

Below is a **copy-paste-ready** **playbook** for wiring GAS-OIDC into a *plain* Django-templates project.

All code assumes **Django 4.x** and the **mozilla-django-oidc** library (lightweight and easy to extend).

If you follow the order you can usually get the first successful login in < 1 hour.

1 Install dependencies

```
pip install django mozilla-django-oidc #  
plus your usual project deps
```

2 Settings (settings.py)

```
INSTALLED_APPS +=
```

```
["mozilla_django_oidc"] # --- OIDC core
```

```
-----
```

```
----- OIDC_RP_CLIENT_ID =
```

```
os.getenv("OIDC_CLIENT_ID")
```

```
OIDC_RP_CLIENT_SECRET =
```

```
os.getenv("OIDC_CLIENT_SECRET")
```

```
OIDC_OP_DISCOVERY_ENDPOINT = (
```

```
"https://sso-int.mercedes-benz.com/.well-known/openid-configuration" # ↑ swap to prod when you move out of integration ) #  
We *must* include `openid`; add  
whatever else you need OI DC _ RP _ SCOPES  
= "openid profile email" # GAS-OI DC  
specific extras  
OI DC _ AUTH _ REQUEST _ EXTRA _ PARAMS =  
{ "acr_values": "gas:standard", # or  
gas:strong in production  
"response_mode": "form_post", # more  
secure than query # "prompt": "login", #  
uncomment if you need to force re-auth } #  
≥128-bit state/nonce length (mozilla-  
django-oidc uses 32 random bytes already)  
OI DC _ STATE _ SIZE = 32 OI DC _ NONCE _ SIZE  
= 32 LOGIN _ URL =  
"oidc_authentication_init"  
LOGIN _ REDIRECT _ URL = "/"  
LOGOUT _ REDIRECT _ URL = "/"  
SESSION _ COOKIE _ SECURE = True
```

```
SESSION_COOKIE_SAMESITE = "Lax"
SESSION_COOKIE_HTTPONLY = True
CSRF_COOKIE_SECURE = True
```

3 URLs (urls.py)

```
from django.urls import path, include from
.views import MyOIDCCallbackView,
my_logout_view urlpatterns = [
    path("oidc/", include([ path("",
    MyOIDCCallbackView.as_view(),
    name="oidc_authentication_callback"),
    path("login/",
    include("mozilla_django_oidc.urls")), ])),
    path("logout/", my_logout_view,
    name="logout"), path("",
    include("myapp.urls")), ]
```

mozilla_django_oidc.urls already gives you `oidc_authentication_init` (the “Login” redirect view).

4 Custom callback & validation (views.py)

```
from datetime import datetime, timedelta,
timezone from django.contrib.auth import
login from django.core.exceptions import
SuspiciousOperation from
mozilla_django_oidc.views import
OIDCAuthenticationCallbackView from
mozilla_django_oidc.auth import
OIDCAuthenticationBackend class
MyOIDCCallbackView(OIDCAuthenticationC
allbackView): """Adds nonce/acr/auth_time
checks and stores tokens in the session."""
def login_success(self): # validate extra ID-
token claims
----- id_token =
self.request.session.get("oidc_id_token",
{}) acr_expected =
self.get_settings("OIDC_AUTH_REQUEST_E
XTRA_PARAMS", {}).get( "acr_values" ) if
id_token.get("acr") != acr_expected: raise
SuspiciousOperation("Bad ACR") #
`prompt=login` handling ( $\leq 30$  s) if
```

```
self.request.session.pop("oidc_prompt_login", False): delta =
datetime.now(timezone.utc) -
datetime.fromtimestamp(
id_token["auth_time"], timezone.utc ) if
delta > timedelta(seconds=30): raise
SuspiciousOperation("auth_time too old")
# create / update user, then log in user =
self.backend.filter_users_by_claims(id_token).first() login(self.request, user) # store
complete token set safely server-side
self.request.session["oidc_access_token"]
= self.token["access_token"]
self.request.session["oidc_refresh_token"]
= self.token["refresh_token"]
self.request.session["oidc_id_token_jwt"]
= self.token["id_token"] # fetch userinfo ---
sub must match userinfo =
self.backend.get_userinfo(self.token["access_token"], id_token["sub"]) if
userinfo["sub"] != id_token["sub"]: raise
```

```
SuspiciousOperation("Userinfo sub mismatch")
```

```
self.request.session["oidc_userinfo"] = userinfo  
return super().login_success()
```

Matching backend

(auth_backend.py – optional)

```
class
```

```
GASOIDCBackend(OIDCAuthenticationBackend):  
    def filter_users_by_claims(self,
```

```
        claims):  
        return
```

```
        self.UserModel.objects.filter(username=claims["sub"])
```

```
    def create_user(self, claims):
```

```
        return self.UserModel.objects.create_user(
```

```
            username=claims["sub"],
```

```
            first_name=claims.get("given_name", ""),
```

```
            last_name=claims.get("family_name", ""),
```

```
            email=claims.get("email", "") )
```

Add it to AUTHENTICATION_BACKENDS if you need anything beyond defaults.

5 Templates

```
{% if user.is_authenticated %} <p>Hello {{
user.first_name }}!</p> <a href="{% url
'logout' %}">Logout</a> {% else %} <a
href="{% url 'oidc_authentication_init'
%}">Login</a> {% endif %}
```

6 Logout (front-channel + Django)

```
from django.contrib.auth import logout
from django.shortcuts import redirect def
my_logout_view(request): logout(request)
# Optional: hit GAS-OIDC end-session
endpoint so their SSO session ends
sso_logout = "https://sso-int.mercedes-
benz.com/.../logout" return
redirect(sso_logout + "?"
post_logout_redirect_uri=" +
request.build_absolute_uri("/"))
```


7 Auto-refresh middleware (access-token expiry)

```
class OLDRefreshMiddleware: def
__init__(self, get_response):
self.get_response = get_response def
__call__(self, request): rt =
request.session.get("oidc_refresh_token")
exp =
request.session.get("oidc_access_token_ex
pires") # store this when you fetch if rt and
exp and exp - time.time() < 60: # <60 s left?
tokens = refresh(request, rt) # do the POST
/token request.session.update({
"oidc_access_token":
tokens["access_token"],
"oidc_refresh_token":
tokens["refresh_token"],
"oidc_access_token_expires": time.time() +
tokens["expires_in"], }) return
self.get_response(request)
(Wire it in MIDDLEWARE after
AuthenticationMiddleware).
```


8 Security checklist (from the spec)

ItemWhere we satisfied itHTTPS / TLS validation everywheredefault requests + Django settingsCookies: Secure, HttpOnly, SameSiteSettings.py aboveState / Nonce 128-bit, never reusedlibrary defaultsRefresh-token server-side onlystored in session, never exposedID-token extra checks (nonce, acr, auth_time)custom callbackUserInfo sub must match ID-token subcustom callbackRotate refresh-tokensmiddleware above

9 Smoke-test script

in another terminal python manage.py runserver 8000 # browser: http://localhost:8000 → click “Login” #  you should be redirected through GAS login and land back authenticated

If login loops or errors, set `DEBUG=True` and add:

`OIDC_VERIFY_SSL = False` # ONLY if your corporate proxy does TLS inspection

10 Next steps

- **Add unit tests** for the callback view (nonce/ acr mismatches).
- Point discovery URL to production when you get real credentials.
- Harden CSP headers if you embed the site in other Mercedes portals.
- Instrument token-refresh path with Sentry in case the “max 7 refresh tokens” limit bites.

That’s a complete, **spec-compliant** Django-templates integration for GAS-OIDC.

Feel free to drop back if you need help with logout propagation, token-introspection for an API, or CSP settings!