Power Apps Write-Back Feature

Implementation Guide

**USCIS Office of the Chief Data Officer**

Data Engineering Team

*Version 1.0 | October 2025*

# Executive Summary

The Power Apps write-back feature enables users to create interactive applications that can both read from and write data back to various data sources including SharePoint lists, SQL databases, and Dataverse tables. This capability is essential for creating data validation workflows, approval processes, and interactive dashboards that complement our existing Databricks and Tableau infrastructure.

Unlike Tableau's write-back extensions which require additional licensing and configuration, Power Apps provides native write-back functionality through its connector architecture, making it an ideal solution for USCIS data operations requiring user input and data updates.

# Overview and Architecture

## What is Power Apps Write-Back?

Power Apps write-back refers to the capability of Power Apps applications to:

* **Read data from connected data sources**
* **Display data in interactive forms and galleries**
* **Capture user input through various controls**
* **Validate data before submission**
* **Write modified or new data back to the source system**
* **Trigger workflows and notifications upon data changes**

# Use Cases and Benefits

## Primary Use Cases at USCIS

**1. Data Validation Workflows**

* Quality assurance checklists for data pipelines
* Manual data corrections and annotations
* Exception handling for ETL processes

**2. Approval Processes**

* Data access request approvals
* Report publication sign-offs
* Change management workflows

**3. Configuration Management**

* Pipeline parameter updates
* Business rule modifications
* Threshold and alert configurations

# Prerequisites and Requirements

## Technical Prerequisites

| **Category** | **Requirements** |
| --- | --- |
| **User Environment** | * Microsoft 365 E3/E5 license or Power Apps per-user license * Modern web browser (Edge, Chrome, Firefox) * Access to Power Apps maker portal |
| **Data Sources** | * SharePoint: Site collection with appropriate lists * SQL: Database with tables and schemas * Dataverse: Environment with required entities |
| **Network** | * Access to Power Platform services (\*.powerapps.com) * On-premises data gateway (if needed) * Firewall rules for Power Apps traffic |

# Step-by-Step Implementation Guide

## Phase 1: Environment Setup

**Step 1: Access Power Apps Portal**

1. Navigate to https://make.powerapps.com
2. Select appropriate environment (e.g., "USCIS Production")
3. Verify you have Maker permissions

**Step 2: Create New Canvas App**

1. Click "Create" → "Canvas app"
2. Choose "Blank app"
3. Select format: Tablet (recommended for desktop use)
4. Name: "USCIS\_[Purpose]\_WriteBack"
5. Click "Create"

## Phase 2: Data Connection Setup

**SharePoint Connection**

1. Click "Data" in left navigation

2. Click "Add data"

3. Search for "SharePoint"

4. Choose "SharePoint" connector

5. Enter SharePoint site URL

6. Select your list(s)

7. Click "Connect"

# Example: Validation Checklist Application

## Scenario

Create a data quality validation checklist for Databricks pipeline outputs before they're consumed by Tableau dashboards.

## Implementation Steps

### Step 1: Create SharePoint List

Navigate to SharePoint site and create a new list called "Pipeline\_Validation\_Checklist" with the following columns:

| **Column Name** | **Type** | **Settings** |
| --- | --- | --- |
| PipelineName | Single line of text | Required |
| RunDate | Date and Time | Required, Default=Today |
| RecordCount | Number | Required, Min=0 |
| NullCheckPassed | Yes/No | Default=No |
| SchemaValidated | Yes/No | Default=No |
| ApprovalStatus | Choice | Pending/Approved/Rejected |

### Step 2: Implement Write-Back Logic

The following Power Apps formula implements the write-back functionality for the validation checklist:

// Submit button logic

Button\_Submit.OnSelect =

// Validate required fields

If(

IsBlank(Dropdown\_Pipeline.Selected.Value) ||

IsBlank(TextInput\_RecordCount.Text),

Notify("Please complete all required fields",

NotificationType.Error),

// Submit the checklist

Patch(

Pipeline\_Validation\_Checklist,

Defaults(Pipeline\_Validation\_Checklist),

{

PipelineName: Dropdown\_Pipeline.Selected.Value,

RunDate: Now(),

RecordCount: Value(TextInput\_RecordCount.Text),

NullCheckPassed: Toggle\_NullCheck.Value,

SchemaValidated: Toggle\_SchemaCheck.Value

}

);

Notify("Checklist submitted successfully",

NotificationType.Success)

)

# Best Practices and Governance

## Naming Conventions

Establish consistent naming patterns across your organization:

* **Apps:** USCIS\_[Department]\_[Function]\_[Version]
* **Controls:** [Type]\_[Purpose]\_[Screen]
* **Variables:** var[Scope][Purpose]
* **Collections:** col[DataType]

## Performance Optimization

Follow these guidelines to ensure optimal app performance:

* Use delegation-friendly filters for large datasets
* Cache frequently used data in collections
* Minimize API calls by batching updates
* Implement pagination for galleries with many items
* Use concurrent functions when possible

# Troubleshooting Guide

## Common Issues and Solutions

| **Issue** | **Cause** | **Solution** |
| --- | --- | --- |
| Delegation warnings | Non-delegable functions | Use delegable alternatives like StartsWith |
| Write-back failures | Permission issues | Verify user has Edit permissions |
| Slow performance | Large datasets | Implement pagination |
| Connection errors | Network/gateway issues | Check gateway status |

# Support and Resources

## Internal Resources

* USCIS Power Platform Center of Excellence
* Data Engineering Team Channel on Microsoft Teams
* Service Desk: Submit tickets for Power Apps support

## External Resources

* Microsoft Power Apps Documentation
* Power Apps Community Forums
* Microsoft Learn Training Modules

*This document is maintained by the USCIS Office of the Chief Data Officer*

*For updates or corrections, please contact the Data Engineering Team*