Technical Solution Design - Zapdesk

KnowAll AI - Zendesk Lightning Network Tipping Application

Table of Contents

General Terms & Conditions

This technical solution design document is provided by KnowAll AI for the Zapdesk Lightning Network Tipping Application project under the terms of the project agreement. The information contained herein is proprietary and confidential.

Scope of Document: This document covers the technical implementation details, architecture, and system specifications for the Zapdesk ZAF v2 sidebar application as deployed in Zendesk environments.

Limitations: This document reflects the system state as of the date of creation. Any subsequent modifications, enhancements, or integrations may require updates to this documentation.

Support and Maintenance: For technical support, system modifications, or clarifications regarding this document, please contact the KnowAll AI development team through established support channels.

Intellectual Property: All solution components, custom code, and technical implementations described in this document remain the intellectual property of KnowAll AI unless otherwise specified in the project agreement.

Confidentiality Statement

Document History

Version	Date	Author	Changes
1.0	2025-10-22	KnowAll AI	Initial solution design document created for Zapdesk Lightning Network tipping application
1.1	2025-10-22	Akash Jadhav	Updated document structure to follow professional technical design standards with enhanced formatting and organization

1. Overview

Zapdesk is a ZAF sidebar app enabling users to tip agents over the Bitcoin Lightning Network. The MVP is client-only, wallet-agnostic, and supports:

- **Lightning Network Integration**: Send tips using Lightning addresses configured in agent profiles
- **QR Code Generation**: Display QR codes for easy mobile wallet scanning
- **Custom Tip Amounts**: Support for predefined and custom tip amounts
- **Configurable Comment Visibility**: Admin setting to control whether tip comments are public or private
- **LNURL Support**: Full LNURL protocol implementation for Lightning payments
- **Service Layer Architecture**: Clean separation of concerns with dedicated services for Zendesk and Lightning operations

2. Architecture

Component	Description
ZAF React App	Packaged (iframe) React + TypeScript + Vite build to `/assets`. Uses ZAF SDK (`ZAFClient`) to read context and post to the ticket.
Agent Mapping	Resolve agent Lightning Address from a Zendesk **user custom field** (configurable key). Optional fallback address.
Payment – QR / LNURL	Render QR + copyable string for the selected preset amount using LNURL-pay or BOLT11 as provided/configured.
End-User Message	Short text input bound to the payment memo (NWC). Always appended to the ticket post after success.
Ticket Posting	After success, app posts a message to the ticket via ZAF (visibility = `public` or `internal`, from settings). Contains amount, agent target, and the user message.

3. Key Flows

3.1. Tip via QR / LNURL-pay

1. App loads; determines target agent and payout address.
2. User enters an optional message; selects a preset.
3. App displays QR + copyable payment string.
4. User pays with their wallet.
5. User clicks “I’ve paid” (manual acknowledge) → app posts to ticket with the message.

4. UI

- Header with branding.
- Agent context (avatar/name) + payout destination (masked).
- Message textarea (limit, e.g., 140 chars).
- Preset buttons (100 / 1,000 / 10,000 sats; configurable).
- Actions: **Tip by QR**, **Connect wallet (NWC)** / **Tip now**.
- Status banners; copy helpers; balance indicator (NWC).

5. ZAF Integration

- `client.get(['ticket.id', 'ticket.assignee', 'currentUser'])`
- Post to ticket:
  - Public reply append: `client.invoke('comment.appendText', msg)` then agent/end-user sends
  - or direct API call via `client.request()` to create a comment (visibility from settings)
- Settings read via `client.metadata()` / app settings API.

6. Settings

- `presets`: hardcoded sats amounts (e.g., `100,1000,10000`)
- `agentLightningFieldKey`: string (e.g., `user.custom_fields.lightning_address`)
- `fallbackLightningAddress`: string (optional)
- `postVisibility`: enum `public` | `internal`
- `brandingTitle` / `brandingDescription` (optional)

7. Security

- All code runs inside the ZAF iframe sandbox.
- No custody; payments occur in the end-user’s wallet.
- Respect CSP (Content Security Policy); avoid unnecessary external scripts.

8. Build & Packaging

- React + Vite + TypeScript → `/assets`: `iframe.html`, `app.js`, `app.css`
- `manifest.json` points `support.ticket_sidebar` → `assets/iframe.html`
- `zcli apps:package` to produce distributable zip

9. Testing

9.1. Unit Testing

React Component Testing:

- Component rendering validation using React Testing Library
- Props handling and state management testing
- Event handler and callback function testing
- Custom hooks testing (useZafClient, useNWC)
- Error boundary component testing

TypeScript Type Validation:

- Interface and type definition validation
- Type safety verification across components
- Generic type parameter testing
- Enum and constant value testing

Service Layer Testing:

- Lightning Address validation logic
- Payment string generation (LNURL-pay, BOLT11)
- QR code generation accuracy
- NWC SDK integration testing
- Ticket posting service validation

9.2. Integration Testing

ZAF Client Integration:

- ZAF Client initialization and connection
- Context data retrieval (ticket, assignee, user)
- Ticket comment posting validation
- Settings and metadata retrieval
- Error handling for ZAF operations

Lightning Network Integration:

- LNURL-pay invoice generation
- QR code rendering and accuracy
- NWC wallet connection and pairing

- Payment execution via NWC
- Balance retrieval from NWC wallets

External Services:

- Agent Lightning Address resolution
- Payment string validation
- QR code library integration
- Error handling for network failures

9.3. End-to-End Testing

Complete Tipping Workflows:

- QR/LNURL-pay tipping flow (manual acknowledgment)
- NWC wallet connection and payment flow
- End-user message inclusion in tips
- Ticket posting with public/internal visibility
- Balance display and refresh (NWC)

Error Scenarios:

- Invalid Lightning Address handling
- Network connectivity failures
- Payment execution failures
- Ticket posting errors
- Wallet disconnection scenarios

User Experience Testing:

- UI responsiveness and load times
- Form validation and error messages
- Accessibility compliance (WCAG 2.1)
- Cross-browser compatibility (Chrome, Firefox, Edge, Safari)
- Mobile device compatibility

9.4. Performance Testing

Performance Criteria:

- Initial app load time < 2 seconds
- QR code generation < 500ms

- NWC payment execution < 5 seconds
- Ticket posting < 3 seconds
- UI interaction responsiveness < 200ms

Load Testing:

- Multiple concurrent user sessions
- Large message text handling
- Rapid preset selection changes
- Stress testing payment operations

9.5. Security Testing

Data Security:

- NWC connection string secure handling
- No logging of sensitive payment data
- CSP compliance validation
- XSS vulnerability testing
- Input sanitization verification

ZAF Sandbox Security:

- Iframe sandbox restrictions validation
- No unauthorized API calls
- Proper permission handling
- Secure data transmission

9.6. User Acceptance Testing

Test Scenarios:

- Agent receives tip via QR/LNURL-pay
- Agent receives tip via NWC
- End-user message appears in ticket
- Public vs internal comment visibility
- Balance display updates correctly
- Error messages are clear and actionable

Acceptance Criteria:

- All payment methods work reliably

- Ticket posts are accurate and complete
- UI is intuitive and easy to use
- Error handling provides clear guidance
- Performance meets defined criteria

10. Deployments

10.1. Build Process

Development Build:

```
# Install dependencies
npm install

# Run development server with hot reload
npm run dev

# Build for production
npm run build
```

Production Build:

```
# Clean previous builds
npm run clean

# Build optimized production assets
npm run build

# Package Zendesk app
zcli apps:package
```

Build Output:

- `/dist/assets/iframe.html` - Main application entry point
- `/dist/assets/app.js` - Bundled JavaScript (React + TypeScript)
- `/dist/assets/app.css` - Compiled CSS styles
- `/dist/manifest.json` - Zendesk app manifest
- `/dist/translations/` - Localization files
- `zapdesk-{version}.zip` - Packaged app for upload

10.2. Environment Deployment

Zendesk Sandbox Environment:

1. Build the application with `npm run build`
2. Package the app with `zcli apps:package`
3. Upload to Zendesk Sandbox: `bash zcli apps:create dist/`
4. Test in Zendesk sandbox ticket sidebar

5. Verify ZAF client integration
6. Test all payment workflows

Zendesk Production Environment:

1. Complete UAT testing in Sandbox
2. Obtain deployment approval from stakeholders
3. Schedule deployment window (low-traffic hours recommended)
4. Create backup of current production app version
5. Upload new version to production: `bash zcli apps:update --path dist/`
6. Verify deployment success
7. Perform smoke testing in production
8. Monitor for errors in first 24 hours
9. Notify end-users of new version

10.3. Configuration Management

App Settings (manifest.json):

```
{
  "parameters": [
    {
      "name": "presets",
      "type": "text",
      "default": "100,1000,10000",
      "required": false
    },
    {
      "name": "showQrMode",
      "type": "checkbox",
      "default": true
    },
    {
      "name": "showNwcMode",
      "type": "checkbox",
      "default": true
    },
    {
      "name": "agentLightningFieldKey",
      "type": "text",
      "default": "user.custom_fields.lightning_address"
    },
    {
      "name": "fallbackLightningAddress",
      "type": "text",
      "required": false
    },
    {
      "name": "postVisibility",
      "type": "dropdown",
      "default": "public"
    }
  ]
}
```

Deployment Checklist:

- [ ] Code review completed and approved
- [ ] All unit tests passing
- [ ] Integration tests successful
- [ ] Build artifacts generated successfully
- [ ] App packaged with correct version number
- [ ] Settings configured for target environment
- [ ] Lightning Address field configured in Zendesk
- [ ] Zendesk custom field created (if required)
- [ ] Backup of current version created
- [ ] Rollback plan documented
- [ ] Deployment window communicated to users
- [ ] Post-deployment testing plan ready

10.4. Version Management

Versioning Strategy:

- **Major version (X.0.0)**: Breaking changes, major feature releases
- **Minor version (0.X.0)**: New features, non-breaking changes
- **Patch version (0.0.X)**: Bug fixes, minor updates

Changelog Maintenance:

- Document all changes in `CHANGELOG.md`
- Include migration notes for breaking changes
- Reference GitHub issues/pull requests
- Provide upgrade instructions

Git Tagging:

```
# Tag release version
```

```
git tag -a v1.0.0 -m "Release version 1.0.0"

# Push tags to remote
git push origin --tags
```

10.5. Monitoring and Maintenance

Application Health Monitoring:

- Monitor ZAF client errors via browser console
- Track payment success/failure rates
- Monitor NWC connection reliability
- Track ticket posting errors
- Monitor app load times and performance

User Feedback Collection:

- Monitor Zendesk app reviews and ratings
- Collect user feedback via support channels
- Track feature requests and enhancement ideas
- Analyze usage patterns and adoption metrics

Maintenance Schedule:

- **Weekly**: Review error logs and user feedback
- **Monthly**: Dependency updates and security patches
- **Quarterly**: Feature releases and major updates
- **As needed**: Critical bug fixes and security updates

10.6. Rollback Procedures

Emergency Rollback:

1. Identify critical issue requiring rollback
2. Notify stakeholders of rollback decision
3. Restore previous version from backup: `bash zcli apps:update --path backup/zapdesk-{previous-version}.zip`
4. Verify rollback successful
5. Communicate rollback to users
6. Investigate root cause of issue
7. Plan hotfix or corrected deployment

Rollback Validation:

- Test core functionality after rollback
- Verify settings are preserved
- Check user data integrity
- Monitor for additional errors

Appendix A: Troubleshooting Guide

Common issues identified during development, testing, and deployment. Most of these have been resolved, but may recur in similar scenarios.

Problem	Solution
App fails to load in Zendesk sidebar	Verify app is properly installed and enabled. Check ZAF client initialization in browser console. Ensure manifest.json location settings are correct. Try refreshing the ticket page.
Agent Lightning Address not displaying	Check that the custom field key in settings matches Zendesk user field. Verify agent has Lightning Address populated in their user profile. Check fallback address configuration.
QR code not generating	Verify Lightning Address is valid (format: user@domain.com). Check QR code library is loaded correctly. Inspect browser console for errors. Ensure payment string is properly formatted.
Ticket comment not posting after payment	Check ZAF client permissions for ticket.comment operations. Verify comment text is properly formatted. Ensure postVisibility setting is correctly configured. Check network errors in browser console.
End-user message not included in ticket post	Verify message textarea value is captured correctly. Check that message is properly appended to ticket comment. Ensure character limit validation doesn't truncate message.
Preset amounts not displaying	Check presets setting in app configuration (CSV format: 100,1000,10000). Verify preset parsing logic handles values correctly. Ensure UI components are rendering preset buttons.
Error: "Invalid Lightning Address format"	Lightning Address must follow format: user@domain.com. Check for extra spaces or special characters. Verify domain has proper Lightning Address DNS records. Test address with external validator.
App styling appears broken	Verify app.css is loaded correctly. Check for CSP violations blocking stylesheets. Clear browser cache and reload. Ensure Zendesk Garden CSS doesn't conflict.
ZAF Client context data returns undefined	Check that context data is available in current ticket view. Verify ZAF client is fully initialized before accessing data. Add proper error handling for undefined context.
Multiple rapid clicks cause duplicate actions	Implement button disable state during async operations. Add debouncing to click handlers. Show loading indicators during processing. Prevent multiple simultaneous requests.
App performance slow in Zendesk	Optimize React re-renders using useMemo and useCallback. Implement lazy loading for heavy components. Minimize bundle size by code splitting. Check for memory leaks in useEffect hooks.

Problem	Solution
Settings not persisting between sessions	Verify settings are accessed via ZAF client metadata API. Check that settings are saved in Zendesk app configuration. Ensure app parameters in manifest.json are correct.
Cross-origin errors in browser console	Verify all external API calls comply with CSP. Check CORS headers for external services. Ensure ZAF client is used for all Zendesk API calls. Review manifest.json domainWhitelist.
Browser console shows TypeScript errors	Verify TypeScript compilation is successful. Check tsconfig.json configuration. Ensure all dependencies have proper type definitions. Run `tsc --noEmit` to check for type errors.
Vite build fails with dependency errors	Clear node_modules and reinstall: `rm -rf node_modules package-lock.json && npm install`. Check for version conflicts in package.json. Verify Node.js version compatibility. Update dependencies if needed.
Lightning Address resolution takes too long	Implement timeout for address resolution (e.g., 5 seconds). Show loading indicator during resolution. Cache resolved addresses for better performance. Handle timeout errors gracefully.
Tip amount validation not working	Verify input validation logic for numeric values. Check for edge cases (negative, zero, decimal values). Implement proper error messages for invalid amounts. Test with various input formats.
Mobile device layout issues	Ensure responsive CSS is applied correctly. Test on various screen sizes and orientations. Use Zendesk Garden responsive utilities. Implement mobile-first design approach.
Accessibility issues (keyboard navigation)	Ensure all interactive elements are keyboard accessible. Implement proper ARIA labels and roles. Test with screen readers. Follow WCAG 2.1 accessibility guidelines.
App crashes when switching between tickets	Implement proper cleanup in useEffect hooks. Reset state when component unmounts. Handle ZAF context changes gracefully. Add error boundaries to prevent full app crashes.
Environment variables not loaded in production	Check Vite environment variable naming (VITE_ prefix). Verify .env files are properly configured. Ensure environment variables are set in build process. Don't commit .env files to version control.
ZCLI package command fails	Verify zcli is installed: `npm install -g @zendesk/zcli`. Check manifest.json syntax is valid. Ensure all required assets are in dist/ folder. Run `zcli apps:validate` to check for issues.
PDF documentation not rendering diagrams	Install required dependencies: asciidoctor-pdf, asciidoctor-diagram. Check Mermaid diagram syntax is valid. Verify asciidoctor-diagram extension is loaded. For Ubuntu 23.10+, disable AppArmor restrictions: `sudo sysctl -w kernel.apparmor_restrict_unprivileged_userns=0`.

Appendix B: Frequently Asked Questions

This document provides answers to frequently asked questions about the Zapdesk Lightning Network Tipping Application. It is organized by stakeholder groups and common scenarios to help users quickly find the information they need.

Target Audiences:

- Support agents receiving tips
- Zendesk administrators
- System administrators and IT support
- Project stakeholders and management

B.1. General Usage

B.1.1. Q: What is Zapdesk and how does it work?

A: Zapdesk is a Zendesk sidebar application that enables ticket requesters (customers) to tip support agents using Bitcoin's Lightning Network:

Key Features:

1. **QR/LNURL-pay**: Display QR code for manual wallet payments
2. **Custom Messages**: Users can include personalized thank-you messages
3. **Ticket Integration**: Tips are recorded in ticket comments (public or internal)
4. **Balance Display**: View wallet balance when using NWC (if supported by wallet)

How It Works:

1. User opens support ticket in Zendesk
2. Zapdesk app appears in ticket sidebar
3. Agent's Lightning Address is automatically resolved
4. User selects tip amount from presets (e.g., 100, 1,000, 10,000 sats)
5. User can add optional message
6. Payment via QR code scan or NWC direct payment
7. Confirmation posted to ticket as comment

B.1.2. Q: What is the Lightning Network?

A: The Lightning Network is a "layer 2" payment protocol built on top of Bitcoin:

Key Characteristics:

- **Instant Payments**: Transactions settle in seconds, not minutes/hours

- **Low Fees**: Minimal transaction costs (often less than 1 cent)
- **Scalability**: Handles millions of transactions per second
- **Micropayments**: Enables tiny payments (even sub-cent amounts)
- **Bitcoin-Native**: Secured by Bitcoin blockchain

Why Lightning for Tipping:

- Traditional payment systems have high fees that make small tips uneconomical
- Credit cards charge 2-3% + fixed fees, making $1-5 tips impractical
- Lightning enables true micropayments with negligible fees
- No intermediaries or payment processors required
- Global and permissionless - works anywhere

B.1.3. Q: What is a Lightning Address?

A: A Lightning Address is a human-readable identifier for receiving Lightning payments, similar to an email address:

Format: `username@domain.com`

Examples: - `alice@getalby.com` - `bob@walletofsatoshi.com` - `support@company.com`

How It Works:

1. User enters Lightning Address in their Zendesk profile
2. Zapdesk resolves the address to payment endpoint (LNURL-pay)
3. Generates payment invoice for selected amount
4. Creates QR code for scanning or direct NWC payment

Setting Up Lightning Address:

- Most Lightning wallets provide free Lightning Addresses
- Popular providers: Alby, Wallet of Satoshi, Strike, Cash App
- Can also self-host using LNbits or custom LNURL server

B.1.4. Q: How do I tip an agent using QR code?

A: QR code tipping process:

Prerequisites:

- Lightning-compatible wallet app (Alby, Wallet of Satoshi, Phoenix, Breez, etc.)
- Small amount of Bitcoin/sats in wallet

Steps:

1. Open ticket in Zendesk where agent has responded
2. Zapdesk app appears in ticket sidebar showing agent info
3. Select tip amount from preset buttons (e.g., 1,000 sats)
4. Optionally add personal thank-you message (e.g., "Thanks for the great support!")
5. Click **"Tip by QR"** button
6. QR code displays with payment string
7. Open your Lightning wallet app
8. Scan QR code with wallet camera
9. Confirm payment in wallet app
10. Click **"I’ve Paid"** button in Zapdesk
11. Confirmation message posts to ticket

Tip: You can also copy the payment string and paste it into your wallet manually.

B.1.5. Q: How do I tip using Nostr Wallet Connect (NWC)?

A: NWC enables direct programmatic payments from your wallet:

Prerequisites:

- NWC-compatible wallet (Alby, Mutiny, Coinos, etc.)
- NWC connection string from your wallet

First-Time Setup:

1. Open your Lightning wallet settings
2. Find "Nostr Wallet Connect" or "NWC" section
3. Generate NWC connection string (looks like `nostr+walletconnect://…`)
4. Copy connection string

Tipping with NWC:

1. Open ticket in Zendesk
2. Click **"Connect Wallet (NWC)"** in Zapdesk sidebar
3. Paste your NWC connection string
4. Click **"Connect"**
5. Wallet balance displays (if supported by wallet)
6. Select tip amount from presets
7. Add optional message
8. Click **"Tip Now"** button
9. Payment executes automatically

10. Balance updates and confirmation posts to ticket

Benefits:

- No scanning QR codes
- Faster payment experience
- See wallet balance in app
- One-click tipping after initial setup

B.1.6. Q: Is my payment information secure?

A: Yes, Zapdesk follows security best practices:

Security Measures:

1. **No Custody**: Zapdesk never holds your Bitcoin or has access to your wallet
2. **NWC Secrets**: Connection strings are stored in session storage only (not logged or transmitted to servers)
3. **ZAF Sandbox**: App runs in Zendesk's isolated iframe sandbox
4. **No Backend**: Client-only app with no server to be compromised
5. **CSP Compliance**: Strict Content Security Policy prevents injection attacks
6. **HTTPS Only**: All communications encrypted via HTTPS

What Zapdesk CAN Access:

- Ticket context (ticket ID, agent name, user info)
- Agent's Lightning Address (from public user profile)
- Payment amounts and messages you enter

What Zapdesk CANNOT Access:

- Your wallet private keys or seed phrase
- Your full wallet balance (unless explicitly exposed via NWC getBalance)
- Payment details beyond what you explicitly authorize
- Your NWC connection string after session ends

B.1.7. Q: What happens to my tip after I send it?

A: The tip flow after payment:

Immediate Actions:

1. **Payment Execution**: Sats transferred from your wallet to agent's Lightning Address
2. **Confirmation**: Payment success/failure confirmation displayed in app
3. **Ticket Comment**: App posts confirmation to ticket with:

    - Tip amount (e.g., "1,000 sats")
    - Your personal message (if provided)
    - Agent information
    - Timestamp
4. **Balance Update**: If using NWC, wallet balance refreshes

Comment Visibility:

- **Public Comment**: Visible to ticket requester and agent (default)
- **Internal Note**: Only visible to support staff (if configured)

Agent Receives:

- Bitcoin/sats in their Lightning wallet immediately
- Notification from their wallet app
- Ticket comment showing appreciation

No Reversals:

- Lightning payments are final and cannot be reversed
- Ensure correct amount and agent before sending
- Contact agent directly if payment sent in error

B.2. Wallet Setup

B.2.1. Q: What Lightning wallet should I use?

A: Recommended wallets by experience level:

Beginners (Custodial - Easy Setup):

Wallet	Description	Best For
Wallet of Satoshi	Simplest Lightning wallet, iOS/Android	First-time users, casual tippers
Alby	Browser extension + mobile, great NWC support	Browser-based tipping, web users
Strike	Fiat on-ramp, easy Bitcoin purchase	Users wanting to buy Bitcoin easily
Cash App	Popular payment app with Bitcoin support	US users already on Cash App

Intermediate (Hybrid - Balance of Ease & Control):

Wallet	Description	Best For
Phoenix	Self-custodial, automated channel management	Users wanting more control
Breez	Self-custodial, full Lightning node in pocket	Technical users, privacy-focused
Blink (Bitcoin Beach)	Community-focused, good for regions	Community tipping, international

Advanced (Self-Custodial - Full Control):

Wallet	Description	Best For
LNbits	Self-hosted wallet server	Power users, developers
Umbrel	Full Bitcoin/Lightning node	Technical users, maximum sovereignty
Voltage	Cloud-hosted Lightning node	Users wanting self-custody without hardware

NWC Support:

- □ Full NWC: Alby, Mutiny, Coinos, LNbits
- □□ Limited: Some wallets may not support balance queries
- □ No NWC: Wallet of Satoshi, Phoenix, Breez (use QR code instead)

B.2.2. Q: How do I get Bitcoin to tip with?

A: Several ways to acquire Bitcoin for tipping:

Buy Bitcoin Directly:

1. **Strike**: Buy Bitcoin with bank account or debit card (US, EU, select countries)
2. **Cash App**: Add money, convert to Bitcoin (US only)
3. **Swan Bitcoin**: Recurring Bitcoin purchases
4. **River Financial**: Buy and hold Bitcoin, automatic withdrawals to Lightning

Receive from Others:

- Have someone send you sats to your Lightning Address
- Earn Bitcoin through work, freelancing, or content creation
- Participate in Bitcoin community giveaways ("sats giveaways")

Bitcoin ATMs:

- Find Bitcoin ATM near you (CoinATMRadar.com)
- Buy Bitcoin, send to your wallet address

- May have higher fees than online purchases

Starting Small:

- Many wallets allow deposits as low as $1-5
- 10,000 sats ≈ $2-5 (varies with Bitcoin price)
- Start small to learn, increase as comfortable

Price Awareness:

- 1 Bitcoin = 100,000,000 sats
- Sats are convenient for small payments
- Example: 1,000 sats ≈ $0.20-$0.50 (price varies)

B.2.3. Q: How do I set up my Lightning wallet for NWC?

A: NWC setup varies by wallet:

Alby (Recommended for NWC):

1. Install Alby browser extension or mobile app
2. Create/import wallet
3. Go to Settings → Wallet Connect
4. Click "Create Connection"
5. Set spending limits if desired
6. Copy connection string (starts with `nostr+walletconnect://`)
7. Paste into Zapdesk when prompted

Mutiny Wallet:

1. Open Mutiny wallet (https://app.mutinywallet.com)
2. Navigate to Settings → Connections
3. Create new NWC connection
4. Configure permissions (payment execution, balance read)
5. Copy connection string
6. Use in Zapdesk

LNbits (Self-Hosted):

1. Access your LNbits instance
2. Install Nostr Wallet Connect extension
3. Create new NWC connection
4. Set budget and expiry

5. Copy connection string
6. Use in Zapdesk

Connection String Security:

- Treat NWC connection strings like passwords
- Set spending limits to minimize risk
- Use separate connection for tipping vs. main wallet
- Revoke connections when no longer needed
- Don't share connection strings publicly

B.3. Agent Configuration

B.3.1. Q: How do agents set up their Lightning Address?

A: Agents configure Lightning Address in Zendesk user profile:

Prerequisites:

1. Get Lightning wallet that provides Lightning Address
2. Note your Lightning Address (e.g., `agent@getalby.com`)
3. Test receive payment to verify address works

Configuration in Zendesk:

1. Zendesk admin creates custom user field:
    - Field Type: Text
    - Field Key: `lightning_address` (or as configured in Zapdesk settings)
    - Visibility: Public or agents only
2. Agents update their user profile:
    - Go to profile settings
    - Find Lightning Address custom field
    - Enter Lightning Address (e.g., `username@domain.com`)
    - Save changes

Verification:

1. Open test ticket assigned to agent
2. Check Zapdesk sidebar shows agent's Lightning Address
3. Send small test tip (e.g., 10 sats) to verify
4. Confirm agent receives payment in wallet

Fallback Address:

- Zendesk admin can configure fallback Lightning Address in Zapdesk settings
- Used when agent hasn't set personal Lightning Address
- Useful for team wallets or temporary setup

B.3.2. Q: Can multiple agents share one Lightning Address?

A: Yes, but with limitations:

Shared Wallet Approach:

Pros: - Simple setup - one address for all agents - Good for small teams or testing - Centralized receiving (team wallet)

Cons: - No individual tip tracking per agent - All tips go to same wallet - Requires manual distribution to individual agents

Implementation:

1. Create team Lightning wallet (e.g., LNbits)
2. Set team Lightning Address as fallback in Zapdesk settings
3. Agents without personal address use fallback
4. Manually track and distribute tips periodically

Individual Address Approach (Recommended):

Pros: - Each agent receives tips directly - Clear attribution and tracking - Better agent motivation - Automatic distribution (no manual work)

Cons: - Requires each agent to set up wallet - More initial configuration - Agents must manage own wallets

Recommendation:

- Start with shared address for testing
- Migrate to individual addresses for production
- Provides best experience for agents and users

B.3.3. Q: What if an agent doesn't have a Lightning Address?

A: Fallback handling:

Zapdesk Behavior:

1. Checks ticket assignee's user profile for Lightning Address
2. If not found, uses fallback Lightning Address from settings
3. If no fallback configured, displays error message

Error Message Example:

```
"Lightning Address not configured for this agent.
Please contact support to set up tipping."
```

Solutions:

Option 1: Configure Fallback Address (Quick) - Admin sets fallback Lightning Address in Zapdesk settings - All tips for agents without addresses go to fallback - Good for initial deployment

Option 2: Agent Sets Personal Address (Best) - Agent creates Lightning wallet - Agent adds Lightning Address to Zendesk profile - Receives tips directly

Option 3: Disable App for Specific Agents - Zendesk admin can control app visibility - Hide Zapdesk for agents who don’t want tips - Prevents confusion for users

B.4. Administration

B.4.1. Q: How do I install Zapdesk in my Zendesk instance?

A: Installation process:

Prerequisites:

- Zendesk admin access
- Zapdesk app package (`zapdesk-{version}.zip`)
- Custom user field for Lightning Address (recommended)

Installation Steps:

1. **Create Custom Field (if not exists):**
   - Admin Center → People → Configuration → User fields
   - Add custom field: `lightning_address` (type: Text)
   - Make editable by agents
2. **Upload Zapdesk App:**
   - Admin Center → Apps and integrations → Apps → Zendesk Support apps
   - Click "Upload private app"
   - Select `zapdesk-{version}.zip`
   - Click "Upload"
3. **Configure Settings:**
   - Preset amounts (CSV): `100,1000,10000`
   - Enable QR mode: Yes
   - Enable NWC mode: Yes

    - Lightning field key: `user.custom_fields.lightning_address`
    - Fallback address: `team@company.com` (optional)
    - Comment visibility: Public or Internal
4. **Install to Ticket Sidebar:**
    - Select installation location: Ticket Sidebar
    - Choose visibility: All agents or specific groups
    - Click "Install"
5. **Test Installation:**
    - Open test ticket
    - Verify Zapdesk appears in sidebar
    - Test tipping workflow

B.4.2. Q: How do I configure Zapdesk settings?

A: Settings configuration in Zendesk Admin:

Available Settings:

Setting	Type	Description
Preset Amounts	Text (CSV)	Comma-separated sat amounts, e.g., `100,1000,10000`
Show QR Mode	Checkbox	Enable QR code / LNURL-pay tipping
Show NWC Mode	Checkbox	Enable Nostr Wallet Connect tipping
Agent Lightning Field	Text	Zendesk user field key, e.g., `user.custom_fields.lightning_address`
Fallback Address	Text	Default Lightning Address when agent doesn’t have one
Comment Visibility	Dropdown	`public` (end-user sees) or `internal` (agents only)
Branding Title	Text	Custom title in app header
Branding Description	Text	Custom description text

Modifying Settings:

1. Admin Center → Apps and integrations → Apps → Zendesk Support apps
2. Find "Zapdesk" in installed apps
3. Click gear icon → "Settings"
4. Update desired settings
5. Click "Update" to save
6. Changes take effect immediately (may require page refresh)

Best Practices:

- Start with QR mode only for initial rollout
- Add NWC mode after user education
- Use realistic preset amounts based on ticket value
- Set internal visibility initially for testing
- Configure fallback address for smoother onboarding

B.4.3. Q: How do I monitor tipping activity?

A: Monitoring options:

Zendesk Ticket Comments:

- All tips recorded as ticket comments
- Search tickets for "Lightning tip" or "sats"
- Export ticket data with comments
- Use Zendesk Explore for reporting

Custom Reporting:

- Build Zendesk Explore dashboard
- Filter comments containing tip confirmations
- Aggregate by agent, date, amount
- Track adoption metrics

Manual Tracking:

- Agents report tips received
- Team tracks in spreadsheet
- Compare wallet receipts to ticket comments

Future Enhancements:

- Centralized analytics dashboard
- Real-time tip tracking
- Agent leaderboards
- Automated reporting emails

B.4.4. Q: Can I customize the preset tip amounts?

A: Yes, presets are fully configurable:

Configuration:

1. Admin Center → Zapdesk settings
2. Find "Preset Amounts" setting
3. Enter comma-separated values in sats
4. Save changes

Examples:

- **Small tips**: `10,50,100`
- **Standard**: `100,1000,10000`
- **Premium support**: `1000,5000,10000,25000`
- **Custom**: `500,2500,5000`

Best Practices:

- Offer 3-4 preset options (not too many)
- Include range (small, medium, large)
- Consider typical ticket value/complexity
- Update based on Bitcoin price changes
- Test with real users for feedback

Satoshi Values:

- 1,000 sats ≈ $0.20 - $0.50
- 10,000 sats ≈ $2 - $5
- 100,000 sats ≈ $20 - $50
- Prices vary with Bitcoin exchange rate

B.5. Technical Questions

B.5.1. Q: What technologies does Zapdesk use?

A: Technical stack:

Frontend:

- **React** (18.x): UI framework
- **TypeScript** (5.x): Type-safe JavaScript
- **Vite** (5.x): Build tool and dev server
- **Zendesk App Framework (ZAF)** (v2): Zendesk integration SDK

Lightning Integration:

- **@getalby/sdk**: Nostr Wallet Connect implementation

- **qrcode.react**: QR code generation
- **LNURL-pay protocol**: Lightning Address resolution

Build & Deployment:

- **@zendesk/zcli**: Zendesk CLI for app packaging
- **npm/pnpm**: Package management
- **Git**: Version control

Testing:

- **Vitest**: Unit testing framework
- **React Testing Library**: Component testing
- **TypeScript**: Compile-time type checking

B.5.2. Q: Is Zapdesk open source?

A: License and source availability:

Current Status:

- Proprietary codebase by KnowAll AI
- Not currently open source
- Available for licensed deployment

Future Plans:

- Consider open sourcing under MIT license
- Community contributions welcome
- Transparent development roadmap

Access:

- Contact KnowAll AI for licensing
- Custom deployments available
- Source code access for licensed customers

B.5.3. Q: Does Zapdesk work with Zendesk mobile app?

A: Mobile compatibility:

Current Support:

- **Web Browser (Mobile)**: □ Fully supported
- Responsive design adapts to mobile
- Touch-optimized controls

- QR scanning via wallet apps works
- **Zendesk Mobile App**: □□ Limited
- Sidebar apps may not render
- Dependent on Zendesk mobile SDK
- Recommend mobile web browser instead

Best Mobile Experience:

1. Open Zendesk in mobile web browser (not app)
2. View ticket normally
3. Zapdesk sidebar appears and works fully
4. QR scanning seamless with wallet apps
5. NWC connection works in browser

Future Enhancements:

- Native mobile app integration (if Zendesk SDK improves)
- Progressive Web App (PWA) version
- Standalone mobile interface

B.5.4. Q: Can I integrate Zapdesk with other ticketing systems?

A: Platform compatibility:

Current Support:

- **Zendesk**: □ Fully supported (ZAF v2 framework)
- **Other Platforms**: □ Not currently supported

Potential Future Integrations:

- **Intercom**: Similar iframe app model
- **Freshdesk**: App marketplace support
- **Help Scout**: Custom app framework
- **Salesforce Service Cloud**: Lightning components

Custom Development:

- Core Lightning payment logic is platform-agnostic
- Can be adapted to other ticketing platforms
- Requires platform-specific SDK integration
- Contact KnowAll AI for custom development

Self-Service Option:

- Extract core payment components
- Build custom UI for your platform
- Maintain Zendesk version separately

B.5.5. Q: How do I report bugs or request features?

A: Bug reporting and feature requests:

Bug Reporting:

1. **Verify Issue**: Confirm bug is reproducible
2. **Check Documentation**: Review Troubleshooting Guide first
3. **Gather Information**:
    - Steps to reproduce
    - Expected vs actual behavior
    - Screenshots or screen recordings
    - Browser and Zendesk version
    - Console errors (F12 developer tools)
4. **Submit Report**:
    - Email: support@knowallai.com
    - GitHub Issues: (if open source)
    - Include all diagnostic information

Feature Requests:

1. **Business Case**: Describe need and benefit
2. **Use Case**: Explain specific scenario
3. **Priority**: Business critical vs. nice-to-have
4. **Mockups**: Include UI mockups if relevant

Contact Channels:

- **Email**: support@knowallai.com
- **GitHub**: (when available)
- **Community Forum**: (planned)

Response Times:

- **Critical Bugs**: 24-48 hours
- **Standard Issues**: 3-5 business days
- **Feature Requests**: Evaluated in planning cycles

Appendix C: Contact Information

KnowAll AI Support:

- Technical Support: support@knowallai.com
- General Inquiries: info@knowallai.com
- GitHub Issues: https://github.com/knowall-ai/zendesk-zapdesk/issues

Project Team:

- Lead Developer: Akash Jadhav - Akash.Jadhav@knowall.ai
- Product Manager: Ben Weeks - Ben.Weeks@KnowAll.ai

Business Hours:

- Monday - Friday: 9:00 AM - 6:00 PM IST
- Saturday - Sunday: Limited support for critical issues
- Emergency Contact: Please email with [URGENT] in subject line

Community Resources:

- Documentation: https://docs.knowallai.com/zapdesk
- Lightning Network Resources: https://lightning.network
- Zendesk Developer Portal: https://developer.zendesk.com