



## SALESFORCE **FLOW** BEST PRACTICES

### **WHITE PAPER**

Practical standards for naming conventions, design patterns, error handling, and organizational strategies that ensure your automation remains sustainable as your team and requirements grow.

*Last Revised June 2025*

**[cldpartners.com](https://cldpartners.com)**

## Overview

With the evolution Salesforce Flows have had in the past several years, as architects and admins it might be challenging to figure out the best way to incorporate them into your implementations.

This document is intended to address how to use flows rather than when to use them. It provides guidance for new Salesforce implementations and can also be applied by established Salesforce organizations looking to apply standard Flow conventions.

This document provides guidelines in terms of number of Flows per object, general naming conventions, error tracking, and more. It's well suited for organizations with larger admin teams or teams that employ a sophisticated approach to IT management.



"If you have multiple people writing Flows in the org, we find these best practices and guidelines really help handoff from one team to another, or from Dev to QA. Everybody involved knows *'how to read'* the Flow."

-- Cecilia Chiderski, Solution Architect, CLD Partners

This document focuses on two primary areas when working with Flows:

- Process-related (flow design, Org standards, naming conventions)
- Technical-related advice (such as don't hard code IDs, when to use subflows, error handling, and more)

## When should you use flows?

For help on deciding when to use a Flow vs Apex, Salesforce has great documentation on the [Salesforce Architects page](#) or you can talk with our experts! We're here to help!

## Table of Contents

Overview .....	2
Flow Design Strategy .....	4
One Flow per object per event, or multiple Flows? .....	4
When to create a new flow? .....	4
General Design Guidelines .....	6
Naming Conventions .....	8
Naming Flows .....	9
Naming Flow Elements .....	10
Using Subflows .....	12
Flow Context .....	13
Testing Flows .....	14
Error Handling .....	16
Checklist of 20-ish Flow Tips .....	17

# Flow Design Strategy

## One Flow Per Object Per Event, or Multiple Flows?

Early on, Salesforce advised to have one Flow per object per event when it comes to Record-Triggered Flows. After retiring workflow rules and replacing them with Flows, it became very apparent that “one Flow per object per event” was not compatible with the guideline of defining good entry criteria to minimize execution.

“One Flow per object per event” is only achievable these days in small orgs with minimal automation, or if the org is Essentials or Professional. In this last case, you can have up to 5 *active* Flows per type. ([Read more about Salesforce Flow limits.](#))

On Enterprise editions, you should plan for scalability, and recognize that one Flow per object per event might not be reasonable. This is why a Flow design strategy is key, particularly when working with record-triggered Flows, to ensure sustainability, scalability, and simplicity for future troubleshooting.

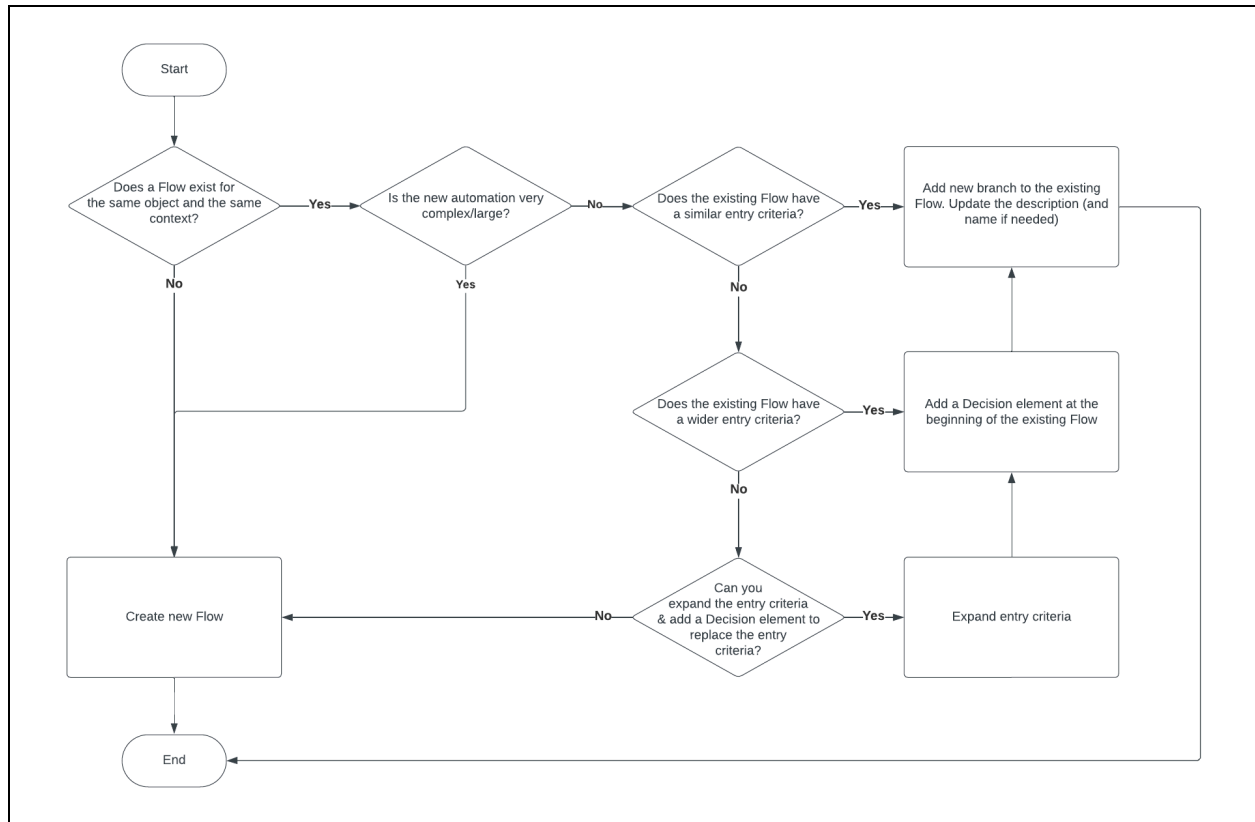
It's worth noting that the discussion about the number of Flows per object is relevant only on *Record-Triggered-Flows*. Other Flow types are typically standalone unless they are expanding and extending or modifying an existing one.

## When to Create a New Flow?

There is a technical limitation to the concept of one Record-Triggered Flow per object: a Record-Triggered-Flow can run in a before context OR in an After context, not both. At the same time, they can only run on Create, Update, Create or Update, or Delete. This means that at least, you need three Flows:

- 1) Before create, before update
- 2) After create, after update
- 3) Before delete

This structure should be the guiding principle. However, this will evolve as the business processes that Flow solves grows. Deciding to create a new Flow or reuse an existing one is challenging and so far there is no magic answer. As a starting point, use the following decision tree to get started.



*Flow Creation Decision Tree*

The diagram above should be used as a starting point. It's important to note business analysis and Flow design are key in the decision process. Keep these best practices in mind while you design the perfect flow.

- 1) If the Flow that you are adding is a very complex and standalone process → call a Subflow if possible or create a new Flow.
- 2) If the existing Flow is already in the Org - consider if it is properly written. If not, create a new Flow following our recommended standards in this doc, OR talk with your team about revising the existing Flow.
- 3) If you need to change the Flow name for an existing Flow to expand its scope, be sure to share with your team. It might have training or documentation implications.
- 4) If you are not sure, contact the lead Salesforce admin to discuss your use case.

# General Design Guidelines

1. [Design your Flow before you start building](#). Have a clear understanding of the desired outcome, the information you need, where it is coming from, and a general idea of the actions to perform.
2. Always fill in the Flow description field. Clearly define the problem that the Flow is solving, the objects that are somehow involved, and if relevant, how it is called (e.g. a button on a layout).
  - o If possible, include a brief description of the related business process. Always ask yourself “if I’m seeing this Flow for the first time, what would be relevant to know?”. Whoever is coming after you will be thankful!
  - o If the Flows have descriptions, a quick and easy way to see them all is [creating a Flow list view](#), comprising the Flow name and description and applying any relevant filter.

*A simple Flow list view makes it easy to see Flow descriptions.*

- o On the Flow elements that allow for Description, always provide a brief explanation of what the element is doing. Not only it helps with documentation, but is also supports your Agentforce Flow Actions.
3. Use Fast Field Update any time you can. This is the equivalent to a “before” context in Apex, and the same rules apply. You can use Fast Field Update if:
    - o The Flow is updating only the record that triggered the Flow.
    - o The Flow doesn’t contain any Create Records, Action, Delete Records, or Subflow elements.
  4. Same as Apex, Flows have governor limits. Limit the number of queries and DML operations performed within the Flow to avoid hitting them, and never perform a DML inside of a loop.
  5. To avoid a mixed DML error during the execution of the Flow, if you are inserting/updating setup and non-setup objects in the same transaction, add an Asynchronous path to the flow. Then, perform the DML of the setup object

on the "Immediately" branch, and the DML for the non-setup object in the Asynchronous branch.

6. **Always** use Assignment and not Update on a Fast Field Updates Flow.
7. **Never** use hard-coded IDs. If you need an ID in a Flow, use a Get Records element to retrieve it instead.
  - This includes getting the record type ID: run Get Records on the Record Type object.
  - If you see hard-coded IDs in a Flow, talk to the customer about it. Sooner or later it is going to cause issues.
8. Refine when the Flow runs using entry criteria. As the number of Flows grows, this reduces the number of times a Flow is called, and leads to performance improvements during Flow execution.
9. Include a bypass (circuit breaker) in all your record triggered Flows to disable them in bulk if needed. This is especially useful during a data migration or sandbox seeding.
  - If the org already has the CLD trigger framework in place, add a field on the existing custom setting "Trigger Control Settings" called "Deactivate Flows". Then on every Flow include the custom setting as part of your entry criteria. Two options to do this:
    - Add the custom setting in a formula, as part of the Flow entry criteria. For example, the following Flow on the Contact object will only run if the Custom Setting field to "Disable Flows" is unchecked, and the Is Resource field is checked.

**Set Entry Conditions**

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements

Formula Evaluates to True

Formula

Insert a resource... All Functions Insert a function... Select an Operator...

AND(NOT({!\$Setup.Trigger\_Control\_Settings\_\_c.Disable\_Flows\_\_c}), {\$Record.pse\_\_Is\_Resource\_\_c})

- Add a decision node at the very beginning of the Flow that checks for the custom setting value.

- If the org doesn't have the CLD trigger framework in place, use your best judgment to determine where to include this circuit breaker functionality. It can be a custom setting, custom metadata, or a custom permission.
  - Custom settings and custom permissions are the preferred approach because they allow for easily narrowing down the users to which the functionality applies.
- 10. Check for null or empty results in Decision elements before acting on a set of records.
- 11. Control the order of execution: Navigate to the Flow Trigger Explorer by clicking the Flow Trigger Explorer button on the Flows page. By object and context, define the order of execution of the Flows.
- 12. Be careful with complex formula variables. Per the [Record-Triggered Automation Decision Guide](#), "Flow's formula engine sporadically exhibits poor performance when resolving extremely complex formulas. This issue is exacerbated in batch use cases because formulas are currently both compiled and resolved serially during runtime".
- 13. Reactivity in Flows is supported with API version 57.0 or later. You can easily change the API version on an old Flow by saving it as a new version and editing the API on the Flow properties.
- 14. Use a Flow Action Button or a Screen Action on Screen Flows to trigger actions from a Screen Flow. It's a great way to enhance user experience!
- 15. Enhance the Screen Flow UI with pictures (static resources), using icons on choice resources, adjusting the width of the components, or using HTML markups.
- 16. Always check with the team or end customer to see if they have any Flow guidelines that should be followed. Don't be afraid to ask challenging questions if you read something that seems counterintuitive or unwise.

## Naming Conventions

Ideally, your entire org should follow a consistent nomenclature. Names should be concise and immediately give an insight into the Flow's functionality. *(Note: the following naming conventions are referenced in this section.)*



## camelCase

- Capital letters can only appear at the start of the second word (like someVariable).
- No dots, underscores, numbers, dashes, or any other special characters are allowed within the word

## PascalCase

- A compound word that uses capital letters to differentiate words. Includes the first letter in the first word.

## Naming Flows

Flow Type	Name Guideline	Example
Record Triggered Flow	{Object Name}-{AbbreviatedEvent}-{Action in Plain Words}  <i>Note: If relevant for the Org, prefix the record triggered Flows with "PSA" or "FFA" for the Certinia managed packaged objects, "PSQ" for PSQuote managed packaged objects, etc.</i>	Opportunity-AI-Create Projects, Project-AI-Create Milestones, Contact-BI-Set Default Fields
Autolaunched Flow (no-subflow)	{Object Name}-{Action in Plain Words}	Opportunity-Check On Close Won, Milestone-Validate Billing Fields
Subflow	{Object Name}-SFL-{Action in Plain Words}	Resource Request-SFL-Populate Skill Requests
Screen (all)*	{Object Name}-SCR-{Action in Plain Words}	Resource Request-SCR-Create New RR, Opportunity-SCR-Initiate Change Order
Scheduled	{Object Name}-SCH- {Action in Plain Words}	Project-SCH-Create Monthly Milestones
Platform Event	{Platform Event}-EVT-{Action in Plain Words}	Integration Log-EVT-Send Notification

Flow Type	Name Guideline	Example
Orchestration*	{Object Name}-{AbbreviatedEvent**}-ORCH-{Action in Plain Words}  <i>Note: AbbreviatedEvent does not apply to Autolaunched Flow Approval Processes or Autolaunched Orchestrations.</i>	Resource Request-AIAU-ORCH-Staffing and Interview Process, Quote-ORCH-Approval Process
Evaluation	{Object Name}-EVAL-{Action in Plain Words}	Quote-EVAL-Check Level 1 Approvals
Template-Trig gered Prompt Flow*	{Object Name}-PRO-{Action in Plain Words}	Opportunity-PRO-Get All Open Opportunities for Account

(\*) Screen Flows, non-Approval Orchestrations and Template-Triggered Prompt Flows are more versatile because they might not be related to a particular object. In these cases, use a reasonable name that clearly identifies what the Flow is doing.

(\*\*) Abbreviated Event can be as follows:

- Before Insert → BI
- Before Update → BU
- After Insert → AI
- After Update → AU
- Before Delete → BD

*Note: Some organizations prefer to call the “insert” event “create” instead. In this case, replace “I” by “C”.*

## Naming Flow Elements

We referenced the [SFXD wiki](#) on naming conventions as a baseline for creating our conventions. Our list represents a simplified version designed for ease of use.

When it comes to Flow elements, it is important to have a naming convention so the screen is easier to navigate. However, it is critical to follow a pattern for the API name to facilitate the development and maintenance of the Flow.

Component Type	API Guideline	API Example	Name Example
DML - Query	Start with <i>Get</i> for any Objects, or <i>Fetch</i> for	<a href="#">Get</a> _Projects, <a href="#">Fetch</a> _DeactivateFlows	Get Projects, Fetch Deactivate Flows

Component Type	API Guideline	API Example	Name Example
	CMTD or custom settings		
DML - Update	Start with <i>Update</i> . If is updating a list, use <i>UpdateList</i>	<a href="#">Update</a> _ActiveAssignment, <a href="#">UpdateList</a> _ActiveAssignments	Update Active Assignment, UpdateList Active Assignments
DML - Insert	Start with <i>Insert</i> . If is inserting a list, use <i>InsertList</i>	<a href="#">Insert</a> _Milestone, <a href="#">InsertList</a> _Milestones	Insert Milestone, InsertList Milestones
DML - Delete	Start with <i>Del</i> . If is deleting a list, use <i>DelList</i>	<a href="#">Del</a> _ResourceRequest, <a href="#">DelList</a> _ResourceRequests	Del Resource Request, DelList Resource Requests
Action - Apex	Start with <i>APEX</i>	APEX_CreateProjects	APEX Create Projects
Action - Subflow	Start with <i>SUB</i>	<a href="#">SUB</a> _AddSkillsToRRs	SUBFLOW Add Skills To RRs
Action - Email Alert	Start with EMAIL	<a href="#">EMAIL</a> _NewAssignmentToAssignee	EMAIL New Assignment To Assignee
Screen	Start with SC01, +1 for each new screen	<a href="#">SC01</a> _CreateResourceRequests	Create Resource Requests
Screen Element	Start with the screen number that it belongs to	SC01_NumberOfMonths	Number Of Months
Resource (variable, constant, formula, text template)	Start with the variable type, and use camelCase. Optionally, start with <i>var</i> , and use camelCase.	num_numberOfMilestones, var_numberOfMilestones, txt_errorMessageTemplate	Number Of Milestones, Error Message Template

Component Type	API Guideline	API Example	Name Example
Decision Element	Start with <i>Is</i> , <i>Check</i> or <i>Can</i> , or similar	<i>Is</i> ProjectBillable, <i>Check</i> RecordType, <i>Can</i> ProjectBeClosed	Is Project Billable?, Check Record Type, Can Project Be Closed?
Decision Outcome	Start with the Decision Element and add <i>_Yes</i> , <i>_No</i> , or <i>_{PascalCase}</i> Always Rename the default outcome	<i>Is</i> ProjectBillable_Yes, <i>Is</i> ProjectBillable_No, <i>Check</i> RecordType_Is Resource	Yes, No, Is Resource
Assignment	Start with: <i>SET</i> : variable updates. <i>ASSIGN</i> : variable initialization, or updates on Non-Object variables. <i>ADD</i> : add elements to Collections. <i>REMOVE</i> : remove elements from Collections. <i>CALC</i> : mathematical assignment or complex collection manipulation.	<i>SET</i> _ProjectValues, <i>ADD</i> _SelectedRoles	SET Project Values, ADD Selected Roles
Loop	Start with <i>LOOP</i>	<i>LOOP</i> _ActiveProjects	LOOP Active Projects
Transform	Start with a relevant action, i.e. <i>GET</i> , <i>SET</i> , <i>CALC</i> , <i>ADD</i>	<i>CALC</i> _AggregateTotal Cost, <i>GET</i> _AllOpptyIds	<i>CALC</i> Aggregate Total Cost, <i>GET</i> all Oppty Ids

## Using Subflows

Think about Subflows as helper classes in Apex. They are a good way to standardize operations, minimize the number of Flows in the org, and simplify maintenance.

*Note: Subflows are not available in record-triggered before-save Flows (yet).*

Here are some classic use cases for when you should consider a subflow:

- Re-use: If you're doing the same thing in your Flow multiple times, or doing the same thing you did with another Flow, call a Subflow to do it.
- Complex processes/subprocesses: If your Flow involves multiple processes and branching logic, create a main Flow that launches other secondary Flows. For example, on a Flow that creates projects, a Subflow can be called to create resource requests. The same Subflow can be called from an Opportunity to create pipeline Resource Requests!
- Your Flow is hard to read: That's a sign that it got too big, or is poorly organized. Perhaps a Subflow to break it down into sub processes might help.
- Handle complex permissions scenarios: Subflows do not inherit the permissions from the calling Flow. For example on a Screen Flow that is running in user context, you need to update an object that the user doesn't have Edit access to. By using a subflow with system context permissions, you're able to temporarily grant Edit access to that user.

## Flow Context

Context matters! If a Flow is running in user context, the running user of a Flow is the one that launches it. Permissions on the user's profile including object and field level access will affect the way the Flow runs. This includes sharing rules, role hierarchies, etc.

Some types of Flows allow for determining the context from the following options:

- User or system context: The context is determined by how the Flow is launched.
- System Context with Sharing: The Flow respects org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories—but it doesn't respect object permissions, field-level access, or other permissions of the running user.
- System Context without Sharing: The Flow can access all data.

Record triggered Flows always run in system context without sharing. [Click here to learn how to change the flow context.](#)

The running context on a Flow can be defined in the Flow settings, "How to Run the Flow" drop down.

**Edit version properties**

• Flow Label  
Contract Unit: Create Consumption Schedule

• Flow API Name  
Contract\_Unit\_Create\_Consumption\_Schedule

Description

Hide Advanced

How to Run the Flow ⓘ

Select...

- User or System Context—Depends on How Flow is Launched
- System Context with Sharing—Enforces Record-Level Access
- System Context Without Sharing—Access All Data

There are a number of limitations when it comes to context and Flow/Apex/Lightning component interactions. In order to avoid transforming this document into a training guide, please refer to [this article](#) for details.

## Testing Flows

One of the appeals of Flows is that they don't require test coverage in order to get deployed. Flows need to be thoroughly tested, however. Always test your Flow as the user that will be running it, since permissions might impact the processing.

Flow debug allows for testing as other users, as long as the process automation setting is properly configured ([Automate This - Discover Tips and Best Practices for Record Triggered Flows](#)):

- In Setup, on the Process Automation Settings page, there's a setting called "Let admins debug flows as other users".
  - Make sure that this setting is checked in order to run a Flow as another user in the Flow Debug window.

**Process Automation Settings**

screen components, and local actions.

Enable Lightning runtime for flows ☒

Change the required permission to view all charts on the Automation Home page in Setup. By default, users with the View Setup and Configuration permission can view all charts. When this setting is enabled, users with View Setup and Configuration can view only the Total Started Automations by Process Type chart.

Require the Manage Flow permission to view all Automation Home charts ☐

This setting affects Lightning Experience only. On the Flows page in Setup, use an enhanced flow list view that includes standard flows. On a separate page, view paused flows and process scheduled actions. When disabled or using Salesforce Classic, the Flows page displays custom flows, paused flows, and process scheduled actions. The separate page isn't available.

In Lightning Experience, use the enhanced Flows page and separate Paused and Scheduled Automations page ☒

This setting affects flows in API version 51.0 or earlier. When creating or updating records, the flow can use record variables to define the field values. If the running user doesn't have edit access to those fields, by default the flow fails to create or update the records. This setting filters inaccessible fields from the request for Create Records and Update Records elements.

Filter inaccessible fields from flow requests ☒

Allows Apex code to set and remove approval process locks.

Enable record locking and unlocking in Apex ☐

When this setting is enabled, users with the Manage Flow and View All Data permissions can specify the running user each time they debug a flow.

**Let admins debug flows as other users** ☒

When you debug a flow as another user, the flow's record changes and actions are performed as that user. Also, the user's profile and permission sets determine the object permissions and field-level access of the flow. However, flows that always run in system context ignore the user's object permissions and field-level access.

When a process or flow interview fails, Salesforce sends an email with details about everything that was executed and what failed. The details include the data that's involved in the process or flow, including user-entered data.

Send Process or Flow Error Email to **User Who Last Modified the Process or Flow** ☒

Send an email notification every time a process, workflow rule, or Apex trigger creates or updates the same record that triggered the automation. When this setting is disabled, no email notification is sent when the same record is updated via automated updates.

Send an email each time automation updates the same record ☐

Save Cancel

- In the Debug Flow screen, make sure that “Run flow as another user” is checked, and select the running user:

**Debug flow**

**Debug Options**

☒ Run the latest version of each flow called by subflow elements

☒ Show details of what's executed and render flow in Lightning runtime ⓘ

☒ Run flow as another user ⓘ

User

Mark Ross ×

**Input Variables**

Enter values for the flow's input variables. For each value left blank, the flow starts with the variable's default value. You can't enter values for collection variables or Apex-defined variables.

recordId

Run

When you run the debug, you see what they would see and get details about any errors they would receive.

Also, on Record Triggered Flows you can create Test records and assertions to make sure that all the use cases are properly covered. See [Salesforce's help](#) for more information about testing flows.

## Error Handling

Flows are a declarative way to execute Apex. As such, they can (and will) fail. A fault connection in Flows allows for handling errors and exceptions. A recommended practice is to always include a fault action after Data and Action elements.

A fault action can include an email notification, chatter post, or display an error message to the user. In order to capture the error, use the `!$Flow.FaultMessage` variable. This can be exposed in a screen component, and included as text on a text template that is then part of a send email action.

Note that the screen element can be used to show an error message only when creating a screen Flow. For other Flow types (record triggered, schedule triggered, non-triggered Flows) you can post to Chatter, send an email, send a Slack message or create a new record to log error details.

If your org has an error log object, ideally the Flow fault should create a new entry in the error log.



# Checklist of 20-ish Flow Tips

If you're short on time, here's a summarized list of things to consider to design and use Flows in your Salesforce org.

1. For Essentials or Professional orgs, use one Flow per object per type.
2. On Enterprise Orgs, use common judgment to determine how to group Flows.
3. Design your Flow before you start building.
4. Document your Flow and Flow Elements by providing meaningful descriptions.
5. Use Fast Field Update any time you can.
6. Watch out for governor limits.
7. Use an asynchronous path if inserting/updating setup and non-setup objects on the same transaction.
8. Use Assignment and not Update on Fast Field Updates Flow.
9. Never hard-code IDs.
10. Define good entry criteria, if possible.
11. Add a bypass Flow mechanism in all record triggered Flows.
12. Check for null or empty results in Decision elements.
13. Control the order of execution using Flow Trigger Explorer.
14. Avoid complex formula variables.
15. Use Flow Action Buttons, Screen Actions, pictures, icons, HTML markups and any other available resource to enhance user experience on Screen Flows.
16. Check with the customer if they have Flow guidelines in place.
17. Follow naming guidelines and conventions to make Flows more standardized and readable.
18. Use Subflows for repetitive/reusable processes.
19. Carefully define Flow Context.
20. Test your Flow as the running user.
21. Have an error handling strategy to manage errors and exceptions.
22. Have fun!

# About Us

We help enterprises implement Salesforce and Certinia solutions while optimizing their services business and financial operations.

We advise, design and implement solutions for professional services automation and help services businesses have a comprehensive view of their quote to cash operations. We created [PSQuote](#) for professional services quoting (available on the AppExchange) and various [Accelerators](#) to improve accuracy & save time.

## Who We Are

Our consultants and our leadership have years of experience in implementing and developing cutting-edge technology that streamlines business processes and improves service to customers, employees and vendors.

## What We Do

Salesforce, Certinia PSA and ERP implementations are complicated. Without advanced planning, designing new processes, migrating data, and training employees—the system won't perform the way you want. CLD Partners guides you through your business transformation, giving you the value you expect from a big investment.



## Our Mission

By taking everything we have learned over the years in traditional custom application development, and applying our extensive knowledge and experience with the Salesforce platform, we repeatedly deliver high-quality solutions tailored to our customers needs while providing all of the benefits that the cloud paradigm has to offer.



## Our Vision

What drives us? Our genuine desire to help our clients be successful. We know that every client's operations are unique. That's why we approach each with a fresh perspective, without assuming we already know the solution. We listen. And then we develop and deliver a solutions that fits their enterprise and processes. You might say we're hopelessly devoted to delivery.

**CLD**Partners

# CLD Partners

## THANK YOU

We hope this document helps your organization operate efficiently and effectively with a well-run Salesforce organization.

Take time to review and manage your Salesforce configuration to maintain data integrity. **Don't have time? Not sure where to start?**

### CONTACT US.

WE'RE HERE TO HELP.



[cldpartners.com](https://cldpartners.com)



[info@cldpartners.com](mailto:info@cldpartners.com)