FHIR Integration

What the AI can do

Summary

FHIR Integration connects any **HL7 FHIR R4**-compliant scheduling/EHR system (reference implementation: **Aidbox**) with the **Newo Platform**.

It enables:

- Checking available appointment slots for a pre-selected provider and facility
- Booking appointments with automatic patient lookup and creation
- Cancelling existing appointments
- Pre-configured facility (`Organization`) and provider (`PractitionerRole`) selection during setup
- OAuth2 token management with `authorization_code` or `client_credentials` grant and automatic refresh

This integration is designed for **clinics, medical practices, and any scheduling-capable EHR that exposes a FHIR R4 REST API** who need **automated appointment scheduling through a conversational AI agent** (voice or chat).

Because the integration speaks standard FHIR REST, it is **vendor-agnostic**: the same skills work against Aidbox, Epic FHIR, Cerner/Oracle Health, Azure FHIR, Google Cloud Healthcare API, HAPI FHIR, or any other conformant server — only the base URL and OAuth endpoints change.

Common use cases

- When a patient asks about available slots → AI queries `Schedule` for the selected provider and then `Slot` for free time windows
- When a patient wants to book an appointment → Search `Patient` by phone, create one if not found, then create an `Appointment` resource
- When a patient wants to cancel an appointment → Patch the existing `Appointment` resource with `status = cancelled`
- When a new patient books → Automatically create a `Patient` record in FHIR before the appointment is written
- When the project is published → Facilities (`Organization`) and providers (`PractitionerRole`) are fetched and exposed as enum dropdowns in the Builder

Features at a glance

Feature	Supported	Notes
Check Availability	Yes	Date range lookup via `Schedule` + `Slot?status=free`
Book Appointment	Yes	With automatic `Patient` creation if no record matches the caller
Cancel Appointment	Yes	the update request with `status = cancelled`
Facility Selection	Yes	Pre-configured during setup via `Organization?active=true&type=prov`

Provider Selection	Yes	Pre-configured during setup, filtered by selected facility
Patient Search	Yes	the lookup request
Patient Creation	Yes	Auto-creates during BookingFlow if no match is found
Token Auto-Refresh	Yes	OAuth2 `refresh_token` with fallback to `client_credentials`
OAuth Authorization Code Flow	Yes	One-time interactive login via hosted redirect page
Client Credentials Grant	Yes	Headless alternative for system-level integrations
Multi-Provider Scheduling	No	A single provider is selected during setup
Reschedule Appointment	No	Cancel + rebook as workaround
Multiple Locations	No	One `Organization` is selected per project
Historical Data Import	No	Only events created after activation are handled

Before You Start

Before installation:

- Access to a **FHIR R4**-compliant server that supports the following resources: `Organization`, `PractitionerRole`, `Patient`, `Schedule`, `Slot`, `Appointment`
- **OAuth2 Client ID** and **Client Secret** registered with the FHIR server
- The **Base URL** of the FHIR server (without the trailing `/fhir` segment)
- SMART-on-FHIR scopes granted on the server side:
  - `system/Patient.read` / `system/Patient.write`
  - `system/Organization.read`
  - `system/PractitionerRole.read`
  - `system/Schedule.read`
  - `system/Slot.read`
  - `system/Appointment.write`
- The FHIR server must be reachable over HTTPS from the Newo Platform

Setup

Create the Newo project

If this project already exists in **Newo Builder**, skip this subsection and continue with the integration-specific settings below.

1. In **Newo Builder**, open the projects list and click **Create Project** (or **Create New Project** from the top-right menu).

 Create New Project menu in Newo Builder

2. Fill **IDN** and **Title**. The exact names do not matter; use any clear names your team will recognize.
3. In **Registry**, choose the release channel:

- **staging** — the newest module fixes appear here first. Use it when you need the latest fix, but expect possible unfinished changes.
- **production** — the final stable version for live projects.

4. In **Module**, select the module for this integration.

 Create Project form showing IDN, Title, Registry, and Module fields 5. Leave **Module version** on **Latest version** unless support tells you to pin a specific version, then click **Create**.

3.1 Step 1 — Register an OAuth2 Client

1. Log in to your FHIR server's admin console (Aidbox, Azure Health Data Services, Epic App Orchard, etc.)
2. Register a new OAuth2 client application
3. Add the Newo redirect URL: `https://static.newo.ai/oauth/index.html`
4. Request the SMART scopes listed in section 2
5. Note the **Client ID** and **Client Secret**

3.2 Step 2 — Connect in Platform

1. Open **Newo → Projects**
2. Set the following attributes in **Builder / Attributes**:

Attribute	Required	Description
`fhir_base_url`	Yes	FHIR server base URL (e.g., `https://fhir.example.com`). The integration appends `/fhir/...` for resource calls and `/auth/token` for OAuth
`fhir_client_id`	Yes	OAuth2 Client ID
`fhir_client_secret`	Yes	OAuth2 Client Secret
`fhir_grant_type`	Yes	`authorization_code` (interactive) or `client_credentials` (headless)
`fhir_oauth_code`	Yes*	One-time authorization code; required only when `fhir_grant_type = authorization_code`. Obtain via the hosted login link shown in the attribute description
`fhir_scope`	Yes	OAuth2 scope list (see section 2)
`fhir_facility`	No	Facility (`Organization`) — auto-populated during setup as an enum
`fhir_provider`	No	Provider (`PractitionerRole`) — auto-populated after facility is selected
`fhir_enable_slot_check`	No	Enable availability checks (default: `True`)
`fhir_enable_booking`	No	Enable appointment creation (default: `True`)
`fhir_enable_cancellation`	No	Enable cancellations (default: `True`)

`fhir_check_existing_client`	No	Search `Patient` by phone before creating a new record (default: `True`)
`fhir_show_for_days_customer`	No	Number of days ahead to search for available slots (default: `5`)
`fhir_use_timezone_from_conversation`	No	If `True`, detect caller time zone from conversation; if `False`, use business TZ
`fhir_override_agent_attributes`	No	Override SuperAgent attribute schemas on publish (default: `True`)
`fhir_access_token`	No	Populated automatically after successful OAuth exchange
`fhir_refresh_token`	No	Populated automatically after successful OAuth exchange

3. If `fhir_grant_type = authorization_code`:
    1. Open the **FHIR OAuth Code** (`fhir_oauth_code`) attribute — its description contains a clickable hosted login link
    2. Log in to the FHIR server, confirm the requested scopes
    3. After redirect, copy the returned code and paste it into the attribute
4. Click **Save + Publish All**
5. On publish, the **SetupFlow** automatically:
    - Exchanges the credentials for an `access_token` + `refresh_token` via the create/update request
    - Creates `fhir_connector` (resource calls) and `fhir_token_connector` (token calls)
    - Fetches all facilities with the lookup request and populates the facility selector
    - Fetches providers for the selected facility with the lookup request and populates the provider selector
    - Registers availability, booking, and cancellation tools with the ConvoAgent
    - Injects payload schemas for the ConvoAgent tool-calling framework

How to use the integration

This section explains how the integration works, how to configure automation, and how to test it.

4.1 How the Integration Works

The integration uses the standard FHIR REST API for scheduling and demographic data. Unlike multi-provider integrations, the facility and provider are pre-selected during setup, which keeps runtime queries simple and deterministic. The access token is automatically refreshed on `401` responses using the `refresh_token` grant, with a fallback to `client_credentials` when no refresh token is available.

- A **Trigger** is a system event that starts a flow
- An **Action** is the FHIR REST operation executed against the server

FHIR Resources Used

Resource	Access	Purpose
`Organization`	read	Facilities / clinics available for booking

`PractitionerRole`	read	Providers linked to each organization
`Schedule`	read	Schedule envelope for a provider within a date range
`Slot`	read	Free time windows within a `Schedule`
`Patient`	read + write	Look up caller by phone, create a record if not found
`Appointment`	write	Create and cancel appointments

Resources that are **not** required (and therefore may remain disabled on the server): `RelatedPerson`, standalone `Practitioner`, `Coverage`, `AppointmentResponse`, `Location`, `HealthcareService`, `Encounter`.

Where to test

Testing can be done through the **Newo conversation interface** (voice or chat). Each flow (Availability, Booking, Cancellation) has a dedicated test skill (`CheckAvailabilityTestSkill`, `CreateAppointmentTestSkill`, `CancelAppointmentTestSkill`) that can be triggered directly for connectivity validation without running the full conversational flow.

How to test that everything works

To test the integration:

1. **Setup**: Publish the project and verify the OAuth token exchange succeeds (check `fhir_access_token` is populated after publish)
2. **Facilities**: After publish, verify `fhir_facility` enum is populated with your `Organization` entries
3. **Providers**: Select a facility and re-publish — verify `fhir_provider` enum shows the `PractitionerRole` entries for that facility
4. **Availability**: Ask "What slots are available on Monday?" — verify slots are returned with correct times
5. **Booking**: Complete a booking conversation — verify the `Appointment` resource appears on the FHIR server and is linked to a valid `Patient` reference
6. **Cancellation**: Request cancellation of a booked appointment — verify the `Appointment.status` changes to `cancelled` on the server

If no action occurs:

- Ensure `fhir_base_url`, `fhir_client_id`, and `fhir_client_secret` are set correctly
- Verify `fhir_grant_type` matches what the server actually supports; if `authorization_code`, make sure `fhir_oauth_code` is populated and still valid (authorization codes are short-lived)
- Confirm `fhir_access_token` was obtained after setup (it should not be empty)
- Ensure the relevant feature flags are enabled: `fhir_enable_booking`, `fhir_enable_slot_check`, `fhir_enable_cancellation`
- Verify `fhir_facility` and `fhir_provider` were populated during setup — both attributes are stored in the format `id|display_name` and split on `|`
- Check that the FHIR server is reachable over HTTPS from the platform and that the configured scopes include all required resources
- Confirm the server supports the exact search parameters used by the integration (`Organization?active=true&type=prov`, `PractitionerRole?organization:identifier=…`, `Schedule?actor:identifier=…`, `Slot?schedule:identifier=…&status=free`)

- Re-publish the project if any credentials were changed

> *Note: This integration performs real operations against the configured FHIR server. Appointments created or cancelled through the agent are reflected in the live EHR.*

FAQ

Q: Which FHIR version is supported? A: FHIR **R4**. The integration uses R4 search parameter names (`actor:identifier` , `schedule:identifier`) and JSON Patch for status updates. Earlier versions (DSTU2, STU3) are not supported.

Q: Which FHIR servers have been tested? A: The reference implementation targets **Aidbox**, but the integration only uses standard FHIR REST and SMART-on-FHIR OAuth, so any R4-compliant server should work provided it supports the search parameters listed above. The expected token endpoint is the create/update request — if your server exposes a different path (e.g., `/oauth2/token`), the `fhir_base_url` or the skill's token URL must be adjusted.

Q: How does facility and provider selection work? A: On publish, the SetupFlow fetches all `Organization` entries (filtered by `active=true&type=prov`) and populates the `fhir_facility` enum. After you pick a facility and re-publish, it fetches the `PractitionerRole` entries linked to that organization and populates the `fhir_provider` enum. Both attributes are stored as `"{id}|{display_name}"` strings.

Q: What patient information is required for booking? A: First name, last name, email, and phone number are required (validated in `_createContactSkill`). If any are missing the booking flow returns an error before touching the FHIR server.

Q: How is patient deduplication handled? A: The BookingFlow searches `Patient` by the caller's phone number via the lookup request. If a match is found, the existing `Patient.id` is reused; otherwise a new `Patient` is created. If you need deduplication on other identifiers (email, MRN, SSN), the `_getContactSkill` would need to be extended.

Q: What OAuth2 grants are supported? A: Both `authorization_code` (interactive one-time login through `https://static.newo.ai/oauth/index.html`) and `client_credentials` (headless, system-level). The grant type is selected via the `fhir_grant_type` attribute. Token refresh uses `refresh_token` when available, with fallback to `client_credentials` .

Q: What happens if the OAuth token expires? A: The integration refreshes credentials automatically when the server says the token is no longer valid. If refresh is not possible, paste fresh credentials in Newo Builder and click **Publish All**.

Q: How many days of availability does the agent check? A: Configurable via the `fhir_show_for_days_customer` attribute (default: 5). The `Schedule` and `Slot` queries use `date=ge{from}` and `date=le{to}` where `to = from + show_for_days` .

Q: What is the default appointment duration? A: 1 hour by default. If your clinic needs a different duration, update the appointment-duration setting in Newo Builder or ask support to adjust the module configuration.

Q: How is the practitioner identified in the created Appointment? A: The integration writes the provider into `Appointment.participant[].actor` using an NPI identifier (`system = http://hl7.org/fhir/sid/us-npi`) derived from the selected `fhir_provider` attribute. If your server uses a different identifier system for providers, ask support to adjust the provider mapping.

Q: Why does cancellation update the appointment instead of deleting it? A: FHIR best practice is to preserve the appointment record and mark its status as cancelled. This keeps audit history intact and matches the behavior expected by most EHRs.

Q: Does the integration support rescheduling? A: Not as a single action. Reschedule is currently handled as cancel + rebook. Adding a true the update request with new `start` / `end` would require a new flow.

Q: Does the integration support multiple facilities simultaneously? A: No. A single facility and provider are selected during setup. To switch facilities, update the `fhir_facility` attribute and re-publish to refresh the provider list.

Q: What minimum SMART scopes does the integration actually need? A:

```
system/Patient.read  system/Patient.write
system/Organization.read
system/PractitionerRole.read
system/Schedule.read
system/Slot.read
system/Appointment.write
```

These are the minimum scopes needed to cover *retrieve customer details* and *schedule appointments*. Enabling additional resources (`Coverage` , `Location` , `RelatedPerson` , etc.) is not required for the current feature set.