Flow Best Practices

A. Use descriptive and consistent names

1. Flow names should be descriptive of their functionality.
2. Consider changing names when their functionality changes.
3. Flow elements should be descriptive of their functionality.
4. Change names when their functionality changes.
5. Flow resources: let's use camel case starting with a lowercase letter, eg. customerName, customerNames, customerList

B. Never hard-code IDs or reference data.

6. Don't hard-code IDs for things like Owner IDs, Queue IDs, Groups, or Record Type Ids. Do a 'Get' for the respective object using the DeveloperName.
7. Use the custom setting "Configuration Ids" for record IDs.
8. Don't hard-code reference logic in a decision that could change frequently such as Queue IDs, OwnerIDs, or numbers (like a discount percentage). Use Custom Labels and Custom Metadata!

C. Prioritize understandability, maintainability, and performance for design decisions

Choose between flow and Apex: Avoid complex flows

9. Choose flow or Apex following the Record-Triggered Automation decision guide.

Deliberate one or multiple triggered flows (with trigger order) on an object

10. Use multiple triggered flows on the same object if necessary.
11. Set the Trigger Order on triggered flow to be 10, 20, 30, …
12. The order of execution needs to be deterministic and known to make debugging easy.

Use subflows or Apex invocable methods

13. Use subflows to simplify flows.
14. Use well designed Apex invocable methods.

Leverage External Components

15. Use of components from the Automation Component library or UnofficialSF need to be documented for future upgrade.

Avoid anything obscure

16. Avoid any obscure mechanisms. Document anything that can confuse a colleague. For example, Was Set may not be well understood by everyone yet.

D. Check record and field security & flow context

17. Don’t assume your user can see and do everything you designed if you’re building a screen flow. Test your flows as both your intended and unintended audiences.

E. Keep limits and performance in mind

18. Don’t loop over large collections of records that could trigger the Flow element limit (currently 2,000).

19. Bulkify DML: Leave any DML towards the end of the flow whenever possible.

20. No DML Statement in Loops: Never performing a repetitive Get, Update, Create, or Delete inside of a Loop in an Autolaunched flow or Record-Triggered flow. Similar to Apex, we should always use bulkification wherever we can apply that otherwise we will get SOQL limit exception error. Screen flows are okay if you’re 100% sure the scope won’t trigger governor limits.

21. Re-use Get Records to retrieve other SObjects that are not directly related to the record being processed. I.e. If new functionality requires retrieving new fields of an unrelated object, find out if the flow is already retrieving it, and use it instead of adding different Get Records that may increase the SOQL count.

F. Keep scheduled flows in reasonable numbers.

22. Try to consolidate business logic on the same object & schedule, make the conditions in the start element more generic to support all the logic.

G. Handle Errors

23. What do you want to happen when the flow hits an error? Who should be notified? Use fault paths!
24. In a few months, we will have a platform event based asynchronous error logging mechanism to use.

H. Support Automation Bypass

25. Use the custom setting “Is Disabled” to support automation bypass. Add checkbox fields to disable different functionalities. It allows us to bypass flows, subflows or even specific parts of flows. And we have the advantage of being able to use the same flags in Apex too or even any PB so it will allow us to move the PBs to flow safely.
26. Ideally we all should use the same custom setting once it's in production. Let's avoid every developer creating their own custom setting or using labels or any other mechanism.

References

1. Record-Triggered Automation
2. The Ultimate Guide to Flow Best Practices and Standards By Adam White, June 16, 2021
3. Automation Champion: salesforce-flow-design-patterns-from-fundamentals-to-mastery