Flow Naming Conventions

Meta-Flow Naming

1. A Flow name shall always start by the name of the **Domain from which it originates**, followed by an underscore.
   In most cases, for Flows, the Domain is equivalent to the Object that it is hosted on.
   As per structural conventions, cross-object Flows should be avoided and reliance on Events to synchronize flows that do cross-object operations should be used.

> In `Account_BeforeSave_SetClientNumber`, the Domain is Account, as this is where the automation is started. It could also be something like `AccountManagement` , if the Account Management team owned the process for example.

2. The Domain of the shall be followed by a code indicating the type of the Flow, respecting the cases as follows:
   1. If the flow is a Screen Flow, the code shall be `SCR`.
   2. If the flow is a SubFlow, the code shall be `SFL`.
   3. If the flow is specifically designed to be a scheduled flow that runs on a schedule, the code shall be `SCH`.
   4. If the flow is a Record Triggered flow, the code shall instead indicate the contexts in where the Record Triggered Flow executes.
      In addition, the flow name shall contain the context of execution, meaning either `Before` or `After`, followed by either `Create`, `Update` or `Delete`.
   5. If the flow is an Event Triggered flow, the code shall be `EVT` instead.
   6. If the flow is specifically designed to be a Record Triggered flow that ONLY handles email sends, the code shall be `EML` instead.

> In `Account_AftercreateAftersave_StatusUpdateActions`, you identify that it is Record-Triggered, execute both on creation and update, in the After Context,

> and that it carries out actions related to when the entry criteria (the status has changed) are met.

> In the case of `Invoice_SCR_CheckTaxExemption`, you know that it is a Screen Flow, executing from the Invoice Lightning Page, that handles Tax Exemption related matters.

3. A Flow name shall further be named after the action being carried out in the most precise manner possible. For Record Triggered Flows, this is limited to what triggers it. See example table for details.
4. A Flow Description should always indicate what the Flow requires to run, what the entry criteria are, what it does functionally, and what it outputs.

Type	Name	Description
Screen Flow	Quote_SCR_addQuoteLines	[Entry = None] A Screen flow that is used to override the Quote Lines addition page. Provides function related to Discount calculation based on `Discounts_cmtd`.
Scheduled Flow	Contact_SCH_SendBirthdayEmails	[Entry = None] A Scheduled flow that runs daily, checks if a contact is due a Birthday email, and sends it using the template marked `Marketing_Birthday`
Before Update Flow, on Account	Account_BeforeUpdate_SetTaxInformation	[Entry = IsChanged(ShippingCountry)] Changes the tax information, rate, and required elements based on the new country.
After Update Flow, on Account	Account_AfterUpdate_NewBillingInfo	[Entry = IsChanged(ShippingCountry)] Fetches related future invoices and updates their billing country and billing information. Also sends a notification to Sales Support to ensure country change is legitimate.

Event-Triggered Flow, creating Invoices, which triggers when a Sales Finished event gets fired	Invoice_EVT_SalesFinished	Creates an Invoice and notifies Invoicing about the new invoice to validate based on Sales information
Record-triggered Email-sending Flow, on Account.	Account_EML_AfterUpdate	[Entry = None] Handles email notifications from Account based on record changes.

Flow Elements

DMLs

1. Any Query shall always start by `Get` for any Objects, followed by an underscore, or `Fetch` for CMTD or Settings.
2. Any Update shall always start by `Update` followed by an underscore. If it Updates a Collection, it shall also be prefixed by `List` after the aforementioned underscore.
3. Any Create shall always start by `Create` followed by an underscore. If it Creates a Collection, it shall also be prefixed by `List` after the aforementioned underscore.
4. Any Delete shall always start by `Del` followed by an underscore. If it Deletes a Collection, it shall also be prefixed by `List` after the aforementioned underscore.

Type	Name	Description
Get accounts matching active = true	Get_ActiveAccounts	Fetches all accounts where IsActive = True
Update Modified Contacts	Update_ListModifiedContacts	Commits all changes from previous assignments to the database
Creates an account configured during a Screen Flow in a variable called `var_thisAccount`	Create_ThisAccount	Commits the Account to the database based on previous assignments.

Interactions

1. Any Screen ***SHALL*** always start by `S`, followed by a number corresponding to the current number of Screens in the current Flow plus 1, followed by an underscore.
2. Any Action ***SHALL*** always start by `ACT`, followed by an underscore. The Action Name ***SHOULD*** furthermore indicate what the action carries out.
   - Any APEX Action ***SHALL*** always start by `APEX` instead, followed by an underscore, followed by a shorthand of the outcome expected. Properly named APEX functions

should be usable as-is for naming.
    - Any Subflow ***SHALL*** always start by `SUB` instead, followed by an underscore, followed by the code of the Flow triggered (FL01 for example), followed by an underscore, followed by a shorthand of the outcome expected.
3. Any Email Alert ***SHALL*** always start by `EA`, followed by an underscore, followed by the code of the Email Template getting sent, an underscore, and a shorthand of what email should be sent.

Type	Name	Description
Screen within a Flow	Label: Select Price Book Entries Name: S01_SelectPBEs	Allows selection of which products will be added to the quote, based on pricebookentries fetched.
Screen that handles errors based on a DML within a Flow	SERR01_GET_PBE	Happens if the GET on Pricebook Entries fails. Probably related to Permissions.
Text element in the first screen of the flow	S01_T01	*Fill with actual Text from the Text element - there is no description field*
DataTable in the first screen of the flow	S01_LWCTable_Products	*May be inapplicable as the LWCs may not offer a Description field.*

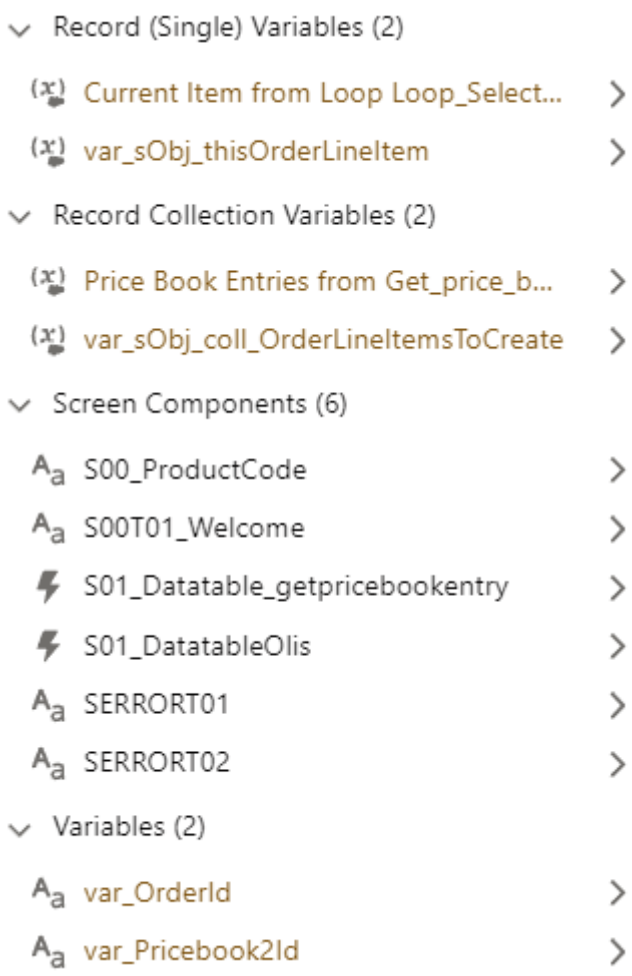
Example of a Screen containing a Text element

Screen Elements

1. Any variable ***SHALL*** always start by `var` followed by an underscore.
    - Any variable that stores a Collection ***SHALL*** always in addition start by `coll` followed by an underscore.
    - Any variable that stores a Record ***SHALL*** always in addition start by `sObj` followed by an underscore.
    - Any other variable type ***SHALL*** always in addition start by an indicator of the variable type, followed by an underscore.
2. Any formula ***SHALL*** always start by `form` followed by an underscore, followed by the data type returned, and an underscore.
3. Any choice ***SHALL*** always start by `ch` followed by an underscore. The Choice name should reflect the outcome of the choice.

Type	Name	Description
Formula to get the total number of Products sold	formula_ProductDiscountWeighted	Weights the discount by product type and calculates actual final discount. Catches null values for discounts or prices and returns 0.

Variable to store the recordId	recordId	Stores the record Id that starts the flow. *Exempt from normal conventions because legacy Salesforce behavior.* *Note: This var name is CASE SENSITIVE.*
Record that we create from calculated values in the Flow in a Loop, before storing it in a collection variable to create them all	sObj_This_OpportunityProduct	The Opportunity Product the values of which we calculate.



Screenshot from the Manager, with examples of Variables and Screen elements

Logics

1. Any Decision ***SHALL*** start by `DEC` if the decision is an open choice, or `CHECK` if it is a logical terminator, followed by an underscore. The Action Name ***SHOULD*** furthermore be prefixed by `Is`, `Can`, or another adverb indicating the nature of the decision, as well as a short description of what is checked.
   - Any Decision Outcome ***SHALL*** start with the Decision Name without any Prefixes, followed by an underscore, followed by the Outcome.
   - The Default Outcome ***SHOULD*** be used for error handling and relabeled `ERROR` where applicable - you *can* relabel the default outcome!

2. Any Assignment ***SHALL*** always start with `SET`, `ASSIGN`, `STORE`, `REMOVE` or `CALC` (depending on the type of the assignation being done) followed by an underscore.
   - `SET` ***SHOULD*** be used for variable updates, mainly for Object variables, where the variable existed before.
   - `ASSIGN` ***SHOULD*** be used for variable initialization, or updates on Non-Object variables.
   - `STORE` ***SHOULD*** be used for adding elements to Collections.
   - `REMOVE` ***SHOULD*** be used for removing elements from Collections.
   - `CALC` ***SHOULD*** be used for any mathematical assignment or complex collection manipulation.
3. Any Loop ***SHALL*** always start with `LOOP`, followed by an underscore, followed by the description of what is being iterated over. This can vary from the Collection name.

Type	Name	Description
Assignment to set the sObj_This_OpportunityProduct record values	SET_OppProdValues	Sets the OppProd based on calculated discounts and quantities.
Assignment to store the Opportunity Product for later creation in a collection variable	Name: STORE_ThisOppProd Assignment: {!sObj_coll_OppProdtoCreate} Add {!sObj_This_OpportunityProduct}	Adds the calculated Opp Prod lines to the collvar to create.
DML to create multiple records store in a collection sObj variable	CREATE_OppProds	Creates the configured OppProd.
Decision to check selected elements for processing	Decision: CHECK_PBESelected Outcome one: CHECK_PBESelected_Yes Outcome two: CHECK_PBESelected_No Default Outcome: Catastrophic Failure	Check if at least one row was selected. Otherwise terminates to an error screen.
Decision to sort elements based on criteria	Decision: DEC_SortOverrides Outcome one: SortOverrides_Fields Outcome two: SortOverrides_Values Outcome three: SortOverrides_Full Default Outcome: Catastrophic Failure	Based on user selection, check if we need to override information within the records, and which information needs to be overridden.
Email Alert sent from Flow informing user of Invoice reception	EA01_EI10_InvoiceReceived	Sends template EI10 with details of the Invoice to pay

---

Revision #3
Created 2023-08-10 13:50:17 UTC by Windyo
Updated 2023-08-28 07:11:25 UTC by Windyo