AutoMotion MCP Server - Partner Integration Guide

June 17, 2026

Contents

AutoMotion MCP Server - Partner Integration Guide **2**
- Overview . . . 2
  - Capabilities . . . 2
- MCP Tools Reference . . . 2
  - Error Response Format . . . 2
  - 1. `list_brands` . . . 3
  - 2. `find_locations` . . . 4
  - 3. `get_booking_details` . . . 5
  - 4. `get_availability` . . . 6
  - 5. `book_appointment` . . . 7
- Connection Options . . . 8
  - Option A: MCP Stdio (Native Protocol) . . . 8
  - Option B: MCP over HTTP (StreamableHTTPServerTransport) . . . 9
- Integration Patterns . . . 10
  - Voice Assistant Integration . . . 10
  - Location Handling . . . 10
  - Location Filtering . . . 11
- Caching Behavior . . . 11
- Error Handling . . . 11
  - MCP Tool Error Response Format . . . 11
  - Booking Error Codes . . . 12
  - Degradation Behavior . . . 13
- Authentication . . . 13
  - Rate Limits . . . 14
- Interactive Maps Widget . . . 14
  - Retrieving the Widget . . . 14
  - Embedding the Widget . . . 15
  - Mapbox Token . . . 15
  - Browser Support . . . 15
- Testing . . . 16
  - Sandbox Environment . . . 16
  - MCP Inspector (Local Testing) . . . 16
- Support & Contact . . . 16
- Roadmap . . . 16
  - Sales Appointments . . . 16
- Appendix: Brand Name Usage . . . 16

AutoMotion MCP Server - Partner Integration Guide

Document Version: 1.6 **Last Updated**: June 17, 2026

Overview

AutoMotion provides a Model Context Protocol (MCP) server that enables conversational access to location search and service booking capabilities. This guide covers integration patterns for voice assistants, chatbots, and other AI-powered interfaces.

Sandbox connector URL (for evaluation):

`https://automotion-mcp-sandbox-439bba1a8b9e.herokuapp.com/mcp`

Point your MCP client at this host to exercise the full flow against disposable data. It is unauthenticated and exposes all five tools, including `get_availability` and `book_appointment`. The staging and production hosts are different URLs and do not expose the booking tools (`SCRAPER_BASE_URL` is unset there); see Testing for details. Use the path `/mcp` exactly as shown, with no trailing slash.

Capabilities

Capability	Description
Location Search	Find service locations by proximity, brand, or both
Brand Discovery	List available brands (car manufacturers + service providers)
Service Booking	Retrieve booking URLs and contact details for selected locations
Agentic Booking	Programmatically get appointment availability and confirm bookings
Interactive Maps	Embeddable Mapbox widget for visual location display

---

MCP Tools Reference

The server exposes up to 5 MCP tools. The three read-only tools — `list_brands`, `find_locations`, `get_booking_details` — are always available. The two scraper-backed tools — `get_availability` and `book_appointment` — register only when the deployment sets `SCRAPER_BASE_URL` (see `src/tools/index.ts`); they are present on sandbox but not on staging or production. Do not assume the booking tools exist in every deployment.

Error Response Format

All tools return errors in a consistent structure when validation fails, resources are not found, or internal errors occur. The object below is the tool **result**; over the wire it is returned inside the JSON-RPC `result` field of a `tools/call` response (see Authentication for the request envelope), not as a bare top-level HTTP body:

```
{
  "isError": true,
  "content": [
    {
      "type": "text",
      "text": "Brand \"Toyotta\" not found. Did you mean: Toyota, Honda, Chevrolet?"
    }
  ]
}
```

Common error scenarios:

Error Type	Example Message	Resolution
Validation Error	"Both latitude and longitude are required for proximity search. You provided only one."	Ensure both coordinates are supplied or omit both
Brand Not Found	"Brand "Toyotta" not found. Did you mean: Toyota, Honda…"	Retry with suggested brand name from error message or call `list_brands`
Invalid Coordinates	"Latitude must be between -90 and 90"	Validate coordinate ranges before calling
Resource Not Found	"Location with account_id 'abc-123' not found"	Verify account_id from `find_locations` results
Booking Not Available	"BOOKING NOT AVAILABLE: {Name} does not have online scheduling configured."	Use `get_booking_details` for contact information; online booking is not available at this location
Booking Failed	"BOOKING FAILED [`scheduling_failed`]: {detail}"	Retry after a short delay; check the error code for specific guidance; suggest a nearby alternative if retries fail

All error messages are in `content[0].text`. They are written for the calling agent and often carry workflow guidance (for example a `VALIDATION ERROR:` prefix or parameter hints), so paraphrase them for end users rather than displaying the raw text. The `isError` field is `true` for errors, `false` or omitted for successful responses.

---

1. `list_brands`

Return the complete list of brands in the network, categorized as car manufacturers (Honda, Toyota, Chevrolet, …) and service providers (Glass Repair, Rent a Car, Tire and Wheel, …).

Input Schema:

```
{
  "type": "object",
  "properties": {}
}
```

No inputs. Call this for discovery when the user asks "what brands do you have?", or before a location search where the valid brand vocabulary is unknown.

Response (abridged):


```
{
  "summary": "Found 18 total: 12 car manufacturers + 6 service providers",
  "car_manufacturers": [
    {
      "id": "a7f3b2c1-8e9d-4f5a-b1c2-d3e4f5a6b7c8",
      "name": "Honda",
      "logo": "https://assets.automotion.tv/honda.png",
      "category": "manufacturer"
    }
  ],
  "service_providers": [
    {
      "id": "b8e4c3d2-9f0e-5a6b-c2d3-e4f5a6b7c8d9",
      "name": "Tire and Wheel",
      "logo": null,
      "category": "service"
    }
  ]
}
```


---

2. `find_locations`

Primary search tool, finds service locations by proximity, by brand, or by both. Unifies what were previously three separate search tools.

Input Schema:

```
{
  "type": "object",
  "properties": {
    "brand_name": {
      "type": "string",
      "description": "Human-readable brand name (e.g., 'Honda', 'Tire and Wheel').
      ↪  Case-insensitive. Server resolves to internal brand IDs."
    },
    "latitude": {
      "type": "number",
      "minimum": -90,
      "maximum": 90,
      "description": "Search center latitude."
    },
    "longitude": {
      "type": "number",
      "minimum": -180,
      "maximum": 180,
      "description": "Search center longitude."
    }
  },
  "additionalProperties": false
}
```

All fields are optional, but at least one of `brand_name` or `(latitude, longitude)` must be provided. `latitude` and `longitude` must be supplied as a pair; providing only one coordinate returns an

error response (`isError: true`). Search behavior is determined by which fields are present:

Inputs	Behavior
`brand_name` + `latitude` + `longitude`	Up to 20 nearest locations of that brand (most common)
`latitude` + `longitude` only	Up to 20 nearest locations across all brands
`brand_name` only	All locations for that brand nationwide (rarely needed)

Response:

Markdown-formatted location list. Each rendered entry shows the location name, address, and a Schedule Service booking link (or a "No booking URL available" note). Business category and services are not in the markdown; they are available in `structuredContent.locations`. Distance is not currently returned in either surface. Each location exposes an `account_id` (UUID). Pass it to `get_booking_details` or `get_availability`. `book_appointment` does not take `account_id`; it takes the `session_id` returned by `get_availability`.

---

3. `get_booking_details`

Retrieve booking URL and contact information for a specific location returned by `find_locations`.

Input Schema:

```
{
  "type": "object",
  "properties": {
    "account_id": {
      "type": "string",
      "format": "uuid",
      "description": "Location account UUID from find_locations results."
    }
  },
  "required": ["account_id"],
  "additionalProperties": false
}
```

Response:

```
{
  "name": "Honda of Austin",
  "booking_url": "https://service.honda-of-austin.com/schedule",
  "contact": {
    "email": "service@example.com"
  },
  "address": {
    "street": "123 Main St",
    "city": "Austin",
    "state": "TX",
    "zip": "78701",
    "country": "US"
  },
  "brands": [
```

```
    { "id": "a7f3b2c1-8e9d-4f5a-b1c2-d3e4f5a6b7c8", "name": "Honda" }
  ],
  "products": [
    { "name": "Civic", "brand_id": "a7f3b2c1-8e9d-4f5a-b1c2-d3e4f5a6b7c8", "year": 2024 }
  ]
}
```

`booking_url` is the location's `schedule_service_url` (from the account's navigation menu). It may point at a dealer or third-party scheduler rather than an AutoMotion-hosted form, and it may be `null` if online booking is not configured at that location. Phone numbers are intentionally omitted; users should complete booking through the URL.

This tool returns a contact/booking-URL summary plus a second `content` block containing the full location data as JSON (`Full Data:\n{...}`). Appointment availability is handled by `get_availability`.

---

4. `get_availability`

Return available appointment slots for a location. Call this after `find_locations` to begin the booking flow.

Input Schema:

```
{
  "type": "object",
  "properties": {
    "account_id": {
      "type": "string",
      "format": "uuid",
      "description": "Location account UUID from find_locations results."
    },
    "vehicle": {
      "type": "object",
      "properties": {
        "make":  { "type": "string", "description": "Vehicle make (e.g., 'Honda')." },
        "model": { "type": "string", "description": "Vehicle model (e.g., 'Civic')." },
        "year":  { "type": "integer", "description": "Model year (e.g., 2022)." }
      },
      "required": ["make", "model", "year"]
    }
  },
  "required": ["account_id"],
  "additionalProperties": false
}
```

Field	Required	Description
`account_id`	Yes	Location account UUID from `find_locations` results
`vehicle`	No	Vehicle info object — used to filter relevant services when provided
`vehicle.make`	Yes (if `vehicle` provided)	Vehicle make
`vehicle.model`	Yes (if `vehicle` provided)	Vehicle model
`vehicle.year`	Yes (if `vehicle` provided)	Model year

Response:

The tool returns a formatted text summary of available slots. Slot dates and times are in the **location's local timezone**; display them with a note ("All times shown in location local time").

Errors are returned with `isError: true`; the message is in `content[0].text`.

---

5. `book_appointment`

Confirm an appointment at a location. Call this after `get_availability` once the user has selected a slot.

Input Schema:

```
{
  "type": "object",
  "properties": {
    "session_id": {
      "type": "string",
      "description": "Session ID returned by get_availability. Required to complete the
      ↪  booking."
    },
    "customer": {
      "type": "object",
      "properties": {
        "first_name": { "type": "string" },
        "last_name":  { "type": "string" },
        "email":      { "type": "string", "format": "email" },
        "phone":      { "type": "string", "description": "10-digit US phone number,
        ↪  digits only." }
      },
      "required": ["first_name", "last_name", "email", "phone"]
    },
    "vehicle": {
      "type": "object",
      "properties": {
        "make":    { "type": "string" },
        "model":   { "type": "string" },
        "year":    { "type": "integer" },
        "mileage": { "type": "integer", "description": "Odometer reading. Optional and
        ↪  currently unused by book_appointment (not sent to the scheduler or shown in
        ↪  the confirmation)." }
      },
      "required": ["make", "model", "year"],
      "description": "Vehicle info. Collect from the user if available."
    },
    "appointment": {
      "type": "object",
      "properties": {
        "date": { "type": "string", "description": "Appointment date (YYYY-MM-DD) from
        ↪  get_availability results." },
        "time": { "type": "string", "description": "Appointment time (HH:MM, 24-hour)
        ↪  from get_availability results." },
        "service_type": { "type": "string", "description": "Service type from the
        ↪  selected slot, if one was shown." }
```

```
      },
      "required": ["date", "time"]
    }
  },
  "required": ["session_id", "customer", "appointment"],
  "additionalProperties": false
}
```

Field	Required	Description
`session_id`	Yes	Session ID returned by `get_availability`
`customer.first_name`	Yes	
`customer.last_name`	Yes	
`customer.email`	Yes	
`customer.phone`	Yes	10-digit US phone number, digits only
`vehicle`	No	Vehicle info object. Used only for the confirmation summary; not sent to the scheduler (vehicle reaches the scheduler at `get_availability`). Collect from the user if available.
`vehicle.make`	Yes (if vehicle provided)	
`vehicle.model`	Yes (if vehicle provided)	
`vehicle.year`	Yes (if vehicle provided)	
`vehicle.mileage`	No	Odometer reading. Optional and unused by `book_appointment` (not sent to the scheduler; no default applied).
`appointment.date`	Yes	Must match a date from `get_availability` results
`appointment.time`	Yes	Must match a time from `get_availability` results
`appointment.service_type`	No	Service type from the selected slot, if provided

Response:

The tool returns a formatted text confirmation including the confirmation number, customer name, vehicle, and appointment date/time. Errors are returned with `isError: true`; the message is in `content[0].text`.

Connection Options

Option A: MCP Stdio (Native Protocol)

Standard MCP transport for direct integration with MCP-compatible clients.

```
npm run dev
```

Transport: Standard input/output streams **Protocol**: JSON-RPC 2.0 over MCP **Use Case**: Native MCP clients, local development

Option B: MCP over HTTP (StreamableHTTPServerTransport)

HTTP transport for web-based integrations and cloud-deployed clients (ChatGPT, web MCP clients).

```
npm run http
```

Base URL: `http://localhost:8081` (or deployed endpoint) **Protocol**: JSON-RPC 2.0 over HTTP with session tracking **Use Case**: Voice assistant backends, web services, ChatGPT connectors

HTTP Endpoints

Method	Endpoint	Description
`POST`	`/mcp`	MCP JSON-RPC requests (initialize, tools/list, tools/call)
`GET`	`/mcp`	SSE stream for server-to-client notifications
`DELETE`	`/mcp`	Terminate an MCP session
`GET`	`/`	Health check
`GET`	`/capabilities`	Debug: server name, version, and the registered tools and resources. Capability flags are emitted authoritatively by `initialize` over `/mcp`, not by this endpoint.
`GET`	`/cache-stats`	Cache statistics

HTTP Tool Invocation (JSON-RPC) MCP over HTTP uses the JSON-RPC 2.0 protocol with session tracking. A client must (1) initialize a session, (2) capture the `mcp-session-id` response header, and (3) include it on subsequent calls. The `Accept` header **must** include `text/event-stream`, or the server responds `406 Not Acceptable`.

Step 1. Initialize:

```
curl -i -X POST http://localhost:8081/mcp \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -d '{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
      "protocolVersion": "2024-11-05",
      "capabilities": {},
      "clientInfo": { "name": "partner-integration", "version": "1.0.0" }
    }
  }'
```

Capture the `mcp-session-id` response header.

Step 2. Call a tool:

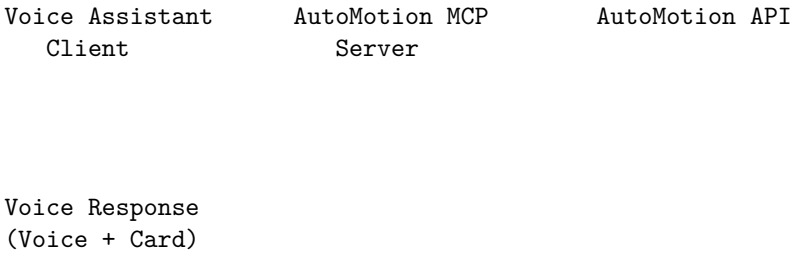
```
curl -X POST http://localhost:8081/mcp \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -H "mcp-session-id: <session-id-from-step-1>" \
```

```
-d '{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/call",
  "params": {
    "name": "find_locations",
    "arguments": {
      "brand_name": "Honda",
      "latitude": 30.2672,
      "longitude": -97.7431
    }
  }
}'
```

Integration Patterns

Voice Assistant Integration

Recommended flow for voice-initiated searches:



Sample Intent Mapping:

User Intent	MCP Tool	Required Arguments
"Find nearby Honda locations"	`find_locations`	`brand_name`, `latitude`, `longitude`
"Where can I get an oil change?"	`find_locations`	`latitude`, `longitude`
"What brands do you support?"	`list_brands`	(none)
"Get booking details for [location]"	`get_booking_details`	`account_id`
"What appointment slots are available?"	`get_availability`	`account_id`, `vehicle` (optional)
"Book the 9am slot on June 15"	`book_appointment`	`session_id`, `appointment`, `customer`, `vehicle`

Location Handling

The MCP server expects latitude/longitude coordinates. For voice assistants:

1. Request device location permission

2. Convert device location to coordinates
3. Pass coordinates to `find_locations`

Location Filtering

The AutoMotion API filters locations by `partner_category` before returning results. The MCP server forwards requests to the upstream locations endpoint; filtering is applied there, not in the MCP layer.

All searches (brand + proximity, brand-only):

`partner_category`	Business type
`transportation.vehicle_dealer.car_dealership`	Car dealership
`travel_and_leisure.vehicle_rental`	Vehicle rental
`travel_and_leisure.vehicle_rental.car_rental`	Car rental
`transportation.auto_body_shop.auto_glass_shop`	Auto glass shop
`transportation.vehicle_repair_service.auto_repair_shop`	Auto repair shop
`transportation.vehicle_parts_store.tire_shop`	Tire shop

Proximity-only searches (no `brand_name`) return a narrower subset — the AutoMotion API excludes rental and glass repair locations because they do not offer general automotive services such as oil changes:

`partner_category`	Business type
`transportation.vehicle_dealer.car_dealership`	Car dealership
`transportation.vehicle_repair_service.auto_repair_shop`	Auto repair shop
`transportation.vehicle_parts_store.tire_shop`	Tire shop

To surface rental or glass repair locations, pass the corresponding `brand_name` (`"Rent a Car"` or `"Glass Repair"`).

Caching Behavior

Cache Type	TTL	Description
Account data	5 minutes	Brand lists, account configuration
Location data	1 hour	Service location data

Cache Stats Endpoint:

```
curl http://localhost:8081/cache-stats
```

Error Handling

MCP Tool Error Response Format

When a tool call fails, the MCP server returns `isError: true`. The error message is in `content[0].text` and describes what went wrong and what to try next.

Booking Error Codes

When `get_availability` or `book_appointment` encounters an error from the booking layer, the `error_code` is included in the MCP error message. Use it to determine the appropriate client behavior.

Several error codes include a `Suggested response:` paragraph in the error message text with inline agent guidance: `wizard_step_failed`, `browser_transport`, `session_not_found`, `session_gone`, `booking_not_supported`, `captcha_timeout`, and `needs_fallback`.

error_code	Description	Recommended Action
`wizard_step_failed`	Scheduling system integration issue at this dealer's platform	Offer to find a nearby alternative or call `get_booking_details` for the dealer's direct booking URL
`browser_transport`	Transient browser infrastructure failure	Retry once automatically; if still failing, offer a nearby alternative dealer
`submission_rejected`	Scheduling system rejected the booking; error message contains provider-supplied detail	Paraphrase reason to user; do not retry automatically
`dealer_cooling_off`	This dealer's booking service is temporarily suppressed due to captcha rate-limiting	Retry after the indicated `retry_after` seconds; suggest a nearby alternative
`captcha_timeout`	Booking could not complete because the scheduling system requires human verification	Offer to find a nearby alternative or call `get_booking_details` for the dealer's direct booking URL
`slot_not_found`	Requested date/time does not match an available slot	Call `get_availability` again for a fresh slot list
`vehicle_not_supported`	This location does not support the requested make/model	Inform user; suggest contacting the location directly
`vehicle_ambiguous`	Multiple vehicle options match; request is ambiguous	Ask user to clarify make/model/year
`pool_full`	Booking service at capacity	Retry once after 30 seconds; if still full, inform the user to try again shortly
`session_not_found`	Session ID is unknown or was never created	Call `get_availability` again to start a new session
`session_gone`	Session expired mid-flow	Call `get_availability` again to start a new session
`booking_not_supported`	Online booking is not available for this dealer	Call `get_booking_details` to surface the dealer's direct booking URL
`needs_fallback`	Automatic online booking unavailable for this dealer (no supported booking platform detected)	Call `get_booking_details` to surface the dealer's direct booking URL
`client_cancelled`	Client disconnected mid-request	Retry if user still wants to book
`internal_error`	Unexpected error	Contact AutoMotion support

> **Note on `scheduling_failed`:** This string appears in error messages when the scraper client throws an error with no `error_code` field. It is a client-side MCP fallback, not a booking-layer classifier code. Retry once; if still failing, call `get_booking_details` to surface the dealer's direct booking URL.

Degradation Behavior

Some locations periodically enter a **cool-off period** during which `get_availability` returns `dealer_cooling_off` immediately. The error includes a `retry_after` field indicating how many seconds to wait.

Recommended client behavior:

1. On `dealer_cooling_off`: surface to the user that scheduling is temporarily unavailable at this location, suggest trying a nearby alternative, and do not retry until the indicated `retry_after` seconds have elapsed.
2. On `pool_full`: retry once after 30 seconds; if still full, inform the user to try again shortly.
3. On `wizard_step_failed`, `browser_transport`, `captcha_timeout`, `session_not_found`, `session_gone`, `booking_not_supported`, or `needs_fallback`: the inline `Suggested response:` paragraph in the error message provides specific guidance. Follow it.

Authentication

Production and staging partner deployments require an API key issued per partner. The sandbox environment (see Sandbox Environment below) is unauthenticated for evaluation — no API key is required there. Confirm the requirement for your target endpoint.

HTTP Header:

```
Authorization: Bearer <your-api-key>
```

Example:

```
curl -X POST https://<mcp-endpoint>/mcp \
  -H "Content-Type: application/json" \
  -H "Accept: application/json, text/event-stream" \
  -H "Authorization: Bearer <your-api-key>" \
  -H "mcp-session-id: <your-session-id>" \
  -d '{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "tools/call",
    "params": {
      "name": "find_locations",
      "arguments": {
        "latitude": 30.2672,
        "longitude": -97.7431
      }
    }
  }'
```

MCP server API keys are provisioned on a per-partner basis. Contact AutoMotion to request credentials and confirm your production endpoint URL.

Rate Limits

Rate limits are enforced per partner API key:

Tier	Requests/minute	Requests/hour	Burst Allowance
Development	30	500	5
Production	300	10,000	20
Enterprise	Custom	Custom	Custom

HTTP Response Headers:

```
X-RateLimit-Limit: 300
X-RateLimit-Remaining: 287
X-RateLimit-Reset: 1716758400
```

`X-RateLimit-Reset` is a Unix timestamp (seconds since epoch) indicating when the rate limit window resets.

429 Too Many Requests Response:

Rate limiting is enforced at the HTTP layer (gateway), not by the MCP tool calls. A throttled request returns an HTTP `429` status with a `Retry-After` header rather than an MCP `tools/call` result envelope:

```
HTTP/1.1 429 Too Many Requests
Retry-After: 42
```

Retry Strategy:

- Respect the `Retry-After` header on 429 responses and wait that many seconds before retrying
- For `pool_full` booking errors: retry once after 30 seconds; if still full, advise the user to try again shortly (consistent with Booking Error Codes and Degradation Behavior above)
- Implement exponential backoff for transient errors (503, 504)

Development tier is suitable for integration testing and staging environments. Contact AutoMotion to upgrade to Production or Enterprise tier.

Interactive Maps Widget

The MCP server exposes a Mapbox-powered widget for visual location display. The widget is a self-contained HTML document (with its JavaScript inlined) that can be embedded in partner applications, web interfaces, or chat UI overlays.

Retrieving the Widget

The widget is available as an MCP resource, retrieved via the MCP `resources/read` method only. There is no HTTP endpoint for it.

MCP Resource URI: `ui://widget/dealer-map.html`

Example MCP Request:

```
{
  "jsonrpc": "2.0",
  "id": 1,
```

```
  "method": "resources/read",
  "params": {
    "uri": "ui://widget/dealer-map.html"
  }
}
```

Response:

```
{
  "contents": [
    {
      "uri": "ui://widget/dealer-map.html",
      "mimeType": "text/html;profile=mcp-app",
      "text": "<!doctype html>..."
    }
  ]
}
```

Embedding the Widget

Render the widget by embedding the resource text in a sandboxed iframe:

```
<iframe
  srcdoc="<html content from resources/read response text>"
  sandbox="allow-scripts allow-same-origin"
  style="width: 100%; height: 500px; border: none;"
></iframe>
```

Mapbox Token

Partners must provision their own Mapbox access token:

1. Create a free account at mapbox.com
2. Generate a token with `styles:read` and `fonts:read` scopes
3. For production use, restrict the token to the origin(s) where the widget actually renders. When it is embedded in an AI client (ChatGPT, Claude), Mapbox requests originate from that client's iframe origin, not your domain, so a token URL-restricted to `*.yourcompany.com` will fail. Match the restriction to the real embed origin(s), or leave it unrestricted when the origin is unknown.

AutoMotion does not provide Mapbox tokens. The widget bundles Mapbox GL JS inline, but loads the Mapbox GL CSS and map tiles from the Mapbox CDN at runtime, so an active internet connection is required to render the map.

Browser Support

- Chrome/Edge 90+
- Firefox 88+
- Safari 14+
- Modern mobile browsers (iOS Safari 14+, Chrome Android 90+)

Testing

Sandbox Environment

A shared sandbox environment is available for integration testing:

MCP Endpoint: `https://automotion-mcp-sandbox-439bba1a8b9e.herokuapp.com/mcp` (the base host is `https://automotion-mcp-sandbox-439bba1a8b9e.herokuapp.com`, with no trailing slash, so appending `/mcp` does not produce a double slash)

The sandbox contains a set of dealer locations clustered around Las Vegas, NV — ask the agent to find locations near Las Vegas to get results. Availability and booking flows return synthetic data: no real appointments are created and no confirmation emails are sent. Note that a successful `book_appointment` still returns the standard confirmation text ("A confirmation email has been sent…"); in the sandbox that line is part of the synthetic response and no email is actually delivered. No API key is required; connect using the standard MCP over HTTP flow.

MCP Inspector (Local Testing)

```
npm run inspect
```

Opens interactive tool testing interface at `http://localhost:5173`.

Support & Contact

Technical support contacts and status page URL are provided during partner onboarding.

Roadmap

The following capabilities are planned for a future release.

Sales Appointments

We are evaluating support for new vehicle sales appointments in addition to service bookings. The goal is to allow AI assistants to help customers schedule a vehicle purchase consultation or test drive at a location. We will share a proposal with technical partners before implementation begins.

Appendix: Brand Name Usage

`find_locations` accepts `brand_name` as a human-readable string; the server resolves the name to an internal brand ID. Matching is case-insensitive and lenient: prefix and substring matches resolve, so `"Hond"` returns Honda. If no match can be found (including misspellings the lenient matcher can't resolve), the tool returns an error response (`isError: true`) with a list of suggested brand names so clients can implement a retry/fallback flow.

Examples:

Intent	`brand_name` value
Find Honda dealers	`"Honda"`
Find any tire shop	`"Tire and Wheel"`
Find a glass repair shop	`"Glass Repair"`
Find a rental location	`"Rent a Car"`

Call `list_brands` for the canonical set of names.