ATC Crisis Management System – Technical Documentation

Purpose of This Document

This document provides a comprehensive technical overview of the **ATC Crisis Management System**, including the **Power Automate solution** and the **ATC Crisis Management Progressive Web App (PWA)**. It is intended to onboard future developers, administrators, or IT managers responsible for maintaining or extending the system.

PART 1: POWER AUTOMATE

Solution: CrisisApp

Overview

The **CrisisApp** Power Automate solution acts as the secure backend for the ATC Crisis Management App. It validates users, stores incident data, processes attachments, and sends notifications to Microsoft Teams and leadership groups.

Security is enforced through: - Microsoft Entra ID authentication - JWT token validation - Microsoft Graph verification of active user accounts - Custom authorization headers for HTTP endpoints

---

Flow: Submit_Witness_Form (Primary Intake Flow)

Purpose

Handles secure submission of incident and witness reports from the ATC Crisis Management App.

High-Level Flow Logic

1. Validate caller identity via Azure AD token
2. Verify user account is active using Microsoft Graph
3. Store incident data in SharePoint
4. Upload photo attachments
5. Trigger Teams notifications
6. Return a structured HTTP response to the app

Actions Breakdown

- **Compose_AuthHeader**: Builds Bearer token from accessToken (No connector)
- **Compose_PayloadB64 / Compose_PayloadJson**: Decodes and parses JWT payload (No connector)
- **Compose_CallerOID**: Extracts user Object ID (OID) from token (No connector)
- **Invoke_an_HTTP_request_Graph_Call**: Retrieves user details from Microsoft Graph (Connector: shared_webcontents)
- **Condition – Account Enabled**: Ensures the user is active
- **Create_item**: Writes incident record to SharePoint (Connector: shared_sharepointonline-1)
- **Apply_to_each (Add_attachment)**: Uploads photo attachments to SharePoint
- **HTTP_Notify_Teams_GRAPH**: Calls Teams Graph notification flow
- **Condition_DidPostWork**: Falls back to HTTP-based Teams notification if needed
- **Response**: Returns success payload
- **Response_NotActiveID**: Returns HTTP 403 if user is disabled

---

Flow: CrisisApp_Notify_Teams_Channel_GRAPH (Primary Teams Posting)

Purpose

Posts incident details and inline images directly to a Teams channel using Microsoft Graph hosted content.

Key Features

- Inline image rendering (hosted contents)
- Deduplication protection (prevents reposts)
- SharePoint-driven attachment retrieval

Actions Breakdown

- Initialize variables for item ID, hosted content arrays, image HTML, and counters
- **Authorization Condition**: Validates incoming request
- **Get_item**: Retrieves SharePoint list item
- **Get_attachments**: Retrieves attached photos
- **Apply_to_each**:
- Get attachment content
- Convert to Base64
- Normalize image content types
- Append hosted content metadata
- Build HTML image blocks
- **Compose_MessageHtml**: Constructs Teams message body
- **Condition_AlreadyPosted**: Prevents duplicate posts
- **Invoke_an_HTTP_request**: Posts message and images to Teams
- **Update_SharePoint_Posted**: Stores Teams message ID
- **Responses**: Success / Already Exists / Unauthorized

Flow: CrisisApp_Notify_Teams_Channel_HTTP (Fallback)

Purpose

Fallback Teams notification method when Graph posting is unavailable.

Notes

- Posts attachment links instead of inline images
- Uses standard Teams connector

Key Actions

- Validate shared secret header
- Retrieve SharePoint item and attachments
- Build HTML message with attachment links
- Post to Teams channel
- Return HTTP response

Flow: CrisisApp_Notify_Leadership

Purpose

Sends **Executive Committee and Operations Leadership alerts** for critical crisis actions.

Features

- Dynamic group membership via Microsoft Graph
- Deduplication across multiple leadership groups
- Test mode routing
- Job-title filtering

Flow Logic

1. Retrieve Exec and Ops leadership members
2. Combine and deduplicate users
3. Filter excluded job titles
4. Determine test vs production recipients
5. Compose alert message
6. Send 1:1 Teams messages via Graph Chat connector

Connection References

Custom Connector: crisisapp_graphchat

- Uses Microsoft Graph Chat APIs
- Backed by Entra App Registration

App Registration: - Name: ATC_CrisisApp - Client Secret Expiration: 01/19/2028

Other Connections

- Microsoft Graph (pre-authorized)
- Microsoft Teams
- SharePoint Online
- Office 365 Users

PART 2: ATC CRISIS MANAGEMENT APP (PWA)

Overview

The **ATC Crisis Management App** is a mobile-first Progressive Web App (PWA) used by ATC staff to manage emergencies on-site. It is designed to function **offline**, synchronize when online, and provide guided crisis workflows.

Homepage: https://atc-development.github.io/atc-crisisapp/

Technology Stack

- React 19 + TypeScript
- Vite
- Tailwind CSS
- React Router
- Microsoft MSAL (Azure AD)
- GitHub Pages (deployment)
- Workbox (PWA & offline caching)

Authentication

File: msalConfig.ts

- Uses Microsoft Entra ID authentication
- Tokens stored in localStorage (PWA-friendly)

- Scopes include:
- OpenID / Profile / Email
- Microsoft Graph
- Power Automate Flow access
- Authentication state displayed via `AuthStatusBanner`

Routing

File: routes.tsx

Route	Purpose
/	Crisis category selection
/checklist/:category	Crisis checklist
/form/:formId	Incident report form

Crisis Categories Screen

File: CategoriesScreen.tsx

- Entry point for crisis selection
- Color-coded crisis types
- Property background imagery
- Reset button clears all cached state

Checklist System

Files: - ChecklistScreen.tsx - checklistData.ts

Features

- Step-by-step crisis actions
- Phone-call links (e.g., Call 911)
- Offline persistence via localStorage
- Leadership alerts triggered on critical actions

Forms System

File: FormScreen.tsx

Capabilities

- Text, numeric, boolean, date, and photo inputs
- Automatic photo compression
- GPS-based location auto-fill
- User identity auto-filled from Azure AD

Submission Flow

- Photos encoded as Base64
- Submitted to Power Automate
- Cached locally until successful submission

Location & Proximity Detection

Files: - locationService.ts - ProximityProvider.tsx - propertyLocations.json

Features

- Haversine distance calculations
- Property radius detection
- Nearest-property fallback
- Background updates every 2 minutes

Version Management & Updates

Files: - version_check.ts - main.tsx

Behavior

- Compares deployed version.json with build-time version
- Forces service worker refresh when mismatch detected
- Requires manual version bump before deployment

Leadership Chat Notifications

File: postToLeadershipChat.ts

- Triggered by critical checklist completions
- Sends Teams alerts with:
- Crisis category
- Reporter info
- Location

- Optional notes

Photo Compression

File: imageCompression.ts

- Enforces size limits for mobile reliability
- Supports JPEG and WebP
- Optimized for offline-first uploads

Local Storage Management

File: localStorageHelpers.ts

Stored Data

- Checklist state
- In-progress forms
- Leadership alert flags

App Shell & Layout

File: AppShell.tsx

- Persistent authentication banner
- Consistent layout wrapper

Build & Deployment

- Build: Vite + TypeScript
- Deploy: GitHub Pages (gh-pages)
- PWA: Service worker + offline cache
- Environment variables injected at build time

System Philosophy

This system is designed to: - Work reliably during emergencies - Function offline without data loss - Enforce security at every integration point - Minimize cognitive load for field staff

This is a mission-critical application. Changes should be tested thoroughly and deployed deliberately.