Account Protection Local Group Membership Automation with Azure Function

This solution enables targeted, temporary local admin access on Windows devices managed through Intune. It uses an Azure Function (PowerShell) to dynamically create per-device account protection policies and group assignments based on a user-device pair. The design ensures that repeated requests with the same parameters do not produce duplicate configurations (idempotence), making it safe, reliable, and easy to integrate into workflows.

Use case

This automation is designed for longer-term scenarios, where users need local administrator rights for weeks or months on a specific device. It is particularly useful in cases where standard Intune assignments would otherwise affect too many devices, or where multiple tenants must be supported through a unified automation. Because the process creates and manages device-specific groups and policies, it ensures that repeated requests for the same user and device do not result in duplicate assignments.

For short-term or ad-hoc elevation (minutes or hours), other solutions are better suited, such as Intune Endpoint Privilege Management (EPM), MakeMeAdmin, or similar tools.

Prerequisites & Resources

Since this is an Intune automation it is very important that you already have some sort of Intune configuration and devices in place. This guide assumes that you already use Intune Account Protection for deploying permission management of local groups and accounts in Windows. This automation adds the capability of granularly controlling permissions on a per device level in addition to replacing the memberships of local groups for security and compliance view. However, this also requires a **global Account Protection policy** to be available for all/majority of devices. Since the global policy and device specific policies cannot be combined, an **Entra ID exclusion group** must be in place.

Azure Resources

This automation is deployed as an **Azure Function**. At minimum, you need the following resources:

- **Resource Group** – to contain all components.
- **Function App (PowerShell)** – hosts the scripts. Use Consumption or Premium plan.
- **Storage Account** – required for Function state and triggers. Is created automatically while deploying an Azure Function App.
- **Application Insights** – for telemetry and monitoring (not required but recommended).
- **Managed Identity** – for secure authentication to Microsoft Graph. For permissions view next chapter.

Graph API Permissions

Your Bearer access token for the Microsoft Graph API needs at least the following scopes. If you are deploying the solution with the recommended approach using a managed identity, then grant the following permissions:

- Device.Read.All
- Group.ReadWrite.All
- Directory.ReadWrite.All
- DeviceManagementManagedDevices.Read.All
- DeviceManagementConfiguration.ReadWrite.All

When you want to use the solution with multiple tenants and/or delegated permissions, you can also use existing access tokens while calling the API endpoint. More on the authentication for that approach in this chapter: Authentication and token handling

Azure Functions

In an Azure Function app we have to deploy two Azure Functions. One PowerShell code is to create the groups, devices, assignments, exclusions, and policies (`Automation-LocalAdminGrant.ps1`). The second function is responsible for revoking the access and reverting the steps that the grant automation is doing (`<a href="https://github.com/lucanoahcaprez/lnc-docs-resources/blob/main/microsoft-intune/Automation-LocalAdminRevoke.ps1">Automation-LocalAdminRevoke.ps1</a>`). All the code is provided in the Github repository of LNC DOCS.

- `Automation-LocalAdminGrant.ps1`: Automation for granting and creating the permission policies.
- `Automation-LocalAdminRevoke.ps1`: Second automation for reverting the steps (removing the permissions through policy change).

Workflow & Architecture

This chapter describes the workflow and architecture of the two parts (grant & revoke) of the solution. The automation enables per-device exceptions to a global account protection policy which handles local admin rights on Windows clients. It does so by creating device-specific Entra ID groups, updating Intune configuration policies, and managing membership in a global exclusion group.

Important: The automation is implemented in a way that it supports idempotence. This means that multiple requests with the same parameters (UserPrincipalName, DeviceName) will always result in the same outcome, preventing duplicate tasks or conflicting states.

Automation process

This is how the automation processes your request. Between "Grant" and "Revoke" part, the automation is almost similar. The main difference is, that the automation creates or deletes the objects/memberships according to existing environment and requests.

1. Input validation
   - When the Function receives an HTTP request (POST) with parameters UserPrincipalName and DeviceName it validates the provided JSON object.
   - If the validation is rejecting the input, the automation returns 400 codes.
2. Authentication (choose token source)
   - If a parameter with the name `MicrosoftEntraIDAccessToken` is provided in the JSON body of the POST request, this token is used for authentication.
   - Another option is to optain a token from the Managed Identity via IMDS (Azure Instance Metadata Service). This needs to be configured correctly.
3. Resolve device
   - Next the automation queries the Intune managed devices with a filter on DeviceName (prefix match).
   - It ensures that there is an exact match. It aborts on zero or multiple responses.
4. Resolve Entra ID device
   - Afterwards it queries the Microsoft Entra ID devices using the Intune device's AzureADDeviceId.
5. Prepare naming and resources
   - Then it builds policy and group names from device serial (e.g. Windows-COPE-LUSRMGR- for Account Protection policy and MEID-INT-Windows-LUSRMGR- for Entra ID group).
   - Resolves ExclusionGroupID for excluding the device from the global Account Protection policy.
6. Create/Ensure resources
   - It creates a device-specific Entra ID security group if not present.
   - Then it adds the device to the previously created/verified group.
   - In the end it adds the device to the global exclusion group.
7. Create/Update account protection policy
   - First it checks if a policy with the constructed name already exists.
   - If not present, the automation creates a configuration policy that grants the specified user local admin (via AzureAD<UserPrincipalName> reference) on the

target device group.
    - If the policyname is already present, the policy members are updated if needed.
8. Final validation
    - Last but not least it verifies that the device is in the correct groups and the policy is assigned.
    - The Azure Function returns success (200) or a helpful error message describing which step failed.
9. Revoke path (Automation-LocalAdminRevoke)
    - To reverse the actions you can call the "revoke" automation. It remove the user from the policy, removes the policy if empty, removes the device from the exclusion group and deletes the device-specific group if it is empty.

Authentication and token handling

Authentication is required for the function to call Microsoft Graph and apply changes on the Intune platform. The automation supports two flows: using a **Managed Identity**, which is the recommended production setup, or accepting a **Microsoft Entra ID access token** directly in the request body. Both methods provide secure access, but Managed Identity avoids handling tokens and simplifies permission management. For eighter method make sure to include the right permissions (see chapter "Graph API Permissions").

- **Preferred:** Use a Managed Identity (system- or user-assigned) on the Function App.
    - Make sure to enable and configure the managed identity correctly: Use of System Managed Identities
    - Learn how to grant Microsoft Graph API permissions to a Managed Identity: Assign Graph API Permissions
    - If everything is in place and the PowerShell gets executed, the code requests a token from Azure Instance Metadata Service (Learn more about IMDS):

```
GET http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://graph.microsoft.com/
```

- **Alternative:** Provide an access token in the JSON request body with the name of `MicrosoftEntraIDAccessToken`.
    - The function will then only use this token for calling the Microsoft Graph API endpoints.
    - This method could come in handy, when using multiple tenants or while implementing or testing the automation.
    - A lot of times the permissions for Managed Identities are granted by another internal team (needs Global Admin for initial grant) or they need to be verified by an identity and/or security team. In this case the option with access token enables you to check if the solution meets your needs.
    - **Note:** Do not log or expose your access token in logs or external monitoring.

Usage & Implementation

This section guides you through deploying and using the automation. You'll learn how to set up the Azure Function, configure authentication, and call its HTTP interface. Examples will show you how to format requests and interpret responses, enabling you to integrate the solution into your workflows easily.

Deployment steps

1. Open Azure Portal and search your Function App. Create a new resource if needed.
2. Deploy the Azure Function PowerShell code from GitHub (replace your function content or create new function).
3. In the Function App -> Identity -> enable System assigned Managed Identity.
4. For granting the permissions to the Managed Identity follow this guide: Assign Graph API Permissions. Currently it is not supported to add the permissions through the Entra ID admin center.
5. Got to Entra ID admin center -> Enterprise applications -> find the Managed Identity (service principal) and confirm that the Graph API permissions are visible and consented.
6. Run the function the first time using eighter PowerShell, a REST client (postman, Thunder Client, etc.) or from the Azure Portal.

HTTP API Usage

The Local Group Membership Automation exposes a simple **HTTP API** that can be called from other systems, scripts, or manual tools. Its main purpose is to grant and verify temporary local administrator rights for a user on an Intune-managed device. The API is lightweight and designed so that both technical and non-technical staff can trigger the operation with minimal input: just the user's sign-in name and the device name.

Base URL (example): `https://<your-function>.azurewebsites.net/api/Automation-LocalAdminGrant`
Replace `<your-function>` with your Function App name.
Method: `POST`
Request body (JSON):

- `UserPrincipalName` *(string, required)* – the user's user principal name (e.g., `test@example.com`).
- `DeviceName` *(string, required)* – the name of the Windows device (e.g., `Device-123`).
- `MicrosoftEntraIDAccessToken` *(string, optional)* – Only include this if you already have an access token. Otherwise the Function uses its Managed Identity.

Successful response: HTTP `200` with body `FINAL CHECK: SUCCESS`
Errors: HTTP `400` / `401` / `500` depending on validation or token issues.

PowerShell Usage

```
# Replace with your function URL and details
$Url  = "https://<your-function>.azurewebsites.net/api/LocalGroupMembership?code=<function-key>"
$Body = '{"UserPrincipalName":"test@example.com","DeviceName":"Device-123"}'

Invoke-RestMethod -Method Post -Uri $Url -ContentType "application/json" -Body $Body
```

Results and verification

Successful run outcomes:

- A device-specific Entra ID security group created: `MEID-INT-Windows-LUSRMGR-<serial>`
- Device added to:
  - The device-specific group
  - The global exclusion group (`ExclusionGroupID`)
- A device-targeted Account Protection configuration policy created or updated: `Windows-COPE-LUSRMGR-<serial>`
- The policy assigned to the device-specific group
- Function returns `FINAL CHECK: SUCCESS` or a structured OK response

Verification steps:

- Verify the existence of the group via Graph API or Entra ID admin center .
- Check the device is member of both the device-specific group and the exclusion group.
- Inspect the configuration policy in Intune (Endpoint security -> Account protection) for the device-targeted policy and assigned targets.
- Use Graph API or Intune admin center to confirm policy settings include `AzureAD\<UserPrincipalName>` membership.
- Review Function App logs / Application Insights for detailed operation traces.

Cleanup / Revoke verification:

- When revoke-automation is triggered, confirm that this is done:
  - The user is removed from the policy setting values.
  - If the policy contains no other exception members, the policy is deleted.
  - The device is removed from the exclusion group.
  - The device-specific group is deleted if empty.

Error states, tips and tricks

- Common failures:
  - Authentication errors (401): check managed identity and admin consent for Graph scopes.

    - Device lookup errors: ensure `DeviceName` uniquely identifies a single managed device.
    - Graph `ResourceNotFound` when adding/removing members: ensure correct object IDs and that you use the right endpoint (`/groups/{id}/members/$ref` for add, `/groups/{id}/members/{memberId}/$ref` for delete).
- Idempotency:
    - Functions perform existence checks before creating resources; repeated runs for same device/user pair should be safe.
- Observability:
    - Enable Application Insights, log full request/flow errors (avoid logging tokens), and emit structured events for success/failure.

Revision #20
Created 2025-10-04 14:35:49 UTC by Luca Noah Caprez
Updated 2025-10-04 16:57:29 UTC by Luca Noah Caprez