Research Report: Microsoft Authentication Evolution & Graph API Integration Strategy

Date: 2025-12-04 **Researcher:** Claude (AI Assistant) **Requested By:** User designing employee authentication via Microsoft accounts with Graph API integrations for HIPAA-compliant workflows

---

Executive Summary

Microsoft's authentication landscape is undergoing significant changes in 2024-2025, with mandatory MFA enforcement, Azure AD Graph API retirement, and a strong push toward service principals over user-based service accounts. For your use case involving employee sign-in, Teams meeting transcripts, OneNote publishing, and HIPAA-compliant file storage, the key strategic insight is: **Microsoft is actively discouraging the "dedicated user account for automation" pattern (like your autopilot@southviewteam.com approach) in favor of service principals with delegated or application permissions.**

The good news: MCP (Model Context Protocol) servers for Microsoft/Azure are now production-ready, enabling Claude to interact with Azure services directly. Combined with Azure Foundry's HIPAA BAA coverage for Claude models, there's a viable path forward—but it requires restructuring your authentication approach away from user impersonation toward proper service principal architecture.

Your Teams meeting transcript challenge has a specific technical constraint: **the Graph API only supports transcripts for meetings associated with calendar events**, not ad-hoc meetings created via the `create onlineMeeting` API. This explains your difficulty with channel calendar meetings.

Background

You're building applications that need to:

1. **Authenticate employees** via their Microsoft accounts
2. **Leverage Microsoft security** features for HIPAA compliance
3. **Publish reports** to OneNote notebooks accessible via Teams
4. **Pull meeting transcripts** from Teams recordings
5. **Automate meeting scheduling** with auto-record/transcribe settings
6. **Potentially orchestrate** via Claude/Azure AI with MCP

The challenges you've encountered reflect real architectural constraints in Microsoft's ecosystem, not just documentation gaps.

Key Findings

Finding 1: Microsoft Authentication is Shifting Away from User-Based Service Accounts

The 2025 MFA Mandate Changes Everything

Microsoft has implemented mandatory MFA enforcement in two phases:

- **Phase 1 (H2 2024):** MFA required for Entra admin center, Azure portal, Intune admin center
- **Phase 2 (October 2025):** MFA required for Azure CLI, PowerShell, mobile app, and IaC tools

Impact on your autopilot@southviewteam.com approach:

> *"Microsoft does not recommend user accounts as service accounts because they are less secure... With mandatory MFA enforcement, it's critical to migrate user-based service accounts to secure cloud-based service accounts with workload identities."* — *Microsoft Best Practices*

Microsoft's recommended priority order:

1. **Managed Identity** (best, but only for Azure-hosted services)
2. **Service Principal** with certificate authentication
3. **Service Principal** with client secret
4. **User account** (least recommended, impacted by MFA)

Critical Deadline: March 31, 2026 — Service-principal-less authentication will fail. All apps must have service principals in the tenant.

Finding 2: Azure AD Graph API Retirement Timeline

The legacy Azure AD Graph API is being completely retired:

Date	Impact
August 31, 2024	New apps blocked unless opt-in
February 1, 2025	All apps must explicitly opt-in
June 30, 2025	**Complete retirement—all calls fail**

Action Required: Migrate all integrations to Microsoft Graph API before June 2025.

Source: Azure AD Graph API Retirement

Finding 3: Teams Meeting Transcripts Have Calendar Association Requirements

This is the root cause of your channel calendar difficulties:

> ***"This API doesn't support meetings created using the create onlineMeeting API that are not associated with an event on the user's calendar."***

What works:

- Private chat meetings (scheduled via Outlook/Teams with calendar event)
- Channel meetings (with calendar event association)
- Meetings where transcription was enabled

What doesn't work:

- Private channel meetings (explicitly unsupported)
- Ad-hoc meetings without calendar events
- Meetings created purely via `create onlineMeeting` API

Permissions Required:

Access Type	Permission	Notes
Delegated	`OnlineMeetingTranscript.Read.All`	Requires signed-in user
Application	`OnlineMeetingTranscript.Read.All`	Plus Application Access Policy via PowerShell

Resource-specific	`OnlineMeetingTranscript.Read.Chat`	Private chat meetings only

The Workaround: Use the `Create event` API (not `create onlineMeeting`) to schedule meetings with Teams meeting settings. This ensures calendar association.

Source: Microsoft Graph Transcripts API

Finding 4: OneNote API Losing App-Only Authentication in March 2025

Critical Change:

> "*The Microsoft Graph OneNote API will no longer support app-only authentication effective March 31, 2025. Microsoft recommends that you update your solutions to use delegated authentication.*"

This means your report publishing to OneNote notebooks will require:

- A signed-in user context (delegated permissions)
- OR interactive auth flow with token caching

Available Endpoints:

```
POST /sites/{site-id}/onenote/pages  ← SharePoint team site notebooks
POST /groups/{group-id}/onenote/pages  ← Teams group notebooks
POST /me/onenote/pages  ← User's personal notebook
```

Limitation: Creating OneNote tabs in Teams channels via Graph API has configuration restrictions. You can create the tab, but cannot specify which notebook it opens to.

Source: OneNote API Overview

Finding 5: Power Automate Custom Connectors Are Required for Meeting Management

Your experience with Power Automate and meeting settings is typical—there's no out-of-box solution.

Working Approach (from community solutions):

1. Create Entra ID app registration with delegated permissions
2. Build custom connector for Graph API
3. Configure Application Access Policy via PowerShell:

   ```
   New-CsApplicationAccessPolicy -Identity "MeetingTranscriptPolicy" -AppIds "your-client-id" -Description "Allow transcript access"
   Grant-CsApplicationAccessPolicy -PolicyName "MeetingTranscriptPolicy" -Identity "autopilot@southviewteam.com"
   ```

For auto-recording/transcription settings:

```
PATCH /users/{user-id}/onlineMeetings/{meeting-id}
{
  "recordAutomatically": true,
  "transcription": { "transcriptionEnabled": true }
}
```

Requirement: Power Apps/Power Automate Premium license for custom connectors.

Source: Enable Auto-Recording via Power Automate

Finding 6: MCP Servers for Microsoft/Azure Are Production-Ready

This is your best path forward for Claude orchestration.

Microsoft has released official MCP servers:

- **Azure MCP Server:** All Azure tools in one server
- **Azure DevOps MCP Server:** Full DevOps integration
- **Azure Data Explorer MCP Server:** Natural language queries

Claude-Ready Integration:

> *"Microsoft has published guidance on how to integrate Azure AI Foundry's Agent Service with Claude Desktop using the Model Context Protocol (MCP)."*

Key capability: Claude can query and manage Azure resources through natural language via MCP.

Community Option: azure-mcp on GitHub enables Claude Desktop to interact with Azure services directly.

Enterprise Security: Microsoft has published guidance on building Entra ID-protected MCP servers with Azure API Management.

Sources:

- Microsoft Official MCP Catalog
- Azure AI Foundry MCP Integration

Finding 7: HIPAA Compliance Architecture

Your Microsoft 365 BAA Coverage: The HIPAA BAA is automatically included in the Microsoft Online Services Data Protection Addendum. Services covered include:

- Office 365 / Microsoft 365
- Power Automate
- PowerApps
- Power BI
- Dynamics 365
- SharePoint Online
- Teams

Azure Claude Coverage: Claude models on Azure Foundry inherit Azure's HIPAA compliance when:

- Deployed in HIPAA-compliant regions
- Accessed via Azure API endpoints (not Anthropic direct)
- Used with proper encryption, access controls, and logging

Important Distinction:

> *"Entering into a BAA does not, in itself, ensure that you are HIPAA compliant. You can work with PHI in Office 365 in many ways that are not compliant."*

Required Security Controls:

Control	Implementation
Access Management	RBAC, Entra ID, Conditional Access, MFA

Data Security	Encryption at rest/transit, Key Vault for BYOK
Threat Detection	Defender for Cloud, Azure Monitor, Log Analytics
Compliance Tracking	Microsoft Compliance Manager with HIPAA template

Source: [Microsoft HIPAA Compliance]()

Technical Details

Recommended Architecture Pattern

Instead of user impersonation via [autopilot@southviewteam.com](), implement:

```
┌─────────────────────────────────────────────────────────────────┐
│                    Your Application                              │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│  ┌──────────────────┐    ┌──────────────────────────────────┐  │
│  │ Employee Auth    │    │ Backend Service Principal        │  │
│  │ (Delegated)      │    │ (Application Permissions)        │  │
│  │                  │    │                                  │  │
│  │ - User sign-in   │    │ - Meeting management             │  │
│  │ - OneNote write  │    │ - Transcript retrieval           │  │
│  │ - Personal files │    │ - SharePoint operations          │  │
│  └────────┬─────────┘    └──────────────┬───────────────────┘  │
│           │                              │                      │
│           ▼                              ▼                      │
│  ┌─────────────────────────────────────────────────────────┐  │
│  │              Microsoft Graph API                         │  │
│  └─────────────────────────────────────────────────────────┘  │
│                                                                  │
└─────────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────────┐
│                    Claude/MCP Layer                              │
├─────────────────────────────────────────────────────────────────┤
│  ┌──────────────────┐    ┌──────────────────────────────────┐  │
│  │ Azure Claude     │◄──►│ Azure MCP Server                 │  │
│  │ (via Foundry)    │    │                                  │  │
│  │ HIPAA-covered    │    │ - Azure resource management      │  │
│  └──────────────────┘    │ - Graph API orchestration        │  │
│                          │ - DevOps automation              │  │
│                          └──────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────────┘
```

Service Principal Setup for Your Use Cases

1. Meeting Management Service Principal:

```
{
  "appPermissions": [
    "OnlineMeetings.ReadWrite.All",
    "OnlineMeetingTranscript.Read.All",
    "Calendars.ReadWrite"
  ],
  "authentication": "Certificate (recommended) or Client Secret",
  "additionalConfig": "Application Access Policy required"
}
```

2. OneNote/SharePoint Service (Delegated Required post-March 2025):

```
{
  "delegatedPermissions": [
    "Notes.Create",
    "Notes.ReadWrite.All",
    "Sites.ReadWrite.All"
  ],
  "authentication": "MSAL with token caching",
  "note": "Requires user context; consider hybrid approach"
}
```

Meeting Transcript Retrieval Pattern

```python
# Correct approach: Use calendar events for meetings
async def schedule_recorded_meeting(graph_client, subject, start, end, attendees):
    """Schedule a meeting via calendar event (not onlineMeeting API directly)"""

    event = {
        "subject": subject,
        "start": {"dateTime": start, "timeZone": "Pacific Standard Time"},
        "end": {"dateTime": end, "timeZone": "Pacific Standard Time"},
        "attendees": [{"emailAddress": {"address": a}} for a in attendees],
        "isOnlineMeeting": True,
        "onlineMeetingProvider": "teamsForBusiness",
        # Meeting settings for auto-record require PATCH after creation
    }

    created_event = await graph_client.post("/me/events", json=event)

    # Enable auto-recording via separate call
    meeting_id = created_event["onlineMeeting"]["joinUrl"]  # Extract meeting ID
    await graph_client.patch(
        f"/me/onlineMeetings/{meeting_id}",
        json={"recordAutomatically": True}
    )

    return created_event
```

Recommendations

1. Migrate autopilot@southviewteam.com to Service Principal Architecture

Rationale: MFA enforcement makes user-based automation unsustainable. Service principals are explicitly exempted from MFA requirements and are Microsoft's recommended approach.

Action Items:

- [ ] Create dedicated app registration in Entra ID
- [ ] Configure service principal with certificate authentication
- [ ] Set up Application Access Policies for Teams APIs
- [ ] Migrate existing Power Automate flows to use service principal

2. Use Calendar Event API for Meeting Scheduling (Not onlineMeeting API)

Rationale: Transcript access requires calendar association. The `create event` API with `isOnlineMeeting: true` provides this automatically.

Action Items:

- [ ] Refactor meeting creation to use `/me/events` or `/users/{id}/events`
- [ ] PATCH meeting settings for auto-record after event creation
- [ ] Test transcript retrieval with new meeting creation pattern

3. Implement Hybrid Auth for OneNote Publishing

Rationale: App-only auth for OneNote ends March 2025. You'll need delegated access.

Options:

- **Option A:** Interactive auth with long-lived refresh tokens (requires periodic user login)
- **Option B:** On-behalf-of flow where backend exchanges user token for Graph access
- **Option C:** Publish to SharePoint document library instead (retains app-only capability)

4. Deploy Azure MCP Server for Claude Orchestration

Rationale: This is the most sustainable path to letting Claude manage Microsoft workflows without building custom integrations.

Action Items:

- [ ] Set up Azure MCP Server from Microsoft's official catalog
- [ ] Configure Entra ID protection per Microsoft's APIM guidance
- [ ] Test Graph API orchestration via Claude Desktop first
- [ ] Scale to Azure Foundry Claude for HIPAA-covered PHI workflows

5. Consolidate HIPAA Compliance Controls

Rationale: BAA coverage is necessary but not sufficient. Implement defense-in-depth.

Action Items:

- [ ] Enable Microsoft Compliance Manager with HIPAA template
- [ ] Configure Conditional Access policies for all Graph API access

- [ ] Enable audit logging for all PHI-touching operations
- [ ] Document data flows for HIPAA risk assessment

Risks & Considerations

Risk	Impact	Mitigation
OneNote app-only deprecation (March 2025)	Report publishing breaks	Migrate to delegated auth or SharePoint alternative
MFA enforcement blocks autopilot user	Automation workflows fail	Migrate to service principal before October 2025
Azure AD Graph retirement (June 2025)	Legacy integrations fail	Audit all integrations; migrate to Microsoft Graph
Transcript API limitations	Cannot access ad-hoc meeting transcripts	Use calendar event creation pattern consistently
MCP server security	PHI exposure via natural language interface	Implement APIM gateway with Entra ID protection
Channel meeting transcripts	Limited API support	Accept limitation or use webhook notifications

---

Next Steps

- [ ] **Audit existing integrations** for Azure AD Graph usage (deadline: June 2025)
- [ ] **Create service principal** with appropriate permissions as replacement for autopilot user
- [ ] **Test meeting creation** via calendar event API with auto-record settings
- [ ] **Evaluate MCP server deployment** for Claude orchestration pilot
- [ ] **Configure Application Access Policies** via Teams PowerShell for transcript access
- [ ] **Document HIPAA controls** in Microsoft Compliance Manager
- [ ] **Plan OneNote migration** before March 2025 app-only auth deprecation

---

Sources & References

1. Microsoft Entra Breaking Changes - Official deprecation timeline
2. Azure AD Graph API Retirement Update - June 2025 deadline
3. Microsoft Graph Transcripts API - Calendar event requirements
4. Securing Service Principals - Best practices
5. Best Practices for Microsoft Entra - MFA enforcement details
6. OneNote API Overview - March 2025 app-only deprecation
7. Microsoft Official MCP Catalog - Azure MCP servers
8. Azure AI Foundry MCP Integration - Claude integration guide
9. Claude-Ready MCP with APIM - Enterprise security
10. Microsoft HIPAA Compliance - BAA coverage details
11. Enable Auto-Recording via Power Automate - Custom connector approach
12. Microsoft 365 Copilot APIs - Future orchestration options

13. Power Platform Service Accounts vs Service Principals - Architecture guidance

---

Addendum A: Step-by-Step Implementation Guides

Added: 2025-12-04

This addendum provides detailed implementation guides for configuring MCP servers with Microsoft Entra ID for Claude orchestration, plus specific permission configurations for your transcript and OneNote workflows.

Clarification: Microsoft Graph vs Azure AD Graph

You are using the CORRECT API. When you add permissions in Azure App Registration and select "Microsoft Graph", that is the current, supported API.

API	Endpoint	Azure Portal Label	Status
Microsoft Graph	`graph.microsoft.com`	"Microsoft Graph"	✅ Current
Azure AD Graph	`graph.windows.net`	"Azure Active Directory Graph"	❌ Deprecated (dies June 2025)

Both portals are equivalent:

- Azure Portal → Microsoft Entra ID → App registrations
- Entra Admin Center (entra.microsoft.com) → Applications → App registrations

Same backend, same result.

Option A: Azure API Management Gateway (Production/HIPAA)

This is Microsoft's recommended approach for enterprise deployments requiring HIPAA compliance. APIM acts as an OAuth 2.0 gateway between Claude and your MCP server.

Architecture

```
┌─────────────────┐     ┌──────────────────────┐     ┌─────────────────┐
│  Claude         │────▶│  Azure API Management │────▶│  MCP Server     │
│  (Max/Team/     │     │  (OAuth Gateway)      │     │  (Azure Func)   │
│   Enterprise)   │     └──────────────────────┘     └─────────────────┘
└─────────────────┘                │                          │
                                   │                          │
                                   │                          │
                        ┌──────────▼───────────┐              │
                        │  Microsoft Entra ID   │◀─────────────┘
                        │  (Identity Provider)  │
                        └──────────────────────┘
```

Prerequisites

- Azure subscription with resource creation permissions

- Claude Max, Team, or Enterprise plan (required for custom integrations)
- Azure Developer CLI (azd) installed
- PowerShell with MicrosoftTeams module

Step 1: Deploy the Reference Solution

```
# 1. Clone Microsoft's reference implementation
git clone https://github.com/Azure-Samples/remote-mcp-apim-functions-python
cd remote-mcp-apim-functions-python

# 2. Login to Azure
azd auth login

# 3. Initialize the environment (creates resource group, etc.)
azd init

# 4. Deploy all resources (APIM, Functions, Entra ID app registration)
azd up
```

This deploys:

- Azure API Management instance (OAuth gateway)
- Azure Functions (MCP server backend)
- Entra ID app registration (pre-configured)
- All necessary networking and security settings

Step 2: Note the Deployed Resources

After deployment, note these values from the Azure Portal:

```
APIM Endpoint URL: https://<your-apim-name>.azure-api.net/mcp
Client ID: <auto-created-app-registration-id>
Tenant ID: <your-tenant-id>  (southviewteam.com)
```

Step 3: Configure Additional Graph API Permissions

The reference solution includes basic permissions. Add your specific requirements:

1. Navigate to **Azure Portal** → **Microsoft Entra ID** → **App registrations**
2. Find the app created by `azd up` (usually named after your project)
3. Go to **API permissions** → **Add a permission** → **Microsoft Graph**
4. Add the permissions from the table in "Permission Matrix" section below
5. Click **Grant admin consent for southviewteam.com**

Step 4: Configure Application Access Policy for Teams

```
# Install Teams module if not present
Install-Module -Name MicrosoftTeams -Force -AllowClobber

# Connect with admin credentials
Connect-MicrosoftTeams
```

```
# Create policy for your MCP app (use Client ID from Step 2)
New-CsApplicationAccessPolicy `
    -Identity "MCP-Claude-Integration-Policy" `
    -AppIds "<your-client-id>" `
    -Description "Allow Claude MCP server to access Teams meetings and transcripts"

# Option A: Grant to specific users (recommended for testing)
Grant-CsApplicationAccessPolicy `
    -PolicyName "MCP-Claude-Integration-Policy" `
    -Identity "autopilot@southviewteam.com"

# Option B: Grant tenant-wide (for production)
Grant-CsApplicationAccessPolicy `
    -PolicyName "MCP-Claude-Integration-Policy" `
    -Global
```

Note: Policy changes take up to 30 minutes to propagate.

Step 5: Connect Claude to Your MCP Server

1. Open Claude Desktop or go to claude.ai
2. Navigate to **Settings** → **Integrations** (or organizational settings for Team/Enterprise)
3. Click **Add More**
4. Enter your APIM endpoint URL: `https://<your-apim-name>.azure-api.net/mcp`
5. Click **Connect**
6. Complete the Entra ID authentication flow (sign in with your Microsoft account)
7. Consent to the requested permissions

The integration is now available to your team.

Step 6: Verify the Connection

In Claude, try:

```
"List my upcoming Teams meetings"
"Show me recent transcripts from my meetings"
"Create a calendar event for tomorrow at 2pm with auto-recording enabled"
```

Option B: Local MCP Server with Claude Desktop (Development)

For development, testing, or non-HIPAA scenarios where you want faster iteration.

Prerequisites

- Node.js 18+ installed
- Claude Desktop installed
- Azure subscription (for app registration only)
- PowerShell with MicrosoftTeams module

Step 1: Create App Registration in Entra ID

1. Go to **Azure Portal** → **Microsoft Entra ID** → **App registrations**
2. Click **New registration**

3. Configure:

```
Name: Claude-MCP-Local-Dev
Supported account types: Accounts in this organizational directory only
                        (southviewteam.com only - Single tenant)
Redirect URI: (leave blank for now)
```

4. Click **Register**
5. Note the **Application (client) ID** and **Directory (tenant) ID**

Step 2: Configure Authentication Settings

1. In your app registration, go to **Authentication**
2. Click **Add a platform** → **Mobile and desktop applications**
3. Add redirect URI: `http://localhost`
4. Under **Advanced settings**:
   - ✅ Enable **Allow public client flows** = Yes
5. Click **Save**

Step 3: Add API Permissions

Go to **API permissions** → **Add a permission** → **Microsoft Graph**

Add these **Delegated permissions**:

Permission	Purpose
`User.Read`	Basic profile info
`Calendars.ReadWrite`	Create calendar events with Teams meetings
`OnlineMeetings.ReadWrite`	Manage meeting settings (auto-record, etc.)
`OnlineMeetingTranscript.Read.All`	Read meeting transcripts
`Notes.ReadWrite.All`	OneNote publishing (requires user context)
`Sites.ReadWrite.All`	SharePoint access for team notebooks
`Files.ReadWrite.All`	File access in SharePoint/OneDrive

Add these **Application permissions** (for background operations):

Permission	Purpose
`OnlineMeetings.ReadWrite.All`	Background meeting management
`OnlineMeetingTranscript.Read.All`	Background transcript retrieval
`Calendars.ReadWrite`	Background calendar access

Click **Grant admin consent for southviewteam.com**.

Step 4: Create Client Secret

1. Go to **Certificates & secrets**
2. Click **New client secret**

3. Description: `Claude MCP Local Dev`
4. Expires: 24 months (or your preference)
5. Click **Add**
6. **IMMEDIATELY copy the Value** (shown only once)

Step 5: Configure Application Access Policy

```
# Connect to Teams
Connect-MicrosoftTeams

# Create policy
New-CsApplicationAccessPolicy `
    -Identity "MCP-Local-Dev-Policy" `
    -AppIds "<your-client-id>" `
    -Description "Local MCP development access"

# Grant to your test user
Grant-CsApplicationAccessPolicy `
    -PolicyName "MCP-Local-Dev-Policy" `
    -Identity "autopilot@southviewteam.com"
```

Step 6: Install MCP Server Package

Choose one of these MCP server implementations:

Option A: Microsoft Official (Azure-focused)

```
npm install -g @microsoft/mcp-server-azure
```

Option B: Community Graph API Server (more Graph features)

```
npm install -g @anthropic/mcp-server-graph
```

Option C: Direct from GitHub

```
# For Outlook/Calendar/Teams focus
npx -y github:ryaker/outlook-mcp

# For general Entra ID operations
npx -y github:uniQuk/mcp-entra
```

Step 7: Configure Claude Desktop

Edit the Claude Desktop configuration file:

Windows: `%APPDATA%\Claude\claude_desktop_config.json` **macOS:** `~/Library/Application Support/Claude/claude_desktop_config.json`

```
{
  "mcpServers": {
    "microsoft-graph": {
      "command": "npx",
      "args": ["-y", "github:ryaker/outlook-mcp"],
      "env": {
        "AZURE_TENANT_ID": "<your-tenant-id>",
        "AZURE_CLIENT_ID": "<your-client-id>",
        "AZURE_CLIENT_SECRET": "<your-client-secret>",
        "GRAPH_USER_ID": "autopilot@southviewteam.com"
      }
    }
  }
}
```

Step 8: Restart Claude Desktop and Test

1. Completely quit Claude Desktop (check system tray)
2. Relaunch Claude Desktop
3. Look for the MCP server indicator (hammer icon or "microsoft-graph" in tools)
4. Test with: `"What meetings do I have this week?"`

Permission Matrix: Your Specific Workflows

Workflow 1: Meeting Transcript Retrieval

Scenario: Pull transcripts from Teams meetings that were automatically recorded.

Permission	Type	Required	Notes
`OnlineMeetingTranscript.Read.All`	Delegated	✅	For signed-in user's meetings
`OnlineMeetingTranscript.Read.All`	Application	✅	For background/batch processing
`OnlineMeetings.Read.All`	Application	✅	To list meetings
Application Access Policy	PowerShell	✅	Required for application permissions

Critical Constraint: Meetings must be created via calendar event (not `create onlineMeeting` API).

API Call Pattern:

```
# 1. Get meetings for a user (requires Application Access Policy)
GET https://graph.microsoft.com/v1.0/users/{user-id}/onlineMeetings

# 2. List transcripts for a specific meeting
GET https://graph.microsoft.com/v1.0/users/{user-id}/onlineMeetings/{meeting-id}/transcripts

# 3. Get transcript content
GET https://graph.microsoft.com/v1.0/users/{user-id}/onlineMeetings/{meeting-
id}/transcripts/{transcript-id}/content?$format=text/vtt
```

Workflow 2: OneNote Publishing to Teams Notebook

Scenario: Generate reports and publish to a OneNote notebook visible to a Teams team.

Permission	Type	Required	Notes
`Notes.ReadWrite.All`	Delegated	✅	Required after March 2025
`Notes.Create`	Delegated	Optional	Subset of ReadWrite.All
`Sites.ReadWrite.All`	Delegated	✅	For SharePoint-hosted notebooks

⚠️ **March 2025 Change:** App-only authentication for OneNote API ends. Must use delegated auth.

API Call Pattern:

```
# 1. Get the SharePoint site ID for your Team
GET https://graph.microsoft.com/v1.0/groups/{team-id}/sites/root

# 2. Get the notebook in that site
GET https://graph.microsoft.com/v1.0/sites/{site-id}/onenote/notebooks

# 3. Create a page in a specific section
POST https://graph.microsoft.com/v1.0/sites/{site-id}/onenote/sections/{section-id}/pages
Content-Type: text/html

<!DOCTYPE html>
<html>
<head><title>Report: Daily Summary</title></head>
<body>
  <h1>Daily Summary Report</h1>
  <p>Generated: 2025-12-04</p>
  <!-- Your report content -->
</body>
</html>
```

Workflow 3: Schedule Meeting with Auto-Record

Scenario: Programmatically create a Teams meeting with auto-record and transcription enabled.

Permission	Type	Required	Notes
`Calendars.ReadWrite`	Delegated or Application	✅	Create calendar event
`OnlineMeetings.ReadWrite`	Delegated or Application	✅	Modify meeting settings

API Call Pattern:

```
# 1. Create calendar event with Teams meeting
POST https://graph.microsoft.com/v1.0/users/{user-id}/events
Content-Type: application/json

{
  "subject": "Weekly Team Sync",
  "start": {"dateTime": "2025-12-05T14:00:00", "timeZone": "Pacific Standard Time"},
  "end": {"dateTime": "2025-12-05T15:00:00", "timeZone": "Pacific Standard Time"},
  "attendees": [
    {"emailAddress": {"address": "employee@southviewteam.com"}}
  ],
  "isOnlineMeeting": true,
  "onlineMeetingProvider": "teamsForBusiness"
}

# 2. Get the online meeting ID from the response
# Response includes: "onlineMeeting": {"joinUrl": "https://teams.microsoft.com/..."}

# 3. PATCH the meeting to enable auto-record (requires meeting ID extraction)
PATCH https://graph.microsoft.com/v1.0/users/{user-id}/onlineMeetings/{meeting-id}
Content-Type: application/json

{
  "recordAutomatically": true
}
```

Note: The `recordAutomatically` setting requires the meeting organizer to have a Teams Premium license or the tenant to have meeting recording enabled.

Workflow 4: SharePoint Alternative to OneNote (Post-March 2025)

If OneNote delegated auth is problematic, consider publishing reports as files to SharePoint:

Permission	Type	Required	Notes
`Sites.ReadWrite.All`	Application	✅	Works with app-only auth
`Files.ReadWrite.All`	Application	✅	Upload files

API Call Pattern:

```
# Upload a report file to a SharePoint document library
PUT https://graph.microsoft.com/v1.0/sites/{site-id}/drive/root:/Reports/2025-12-
04_DailyReport.pdf:/content
Content-Type: application/pdf

<binary PDF content>
```

This approach retains app-only capability and files are accessible via Teams Files tab.

Complete Permission Summary

Minimum Permissions for All Your Workflows

Delegated Permissions (for interactive user scenarios):

```
User.Read
Calendars.ReadWrite
OnlineMeetings.ReadWrite
```

```
OnlineMeetingTranscript.Read.All
Notes.ReadWrite.All
Sites.ReadWrite.All
Files.ReadWrite.All
```

Application Permissions (for background automation):

```
Calendars.ReadWrite
OnlineMeetings.ReadWrite.All
OnlineMeetingTranscript.Read.All
Sites.ReadWrite.All
Files.ReadWrite.All
```

Additional Configuration:

```
# Application Access Policy for Teams APIs
New-CsApplicationAccessPolicy -Identity "MCP-Policy" -AppIds "<client-id>"
Grant-CsApplicationAccessPolicy -PolicyName "MCP-Policy" -Global
```

OnlineMeetingTranscript.Read.All vs .Read.Chat

Aspect	OnlineMeetingTranscript.Read.All	OnlineMeetingTranscript.Read.Chat
Scope	Tenant-wide (all users' meetings)	Only meetings where Teams app is installed
Permission Type	Standard Graph API (Azure AD)	Resource-Specific Consent (RSC)
Admin Consent	Required from tenant admin	Users can consent per-meeting
Meeting Types	All scheduled meetings	Private chat meetings only
Configuration	Azure Portal + Application Access Policy	Teams app manifest (JSON)
Use Case	Background automation, MCP servers	Teams bot/app installed in specific meetings

For your MCP server scenario: Use `OnlineMeetingTranscript.Read.All` with Application Access Policy.

Troubleshooting

"Application is not allowed to perform operations on the user"

Cause: Application Access Policy not configured or not propagated.

Fix:

```
# Verify policy exists
Get-CsApplicationAccessPolicy

# Verify policy is granted
Get-CsOnlineUser -Identity "autopilot@southviewteam.com" | Select ApplicationAccessPolicy
```

```
# Wait 30 minutes after policy changes
```

"Transcript not found" for a meeting that had transcription

Causes:

1. Meeting was created via `create onlineMeeting` API (not calendar event)
2. Graph hasn't indexed the transcript yet (wait 5-10 minutes after meeting ends)
3. User in API path isn't the meeting organizer

Fix: Ensure meetings are created via calendar event API with `isOnlineMeeting: true` .

OneNote API returns 401 after March 2025

Cause: App-only authentication deprecated.

Fix: Switch to delegated authentication with user sign-in flow, or use SharePoint file upload as alternative.

MCP server not appearing in Claude Desktop

Causes:

1. Config file syntax error (JSON)
2. Environment variables not set correctly
3. Claude Desktop not fully restarted

Fix:

```
# Validate JSON syntax
npx jsonlint %APPDATA%\Claude\claude_desktop_config.json

# Check Claude logs
# Windows: %APPDATA%\Claude\logs\
# macOS: ~/Library/Logs/Claude/
```

Sources & References (Addendum A)

14. OnlineMeetingTranscript.Read.All Permission Details - Scope comparison
15. Configure Application Access Policy - Official guide
16. Grant-CsApplicationAccessPolicy - PowerShell reference
17. Notes.ReadWrite.All Permission - OneNote scope details
18. MCP Server Authorization for Azure App Service - PRM configuration
19. Building Claude-Ready MCP with APIM - Enterprise architecture
20. Outlook MCP Server - Community Graph implementation

Related Reports

- **Teams Bookings & Virtual Visits for Telehealth**: See separate report `2025-12-04_research_teams-bookings-virtual-visits-telehealth.md` for detailed guidance on configuring Bookings with autopilot@southviewteam.com as admin, Virtual Appointments setup, and telehealth-specific HIPAA considerations.

Addendum A added: 2025-12-04 Research conducted: 2025-12-04 Generated by Claude via /research-topic command