LeadSuccess API for Microsoft Dynamics - Technical Documentation

Table of Contents

Overview

The **LeadSuccess API for Microsoft Dynamics** provides access to server-side data structures and procedures via a RESTful API to enable 3rd party developers building customized Export interface from the LeadSuccess System to Microsoft Dynamics CRM system with standard web techniques and is covering the data created in the original LeadSuccess App in objects compatible to Microsoft Dynamics objects.

Key Features

- **Secure Client-Side Authentication**: Uses Microsoft Authentication Library (MSAL) for secure OAuth 2.0 authentication
- **Lead Data Transfer**: Complete lead information transfer including attachments
- **Real-Time Preview**: Preview lead data before transfer
- **Multi-Format Support**: Handles various attachment types (PDF, images, documents)
- **RESTful Interface**: Compatible with Microsoft Dynamics objects
- **OData v2.0 Protocol**: Based on SAP SQL Anywhere 17 OData implementation
- **Responsive Design**: Works on desktop and mobile devices
- **Configuration Management**: Easy setup and management of Dynamics 365 connections

LeadSuccess App

Introduction

The LeadSuccess App allows you to collect data and information about the visitors in an easy, fast and reliable way. It allows you to get the visitor's data in various ways:

- Scan barcodes, QR-Codes
- Scan business cards
- Enter information manually
- Fill out a customizable questionnaire
- Take notes and add pictures and sketches to every contact

Exhibitor Portal

Everything can be managed through the "Exhibitor Portal" which allows you to:

- Manage your App users
- Create your own event-specific questionnaire
- Check and edit leads
- Export your collected leads to different Excel formats

LeadSuccess App Structure

The LeadSuccess App user interface offers the following functional areas:

- **Start Page**: Event name, user name, number of contacts, present state, new message flag
- **Capture Barcode**: Create a new contact based on the visitor's badge barcode
- **Capture Business card**: Create a new contact based on the visitor's business card photo and save the photo linked to the contact
- **Edit contact manually**: Create a new contact or edit existing address fields of a selected visitor contact
- **Edit questionnaire**: Offers a form to answer the questions of the questionnaire based on the questionnaire configuration (multiselect, single select, combo, rating, text notes, date picker, optional questions, mandatory questions). Each questionnaire is linked to a selected contact
- **Create notes**: Create and edit sketches in SVG format, capture photos, capture audio messages. All notes are linked to a selected contact
- **Edit user state**: Edit personal user data, toggle the present state, receive and send user messages and capture a user photo. The data is linked to the employee data of the app user and can be administrated in the Exhibitor Portal

Environments

- **Production**: https://polite-pond-04d9e9503.6.azurestaticapps.net/
- **Development**: http://localhost:5504/

Technology Stack

Frontend:

- HTML5, CSS3, JavaScript (ES6+)
- Microsoft Authentication Library (MSAL) v2.x
- OData Protocol for API communication
- Responsive Web Design

Backend:

- SAP SQL Anywhere 17 OData Server
- OData Protocol v2.0
- RESTful API architecture
- HTTPS Basic Authentication

Authentication:

- Azure Active Directory (Azure AD)
- OAuth 2.0 / OpenID Connect
- Client-side token management

Integration:

- Microsoft Dynamics 365 Web API v9.2
- OData v4.0 for data queries
- RESTful API architecture

Prerequisites

System Requirements

- Modern web browser (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- Internet connection
- JavaScript enabled

LeadSuccess Environment Requirements

- **Valid LeadSuccess Account**: *Admin account to configure events and app users*
- **Server Access**: *Server name and API name provided by convey*

- **User Credentials**: *App user credentials configured in the Exhibitor Portal*

Microsoft Environment Requirements

- **Azure Active Directory**: *Access to Azure AD tenant*
- **Microsoft Dynamics 365**: *Valid Dynamics 365 license and instance*
- **Azure AD App Registration**: *Permissions to create/manage app registrations*
- **Dynamics 365 Permissions**: *User must have appropriate permissions to create leads*

Required Permissions

Azure AD App Registration:

- *user.read (Microsoft Graph)*
- *user_impersonation (Dynamics CRM)*

Dynamics 365:

- *Create permissions on Lead entity*
- *Read permissions on system entities*

Installation & Setup

For Development Environment

1. **Clone the Repository**

   ```
   git clone https://github.com/conveyGmbH/LSAPIMSSamples_www
   ```

2. **Install Dependencies**

   ```
   # No build process required - static files only
   # Ensure you have a local web server (e.g., Live Server, http-server)
   ```

3. **Start Development Server**

   ```
   # Using Live Server (VS Code extension) or
   npx http-server -p 5504
   ```

4. **Access Application**

   - Open browser to *http://localhost:5504*

For Production Deployment

The application is deployed as a static web application on Azure Static Web Apps.

Deployment URL: https://polite-pond-04d9e9503.6.azurestaticapps.net/

Azure AD Configuration

Step 1: Create App Registration

1. Navigate to Azure Portal
2. Go to **Azure Active Directory** → **App registrations**
3. Click **New registration**
4. Configure the application:
   - **Name**: *LeadSuccess Dynamics Integration*
   - **Supported account types**: *Accounts in this organizational directory only*
   - **Redirect URI**:
     - Type: *Single-page application (SPA)*
     - URL: *https://polite-pond-04d9e9503.6.azurestaticapps.net/*
     - For development: *http://localhost:5504/*

Step 2: Configure Authentication

1. In your app registration, go to **Authentication**
2. Under **Single-page application**, add redirect URIs:
   `https://polite-pond-04d9e9503.6.azurestaticapps.net/`
   `http://localhost:5504/`
3. Under **Implicit grant and hybrid flows**:
   - ✅ **Access tokens**
   - ✅ **ID tokens**

Step 3: API Permissions

1. Go to **API permissions**
2. Click **Add a permission**
3. Add **Microsoft Graph** permissions:
   - *User.Read (Delegated)*
4. Add **Dynamics CRM** permissions:
   - Click **APIs my organization uses**
   - Search for **Dynamics CRM**
   - Select **Delegated permissions** → *user_impersonation*
5. Click **Grant admin consent** (if you have admin rights)

Step 4: Collect Configuration Values

Note these values for application configuration:

- **Directory (tenant) ID**
- **Application (client) ID**
- **Dynamics 365 URL** (e.g., `https://yourorg.crm.dynamics.com`)

Application Configuration

Initial Setup

1. **Access the Application**
    - Navigate to the application URL
    - Log in with your LeadSuccess credentials
2. **Configure Dynamics 365 Connection**
    - Go to any lead transfer page
    - Click **Configure Dynamics CRM**
    - Enter your Azure AD configuration:
        - **Client ID**: *Your Azure AD App Registration Client ID*
        - **Tenant ID**: *Your Azure AD Tenant ID*
        - **Dynamics CRM URL**: *Your Dynamics 365 instance URL*

Configuration Management

The application stores configuration in browser **localStorage**:

- *dynamics_client_id*
- *dynamics_tenant_id*
- *dynamics_resource_url*

Configuration is persistent per browser/device.

OData Access Protocol

Protocol Overview

To access this interface is used a subset of ODATA protocol. See ODATA org. ODATA server is based on SAP SQL Anywhere 17 OData implementation. Therefore currently only ODATA version 2.0 syntax can be used.

Authentication

To access this API, HTTPS basic authentication is used. It means user and password has to be transmitted via HTTPS with each request. Since HTTP(S) is a stateless protocol, each request is computed in individual transactions.

```
var options = {
    type: "GET",
    user: <user-name>,
    password: <password>,
    url: "https://<server-name>/<api-name>/<endpoint>?$format=json",
};
```

Data Format

All parameter data must be transferred in JSON format to the ODATA server. For results the JSON format can be selected with the "$format=json" request option or with the HTTP header "Accept: application/json", otherwise the server returns the results in XML format.

- All text data is UTF-8 encoded, except described otherwise for specific data fields
- All date-time data is in UTC time zone
- Pay attention to the fact, that the OData request syntax is **case sensitive**!

Transaction Limitations

A user can only keep one transaction active at a time. If several parallel transactions are to be supported, different users must be used for this!

Request Types

Select Data (GET Requests)

Select a list of rows

You can select lists of data from API objects with requests like:

```
GET https://<server-name>/<api-name>/<entity-name>?$format=json
```

You can use the `$filter=(<filter-list>)` and `$orderby=(<orderby-list>)` options to specify restrictions and list order to the request.

Example Response Handling:

```
function xhrSuccess(response) {
    var obj = JSON.parse(response.responseText);
    var results = obj && obj.d && obj.d.results;
    handleResults(results);
}
```

Pagination: The result size maximum to be fetched by one request is limited to 100 rows. To fetch the next row-set, you can use the parameter `$skiptoken(<primary-key-value>)` in a following GET request to the same relation.

```
// Method to get the next URL from the data
getNextUrl(data) {
  let url = "";
  if (data && data.d) {
    const next = data.d.__next;

    if (next && typeof next === "string") {
      const viewNamePos = next.lastIndexOf("/");

      if (viewNamePos >= 0) {
```

```
          url = `https://${this.serverName}/${this.apiName}${next.substring(
            viewNamePos
          )}`;
        }
      }
    }
    return url;
  }

  // Method to fetch the next set of rows using the __next URL
  async fetchNextRows(nextUrl) {
    const errorElement = document.getElementById("errorMessage");
    if (errorElement) errorElement.style.display = "none";

    try {
      const headers = new Headers({
        Accept: "application/json",
        Authorization: `Basic ${this.credentials}`,
      });

      const config = {
        method: "GET",
        headers,
        credentials: "same-origin",
      };

      console.log("Fetching next rows with URL:", nextUrl);
      const response = await fetch(nextUrl, config);

      if (!response.ok) {
        const errorData = await response.json().catch(() => ({}));
        this.handleHttpError(response.status, errorData);
        return null;
      }

      return await response.json();
    } catch (error) {
      this.handleNetworkError(error);
      return null;
    }
  }
```

Select single row

```
GET https://<server-name>/<api-name>/<entity-name>(<primary-key>)?$format=json
```

Primary keys are always attributes of an integer type.

Procedure Call (GET Requests)

Use requests of type: **GET** to call procedures. Parameters need to be placed URL-encoded within the URL.

```
var options = {
    type: "GET",
    user: <user-name>,
    password: <password>,
    url: "https://<server-name>/<api-name>/<procedure-name>?<parameters>&$for
mat=json",
};
```

Parameters format: `<param1>=<value1>&<param2>=<value2>...`

CORS Requirements

To support cross-site access-control requests, you should specify the withCredentials member of the XMLHttpRequest to true:

```
var options = {
    // other options depending from action, see above
    customRequestInitializer: function(req) {
        if (typeof req.withCredentials !== "undefined") {
            req.withCredentials = true;
        }
    }
};
```

Authorization Header

To support server-side authorization propagation by some browser clients, like Google Chrome, you should add an “Authorization” header:

```
var options = {
    // other options depending from action, see above
    headers: {
        "Authorization": "Basic " + btoa(<user> + ":" + <password>)
    }
};
```

Data Schema

Schema Retrieval

You can retrieve the full schema data with the following request:

```
GET https://<server-name>/<api-name>/$metadata
```

Data Categories

Based on the given function areas, the frontend needs to handle different data structures to offer the functionality of the LeadSuccess App. This data can be divided in:

- **Static data**: to be loaded only once after app installation
- **Event data**: to be loaded at least once after app installation and updated after administrative changes in the LeadSuccess Exhibitor Portal
- **Application runtime data**: to be loaded at least once after app installation and updated after interactive changes in the app or in the LeadSuccess Exhibitor Portal
- **Visitor data**: to be created interactively in the app and to be retrieved via search and list panes

All data is organized in an object structure similar to Microsoft Dynamics objects. Object contain Microsoft Dynamics system fields like Id, CreatedDate and LastModifiedDate and object-specific fields. Each Object has a unique Id, that can be referenced as OwnerId field in another object.

Access Rights

The LeadSuccess API for Microsoft Dynamics offers access rights on object-level to the provided views in order to their usage. For static application and runtime data only GET requests are offered. Accessing an object with an unsupported request option will return an exception.

The LeadSuccess API for Microsoft Dynamics offers access rights on row-level to the provided views. You can only access rows of data in the user context of the currently logged-in user.

API Reference

Static Master Data

WCE_Country

Select the entries of this view to receive a list of CountryCodes and Countries that can be referenced in LeadSuccess API for Microsoft Dynamics.

Name	Type	Description
LGNTINITLandVIEWID	Int32	Primary key value
CountryId	Single-Byte-String(2)	Two letter ISO code. Only single-byte characters are allowed in this field
DisplayName	String(255)	Display text for the country to select

You can use several reference columns to identify country selection for data export.

Endpoint:

```
GET https://<server-name>/<api-name>/WCE_Country?$format=json
```

Application Runtime Data

WCE_Event

Select the entries of this view to show information about the events referenced by collected data.

Name	Type	Description
VeranstaltungViewId	Int32	Primary key value
EventId	UUID-String(36)	Universal Unique Identifier. Use this value as reference in objects referencing the **Event** object
DisplayName	String(127)	Display text for the event shown in the app. The value can be edited in LeadSuccess portal
CreatedBy	Single-Byte-String(63)	Login name of the user who created the event object
CreatedOn	DateTime	Creation timestamp of the event object
ModifiedBy	Single-Byte-String(63)	Login name of the user who created the event object last recently
ModifiedOn	DateTime	Timestamp of last modification of the event object
StartDate	Date	Start date of event
EndDate	Date	End date of event
Type	String(255)	Name of event organizer, defaults to value “LeadSuccess”
EventSubtype	String(1000)	Title of the event, usually the name of a trade show
Description	String(255)	A description of the event. The value can be edited in the LeadSuccess portal

Endpoint:

```
GET https://<server-name>/<api-name>/WCE_Event?$format=json
```

WCE_User

Select the entries of this view to show information about the users creating or modifying the collected data.

Name	Type	Description
MitarbeiterViewId	Int32	Primary key value
UserId	Single-Byte-String(63)	Login name of the user. Only single-byte characters are allowed in this field
DisplayName	String(320)	Display name of user

Name	Type	Description
CreatedBy	Single-Byte-String(63)	Login name of the user who created the user object
CreatedOn	DateTime	Creation timestamp of the user object
ModifiedBy	Single-Byte-String(63)	Login name of the user who created the event object last recently
ModifiedOn	DateTime	Timestamp of last modification of the event object
FirstName	String(64)	First name of the user. Can be edited in LeadSuccess portal and app
LastName	String(255)	Last name of the user. Can be edited in LeadSuccess portal and app
Email	String(500)	Email address of the user. Can be edited in the LeadSuccess app
Phone	String(64)	Phone number of the user. Can be edited in the LeadSuccess app
MobilePhone	String(64)	Cellphone number of the user. Can be edited in the LeadSuccess app
Street	String(127)	Street address of the user. Can be edited in the LeadSuccess app
PostalCode	String(12)	Postal code / ZIP address of the user. Can be edited in the LeadSuccess app
City	String(127)	City address of the user. Can be edited in the LeadSuccess app
Country	String(255)	Country address of the user. Can be selected in the LeadSuccess app
CountryISOCode	String(2)	2-character ISO-code of the country address of the user
Currentstatus	String(32)	Role of the user in LeadSuccess system, e.g. booth staff member or administrator
EventID	UUID-String(36)	Universal Unique Identifier. References the **Event** where the user is currently related to and is able to collect data for in the LeadSuccess app

Endpoint:

```
GET https://<server-name>/<api-name>/WCE_User?$format=json
```

Visitor Data

WCE_Lead

Select the entries of this view to retrieve visitor contact data collected or edited on LeadSuccess App Portal, Kiosk or Service devices or via LeadSuccess API.

Name	Type	Description
KontaktViewId	Int32	Primary key value
LeadId	UUID-String(36)	Universal Unique Identifier. Use this value as reference in objects referencing the **Lead** object
CreatedBy	Single-Byte-String(63)	Reference to the **User** who created this **Lead**
CreatedOn	DateTime	Creation timestamp of the lead object
ModifiedBy	Single-Byte-String(63)	Reference to the **User** who modified this **Lead** most recently
ModifiedOn	DateTime	Timestamp of last modification of the lead object
Salutation	String(32)	Salutation
Suffix	String(80)	Name suffix
FirstName	String(255)	First name
MiddleName	String(64)	Middle name
LastName	String(128)	Last name
CompanyName	String(1024)	Company name
JobTitle	String(80)	Job title
Address1_Telephone1	String(64)	Phone number
MobilePhone	String(64)	Cellphone number
Address1_Fax	String(64)	Fax number
EmailAddress1	String(500)	Email
WebsiteUrl	String(500)	Website
Address1_Line1	String(128)	Street address
Address1_PostalCode	String(12)	Postcode / ZIP of address
Address1_City	String(128)	Name of city
Address1_Country	String(255)	Name of country
Address1_CountryISOCode	Single-Byte-String(2)	Two letter ISO code. Only single-byte characters are allowed in this field
Address1_StateOrProvince	String(128)	Name of federal state
Description	String(4000)	User edited comments or automatically recognized other information that isn’t related to the other fields

Name	Type	Description
AttachmentIdList	LONG String: List of UUID-String(36)	Comma-separated list of UUID-Strings referencing **Attachment** objects related to the **Lead** object
SalesArea	String(4000)	User edited or automatically created text to describe the sales area where the lead might be related to
RequestBarcode	String(1000)	Barcode identifier in case of the lead was collected via ticket barcode-scan
StatusMessage	String(255)	Error message from trade show organizers visitor database in case of failed lead retrieval by ticket barcode-scan
DeviceId	Int32	Identifier of the local app database on the device where the lead was collected
DeviceRecordId	Int32	Identifier of the lead on the local app database on the device where the lead was collected
ModifiedBySystemOn	DateTime	Timestamp of last modification of the lead object by automatic processing on the LeadSuccess server
EventID	UUID-String(36)	UUID reference to the **Event** where the Lead was collected
Gender	String(32)	Gender selection from Salutation: Male, Female or null
Topic	String(127)	Event name
Newsletter	String(100)	Answer from questionnaire to first question containing the word ‘newsletter’
Department	String(100)	Single-Selection-answer from questionnaire to first question containing the word ‘department’
DisplayName	String (320)	Computed display name
IsReviewed	String(100)	Review status
DepartmentText	String (100)	Department description text

Endpoints:

Get all leads

```
GET https://<server-name>/<api-name>/WCE_Lead?$format=json
```

Get leads by event

```
GET https://<server-name>/<api-name>/WCE_Lead?$format=json&$filter=EventId eq '<event-id>'
```

Get leads modified since specific date

```
GET https://<server-name>/<api-name>/WCE_Lead?$format=json&$filter=(ModifiedBysystemOn ge cast('2023-11-02T07:00:00Z','Edm.DateTime'))
```

 # Order results

```
GET https://<server-name>/<api-name>/WCE_Lead?$format=json&$orderby=KontaktViewId desc
```

WCE_AttachmentById

Call this procedure to select any Attachment related to a Lead object, like the PDF file of the questionnaire, the business card image, questionnaire-related or other attached photos, attached sketches or voice-notes.

Parameters:

Parameter	Type	Description
Id	UUID-String(36)	UUID of the attachment

Endpoint:

```
GET https://<server-name>/<api-name>/WCE_AttachmentById?Id='<attachment-id>'&$format=json
```

WCE_Attachment

Attachments can be saved as document files of different file types specified via MIME type. Binary data of the file content is Base64 encoded.

Name	Type	Description
AttachmentId	UUID-String(36)	Universal Unique Identifier. Use this value as reference in objects referencing the **Attachment** object
CreatedBy	Single-Byte-String(63)	Reference to the **User** who created this **Attachment**
CreatedOn	DateTime	Creation timestamp of the attachment object
ModifiedBy	Single-Byte-String(63)	Reference to the **User** who modified this **Attachment** most recently
ModifiedOn	DateTime	Timestamp of last modification of the attachment object
FileName	String(255)	File name of the attachment

Name	Type	Description
Subject	String(4000)	Optional description of the attachment
MimeType	String(255)	MIME-type of the file data
Body	LONG String	Base64 encoded data of the file
FileSize	Int32	File size
EventId	UUID-String(36)	Reference to the **Lead** object where this **Attachment** object is related to

LeadSuccess API Endpoints

Authentication

```
GET https://{serverName}/{apiName}/
Authorization: Basic {base64(username:password)}
```

Get Entities

```
GET https://{serverName}/{apiName}/?$format=json
```

Get Events

```
GET https://{serverName}/{apiName}/WCE_Event?$format=json
```

Get Leads

```
GET https://{serverName}/{apiName}/WCE_Lead?$format=json&$filter=EventId eq '
{eventId}'
```

Get Attachments

```
GET https://{serverName}/{apiName}/WCE_AttachmentById?Id='{attachmentId}'&$fo
rmat=json
```

Dynamics 365 Integration

Lead Creation

```javascript
const leadData = {
  subject: "Lead from LeadSuccess",
  firstname: "John",
  lastname: "Doe",
  companyname: "Acme Corp",
  emailaddress1: "john.doe@acme.com",
  telephone1: "+1-555-0123",
  // ... additional fields
};

// Handled automatically by DynamicsService
await dynamicsService.transferLead(leadData, attachments);
```

Field Mapping

LeadSuccess Field	Dynamics 365 Field	Type	Max Length
FirstName	firstname	String	50

LeadSuccess Field	Dynamics 365 Field	Type	Max Length
LastName	lastname	String	50
CompanyName	companyname	String	100
EmailAddress1	emailaddress1	String	100
Address1_Telephone1	telephone1	String	20
MobilePhone	mobilephone	String	20
JobTitle	jobtitle	String	100
Topic	subject	String	300
Description	description	String	2000
Address1_Line1	address1_line1	String	250
Address1_City	address1_city	String	80
Address1_PostalCode	address1_postalcode	String	20
Address1_Country	address1_country	String	80
WebsiteUrl	websiteurl	String	200

User Guide

Accessing Lead Data

1. **Login to LeadSuccess**
    - Enter your
        - Server name
        - API name
        - Username
        - Password
    - Click **Login**
2. **Navigate to Leads**
    - Select **WCE_Event** from the entity dropdown
    - Choose an event to view associated leads
    - Click **View Leads**

Transferring Leads to Dynamics 365

1. **Select a Lead**
    - Click on any lead row to select it
    - The **Transfer to CRM** button becomes enabled
2. **Configure Dynamics 365 (First Time)**
    - Click **Configure Dynamics CRM**
    - Enter your Azure AD configuration details
    - Click **Save & Test**
3. **Authenticate with Dynamics 365**
    - Click **Connect to Dynamics 365**
    - Complete OAuth authentication in popup window
    - Verify connection status shows “Connected”
4. **Transfer Lead**
    - Click **Transfer to Dynamics CRM**
    - Review lead data preview
    - Confirm transfer
    - Monitor transfer status

Managing Attachments

The application automatically handles lead attachments:

- **Preview**: View attachments before transfer
- **Supported Formats**: PDF, images (JPG, PNG, GIF), documents (DOC, DOCX)

- **Transfer**: Attachments are included with lead data

Error Codes

For all requests, a result code and possibly a textual error message is returned.

The error message serves to describe the error for a developer as exactly as possible. Ideally, it should be saved in case of error in a log file. It is not intended to be displayed to an end user (exhibitor).

The result code is roughly based on the HTTP status codes and is intended as the basis for an automatic response to specific errors. The three-digit numeric error code can be followed by a detail error code with a dot.

Error Code Classification

The rough classification of the error is made possible by the first digit of the result code:

- **"2"** for success
- **"4"** for client-side errors, e.g. wrong or missing parameters
- **"5"** for server-side errors

Common Error Codes

Code	Description	Solution
200	Success	Request completed successfully
400	Bad Request	Incorrect request, e.g. missing mandatory parameter
400.1	Invalid Parameter	Check parameter format and values
400.2	Missing Parameter	Ensure all required parameters are provided
401	Unauthorized	Check authentication credentials
403	Forbidden	User lacks required permissions
404	Not Found	No matching record was found
404.1	Entity Not Found	Specified entity does not exist
404.2	Record Not Found	Specified record does not exist
500	Internal Server Error	Server problem, contact LeadSuccess administrator
500.1	Database Error	Database connection or query error
500.2	Service Unavailable	LeadSuccess service temporarily unavailable

Authentication Errors

Error	Description	Solution
Authentication failed	Invalid credentials	1. Verify Azure AD app registration configuration2. Check redirect URIs match exactly3. Ensure proper

Error	Description	Solution
		API permissions granted4. Clear browser cache and try again
Popup blocked	Authentication popup blocked	1. Allow popups for the application domain2. Disable popup blockers temporarily3. Try different browser
Invalid Client ID	Client ID format error	1. Verify Client ID is correct GUID format2. Check Azure AD app registration exists3. Ensure using correct tenant

Configuration Issues

Error	Description	Solution
Invalid Dynamics CRM URL	URL format error	1. Verify URL format: https://yourorg.crm.dynamics.com2. Ensure URL is accessible3. Check Dynamics 365 instance is active
Connection timeout	Network connectivity issue	1. Check internet connection2. Verify firewall settings3. Test with different network

Transfer Issues

Error	Description	Solution
Transfer failed	Lead transfer error	1. Check Dynamics 365 permissions2. Verify lead entity is accessible3. Check field validation errors4. Review browser console for details
Field validation error	Data format issue	1. Check field length limits2. Verify required fields are present3. Check data type compatibility

Please note that in addition to the application error codes defined here, further error messages of the overlying protocol levels (ODATA, web server) can occur.

Security Considerations

Authentication Security

- **OAuth 2.0**: Industry-standard authentication protocol
- **HTTPS Basic Authentication**: For LeadSuccess API access
- **Token Storage**: Secure token storage in browser sessionStorage
- **Token Refresh**: Automatic token refresh when possible
- **Session Isolation**: Each browser session is independent

Data Protection

- **HTTPS Only**: All communication encrypted in transit

- **No Server Storage**: No credentials or tokens stored on server
- **Client-Side Processing**: All data processing happens in browser
- **Minimal Permissions**: App requests only necessary permissions
- **UTF-8 Encoding**: All text data properly encoded
- **UTC Timezone**: Consistent datetime handling

Access Control

- **Row-Level Security**: Users can only access data in their context
- **Object-Level Permissions**: Different access rights per entity
- **Single Transaction Limit**: One active transaction per user
- **Cross-Origin Resource Sharing (CORS)**: Properly configured for security

Best Practices

1. **Regular Token Refresh**: Tokens automatically refreshed
2. **Secure Configuration**: Store configuration securely
3. **Access Control**: Use Azure AD conditional access policies
4. **Audit Logging**: Enable Dynamics 365 audit logging
5. **Regular Reviews**: Periodically review app permissions
6. **URL Encoding**: Properly encode all URL parameters
7. **Error Handling**: Implement comprehensive error handling

---

Troubleshooting

Common Issues

Authentication Errors

Error: “Authentication failed”

```
Solution:
1. Verify Azure AD app registration configuration
2. Check redirect URIs match exactly
3. Ensure proper API permissions granted
4. Clear browser cache and try again
```

Error: “Popup blocked”

```
Solution:
1. Allow popups for the application domain
2. Disable popup blockers temporarily
3. Try different browser
```

Configuration Issues

Error: “Invalid Client ID”

```
Solution:
1. Verify Client ID is correct GUID format
2. Check Azure AD app registration exists
3. Ensure using correct tenant
```

Error: "Invalid Dynamics CRM URL"

```
Solution:
1. Verify URL format: https://yourorg.crm.dynamics.com
2. Ensure URL is accessible
3. Check Dynamics 365 instance is active
```

OData Protocol Issues

Error: "OData syntax error"

```
Solution:
1. Remember OData request syntax is case sensitive
2. Check URL encoding for spaces (%20) and quotes (%27)
3. Verify filter and orderby syntax
4. Use only OData v2.0 syntax
```

Error: "Transaction limit exceeded"

```
Solution:
1. Only one transaction per user allowed
2. Use different user accounts for parallel transactions
3. Wait for current transaction to complete
```

Transfer Issues

Error: "Transfer failed"

```
Solution:
1. Check Dynamics 365 permissions
2. Verify lead entity is accessible
3. Check field validation errors
4. Review browser console for details
```

Error: "Attachment too large"

```
Solution:
1. Check attachment file size limits
2. Compress large images before transfer
3. Consider alternative file formats
```

Debug Information

Enable browser developer tools to view detailed logs:

1. Press F12 to open developer tools
2. Go to Console tab
3. Reproduce the issue

4. Review console messages for detailed error information

Performance Optimization

1. **Pagination**: Use $skiptoken for large result sets (limited to 100 rows per request)
2. **Filtering**: Apply $filter to reduce data transfer
3. **Field Selection**: Use $select to request only needed fields
4. **Caching**: Cache static data locally when possible
5. **Connection Pooling**: Reuse authentication tokens

Getting Help

For technical support:

1. Check browser console for error messages
2. Verify all prerequisites are met
3. Test with minimal configuration
4. Contact system administrator
5. Review LeadSuccess documentation at http://www.convey.de/

Support

Documentation

- This technical documentation
- LeadSuccess system information: http://www.convey.de/
- Azure AD documentation: https://docs.microsoft.com/azure/active-directory/
- Dynamics 365 Web API: https://docs.microsoft.com/dynamics365/customer-engagement/web-api/
- OData Protocol: http://www.odata.org/documentation/odata-version-2-0

System Requirements

- **Browsers**: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- **Network**: HTTPS required for production
- **Permissions**: Azure AD app registration access
- **LeadSuccess Account**: Valid admin account required

Version Information

- **Application Version**: 1.0.0
- **MSAL Version**: 2.x
- **Dynamics API Version**: 9.2
- **OData Version**: 4.0 (Dynamics) / 2.0 (LeadSuccess)
- **Manual Version**: 0.1
- **Database Version**: 8.2.24

Appendix

Environment URLs

Environment	URL
Production	https://polite-pond-04d9e9503.6.azurestaticapps.net/
Development	http://localhost:5504/

Azure AD Endpoints

Endpoint	URL Template
Authority	`https://login.microsoftonline.com/{tenantId}`
Token	`https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/token`
Authorize	`https://login.microsoftonline.com/{tenantId}/oauth2/v2.0/authorize`

Dynamics 365 API Endpoints

Resource	URL Template
Web API Base	`{resourceUrl}/api/data/v9.2/`
Leads	`{resourceUrl}/api/data/v9.2/leads`

LeadSuccess API Endpoints

Resource	URL Template
Base	`https://{serverName}/{apiName}/`
Metadata	`https://{serverName}/{apiName}/$metadata`
Events	`https://{serverName}/{apiName}/WCE_Event`
Leads	`https://{serverName}/{apiName}/WCE_Lead`
Users	`https://{serverName}/{apiName}/WCE_User`
Countries	`https://{serverName}/{apiName}/WCE_Country`
Attachments	`https://{serverName}/{apiName}/WCE_AttachmentById`

System Architecture

LeadSuccess API - Dynamics Documentation (Postman)

Overview

The LeadSuccess - Dynamics API enables integration and synchronization of data between your system and Microsoft Dynamics CRM. This API provides secure access to core entities: countries, users, events, leads, and attachments.

Important Security Notice

⚠️ **This Postman collection is designed for Web Applications only, not Single Page Applications (SPA)**

Due to Microsoft's security requirements:

- **OAuth2 with PKCE** requires a server-side component to securely handle tokens
- **Client Secret** must be stored securely on the server, never exposed to browsers
- **Redirect URI** points to `http://localhost:3000/api/dynamics/oauth2/callback` indicating a Node.js server endpoint
- **SPA applications** cannot securely store client secrets and require a different OAuth2 flow

Since this application does not integrate a Node.js server, the OAuth2 endpoints are for **web applications with backend servers only**.

Base Configuration

Required Environment Variables

Configure the following variables in your Postman environment:

Variable	Description	Example
servername	*API server URL*	*https://your-server.com*
apiname	*API name/path*	*api/v1*
username	*Basic auth username*	*your-username*
password	*Basic auth password*	*your-password*
dynamicsBaseUrl	Dynamics base URL	https://your-org.crm4.dynamics.com/api/data/v9.2
tenantId	Azure tenant ID	your-tenant-id
clientId	Azure app registration ID	your-client-id
secret	Azure app secret	your-secret
scope	OAuth2 scope	https://your-org.crm4.dynamics.com/.default

Authentication Methods

1. Basic Authentication (WCE API)

- **Type**: Basic Auth
- **Username**: `{{username}}`
- **Password**: `{{password}}`
- **Usage**: All WCE endpoints (Countries, Users, Events, Leads, Attachments)

2. OAuth2 Authentication (Dynamics API) - Web Apps Only

- **Type**: OAuth 2.0 with PKCE
- **Grant Type**: Authorization Code with PKCE
- **Client Authentication**: Body
- **Auth URL**: `https://login.microsoftonline.com/{{tenantId}}/oauth2/v2.0/authorize`
- **Token URL**: `https://login.microsoftonline.com/{{tenantId}}/oauth2/v2.0/token`
- **Redirect URI**: `http://localhost:3000/api/dynamics/oauth2/callback`
- **Scope**: `{{scope}}`

API Endpoints

Countries

Get All Countries

```
GET {{servername}}/{{apiname}}/WCE_Country?$format=json
```

- **Authentication**: Basic Auth
- **Response**: JSON array of country entities
- **OData Format**: Supports OData query parameters

Users

Get All Users

```
GET {{servername}}/{{apiname}}/WCE_User?$format=json
```

- **Authentication**: Basic Auth
- **Response**: JSON array of user entities
- **OData Format**: Supports OData query parameters

Events

Get All Events

```
GET {{servername}}/{{apiname}}/WCE_Event?$format=json
```

- **Authentication**: Basic Auth
- **Response**: JSON array of event entities
- **OData Format**: Supports OData query parameters

Leads

Get All Leads

```
GET {{servername}}/{{apiname}}/WCE_Lead?$format=json
```

- **Authentication**: Basic Auth
- **Response**: JSON array of lead entities with OData structure
- **Pagination**: Supports OData pagination via `$top`, `$skip`

Get Leads by Event ID

```
GET {{servername}}/{{apiname}}/WCE_Lead?$format=json&$filter=EventId eq 'EVENT_ID'
```

- **Authentication**: Basic Auth
- **Parameters**:
  - `EVENT_ID`: UUID of the event to filter by
- **Response**: JSON array of leads belonging to the specified event
- **OData Filter**: Uses OData `$filter` parameter

Attachments

Get Attachment by ID

```
GET{{servername}}/{{apiname}}/WCE_AttachmentById?Id='ATTACHMENT_ID'&$format=json
```

- **Authentication**: Basic Auth
- **Parameters**:
  - `ATTACHMENT_ID`: UUID of the attachment (URL encoded with single quotes)
- **Response**: JSON object containing attachment data

Dynamics CRM (Web Apps Only)

OAuth2 Login

```
GET https://login.microsoftonline.com/common/oauth2/authorize?resource={{dynamicsBaseUrl}}
```

- **Purpose**: Initiates OAuth2 authentication flow
- **Requirement**: Server-side handling required

Get Dynamics Leads

```
GET {{dynamicsBaseUrl}}/leads
```

- **Authentication**: OAuth2 Bearer Token
- **Response**: Dynamics CRM lead entities

Response Format

All WCE API responses follow the OData v2 format:

```
{
  "d": {
    "results": [
      {
        "Property1": "value1",
        "Property2": "value2"
      }
    ],
    "__next": "url_to_next_page" // If pagination available
  }
}
```

Error Handling

HTTP Status Codes

- **200**: Success
- **400**: Bad Request - Check request parameters
- **401**: Unauthorized - Check authentication credentials
- **404**: Not Found - Resource doesn't exist
- **500**: Internal Server Error - Contact API support

Error Response Format

```
{
  "error": {
    "code": "ERROR_CODE",
    "message": "Error description"
  }
}
```

Best Practices

Security

6. **Never expose credentials** in client-side code
7. **Use HTTPS** for all API calls
8. **Store secrets securely** on the server side
9. **Implement token refresh** for OAuth2 flows
10. **Validate SSL certificates** in production

Performance

11. **Use OData filtering** to reduce response size
12. **Implement pagination** for large datasets
13. **Cache responses** when appropriate
14. **Monitor rate limits** if applicable

Data Handling

15. **Validate UUIDs** before making requests
16. **Handle empty responses** gracefully

17. **Parse OData structure** correctly
18. **Check for pagination** in responses

Implementation Notes

For Web Applications

- Implement server-side OAuth2 callback handling
- Store tokens securely on the server
- Use server-to-server API calls
- Implement proper session management

For SPA Applications

- Use Basic Auth endpoints only
- Consider implementing a backend proxy
- Do not attempt OAuth2 flow without server
- Use environment-specific configurations

Support

For technical support or API access issues, please contact your LeadSuccess administrator


```
{
  "info": {
    "_postman_id": "ec86e503-6edc-483c-abf0-9a731e93955d",
    "name": "LeadSuccess  API - Dynamics",
    "schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
    "_exporter_id": "17182298"
  },
  "item": [
    {
      "name": "WCE_Country",
      "item": [
        {
          "name": "Get All Country",
          "request": {
            "auth": {
              "type": "basic",
              "basic": [
                {
                  "key": "username",
                  "value": "{{username}}",
                  "type": "string"
                },
                {
                  "key": "password",
                  "value": "{{password}}",
                  "type": "string"
                }
              ]
            },
```

```
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{servername}}/{{apiname}}/WCE_Country?$format=json",
          "host": ["{{servername}}"],
          "path": ["{{apiname}}", "WCE_Country"],
          "query": [
            {
              "key": "$format",
              "value": "json"
            }
          ]
        }
      },
      "response": []
    }
  ]
},
{
  "name": "WCE_User",
  "item": [
    {
      "name": "Get All User",
      "request": {
        "auth": {
          "type": "basic",
          "basic": [
            {
              "key": "username",
              "value": "{{username}}",
              "type": "string"
            },
            {
              "key": "password",
              "value": "{{password}}",
              "type": "string"
            }
          ]
        },
        "method": "GET",
        "header": [],
        "url": {
          "raw": "{{servername}}/{{apiname}}/WCE_User?$format=json",
          "host": ["{{servername}}"],
          "path": ["{{apiname}}", "WCE_User"],
          "query": [
            {
              "key": "$format",
              "value": "json"
            }
          ]
        }
      },
      "response": []
    }
  ]
},
{
```

```
      "name": "WCE_Events",
      "item": [
        {
          "name": "WCE_Leads",
          "item": [
            {
              "name": "Get All Leads",
              "event": [
                {
                  "listen": "test",
                  "script": {
                    "exec": [
                      "// Test response status\r",
                      "pm.test(\"Status code is 200\", function () {\r",
                      "    pm.response.to.have.status(200);\r",
                      "});\r",
                      "\r",
                      "// Test response time\r",
                      "pm.test(\"Response time is less than 5000ms\", function
() {\r",
                      "
pm.expect(pm.response.responseTime).to.be.below(5000);\r",
                      "});\r",
                      "\r",
                      "// Test response headers\r",
                      "pm.test(\"Response has JSON content type\", function ()
{\r",
                      "    pm.expect(pm.response.headers.get(\"Content-
Type\")).to.include(\"application/json\");\r",
                      "});\r",
                      "\r",
                      "// Parse and validate response body\r",
                      "pm.test(\"Response body has valid structure\", function
() {\r",
                      "    const responseJson = pm.response.json();\r",
                      "    \r",
                      "    // Check for OData structure\r",
                      "    pm.expect(responseJson).to.have.property('d');\r",
                      "
pm.expect(responseJson.d).to.have.property('results');\r",
                      "
pm.expect(responseJson.d.results).to.be.an('array');\r",
                      "    \r",
                      "    console.log(\"Total leads found:\",
responseJson.d.results.length);\r",
                      "    \r",
                      "    // Store results count for other tests\r",
                      "    pm.environment.set(\"totalLeadsCount\",
responseJson.d.results.length);\r",
                      "});\r",
                      "\r",
                      "// Test individual lead structure (if leads exist)\r",
                      "pm.test(\"Lead objects have required fields\", function
() {\r",
                      "    const responseJson = pm.response.json();\r",
                      "    const leads = responseJson.d.results;\r",
                      "    \r",
                      "    if (leads.length > 0) {\r",
```

```
                        "        const firstLead = leads[0];\r",
                        "        \r",
                        "        // Required fields validation\r",
                        "        const requiredFields = [\r",
                        "            'LeadId', 'CreatedOn', 'ModifiedOn',
'CreatedBy', \r",
                        "            'EventId', 'KontaktViewId'\r",
                        "        ];\r",
                        "        \r",
                        "        requiredFields.forEach(field => {\r",
                        "
pm.expect(firstLead).to.have.property(field);\r",
                        "        });\r",
                        "        \r",
                        "        // Test UUID format for LeadId\r",
                        "        const uuidRegex = /^[0-9a-f]{8}-[0-9a-f]{4}-[0-
9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/i;\r",
                        "
pm.expect(firstLead.LeadId).to.match(uuidRegex);\r",
                        "        \r",
                        "        console.log(\"Sample Lead ID:\",
firstLead.LeadId);\r",
                        "        console.log(\"Sample Lead Name:\",
`${firstLead.FirstName || ''} ${firstLead.LastName || ''}`.trim());\r",
                        "        \r",
                        "        // Store sample lead ID for other tests\r",
                        "        pm.environment.set(\"sampleLeadId\",
firstLead.LeadId);\r",
                        "        if (firstLead.EventId) {\r",
                        "            pm.environment.set(\"sampleEventId\",
firstLead.EventId);\r",
                        "        }\r",
                        "    }\r",
                        "});\r",
                        "\r",
                        "// Test pagination information\r",
                        "pm.test(\"Check pagination support\", function () {\r",
                        "    const responseJson = pm.response.json();\r",
                        "    \r",
                        "    if (responseJson.d.__next) {\r",
                        "        console.log(\"Next page URL available:\",
responseJson.d.__next);\r",
                        "        pm.environment.set(\"nextPageUrl\",
responseJson.d.__next);\r",
                        "    } else {\r",
                        "        console.log(\"No next page available - all
results returned\");\r",
                        "    }\r",
                        "});\r",
                        "\r",
                        "// Log response summary\r",
                        "const responseJson = pm.response.json();\r",
                        "console.log(\"=== RESPONSE SUMMARY ===\");\r",
                        "console.log(\"Status:\", pm.response.status);\r",
                        "console.log(\"Response Time:\",
pm.response.responseTime + \"ms\");\r",
                        "console.log(\"Total Leads:\",
responseJson.d.results.length);\r",
```

```
                    "console.log(\"Has Next Page:\",
!!responseJson.d.__next);"
                  ],
                  "type": "text/javascript",
                  "packages": {}
                }
              },
              {
                "listen": "prerequest",
                "script": {
                  "exec": [
                    "// Set timestamp for this request\r",
                    "pm.environment.set(\"requestTimestamp\", new
Date().toISOString());\r",
                    "\r",
                    "// Log request details\r",
                    "console.log(\"=== GET ALL LEADS REQUEST ===\");\r",
                    "console.log(\"URL:\", pm.request.url.toString());\r",
                    "console.log(\"Timestamp:\",
pm.environment.get(\"requestTimestamp\"));\r",
                    "\r",
                    "// Validate required environment variables\r",
                    "const requiredVars = ['servername', 'apiname',
'username', 'password'];\r",
                    "let missingVars = [];\r",
                    "\r",
                    "requiredVars.forEach(varName => {\r",
                    "    if (!pm.environment.get(varName)) {\r",
                    "        missingVars.push(varName);\r",
                    "    }\r",
                    "});\r",
                    "\r",
                    "if (missingVars.length > 0) {\r",
                    "    console.error(\"Missing required environment
variables:\", missingVars);\r",
                    "    throw new Error(`Please set the following
environment variables: ${missingVars.join(', ')}`);\r",
                    "}"
                  ],
                  "type": "text/javascript",
                  "packages": {}
                }
              }
            ],
            "request": {
              "auth": {
                "type": "basic",
                "basic": [
                  {
                    "key": "username",
                    "value": "{{username}}",
                    "type": "string"
                  },
                  {
                    "key": "password",
                    "value": "{{password}}",
                    "type": "string"
                  }
```

```
                ]
              },
              "method": "GET",
              "header": [],
              "url": {
                "raw": "{{servername}}/{{apiname}}/WCE_Lead?$format=json",
                "host": ["{{servername}}"],
                "path": ["{{apiname}}", "WCE_Lead"],
                "query": [
                  {
                    "key": "$format",
                    "value": "json"
                  }
                ]
              }
            },
            "response": []
          },
          {
            "name": "Get  WCE Lead by EventId",
            "event": [
              {
                "listen": "test",
                "script": {
                  "exec": [
                    "// Test response status\r",
                    "pm.test(\"Status code is 200\", function () {\r",
                    "    pm.response.to.have.status(200);\r",
                    "});\r",
                    "\r",
                    "// Test response time\r",
                    "pm.test(\"Response time is less than 5000ms\", function
() {\r",
                    "
pm.expect(pm.response.responseTime).to.be.below(5000);\r",
                    "});\r",
                    "\r",
                    "// Test response headers\r",
                    "pm.test(\"Response has JSON content type\", function ()
{\r",
                    "    pm.expect(pm.response.headers.get(\"Content-
Type\")).to.include(\"application/json\");\r",
                    "});\r",
                    "\r",
                    "// Parse and validate response body\r",
                    "pm.test(\"Response body has valid OData structure\",
function () {\r",
                    "    const responseJson = pm.response.json();\r",
                    "    \r",
                    "    pm.expect(responseJson).to.have.property('d');\r",
                    "
pm.expect(responseJson.d).to.have.property('results');\r",
                    "
pm.expect(responseJson.d.results).to.be.an('array');\r",
                    "    \r",
                    "    console.log(\"Leads found for Event ID:\",
responseJson.d.results.length);\r",
```

```
                        "    pm.environment.set(\"eventLeadsCount\",
responseJson.d.results.length);\r",
                        "});\r",
                        "\r",
                        "// Test EventId filtering\r",
                        "pm.test(\"All returned leads belong to the requested
EventId\", function () {\r",
                        "    const responseJson = pm.response.json();\r",
                        "    const leads = responseJson.d.results;\r",
                        "    const requestedEventId =
pm.environment.get(\"eventId\");\r",
                        "    \r",
                        "    if (leads.length > 0) {\r",
                        "        leads.forEach((lead, index) => {\r",
                        "            pm.expect(lead.EventId, `Lead ${index + 1}
EventId mismatch`).to.equal(requestedEventId);\r",
                        "        });\r",
                        "        \r",
                        "        console.log(\"✓ All leads belong to EventId:\",
requestedEventId);\r",
                        "        \r",
                        "        // Store lead details for potential follow-up
tests\r",
                        "        const firstLead = leads[0];\r",
                        "        pm.environment.set(\"eventFilteredLeadId\",
firstLead.LeadId);\r",
                        "        \r",
                        "        console.log(\"Sample filtered lead:\", {\r",
                        "            LeadId: firstLead.LeadId,\r",
                        "            Name: `${firstLead.FirstName || ''}
${firstLead.LastName || ''}`.trim(),\r",
                        "            Company: firstLead.CompanyName,\r",
                        "            Email: firstLead.EMailAddress1\r",
                        "        });\r",
                        "    } else {\r",
                        "        console.log(\"No leads found for EventId:\",
requestedEventId);\r",
                        "    }\r",
                        "});\r",
                        "\r",
                        "// Test lead data quality\r",
                        "pm.test(\"Filtered leads have expected data quality\",
function () {\r",
                        "    const responseJson = pm.response.json();\r",
                        "    const leads = responseJson.d.results;\r",
                        "    \r",
                        "    if (leads.length > 0) {\r",
                        "        const leadsWithEmail = leads.filter(lead =>
lead.EMailAddress1 && lead.EMailAddress1.trim() !== '');\r",
                        "        const leadsWithPhone = leads.filter(lead =>
\r",
                        "            (lead.Address1_Telephone1 &&
lead.Address1_Telephone1.trim() !== '') ||\r",
                        "            (lead.MobilePhone &&
lead.MobilePhone.trim() !== '')\r",
                        "        );\r",
                        "        const leadsWithCompany = leads.filter(lead =>
lead.CompanyName && lead.CompanyName.trim() !== '');\r",
```

```
                    "        \r",
                    "        console.log(\"Data Quality Analysis:\");\r",
                    "        console.log(\"- Leads with email:\",
leadsWithEmail.length,
`(${Math.round(leadsWithEmail.length/leads.length*100)}%)`);\r",
                    "        console.log(\"- Leads with phone:\",
leadsWithPhone.length,
`(${Math.round(leadsWithPhone.length/leads.length*100)}%)`);\r",
                    "        console.log(\"- Leads with company:\",
leadsWithCompany.length,
`(${Math.round(leadsWithCompany.length/leads.length*100)}%)`);\r",
                    "        \r",
                    "        // Basic quality check - at least 50% should
have email or phone\r",
                    "        const leadsWithContact = leads.filter(lead =>
\r",
                    "            (lead.EMailAddress1 &&
lead.EMailAddress1.trim() !== '') ||\r",
                    "            (lead.Address1_Telephone1 &&
lead.Address1_Telephone1.trim() !== '') ||\r",
                    "            (lead.MobilePhone &&
lead.MobilePhone.trim() !== '')\r",
                    "        );\r",
                    "        \r",
                    "        const contactDataQuality =
leadsWithContact.length / leads.length;\r",
                    "        pm.expect(contactDataQuality).to.be.above(0.3,
\"At least 30% of leads should have contact information\");\r",
                    "    }\r",
                    "});\r",
                    "\r",
                    "// Test attachment information\r",
                    "pm.test(\"Check for attachments in leads\", function ()
{\r",
                    "    const responseJson = pm.response.json();\r",
                    "    const leads = responseJson.d.results;\r",
                    "    \r",
                    "    if (leads.length > 0) {\r",
                    "        const leadsWithAttachments = leads.filter(lead
=> \r",
                    "            lead.AttachmentIdList &&
lead.AttachmentIdList.trim() !== ''\r",
                    "        );\r",
                    "        \r",
                    "        if (leadsWithAttachments.length > 0) {\r",
                    "            console.log(\"Leads with attachments:\",
leadsWithAttachments.length);\r",
                    "            \r",
                    "            // Store sample attachment IDs for
potential testing\r",
                    "            const sampleAttachmentList =
leadsWithAttachments[0].AttachmentIdList;\r",
                    "            const attachmentIds =
sampleAttachmentList.split(',').map(id => id.trim()).filter(id => id !==
'');\r",
                    "            \r",
                    "            if (attachmentIds.length > 0) {\r",
```

```
                        "
pm.environment.set(\"sampleAttachmentId\", attachmentIds[0]);\r",
                        "                console.log(\"Sample attachment ID:\",
attachmentIds[0]);\r",
                        "            }\r",
                        "        } else {\r",
                        "            console.log(\"No leads with attachments
found\");\r",
                        "        }\r",
                        "    }\r",
                        "});\r",
                        "\r",
                        "// Log response summary\r",
                        "const responseJson = pm.response.json();\r",
                        "console.log(\"=== FILTERED RESPONSE SUMMARY ===\");\r",
                        "console.log(\"Status:\", pm.response.status);\r",
                        "console.log(\"Response Time:\",
pm.response.responseTime + \"ms\");\r",
                        "console.log(\"Event ID:\",
pm.environment.get(\"eventId\"));\r",
                        "console.log(\"Filtered Leads Count:\",
responseJson.d.results.length);\r",
                        "console.log(\"Has Next Page:\",
!!responseJson.d.__next);"
                    ],
                    "type": "text/javascript",
                    "packages": {}
                }
            },
            {
                "listen": "prerequest",
                "script": {
                    "exec": [
                        "// Set timestamp for this request\r",
                        "pm.environment.set(\"requestTimestamp\", new
Date().toISOString());\r",
                        "\r",
                        "// Log request details\r",
                        "console.log(\"=== GET LEADS BY EVENT ID REQUEST
===\");\r",
                        "\r",
                        "// Validate required environment variables (eventId is
optional - will be auto-set)\r",
                        "const requiredVars = ['servername', 'apiname',
'username', 'password'];\r",
                        "let missingVars = [];\r",
                        "\r",
                        "requiredVars.forEach(varName => {\r",
                        "    if (!pm.environment.get(varName)) {\r",
                        "        missingVars.push(varName);\r",
                        "    }\r",
                        "});\r",
                        "\r",
                        "if (missingVars.length > 0) {\r",
                        "    console.error(\"Missing required environment
variables:\", missingVars);\r",
                        "    throw new Error(`Please set the following
environment variables: ${missingVars.join(', ')}`);\r",
```

```
                    "}\r",
                    "\r",
                    "// Handle EventId - auto-populate if not set\r",
                    "let eventId = pm.environment.get(\"eventId\");\r",
                    "\r",
                    "// Try to use sample EventId from previous \"Get All
Leads\" request\r",
                    "if (!eventId && pm.environment.get(\"sampleEventId\"))
{\r",
                    "    eventId =
pm.environment.get(\"sampleEventId\");\r",
                    "    pm.environment.set(\"eventId\", eventId);\r",
                    "    console.log(\"✓ Auto-populated EventId from
previous request:\", eventId);\r",
                    "}\r",
                    "\r",
                    "// If still no eventId, provide helpful guidance\r",
                    "if (!eventId) {\r",
                    "    console.warn(\" No EventId found!\");    \r",
                    "    throw new Error(\"EventId not found. Please run
'Get All Leads' first or set eventId manually.\");\r",
                    "}\r",
                    "\r",
                    "// Validate EventId format (UUID)\r",
                    "const uuidRegex = /^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-
f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/i;\r",
                    "if (!uuidRegex.test(eventId)) {\r",
                    "    console.warn(\"⚠  EventId does not appear to be a
valid UUID format:\", eventId);\r",
                    "    console.log(\"Expected format: xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx\");\r",
                    "}\r",
                    "\r",
                    "console.log(\"URL:\", pm.request.url.toString());\r",
                    "console.log(\"Event ID:\", eventId);\r",
                    "console.log(\"Timestamp:\",
pm.environment.get(\"requestTimestamp\"));"
                  ],
                  "type": "text/javascript",
                  "packages": {}
                }
              }
            ],
            "request": {
              "auth": {
                "type": "basic",
                "basic": [
                  {
                    "key": "username",
                    "value": "{{username}}",
                    "type": "string"
                  },
                  {
                    "key": "password",
                    "value": "{{password}}",
                    "type": "string"
                  }
                ]
```

```
                },
                "method": "GET",
                "header": [],
                "url": {
                  "raw":
"{{servername}}/{{apiname}}/WCE_Lead?$format=json&$filter=EventId eq
'{{eventId}}'",
                  "host": ["{{servername}}"],
                  "path": ["{{apiname}}", "WCE_Lead"],
                  "query": [
                    {
                      "key": "$format",
                      "value": "json"
                    },
                    {
                      "key": "$filter",
                      "value": "EventId eq '{{eventId}}'"
                    }
                  ]
                }
              },
              "response": []
            }
          ]
        },
        {
          "name": "Get All Events",
          "request": {
            "auth": {
              "type": "basic",
              "basic": [
                {
                  "key": "username",
                  "value": "{{username}}",
                  "type": "string"
                },
                {
                  "key": "password",
                  "value": "{{password}}",
                  "type": "string"
                }
              ]
            },
            "method": "GET",
            "header": [],
            "url": {
              "raw": "{{servername}}/{{apiname}}/WCE_Event?$format=json",
              "host": ["{{servername}}"],
              "path": ["{{apiname}}", "WCE_Event"],
              "query": [
                {
                  "key": "$format",
                  "value": "json"
                }
              ]
            }
          },
          "response": []
```

```
        }
      ]
    },
    {
      "name": "WCE_Attachments",
      "item": [
        {
          "name": "Get Attachment by ID",
          "event": [
            {
              "listen": "prerequest",
              "script": {
                "exec": [""],
                "type": "text/javascript",
                "packages": {}
              }
            },
            {
              "listen": "test",
              "script": {
                "exec": [""],
                "type": "text/javascript",
                "packages": {}
              }
            }
          ],
          "request": {
            "auth": {
              "type": "basic",
              "basic": [
                {
                  "key": "username",
                  "value": "{{username}}",
                  "type": "string"
                },
                {
                  "key": "password",
                  "value": "{{password}}",
                  "type": "string"
                }
              ]
            },
            "method": "GET",
            "header": [],
            "url": {
              "raw":
"{{servername}}/{{apiname}}/WCE_AttachmentById?Id=%27{{sampleAttachmentId}}%27
&$format=json",
              "host": ["{{servername}}"],
              "path": ["{{apiname}}", "WCE_AttachmentById"],
              "query": [
                {
                  "key": "Id",
                  "value": "%27{{sampleAttachmentId}}%27"
                },
                {
                  "key": "$format",
                  "value": "json"
```

```
                }
              ]
            }
          },
          "response": []
        }
      ]
    },
    {
      "name": "Dynamics",
      "item": [
        {
          "name": "Login",
          "request": {
            "method": "GET",
            "header": [],
            "url": {
              "raw":
"https://login.microsoftonline.com/common/oauth2/authorize?resource={{dynamics
BaseUrl}}",
              "protocol": "https",
              "host": ["login", "microsoftonline", "com"],
              "path": ["common", "oauth2", "authorize"],
              "query": [
                {
                  "key": "resource",
                  "value": "{{dynamicsBaseUrl}}"
                }
              ]
            }
          },
          "response": []
        },
        {
          "name": "Get All Dynamics Leads",
          "request": {
            "method": "GET",
            "header": [],
            "url": {
              "raw": "{{dynamicsBaseUrl}}/leads",
              "host": ["{{dynamicsBaseUrl}}"],
              "path": ["leads"]
            }
          },
          "response": []
        },
        {
          "name": "PCKE",
          "request": {
            "method": "GET",
            "header": []
          },
          "response": []
        }
      ],
      "auth": {
        "type": "oauth2",
        "oauth2": [
```

```
            {
              "key": "refreshRequestParams",
              "value": [],
              "type": "any"
            },
            {
              "key": "tokenRequestParams",
              "value": [],
              "type": "any"
            },
            {
              "key": "authRequestParams",
              "value": [],
              "type": "any"
            },
            {
              "key": "tokenName",
              "value": "New Oauth Token 2.0",
              "type": "string"
            },
            {
              "key": "code_verifier",
              "value":
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789000000",
              "type": "string"
            },
            {
              "key": "challengeAlgorithm",
              "value": "S256",
              "type": "string"
            },
            {
              "key": "scope",
              "value": "https://org54e712bc.crm4.dynamics.com/.default",
              "type": "string"
            },
            {
              "key": "redirect_uri",
              "value": "http://localhost:3000/api/dynamics/oauth2/callback",
              "type": "string"
            },
            {
              "key": "grant_type",
              "value": "authorization_code_with_pkce",
              "type": "string"
            },
            {
              "key": "clientSecret",
              "value": "A5A8Q~G_.RnP1whB8Qcw8TqrOFrOt5._I4PSxaX8",
              "type": "string"
            },
            {
              "key": "clientId",
              "value": "d97e92f9-5dd8-430b-8c3c-d084d029d17f",
              "type": "string"
            },
            {
              "key": "authUrl",
```

```
            "value": "https://login.microsoftonline.com/c8e999a5-7cc1-4063-
a8ce-791ed474039f/oauth2/v2.0/authorize",
            "type": "string"
          },
          {
            "key": "addTokenTo",
            "value": "header",
            "type": "string"
          },
          {
            "key": "client_authentication",
            "value": "body",
            "type": "string"
          },
          {
            "key": "accessTokenUrl",
            "value": "https://login.microsoftonline.com/c8e999a5-7cc1-4063-
a8ce-791ed474039f/oauth2/v2.0/token",
            "type": "string"
          }
        ]
      },
      "event": [
        {
          "listen": "prerequest",
          "script": {
            "type": "text/javascript",
            "packages": {},
            "exec": [""]
          }
        },
        {
          "listen": "test",
          "script": {
            "type": "text/javascript",
            "packages": {},
            "exec": [""]
          }
        }
      ]
    }
  ],
  "auth": {
    "type": "oauth2",
    "oauth2": [
      {
        "key": "tokenName",
        "value": "Dynamics Oauth Token 2.0",
        "type": "string"
      },
      {
        "key": "scope",
        "value": "{{scope}}",
        "type": "string"
      },
      {
        "key": "clientSecret",
        "value": "{{secret}}",
```

```
        "type": "string"
      },
      {
        "key": "clientId",
        "value": "{{clientId}}",
        "type": "string"
      },
      {
        "key": "accessTokenUrl",
        "value":
"https://login.microsoftonline.com/{{tenantId}}/oauth2/v2.0/token",
        "type": "string"
      },
      {
        "key": "authUrl",
        "value":
"https://login.microsoftonline.com/{{tenantId}}/oauth2/v2.0/authorize",
        "type": "string"
      },
      {
        "key": "refreshRequestParams",
        "value": [],
        "type": "any"
      },
      {
        "key": "tokenRequestParams",
        "value": [],
        "type": "any"
      },
      {
        "key": "authRequestParams",
        "value": [],
        "type": "any"
      },
      {
        "key": "code_verifier",
        "value":
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789000000",
        "type": "string"
      },
      {
        "key": "challengeAlgorithm",
        "value": "S256",
        "type": "string"
      },
      {
        "key": "redirect_uri",
        "value": "http://localhost:3000/api/dynamics/oauth2/callback",
        "type": "string"
      },
      {
        "key": "grant_type",
        "value": "authorization_code_with_pkce",
        "type": "string"
      },
      {
        "key": "addTokenTo",
        "value": "header",
```

```
        "type": "string"
      },
      {
        "key": "client_authentication",
        "value": "body",
        "type": "string"
      }
    ]
  },
  "event": [
    {
      "listen": "prerequest",
      "script": {
        "type": "text/javascript",
        "packages": {},
        "exec": [""]
      }
    },
    {
      "listen": "test",
      "script": {
        "type": "text/javascript",
        "packages": {},
        "exec": [""]
      }
    }
  ]
}
```