auto roadmap: core workflows first (service scheduling, dealer search, and inventory), then connected journeys such as vehicle comparison and trade-in, then higher-trust transactions such as financing, purchase, and delivery.
- **Journey Skills.** Skills have emerged as a preferred way to steer host AIs through multi-step workflows. This will enable using fewer “wizards” in the widget UIs and keeping the interface more conversational.
- **Quality Automation.** The more layers of testing and volume of tests that we assemble, the better initial and ongoing product we can expect.
- **Cloud Platforms.** We will expand supported cloud platforms to include at least Azure and GCP.
- **Extensions and Plugin Packaging.** These standards continue to evolve and this project will track them closely.

Other forthcoming work from Open Auto will include:

- **Agentic Enablement.** Interactive conversation with consumers requires one set of protocols and performance expectations. Long-lived agents acting on behalf of consumers non-interactively require a different approach.

- OEMs and Dealers. OEM and dealer staff have as much to gain from conversational interfaces as do consumers. Future work will focus on bringing “back office” efficiency through standardized approaches.

Get Involved

Open Auto is an independent, neutral standards initiative working toward becoming a member-governed standards body. Early participants can help set priorities, improve draft standards, and test how they work in practice. Organizations can engage through several paths:

- Founding briefing
- Technical contribution
- Reference implementation participation
- Potential pilot participation
- Strategic partnership
- Public support

To get the source code for the CCRA reference implementation or to discuss Open Auto’s priorities, standards roadmap, and how your organization could contribute, please [request a briefing](https://www.openautoforum.org/request-a-briefing..) at <https://www.openautoforum.org/request-a-briefing..>

Further Reading

The following resources provide background on the protocols, standards, market data, and policy context discussed in this paper.

Protocols

- [Model Context Protocol \(MCP\) Apps](#). The base standard for the reference implementation.
- [Model Context Protocol \(MCP\)](#). The service protocol underlying MCP Apps.
- [Model Context Protocol \(MCP\) Authorization](#). The OAuth 2.1 sign-in flow the reference implementation follows.

Automotive Standards

- [Standards for Technology in Automotive Retail](#) (STAR). The automotive retail standards body whose data models the reference implementation builds on.

- [STAR unveils the Automotive Retail Domain Model](#). January 2026 announcement of STAR's unified data model for automotive retail.

Market Evidence

- [AI in Auto Retail Tracker](#) (Cox Automotive, 2026). Quarterly research on how car shoppers and dealers use AI, including the 63% of shoppers who expect to use AI on their next purchase.
- [AI traffic grows but retail sites lag in AI search visibility](#) (Adobe, 2026). AI-referred traffic to US retail sites grew 393% year over year in Q1 2026.
- [Americans and AI 2026: chatbots, smart devices and views on impact](#) (Pew Research Center, June 2026). About half of US adults now use AI chatbots.
- [CarMax launches car shopping and selling in the ChatGPT app store](#) (CarMax, February 2026). The first US auto retailer app in ChatGPT.
- [OpenAI's latest features take direct aim at the app store model](#) (TechCrunch, September 2026). How ChatGPT is becoming a place where software is discovered and used.
- [MCP 2026-07-28 spec: stateless core, coming to Claude](#) (Anthropic, July 2026). Claude's connector directory lists over 950 MCP servers used by millions of people every day.
- [BMW launches vehicle configurator as a dialogue-based plugin in OpenAI's ChatGPT](#) (BMW Group, July 2026). The first automaker app in ChatGPT.

Appendix A: Tool Reference

Table 1 lists every tool in the reference implementation, what it does, the sign-in scope it requires, and whether the assistant can call it or only a widget can.

Table 1. Tools in the reference implementation

Tool	What it does	Sign-in scope	Called by
get_system_capabilities	Lists what the app can and cannot do, so the assistant only claims supported actions	None	Assistant
search_products	Finds models that fit open-ended needs, such as a family SUV for road trips	None	Assistant
get_product_trims	Lists the trims available for a model	None	Assistant
compare_products	Compares vehicles side by side, including across makes	None	Assistant
search_inventory	Searches current dealer stock by model, price, body style, and location	None	Assistant
get_inventory_listing	Loads full details for one vehicle listing	None	Widget only
search_dealers	Finds dealers near a location	None	Assistant

Tool	What it does	Sign-in scope	Called by
get_customer_profile	Reads the signed-in customer's profile	profile	Assistant
get_customer_vehicles	Lists the customer's registered vehicles	vehicles	Assistant
get_vehicle_details	Reads details for one of the customer's vehicles	vehicles	Assistant
get_service_operations	Lists services and prices for a vehicle at a dealer	scheduling	Assistant
get_available_time_slots	Lists open appointment times at a dealer	scheduling	Assistant
get_appointments	Lists the customer's upcoming and past appointments	scheduling	Assistant
start_booking	Opens the booking wizard	scheduling	Assistant
start_update	Opens the wizard to reschedule or change an appointment	scheduling	Assistant
start_cancellation	Opens the wizard to cancel an appointment	scheduling	Assistant
get_booking_options	Loads advisors and transportation options for the wizard	scheduling	Widget only
book_appointment	Books the appointment the customer confirmed	scheduling	Widget only
update_appointment	Saves the changes the customer confirmed	scheduling	Widget only
cancel_appointment	Cancels the appointment the customer confirmed	scheduling	Widget only