Code Review for  
ZNA SharePoint Custom Migrations

Robyn, Steve, and Troy are implementing a code review process to ensure the standards and patterns we have established are being appropriately followed. To ensure consistency, we have developed three checklists. The first covers Power Automate flows. The second covers Power Apps. The third covers out-of-the-box forms.

When a developer has completed the initial development for a customization, the customization tracker on the Migration Communication Center (MCC) should be updated to indicate the customization is ready for its code review. Daily, Robyn, Steve, and/or Troy will check the customizations awaiting review, assign one of us to be the primary reviewer, and check the developed customization against the appropriate checklist. Comments regarding the code review will be kept separate from the MCC and communicated directly to the assigned developer via either Teams chat or email. Once the code review is passed, a date will be recorded in the customization tracker to indicate this and testing can commence.

This document will be a living, ongoing work in progress. It may be updated periodically. Before indicating that code is ready for review, developers should be validating their customization against the appropriate checklist first.

# Power Automate Flow Checklist

| Topic | Expectation |
| --- | --- |
| Flow Name | Ensure that the flow name starts with the site URL and then matches the name of the original SharePoint Designer workflow. (i.e. 00412- Sys Export Notification) |
| Trigger Condition | Ensure the trigger condition is used as appropriate. The trigger condition should be used when the first action of the flow is a condition. |
| Email Importance | By default, the Importance is left blank which sends out emails with the Low importance flag. On each email that is sent, unless other indicated in the requirements, the Importance should be set to Normal. |
| Action Names | Each action should be renamed from its default to a name that describes the purpose for the action. For example, instead of the default “Send an email (V2)” action name, use a more descriptive name such as “Send the Approver an email asking to review this item” |
| Print to PDF Pattern | If implementing Print to PDF functionality, ensure the [documented pattern on GitHub](https://github.dxc.com/SharePointMigrations/modernize-sharepoint-applications/blob/main/azure-logic-apps-power-automate/print-to-pdf/README.md) has been followed. |
| Updating a list item | Do not use an “Update Item” task but implement the SystemUpdate pattern to update list items without causing flows to trigger or the last modified date/user to be updated. Ensure the [documented pattern on GitHub](https://github.dxc.com/SharePointMigrations/modernize-sharepoint-applications/blob/main/azure-logic-apps-power-automate/system-update/README.md) has been followed. |
| Multi-Select Values in Emails | Apply to Each should be avoided due to the To field or a field in the body having multiple values. Those fields need to be converted to string variables and that variable used in the email instead. Ensure the logic [found here](https://dxcportal.sharepoint.com/:p:/r/sites/ZurichNACustomSPOMigration/Shared%20Documents/General/Development%20Templates%20and%20Standards/Power%20Automate/Converting%20a%20Multi-people%20field%20to%20a%20string%20for%20emails.pptx?d=w60baf3556cb9447a9a41bb26d8c1b818&csf=1&web=1&e=Hb1Aos) has been followed. |
| Date/Time Values | Be aware of the time zone and format of all date/time values. Do not show UTC dates/times to the user unless specifically requested in the requirements. Instead, convert all end-user facing dates/times to the US Central Time Zone before displaying, emailing, etc. Also ensure that end-user facing dates/times are formatted correctly (e.g. 12/31/2099 vs. 2099-12-31T01:23:45Z) before displaying them to the user. |
| Trigger Conditions | Ensure there is a comment in the trigger action if you use a trigger condition. Use a trigger condition if the flow has a condition at the beginning where the No path is then empty. |

# Power App Checklist

| Topic | Expectation |
| --- | --- |
| Variable Naming | Variables should be named using camelCase.  The prefix ‘loc’ should be added to context variables. The prefix ‘glo’ should be added to global variables. |
| Control Naming | Controls should be named using camelCase.  Control names should start with a three-letter prefix based on the type of control. Control names should then contain a name describing what the control represents. For example, **lblTitle** would be a label for the Title field.  Use the following prefixes based on the type of control:   * att Attachments * btn Button * chk CheckBox * cmb ComboBox * crd DataCard * dtp DateTimePicker * frm Form * gal Gallery * icn Icon * lbl Label * lst ListBox * rct Rectangle * rdo RadioButton * rtx RichTextBox, Rich Text Editor * scr Screen * txt TextBox |
| Style Sheet | The **gloStyleSheet** variable should contain the following object. This variable should be set in the app start formula.  {  textLight: ColorValue("#FFFFFF"),  textDark: ColorValue("#000000"),  paleWhite: ColorValue("#F4F4F4"),  titleFont: Font.'Open Sans',  titleSize: 21,  titleColor: ColorValue("#003399"),  titleEmph: FontWeight.Bold,  fieldFont: Font.'Open Sans',  fieldSize: 13,  fieldColor: ColorValue("#000000"),  fieldEmph: FontWeight.Bold,  inputFont: Font.'Open Sans',  inputSize: 13,  inputHeight: 40,  inputColor: ColorValue("#000000"),  inputEmph: FontWeight.Normal,  backgroundColor: ColorValue("#FFFFFF"),  tabFont: Font.'Open Sans',  tabSize: 16,  tabColor: ColorValue("#FFFFFF"),  tabBackgroundColor: ColorValue("#003399"),  tabDisabledBackgroundColor: ColorValue("#F4F4F4"),  tabEmph: FontWeight.Bold,  requiredAstColor: ColorValue("#FF0000"),  requiredAstSize: 15  }  The title values should be used in the properties of the application title’s label. The field values should be used in the properties of the form field labels. The input values should be used in the properties of the form field input controls. The tab values should be used in the properties of the tabs at the top of the screen, if applicable. The requested ast values should be used for the asterisks marking fields as required. Do not hard code colors and fonts. Use references to this global style sheet instead. |
| Using tabs vs. screens | Tabs should be used when separating portions of a form into distinct groupings. For example, if a form has several fields about a claimant’s information, and also several fields about the claim being made, it might make sense to separate these into two tabbed sections. One for the personal info, and one for the claim information, in order to make the form easier to use for the end user. Tabs should operate by setting a global or context (preferred, if possible) variable to the currently selected tab, and all form fields refer to that variable to decide if they should be visible or not.  Screens are more appropriate for standalone canvas apps as opposed to list form canvas apps. Screens are used when there are several stages to an application. For example, on the first screen a user could select an item from a gallery or click a button to create a new item. Both of these actions would transition to a second screen where the form resides. Due to the way screen transitioning works, data can be passed as input to the new screen. This is a good method for passing selected item information and whether the form is for display, new, or editing. |
| Environment | For list form canvas apps on non-confidential sites with no data sources outside SharePoint, the app should be within the Personal Productivity environment. For all other types of apps or scenarios, the app should be within the UAT environment. |
| Responsive Design | The size of the Power App should be set to Landscape and Large. Custom sizes or Portrait should not be used.  All controls should be designed with a responsive layout in mind. **App.Width** and **App.Height** will provide the current value of the width and height of the app respectively. Every control in the app will need to be made to consider these.  For example,  A bar that needs to span the whole width of the app would have its **Width** property set to **App.Width**, its **Height** property set to a fixed numeric value, its **X** property set to 0, and its **Y** property set to a fixed numeric value.  A form that would sit under this bar and stretch to the bottom of the screen would have its **Width** property set to **App.Width**, its **Height** property set to **App.Height – bar.Height**, its **X** property set to 0, and its **Y** property set to **bar.Y + bar.Height**.  All controls can be organized in this way using a bit of math so that they will respond to changes in the app’s width and height on different devices and screen sizes. Note that if statements can be used in **Width**, **Height**, **X**, and **Y** properties to vary layouts based on screen size. For example, a control could be set to fill the entire width of the app if the screen size was less than 800 pixels wide. Or, it could be set to only fill half of the width of the app (to allow for a two column layout) if the screen size was more than 800 pixels wide. |
| Stacking Form Labels and Controls | All controls should be setup as “stacked” beneath their corresponding label. Controls should not be placed to the right of their corresponding labels.  The spacing between the field label & the field control can be set as **5**. Meaning, that the formula for the field control can be set to the field label’s **Y** property, plus the field label’s **Height** property, plus 5. Do not hard code the number 5. Set up a global variable to contain that value (e.g. gloSpacingBetweenFieldLabelAndValue), initialize that variable to 5, then use that variable in formulas.  For descriptions that are added to fields, ensure that the description label is **underneath** the error message label. This mimics the way SharePoint Online works in its out-of-the-box list forms. |
| Hard Coded Constants, Literals | In formulas, one may wish to use either numeric or string constants. Whenever practical, please extract these constants into global or local variables as appropriate and use those variables in the formulas instead of the constants; especially when the constant is used in more than one formula. That way, the constant can be easily updated later plus there is more description in the formulas about what the number represents. (e.g. “gloSpacingBetweenFieldLabelAndValue” versus 5) |
| In-list Forms | *The following applies only for in-list Power App forms.*  The form should not have a header with the title of the list.  The form should not have any created, created by, last modified, or last modified by information.  The form should not have any save or cancel buttons. Instead, those are supplied automatically by SharePoint Online.  Should the form need to access the currently selected list item immediately upon opening the form, do not leverage SharePointIntegration.Selected. Instead, perform a lookup based on SharePointIntegration.SelectedListItemID. See <https://github.dxc.com/SharePointMigrations/modernize-sharepoint-applications/blob/main/power-apps/using-current-item-when-in-list-form-opens/README.md> for details. |

# Out-of-the-box Form Checklist

| Topic | Expectation |
| --- | --- |
| Cascading Dropdowns | If converting cascading dropdowns to managed metadata fields to enable an out-of-the-box form to be used, ensure the documented pattern on GitHub has been followed. |
| Indentation | All JSON should be indented using four spaces. |
| Spacing | All JSON key/value pairs should have a single space after the colon and no space after the value (between the value and a comma or the end of the line). For example,  "key": "value",  All JSON array values should have no space after the value (between the value and a comma or the end of the line). For example,  "value", |
| Quotation Marks | All JSON should use double quotation marks for all string literals. |
| Lines | All JSON key/value pairs should exist on their own lines. |
| Object Key Casing | All JSON keys should be either camelCase (preferred) or snake-case. |
| Debug Mode | Where available, the debugMode key can be set to true in certain areas to enable additional logging and debugging features. While this is very helpful and useful for development purposes, it should be omitted or set to false before finalizing the development so that it does not negatively impact the production use of the customization. |
| Body Sections | When creating one or more sections in the body of the form, the fields listed in each section must be fields that exist on the list. No additional fields not present on the list should be listed in any section. |
| Object Formatting | All empty JSON objects should contain one opening curly brace followed by a single space followed by one closing curly brace. For example,  { }  All JSON objects with at least one key/value pair should contain one opening curly brace on its own line, all key/value pairs indented on their own lines (one per line), and one closing curly brace on its own line. This applies for any number of key/value pairs in an object. Do not condense single key/value pair objects into a single line. For example,  {  "key1": "value1",  "key2": 123,  "key3": null  }  If a JSON object is a value of a key/value pair, the opening curly brace should be on the same line as the key. For example,  {  "keyWithAnObjectValue": {  "key1": "value1"  }  }  If JSON objects are members of JSON arrays, they should still follow this pattern, even if there is only a single key/value pair in the object. Each object should be on its own lines. For example,  {  "keyWithAnArrayOfObjectsValue": [  {  "key": "value"  },  {  "key": "value"  }  ]  } |
| Array Formatting | All empty JSON arrays should contain one opening square bracket followed by a single space followed by one closing square bracket. For example,  [ ]  All JSON arrays with at least one value should contain one opening square bracket on its own line, all values indented on their own lines (one per line), and one closing square bracket on its own line. This applies for any number of values in an array. Do not condense single value arrays into a single line. For example,  [  "value1",  123,  null  ]  If a JSON array is a value of a key/value pair, the opening square bracket should be on the same line as the key. For example,  {  "keyWithAnArrayValue": [  "value",  "value"  ]  }  If JSON arrays are values of JSON arrays, they should still follow this pattern, even if there is only a single value in the array. Each array should be on its own lines. For example,  [  [  "value1",  123  ],  [  "value2",  456  ]  ] |