indition™

INDITION CHAT

Setup Guide

Install Chat from zero: module enablement, migrations, access sync, the frontend, real-time delivery, and taking your first site live.

Chat Module · Version 1.0 · June 2026

Indition Chat — Setup Guide

> An installation guide for administrators standing up Chat from zero: enabling the module, running the database migrations, syncing access, deploying the frontend, bringing up real-time delivery, and taking your first site live.

This guide is for platform administrators and deployers. For day-to-day use once Chat is live, see the **User Manual**; for architecture and internals, see the **Technical Reference**.

Before you start

- Administrator access to the target site/instance and its databases (owner, config, instance).
- The ability to deploy the paired React frontend and import route/menu/access metadata.
- A domain where the public widget endpoints are reachable over HTTPS.
- (Optional, for live chat) a real-time gateway host with Redis and Node.js for the WebSocket server.

Install at a glance

1. Enable the Chat module for the site.
2. Run the package migrations and apply the seed SQL.
3. Sync OAuth access, clear caches, and re-login.
4. Deploy the frontend and import route/menu/access metadata.
5. (Optional) Bring up real-time delivery.
6. Create your first site, install the widget, assign agents, and go live.
7. Configure email delivery, notifications, and the auto-close cron.

1. Install the module

Enable Chat

Add the module to the enabled-module list for the target site, following your platform's standard module-enablement process.

Run the database migrations

Run the Chat package migrations in order — they create the runtime tables (all tenant-prefixed `{SITE_ID}_chat_*`):

1. `Patch_CreateChatTables` — core conversation, message, and settings tables.
2. `Patch_AddChatMultiSiteFoundation` — multi-site, visitor, and widget foundations.
3. `Patch_AddChatSiteExperience` — site widget-experience fields.

The multi-site migration seeds one default site (`Primary Website`, key `default-<SITE_ID>`) so existing conversations have a safe site context to backfill into.

Apply the seed SQL

Apply the seed scripts under `core/modules/chat/sql/chat_migration/` to the correct databases:

Script	Database	Seeds
`01_owner_workflow_chat_seed.sql`	Owner	Site type + OAuth groups
`02_config_chat_seed.sql`	Config	OAuth components, form/grid/route/menu metadata
`03_instance_chat_seed.sql`	Instance	Roles + OAuth role access
`04_instance_chat_mailboxsuite_metadata_backfill.sql`	Instance	Email metadata backfill (only if you have legacy mailbox data)

The `bin/generate_chat_migration.php` and `bin/apply_chat_migration.php` utilities can generate and apply these artifacts if your environment uses them.

Sync access & refresh

1. Run `core/modules/chat/bin/sync_chat_oauth_access.php` to sync OAuth component/group definitions into the database.
2. Validate and (if needed) repair the navigation menus: `bin/validate_chat_menu_config.php`, then `bin/reset_chat_menu_configs.php --apply`.

3. Clear caches and re-login so the new roles, menus, and routes take effect.

Deploy the frontend

Deploy the paired React frontend (`core/app/chat`) and import the route/menu/access metadata using your site's metadata-sync tooling. Confirm the Chat menus now appear for an admin account.

2. Real-time delivery (optional but recommended)

Chat works with polling alone, but live chat is best with the real-time gateway. Without it, the agent console falls back to polling (a few seconds' latency).

- **Gateway + Redis** — Chat publishes events to the shared XRealtime gateway over Redis pub/sub. Ensure Redis is reachable and the gateway is running.
- **WebSocket server** — start the Node entrypoint at `core/modules/chat/realtime/server.js` (it bootstraps the shared realtime server).
- **Enable it** — turn on the real-time setting in **Settings** → **General**. Scoped JWT tokens are issued only when the setting is on and the gateway is healthy.

> **Verify:** with the setting on and the gateway up, the conversation workspace shows a live connection indicator and new messages appear without a poll delay. If not, Chat is silently in polling fallback — check the gateway/Redis/Node server.

Public widget endpoints

Confirm these public endpoints are reachable over HTTPS from the internet (they're CORS-enabled and used by the embedded widget):

```
GET  /cars/Chat/widgetAsset/loader?site_key=<key>
GET  /cars/Chat/widgetAsset/frame?site_key=<key>
GET  /cars/Chat/widgetAsset/hosted?slug=<slug>
GET  /cars/Chat/widget/bootstrap        POST /cars/Chat/widget/start
POST /cars/Chat/widget/send             GET  /cars/Chat/widget/poll
GET  /cars/Chat/widget/heartbeat        POST /cars/Chat/widget/offline
POST /cars/Chat/widget/transcript
```

3. Create your first site & go live

Create a site

From **Sites**, create a site (or edit the seeded `Primary Website`). Set the name, status, allowed domains, welcome/offline messages, business hours, theme colour, and email delivery. The site key and widget token are generated automatically.

Install the widget

1. Open the site's **Install** panel and copy the snippet.
2. Paste it before `</body>` on the pages where chat should appear:

```
<script async src="https://your-domain.example/cars/Chat/widgetAsset/loader?site_key=your-site-key"></script>
```

Then use **Verify Install** to confirm the widget is live (it reports the last URL and time it was seen).

Assign agents & go available

- On the site team, choose **All Users** or **Specific Users** (remember: external agents must be assigned explicitly).
- Have an agent set status to *Available* and keep Chat open (the two-factor availability requirement).

Smoke test

1. From the customer site, open the widget and start a chat.
2. Confirm it appears in **Incoming Chats** / **Active Conversations**, claim it, and reply.
3. Set all operators unavailable and confirm the widget shows the offline form and the submission lands in **Email Follow-ups**.

4. Operational configuration

Email delivery

Under **Settings** → **Email Delivery** (or per site), choose a **shared mailbox** (via MailboxSuite — inbound replies sync back into the conversation) or a **fallback email** destination. If neither is set, offline delivery is marked `not_configured`.

Notifications & moderation

- Confirm browser-notification permission and sound alerts work for at least one agent (Settings → Notifications).
- Review **Settings** → **Banned IPs** if you need to pre-block abusive addresses.

Auto-close cron

Schedule the idle-conversation auto-close command so stale chats close on their own:

```
php yiic chatAutoClose run        # dry run: --dryRun=1
```

Run it on a short interval (e.g. every 5 minutes). It only closes conversations whose last message is older than the site's TTL, and re-checks activity before closing.

Tune general settings

In **Settings** → **General**, set the conversation TTL (default 5 min), visitor session idle time (default 30 min), the session reset policy, default messages, business hours, and theme.

5. Verification checklist

Check	Expected
Migrations + seeds applied	`{SITE_ID}_chat_*` tables exist; roles and menus present.
OAuth sync + re-login	Chat menus visible to an admin; agents see their scoped menus.
Frontend deployed	Dashboard, inbox, sites, and settings pages load.
Widget installed	Verify Install reports the widget live on the customer domain.
Live chat path	Visitor chat reaches an available agent in real time.
Offline path	With no agents available, offline form → Email Follow-ups.
Real-time (if enabled)	Connection indicator green; no poll delay.
Auto-close cron	Scheduled and running.

Stuck? The most common issues are covered in the **FAQ & Troubleshooting** guide — widget not appearing (snippet/domain/site key), only the offline form (no available operator or outside business hours), and laggy updates (real-time disabled or gateway down).