AHCA Florida Health Care Connections (FX)

FX-UOCI-PD-8b-1 Appendix A: Interface Control Documentation

Version: 100

Date: May 9, 2025

Author: Automated Health Systems (AHS), UOC Vendor

Submitted to: AHCA Contract Manager

Revision History

DATE	VERSION	DESCRIPTION	AUTHOR
12/05/2024	001	FX-UOCI-PD-8b-1 Appendix A: Interface Control Documentation	Stephen Mitchell
3/27/2025	002	FX-UOCI-PD-8b-1 Appendix A: Interface Control Documentation	Stephen Mitchell
5/5/2025	003	FX-UOCI-PD-8b-1 Appendix A: Interface Control Documentation Response to Comments	Stephen Mitchell
5/9/2025	100	FX-UOCI-PD-8b-1 Appendix A: Interface Control Documentation Final Version approved by the Agency	Drew Anderson

Quality Review History

DATE	REVIEWER	COMMENTS
12/05/2024	Jason Falorio	Reviewed and revised referenced documents, and contact information, and Change Management sections.
12/9/2024	Patricka Coleman	Reviewed for grammar and formatting
3/31/2025	Patricka Coleman	Reviewed for grammar and formatting
5/5/2025	Patricka Coleman	Reviewed for grammar and formatting

Table of Contents

Table of Exhibits

SECTION 1 OVERVIEW

The *Ticket Resolution API* enables integrating systems to resolve (close) a Ticket record within Dynamics 365 (D365). In standard, out-of-the-box D365 environments, what is referred to as "Tickets" within the Florida UOC are commonly known as "Cases" or "Incidents." As a result, the underlying code and references within the API utilize the term "incident" to align with D365's database schema.

1.1 PURPOSE OF THE ICD

This Interface Control Document (ICD) documents and tracks the necessary information to effectively define the D365 Ticket Resolution API interface within the Florida Health Care Connections (FX) system. This ICD ensures compatibility between system segments and components while providing clear guidance on the architecture and implementation requirements for resolving.

1.2 INTRODUCTION

This ICD describes the relationship between:

- **Source System**: ServiceNow (FX Help Desk) (Used to manage customer support processes, such as tracking customer issues, automating resolutions, and improving communication).
- **Target System**: D365 platform (specifically the incident/case management module)

The interface enables programmatic closure of ticket records within FX Service Desk, where tickets are also known as "Cases" or "Incidents" in standard D365 terminologies. Note tickets that originate in AHCA IT Help Desk or FX Help Desk (SNOW) are not managed by this API.

1.3 ICD UPDATE PROCEDURES

1.3.1 UPDATE REQUIREMENTS

Updates to this ICD will be required when:

- Changes occur to the D365 API endpoints. See section 2.1.4 Transaction Types, Volumes, and Frequency for more information about endpoints.
- New status codes or fields are added.
- Authentication mechanisms are modified.
- Integration requirements change.

1.3.2 Timing of Update

Updates will align with the Interface Development Lifecycle (IDLC) phases and any significant D365 platform update. Microsoft determines if an update is significant and notifies the D365 community of these updates in advance. Examples of a significant update would be when Microsoft added Co-Pilot to their products, retiring outdated client sided APIs, or Microsoft incorporating Portals in their products.

1.3.3 Update Procedure

Updates will follow the Change Control Process outlined in the Change Management Plan and FX Project Management Standards.

1.4 Overview

The Ticket Resolution API facilitates communication between ServiceNow and Microsoft D365, enabling integrating systems to resolve (close) Ticket records within D365. In the Florida UOC context, these "Tickets" correspond to "Cases" or "Incidents" in standard, out-of-the-box D365 environments. Consequently, the API references and underlying code utilize the term "incident" to align with D365's default terminology.

1.5 Assumptions/Constraints/Risks

1.5.1 General Interface Assumptions

- Valid Azure AD service accounts are established.
- App Registrations are completed in FL UOC Azure.
- Appropriate D365 security roles are configured.
- Systems can process JSON payloads.
- The ServiceNow team will coordinate with the UOC team before making updates to their API definitions.

1.5.2 Software and Hardware Assumptions

- FX Help Desk and FX Service Desk support HTTPS protocol.
- FX Help Desk and FX Service Desk can implement OAuth 2.0 authentication.
- FX Help Desk and FX Service Desk can handle JSON request/response formats.
- TLS 1.2 or higher is supported.

1.5.3 Operational Assumptions

- Bearer tokens are managed and refreshed appropriately.
- Systems can handle standard HTTP status codes.
- Error handling procedures are implemented.

1.6 Constraints

1.6.1 Hardware or Software Environment Constraints

- Must use HTTPS for all communications.
- Must support TLS 1.2 or higher.
- Must maintain active Azure AD registration.

1.6.2 Availability of Resources Constraints

- Microsoft's and ServiceNow's standard API rate limiting may apply. ServiceNow states most endpoints have a rate limit of 100 requests per second. Microsoft sets rate limits at approximately 6000 per 5 minute window.
- Token refresh requirements must be met. See: Token refresh
- Service account maintenance is required.

1.6.3 Interoperability Constraints

- JSON format is required for all requests.
- Systems must communicate using REST and support OAuth 2.0.
- Standard HTTP methods must be supported.

1.6.4 Interface/Protocol Constraints

- POST method is only for ticket resolution.
- Specific header requirements must be met.
- Character limits on text fields.

1.6.5 DATA REPOSITORY AND DISTRIBUTION CONSTRAINTS

- Data storage is limited to D365.
- Field length restrictions apply (see 2.1.6.3 Field/Element Definition).
- Status code values are fixed. (see 2.1.8.3 Flow Control)

1.7 RISKS AND MITIGATION STRATEGIES

Exhibit 1-1: Risks and Mitigation Strategies outlines potential risks encountered during the operation of the API and the corresponding strategies to mitigate them. To add further descriptive details to the table for clarity:

Purpose of the Table: The table is designed to proactively identify risks impacting API functionality and provide actionable strategies to address these risks effectively. This ensures that the system remains resilient and continues to function as intended.

Column Descriptions:

- **Risk**: List the specific challenges or failure points that could disrupt the API's operations, such as token expiration or service unavailability.
- **Mitigation Strategy**: Details the countermeasures or solutions to prevent or minimize the impact of these risks, like implementing a token refresh mechanism or a circuit breaker pattern.

Examples in Context:

For "Token expiration," the proposed token refresh mechanism ensures uninterrupted access to the API by preemptively renewing expired tokens.

The "Retry logic with exponential backoff" for rate-limiting helps maintain system stability by managing excessive requests in a controlled manner.

RISK	MITIGATION STRATEGY
38	Implement a token refresh mechanism. Tokens have a default lifetime of one hour and can be refreshed for up to 90 days by default.
Rate limiting	Implement retry logic with exponential backoff.
Data validation failures	Implement proper error handling.
Service unavailability	Implement circuit breaker pattern.

Exhibit 1-1: Risks and Mitigation Strategies

SECTION 2 DETAILED INTERFACE REQUIREMENTS

2.1 FUNCTIONAL SUMMARY

2.1.1 DATA EXCHANGE SUMMARY

The interface facilitates one-way communication to resolve tickets in D365 via a REST API endpoint. JSON data is exchanged between the two systems described in this ICD, outlining the interaction between ServiceNow and D365.

Base URL: https://[base-instance-url].crm9.dynamics.com/api/data/v9.0/ResolveIncident.

2.1.2 ASSUMPTION.

- Ticket ID must exist in the FX Service Desk.
- The service account within FX. Service Desk has service writer and service reader permissions

2.1.3 GENERAL PROCESSING STEPS

1. Obtain OAuth 2.0 token (below items are required in sequence):
    - POST to: https://login.microsoftonline.com/[tenant-id]/oauth2/token.
    - Include the required credentials and parameters.
    - Construct ticket resolution request.
2. Submit a POST request with a valid token.
3. Process response (204 No Content for Success).

2.1.4 TRANSACTION TYPES, VOLUMES, AND FREQUENCY

Transaction Type: POST - Ticket Resolution

Expected Response: 204 No Content

Frequency: On-demand

Volume: The volume of transactions will depend on system usage patterns and operational demands, specifically the ticket volume in ServiceNow, as it initiates the transaction. Historically, the number of tickets processed in ServiceNow has been 50–70 per month. However, as additional solution releases are implemented and more users enter the environment, this volume could increase significantly, potentially reaching over one hundred (100) tickets per day. Service now states most endpoints have a rate limit of 100 requests per second. Microsoft sets rate limits at approximately 6000 per 5 minute window.

ServiceNow will provide volume metrics through tools such as *Now Platform Reporting* and D365 *Insights*, which track the number of incidents requiring resolution and their statuses (tickets or cases). These metrics will give an accurate measure of transaction volumes. Seasonal peaks or fluctuations, such as higher transaction volumes during specific quarters or events, may also influence these estimates.

2.1.5 Interface Processing Time Requirements

2.1.5.1 Predecessors and Dependencies

- A valid OAuth token must be obtained.
- A Ticket must exist in D365.
- The status code must be valid. Invalid status codes could occur if changes are made without coordination between the FX Service Desk and FX Help Desk teams.

2.1.5.2 Interface Schedule and Cut-Off Times

Available 24/7, subject to:

- Azure AD authentication service availability
- D365 platform availability
- Token expiration (3599 seconds).

2.1.6 Information Exchange Model or Message Format or Record Layout

2.1.6.1 File Layout

Request Body Format:

Exhibit 2-1: Request Body Format JSON code block, presented below, illustrates the required structure and format of the request body when interacting with the D365 Ticket Resolution API.

Field Breakdown and Usage:

- **IncidentId**:

  The **incidentid** field identifies the ticket to be resolved using a globally unique identifier (GUID, example aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb). This is critical since it pinpoints the specific D365 record to be updated.

  The **@odata.type** field specifies the D365 entity type (Microsoft.Dynamics.CRM.incident), ensuring proper handling by the API.

- **Status**:

  Indicates the final state of the ticket with a predefined status code (e.g., 183410000 for "Closed-Resolved"). This field is integral to marking the ticket as resolved within D365.

- **BillableTime**:

  An integer value for the billable minutes spent resolving the ticket (default: 0). This field provides flexibility for future billing-related functionality, even if not actively used in the current implementation.

- **Resolution**:

  A short text field that maps to the "subject" of a Case Resolution activity record in D365, providing a concise explanation of the resolution.

- **Remarks**:

  A detailed text or memo field that maps to the "description" of a Case Resolution activity record, offering additional information about the resolution process.

Connection to the API Workflow:

This request body format is crucial to the **POST** request sent to the /**ResolveIncident** endpoint. The format directly impacts the success of operations by ensuring that the data adheres to D365's requirements for processing ticket resolution requests.

```
{
    "IncidentId": {
        "incidentid": "[GUID]", // Unique identifier for the ticket
        "@odata.type": "Microsoft.Dynamics.CRM.incident" / Entity type
    },
    "Status": [STATUS_CODE], // Resolution status code
    "BillableTime": 0, // Defaults to 0 for FX implementation
    "Resolution": "[RESOLUTION_TEXT]", // Summary of resolution
    "Remarks": "[REMARKS_TEXT]" // Optional additional notes
}
```

Exhibit 2-1: Request Body Format

2.1.6.2 DATA ASSEMBLY CHARACTERISTICS

Headers Required:

- **Accept**: /
- **Accept-Encoding**: gzip, deflate, br
- **Connection**: keep-alive
- **Authorization**: Bearer [token]

2.1.6.3 FIELD/ELEMENT DEFINITION

Exhibit 2-2: Field/Element Definition table below provides a detailed breakdown of the fields or elements required to interact with the D365 Ticket Resolution API. It serves as a reference guide for developers, ensuring they understand the structure, data types, constraints, and purpose of each field used in API requests.

Column Explanations

- **Field Name**:

  Lists the specific attributes or fields that must be included in the API's request payload, each corresponding to a key in the JSON request body. These are integral for processing and resolving tickets in D365.

- **Type**:

  Defines the data type for each field, such as Lookup, Option Set, Integer, String, or Memo. These types ensure compatibility with D365's internal schema.

- **Size**:

  Specifies the maximum allowable size for the data in each field. For example:

  - GUID for IncidentId is a globally unique identifier.
  - Strings like *Resolution* are limited to 200 characters.
  - Remarks allow up to 100,000 characters, providing ample space for detailed descriptions.

- **Required**:

  Indicates whether the field is mandatory (Yes) or optional (No). This distinction ensures developers know which fields (e.g., IncidentId, Status) must always be included in their requests.

- **Valid Values**:

  Defines acceptable input values or constraints for each field:

  - **IncidentId**: Must be a valid GUID.
  - **Status**: Must match specific status codes (referenced elsewhere in the ICD).
  - **BillableTime**: Defaults to 0, this field is required by Microsoft, however, is a field not used in Service Desk.
- **Description**:

  The table briefly explains the purpose or role of each field within the API. For instance:

  - IncidentId uniquely identifies the ticket to be resolved.
  - Resolution serves as the subject of the resolution activity record in D365.

The table provides a comprehensive reference for developers and technical users, ensuring they know how to structure their requests.

It connects the request body format (as shown in Exhibit 2-1: Request Body Format) to its individual components, clarifying the role and constraints of each field.

Specifying sizes, required fields, and valid values reduces the likelihood of errors during API integration.

FIELD NAME	TYPE	SIZE	REQUIRED	VALID VALUES	DESCRIPTION
IncidentId	Lookup	GUID	Yes	Valid GUID	Unique identifier of ticket
Status	Option Set	Int	Yes	See Status Codes	Resolution status code
BillableTime	Integer	0-2147483647	Yes	0	Minutes spent (default 0)
Resolution	String	200	Yes	Text	Resolution subject
Remarks	Memo	100000	No	Text	Resolution description

Exhibit 2-2: Field/Element Definition

Status Codes:

- 1: In Progress
- 183410000: Closed-Resolved
- 183410001: Closed – Unable to Resolve
- 183410002: Closed – Partially Resolved
- 183410003: Closed - Closed by Originator, Not needed
- 183410004: Closed – Closed by Originator, User Resolved Issue
- 6: Cancelled

2.1.6.4 SAMPLE MESSAGES/PAYLOADS

Example Request:

Exhibit 2-3: Sample Messages/Payloads below demonstrate an example of a payload or message format that can be sent to the D365 Ticket Resolution API. It serves as a practical reference for developers, illustrating how to construct a valid API request that adheres to the required structure, data types, and field definitions outlined in Exhibit 2-2: Field/Element Definition.

Field Breakdown and Usage

Each field in the code block is derived from the Field/Element Definition (Exhibit 2-3: Sample Messages/Payloads), and their purposes are outlined below:

- **IncidentId**:
  - **Purpose**: Specifies the unique identifier of the ticket (or incident) to be resolved. This ensures the API targets the correct record in D365 for processing.
  - **Example Value**: "incidentid": "7c9b5b85-3471-ef11-a670-001dd804ec22"

    A GUID (globally unique identifier) that uniquely identifies the ticket.

    @odata.type: "Microsoft.Dynamics.CRM.incident" specifies the entity type in D365 to indicate that the resolution is associated with an "incident" or "case" entity.

- **Status**:
  - **Purpose**: Indicates the resolution status of the ticket using predefined status codes.
  - **Example Value**: "Status": 183410000

  Status code 183410000 corresponds to "Closed-Resolved," marking the ticket as successfully resolved.

- **BillableTime**:
  - **Purpose**: Represents the number of minutes spent resolving the ticket.
  - **Example Value**: "BillableTime": 0

  In this case, 0 is used as no billable time is defined for this specific ticket resolution scenario.

- **Resolution**:
  - **Purpose**: Serves as the subject or summary of the resolution activity. This provides a concise description of how the ticket was resolved.
  - **Example Value**: "Resolution": "Issue resolved per customer confirmation."

  In this case, the resolution is based on customer confirmation that the issue was resolved.

- **Remarks**:
  - **Purpose**: Provides additional details or context about the ticket resolution process, captured as a descriptive memo.
  - **Example Value**: "Remarks": "" (indicates that the value for the **Remarks** field is an empty string.) No additional remarks are provided here, but the field is included for scenarios requiring extra documentation or commentary.

Additional Notes:

The **IncidentId** and **Status** fields are mandatory and must always be present with valid values.

Default values, such as **BillableTime**: 0 and an empty string for **Remarks**, are provided for fields without specific requirements, ensuring the request remains valid.

This example is vital for developers to understand the proper syntax and data structure required for successful communication with the API.

```
{
    "IncidentId": {
        "incidentid": "7c9b5b85-3471-ef11-a670-001dd804ec22", // Unique
identifier (GUID) of the ticket/incident to be resolved
        "@odata.type": "Microsoft.Dynamics.CRM.incident"  // Specifies the
entity type (incident) in Dynamics 365
    },
    "Status": 183410000, // Resolution status code (e.g., 183410000 = Closed-
Resolved)
    "BillableTime": 0, / Time spent resolving the ticket in minutes; default
is 0
    "Resolution": "Issue resolved per customer confirmation",  // A short
description or subject of the resolution activity
    "Remarks": "" // Additional notes or detailed description of the
resolution (optional; left blank here)
}
```

Exhibit 2-3: Sample Messages/Payloads

2.1.7 ENVIRONMENT SPECIFICATIONS

2.1.7.1 ENVIRONMENT INVENTORY

- **FL UOC – DR (Disaster Recovery) Environment**:

  The FL UOC - DR (Disaster Recovery) environment is a secondary environment designed to ensure UOC business continuity in the event of a disaster or system failure. Currently hosted and managed through the Microsoft Power Platform Admin Center, the DR environment mirrors critical configurations and data from the Production environment. Backups are performed regularly to minimize data loss. While it is not actively used, it is a standby environment to support failover processes and ensure uninterrupted API operations, such as the D365 Ticket Resolution API, during outages. This environment provides a critical safety net for mission-essential functions.

- **FL UOC – Hotfix Environment**:

  The FL UOC - Hotfix environment is a specialized environment within UOC that is used to implement and test urgent fixes to the system. It provides a controlled space to address critical issues or defects identified in the Production environment, ensuring that changes are validated before deployment. The Hotfix environment allows developers to make isolated adjustments to code, configurations, or integrations—such as the D365 Ticket Resolution API—without disrupting ongoing operations in other environments. Once tested and verified, the fixes are deployed to Production to resolve issues with minimal downtime and risk.

- **FL UOC – QA (Quality Assurance) Environment**:

  The FL UOC - QA (Quality Assurance) environment is used within UOC to thoroughly test system functionality, integration, and performance. It provides a controlled and stable environment that closely mirrors Production, enabling teams to validate features, workflows, and configurations—such as the D365 Ticket Resolution API—before being promoted to higher environments. The QA environment focuses on identifying defects, verifying fixes, and ensuring overall system reliability, serving as a critical step in maintaining the quality of the UOC.

- **FL UOC – Sandbox Environment**:

  The FL UOC - Sandbox environment is a flexible, isolated environment within UOC designed for experimentation, testing, and prototyping. It allows teams to validate configurations, develop proof-of-concept solutions, and test system changes—such as the D365 Ticket Resolution API—without impacting other environments. The Sandbox environment provides a safe space for trying out new ideas or testing potential updates, ensuring that changes can be evaluated and refined before moving to formal testing or Production environments.

- **FL UOC SIT (System Integration Testing) Environment**:

  The FL UOC - SIT (System Integration Testing) environment validates the integration and interaction of various components and systems within the UOC. It ensures that workflows, data exchanges, and interfaces—such as the D365 Ticket Resolution API and its connection to external systems like ServiceNow—operate seamlessly in a cohesive system. The SIT environment replicates Production-like conditions for testing end-to-end business processes, error handling, and data accuracy. This environment is crucial for identifying and resolving integration-related issues before moving to UAT or Production.

- **FL UOC - UAT (User Acceptance Testing) Environment**:

  In the User Acceptance Testing (UAT) environment, UOC and the Agency end users validate that the system meets business requirements and performs as expected. This environment allows stakeholders to test workflows and provide approval for production readiness.

- **FL UOC (Production) Environment**:

  The FL UOC - Production environment is the live operational environment within UOC. It hosts all finalized and validated configurations, workflows, and data—such as the D365 Ticket Resolution API—and serves as the primary system used by end users for daily operations. The Production environment is designed to handle real-time transactions and support business-critical processes, ensuring reliability, performance, and security to meet organizational and client needs. This environment represents the culmination of all development, testing, and validation efforts.Environment Alignment.

The environments supporting the UOC are aligned to ensure seamless integration, testing, and production readiness for the D365 Ticket Resolution API. Each environment—sandbox, System Integration Testing (SIT), User Acceptance Testing (UAT), and Production—serves distinct roles within the development lifecycle and is synchronized to support consistent functionality across all stages.

Environment mapping between the Agency, UOC, and its partners will ensure that all systems operate cohesively, maintaining data integrity and configuration parity. This alignment is critical to ensure that updates, testing, and deployments occur in a controlled and predictable manner.

2.1.7.2 Disaster Recovery (DR) Procedure

The Disaster Recovery (DR) procedure for the D365 system leverages its cloud-based architecture. Backups are performed every 30 minutes and retained for 30 days. These backups are stored as system images and can be managed efficiently through the Microsoft Power Platform Admin Center.

2.1.8 Communication Methods

2.1.8.1 Interface Initiation

- HTTPS POST request to the specified endpoint
- OAuth 2.0 authentication required
- JSON payload required

2.1.8.2 Protocol Specifications

- Protocol: HTTPS
- Authentication: OAuth 2.0
- Data Format: JSON
- Method: POST.

2.1.8.3 FLOW CONTROL

Error Handling:

1. **HTTP Status Codes**:
    - 200: Success
    - 204: Success (No Content)
    - 400: Bad Request
    - 401: Unauthorized
    - 403: Forbidden
    - 404: Not Found
    - 429: Too Many Requests
    - 500: Internal Server Error
2. **D365-Specific Error Codes**:
    - 0x8006088a: Resource Segment Error
    - 0x80040265: Invalid Argument
    - 0x80040237: Insufficient Access Rights
    - 0x80044150: Record Not Found
    - 0x80060891: Invalid Entity

2.1.9 MOCK INTERFACE REQUIREMENTS

The D365 Ticket Resolution API interface is fully implemented in the Sandbox, Test (QA), and UAT environments.

2.1.10 MOCK INTERFACE AGENCY TEST CASES

No mock interface requirements exist for this API integration, as the functionality is fully implemented and validated in the designated environments (e.g., Sandbox, QA, UAT). Agency test cases are not required for this specific implementation

2.1.11 MOCK INTERFACE PARTNER TEST CASES

No mock interface requirements exist for this API integration, as all testing is conducted directly within the controlled environments and follows the established testing procedures for partner systems.

2.2 Interface Security Requirements

2.2.1 Data Encryption (At-Rest)

- Data is stored within the D365 platform.
- Follows Microsoft's standard encryption protocols.

2.2.2 Data Encryption (In Transit)

- TLS 1.2 or higher required
- OAuth 2.0 token encryption
- HTTPS is required for all communications

2.3 Operational Requirements

2.3.1 Operational Specifications

- **Service availability**: 24/7
- **Token refresh**: Every 3599 seconds
- **Rate limiting**: The rate-limiting configuration—Maximum 1,000 requests per minute per token, with a burst rate of 5,000 requests for up to 5 seconds—is based on current projections of system usage and anticipated workload. As the environment evolves and additional solution releases are implemented, this rate limit may need to be adjusted, resulting in higher transaction volumes. Regular monitoring of API usage patterns will help determine if modifications are necessary to support increased demand while maintaining system stability and performance.

2.3.2 Exception Handling Procedures

1. **Token Expiration**:
    - Implement token refresh.
    - Retry with a new token.
2. **Service Unavailability**:
    - Implement retry with backoff.
    - Log failure after retry exhaustion.

3. **Invalid Data**:
    - Log validation errors.
    - Return appropriate error response.

2.3.3 OPERATIONAL COORDINATION

Operational coordination for the D365 Ticket Resolution API ensures efficient communication and resolution of issues across all teams and partners involved in the Florida UOC. Escalation procedures are in place to address incidents and ensure timely resolution while minimizing disruption to operations.

Escalation Procedures:

The escalation process follows a tiered structure to ensure that issues are addressed efficiently and escalated to the appropriate personnel if not resolved at the initial level:

1. **Tier 1**: Service Desk Support
    - Initial troubleshooting and resolution of common or low-complexity issues.
    - Escalates unresolved issues to Tier 2 support.
2. **Tier 2**: Technical Support/Development Team
    - Handles more complex API functionality, system integrations, or configuration issues.
    - Escalates critical or enterprise-wide issues to Tier 3 or UOC management.
3. **Tier 3**: UOC IT Management and Vendors
    - Addresses critical issues impacting multiple systems, environments, or stakeholders.
    - Coordinates with Microsoft Power Platform Admin Center for issues requiring platform-level intervention.
    - Ensures timely updates to the Agency and key stakeholders.
4. **Final Escalation**: Executive or Agency Notification

If resolution efforts fail or the issue impacts compliance or contractual obligations, the matter is escalated to the Agency's leadership or UOCs' executive team for further action.

Exhibit 2-4: Contact Information provides a detailed list of primary and backup contacts for each escalation tier in the support process. The table outlines the designated roles and responsibilities, contact details, and escalation hierarchy, ensuring that issues are directed to

the appropriate personnel at each level of support. This information serves as a quick reference for initiating and managing the escalation process, facilitating effective communication, and timely resolution of technical issues.

Contact Information

Escalation Tier	Role/Team	Primary Contact Name	Backup Contact Name	Phone Number	Email Address
1	Service Desk Support Specialists	Patrick McNutt	Jose Vera	(850) 402-4666 ext. 2240 (850) 402-4666 ext. 2240	pmcnutt@automated-health.com jvera@automated-health.com
2	Technical Architect	Jason Falorio	Greg Holtz	(412) 367-3030 ext. 2645 (412) 367-3030 ext. 2609	jfalorio@automated-health.com gholtz@automated-health.com
3	Director of Technology	Gus Costa	Patrick McNutt	(850).402-4666 ext. 2206 (850).402-4666 ext. 2240	gustavo.costa@automated-health.com PMcNutt@automated-health.com
Executive	Account Executive	Derek Jackson	Braxton Wilson	(850).402-4666 ext. 2201 (850).402-4666 ext. 2327	djackson@automated-health.com Braxton.wilson@automated-health.com

Exhibit 2-4: Contact Information

2.4 Change Management and Maintenance

Change management and maintenance for the D365 Ticket Resolution API will adhere to established organizational policies and procedures to ensure system stability, compliance, and operational efficiency. The process minimizes disruptions while allowing for controlled and documented system updates.

Change Management: All API or related systems changes must follow a formal Change Request Process, including review, approval, and implementation tracking.

- **Changes will be categorized as**:
    - **Routine Changes**: Minor updates with low risk and minimal impact
    - **Urgent Changes**: Critical updates are required to resolve significant issues or ensure business continuity.
    - **Major Changes**: Substantial updates that require extensive testing and stakeholder approval.

Change requests will be documented in the change management system, Jira, and must include the following details:

- **Description of the Change**: A clear and concise summary of the requested change, including the problem it addresses and the proposed solution.
- **Justification and Impact Analysis**: A detailed explanation of the reason for the change, its necessity, and an assessment of its potential impact on systems, stakeholders, and business operations
- **Testing and Validation Plans**: A comprehensive plan outlining the testing and validation approach to ensure the change functions as intended without introducing new issues.
- **Approval from Designated Stakeholders**: Formal review and sign-off from all relevant stakeholders, ensuring alignment and accountability before implementation.

Jira will serve as the centralized platform for logging, tracking, and managing change requests, ensuring a structured and traceable change management process.

Maintenance:

Regular maintenance activities will include:

- Monitoring API performance and resolving issues proactively.
- Applying updates, patches, or bug fixes in alignment with the maintenance schedule.
- Performing periodic reviews of API functionality to ensure it aligns with current business requirements.

Maintenance windows will be coordinated with all stakeholders to minimize operational impact.

Roles and Responsibilities:

- **Development Team**: Implements and tests changes in the Sandbox or QA environments.
- **UOC IT Management**: Oversees the approval process and ensures alignment with enterprise policies.
- **Stakeholders**: Provide input on requested changes and validate major updates during UAT.

Documentation:

- All changes and maintenance activities will be documented, including their impact and resolution, to maintain a clear history of modifications for audit and compliance purposes.

SECTION 3 QUALIFICATION METHODS

The following qualification methods will be used to verify the interface requirements:

3.1 DEMONSTRATION

1. **Authentication Flow**:
    - Demonstrate successful OAuth token acquisition.
    - Demonstrate token refresh process.
    - Demonstrate token usage in API calls.
2. **Basic Operations**:
    - Demonstrate successful ticket resolution.
    - Demonstrate the handling of various status codes.
    - Demonstrate proper field validation.

3.2 TESTING

1. **Functional Testing**:
    - Verify all status codes are accepted.
    - Verify field length restrictions.
    - Verify required field validation.
    - Verify proper error handling.
2. **Integration Testing**:
    - Verify end-to-end workflow.
    - Verify system integration points.
    - Verify error handling across systems.
3. **Performance Testing**:
    - Verify response times.
    - Verify concurrent request handling.
    - Verify token refresh impact.

3.3 ANALYSIS

1. **Security Analysis**:
    - Review of authentication implementation.
    - Review of data encryption methods.
    - Review of error handling security.
2. **Performance Analysis**:
    - Review of response times.
    - Review of resource utilization.
    - Review of error rates and types.

3.4 INSPECTION

1. **Code Review**:
    - Review of implementation code.
    - Review of security practices.
    - Review of error handling implementation.
2. **Documentation Review**:
    - Review of API documentation.
    - Review of error-handling procedures.
    - Review of operational procedures.

APPENDIX B – DATA DICTIONARY

Exhibit 4-1: Data Dictionary below, provides a structured reference for the fields used by the D365 Ticket Resolution API. It outlines critical information about the database fields, including their names, types, lengths, descriptions, and valid values or codes. This table is a foundational resource for ensuring consistency and clarity in data entry, processing, and reporting. Each field is designed to capture specific details about an incident, resolution activity, or associated metadata, enabling efficient tracking and resolution of tickets through the API.

Column Explanations:

- **Field Name**: Specifies the field's name as it is used in the system to represent specific data elements.
- **Database Field**: Indicates the corresponding name of the field in the database schema.
- **Type**: Describes the data type of the field, such as GUID, Integer, String, or Memo, to define the nature of the data stored.
- **Length**: Displays the character limit or size of the data field where applicable (e.g., 36 for GUIDs, 200 for strings).
- **Description**: Provides a concise explanation of the field's purpose and the data it captures.
- **Valid Values/Codes**: Lists acceptable values, formats, or codes for the field to ensure standardization and data validation.

Field Examples:

- **IncidentId**: Uniquely identify each ticket using a GUID (globally unique identifier) with a length of 36 characters.
- **Status**: Reflects the resolution status of a ticket using predefined codes (e.g., 1: In Progress or 183410000: Closed - Resolved) to indicate the ticket's current state.
- **BillableTime**: Captures the time spent resolving a ticket as an integer, with valid values ranging from 0 to 2,147,483,647.
- **Resolution**: Allows free-text entry (up to 200 characters) to summarize the resolution activity.
- **Remarks**: A "Memo" type field that supports extensive text for detailed descriptions of the resolution process.

This Data Dictionary is an essential guide for understanding the structure and requirements of data fields within the API, ensuring accurate and meaningful documentation for incidents and resolutions.

Field Name	Database Field	Type	Length	Description	Valid Values/Codes
IncidentId	incidentid	GUID	36	Unique identifier of ticket	Valid GUID format
Status	statuscode	Integer	-	Resolution status code	1: In Progress 183410000: Closed-Resolved 183410001: Closed-Unable to Resolve 183410002: Closed-Partially Resolved 183410003: Closed-Not Needed 183410004: Closed-User Resolved 6: Cancelled
BillableTime	billabletime	Integer	-	Time spent on resolution0	0-2147483647 (Default: 0)
Resolution	subject	String	200	Subject of resolution activity	Free text
Remarks	description	Memo	No 100000	Detailed resolution description	Free text

Exhibit 4-1: Data Dictionary

.

APPENDIX C – DATA MAPPING CROSSWALK

C.1 REQUEST MAPPING

Exhibit 5-1: Data Mapping Crosswalk below provides a precise mapping of the source fields from the request payload to their corresponding target fields within the D365 system. The table details any transformations applied during mapping, ensuring the data is accurately processed and stored in the appropriate fields.

Column Explanations:

- **Source Field**: Represents the field name in the API request payload submitted to the D365 system.
- **Target Field**: Indicates the corresponding field in the D365 database where the data will be stored or updated.
- **Transformation**: Describes any processing or adjustments applied to the data before mapping. For example, default values may be assigned, or direct mappings may include additional annotations like @odata.type.

Purpose and Relevance:

This crosswalk is a vital resource for developers and integrators working with the API. It ensures that the source data is correctly aligned with the system's data schema, maintaining consistency and accuracy during API transactions. Additionally, by explicitly documenting transformations, the crosswalk aids in debugging, data validation, and understanding the system's behavior during data transfers.

SOURCE FIELD	TARGET FIELD	TRANSFORMATION
IncidentId	incident.incidentid	Direct mapping with @odata.type
Status	incident.statuscode	Direct mapping
BillableTime	incident.billabletime	Default to 0
Resolution	incident.resolution.subject	Direct mapping
Remarks	incident.resolution.description	Direct mapping

Exhibit 5-1: Data Mapping Crosswalk

Note:

The "Direct Mapping" designation indicates that the source field in the request payload corresponds exactly to the target field in the D365 database, with no modifications or processing required. In contrast, a "Transformation" occurs when the data from the source field is altered, processed, or assigned a default value before being mapped to the target field. Transformations ensure that the data adheres to specific business rules, formats, or defaults the D365 system requires.

For example:

- **Direct Mapping**: The **IncidentId** field is passed directly to the target field without modification.
- T**ransformation**: The **BillableTime** field is automatically set to a default value of 0 if no input is provided in the request payload.

C.2 Response Mapping

Exhibit 5-2: Response Mapping below correlates HTTP status codes returned by the D365 Ticket Resolution API and their corresponding business meanings. This table serves as a guide to understanding the outcome of API requests, helping developers and stakeholders interpret the results and diagnose issues effectively. Each status code conveys the success or failure of a request, along with the specific reason for any errors encountered.

HTTP Status	Business Meaning
204	Successful ticket resolution
400	Invalid request data
401	Authentication failed
403	Insufficient permissions
404	Ticket not found

Exhibit 5-2: Response Mapping

APPENDIX D – GLOSSARY AND ACRONYMS

Exhibit 6-1: Glossary and Acronyms below provide a comprehensive reference for key terms and acronyms used throughout this document. The glossary ensures a consistent understanding of technical terminology and abbreviations, supporting clear communication among developers, stakeholders, and other D365 Ticket Resolution API users. Each entry includes a concise definition to clarify its meaning within the context of this document.

TERMINOLOGY	DEFINITION
API	Application Programming Interface
D365	Microsoft Dynamics 365
FX	Florida Health Care Connections
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol
ICD	Interface Control Document
IDLC	Interface Development Life Cycle
JSON	JavaScript Object Notation
OAuth	Open Authorization
QA	Quality Assurance
REST	Representational State Transfer
SIT	System Integration Testing
TLS	Transport Layer Security
UAT	User Acceptance Testing
UOC	Unified Operations Center

TERMINOLOGY	DEFINITION
URI	Uniform Resource Identifier

Exhibit 6-1: Glossary and Acronyms