Schema Migration Deployment Guide

Overview

This guide provides instructions for deploying the schema changes where fields have been moved from ORDER forms to POTENTIAL_SITE forms.

Files Created/Modified

Core Files

1. `order-update.js` - Complete update system for orders and potential sites
2. `config.js` - Enhanced with new update forms and schemas
3. `migration-triggers.js` - Automated schema migration system
4. `trigger-setup.js` - Complete trigger management system
5. `deployment-script.js` - Deployment orchestration script

Deployment Process

Step 1: Pre-Deployment Testing

```
// Test deployment readiness
var testResult = testDeployment();
console.log(testResult);
```

Step 2: Quick Deployment (Recommended)

```
// Run complete deployment with safety checks
var deployResult = quickDeploy();
console.log(deployResult);
```

Step 3: Manual Deployment (If needed)

```
// Run full deployment process
var deployResult = deploySchemaChanges({
  forceDeployment: false,
  autoRollbackOnFailure: true
});
console.log(deployResult);
```

Step 4: Force Deployment (Emergency only)

```
// Force deployment bypassing validation
var deployResult = forceDeploy();
console.log(deployResult);
```

What the Deployment Does

1. Pre-Deployment Validation

- ☑ Verifies spreadsheet access
- ☑ Checks required sheets exist
- ☑ Validates configuration integrity
- ☑ Confirms necessary permissions

2. System Backup

- 📋 Creates backup copies of all relevant sheets
- 📋 Adds timestamp to backup names
- 📋 Creates deployment log sheet

3. Schema Migration

- 🔄 Moves specified fields from ORDERS to POTENTIAL_SITE_APPROVALS
- 🔄 Preserves existing data integrity
- 🔄 Updates form schemas

4. Trigger Setup

- ⚙ Creates form submission triggers
- ⚙ Sets up periodic migration checks
- ⚙ Enables automated validation

5. Post-Deployment Validation

- ☑ Verifies migration completed successfully
- ☑ Confirms triggers are active
- ☑ Tests update functions

6. Form Initialization

- 📝 Creates ORDER_UPDATES sheet
- 📝 Creates POTENTIAL_SITE_UPDATES sheet
- 📝 Sets up proper headers and formatting

7. Notifications

- 📧 Sends deployment status to administrators
- 📧 Includes detailed step-by-step results

Fields Being Migrated

From ORDERS Schema → To POTENTIAL_SITE_APPROVALS Schema:

1. **Start Building → Start Building**
2. **End Building → End Building**
3. **Project Address → Project Address**
4. **Estimated Quantity → Estimated Quantity**
5. **Delivery Timeline → Delivery Timeline**
6. **Custom Timeline → Custom Timeline**

New Update Forms

1. ORDER_UPDATE Form

- **Purpose**: Update existing orders with new data
- **Fields**: All current order fields except the migrated ones
- **Triggers**: Automatic processing on form submission

2. POTENTIAL_SITE_UPDATE_ENHANCED Form

- **Purpose**: Update potential sites with enhanced data
- **Fields**: All potential site fields plus the migrated fields
- **Triggers**: Automatic processing and cross-reference updates

Monitoring and Maintenance

Check Migration Status

```javascript
// Check if migration is needed
var migrationNeeded = isMigrationNeeded();
console.log('Migration needed:', migrationNeeded);
```

Manual Migration (If needed)

```javascript
// Run migration manually
var migrationResult = runSchemaMigration();
console.log(migrationResult);
```

Setup Triggers Manually

```javascript
// Setup triggers manually
var triggerResult = setupMigrationTriggers();
console.log(triggerResult);
```

View Active Triggers

```
// List all project triggers
var triggers = ScriptApp.getProjectTriggers();
triggers.forEach(trigger => {
  console.log(`Function: ${trigger.getHandlerFunction()}, Type:
${trigger.getEventType()}`);
});
```

Troubleshooting

If Deployment Fails

1. Check the error message in the deployment result
2. Verify permissions are sufficient
3. Ensure all required sheets exist
4. Try force deployment if safe to do so

If Triggers Don't Work

1. Check trigger permissions
2. Manually run `setupMigrationTriggers()`
3. Verify function names are correct

If Data is Missing

1. Check backup sheets for original data
2. Use `transferMovedFieldsData()` to re-migrate specific data
3. Contact system administrator

Emergency Rollback

```
// Attempt rollback (implementation dependent)
var rollbackResult = rollbackDeployment();
console.log(rollbackResult);
```

Post-Deployment Checklist

- [ ] Deployment completed successfully
- [ ] All triggers are active
- [ ] Update forms are accessible
- [ ] Data migration completed without loss
- [ ] Notifications sent to administrators
- [ ] Test order and potential site updates

- [ ] Verify cross-reference functionality
- [ ] Document any issues or warnings

Contact Information

For deployment issues or questions:

- **System Administrator**: As configured in `CONFIG.NOTIFICATION_CONFIG`
- **Business Head**: As configured in `CONFIG.NOTIFICATION_CONFIG`

Notes

- Always test deployment in a safe environment first
- Keep backup sheets until deployment is fully verified
- Monitor system for 24-48 hours after deployment
- Update documentation if any changes are made