Configuring the SDE for CIAM Integration

To integrate with CIAM, add the required transformation files to the customer repository so that the Aras DevOps pipeline will inject CIAM configuration into OAuth server configuration at deploy time. This needs to be done once per repository.

The transformation files introduce the following placeholders:

- `${CIAM_INTEGRATION_OAUTHSERVER_CLIENTREGISTRY_USERPROVISIONER}` placeholder in `OAuth.config`
- The four `"${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_*}"` placeholders in `OAuthServer.Plugins.json`

These values are supplied as placeholders in the transformations, and the pipeline replaces them with initialized values during deployment process based on the CIAM application configuration.

> **Important**
>
> These variables are calculated automatically by Aras DevOps pipelines and do not need to be specified in a variable group.

1. Pre-requisites

Before initiating the migration, confirm with the Aras Global Cloud Services Team that:

1. The SDE has been updated to a Aras DevOps 1.15.0+ version
2. The `External_Authentication_Integration_CIAM` variable group is configured for customer's SDE.
3. The new `CIAMIntegrationConfigValues` pipeline library variable group exists and each environment type flag (`SIT/UAT/STG/PROD`) is set to `true` or `false` according to what should be migrated to CIAM.

> **Important**
>
> You need to contact Aras Global Cloud Services Team to enable CIAM integration for your SDE. Global Cloud Services Team will configure SDE infrastructure.

Important

The Local Development Environment (LDE) will not be integrated with CIAM automatically. CIAM integration applies to SDE pipeline-based deployments only.

2. Configure CIAM application and user mapping

Configure CIAM application and user mapping according to documentation on https://docs.aras.com/iam-user-guide

Changes made on the CIAM side for User Mapping configuration, or any updates to the Innovator Application settings in the CIAM portal, require a deploy pipeline run with Update strategy to be applied to the SaaS Aras Innovator instance.

The Aras DevOps pipeline fetches the latest configuration from the CIAM API on each run. So, to sync configuration into the running instance, trigger Deploy pipeline with update strategy.

3. Commit the required transformation file changes to the SDE

3.1 Commit the OAuth.config transformation file changes to the SDE

Create a new transformation file for `OAuth.config` (or modify the existing one if already present) under `~WorkRepository/TransformationsOfConfigFiles/OAuthServer/OAuth.config` with the content below.

Add or replace the body of the `UserProvisioner` client registry with the placeholder bellow:

```
<?xml version="1.0" encoding="utf-8"?>
<oauth xmlns:xdt="http://schemas.microsoft.com/XML-Document-Transform">
 <server xdt:Transform="InsertIfMissing">
  <clientRegistries xdt:Transform="InsertIfMissing">
   <clientRegistry id="UserProvisioner" enabled="true" xdt:Transform="InsertIfMissing" xdt:Locator="Match
    ${CIAM_INTEGRATION_OAUTHSERVER_CLIENTREGISTRY_USERPROVISIONER}
   </clientRegistry>
  </clientRegistries>
 </server>
</oauth>
```

3.2 Commit the OAuthServer.Plugins.json transformation file changes to the SDE

Create a new transformation file for `OAuthServer.Plugins.json` (or modify the existing one if already present) under

`~WorkRepository/TransformationsOfConfigFiles/OAuthServer/OAuthServer.Plugins.jsc`

.

Remove any existing entries for the following plugins if present:

- `Aras.OAuth.Server.Plugins.Saml2Authentication` plugin
- `Aras.OAuth.Server.Plugins.GenericUserMapper` plugin
- `Aras.OAuth.Server.Plugins.OpenIdConnectAuthentication` plugin.

Then add the following content:

```
{
  "@jdt.merge": [
    {
      "@jdt.path": "$.['OAuthServer.Plugins']",
      "@jdt.value": [
        {
          "Name": "Aras.OAuth.Server.Plugins.OpenIdConnectAuthentication",
          "Enabled": "${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_OPENIDCONNECTAUTHENTICATION_ENABLED}",
          "Options": "${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_OPENIDCONNECTAUTHENTICATION_OPTIONS}"
        },
        {
          "Name": "Aras.OAuth.Server.Plugins.GenericUserMapper",
          "Enabled": "${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_GENERICUSERMAPPER_ENABLED}",
          "Options": "${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_GENERICUSERMAPPER_OPTIONS}"
        }
      ]
    }
  ]
}
```

4. Run the CI pipeline on the target branch

Commit the changes above as a single Pull Request and merge it into the target branch.
Run CI pipeline on merged branch, which will build on this changes and will embed the configuration files into the OAuth server, making them available for deployment.

> **Important**
>
> Without a successful CI run that includes these transformations, CIAM integration cannot be enabled. The deploy pipeline will fail at validation if the required placeholders are missing from the CI artifacts.

5. Enable CIAM for the target environment type

The four `${CIAM_INTEGRATION_OAUTHSERVER_PLUGINS_*}` placeholders are replaced by the pipeline at deploy time based on the `CIAMIntegrationConfigValues` variable group. The behavior depends on whether CIAM is enabled for the target environment:

- When CIAM is enabled:
  - the pipeline substitutes `Enabled=true` and injects the actual plugin options received from CIAM to OAuthServer.Plugins.json.
  - the pipeline injects the actual `<scerets>` , `<allowedScopes>` and `<allowedGrantTypes>` block received from CIAM to OAuth.config.
- When CIAM is disabled:
  - the pipeline substitutes `Enabled=false` and `Options=[]` no authentication plugins are enabled in OAuthServer.Plugins.json.
  - the pipeline injects an empty value no `UserProvisioner` secrets/scopes are written to OAuth.config.

The following variables for managing CIAM integration by environment type in `CIAMIntegrationConfigValues` control this behavior:

- `CIAM_integration_environment_SIT_enabled` - when set to `true` , the deploy pipeline creates/updates CIAM applications for all instances deployed to SIT. When `false` , CIAM integration is disabled for all SIT instances.
- `CIAM_integration_environment_UAT_enabled` - when set to `true` , the deploy pipeline creates/updates CIAM applications for all instances deployed to UAT. When `false` , CIAM integration is disabled for all UAT instances.
- `CIAM_integration_environment_STG_enabled` - when set to `true` , the deploy pipeline creates/updates CIAM applications for all instances deployed to STG. When `false` , CIAM integration is disabled for all STG instances.
- `CIAM_integration_environment_PROD_enabled` - when set to `true` , the deploy pipeline creates/updates CIAM applications for all instances deployed to PROD. When `false` , CIAM integration is disabled for all PROD instances.

The following variables for plugin configuration control the behavior of the CIAM login:

- `ciam-plugin-setting-display-name` - controls the display name of the CIAM-backed identity provider on the Aras Innovator login screen (e.g., "Aras CIAM"). You can set desired display name that will be displayed on login screen.

- `ciam-plugin-setting-enable-external-signout` - controls whether external sign-out is enabled for the CIAM-backed identity provider.

Set flag to `true` in the `CIAMIntegrationConfigValues` variable group for the desired environment.
For example, to enable CIAM in SIT set:

```
CIAM_integration_environment_SIT_enabled = true
```

Repeat for `UAT` , `STG` , `PROD` as needed this can be done incrementally.

Important

Once this flag is set to true,

- All subsequent runs of DeployInnovator_* targeting SIT will create or update CIAM applications.
- All DeleteInnovator_* runs targeting SIT will delete the associated CIAM application.

This affects all Innovator instances in that environment across the SDE.
You cannot enable CIAM for one instance in SIT and leave others without it within this environment — the flag is SDE-wide per environment type.

Important

The settings ciam-plugin-setting-display-name and ciam-plugin-setting-enable-external-signout (from the CIAMIntegrationConfigValues variable group) are only applied during a deploy pipeline run with Deploy or Update strategy.
If these values changed in the variable group, re-run DeployInnovator_* with Update strategy for each affected instance to propagate the change.

6. Deploy Aras Innovator with CIAM integration

Once the CI pipeline has completed successfully and CIAM is enabled for the target environment type, run the deploy pipeline for each Innovator instance.

The `DeployInnovator_*` pipeline manages CIAM applications based on the instance name. On each run, the pipeline will:

1. Look up the CIAM application by instance name in the configured CIAM tenant.
    - If found → continue with that application.

- If not found → create a new CIAM application.

2. Set the CIAM application as enabled after deploying the instance.
3. Initialize the OAuth configuration placeholders for the deployed instance.

> **Important**
>
> Without a successful CI run that includes the transformation files, CIAM integration cannot be enabled. The deploy pipeline will fail at validation if the required placeholders are missing from the CI artifacts.

CIAM application is created once per Aras Innovator instance and reused during all updates of this deployment. The deploy pipeline will look up the existing CIAM application by name and update it in place.

The `DeleteInnovator_*` pipeline run against a CIAM-enabled instance will automatically disable and delete the associated CIAM application. No additional manual cleanup in CIAM is needed.

6.1 Enable CIAM for existing Innovator instances

The `DeployInnovator_*` pipeline registers/updates CIAM applications based on the instance name. Migration is therefore done per instance, per environment type.

Run the `DeployInnovator_*` pipeline for the existing instance, using as Pipeline artifacts `CI pipeline run` which completes on branch with commited transformations from “4 Run the CI pipeline on the target branch” step and configure Advanced options pipeline Variables as follows:

```
innovator_release_name = <instance_name> # e.g. myinstance-sit
setupInstanceStrategy = Update
```

> **Important**
>
> Make sure that all required Advanced options for the Update strategy are specified.

> **Important**
>
> Without a successful CI run that includes the transformations, CIAM integration cannot be enabled.

> **Important**

> The deploy pipeline will fail at validation if the required placeholders are missing in configuration from the source Continuous Integration (CI) artifacts.

The pipeline will:

1. Look up the CIAM application by instance name in the configured CIAM tenant.
    - If found → continue with that application.
    - If not found → the pipeline will create new CIAM application.
2. Set CIAM application as enabled after deploying the instance.
3. Initialize the placeholders in OAuth configuration for deployed instance.

6.2 Enable CIAM for new Innovator instances

For brand-new instances run the `DeployInnovator_*` pipeline using as Pipeline artifacts `CI pipeline run` which completes on branch with commited transformations from "*4 Run the CI pipeline on the target branch*" step and configure Advanced options pipeline Variables as follows:

```
setupInstanceStrategy = Deploy
```

The pipeline will:

1. Look up the CIAM application by instance name in the configured CIAM tenant.
    - If found → continue with that application.
    - If not found → the pipeline will create new CIAM application.
2. Set CIAM application as enabled after deploying the instance.
3. Initialize the placeholders in OAuth configuration for deployed instance.

7. Verify the instance is integrated with CIAM

After the pipeline completes successfully:

- Configure user access in CIAM for the instance according to " Centralized Identity Access Management "

Now you are able to log in to the instance using CIAM via redirecting from Aras Innovator login page to CIAM authentication (display name matches ciam-plugin-setting-display-name)