Dynamic Action Plans for Experience Cloud - Complete Deployment Guide

🎯 Overview

This solution enables external guest users to create dynamic action plans through Experience Cloud, which are then synchronized with native Salesforce Action Plans. The system provides a secure, scalable bridge between public-facing forms and internal Salesforce processes.

🏗️ Architecture Summary

```
Guest User (Experience Cloud)
   ↓ [Submits via LWC]
Custom Objects (Bridge Layer)
   ↓ [Platform Events]
Async Processing (Batch/Queueable)
   ↓ [Integration Service]
Native Action Plans (Internal Salesforce)
```

📁 Complete File List

```
force-app/main/default/
├── objects/
│   ├── Custom_Action_Plan__c/
│   ├── Custom_Task__c/
│   ├── Task_Template__c/
│   ├── Action_Plan_Submission_Log__c/
│   └── Action_Plan_Event__e/
├── classes/
│   ├── DynamicActionPlanController.cls
```

```
│   ├── DynamicActionPlanController.cls-meta.xml
│   ├── ActionPlanIntegrationService.cls
│   ├── ActionPlanIntegrationService.cls-meta.xml
│   ├── ActionPlanEventHandler.cls
│   ├── ActionPlanEventHandler.cls-meta.xml
│   ├── ActionPlanSyncBatch.cls
│   ├── ActionPlanSyncBatch.cls-meta.xml
│   ├── ActionPlanMonitorController.cls
│   ├── ActionPlanMonitorController.cls-meta.xml
│   ├── ActionPlanSecurityUtils.cls
│   ├── ActionPlanSecurityUtils.cls-meta.xml
│   ├── DynamicActionPlanControllerTest.cls
│   └── DynamicActionPlanControllerTest.cls-meta.xml
├── triggers/
│   ├── ActionPlanEventTrigger.trigger
│   └── ActionPlanEventTrigger.trigger-meta.xml
├── lwc/
│   ├── dynamicActionPlanBuilder/
│   │   ├── dynamicActionPlanBuilder.js
│   │   ├── dynamicActionPlanBuilder.html
│   │   ├── dynamicActionPlanBuilder.css
│   │   └── dynamicActionPlanBuilder.js-meta.xml
│   └── actionPlanMonitor/
│       ├── actionPlanMonitor.js
│       ├── actionPlanMonitor.html
│       ├── actionPlanMonitor.css
│       └── actionPlanMonitor.js-meta.xml
├── permissionsets/
│   └── Guest_Action_Plan_Creator.permissionset-meta.xml
├── customSettings/
│   └── Action_Plan_Settings__c.object-meta.xml
├── labels/
│   └── CustomLabels.labels-meta.xml
```

```
└── remoteSiteSettings/
    └── RemoteSiteSettings.remoteSite-meta.xml
```

🚀 Pre-Deployment Requirements

System Requirements

Component	Requirement
Salesforce Edition	Enterprise or Higher
Experience Cloud	Enabled with available licenses
Sales Cloud	Action Plans feature enabled
Platform Events	Available allocation (min 10/hour)
API Calls	Sufficient daily allocation

Enable Required Features

1. **Experience Cloud**

```
Setup → Digital Experiences → Settings → Enable
```

2. **Action Plans**

```
Setup → Feature Settings → Sales → Action Plans → Enable
```

3. **Platform Events**

```
Setup → Platform Events → Allocations → Verify
```

Development Environment

- [ ] Salesforce CLI v2.0+ installed
- [ ] VS Code with Salesforce Extensions Pack
- [ ] Git configured
- [ ] Sandbox or Developer org ready

📋 Deployment Order of Operations

Phase 1: Foundation (Day 1)

Step 1.1: Authenticate to Org

```bash
# Authenticate to sandbox
sfdx auth:web:login -a MyOrgAlias -r https://test.salesforce.com

# Verify connection
sfdx force:org:display -u MyOrgAlias
```

Step 1.2: Deploy Custom Objects

```bash
# Deploy objects and fields
sfdx force:source:deploy -p force-app/main/default/objects -u MyOrgAlias

# Verify in UI
sfdx force:org:open -p /lightning/setup/ObjectManager/home -u MyOrgAlias
```

Step 1.3: Deploy Platform Event

```bash
# Deploy platform event
sfdx force:source:deploy -p force-app/main/default/objects/Action_Plan_Event__e -u MyOrgAlias
```

Phase 2: Core Logic (Day 2)

Step 2.1: Deploy Apex Classes

```bash
# Deploy all Apex classes
sfdx force:source:deploy -p force-app/main/default/classes -u MyOrgAlias

# Deploy triggers
sfdx force:source:deploy -p force-app/main/default/triggers -u MyOrgAlias
```

Step 2.2: Run Tests

```bash
# Run specific test class
sfdx force:apex:test:run -n DynamicActionPlanControllerTest -r human -u MyOrgAlias

# Run all tests with coverage
sfdx force:apex:test:run -l RunLocalTests -c -r human -u MyOrgAlias
```

Phase 3: User Interface (Day 3)

Step 3.1: Deploy Lightning Web Components

```bash
```

```
# Deploy LWCs
sfdx force:source:deploy -p force-app/main/default/lwc -u MyOrgAlias
```

Step 3.2: Deploy Configuration Files

```bash
# Deploy permission sets
sfdx force:source:deploy -p force-app/main/default/permissionsets -u MyOrgAlias

# Deploy custom settings
sfdx force:source:deploy -p force-app/main/default/customSettings -u MyOrgAlias

# Deploy labels
sfdx force:source:deploy -p force-app/main/default/labels -u MyOrgAlias

# Deploy remote site settings
sfdx force:source:deploy -p force-app/main/default/remoteSiteSettings -u MyOrgAlias
```

Phase 4: Experience Cloud Setup (Day 4)

Step 4.1: Create Experience Site

1. Navigate to **Setup → Digital Experiences → All Sites**
2. Click **New**
3. Choose **Build Your Own (Aura)**
4. Configure:
   - Name: `Action Plan Portal`
   - URL: `actionplans`
   - Template: Build Your Own

Step 4.2: Configure Guest User

1. Go to Experience Builder
2. Settings → General
3. Guest User Profile → View
4. Enable:
   - ☑ API Enabled
   - ☑ View Setup and Configuration
5. Assign Permission Set: `Guest_Action_Plan_Creator`

Step 4.3: Add Components to Pages

1. In Experience Builder, go to Home page
2. Drag `dynamicActionPlanBuilder` component
3. Configure properties:

```
Related Object Type: Lead
Enable CAPTCHA: true
```

Phase 5: Configuration (Day 5)

Step 5.1: Initialize Custom Settings

```apex
```

```apex
// Execute in Anonymous Apex
Action_Plan_Settings__c settings = Action_Plan_Settings__c.getOrgDefaults();
settings.Enable_Auto_Sync__c = true;
settings.Rate_Limit_Per_Hour__c = 5;
settings.Max_Tasks_Per_Plan__c = 20;
settings.Batch_Size__c = 50;
settings.Enable_Email_Notifications__c = true;
settings.Admin_Email__c = 'admin@yourcompany.com';
settings.Experience_Cloud_URL__c = 'https://yourdomain.my.site.com/actionplans';
settings.Enable_CAPTCHA__c = true;
settings.CAPTCHA_Site_Key__c = 'your-recaptcha-site-key';
settings.CAPTCHA_Secret_Key__c = 'your-recaptcha-secret-key';
settings.Data_Retention_Days__c = 90;
upsert settings;
```

Step 5.2: Create Task Templates

```apex
```

```
// Execute in Anonymous Apex
List<Task_Template__c> templates = new List<Task_Template__c>{
    new Task_Template__c(
        Name = 'Initial Contact',
        Description__c = 'Make initial contact with customer',
        Category__c = 'Sales',
        Default_Duration_Days__c = 1,
        Is_Active__c = true,
        Is_Public__c = true,
        Display_Order__c = 10,
        Task_Type__c = 'Call',
        Default_Priority__c = 'High'
    ),
    new Task_Template__c(
        Name = 'Send Documentation',
        Description__c = 'Send required documentation',
        Category__c = 'Sales',
        Default_Duration_Days__c = 2,
        Is_Active__c = true,
        Is_Public__c = true,
        Display_Order__c = 20,
        Task_Type__c = 'Email',
        Default_Priority__c = 'Medium'
    ),
    new Task_Template__c(
        Name = 'Follow Up',
        Description__c = 'Follow up on initial contact',
        Category__c = 'Sales',
        Default_Duration_Days__c = 3,
        Is_Active__c = true,
        Is_Public__c = true,
        Display_Order__c = 30,
        Task_Type__c = 'Call',
```

```
        Default_Priority__c = 'Low'
    )
};
insert templates;
```

Step 5.3: Schedule Batch Jobs

```apex
// Schedule hourly sync
ActionPlanSyncBatch.ActionPlanBatchScheduler scheduler =
    new ActionPlanSyncBatch.ActionPlanBatchScheduler();
System.schedule('Hourly Action Plan Sync', '0 0 * * * ?', scheduler);

// Schedule daily cleanup
ActionPlanEventHandler.ActionPlanScheduler cleanupScheduler =
    new ActionPlanEventHandler.ActionPlanScheduler();
System.schedule('Daily Action Plan Cleanup', '0 0 2 * * ?', cleanupScheduler);
```

✅ Post-Deployment Testing

Test 1: Guest User Access

1. Open incognito browser
2. Navigate to: `https://yourdomain.my.site.com/actionplans`
3. Verify page loads without login

Test 2: Create Action Plan

1. Enter test user information:
    - Email: test@example.com
    - Name: Test User

2. Add 2-3 tasks
3. Submit and note reference number

Test 3: Verify Sync

1. Login to Salesforce
2. Navigate to App Launcher → Action Plans
3. Verify new action plan created
4. Check Lead/Contact created with email

Test 4: Monitor Dashboard

1. Add `actionPlanMonitor` component to Lightning App
2. Verify metrics display
3. Test "Process Pending" button

🔍 Monitoring & Maintenance

Daily Monitoring Tasks

```apex
```

```
// Check sync status
SELECT COUNT(Id), Status__c
FROM Custom_Action_Plan__c
WHERE CreatedDate = TODAY
GROUP BY Status__c

// Check for errors
SELECT Id, External_Reference_Id__c, Error_Message__c
FROM Custom_Action_Plan__c
WHERE Status__c = 'Failed'
AND CreatedDate = LAST_N_DAYS:7
```

Weekly Tasks

- Review sync success rate (target: >95%)
- Check storage usage
- Review security logs
- Update task templates based on usage

Monthly Tasks

- Clean up logs older than 90 days
- Review and optimize batch performance
- Audit guest user submissions

🛠️ Troubleshooting Guide

Issue: Guest User "Insufficient Privileges"

Solution:

```apex
// Verify permission set assignment
SELECT Id, PermissionSetId, AssigneeId
FROM PermissionSetAssignment
WHERE PermissionSet.Name = 'Guest_Action_Plan_Creator'
AND Assignee.UserType = 'Guest'
```

Issue: Action Plans Not Syncing

Check:

1. Platform Event limits: `Setup → Platform Events → Usage`
2. Batch job status: `Setup → Apex Jobs`
3. Error logs: Query `Action_Plan_Submission_Log__c`

Issue: Rate Limiting Not Working

Verify:

```apex
Action_Plan_Settings__c settings = Action_Plan_Settings__c.getOrgDefaults();
System.debug('Rate Limit: ' + settings.Rate_Limit_Per_Hour__c);
```

Issue: CAPTCHA Not Displaying

Check:

1. Remote Site Settings active
2. CAPTCHA keys in Custom Settings
3. Browser console for errors

🔐 Security Best Practices

Data Protection

- ✅ All inputs sanitized via `ActionPlanSecurityUtils`
- ✅ XSS protection enabled
- ✅ SQL injection prevention
- ✅ CSRF token validation

Rate Limiting

- ✅ 5 submissions/hour/email (configurable)
- ✅ CAPTCHA required
- ✅ IP tracking enabled

Audit Trail

- ✅ All submissions logged
- ✅ Security events tracked
- ✅ Error details captured

📊 Performance Optimization

Batch Processing

- Default batch size: 50 records
- Scheduled every hour
- Automatic retry for failed records

Governor Limits

- SOQL queries optimized with selective filters
- Bulk DML operations
- Platform Cache ready (optional)

Page Performance

- LWC lazy loading
- Minimal server calls
- Client-side validation

📞 Support Resources

Documentation Links

- Salesforce Action Plans
- Experience Cloud Setup
- Platform Events Guide

Contact Information

- Technical Support: support@yourcompany.com
- Admin Email: admin@yourcompany.com

📝 Version History

Version	Date	Changes
1.0.0	2025	Initial release
1.0.1	TBD	Performance optimizations

Version	Date	Changes
1.1.0	TBD	Enhanced security features

⚖️ License

Proprietary - Internal Use Only

⚠️ **Important Notes:**

1. Always test in sandbox before production
2. Ensure proper backup before deployment
3. Configure security settings before going live
4. Monitor closely for first 48 hours post-deployment

✅ **Deployment Complete!** Your Dynamic Action Plans solution is now ready for use.