OneSpan Sign – Salesforce OAuth 2.0 Client Credentials Flow

Overview

OneSpan Sign supports a Salesforce OAuth 2.0 **Client Credentials Flow** as an alternative to the Refresh Token Flow for callback event notifications. This flow is better suited for server-to-server integrations where no user interaction is required, eliminating the need to retrieve and rotate a refresh token.

There are two key differences from the Refresh Token Flow setup:

1. **Salesforce App**: Enable the "Client Credentials Flow" option on your Connected App or External Client App (ECA).
2. **OneSpan Sign Callback Service**: The base64-encoded `key` field uses `client_secret` instead of `refresh_token`.

> **Note:** Salesforce no longer allows creating new Connected Apps in some org types. If you cannot create a new Connected App, use an **External Client App (ECA)** instead — the steps below cover both.

Step 1: Configure the Salesforce App

Option A: External Client App (ECA) — recommended for new setups

1. Setup → **External Client Apps** → **New**
2. Fill in the basic info (Label, Contact Email)
3. Under **OAuth Settings**:
   - Enable OAuth: ✓
   - **Enable Client Credentials Flow**: ✓
   - OAuth Scopes — add: `Manage user data via APIs (api)`
   - Do **not** add `refresh_token, offline_access` — this scope is only valid for user-delegated flows and will cause an authentication error with Client Credentials Flow
4. Save, then click **"Manage Consumer Details"** and note down the **Consumer Key (client_id)** and **Consumer Secret (client_secret)**
5. **Required — set a "Run As" user**: go to your ECA → **Policies** → **OAuth Policies** → **OAuth Flows and External Client App Enhancements**, check **Enable Client Credentials Flow**, then fill in **Run As** → select an active integration or admin user

> **Why "Run As" is required:** Salesforce Client Credentials Flow has no interactive user, so Salesforce needs to know which user's context and permissions to apply to the access token. Without this, Salesforce returns `invalid_grant`.
>
> The "Run As" user must have:
>
> - **API Enabled** permission (check via Setup → Profiles → System Permissions, or run `SELECT Name, Profile.PermissionsApiEnabled FROM User WHERE Name = 'Your User'` in Developer Console)

> - Active status
> - Sufficient object/field permissions for what the callback needs to do

Option B: Connected App — for existing setups

Follow the same steps as documented in the Refresh Token Flow guide to register a Connected App, with the following changes:

- Under **API (Enable OAuth Settings)**, check **"Enable Client Credentials Flow"**
- OAuth Scopes: add `Manage user data via APIs (api)` — do **not** add `refresh_token, offline_access`

After saving, note down the **Consumer Key (client_id)** and **Consumer Secret (client_secret)** from "Manage Consumer Details".

- **Required — set a "Run As" user**: go to your Connected App → **Manage** → **Client Credentials Flow**, fill in **Run As** → select an active integration or admin user

---

Step 2: Set Up the Apex REST Callback Endpoint

The OneSpan Sign callback service will POST signing events to an Apex REST endpoint in your Salesforce org. If the OneSpan Sign managed package is not installed, create this class manually (This is just an example code, you need to define the logic when getting a callback event).

1. In your Salesforce org: gear icon (⚙) → **Developer Console** → **File** → **New** → **Apex Class**
2. Name it `OssCallbackResource` and paste the following:

```
@RestResource(urlMapping='/OSS/callback')
global with sharing class OssCallbackResource {

    @HttpPost
    global static void callback() {
        OssCallback ossCallback = (OssCallback) JSON.deserialize(
            RestContext.request.requestBody.toString(),
            OssCallback.class
        );
        //This is just an example code, you need to define the flow when getting a
callback event
        System.debug('Received callback: ' +
RestContext.request.requestBody.toString());

        if (ossCallback.name.equals('PACKAGE_COMPLETE')) {
            // handle completed package
        }

        RestContext.response.statusCode = 200;
    }

    public class OssCallback {
        public String name;
```

```
        public String packageId;
        public String sessionUser;
        public String documentId;
        public String message;

        public OssCallback() {}
    }
}
```

3. **File → Save**

After saving, confirm the endpoint URL by running in Developer Console → Query Editor:

```
SELECT Name, NamespacePrefix FROM ApexClass WHERE Name = 'OssCallbackResource'
```

- If `NamespacePrefix` is blank, the callback URL is: `https://yourdomain.my.salesforce.com/services/apexrest/OSS/callback`
- If `NamespacePrefix` is e.g. `ONSS`, the URL is: `https://yourdomain.my.salesforce.com/services/apexrest/ONSS/OSS/callback`

Step 3: Update the OneSpan Sign Callback Service

Register the callback listener with OneSpan Sign using the following API call:

HTTP Request

```
POST /api/callback/connectors/salesforceOauth2
```

HTTP Headers

```
Content-Type: application/json
Accept: application/json
Authorization: Basic {api_key}
```

Example Payload

```
{
  "url": "https://yourdomain.my.salesforce.com/services/apexrest/OSS/callback",
  "events": ["PACKAGE_COMPLETE", "PACKAGE_CREATE"],
  "key": "<base64_encoded_key>"
}
```

The `key` Field

The `key` value is a **Base64-encoded JSON** string with the following structure:

```
{
  "host": "<your_salesforce_domain>",
  "client_id": "<your_consumer_key>",
  "client_secret": "<your_consumer_secret>"
}
```

> **Notes on `host`:**
>
> - Use your org's My Domain (e.g. `yourcompany.my.salesforce.com`), do **not** include `https://`

To encode the key, run the following. The `-w 0` flag is required on Linux to prevent line wrapping, which would produce an invalid key:

```
echo -n '{"host":"yourdomain.my.salesforce.com","client_id":"
<your_client_id>","client_secret":"<your_client_secret>"}' | base64 -w 0
```

> **Note:** On macOS, omit `-w 0` — the macOS `base64` command does not wrap by default.

Step 4: Verify the Callback Configuration

To confirm the callback service is correctly configured, use:

HTTP Request

```
GET /api/callback/connectors/salesforceOauth2
```

HTTP Headers

```
Content-Type: application/json
Accept: application/json
Authorization: Basic {api_key}
```

Step 5: Test the Integration

To trigger a test:

1. Create a transaction from the sender portal, or via API with `"data" > "origin": "OSS"` in the transaction JSON.

2. Complete the signing process.
3. Monitor the **Salesforce Developer Console** for inbound callback logs (`System.debug` output from `OssCallbackResource`).
4. Verify that signed documents and the evidence summary appear in the configured Salesforce document folder.

Troubleshooting

Error	Cause	Fix
`invalid_grant: request not supported on this domain`	Wrong `host` value in the key	Use `test.salesforce.com` for sandbox, or your My Domain URL if login policy enforces it
`invalid_grant: no valid scopes defined`	`refresh_token, offline_access` scope added, or no valid scope	Remove `refresh_token, offline_access`; add `api` scope to the ECA/Connected App
`invalid_grant` (no "Run As" user)	"Run As" user not set on ECA Consumer Policies	Setup → External Client Apps → Consumer Policies → set Run As user
`NOT_FOUND: Could not find a match for URL`	Apex REST class not deployed in the org	Create `OssCallbackResource` Apex class (see Step 2)
Multi-line base64 key	`base64` command wraps at 76 chars by default on Linux	Use `base64 -w 0`

Comparison: Client Credentials vs. Refresh Token Flow

	Refresh Token Flow	**Client Credentials Flow**
User interaction required	Yes (initial authorization)	No
Token rotation needed	Yes	No
`key` **JSON field**	`refresh_token`	`client_secret`
Salesforce setting	Refresh Token Flow enabled	Client Credentials Flow enabled
"Run As" user required	No	Yes (ECA only)
Scope	`refresh_token, offline_access`	`api`
Best for	User-delegated access	Server-to-server integrations