GDAP and Secure Application Model (SAM)

The Secure Application Model is based on the Microsoft identity Platform - Application model, aligning Partner Center security with best practices. GDAP further expands that alignment. However, with the introduction of a minimum rights concept in GDAP the legacy behavior exposes a security gap that makes full backwards compatibility problematic. Previously, DAP would project all application consents into a customer tenant without review, exposing the customer tenant in a way that was difficult for the partner to govern, and potentially escalating non-privileged users and tasks to a privileged context. This was called “DAP pre-consent”. The risk for using “pre-consent” is further amplified since the customer cannot review or audit the consents since they have no visibility in AAD under “Enterprise Applications” showing what applications have been consented in their tenant. The legacy behavior cannot be aligned with the Microsoft Identity Platform in the context of a minimum-rights concept.

In the following sections we will outline our recommended approach to fully automate the migration of the legacy behavior to GDAP’s minimum-rights and time-bound framework.

GDAP requires that your CPV/CSP applications are “consented to” in the customer’s tenant, providing traceability for the customer and aligning to Microsoft Identity Platform Application model.

- **Q:** Does GDAP require that I create specific relationship/security group for my Application Service Principal?
- **A:** You ***do not*** need a GDAP specific relationship nor Security Group for your Application SP.

Application Permissions and User AAD Roles

Applications have “permissions”. Permissions are assigned to an application in AAD on the “API permissions” tab. You cannot add AAD Roles to an application using GDAP relationships or through inclusion in a security group. (Excluding the DAP Admin Agent consent model which is being deprecated)

Users have Roles that are assigned directly using the Azure Portal or indirectly through Group Assignment. Users cannot have application permissions.

Application Service Model

Application Model	Notes	App Consent	OBO
Multi-Tenant Application Only	**Not Currently Supported**	Yes	No
Multi-Tenant Application + User		Yes	No
Multi-Tenant Application + User + On Behalf Of (OBO)	Requires a GDAP Relation Ship	Yes	Yes

Application Consent

The process of a resource owner granting authorization to a client application, to access protected resources under specific permissions, on behalf of the resource owner.

Explicit multi-tenant application-only consent

Explicit application-only consent to a customer tenant is not supported for third party application developers using GDAP. Such application-only consent is not compatible with the time-limited minimum-rights contract that GDAP manages.

Explicit multi-tenant application + user consent

Partner Center has provided an API that allows a CPV/CSP to consent their application through automation in the customer's tenant. Explicit app + user consent is required for every customer tenant. The app + user pattern will respect the time-limited minimum-rights contract that GDAP manages.

- REST API for automating explicit CPV/CSP application+user consent in customer tenant
- PowellShell cmdlet for calling the above API

This API requires that the calling partner user is a member of a security group that partakes in a GDAP relationship with the target customer/tenant. The GDAP relationship and associated security group must include at least ONE of these roles:

- Global Administrator
- Privileged Role Administrator – (TBD)
- Cloud Application Administrator – (TBD)
- Application Administrator – (TBD)

On Behalf Of (OBO) in Secure Application Model

OBO requires that one or more GDAP relationships have been configured and active. Further, you will need to create or use an existing AAD user account that will be assigned to one or more security groups that are associated with one or more customer GDAP relationships.

1. Create a new or use an existing user; Example: **AdminOnBehalfOf@contoso.onmicrosoft.com**; that will be used as the CPV/CSP OBO Account
2. Create a security group; Example: **AdminOnBehalfOfSG** that will be associated with your Customers GDAP Relationship.
3. Add your user: **AdminOnBehalfOf@contoso.onmicrosoft.com** into the **AdminOnBehalfOfSG**
4. Define the GDAP Relationship parameters (You'll use this later)
   - Duration In Days
   - Requested Azure AD Roles needed for the OBO
   - Note that when you do want to automate the application consent as described in the preceding section, the GDAP relationship should include the roles eligible to do the consent operation using the API
5. CPV/CSP define a new or use an existing Multi-Tenant Application created in the Partner tenant
   - Record the Application ID; Example: A5000000-F000-4000-8000-300000000000
6. Generation and renewal of your OBO Refresh Token created using the multi-tenant application + AdminOnbehalfOf user identity generated in step #1.
   - See here for examples on how to create the respective token using on-time interactive sign-in. Ensure that when signing in this happens using MFA since the refresh token must be created using MFA to work for OBO scenarios:
     i. Enable the Secure App Model
     ii. C# Example: GitHub - Partner Center Secure App Model Samples

- Store your OBO User's Refresh token in a secure location that can be accessed from your multi-Tenant application. Azure Key Vault can be one option but is not a requirement.
    i. See: Azure Key Vault Overview and the Key Vault API documentation
- Ensure that the stored refresh token is renewed at least every 90 days to avoid expiration. For example, ensure you do this within your automation code or develop a standalone app/script that does renew the token. Whenever a new access token is created from the stored refresh token, a new refresh token with a 90-day lifetime is generated.
    i. .Net Example: GitHub - Partner Center Secure App Model Samples
- Whenever you need to authenticate to the customer tenant for operations, get the most recent refresh token from your secure store and use it to generate an access token (and new refresh token) for authentication.

7. *For each* customer that will be using your multi-Tenant application, respectively for each customer where you want to run activities using delegated administration, do the following:
    - Create a GDAP relationship using the information defined in step #4
    - Submit the GDAP relationship for approval
    - Once the GDAP relationship is active, assign the security group defined in step #2 with the appropriate Azure AD Roles defined in step #4
    - Make sure that your application has been consented to, in the customer's tenant. For the automatic consent approach this can be done after GDAP is established, for the manual consent a customer's admin could do this also before GDAP is set up.

Note: The refresh token generated in step 6 can be used for authentication against each customer, it is not required to go through step 6 for each customer separately. When doing operations in a customer tenant, you take the token generated initially and exchange it for a token generated against the customer tenant. This is possible given the GDAP relationship/roles and the application consent.

For setting up this for multiple customers at scale, we recommend the following:

- For new customers where no existing delegation privilege exists:
    - Include one of the prerequisite consenting roles (above) in your initial GDAP relationship request. (for example: Cloud Application Admin).
    - Alternatively, if short-term privileges delegation is preferred, evaluate if a second GDAP relationship request with a short duration (like 1 day) using a permission that allows the consent delegation (like Cloud Application Admin) can be used. This allows removing the more privileged role after consent function is verified. This second GDAP request may also be manually removed at any time.
- For customer cases for an existing delegation privilege exists:
    - Where partner has DAP, perform the automated consent in bulk (e.g., using a PowerShell script New-PartnerCustomerApplicationConsent

When migrating from DAP to GDAP using the Partner Led bulk migration tool, set up the respective roles that allow for partner to consent for migrated customers. Customer where only partial GDAP role delegations exist, the partner will request Cloud App Admin role (or similar from above role prerequisites list) for customer to manually approve.

Appendix A Consent API

This section discusses the API CPV/CSP application+user consent in customer tenant in greater detail.

Identifying Enterprise Applications and Scopes (aka permissions)

For each application you plan to consent, you will need to map the application APIs and permissions into the JSON payload. You can find your application details in the azure portal under **Azure AAD:App Registrations "your app"** and then navigate to the "API Permissions" tab.

View the website: Verify first-party Microsoft applications in sign-in reports followed the steps in AAD to filter on Microsoft Apps. You should be able to locate the Enterprise Application APIs referenced by your application and retrieve their specific application IDs, which will be assigned to the "**enterpriseApplicationId**" in the JSON example below.

Based on an example application permissions, we located the following:

Application Name	Application ID == ***enterpriseApplicationId***
Windows Azure Active Directory	00000002-0000-0000-c000-000000000000
Microsoft Graph	00000003-0000-0000-c000-000000000000

Note the required permissions you want to include for each API and include those in the scope property. The scope property can contain all (comma delimited) or a subset of permissions configured on the application that you want to consent.

Example JSON Payload:

```
{
    "applicationId": "57667d41-992a-49b0-99d8-ddf68328373f",
    "applicationGrants": [
        {
            "enterpriseApplicationId":"00000002-0000-0000-c000-000000000000",
            "scope":"Directory.ReadWrite.All"
        },
        {
            "enterpriseApplicationId":"00000003-0000-0000-c000-000000000000",
            "scope":"Directory.ReadWrite.All"
        }
    ]
}
```

NOTE: Before calling the API, you must ensure that the Auth Token is created using the same Application Id you want to consent into the customer's tenant. You can use a popular website such as https://jwt.ms to decode your bearer token to ensure your Application ID's match. In the bearer token: "appid": "57667d41-992a-49b0-99d8-ddf68328373f" should match the **applicationId** provided in the JSON payload.

Getting Permissions/Scopes Friendly Names

The following information is useful if you want to discover the friendly names for application API's permissions as displayed in the https://portal.azure.com/ AAD Application Registration section for a specific application.

Getting your applications using:

- Graph api get application

Getting a list applications' id and display name. The id is the application object id.

- https://graph.microsoft.com/v1.0/applications?$select=id,displayName

Getting a specific application's requiredResourceAccess which contains your application API and Permissions as viewed in the azure portal. Replace {applicationObjectId} with the application id from the previous step.

- https://graph.microsoft.com/v1.0/applications/**{applicationObjectId}**?$select=requiredResourceAccess

Example Response:

```
"requiredResourceAccess": [
     {
        "resourceAppId": "00000003-0000-0000-c000-000000000000",
        "resourceAccess": [
          {
            "id": "885f682f-a990-4bad-a642-36736a74b0c7",
            "type": "Scope"
          },
          {
            "id": "e1fe6dd8-ba31-4d61-89e7-88639da4683d",
            "type": "Scope"
          },
          {
            "id": "06da0dbc-49e2-44d2-8312-53f166ab848a",
            "type": "Scope"
          },
          {
            "id": "c5366453-9fb0-48a5-a156-24f0c49a4b84",
            "type": "Scope"
          }
        ]
     }
```

Getting service principals:

- Graph api get serviceprincipal

Getting a specific service principal's oauth2PermissionScopes. Replace {resourceAppId} with the value returned in the requiredResourceAccess array (there may be more than one).

- https://graph.microsoft.com/v1.0/servicePrincipals?$filter=appid eq '**{resourceAppId}**'&$select=oauth2PermissionScopes
- You will find the scope definitions by id in the response and the friendly name is stored in the value property.

Example Edited Response (reduced the size for clarity)

```
{
  "adminConsentDisplayName": "Manage Delegated Admin relationships with customers",
  "id": "885f682f-a990-4bad-a642-36736a74b0c7",
  "isEnabled": true,
  "type": "Admin",
  "value": "DelegatedAdminRelationship.ReadWrite.All"
},
{
  "adminConsentDisplayName": "Sign in and read user profile",
  "id": "e1fe6dd8-ba31-4d61-89e7-88639da4683d",
  "isEnabled": true,
  "type": "User",
  "value": "User.Read"
},
{
  "adminConsentDisplayName": "Read directory data",
  "id": "06da0dbc-49e2-44d2-8312-53f166ab848a",
  "isEnabled": true,
  "type": "Admin",
  "value": "Directory.Read.All"
},
{
  "adminConsentDisplayName": "Read and write directory data",
  "id": "c5366453-9fb0-48a5-a156-24f0c49a4b84",
  "isEnabled": true,
  "type": "Admin",
  "value": "Directory.ReadWrite.All"
},
```

Appendix B Manually requesting App Consent

If you have the Global Administrators role in that customer’s tenant, you can consent to the app yourself. In cases where the customer does not want to approve Global Administrator’s role, you can ask them to enter the URI into their browser and manually consent themselves. Once completed your app should be able to run with GDAP enabled customer.

Getting your app consented in the customers tenant by constructing a URI as follows:

- https://login.microsoftonline.com/{customertenant}.onmicrosoft.com/adminconsent?client_id={appid}
- Replace the {customertenant} with the customer's
- Replace {appid} with the app you are looking to consent.
- Enter the newly constructed URI into a web browser and follow the login and consent prompts.

Resources

The following resource links are a suggested reading order

DAP Application Consent (Being Deprecated)

Cloud Solution Provider Application Pre Consent Using Admin Agent

GDAP Application Consent

Partner Center Authentication Partner-Consent
CPV/CSP API for Application Customer Consent

Secure Application Model

Secure Application Model
Use the secure application model framework
Secure partner application for CSP and CPV
Partner Center Samples Secure App Model (SAM)

Microsoft Graph

Microsoft Graph auth overview

Microsoft Identity Platform

Build apps that sign in Azure AD users
Azure AD built-in roles
Microsoft identity platform and OAuth2.0 On-Behalf-Of flow
Microsoft identity platform scopes, permissions, & consent
Microsoft identity platform access tokens
Microsoft identity platform consent framework
Glossary of terms in the Microsoft identity platform (consent)