



metamint

Helpdesk

Developer Documentation

Plugin Version: 1.5.7 | Text Domain: metamint-helpdesk

Text Domain: metamint-helpdesk

License: GPL-2.0

1. Architecture Overview.....	5
2. File Structure.....	6
3. Database Schema.....	7
3.1 Custom Tables.....	7
3.2 Post Meta Keys (mmhd_ticket).....	8
3.3 Comment Meta Keys.....	9
3.4 User Meta Keys.....	9
3.5 Options (mmhd_prefix).....	10
4. Custom Post Types & Taxonomies.....	12
4.1 Post Types.....	12
4.2 Taxonomies.....	12
4.3 Post Meta (FAQ).....	12
5. REST API Reference.....	14
5.1 Public REST Endpoints.....	14
5.2 Admin REST Endpoints.....	15
5.3 WordPress REST Enhancements.....	16
6. Authentication System.....	17
6.1 Primary Auth: WordPress Native.....	17
6.2 JWT Token.....	17
6.3 Nonce for SSE.....	17
7. SSE Streaming System.....	19
7.1 Architecture.....	19
7.2 Public SSE Endpoints.....	19
7.3 Admin SSE Endpoints.....	20
7.4 Frontend SSE Manager.....	20
7.5 Polling Fallback.....	20
8. AJAX Handlers.....	21
8.1 Public AJAX Handlers.....	21
8.2 Admin AJAX Handlers.....	22
9. Shortcodes.....	24
10. User Roles & Capabilities.....	25
10.1 Custom Roles.....	25
10.2 Capabilities Added to Existing Roles.....	25
10.3 Key Custom Capability: mmhd_access_dashboard.....	25
11. Encryption System.....	26
11.1 Algorithm.....	26
11.2 Key Hierarchy.....	26
11.3 Storage Format.....	26
11.4 Usage.....	26
11.5 Use Cases.....	26
12. IMAP Email Fetching.....	27

12.1 Location.....	27
12.2 Flow.....	27
12.3 Email Cleaning.....	27
12.4 Configuration.....	27
12.5 Cron.....	27
13. SMTP Configuration.....	28
14. Notification System.....	29
14.1 In-App Notifications.....	29
14.2 Email Notifications.....	29
15. Hooks (Actions & Filters).....	31
15.1 Plugin-Specific Actions.....	31
15.2 Filters.....	31
15.3 Key WordPress Hooks Registered.....	31
16. Debug Logging.....	34
16.1 mmhd_log().....	34
16.2 mmhd_log_debug().....	34
16.3 Log Files.....	34
16.4 Log Pruning.....	34
16.5 Admin Viewer.....	34
17. Ticket System.....	35
17.1 Statuses.....	35
17.2 Replies.....	35
17.3 Image Upload.....	35
17.4 Internal Notes.....	35
17.5 Assignment.....	36
17.6 Deeplinks.....	36
17.7 Source Context.....	36
17.8 Ticket Numbering.....	36
18. Credentials Vault.....	37
19. Addon Integration Points.....	38
19.1 Primary Hooks.....	38
19.2 Admin Reply Editor Hooks.....	38
20. Security Model.....	40
20.1 Authentication.....	40
20.2 Access Control.....	40
20.3 Encryption.....	40
20.4 IP Blocking.....	40
20.5 Rate Limiting.....	40
21. Frontend Architecture.....	41
21.1 Build Tooling.....	41

21.2 Component Tree.....	41
21.3 State Management.....	41
21.4 API Helpers (utils/modules.js).....	42
21.5 Key Behaviors.....	42
21.6 Notification Sound.....	42
21.7 Admin Dashboard.....	43
22. Extending the Plugin.....	44
22.1 Creating an Addon.....	44
22.2 Common Extension Pattern.....	44
22.3 Programmatic Ticket Creation.....	44

1. Architecture Overview

Metamint Helpdesk is a WordPress plugin implementing a full-featured support ticket system:

- **Server-side:** PHP WordPress plugin (procedural, no main class) with custom post types, taxonomies, REST API, AJAX handlers, admin SPA (Single Page Application), and email integration.
- **Client-side:** React SPA for the customer chat widget, built with `@wordpress/scripts`. Vanilla JS for the admin dashboard.
- **Real-time:** Server-Sent Events (SSE) via short-lived REST endpoints — handlers run once and exit; the browser `EventSource` auto-reconnects via the `retry:` directive. No long-running PHP-FPM workers.
- **Email:** IMAP fetching for email-to-ticket conversion (in a custom PHP file, not via `wp_mail`), optional SMTP override via `phpmailer_init`.
- **Auth:** WordPress native cookie authentication (`wp_signon/wp_set_auth_cookie`). A JWT is generated and returned on login/signup, but the REST layer uses `is_user_logged_in()` and custom permission callbacks — the JWT is not server-verified at the REST layer.

Key Design Decisions

- Tickets stored as a custom post type (`mmhd_ticket`) with WordPress comments as replies (`comment_type = 'ticket_replies'`)
- Internal staff notes stored as WordPress comments — admin uses `comment_type = 'ticket_notes'` (with `is_mmhd_note` meta), frontend uses `comment_type = 'mmhd_note'`. An unused `wp_mmhd_ticket_notes` table is also created but not consumed by the admin UI.
- Notifications use a custom DB table (`wp_mmhd_notifications`) for performance
- Credentials stored in a custom DB table (`wp_mmhd_credentials`) with AES-256-CBC encryption
- Priorities use a custom taxonomy with drag-and-drop ordering stored as term meta (`taxonomy_order`)
- All SSE endpoints are short-lived — from a PHP perspective they are equivalent to polling, but the browser's `EventSource` handles reconnection transparently

2. File Structure

The plugin follows a conventional WordPress plugin layout, separating admin, public, and shared template concerns:

```
metamint-helpdesk/
├── mmhd.php                                # Main plugin bootstrap
├── readme.md                              # Plugin readme
├── admin/
│   ├── mmhd-admin.php                    # Admin menu, taxonomy ordering, enqueue
│   ├── assets/
│   │   ├── css/ (mmhd-admin-styles.css, mmhd-dashboard.css, tailwind/,
font-awesome.min.css)
│   │   ├── js/ (mmhd-admin-scripts.js, mmhd-dashboard.js, custom-taxonomy-order.js)
│   │   └── images/ (logo + icon assets)
│   └── includes/
│       ├── post-types/
│       │   ├── mmhd-ticket-post-type.php # Ticket CPT + taxonomies + webhook
│       │   ├── mmhd-faq-post-type.php    # FAQ CPT
│       │   └── mmhd-notice-post-type.php  # Notice CPT
│       ├── mmhd-admin-rest.php            # Admin SSE endpoints
│       ├── mmhd-conversation-detail.php    # Single ticket detail page (admin)
│       ├── mmhd-conversations.php         # Ticket list page (admin)
│       ├── mmhd-create-conversation.php    # Create ticket page (admin)
│       ├── mmhd-dashboard.php             # Dashboard page bootstrap
│       ├── mmhd-dashboard-data.php        # Dashboard data helpers
│       ├── mmhd-dashboard-stats.php       # Dashboard metrics
│       ├── mmhd-notice-context.php        # Notice context UI helpers
│       ├── mmhd-fetch-tickets.php         # IMAP email fetching
│       ├── mmhd-functions.php            # Admin helpers, SMTP, credentials, AJAX
│       ├── mmhd-settings.php             # Settings UI
│       └── audio/                        # Notification sounds (*.mp3)
├── public/
│   ├── assets/
│   │   ├── beep.mp3
│   │   ├── css/ (mmhd-styles.css, variables.css)
│   │   ├── css_old/ (legacy stylesheet copies)
│   │   └── js/ (mmhd-scripts.js, chart.js)
│   ├── build/                            # React production build (wp-scripts)
│   ├── includes/
│   │   ├── php-jwt/                      # Firebase JWT library (HS256)
│   │   ├── mmhd-custom-rest.php          # Public REST API + auth + SSE
│   │   ├── mmhd-ajax.php                 # Public AJAX handlers
│   │   ├── mmhd-functions.php            # Public helpers
│   │   ├── mmhd-shortcodes.php           # Shortcodes
│   │   ├── mmhd-comments.php             # Comment callback helpers
│   │   ├── mmhd-notes.php                # Note form shortcode
│   │   ├── mmhd-webhook.php              # Commented out (dead code)
│   │   ├── mmhd-rest-cache.php           # REST caching helpers
│   │   └── mmhd-woocommerce.php          # WooCommerce integration
│   └── webpack.config.js
├── templates/
│   ├── comments-ticket.php               # Ticket comment thread template
│   └── single-mmhd_ticket.php            # Single ticket template
```

Development-only assets (React JSX source in public/src/, npm configuration) live in the source repository and are not part of the distribution ZIP. `mmhd_log()` writes to `wp-content/uploads/mmhd/debug.log` (auto-pruned after 30 days); the admin Debug Log page reads that file.

3. Database Schema

3.1 Custom Tables

wp_mmhd_credentials

Stores encrypted customer credentials. Created on activation and lazily on `init` (guarded by option `mmhd_credentials_table_version`).

```
CREATE TABLE wp_mmhd_credentials (
  id          BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  user_id     BIGINT(20) UNSIGNED NOT NULL,
  name        VARCHAR(255) NOT NULL,
  username    VARCHAR(255) NOT NULL,
  password    LONGTEXT NOT NULL,
  url         VARCHAR(255),
  notes       TEXT,
  created_at  DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id),
  KEY user_id (user_id),
  CONSTRAINT fk_mmhd_credentials_user
    FOREIGN KEY (user_id) REFERENCES wp_users(ID) ON DELETE CASCADE
) $charset_collate;
```

wp_mmhd_notifications

Stores in-app notifications for real-time delivery. Created on activation and lazily on `init` (guarded by option `mmhd_notifications_table_version`).

```
CREATE TABLE wp_mmhd_notifications (
  id              BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  notification_date DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
  user_id         BIGINT(20) UNSIGNED NOT NULL,
  message         TEXT NOT NULL,
  ticket_id       BIGINT(20) UNSIGNED DEFAULT NULL,
  status          VARCHAR(20) NOT NULL DEFAULT 'unread',
  created_by      BIGINT(20) UNSIGNED NOT NULL,
  recipient_user_id BIGINT(20) UNSIGNED NOT NULL,
```

```

comment_id      BIGINT(20) UNSIGNED DEFAULT NULL,
type            VARCHAR(32) NOT NULL,
source          VARCHAR(32) NOT NULL,
created_at      DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
read_at        DATETIME DEFAULT NULL,
PRIMARY KEY (id),
KEY recipient_user_id (recipient_user_id),
KEY ticket_id (ticket_id),
KEY comment_id (comment_id),
KEY read_at (read_at),
KEY user_id (user_id),
KEY created_by (created_by),
KEY status (status)
) $charset_collate;

```

wp_mmhd_ticket_notes (Legacy — unused by admin UI)

A custom table created on `init` via `mmhd_maybe_create_ticket_notes_table()`. The admin AJAX note handlers write notes as WordPress comments (`comment_type = 'ticket_notes'`) instead. This table exists but is NOT consumed by the active admin note flow.

3.2 Post Meta Keys (mmhd_ticket)

Key	Type	Description
<code>_mmhd_ticket_status</code>	string	open, in-progress, closed, archived
<code>_mmhd_customer_email</code>	string	Customer email address
<code>mmhd_ticket_origin</code>	string	web, email, chat, admin (also registered as a taxonomy)
<code>ticket_assignee</code>	string	Assignee display name
<code>ticket_assignee_id</code>	int	Assignee user ID
<code>_mmhd_source_ip</code>	string	Source IP address
<code>_mmhd_source_country</code>	string	Source country name
<code>_mmhd_source_browser</code>	string	Browser/email client label
<code>mmhd_all_seen</code>	string	yes/no — read tracking
<code>mmhd_last_reply_id</code>	int	Last reply comment ID (read-only — never set by this plugin)
<code>last_admin_comment_id</code>	int	Last admin reply comment ID (tracked via <code>comment_post</code> hook)
<code>email_message_id</code>	string	Email Message-ID header (IMAP)
<code>email_from</code>	string	Email From address (IMAP)

3.3 Comment Meta Keys

Used on ticket reply comments (`comment_type = 'ticket_replies'`) and note comments (`comment_type = 'ticket_notes' or 'mmhd_note'`):

Key	Type	Description
<code>author_type</code>	string	agent, customer, bot
<code>comment_source</code>	string	web, email, chat, admin
<code>admin_comment</code>	string	yes/no — used by SSE streams for filtering
<code>is_bot_response</code>	string	1/0
<code>is_action</code>	string	1 for status/priority/assignee change action comments
<code>mmhd_suppress_notification</code>	string	1 to skip notification creation
<code>email_message_id</code>	string	Email Message-ID for IMAP replies
<code>mmhd_unseen_email_tracking_token</code>	string	Read-receipt tracking token
<code>is_mmhd_reply</code>	string	1 for admin replies
<code>is_mmhd_note</code>	string	1 for internal notes (admin path only)

3.4 User Meta Keys

Key	Type	Description
<code>mmhd_user_location</code>	string	User location (e.g., timezone/city)
<code>mmhd_user_department</code>	string	User department/team name

3.5 Options (mmhd_ prefix)

Option key	Format	Description
mmhd_advanced_settings	serialized array	Advanced: auto-assign, notifications, rate limiting, file upload size, polling interval, IP blocklist, unseen email delay
mmhd_imap_settings	serialized array	IMAP hostname, port, username, password, auth
mmhd_smtp_settings	serialized array	SMTP enabled, host, port, username, password, encryption, from_email, from_name
mmhd_email_template_new-ticket	string	Email template for new ticket
mmhd_email_template_ticket-closed	string	Email template for closed ticket
mmhd_email_template_auto-reply	string	Email template for auto-reply
mmhd_email_template_header	string	Email header HTML
mmhd_email_template_footer	string	Email footer HTML
mmhd_fetch_interval	int	IMAP fetch interval in minutes (default 15)
mmhd_last_fetched_date	string	Last IMAP fetch timestamp
mmhd_default_agent	int	Default agent user ID
mmhd_support_page_id	int	Created support page ID
mmhd_notification_sound	string	Uploaded notification sound URL
mmhd_popover_icon	string	Chat widget popover icon URL
mmhd_popover_top_icon	string	Chat widget top icon URL
mmhd_logo	string	Widget logo URL
mmhd_wide_logo	string	Wide logo URL
mmhd_site_name	string	Custom site name
mmhd_admin_email	string	Custom admin email
mmhd_theme_color	string	Theme color hex code (default #0073aa)
mmhd_chat_heading	string	Chat heading text
mmhd_welcome_message	string	Welcome message text
mmhd_ticket_intro_message	string	Ticket intro text
mmhd_default_ticket_category	int	Default ticket category term ID
mmhd_blocked_ips	serialized array	Permanently blocked IPs
mmhd_credentials_table_version	string	Schema version for credentials table
mmhd_notifications_table_version	string	Schema version for notifications table

Option key	Format	Description
mmhd_advanced_settings	serialized array	Advanced: auto-assign, notifications, rate limiting, file upload size, polling interval, IP blocklist, unseen email delay
mmhd_imap_settings	serialized array	IMAP hostname, port, username, password, auth
mmhd_smtp_settings	serialized array	SMTP enabled, host, port, username, password, encryption, from_email, from_name
mmhd_email_template_new-ticket	string	Email template for new ticket
mmhd_email_template_ticket-closed	string	Email template for closed ticket
mmhd_email_template_auto-reply	string	Email template for auto-reply
mmhd_assignee_id_backfill_done	string	Flag for assignee ID backfill
mmhd_page_check_complete	string	Flag for support page check

4. Custom Post Types & Taxonomies

4.1 Post Types

Post Type	Key	Public	Queryable	Archive	REST	Menu	Slug	Supports
Ticket	<code>mmhd_ticket</code>	true	logged-in only	true	true	false	mmhd-tickets	title, editor, author, comments, custom-fields
FAQ	<code>mmhd_faq</code>	true	true	true	true	false	faqs	title, editor, excerpt, author, custom-fields, thumbnail
Notice	<code>notice</code>	true	true	true	true	false	default	title, editor, excerpt, author, thumbnail

- `create_posts` → `create_tickets`, `edit_posts` → `edit_tickets`, `edit_others_posts` → `edit_others_tickets`
- `publish_posts` → `publish_tickets`, `read_private_posts` → `read_private_tickets`
- `read_post` → `read_ticket`, `delete_post` → `delete_ticket`
- `map_meta_cap` → true

4.2 Taxonomies

Taxonomy	Post Type	Hierarchical	Slug	Term Meta
<code>mmhd_ticket_origin</code>	mmhd_ticket	yes	<code>ticket-origin</code>	none
<code>mmhd_ticket_priority</code>	mmhd_ticket	yes	<code>ticket-priority</code>	priority_color (string), taxonomy_order (int, drag-drop)
<code>mmhd_ticket_category</code>	mmhd_ticket	yes	<code>ticket-category</code>	none
<code>mmhd_faq_category</code>	mmhd_faq	yes	<code>faq-category</code>	none

4.3 Post Meta (FAQ)

Meta Key	Type	Description
<code>_mmhd_featured</code>	string	'true'/'false' — featured FAQ flag
<code>_mmhd_faq_upvotes</code>	int	Upvote count
<code>_mmhd_faq_downvotes</code>	int	Downvote count

5. REST API Reference

5.1 Public REST Endpoints

All routes registered via `rest_api_init` in `public/includes/mmhd-custom-rest.php`.

The webhook endpoint (`POST /chat/v1/tickets`) is registered in `admin/includes/post-types/mmhd-ticket-post-type.php`.

Authentication Endpoints

Endpoint	Method	Permission	Description
<code>/mmhd/v1/login</code>	POST	<code>__return_true</code>	Login via email/password, returns JWT. Calls <code>wp_signon()</code> + <code>wp_set_current_user()</code> + <code>wp_set_auth_cookie()</code> .
<code>/mmhd/v1/signup</code>	POST	<code>__return_true</code>	Register with email. Creates user (role subscriber, random 12-char password), signs in, sends <code>wp_new_user_notification</code> . Optionally IP rate-limited (5 attempts per 2 hours).
<code>/mmhd/v1/lost-password</code>	POST	<code>__return_true</code>	Password reset email via <code>retrieve_password()</code> . Optionally IP rate-limited (5 per hour).

Ticket Endpoints

POST `/mmhd/chat/v1/tickets` — Permission: `__return_true`

Unified ticket handler. All chat widget operations use this single endpoint with an `event` discriminator in the JSON body:

```
{
  "event": "ticket.created" | "tickets.get" | "replies.receive" | "comments.get",
  "data": { ... }
}
```

Event	Fields	Description
ticket.created	title, description, uid, timestamp	Create ticket with status open, origin chat. Optionally auto-assigns.
tickets.get	uid	List user's tickets via <code>WP_Query</code>
replies.receive	ticket_id, user_id, message, image (base64), is_bot_response, timestamp	Add reply. Image decoded and saved as WP Media Library attachment, then <code></code> tag appended to message. Fires <code>mmhd_after_reply_added</code> .
comments.get	ticket_id	Get all ticket_replies comments for a ticket

POST `/chat/v1/tickets` — Permission: `__return_true`, but verifies `X-Webhook-Signature` header

Webhook endpoint for external integrations. Checks `hash_hmac('sha256', raw_body, mmhd_get_webhook_secret())` and the `X-Webhook-Signature` header against the header value via `hash_equals()`. Routes to `mmhd_handle_tickets_webhook`. Supports the same four events as the chat endpoint.

Selected Staff

GET /mmhd/selected/staff — Permission: `is_user_logged_in()`

Returns default/selected staff member data from `mmhd_default_agent` option: `{staff_display_name, staff_user_image, staff_user_email, staff_user_id}`.

WhatsApp Settings

GET /mmhd-bot/v1/wa-settings — Permission: `__return_true`

Returns WhatsApp phone number from `mmhd_whatsapp_credentials` option: `{phoneNumber}`.

REST Field Enhancements

Registered via `register_rest_field`:

- `author_image` — avatar URL on comment responses (`mmhd-custom-rest.php:128`)
- `is_bot_response` — boolean on comment responses (`mmhd-custom-rest.php:164`)
- `author_name` — on `mmhd_faq` post type (`mmhd-faq-post-type.php:185`)

CORS

Applied via `rest_pre_serve_request` in `mmhd-custom-rest.php:291`:

```
Access-Control-Allow-Origin: <site_url>
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: Content-Type, Authorization
Access-Control-Max-Age: 86400
```

5.2 Admin REST Endpoints

All SSE endpoints in `admin/includes/mmhd-admin-rest.php`.

Endpoint	Method	Permission	Description
<code>/mmhd/v1/adminstream/{post_id}/{last_comment_id?}/{user_id?}</code>	GET	<code>is_user_logged_in()</code> + <code>mmhd_access_dashboard</code>	SSE stream of customer comments for agent dashboard (queries where <code>admin_comment = 'no'</code>). Emits

Endpoint	Method	Permission	Description
			new_admin_comment, disconnect. Sets mmhd_all_seen = 'yes'. Retry 5000ms.
/mmhd/v1/notification-stream	GET	is_user_logged_in()	SSE stream of unread notifications from wp_mmhd_notifications. Emits notifications. Short-lived.
/mmhd/v1/ticket-stream/{ticket_id}	GET	mmhd_access_dashboard	SSE stream of ticket replies + notes + status. Emits sync with full dataset.

5.3 WordPress REST Enhancements

- `rest_mmhd_ticket_query` filter supports search, ticket_priority, ticket_status query params (mmhd-ticket-post-type.php:402)
- `rest_prepare_mmhd_ticket` adds ticket_number (TCKT-XXX format), status, priority, origin, comments to ticket responses
- `rest_comment_query` filter mmhd_include_ticket_replies_in_rest_comments ensures ticket_replies comment type is included (mmhd.php:2221)

6. Authentication System

6.1 Primary Auth: WordPress Native

The plugin uses WordPress's native cookie authentication:

- **Login:** POST /mmhd/v1/login → wp_signon() → wp_set_current_user() → wp_set_auth_cookie().
- **Signup:** POST /mmhd/v1/signup → wp_insert_user() → wp_signon() → wp_set_auth_cookie().
- **Session:** Standard WordPress auth cookies. No custom cookie names, no determine_current_user override.
- **Role-checking:** Protected routes use is_user_logged_in(), current_user_can('mmhd_access_dashboard'), or mmhd_can_view_ticket().

6.2 JWT Token

A JWT (HS256) is generated on login/signup via `mmhd_generate_token()` (`mmhd-custom-rest.php:251`):

- **Library:** Firebase\JWT\JWT
- **Algorithm:** HS256
- **Payload:** {iss: site_url, iat, exp, data: {user_id, user_email}}
- **Expiry:** 7 days (DAY_IN_SECONDS * 7)
- **Secret key:** MMHD_JWT_SECRET constant if defined, else wp_hash('mmhd-jwt-secret') (not persisted)
- **Returned to client** in the login/signup response. GET /mmhd/v1/validate-token validates a Bearer token via mmhd_validate_token() (checks issuer, expiry, user) and sets the current user. Ticket-messaging permission callbacks still rely on cookie auth.

6.3 Nonce for SSE

SSE `EventSource` cannot send custom headers, so REST nonces are passed as `?_wpnonce=<nonce>` query parameters. The nonce is exposed via `window.mmhdAuth.restNonce` (injected by PHP via `wp_create_nonce('wp_rest')` at `mmhd.php:477`), with fallback to `window.wpApiSettings.nonce`. After AJAX login, a fresh nonce is returned in the login response and stored into `wpAuth.restNonce`.

7. SSE Streaming System

7.1 Architecture

All SSE endpoints are short-lived (single-shot). They send available data and exit immediately. The browser's `EventSource` auto-reconnects via the `retry` SSE directive:

Retry is endpoint-specific: tickets 5000, messaging 5000, admin stream 5000, comments 3000.
Ticket-stream also uses 1-second ``usleep``.

This approach:

- Eliminates long-running PHP-FPM processes
- Works with any hosting environment
- Provides equivalent real-time behavior via transparent browser reconnection

7.2 Public SSE Endpoints

GET /mmhd/v1/messaging/{post_id}/{last_comment_id?}/{user_id?}

- Permission: mmhd_can_view_ticket
- Namespace: mmhd/v1
- Events: handshake (connection status), new_comment (new replies), ticket_closed, ticket_open, disconnect

GET /mmhd/v1/stream/{type}/{post_id}/{last_id?}/{user_id?}

- Permission: inline (is_user_logged_in() + user ID match)
- Namespace: mmhd/v1
- type = comments (post_id = ticket ID): retry 3000ms. Events: new_comment, new_mine_comment, ticket_closed, ticket_open, heartbeat
- type = tickets (post_id = 0): retry 5000ms. Events: new_ticket, new_notification, heartbeat
- Other types → error event

•

Endpoint	Retry	Events	Consumer
/mmhd/v1/messaging/{post_id}/{last_comment_id?}/{user_id?}	5000ms	handshake, new_comment, ticket_closed, ticket_open, disconnect	Customer portal
/mmhd/v1/stream/comments/{post_id}/{last_id}/{user_id}	3s	new_comment, new_mine_comment, ticket_closed, ticket_open	Chat widget
/mmhd/v1/stream/tickets/0/{last_id}/{user_id}	5s	new_ticket, new_notification	Chat widget

7.3 Admin SSE Endpoints

Endpoint	Retry	Events	Consumer
/mmhd/v1/ticket-stream/{ticket_id}	1s sleep	sync (full replies + notes + status)	Admin dashboard
/mmhd/v1/notification-stream	none	notifications (items + count)	Admin dashboard
/mmhd/v1/adminstream/{post_id}/{last_comment_id?}/{user_id?}	5s	new_admin_comment (customer replies), disconnect	Admin conversation view

7.4 Frontend SSE Manager

The React app uses a singleton `SSEManager` class (`src/Hooks/sseManager.js`):

- `sseManager.connect(url, channel = "default")` — creates new `EventSource(url)`, supports multiple named channels
- `sseManager.on(event, callback, channel = "default")` — subscribe to named SSE events, returns unsubscribe function
- `sseManager.disconnect(channel)` — closes connection for channel (or all if omitted)

Named events: `new_ticket`, `new_notification`, `new_comment`, `new_mine_comment`, `ticket_closed`, `ticket_open`, `error`.

Channel usage:

- `tickets` channel (in `App.js`): subscribes `new_ticket` → `updateTicket()`, `new_notification` → beep + UI update
- `comments` channel (in `SingleTicket.jsx`): subscribes `new_comment` → beep + append, `new_mine_comment` → silent append, `ticket_closed/ticket_open` → UI state update

7.5 Polling Fallback

The chat widget (`SingleTicket.jsx`) also polls `GET /wp/v2/comments?post={id}&per_page=20&order=desc&comment_type=ticket_replies&fields=id,content,date,author_name,author_image,author_data,time_ago,comment_action,is_bot_response` every 4 seconds as a fallback when SSE fails. New comments are merged by message ID with deduplication.

8. AJAX Handlers

8.1 Public AJAX Handlers

Registered in `public/includes/mmhd-ajax.php` (except `mmhd_submit_comment` in `mmhd-shortcodes.php` and `mmhd_add_note` in `mmhd-notes.php`):

Action	Callback	priority	Nonce Field	Description
<code>mmhd_change_ticket_status</code>	<code>mmhd_change_ticket_status_callback</code>	no	nonce	Update <code>_mmhd_ticket_status</code> (valid: new, in-progress, archived, completed), fires <code>mmhd_notify_status_changed</code>
<code>mmhd_create_new_ticket</code>	<code>mmhd_handle_ticket_submission</code>	no	<code>mmhd_nonce</code>	Create ticket from frontend, assigns initial comment, auto-assigns if enabled
<code>mmhd_faq_vote</code>	<code>mmhd_faq_vote_callback</code>	no	<code>_wpnonce</code> (default)	Increment FAQ upvote/downvote meta
<code>mmhd_get_credential_form</code>	<code>mmhd_get_credential_form_callback</code>	no	nonce	Return HTML credential form
<code>mmhd_add_credential</code>	<code>mmhd_add_credential_callback</code>	no	<code>_wpnonce</code> (default)	Insert credential with encrypted password
<code>mmhd_delete_credential</code>	<code>mmhd_delete_credential_callback</code>	no	nonce	Delete own credential (ownership-checked)
<code>mmhd_reveal_credential_password</code>	<code>mmhd_reveal_credential_password_callback</code>	no	nonce	Decrypt and return credential password
<code>mmhd_mark_user_notification_read</code>	<code>mmhd_mark_user_notification_read</code>	no	nonce	Delete single notification row
<code>mmhd_mark_all_user_notifications_read</code>	<code>mmhd_mark_all_user_notifications_read</code>	no	nonce	Delete all notifications for user
<code>mmhd_check_email</code>	<code>mmhd_check_email</code>	yes	security	Rate-limited (10/hr) email existence check
<code>mmhd_register_user</code>	<code>mmhd_register_user</code>	yes	none	Create WP user
<code>change_password</code>	<code>mmhd_handle_change_password</code>	yes	nonce	Password reset (5/hr rate limit)
<code>mmhd_submit_comment</code>	<code>mmhd_submit_comment</code>	yes	<code>mmhd_reply_nonce</code>	Submit ticket reply via shortcode form
<code>mmhd_add_note</code>	<code>mmhd_add_note_callback</code>	no	nonce	Add internal note as comment (<code>mmhd_note</code> type) via <code>wp_insert_comment</code>
<code>mmhd_add_ticket_note</code>	<code>mmhd_add_ticket_note_callback</code>	no	<code>_wpnonce</code> (default)	Add internal note to legacy <code>wp_mmhd_ticket_notes</code> table

Action	Callback	nopriv	Nonce Field	Description
				(active callback at mmhd-ajax.php:478)

8.2 Admin AJAX Handlers

Registered in `admin/includes/mmhd-functions.php` and `admin/mmhd-admin.php` (nonce only, no nopriv versions — all require login):

Ticket Management

Action	Callback	Nonce Field	Description
<code>mmhd_create_ticket</code>	<code>mmhd_create_ticket</code>	nonce	Create ticket + initial reply comment
<code>get_ticket_detail</code>	<code>mmhd_get_ticket_detail</code>	nonce	Return full ticket detail as JSON
<code>update_ticket_status</code>	<code>mmhd_update_ticket_status</code>	security	Update <code>_mmhd_ticket_status</code> + action comment
<code>update_ticket_priority</code>	<code>mmhd_update_ticket_priority</code>	security	Assign priority term + action comment
<code>update_ticket_assignee</code>	<code>mmhd_update_ticket_assignee</code>	security	Set assignee + action comment
<code>update_ticket_category</code>	<code>handle_update_ticket_category</code>	nonce	Set category term
<code>add_ticket_reply</code>	<code>mmhd_handle_add_ticket_reply</code>	nonce	Insert agent reply comment
<code>close_ticket</code>	<code>mmhd_handle_close_ticket</code>	nonce	Set status closed + optional reply + customer email
<code>get_ticket_replies</code>	<code>mmhd_handle_get_ticket_replies</code>	nonce	Return <code>ticket_replies</code>
<code>get_ticket_notes</code>	<code>mmhd_handle_get_ticket_notes</code>	nonce	Return <code>ticket_notes</code>
<code>get_ticket_replies_and_notes</code>	<code>mmhd_handle_get_ticket_replies_and_notes</code>	nonce	Combined replies + notes
<code>add_ticket_note</code>	<code>mmhd_handle_add_ticket_note</code>	nonce	Insert note comment (<code>ticket_notes</code> type)
<code>mmhd_bulk_action</code>	<code>mmhd_handle_bulk_action</code>	nonce	Bulk delete/status/priority/assignee

Dashboard & Listing

Action	Callback	Nonce Field	Description
mmhd_refresh_dashboard_tickets	mmhd_refresh_dashboard_tickets	nonce	Return all tickets as JSON for dashboard
mmhd_get_notifications	mmhd_get_notifications	nonce	Read mmhd_notifications table
mmhd_mark_notification_read	mmhd_mark_notification_read	nonce	Delete single notification
mmhd_clear_notifications	mmhd_clear_notifications	nonce	Delete all unread for user
mmhd_user_lookup	mmhd_user_lookup	nonce	User autocomplete by name/email

Settings & Configuration

Action	Callback	Nonce Field	Description
mmhd_imap_test_connection	mmhd_imap_test_connection	nonce	Test IMAP connection
mmhd_imap_save_settings	mmhd_imap_save_settings	nonce	Save IMAP settings + reschedule cron
mmhd_smtp_save_settings	mmhd_smtp_save_settings	nonce	Save SMTP settings
mmhd_smtp_test_connection	mmhd_smtp_test_connection	nonce	Test SMTP connection
mmhd_email_template_save	mmhd_email_template_save	nonce	Save email templates
mmhd_save_advanced_settings	mmhd_save_advanced_settings	nonce	Save advanced settings
mmhd_save_basic_settings	mmhd_save_basic_settings	nonce	Save basic settings + logo upload
mmhd_save_cropped_logo	mmhd_save_cropped_logo	nonce	Save cropped logo image
mmhd_save_ticket_assignment_settings	mmhd_save_ticket_assignment_settings	nonce	Save assignment + fetch interval settings

Other

Action	Callback	Nonce Field	Description
mmhd_fetch_emails	mmhd_fetch_emails_handler	nonce	Manually trigger IMAP fetch
update_taxonomy_order	mmhd_update_taxonomy_order	taxonomy_order_nonce	Save dragged priority order
mmhd_track_unseen_reply_email_read	mmhd_track_unseen_reply_email_read	nopriv + token	Email tracking pixel

9. Shortcodes

Shortcode	Callback	Auth	Attributes	Description
[mmhd_frontend_dashboard]	mmhd_frontend_dashboard_shortcode()	Yes	none	Full customer portal SPA. Requires mmhd_is_customer_portal_enabled(). Uses query vars mmhd_page, tid. Menu items via mmhd_frontend_dashboard_menu_items filter.
[mmhd_my_tickets]	mmhd_my_tickets_shortcode()	Yes	none	Paginated ticket table + filter form. Reads GET params orderby, order, mmhd_ticket_priority, mmhd_ticket_status.
[mmhd_ticket_form]	mmhd_ticket_form_shortcode()	Yes	none	Ticket submission form (title, description, origin, priority). Self-handles POST submit via nonce mmhd_submit_ticket_action.
[mmhd_single_ticket id="123"]	mmhd_single_ticket_shortcode()	Yes	id (default 0)	Single ticket view with comments thread + meta panel (status, priority, dates). Permission-checked (author or staff).
[mmhd_login_form]	mmhd_login_form_shortcode()	No	none	Login/registration form. Redirects logged-in users.
[mmhd_note_form]	mmhd_note_form_shortcode()	Yes	ticket_id (default 0)	Internal note textarea + button (nonce mmhd_add_note_nonce).

The chat widget is **not** a shortcode — it's injected via `wp_footer` as `<div id="mmhd-plugin">` by `mmhd_add_plugin()`, which is the React mount point.

10. User Roles & Capabilities

10.1 Custom Roles

Created on plugin activation via `add_role()` (`mmhd.php:138`):

Role	Key	Capabilities
Support Manager	<code>mmhd_support_manager</code>	read, manage_options, edit_posts, manage_categories, mmhd_access_dashboard, create_tickets, edit_tickets, edit_others_tickets, publish_tickets, read_private_tickets, read_ticket, delete_ticket
Support Staff	<code>mmhd_support_staff</code>	read, edit_posts, manage_categories, mmhd_access_dashboard, create_tickets, edit_tickets, edit_others_tickets, read_ticket
Support Customer	<code>mmhd_support_customer</code>	read, create_tickets, read_ticket

10.2 Capabilities Added to Existing Roles

Role	Additional Capabilities
Administrator	All ticket caps + mmhd_access_dashboard
Editor	All ticket caps + mmhd_access_dashboard

10.3 Key Custom Capability: mmhd_access_dashboard

Gates access to the entire Metamint Helpdesk admin menu and dashboard SPA. Also used in:

- `mmhd_get_author_type()` — returns 'agent' if user has this cap or holds any staff/manager/admin role
- `mmhd_show_admin_bar()` — shows admin bar for users with this cap or administrator role
- `mmhd_user_can_access_ticket()` — full access for users with this cap

The three custom roles are removed on plugin deactivation.

11. Encryption System

11.1 Algorithm

AES-256-CBC via `openssl_encrypt` / `openssl_decrypt`.

11.2 Key Hierarchy

```
function mmhd_get_encryption_key() {  
    if (defined('MMHD_ENCRYPTION_KEY') && MMHD_ENCRYPTION_KEY !== '') {  
        return MMHD_ENCRYPTION_KEY;  
    }  
    return wp_hash('mmhd-encryption-key');  
}
```

The raw key (no hashing/derivation) is passed directly to OpenSSL's `openssl_encrypt`.

11.3 Storage Format

```
base64_encode(ciphertext . '::' . iv)
```

Decoding splits on `::` separator. `mmhd_decrypt()` returns `false` if no separator found. The helper `mmhd_get_credential_password_value()` ([public/includes/mmhd-functions.php:154](#)) first tries decryption and falls back to returning the raw value for backward compatibility with legacy unencrypted credentials.

11.4 Usage

```
$encrypted = mmhd_encrypt($plaintext);  
$decrypted = mmhd_decrypt($encrypted);
```

11.5 Use Cases

- Credentials Vault passwords stored in `wp_mmhd_credentials.password`
- Ticket deeplinks use the same `mmhd_encrypt()` via `mmhd_get_encryption_key()`; `mmhd_decrypt()` carries legacy fallbacks for old links

12. IMAP Email Fetching

12.1 Location

`admin/includes/mmhd-fetch-tickets.php` — called from:

- Cron: `mmhd_email_fetch_cron` (recurring)
- Manual: `mmhd-fetch-tickets.php` admin page via `check_admin_referer`
- AJAX: `mmhd_fetch_emails` action → `mmhd_fetch_emails_handler`

12.2 Flow

1. Connect via `imap_open('{hostname:port/imap/ssl}INBOX', username, password)`
2. Search SINCE last fetched date (option `mmhd_last_fetched_date`, default 2021-01-01)
3. Sort oldest to newest
4. For each unseen email: skip duplicates and mailer-daemon senders, check reply threading via In-Reply-To/References headers, then create a new ticket or append a reply comment as appropriate
5. Update `mmhd_last_fetched_date`

For a new ticket: creates a `mmhd_ticket` (origin `email`, status `open`), ensures `Email` origin term + `Medium` priority term exist, adds an initial `ticket_replies` comment, auto-assigns if enabled, and notifies. For a reply: adds a `ticket_replies` comment to the existing ticket.

12.3 Email Cleaning

`mmhd_clean_email_message($message)` strips:

- HTML blockquotes, Gmail quotes, Outlook signatures, stopSpelling/divRplyFwdMsg markers, horizontal rules
- Plain-text: "On ... wrote:", "-----Original Message-----", "From: ... Sent:", 32+ underscores, 10+ dashes
- Lines starting with > (quoted)
- Remaining HTML tags via `wp_strip_all_tags`
- Collapses whitespace and newlines
- Cleanups trailing "On ... wrote:" patterns

12.4 Configuration

Stored in option `mmhd_imap_settings: {hostname, port, username, password, auth}`.

12.5 Cron

- Custom interval `mmhd_fetch_interval` registered via `cron_schedules` filter
- Hook: `add_action('mmhd_email_fetch_cron', 'mmhd_fetch_emails')`
- Interval default: 15 minutes (configurable 1-1440 min, option `mmhd_fetch_interval`)
- Rescheduled on settings save; cleared on deactivation

13. SMTP Configuration

Optional SMTP system in `admin/includes/mmhd-functions.php`. Replaces `wp_mail()` via `phpmailer_init`.

Configured in option `mmhd_smtp_settings`: `{enabled, host, port, username, password, encryption, from_email, from_name}`.

When enabled, sets PHPMailer properties: `isSMTP()`, `Host`, `Port`, `SMTPAuth`, `Username`, `Password`, `SMTPSecure`, `From`, `FromName`.

14. Notification System

14.1 In-App Notifications

Table: `wp_mmhd_notifications`

Types (`type` column):

- `ticket_created` — new ticket notification
- `reply_added` — new reply notification (aggregated: N new replies message for same ticket)
- `status_changed` — status change notification

Routing:

- Assigned tickets → notify only the assignee
- Unassigned tickets → notify all support users (those with `mmhd_access_dashboard` cap)
- Aggregation: consecutive `reply_added` notifications for the same ticket + recipient merge into "N new replies" instead of adding rows. Only applies when `author_type === 'customer'` (customer replies). Agent replies always create individual notifications.

Triggers:

- `mmhd_notify_ticket_created($ticket_id, $source = '')` — on ticket creation. `$source` defaults to empty string.
- `mmhd_notify_status_changed($ticket_id, $new_status, $actor_user_id = 0, $source = '')` — on status change (`mmhd.php:1616`). `$new_status` is required (the new status value).
- `mmhd_notify_comment_created($comment_id)` — on reply (via `mmhd_after_reply_added` hook)

Consumption:

- SSE: `/mmhd/v1/notification-stream` (count + items)
- AJAX: `mmhd_get_notifications`, `mmhd_mark_notification_read`, `mmhd_clear_notifications`

14.2 Email Notifications

Sent via `wp_mail()` (or SMTP if configured):

- New ticket: to customer
- Status changed: to customer
- New reply: to customer (via `mmhd_send_ticket_update_email`)
- Unseen reply reminder: scheduled single event `mmhd_send_unseen_reply_email` after configurable delay (minutes). Includes encrypted deeplink + tracking pixel. Fires on `mmhd_after_reply_added` if `mmhd_all_seen != 'yes'`.

Email templates: stored as options

(`mmhd_email_template_header/footer/new-ticket/ticket-closed/auto-reply`), rendered by `mmhd_get_email_template_parts()` in `admin/includes/mmhd-functions.php` — HTML with logo

header ({{BRAND_HTML}} placeholder injected at render time), subject line, content area (wp_kses_post), and dark footer. {{THEME_COLOR}} is normalized into the header CSS on load.

Template options:

- mmhd_email_template_new-ticket
- mmhd_email_template_ticket-closed
- mmhd_email_template_auto-reply
- mmhd_email_template_header / mmhd_email_template_footer

15. Hooks (Actions & Filters)

15.1 Plugin-Specific Actions

Hook	Arguments	File	Description
<code>mmhd_after_reply_added</code>	<code>\$ticket_id,</code> <code>\$comment_id,</code> <code>\$reply,</code> <code>\$user_object</code>	mmhd-functions.php:1607, mmhd-custom-rest.php:1713, mmhd-ticket-post-type.php:776	Fires after any reply is added. Primary integration hook for addons (Slack, Discord, WhatsApp).
<code>mmhd_settings_content</code>	(none)	mmhd-settings.php:817	Render custom settings tab content.
<code>mmhd_email_fetch_cron</code>	(none)	mmhd-functions.php:923	Cron hook for periodic IMAP email fetching.
<code>mmhd_send_unseen_reply_email</code>	<code>\$comment_id</code>	mmhd.php:1958	Cron hook for unseen-reply reminder emails.
<code>mmhd_before_reply_editor</code>	(none)	mmhd-conversation-detail.php:86	Before admin reply editor rendered.
<code>mmhd_after_reply_editor</code>	(none)	mmhd-conversation-detail.php:104	After admin reply editor rendered.

15.2 Filters

Filter	Args	File	Description
<code>mmhd_settings_tabs</code>	<code>\$tabs</code>	mmhd-settings.php:36	Register addon settings tabs. Returns [tab_key => tab_label].
<code>mmhd_before_create_ticket_from_email</code>	<code>\$proceed,</code> <code>\$email_data</code>	mmhd-fetch-tickets.php:398	Block emails from becoming tickets.
<code>mmhd_frontend_dashboard_menu_items</code>	<code>\$menu_items</code>	mmhd-shortcodes.php:431	Customize frontend dashboard menu items.

15.3 Key WordPress Hooks Registered

Hook	Handler	File	Description
<code>init</code>	<code>mmhd_maybe_create_credentials_table</code>	mmhd.php:523	Ensure credentials table exists
<code>init</code>	<code>mmhd_maybe_create_notifications_table</code>	mmhd.php:571	Ensure notifications table exists

Hook	Handler	File	Description
init	mmhd_backfill_ticket_assignee_ids	mmhd.php:904	Migrate legacy assignee meta
rest_api_init	closures	mmhd-custom-rest.php, mmhd-admin-rest.php	Register all REST routes
wp_enqueue_scripts	mmhd_enqueue_scripts	mmhd.php:222,366	Frontend CSS/JS
wp_enqueue_scripts	mmhd_enqueue_custom_styles	mmhd.php:2217	Inject CSS custom properties for theme color
wp_footer	mmhd_add_plugin	mmhd.php:372	Render React mount <div id="mmhd-plugin">
comment_post	mmhd_comment_creation	mmhd.php:2303	Track last admin comment ID, set source context
wp_insert_comment	mmhd_comment_creation_on_insert	mmhd.php:2316	Set author_type, comment_source, source IP/country/browser
admin_menu	mmhd_add_admin_menu	mmhd-admin.php:15	Register admin menu pages
admin_init	mmhd_prune_debug_log_on_admin_load	mmhd-functions.php:155	Prune debug log on admin load
admin_post_mmhd_save_settings	mmhd_handle_settings_save	mmhd.php:409	Handle logo/image uploads from settings page
admin_post_mmhd_download_debug_log	mmhd_download_debug_log	mmhd-functions.php:186	Download debug log
rest_pre_serve_request	closure	mmhd-custom-rest.php:291	CORS headers
rest_comment_query	mmhd_include_ticket_replies_in_rest_comments	mmhd.php:2221	Include ticket_replies in comment REST queries
rest_prepare_comment	closure	mmhd.php:2231	Add comment meta to REST responses
show_admin_bar	mmhd_show_admin_bar	mmhd.php:2386	Hide admin bar for customers
wp_mail_failed	closure	mmhd.php:2048	Log mail failures
plugins_loaded	mmhd_check_support_page	mmhd.php:2195	Verify support page exists
cron_schedules	mmhd_add_fetch_cron_interval	mmhd-functions.php:911	Register fetch cron interval
phpmailer_init	mmhd_apply_smtp_to_phpmailer	mmhd-functions.php:476	Apply SMTP config to PHPMailer
preprocess_comment	mmhd_add_custom_comment_type	mmhd-admin.php:225	Force ticket comments to ticket_replies type
save_post	set_default_taxonomy_term	mmhd-functions.php:264	Set default ticket category on save

Hook	Handler	File	Description
save_post	mmhd_send_ticket_creation_email	mmhd-functions.php:2732	Send email on ticket creation
pre_get_terms	mmhd_modify_taxonomy_query	mmhd-admin.php:167	Order priorities by taxonomy_order meta
created_term	mmhd_set_default_taxonomy_order	mmhd-admin.php:188	Set default taxonomy_order on new term
get_terms	mmhd_custom_taxonomy_order	mmhd-admin.php:214	Sort terms by taxonomy_order
wp_default_scripts	mmhd_disable_jquery_migrate_logging	mmhd.php:2383	Suppress jQuery migrate deprecation logs
use_block_editor_for_post	mmhd_disable_block_for_post_type	mmhd.php:2092	Disable block editor for mmhd_faq, notice
use_block_editor_for_post_type	closure	mmhd-ticket-post-type.php:849	Disable block editor for mmhd_ticket
retrieve_password_message	mmhd_add_param_to_reset_link	mmhd-custom-rest.php:1847	Add deeplink params to password reset URL

16. Debug Logging

16.1 mmhd_log()

```
mmhd_log('String message');  
mmhd_log(['action' => 'status_change', 'from' => 'open']);
```

Logs include timestamps [Y-m-d H:i:s]. Arrays/objects via `print_r()`. Writes to `wp-content/uploads/mmhd/debug.log`. Auto-prunes entries older than 30 days after each write.

16.2 mmhd_log_debug()

A lighter-weight logging wrapper used in `mmhd-custom-rest.php`. Writes '[Metamint Helpdesk] <message>' to the server's standard `WP debug.log` via `error_log()`, but only when both `WP_DEBUG` and `WP_DEBUG_LOG` are enabled. Used primarily in the REST API and authentication functions.

16.3 Log Files

- `wp-content/uploads/mmhd/debug.log` — primary `mmhd_log()` file

16.4 Log Pruning

`mmhd_prune_debug_log()` removes entries older than 30 days. Called after every `mmhd_log()` and on `admin_init` via `mmhd_prune_debug_log_on_admin_load`.

16.5 Admin Viewer

Admin menu → Metamint Helpdesk → Debug Log. Reads last 30 days (2MB cap via `$max_bytes = 2097152`). Supports download via `admin_post_mmhd_download_debug_log`.

17. Ticket System

17.1 Statuses

Stored as `_mmhd_ticket_status` post meta:

Status	Description
open	New ticket awaiting response
in-progress	Staff actively working
closed	Ticket closed
archived	Closed and archived

Status changes trigger: in-app notifications (`mmhd_notify_status_changed`), action comment (`is_action = 1 on ticket_replies` comment), email notification to customer.

17.2 Replies

WordPress comments with `comment_type = 'ticket_replies'`. Comment meta tracked: `author_type` (agent/customer/bot), `comment_source` (web/email/chat/admin), `admin_comment` (yes/no), `is_bot_response`, `is_action`, `is_mmhd_reply`, `mmhd_suppress_notification`, source IP/country/browser. Note: `mmhd_get_author_type()` (defined at `mmhd.php:1281`) returns 'bot' for user ID <= 0, 'agent' for admin/dashboard-capable roles, and 'customer' for all others.

17.3 Image Upload

When the chat widget sends a reply with an image, the frontend sends a base64 data URL in the `image` field. The PHP handler (`mmhd_handle_reply_receive in mmhd-custom-rest.php:1622`):

- 1. Decodes the base64 data
- 2. Saves to WordPress Media Library as an attachment (attached to the ticket post)
- 3. Appends `<a href="..."></a>` to the comment content with the real URL
- 4. Returns `attachment_url` in the API response

17.4 Internal Notes

Stored as WordPress comments. There are two storage paths:

- Admin (via AJAX `add_ticket_note` in `mmhd-functions.php`): `comment_type = 'ticket_notes'`, meta `is_mmhd_note = true`
- Frontend (via shortcode form `mmhd_add_note_callback` in `mmhd-notes.php`): `comment_type = 'mmhd_note'`, no `is_mmhd_note` meta set

The legacy `wp_mmhd_ticket_notes` custom table is created on init but not consumed by the admin UI.

17.5 Assignment

Manual: Admin selects assignee from ticket detail page.

Auto-assignment (when `enable_auto_assign` is enabled in `mmhd_advanced_settings`):

- Least-Loaded Agent: Counts open + in-progress tickets per staff member, assigns to fewest
- Default Agent: If `mmhd_default_agent` is set, assigns to that agent (when `strict_default_only` flag is true)

Eligible roles: `mmhd_support_staff`, `mmhd_support_manager`, `administrator`, `author`, `editor`.

17.6 Deeplinks

Encrypted URLs for secure ticket access:

`?mmhd_popup=true&initiator=mmhd&rtk=<encrypted>`. Ticket ID encrypted with AES-256-CBC using derived from `MMHD_ENCRYPTION_KEY` constant or `wp_hash("mmhd-encryption-key")`.

17.7 Source Context

Every ticket/reply captures:

- IP address: Cloudflare CF-Connecting-IP or X-Forwarded-For or `REMOTE_ADDR`
- Country: via ipapi.co API (24-hour cache in post meta)
- Browser/Agent: `HTTP_USER_AGENT` or email client label

17.8 Ticket Numbering

REST responses include `ticket_number` in format `TCKT-XXX` (zero-padded to 3 digits). Controlled by `rest_prepare_mmhd_ticket` filter.

18. Credentials Vault

Custom table `wp_mmhd_credentials`. Passwords AES-256-CBC encrypted via `mmhd_encrypt()/mmhd_decrypt()`. Per-user isolation enforced by `user_id` on all operations.

AJAX operations: Add credential (encrypt password), reveal credential (decrypt password), delete credential (ownership-checked).

19. Addon Integration Points

19.1 Primary Hooks

mmhd_after_reply_added (action)

```
add_action('mmhd_after_reply_added', function ($ticket_id, $comment_id, $reply,
$user_object) {
    // Send to Slack, Discord, WhatsApp, etc.
}, 10, 4);
```

mmhd_settings_tabs (filter)

```
add_filter('mmhd_settings_tabs', function ($tabs) {
    $tabs['my-addon'] = __('My Addon', 'metamint-helpdesk');
    return $tabs;
});
```

mmhd_settings_content (action)

```
add_action('mmhd_settings_content', function () {
    $current_tab = $_GET['tab'] ?? '';
    if ($current_tab === 'my-addon') {
        include MY_ADDON_PATH . 'views/settings.php';
    }
});
```

mmhd_before_create_ticket_from_email (filter)

```
add_filter('mmhd_before_create_ticket_from_email', function ($proceed, $email_data) {
    if (strpos($email_data['from'], '@spam.com') !== false) return false;
    return $proceed;
}, 10, 2);
```

mmhd_frontend_dashboard_menu_items (filter)

```
add_filter('mmhd_frontend_dashboard_menu_items', function ($menu_items) {
    $menu_items['my-custom-page'] = 'My Custom Page';
    return $menu_items;
});
```

19.2 Admin Reply Editor Hooks

```
add_action('mmhd_before_reply_editor', function () {  
    // Render content before reply textarea  
});  
add_action('mmhd_after_reply_editor', function () {  
    // Render content after reply textarea  
});
```

19.3 Addon Features (separate plugins)

The codebase references these addon integrations (feature detection via constant/option checks):

- Slack integration
- Discord integration
- WhatsApp / Phone origin
- AI bot (mmhd_addon_ai_bot_status option, mmhd-bot/v1 API namespace)
- Canned replies
- Agent signatures
- Email filtering

20. Security Model

20.1 Authentication

- WordPress native cookie sessions (wp_signon/wp_set_auth_cookie)
- JWT generated but not server-verified at REST layer (token is informational for client-side use)
- REST nonce required for SSE (passed as ?_wpnonce query param)
- Rate-limited signup (5 attempts / 2 hours) and password reset (5 attempts / 1 hour), with optional permanent IP blocking after repeated abuse

20.2 Access Control

- Tickets: mmhd_user_can_access_ticket() — own tickets for customers, all for staff/dashboard users, all for deeplink
- Credentials: Ownership by user_id
- Admin: mmhd_access_dashboard capability on all admin pages/SSE routes
- SSE: Permission callbacks on all routes (mmhd_can_view_ticket, is_user_logged_in, current_user_can)
- Webhook: HMAC-SHA256 signature verification X-Webhook-Signature header against mmhd_get_webhook_secret() (MMHD_WEBHOOK_SECRET constant or mmhd_webhook_secret option)

20.3 Encryption

- Passwords: AES-256-CBC via OpenSSL (openssl_encrypt/openssl_decrypt)
- Key source: MMHD_ENCRYPTION_KEY constant if defined, else wp_hash('mmhd-encryption-key'). Decrypt-side legacy fallbacks: hardcoded 'mmhd' key (old deeplinks), then AUTH_KEY.

20.4 IP Blocking

Permanently blocked IPs in `mmhd_blocked_ips` option. Auto-blocking after repeat signup abuse: first 5 attempts in 2h triggers a 24-hour strike; a second violation within those 24h adds the IP to the permanent blacklist. Login blocked IP check.

20.5 Rate Limiting

Endpoint	Limit	Window
Signup	5 attempts	2 hours (second violation within 24h → permanent ban)
Password reset	5 attempts	1 hour
Email check	10 attempts	1 hour

21. Frontend Architecture

21.1 Build Tooling

- Bundler: @wordpress/scripts (wp-scripts)
- Entry: src/index.js in the development repository (compiled into public/build/index.js) → renders `<App>` into `<div id="mmhd-plugin">`
- Output: public/build/index.js + public/build/index.css + public/build/index-rtl.css + public/build/index.asset.php
- Enqueue: PHP handle mmhd-react-script with deps ['jquery', ...\$asset_file['dependencies']] in footer
- Webpack: extends @wordpress/scripts default config with babel-loader using @babel/preset-react (runtime: automatic) + @babel/preset-env

21.2 Component Tree



21.3 State Management

`AppDataProvider` context (`utils/AppDataContext.jsx`) provides:

State fields: `loadingContext`, `errorContext`, `loggedinContext`, `loggedInUserId`, `ticketsContext[]`, `ticketPagination`, `noticesContext[]`, `isSingleTicketPage`, `openTicketId`, `isLoadingMore`.

Actions: `updateLoginStatus()`, `refreshTickets()` (with search/priority/status filters), `updateTicketNotifications()`, `updateTicket()`, `updateNewCommentNotification()`, `setSingleTicketPage()`, `setOpenTicketId()`, `loadMoreTickets()`.

On mount: fetches notices (`GET /wp/v2/notice`), checks login status from `sessionStorage['wpUserData']`.

When user logs in: fetches tickets (GET `/wp/v2/mmhd_ticket?author=<id>&page=1&per_page=20&orderby=date&order=desc`). The `per_page=5` value is used only by `refreshTickets()` (triggered by manual filter changes).

21.4 API Helpers (utils/modules.js)

Function	Method	Endpoint	Description
<code>createTicket(userId, title, message, timestamp)</code>	POST	<code>/wp-json/mmhd/chat/v1/tickets</code>	Create ticket
<code>getMessages(ticketId, userId)</code>	POST	<code>/wp-json/mmhd/chat/v1/tickets/</code> (trailing slash)	Get ticket comments
<code>sendReply(ticketId, loggedInUserId, message, img, isBotResponse, signal, timestamp)</code>	POST	<code>/wp-json/mmhd/chat/v1/tickets</code>	Send reply (with AbortController). 2nd param is loggedInUserId.
<code>registerUser(email)</code>	POST	<code>/wp-json/mmhd/v1/signup</code>	Register
<code>loginUser(email, password)</code>	POST	<code>/wp-json/mmhd/v1/login</code>	Login
<code>getAllTickets(userId)</code>	POST	<code>/wp-json/mmhd/chat/v1/tickets</code>	List tickets

Other REST calls used directly: GET `/wp/v2/comments` (polling fallback), GET `/wp/v2/notice`, GET `/wp/v2/mmhd_ticket/{id}` (deep links).

21.5 Key Behaviors

- Optimistic sends: Replies appear immediately as temp messages (`isOptimistic: true`), replaced with server data on API response. Failures show `isPending: true` state after 15s timeout.
- Polling fallback: `setInterval(pollLatestComments, 4000)` — GET comments every 4s as backup for SSE
- SSE: Connected to ticket-specific stream when viewing a ticket; general stream for ticket list
- Nonce: `window.mmhdAuth.restNonce` (injected by PHP via `wp_create_nonce('wp_rest')`), with fallback to `window.wpApiSettings.nonce`. Used for SSE `?_wpnonce` parameter. After AJAX login, a fresh nonce is returned in the login response and stored into `wpAuth.restNonce`.
- Widget toggle: `.mmhd-toggle-btn` opens/closes popup. Auto-opens on deep link parameters (`?mmhd_popup=true, ?ticket_id=, ?tid=, ?ticket=`)
- AI bot: Separate panel in Home tab (Metamint HelpdeskBotChatWidgetMain), gated by `mmhd_addon_active` & `mmhd_ai_bot_active` flags

21.6 Notification Sound

`playMetamint HelpdeskBeep()` (`utils/notificationSound.js`):

- Reads sound URL from `window.mmhdAuth.notificationSound`
- Reuses a single Audio element across calls
- Debounces at 1200ms
- Handles browser autoplay policy via gesture-detection unlock
- Triggers: `new_notification` SSE event (App.js), `new_comment` SSE event (SingleTicket.jsx), polling fallback new messages

21.7 Admin Dashboard

The admin dashboard (`mmhd-dashboard.js`) is a **vanilla JS SPA** (not React):

- Ticket list + detail views with SSE connection to `ticket-stream/{ticket_id}` (full sync events)
- SSE connection to `notification-stream` for real-time notification badge updates
- Admin SSE stream `adminstream` for customer reply notification to agents
- Polling fallback for replies + notifications
- All interactions via `wp_ajax_*` handlers

22. Extending the Plugin

22.1 Creating an Addon

- 1. Create a separate plugin with its own bootstrap
- 2. Check for Metamint Helpdesk core: `defined('METAMINT_HELPDESK_PLUGIN_PATH')`
- 3. Use hooks from Section 15 and Section 19
- 4. Add settings tabs via `mmhd_settings_tabs` filter + `mmhd_settings_content` action
- 5. Hook into `mmhd_after_reply_added` for real-time integrations

22.2 Common Extension Pattern

```
if (!defined('METAMINT_HELPDESK_PLUGIN_PATH')) return;
add_action('mmhd_after_reply_added', 'my_addon_handle_reply', 10, 4);
function my_addon_handle_reply($ticket_id, $comment_id, $reply, $user_object) {
    // Send to Slack, Discord, WhatsApp, etc.
}
add_filter('mmhd_settings_tabs', function ($tabs) {
    $tabs['my-addon'] = __('My Addon', 'metamint-helpdesk');
    return $tabs;
});
add_action('mmhd_settings_content', function () {
    $current_tab = $_GET['tab'] ?? '';
    if ($current_tab === 'my-addon') {
        // Render settings
    }
});
add_filter('mmhd_frontend_dashboard_menu_items', function ($menu_items) {
    $menu_items['my-page'] = 'My Custom Page';
    return $menu_items;
});
```

22.3 Programmatic Ticket Creation

```
$ticket_id = wp_insert_post([
    'post_type' => 'mmhd_ticket',
    'post_title' => 'Support Request',
    'post_status' => 'publish',
    'post_author' => $user_id,
    'meta_input' => [
        '_mmhd_ticket_status' => 'open',
        '_mmhd_ticket_origin' => 'web',
        '_mmhd_customer_email' => $customer_email,
    ],
]);
wp_set_object_terms($ticket_id, 'High', 'mmhd_ticket_priority');
wp_set_object_terms($ticket_id, 'General', 'mmhd_ticket_category');
$comment_id = wp_insert_comment([
```

```
'comment_post_ID' => $ticket_id,
'user_id'         => $user_id,
'comment_content' => 'Initial message',
'comment_author'  => $user_name,
'comment_approved' => 1,
'comment_type'    => 'ticket_replies',
]);
add_comment_meta($comment_id, 'comment_source', 'web', true);
add_comment_meta($comment_id, 'author_type', 'customer', true);
add_comment_meta($comment_id, 'admin_comment', 'no', true);
mmhd_notify_ticket_created($ticket_id, 'web');
```

22.4 Programmatic Ticket Query (REST)

The `rest_mmhd_ticket_query` filter supports custom query params:

- `search` — keyword search
- `ticket_priority` — filter by priority term slug
- `ticket_status` — filter by `_mmhd_ticket_status` meta value