

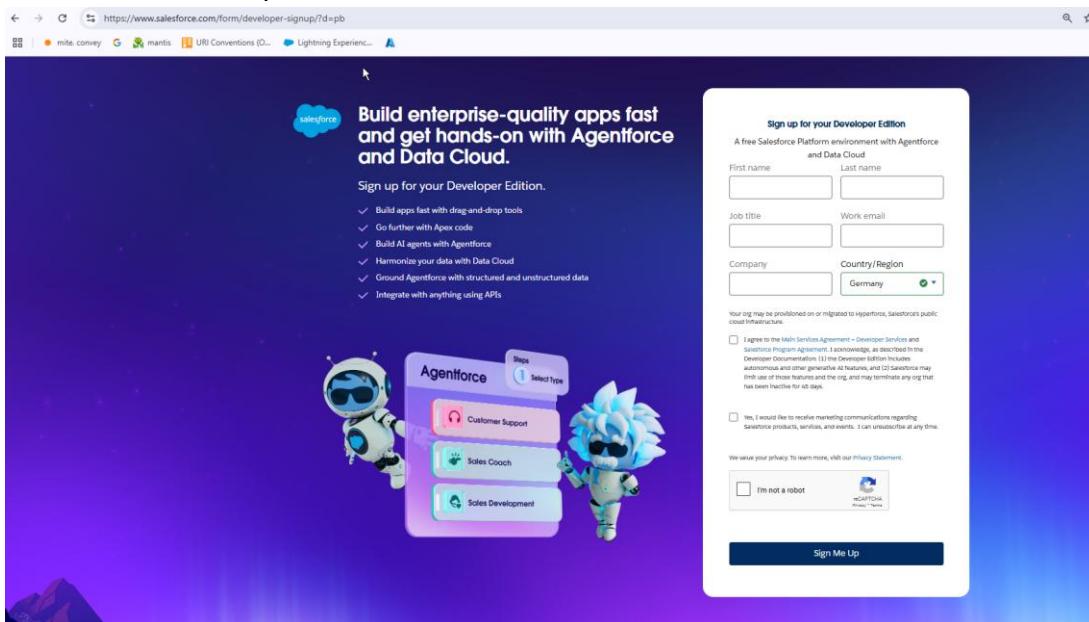
Technical Documentation for Salesforce API Integration

1. Creating a Salesforce Developer Account

1.1. Developer Account Registration

1. Access the registration form

- Go to <https://www.salesforce.com/form/developer-signup/>
- Fill out the form with your personal information
- Choose a secure password
- Accept the terms of use



2. Email verification

- Check your inbox
- Click on the verification link received
- Follow the instructions to finalize your account creation

3. First login

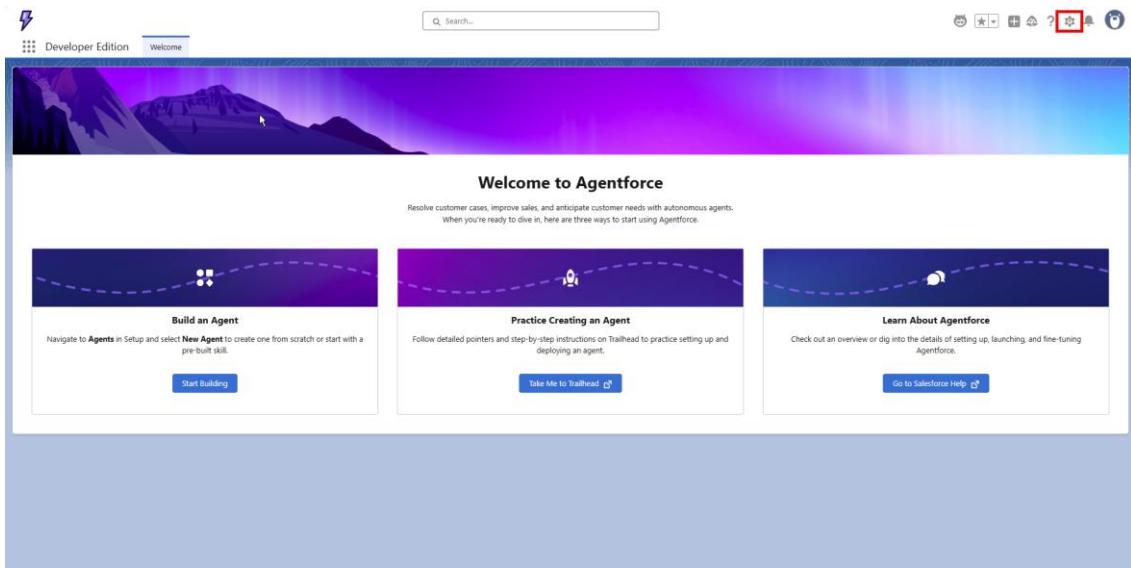
- You will be redirected to the Agentforce home page
- Familiarize yourself with the interface

2. Configuring a Connected App in Salesforce

You are now on the Agentforce home page, which is a Salesforce feature. To develop an application that will connect to your external system, we will follow these steps:

2.1 Accessing Salesforce Configuration (Setup)

- Click on the gear icon in the top right corner
- Select "Configuration" or "Setup"



2.2. Creating a Connected App

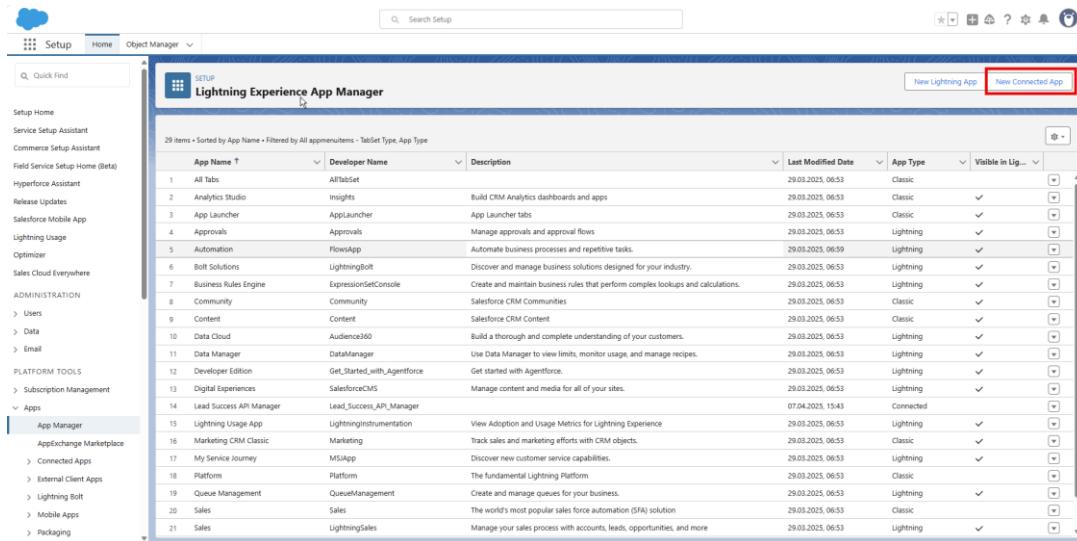
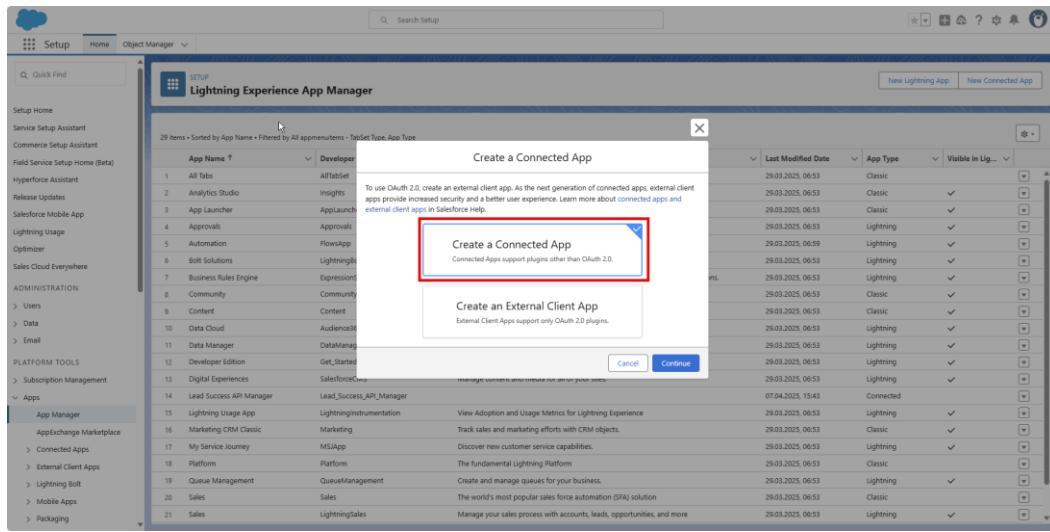
1. Access App Manager

- Click on the gear icon  in the top right corner
- In the configuration search bar, type "App Manager"
- Select "App Manager" from the results

2. Initiate the application creation

- Click on "New Connected App" in the top right
- In the dialog box that appears, select "Create a Connected App"

App Name	Developer Name	Description	Last Modified Date	App Type	Visible in Lig...
All Tabs	AllTabSet		28/02/2025 23:38	Classic	<input checked="" type="checkbox"/>
Analytics Studio	Insights	Build CRM Analytics dashboards and apps	28/02/2025 23:38	Classic	<input checked="" type="checkbox"/>
App Launcher	AppLauncher	App Launcher tabs	28/02/2025 23:38	Classic	<input checked="" type="checkbox"/>
Approvals	Approvals	Manage approvals and approval flows	28/02/2025 23:38	Lightning	<input checked="" type="checkbox"/>



2.3. Configuring the Connected App

1. Basic Information

- **Connected App Name:** Enter the name of your application, for example "*Demo Lead Success API Manager*"
- **API Name:** (automatically filled)
- **Contact Email:** your email address
- **Contact Phone:** your phone number (optional)

2. OAuth Settings

- Check "*Enable OAuth Settings*"
- **Callback URL:**

Enter the URL where our application will redirect after authentication:

<https://lsapisfbackend.convey.de/oauth-callback.html>.

Note: *This callback URL is the address to which Salesforce will redirect after successful authentication. This is where our application processes authentication data and completes the OAuth flow. This URL must be entered exactly as shown - incorrect URLs will result in authentication failures.*

Additional Callback URLs (Optional): If you're testing in different environments, you can add additional callback URLs (one per line):

For testing: <https://your-are-domain/oauth-callback.html>

For local development: <http://localhost:3000/oauth-callback.html>

Select OAuth scopes:

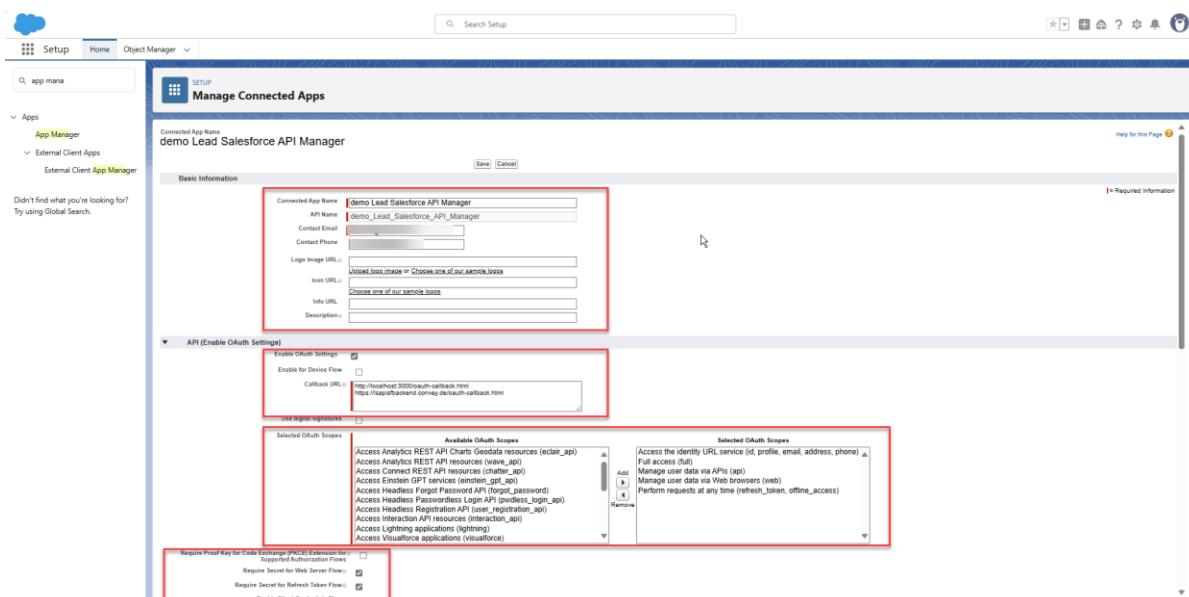
1. **Access the identity URL service (*id, profile, email, address, phone*)** Allows access to basic user information such as profile, email, and contact details.
 2. **Manage user data via APIs (*api*)** Authorizes access and manipulation of Salesforce data via REST, SOAP, and Bulk APIs
 3. **Full access (*full*)** Provides complete access to the Salesforce instance, equivalent to the connected user's rights
 4. **Perform requests at any time (*refresh_token, offline_access*)** Allows using a refresh token to obtain new access tokens without requiring the user to reconnect
- **IMPORTANT:** Check "Require Secret for Web Server Flow" and "Require Secret for Refresh Token Flow"
 - **DO NOT check** "Require Proof Key for Code Exchange (PKCE) Extension"
3. Why not select PKCE?
The "**Require Proof Key for Code Exchange (PKCE) Extension for Supported Authorization Flows**" checkbox is an additional security measure primarily used for mobile applications or public applications (client-side) that cannot securely store a client secret.

In your case, you should not select this option because:

- Your integration is likely a server-side application that can securely store the client secret
- **Enabling PKCE** requires additional authentication steps in your code
- For testing with Postman or other API tools, the absence of PKCE considerably simplifies the authentication process
- If you enable PKCE, you will need to modify your authentication code to include the generation and verification of PKCE codes, which complicates the integration

4. Save the application

- Click on "**Save**" to create the connected app



The screenshot shows the 'Lightning Experience App Manager' page. The left sidebar has 'App Manager' selected under 'Apps'. The main area lists 30 items, with item 12, 'demo Lead Salesforce API Manager', highlighted by a red border.

App Name	Developer Name	Description	Last Modified Date	App Type	Visible in ...
All Tabs	AllTabSet		28/02/2023 23:38	Classic	✓
Analytics Studio	Insights	Build CRM Analytics dashboards and apps	28/02/2023 23:38	Classic	✓
App Launcher	AppLauncher	App Launcher tabs	28/02/2023 23:38	Classic	✓
Approvals	Approvals	Manage approvals and approval flows	28/02/2023 23:38	Lightning	✓
Automation	FlowsApp	Automate business processes and repetitive tasks	28/02/2023 23:43	Lightning	✓
Bolt Solutions	LightningBolt	Discover and manage business solutions designed for your industry	28/02/2023 23:38	Lightning	✓
Community	Community	Salesforce CRM Communities	28/02/2023 23:38	Classic	✓
Content	Content	Salesforce CRM Content	28/02/2023 23:38	Classic	✓
Customer 360 Data Platform	Customer_360_Data_Platform	W-6859028	12/03/2023 18:30	Connected	✓
Data Cloud	Audience360	Build a thorough and complete understanding of your customers	28/02/2023 23:38	Lightning	✓
Data Manager	DataManager	Use Data Manager to view limits, monitor usage, and manage recipes	28/02/2023 23:38	Lightning	✓
demo Lead Salesforce API Manager	demo_Lead_Salesforce_API_Manager		15/05/2023 11:12	Connected	✓

2.4. Obtaining Consumer Keys

1. Access application details

- In the App Manager page, locate your application "**Demo Lead Success API Manager**" or your application
- Click on "**Manage**" to the right of the application

2. Get Consumer Keys

- In the "API (Enable OAuth Settings)" section, click on "**Manage Consumer Details**"
- You will need to confirm your identity with a code sent by email
- Once authenticated, copy:
 - Consumer Key (public key)*
 - Consumer Secret (secret key)*
- Keep this information secure** - you will need it to configure your application

The screenshot shows the 'Manage Connected Apps' page for the 'demo Lead Salesforce API Manager' application. The 'Manage Consumer Details' button is highlighted with a red arrow. The page displays basic app information and OAuth settings.

Connected App Name	demo Lead Salesforce API Manager
Version	1.0
API Name	demo_Lead_Salesforce_API_Manager
Created Date	15/05/2023 11:12
Contact Email	[redacted]
Contact Phone	[redacted]
Last Modified Date	15/05/2023 11:12
Description	[redacted]
Info URL	[redacted]

API (Enable OAuth Settings)

Selected OAuth Scopes: Manage Consumer Details (highlighted with a red arrow)

Manage consumer details: Access the identity (ID, Name, profile, email, address, phone) and manage user data via APIs (get, post, put, patch, delete, etc.)

Callback URL: http://localhost:3000/api/auth/3/callback

Enable for Device Flow: [checkbox]

Require Proof Key for Code Exchange (PKCE) Extension for: [checkbox]

Require Secret for Web Server Flow: [checkbox]

Require Secret for Refresh Token Flow: [checkbox]

Enable Client Credentials Flow: [checkbox]

Enable Authorization Code and Credentials Flow: [checkbox]

Enable Token Exchange Flow: [checkbox]

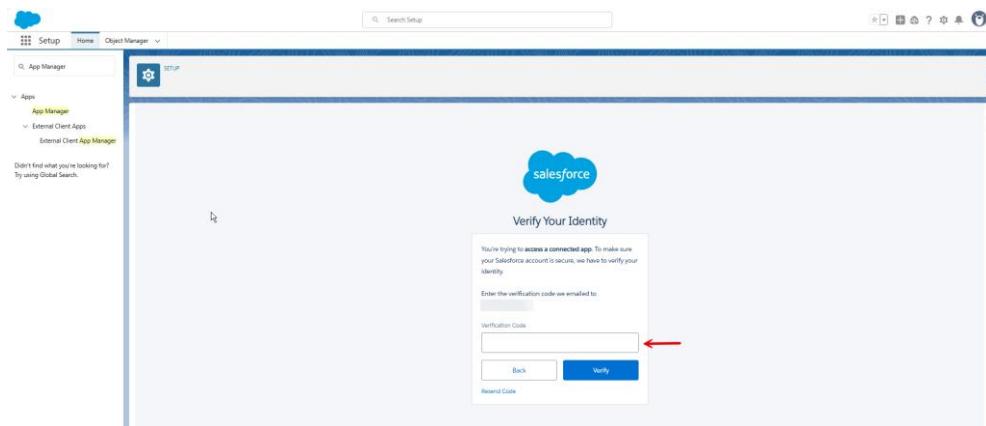
Enable Refresh Token Rotation: [checkbox]

Issue JSON Web Token (JWT)-based access tokens for named users: [checkbox]

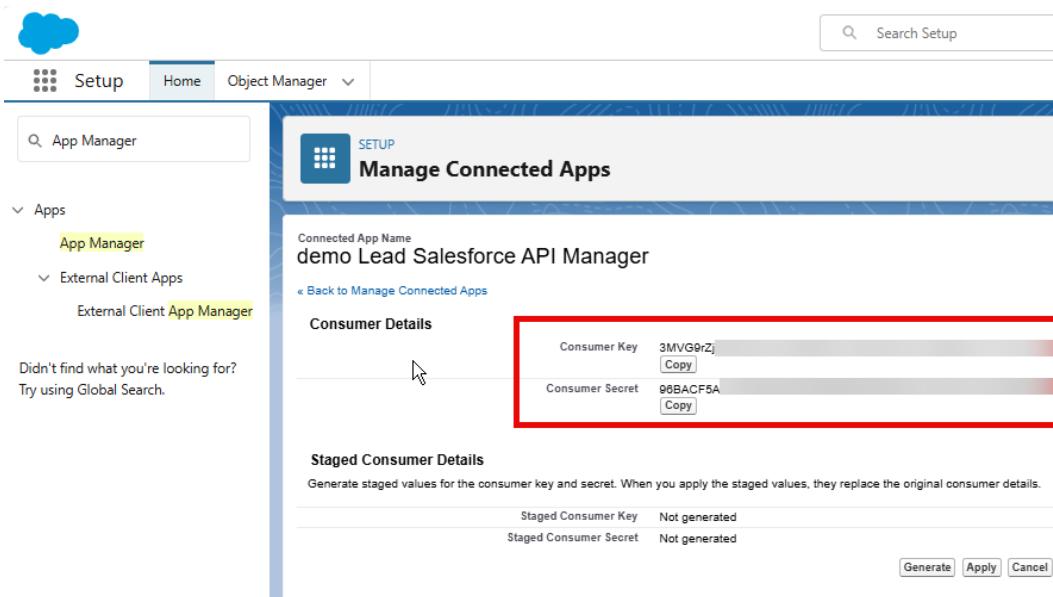
Salesforce Authentication Flow

When connecting to our application through Salesforce, you'll need to verify your identity. After entering your Salesforce credentials, you'll see this verification screen.

Enter the verification code sent to your email address in the field highlighted with the red arrow. This extra security step ensures only authorized users can connect to our application with your Salesforce credentials.



Once verified, you'll be redirected back to our application, and the integration will be complete.



- **Consumer Key:** This is your Client ID (shown as "3MtVG9rZj" in the example)
- **Consumer Secret:** This is your Client Secret (shown as "96BACF6A" in the example)

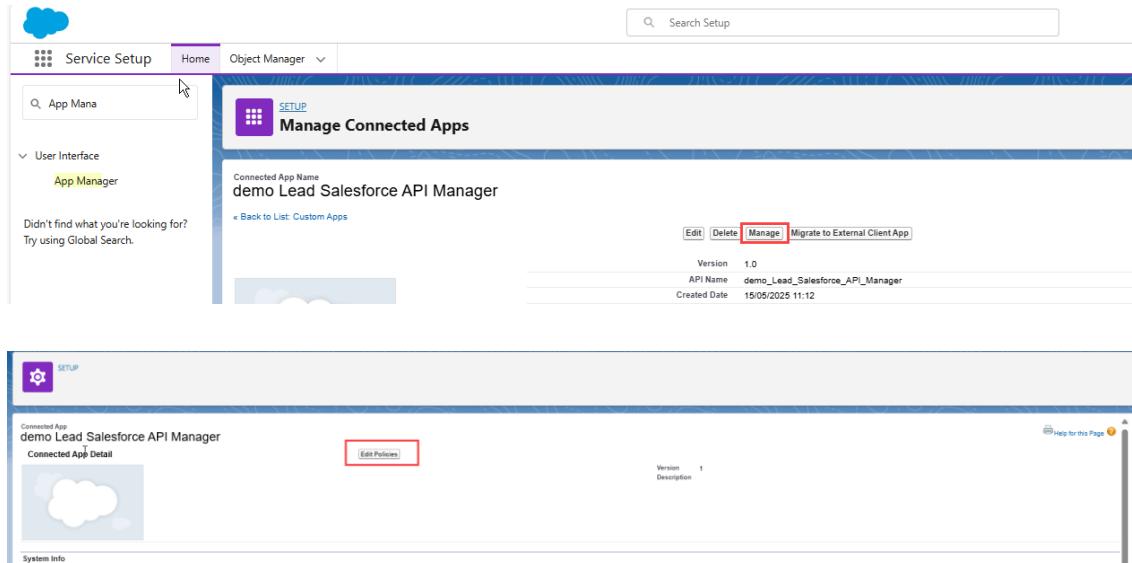
These credentials must be entered in our application's Salesforce configuration section. Use the "Copy" buttons to copy each value without typing errors. Keep these credentials secure as they provide access to your Salesforce **data**.

Note: The values shown are examples and your actual credentials will be different. **Never share these credentials publicly.**

2.5. Configuring OAuth Policies

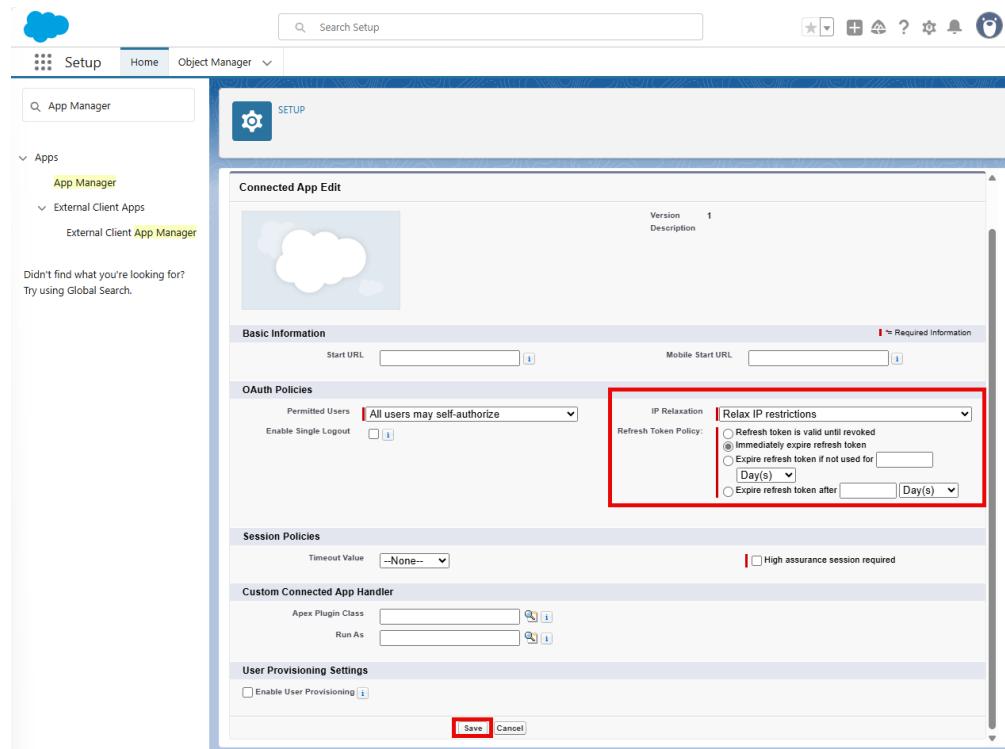
1. Edit policies

- On the connected app details page, click on "Manage" after that **Edit Policies**"



2. Configure security settings

- Permitted Users: "**All users may self-authorize**"
- IP Relaxation: "**Relax IP restrictions**" (allows connections from different IP addresses)
- Refresh Token Policy: "**Immediately expire refresh token**" (or according to your security needs)
- Click on "**Save**"



2.6. Personal Security Configuration

Step 1: Access personal information

- In the configuration menu, navigate to "**My Personal Information**"

- You can view and modify your personal information such as name, email, etc.

The screenshot shows the Salesforce Setup interface. In the top navigation bar, there are icons for Home, Object Manager, and a search bar labeled "Search Setup". Below the navigation, a sidebar on the left lists various setup categories under "My Personal Information". A red box highlights the "Personal Information" button in the sidebar. The main content area is titled "Personal Information" and contains sections for "Details", "Address", and "My Work Information". The "Details" section includes fields for First Name, Last Name, Alias, Email, Username, Nickname (which is set to "Convey GmbH"), Phone, Extension, Fax, and Mobile. The "Address" section shows Germany as the country, Leonrodstraße 68 as the street, München as the city, and 80636 as the zip/postal code. The "My Work Information" section includes Company Name (Convey GmbH), Title (Software Developer), Department, and Division. A red box highlights the "Reset My Security Token" link in the sidebar.

Step 2: Manage security tokens

- In the side menu, click on "**Reset My Security Token**"
- The security token is necessary for API authentication in certain scenarios

Step 3: Reset the security token

- Click on the "**Reset Security Token**" button
- A new token will be sent to your **email** address
- This token may be necessary for API authentication in certain scenarios, particularly when authenticating with username and password

The screenshot shows the "Reset My Security Token" page within the Salesforce Setup. The top navigation bar and sidebar are identical to the previous screenshot. The main content area is titled "Reset Security Token" and includes a note about what a security token is and how it's used. A red box highlights the "Reset Security Token" button at the bottom of the page.

3. Configuring CORS to Allow Cross-Origin Requests

CORS (Cross-Origin Resource Sharing) is essential to allow your frontend application to communicate with Salesforce.

1. Access CORS settings

- In the configuration search bar, type "CORS"
- Select "CORS" under the Security section

2. Add allowed domains

- Click on "New" to add a new origin
- Add **https://*.postman.com** for testing with Postman
- Add **https://*.postman.co** for testing with Postman
- Also add the URL of your application (for example: **https://your-domain.com**)

3. Why configure CORS?

- **CORS** is a security mechanism that prevents websites from making requests to other domains
- Without this configuration, requests from your frontend application to Salesforce would be blocked
- Adding Postman domains allows for easy API testing before complete integration

The screenshot shows the Salesforce Setup interface. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. A search bar is at the top right. Below the navigation, a sidebar on the left has 'Security' expanded, with 'CORS' highlighted and a red box drawn around it. The main content area is titled 'CORS' and contains the following information:

This page lists origins that are allowed for cross-origin resource sharing (CORS). To allow code (such as JavaScript) running in a Web browser to communicate with Salesforce from a specific origin, add the origin to the allowed list.

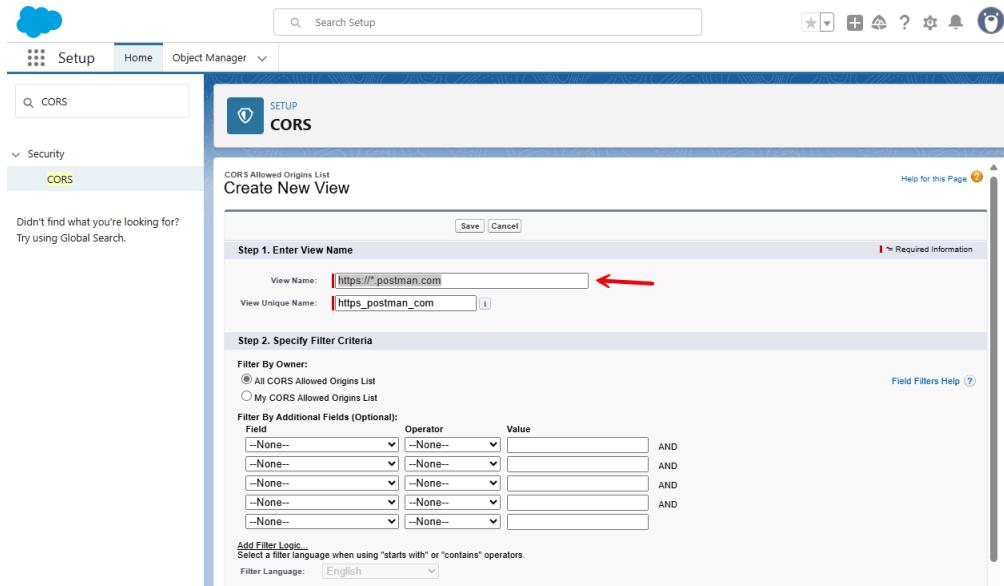
View: All [Create New View](#)

Action	Origin URL Pattern	Created By	Created Date	Last Modified By	Last Modified Date
Edit Del	https://postman.co	max	07.04.2025, 14:42	max	07.04.2025, 14:42
Edit Del	https://postman.com	max	07.04.2025, 14:41	max	07.04.2025, 14:41

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other [All](#)

Cross-Origin Resource Sharing (CORS) Policy Settings

Enable CORS for OAuth endpoints [Edit](#)



4. Architecture and Integration Operation

4.1. System Architecture

The integration between your application and Salesforce relies on three key components that work together to provide a seamless lead transfer experience:

4.1.1. Frontend (Web Application)

The frontend handles the user interface and client-side operations:

- **User Interface**
 - User interface for selecting and viewing leads
 - Lead selection and preview interface
 - Attachment preview and management
 - Transfer button and status indicators
 - Configuration modal for client-specific Salesforce settings
 - Communication with the backend for authentication and transfer operations
- **Data Management**
 - Temporary storage of lead data in **sessionStorage** for transfer operations
 - Secure storage of authentication tokens in **localStorage**
 - Secure storage of client-specific Salesforce configurations in localStorage
 - Client-side validation of lead data
- **Authentication**
 - Management of OAuth2 authentication flow initiation
 - Handling of authentication callbacks
 - Secure storage of tokens
 - Automatic token refresh handling

The screenshot shows a lead management interface with a header "Salesforce API Manager" and "API documentation available here - Test API in Postman". A blue box highlights the "Select Leads or LeadReport" section containing "View Leads" and "View Lead Reports" buttons. Below is a search bar with fields for Id, Subject, StartDate (tt.mm.jjjj), EndDate (tt.mm.jjjj), and a date range selector. A table lists leads with columns: Id, CreatedDate, LastModifiedDate, Subject, StartDate, EndDate, and Type. Two leads are shown: one for "API Test" and another for "API Test2".

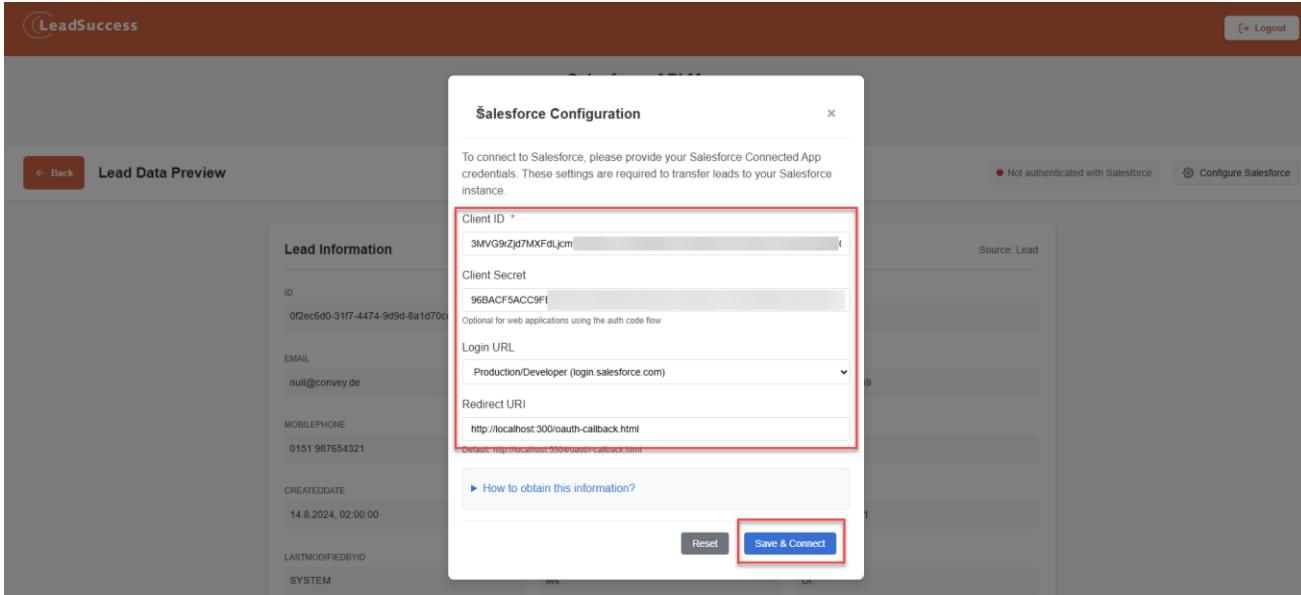
Lead Management Interface LeadSuccess lead management interface with data table view

The screenshot shows the LS (Lead Success) lead management interface with a header "Salesforce API Manager" and "API documentation available here - Test API in Postman". A blue box highlights the "Transfer to Salesforce" button. Below is a search bar with fields for Id, FirstName, LastName, Company, Email, and CreatedDate (tt.mm.jjjj). A table lists leads with columns: Id, CreatedDate, LastModifiedDate, CreatedById, LastModifiedById, and Salutation. Five leads are listed, each with a "Show Attachment" link.

LS (Lead Success) lead management interface with "Show Attachment" and "Transfer to Salesforce" buttons

The screenshot shows the Lead Data Preview screen with a header "Salesforce API Manager" and "API documentation available here - Test API in Postman". A yellow box highlights a notification: "Salesforce Configuration Required" with the message "Before transferring leads to Salesforce, you need to configure your Salesforce connection settings." A blue box highlights the "Configure Now" button. Below is a "Lead Information" table with columns: ID, FIRSTNAME, LASTNAME, and Source: Lead. One lead is previewed: ID 0f2ec6d0-31f7-4474-9d9d-8a1d70cc683, FIRSTNAME Erika, LASTNAME Musterperson.

1 - When you access the Lead Data Preview screen, you'll see a notification indicating that Salesforce configuration is required before you can transfer leads. Click the Configure Now button to begin the setup process.



2 - Enter Salesforce Connected App Credentials

In the configuration modal, you'll need to provide the following information:

Ursprung	Wert
Schlüssel	3MVG9rZjd7MXFdLjcm
SF_CLIENT_ID	96BAFCF5ACC9F1
SF_CLIENT_SECRET	https://login.salesforce.com
SF_LOGIN_URL	http://localhost:300/oauth-callback.html
SF_REDIRECT_URI	

After entering your Salesforce Connected App credentials, click the Save & Connect button. This will store your configuration and initiate the authentication process with Salesforce.

The configuration is saved locally in your browser's storage, as shown in the developer tools

Important Notes

1. The Client ID and Client Secret must match exactly what's provided in your Salesforce Connected App settings.
2. For security reasons, make sure you're using the correct Login URL for your Salesforce environment.
3. The Redirect URI must match precisely with the Callback URL configured in your Salesforce Connected App.
4. If you're testing in a development environment, the Redirect URI will differ from the production URL.

After configuring your Salesforce connection, you'll see a preview of any attachments associated with the lead. The example shows:

- A PDF document (24.85 KB)
- An SVG image file (487 bytes)

*These files will be automatically transferred along with the lead information when you complete the transfer process. Click the **Connect & Transfer to Salesforce** button highlighted in blue to proceed.*

The screenshot shows the LeadSuccess interface with the following details:

- Header: LeadSuccess, Logout
- Page Title: Salesforce API Manager
- Sub-Header: API documentation available [here](#) - [Test API in Postman](#)
- Section: Transfer Lead to Salesforce
- Buttons: Back, Lead Data Preview, Connected to Salesforce (highlighted with a red box), Disconnect from Salesforce, Configure Salesforce

Once you've successfully authenticated with Salesforce, you'll see a green status indicator showing "Connected to Salesforce" as highlighted in the red box. From this screen, you can:

- Confirm your active connection status
- Disconnect from Salesforce if needed
- Access configuration settings

The screenshot shows the Transfer Results page with the following details:

- Section: Transfer Results
- Message: Lead successfully transferred to Salesforce
- Details:
 - Salesforce ID: 00QgK000001f6nFUAQ
 - Status: Transferred
 - Message: Lead successfully transferred to Salesforce
 - Configuration: Custom
 - Attachments: 2/2 transferred

After the transfer is complete, you'll receive a confirmation showing:

- Success status with a green checkmark
- The Salesforce ID of the newly created lead
- Status confirmation ("Transferred")
- Success message
- Configuration type used (Custom)
- Attachment transfer status (2/2 transferred)

This confirmation page provides verification that both your lead data and all associated attachments have been successfully transferred to your Salesforce organization.

4.1.2. Client-Side Data Storage

The system intelligently uses browser storage mechanisms to maintain state and configuration:

SessionStorage - Used for temporary data during page navigation:

- Selected lead data for transfer
- Lead source information
- Attachment IDs

LocalStorage - Used for persistent data across sessions:

- Salesforce authentication tokens
- Client-specific Salesforce configuration
- User preferences

4.2. Authentication Flow

The authentication flow has been enhanced to support client-specific Salesforce configurations:

4.2.1. Authentication Process

1. Initialization

- User clicks “Transfer to Salesforce” button
- System checks for existing valid authentication token
- If no valid token exists, authentication process begins

2. Configuration Selection

- System checks for client-specific Salesforce configuration
- If exists, uses client configuration for authentication
- If not, uses default system configuration

3. Authorization Request

- System generates authorization URL with appropriate parameters
- A popup window opens with the Salesforce login page
- User authenticates with Salesforce credentials

4. Authorization Grant

- User approves access for the application
- Salesforce redirects to the callback URL with an authorization code

5. Token Exchange

- Server exchanges authorization code for access and refresh tokens
- Tokens are passed back to the frontend via `window.postMessage()`
- Frontend securely stores tokens in `localStorage`

4.3. Lead Transfer Process

The lead transfer process now incorporates client-specific Salesforce configuration:

4.3.1. Transfer Preparation

1. Lead Selection

- User selects a lead from the list in displayLsLead.html or displayLsLeadReport.html
- Selected lead data is stored in sessionStorage
- User is redirected to the transfer page

2. Lead Preview

- The transfer page (displayLeadTransfer.html) loads and displays the lead data
- Associated attachments are previewed if available
- User can verify information before transferring

3. Configuration Check

- System checks for client-specific Salesforce configuration
- If available, this configuration will be used for the transfer
- User can see which configuration will be used via the config button

4.3.3. Server-Side Processing

The server-side route /direct-lead-transfer now handles client-specific configuration:

```
// Direct lead transfer endpoint with attachments
apiRouter.post('/direct-lead-transfer', async (req, res) => {
  const { accessToken, instanceUrl, leadData, attachments } = req.body;

  console.log('Token present:', !!accessToken);
  console.log('Instance URL:', !!instanceUrl);
  console.log('Attachments count:', attachments ? attachments.length : '0');

  if (!accessToken || !instanceUrl) {
    console.error('Missing authentication data');
    return res.status(401).json({
      success: false,
      message: 'Missing authentication data. Please connect to Salesforce.'
    });
  }

  if (!leadData) {
    console.error('Missing lead data');
    return res.status(400).json({
      success: false,
      message: 'Lead data missing.'
    });
  }

  try {
    // Ensure token is decoded
    const decodedToken = decodeURIComponent(accessToken);

    // Basic check for duplicate lead by email
    if (leadData.Email) {
      try {
        console.log('Checking for duplicate by email:', leadData.Email);
      }
    }
  }
})
```

```

        const queryResponse = await
fetch(`/${instanceUrl}/services/data/v59.0/query?q=${encodeURIComponent(`SELECT
Id, Name FROM Lead WHERE Email = '${leadData.Email.replace(/\'/g, "\\'")}'})`), {
    method: 'GET',
    headers: {
        'Authorization': `Bearer ${decodedToken}`,
        'Content-Type': 'application/json'
    }
});

if (queryResponse.ok) {
    const queryResult = await queryResponse.json();

    if (queryResult.records && queryResult.records.length > 0) {
        console.log('Duplicate lead found by email:', queryResult.records[0].Id);
        return res.status(409).json({
            success: false,
            message: `A lead with this email already exists in Salesforce (ID: ${queryResult.records[0].Id})`,
            duplicateId: queryResult.records[0].Id
        });
    }
} catch (dupError) {
    console.error('Error checking for duplicate:', dupError);
    // Continue with transfer if duplicate check fails
}
}

// Prepare lead data for Salesforce - with improved validation
console.log('Preparing lead data...');
const sfLeadData = {
    FirstName: leadData.FirstName || '',
    LastName: leadData.LastName || 'Unknown',
    Company: leadData.Company || 'Unknown',
    Email: leadData.Email || '',
    Phone: leadData.Phone || '',
    MobilePhone: leadData.MobilePhone || '',
    Title: leadData.Title || '',
    Industry: leadData.Industry || '',
    Description: leadData.Description || '',
    LeadSource: 'LeadSuccess API'
};

// Function to clean field values before sending to Salesforce
const cleanField = (value) => {
    if (!value || value === 'N/A' || value === 'undefined' || value === 'null')
    {
        return ''; // Empty string is safer than null for Salesforce
    }
    return value;
};

// Add address fields only if they have valid values
if (leadData.Street) sfLeadData.Street = cleanField(leadData.Street);
if (leadData.City) sfLeadData.City = cleanField(leadData.City);
if (leadData.PostalCode) sfLeadData.PostalCode =
cleanField(leadData.PostalCode);
if (leadData.Country) sfLeadData.Country = cleanField(leadData.Country);
if (leadData.State) sfLeadData.State = cleanField(leadData.State);

// Create the lead in Salesforce
console.log('Creating lead in Salesforce...');

```

```

const leadResponse = await
fetch(`/${instanceUrl}/services/data/v59.0/sobjects/Lead`, {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${decodedToken}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(sfLeadData)
});

if (!leadResponse.ok) {
  const leadError = await leadResponse.json();
  console.error('Lead creation failed:', leadError);
  return res.status(leadResponse.status).json({
    success: false,
    message: `Failed to create lead: ${leadError[0] ? .message || JSON.stringify(leadError)} `,
    errors: leadError
  });
}

const leadResult = await leadResponse.json();
const leadId = leadResult.id;
console.log('Lead created successfully, ID:', leadId);

// Process attachments if available
let attachmentsTransferred = 0;
let attachmentErrors = [];

if (attachments && attachments.length > 0) {
  console.log(`Processing ${attachments.length} attachments for lead ${leadId}`);

  for (const attachment of attachments) {
    try {
      // Create ContentVersion record in Salesforce
      const contentVersionData = {
        Title: attachment.Name,
        PathOnClient: attachment.Name,
        VersionData: attachment.Body,
        ContentLocation: 'S', // S for Salesforce, E for External
        FirstPublishLocationId: leadId // Link to the Lead record
      };

      const attachmentResponse = await
fetch(`/${instanceUrl}/services/data/v59.0/sobjects/ContentVersion`, {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${decodedToken}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(contentVersionData)
});

      const attachmentResult = await attachmentResponse.json();

      if (attachmentResponse.ok) {
        attachmentsTransferred++;
        console.log(`Attachment '${attachment.Name}' created, ID: ${attachmentResult.id}`);
      } else {
        console.error(`Attachment creation failed: ${JSON.stringify(attachmentResult)}`);
        attachmentErrors.push(`Failed to upload ${attachment.Name}: ${attachmentResult[0] ? .message || 'Unknown error'}`);
      }
    } catch (attachErr) {

```

```

        console.error(`Error creating attachment '${attachment.Name}':`,
attachErr);
        attachmentErrors.push(`Error with ${attachment.Name}:
${attachErr.message}`);
    }
}

return res.json({
    success: true,
    leadId: leadId,
    status: 'Transferred',
    message: 'Lead successfully transferred to Salesforce',
    attachmentsTransferred,
    attachmentErrors: attachmentErrors.length > 0 ? attachmentErrors :
undefined
});
} catch (error) {
    console.error('Error during lead transfer:', error);
    return res.status(500).json({
        success: false,
        message: `Error: ${error.message}`
    });
}
);
}
);

```

4.3.4. Result Handling

After transfer completion:

1. Success Indication

- Green checkmark with success message
- Salesforce lead ID displayed
- Attachment transfer status (e.g., "1/1 transferred")
- Configuration used (Custom or Default)

2. Error Handling

- Red error icon with specific error message
- Option to retry or reconnect if authentication issue
- Detailed attachment errors if applicable

Successful Transfer Result Lead transfer success with Salesforce ID and configuration information

4.4. Security Considerations

- **No Server Storage:** Client credentials are never stored on the server
- **Isolated Configuration:** Each client's configuration is isolated in their own browser
- **Transparent Usage:** Clear indication of which configuration is being used
- **Controlled Scope:** OAuth scopes limited to only what's necessary
- **Secure Token Management:** Tokens stored in browser localStorage, accessible only to your domain

4.2. Authentication Flow

1. Authentication initiation

- The user clicks on the "Transfer to Salesforce" button

- The `handleTransferButtonClick()` function in `displayLeadTransferController.js` checks for existing authentication data
- If no authentication data is found, a request is sent to `/api/salesforce/auth`

```
/** 
 * Get Salesforce authentication URL
 * @returns {Promise<string>} Authentication URL
 */
async getAuthUrl() {
  try {
    console.log('Requesting Salesforce auth URL...');
    const response = await fetch(`.${this.apiBaseUrl}/salesforce/auth`, {
      method: 'GET',
      headers: {
        'Accept': 'application/json',
        'Cache-Control': 'no-cache, no-store, must-revalidate'
      }
    });

    if (!response.ok) {
      throw new Error(`HTTP error: ${response.status}`);
    }

    const data = await response.json();
    console.log('Auth URL received');
    return data.authUrl;
  } catch (error) {
    console.error('Error getting auth URL:', error);
    throw error;
  }
}
```

2. Redirection to Salesforce

- The server generates a Salesforce authentication URL with OAuth2 parameters
- A popup window opens with this URL
- The user logs in to Salesforce and authorizes the application

3. Callback and token storage

- After authentication, Salesforce redirects to the `/api/oauth2/callback` endpoint
- The server exchanges the authorization code for access and refresh tokens
- The tokens are returned to the frontend via `window.postMessage()`
- The frontend stores the tokens in `localStorage` via `storeAuthData()`

4. Token usage

- Subsequent requests to Salesforce include the access token
- If the token expires, the authentication process is restarted

5. Transferring Leads to Salesforce

5.1. Selecting a Lead

1. Displaying available leads

- The `displayLsLead.html` or `displayLsLeadReport.html` page displays available leads
- The `displayLsLeadController.js` controller retrieves leads from your source API

2. Selecting a lead

- Click on a lead in the list to select it
- The `handleRowSelection()` function activates the "Transfer to Salesforce" button

- The selected lead data is stored in the **selectedRowItem** variable

3. Initiating the transfer

- Click on the "**Transfer to Salesforce**" button
- The lead data is stored in sessionStorage
- The user is redirected to displayLeadTransfer.html

5.2. Preparing Attachments

1. Retrieving attachments

- The displayAttachmentsPreview() function in displayLeadTransferController.js displays attachments associated with the lead
- Attachments are retrieved via ApiService from IDs stored in AttachmentIdList

2. Processing attachments

- During transfer, the fetchAttachments() function retrieves the binary content of attachments
- Files are converted to base64 for transfer to Salesforce

5.3. Sending the Lead to Salesforce

1. Preparing lead data

- The continueTransferWithAuth() function cleans and validates lead data
- Required fields (LastName, Company) are checked and filled if necessary

2. Sending the request

- A **POST** request is sent to **/api/direct-lead-transfer**
- The server then uses Salesforce APIs to create the lead

3. Transferring attachments

- After creating the lead, the server creates ContentVersion entries for each attachment
- Files are associated with the lead via **ContentDocumentLink**

4. Managing results

- The transfer result (success or failure) is displayed to the user
- In case of success, the Salesforce ID of the created lead is displayed

5.4. Displaying Results in the LeadSuccess Application

After clicking the "Transfer to Salesforce" button, the system will process your request and display a summary of the result in the "Transfer Results" section:

- A green banner with a checkmark appears in case of success
- Detailed information is displayed as a list:
 - Salesforce ID: Unique identifier of the lead created in Salesforce (e.g., 00Qgk000000d7zJUAQ)
 - Status: Transfer status ("Transferred")
 - Message: Confirmation of successful transfer
 - Attachments: Number of attachments transferred (e.g., "1/1 transferred")

The screenshot shows a 'Transfer to Salesforce' interface. At the top, it says 'Attachments to be transferred' with a file named 'Drawing1501_377.svg' (image/svg+xml, 488 Bytes). A note below states '1 file(s) will be transferred with this lead'. A blue button at the bottom right says 'Transfer to Salesforce'. Below this, a 'Transfer Results' section shows a green success message: 'Lead successfully transferred to Salesforce'. It also displays the 'Salesforce ID' (00QgK00000d7zJUAQ), status ('Transferred'), message ('Lead successfully transferred to Salesforce'), and attachments ('1/1 transferred').

5.4.1 Accessing the Lead in Salesforce

To verify that the lead and its attachments have been properly created in Salesforce:

1. Log in to Salesforce

- Open your browser and go to your Salesforce instance (typically <https://login.salesforce.com>)
- Enter your login credentials

2. Navigate to leads

- Once logged in, click on the "Leads" tab in the top navigation bar
- If you don't see the tab, click on the App Launcher icon (nine dots) then search for "Leads"

3. Search for the transferred lead

- In the leads view, use the search bar at the top to search for the lead by name
- You can also create a recent view to see leads created today
- Leads are typically marked as "Open - Not Contacted" by default

4. View lead details

- Click on the lead name to open its detail page
- Verify that all information has been correctly transferred
- Note that some custom fields may appear differently in Salesforce

The screenshot shows the 'Leads' list view in Salesforce. The top navigation bar includes 'Sales Console' and 'Leads'. The main area shows a summary of leads: Total Leads (5), No Activity (5), Idle (0), No Upcoming (0), Overdue (0), Due Today (0), and Upcoming (0). Below this is a table with 5 items filtered by Created Date, Age, Total Leads. The columns include Name, Title, Company, Lead Status, Lead Source, Last Activity, and Actions. The leads listed are: Georg Klein (Geschäftsführer VermiB, convey GmbH, Open - Not Contacted, LeadSuccess API), Enka Musterperson (Arbeiter, Musterfirma GmbH, Open - Not Contacted, LeadSuccess API), Gérald Schwab (Systementwicklung, convey, Open - Not Contacted, LeadSuccess API), Unknown (Unknown, Unknown, Open - Not Contacted, LeadSuccess API), and another Unknown entry (Unknown, Unknown, Open - Not Contacted, LeadSuccess API).

5. Access all files

- To see all files in your organization, click on the "Files" tab in the navigation bar
- You can filter by owner or date to find your recently transferred files

The screenshot shows the Salesforce 'Files' page. At the top, there's a search bar and a navigation bar with links like Home, Accounts, Files, Groups, List Emails, Locations, Orders, Opportunities, Products, People, Reports, Solutions, Subscriptions, Profile, and more. The 'Files' tab is selected. On the left, there's a sidebar with sections for 'MY FILES' (Owned by Me, Shared with Me, Following), 'FILES IN MY LIBRARIES' (Private Library, Asset Library), and a 'Recent' section. The main area displays a list of files with columns for Actions, Name, Owner, and Last Modified. The files listed include 'Drawing1521_390.svg', 'DE_API Test_20240902_001521.pdf', 'Drawing1521_390.svg', 'DE_API Test_20240902_001521.pdf', 'NewFileFromPostman', 'Visitenkarte1533.jpg', 'Drawing1533_408.svg', 'DE_API Test2_20250320_Schwaab_001533.pdf', and 'DE API Test 20240714_Mustermann_001442.pdf'. Most files were uploaded by Watcho, Maxim on 08.04.2025, except for 'NewFileFromPostman' which was uploaded on 07.04.2025.

This verification allows you to confirm that the transfer was successful and that all data is correctly available in Salesforce for commercial follow-up.

6. Server-Side Technical Implementation

6.1. Node.js Server Configuration

The `server.js` file contains the server implementation needed for integration:

```
const SF_CLIENT_ID = process.env.SF_CLIENT_ID;
const SF_CLIENT_SECRET = process.env.SF_CLIENT_SECRET;
const SF_REDIRECT_URI = process.env.SF_REDIRECT_URI
const SF_LOGIN_URL = process.env.SF_LOGIN_URL || 'https://login.salesforce.com';
```

SF_CLIENT_ID	The Consumer Key of your Salesforce Connected App. This unique identifier tells Salesforce which app is requesting access.
SF_CLIENT_SECRET	The Consumer Secret of your Connected App. This serves as a password and should be kept secure. Together with the Client ID, it proves your app's identity to Salesforce.
SF_REDIRECT_URI	The callback URL where Salesforce will redirect users after authentication. This must exactly match one of the Callback URLs configured in your Connected App.
SF_LOGIN_URL	The base URL for Salesforce authentication. The default is 'https://login.salesforce.com' for production. For sandboxes, you would use 'https://test.salesforce.com'.

6.2. Authentication Routes

1. Route /api/salesforce/auth

- Generates the Salesforce authentication URL
- Defines necessary OAuth scopes
- Used by the frontend to start the authentication process

2. Route /api/oauth2/callback

- Receives the authorization code from Salesforce
- Exchanges the code for access tokens

- Returns the tokens to the frontend via window.postMessage()

6.3. Lead Transfer Route

1. Route /api/direct-lead-transfer

- Receives lead data and attachments
- Checks for potential duplicates
- Creates the lead in Salesforce
- Uploads attachments and associates them with the lead
- Returns detailed results to the frontend

7. Client-Side Technical Implementation

7.1. Salesforce Service

The **salesforceService.js** file handles communication with Salesforce:

```
// Retrieving stored authentication data
getStoredAuthData() {
  try {
    const authDataStr = localStorage.getItem('sf_auth_data');
    if (authDataStr) {
      return JSON.parse(authDataStr);
    }
    return null;
  } catch (error) {
    console.error('Error parsing stored auth data:', error);
    return null;
  }
}

// Storage of authentication data
storeAuthData(authData) {
  if (authData) {
    localStorage.setItem('sf_auth_data', JSON.stringify(authData));
    this.authData = authData;
  } else {
    localStorage.removeItem('sf_auth_data');
    this.authData = null;
  }
}
```

7.2. Lead Transfer Controller

The **displayLeadTransferController.js** file handles the interface and transfer logic:

```
// Handling the transfer button click
async function handleTransferButtonClick() {

  if (isTransferInProgress) return;

  const transferBtn = document.getElementById("transferToSalesforceBtn");
  const transferResults = document.getElementById("transferResults");
  const transferStatus = document.getElementById("transferStatus");

  if (!selectedLeadData) {
    showError("No lead data available.");
  }
```

```

        return;
    }

isTransferInProgress = true;
transferBtn.disabled = true;
transferBtn.innerHTML = `
<div class="spinner"></div>
Connecting and transferring...
`;

try {
    // Show progress indicator
    transferResults.style.display = "block";

    // Get existing auth data
    const authData = getAuthData();

    // If no auth data, start authentication flow
    if (!authData) {
        updateConnectionStatus("connecting", "Connecting to Salesforce...");

        transferStatus.innerHTML = `
<div class="transfer-pending">
<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
<circle cx="12" cy="12" r="10"/>
<polyline points="12 6 12 12 16 14"/>
</svg>
Salesforce authentication in progress...
</div>
`;

        // Start authentication
        const response = await fetch(`${appConfig.apiBaseUrl}/salesforce/auth`);
        const data = await response.json();
        const authUrl = data.authUrl;

        authWindow = window.open(
            authUrl,
            "salesforceAuth",
            "width=600,height=700"
        );

        if (!authWindow) {
            throw new Error("Popup blocked! Please allow popups for this site.");
        }
    }

    // The rest of the auth flow will be handled by the message event listener
    return;
}

// Otherwise, proceed directly to transfer using existing auth data
await continueTransferWithAuth(authData);
} catch (error) {
    console.error("Process failure:", error);

    // Show error
    transferStatus.innerHTML = `
<div class="status-error">
<svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
<circle cx="12" cy="12" r="10"/>
<line x1="15" y1="9" x2="9" y2="15"/>

```

```

<line x1="9" y1="9" x2="15" y2="15"/>
</svg>
Transfer failed: ${error.message || "Unknown error"}
</div>
`;

// Reset button
isTransferInProgress = false;
transferBtn.disabled = false;
transferBtn.innerHTML =
<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
<path d="M12 5v14M19 12l-7 7-7-7"/>
</svg>
Transfer to Salesforce
`;
}
}

```

7.3. Lead List Controller

The displayLsLeadController.js file handles the display and selection of leads:

```

// Adding the transfer button
function addTransferButton() {
    // Creating the button
    const transferButton = document.createElement('button');
    transferButton.id = 'transferButton';
    transferButton.className = 'action-button';
    transferButton.disabled = true;
    transferButton.innerHTML =
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 24 24" fill="none"
        stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
            <path d="M12 5v14M19 12l-7 7-7-7"/>
        </svg>
        <span>Transfer to Salesforce</span>
    `;
}

// Adding click event
transferButton.addEventListener('click', () => {
    if (selectedRowItem) {
        sessionStorage.setItem('selectedLeadData', JSON.stringify(selectedRowItem));
        sessionStorage.setItem('selectedLeadSource', 'Lead');
        window.location.href = 'displayLeadTransfer.html';
    } else {
        alert('Please select a lead to transfer.');
    }
});

}

}

```

8. Maintenance and Troubleshooting

8.1. Authentication Issues

1. Expired token

- Salesforce access tokens typically expire after 2 hours

- If a request fails with code 401, the application automatically restarts the authentication process

2. Connection problems

- Verify the validity of the Consumer Key and Consumer Secret
- Ensure Callback URL parameters are correct
- Check that OAuth scopes are sufficient

8.2. Lead Transfer Issues

1. Lead creation failure

- Verify that all required fields are filled
- Check logs for specific error messages

2. Attachment issues

- Check file size limits (Salesforce limits to 25MB per file)
- Verify supported file formats

8.3. CORS Issues

1. Errors like "No 'Access-Control-Allow-Origin' header"

- Verify that all necessary domains are configured in Salesforce CORS settings
- Ensure the application uses HTTPS if the CORS configuration requires it

9. Conclusion

This documentation guides you through the complete process of integration with Salesforce, from creating a developer account to transferring leads with attachments. The architecture implemented uses OAuth2 authentication best practices and offers a smooth user experience.

For any additional questions or assistance, please contact the technical support team.

10. Available Resources

10.1. Official Salesforce Documentation

- [Salesforce Developer Guide](#)
- [Salesforce REST API Documentation](#)
- [OAuth 2.0 Authentication Guide](#)
- [Salesforce Trailhead \(free interactive training\)](#)

10.2. Development Tools

- [Postman](#) - For testing API requests
- [Salesforce CLI](#) - Command-line tool for interacting with Salesforce
- [Workbench](#) - Web tool for exploring and manipulating Salesforce data
- [VSCode Extension Pack for Salesforce](#) - Salesforce development in VSCode

10.3. Forums and Communities

- [Salesforce Stack Exchange](#)

- [Salesforce Developers Community](#)
- [Salesforce Ohana Slack - Slack community for Salesforce developers](#)

10.4. Known Limitations and Quotas

- [Salesforce API Limits](#)
- Maximum attachment size: 25 MB per file
- Supported file types: most common formats (PDF, DOCX, XLSX, JPG, PNG, etc.)
- API call quota: varies by Salesforce edition (starting at 1,000 requests/24h for Developer Edition)

10.5. Tutorials and Code Examples

- [Salesforce REST API Integration Examples with Node.js](#)
- [JSforce - JavaScript Library for Salesforce](#)
- [Salesforce Integration Tutorials on YouTube](#)

10.6. Technical Support

- [Salesforce Developer Support](#)
- [Salesforce Services Status](#)