Dreamforce

Architect’s Guide to Prompt Engineering

Chetan Chugh
chetan.chugh@capgemini.com

Sebastiano Schwarz
sebastiano.schwarz@capgemini.com

Forward-looking statements

This presentation contains forward-looking statements about, among other things, trend analyses and statements regarding future events, anticipated growth and industry prospects, and our strategies, expectation or plans regarding product releases and enhancements. The achievement or success of the matters covered by such forward-looking statements involves risks, uncertainties and assumptions. If any such risks or uncertainties materialize or if any of the assumptions prove incorrect, results or outcomes could differ materially from those expressed or implied by these forward-looking statements. The risks and uncertainties referred to above include those factors discussed in Salesforce's reports filed from time to time with the Securities and Exchange Commission, including, but not limited to our ability to meet the expectations of our customers; uncertainties regarding AI technologies and their integration into our product offerings; the effect of evolving domestic and foreign government regulations; regulatory developments and regulatory investigations involving us or affecting our industry; our ability to successfully introduce new services and product features, including related to AI and Agentforce; our ability to execute our business plans; the pace of change and innovation and our ability to compete in the markets in which we participate; and our ability to maintain and enhance our brands.

Dreamforce

Thank you

Coffee on us.

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.*

1. Download the Salesforce Events mobile app.
2. Navigate to My Event, then My Surveys.
3. Complete (4) session surveys and the Event Survey.
4. Redeem your gift at Badge Pickup on September 17th.

*Restrictions apply. See terms and conditions at sforce.co/survey-terms

Who are we ?

Chetan Chugh

Chief Architect, Salesforce CTA & MVP
Capgemini

Sebastiano Schwarz

Salesforce CTO Germany,
Capgemini

Most teams think prompting is about wording.

It’s about the controls.

THE SYMPTOM

When AI Disappoints, We *Blame* the Tool

“The model hallucinated”

“The output was too generic”

“The agent ignored my instructions”

“The responses are inconsistent”

But the Architects should ask a different question.

- What did we actually give the model to work with ?
- Was the necessary context provided ?
- Were constraints explicitly clarified ?
- Did it have access to the right tools for verification ?
- Did we define what a successful output looks like ?

The quality of AI generated output is determined less by wording of prompt and more by control mechanisms around it.

THE PATTERN

Every Prompt needs FOUR Control moves

Swapping vague for specifics: in what the model knows, what it can't do, what it can call, what it must return

01

Context

What the model needs to know about the org, the org's metadata, and the task.

02

Constraints

What the model must not do — and what to do instead, so guardrails redirect, not just block.

03

Tools

What the model can call — describe calls, search, code execution, connected systems.

04

Output + Evaluation

The exact shape of the response, plus the self-check it must pass before it ships.

Together, they turn a single prompt into a system — this is what "prompting as system design" means.

DEMO 1

Assessing Org mergers

UNSTRUCTURED PROMPT

"I pulled the metadata export for Org 1 — objects are Store_Location__c, Product_Bundle__c, and a legacy Order__c that doesn't match our Retail_Order__c.

Can you look through this and tell me what's going on before we merge it into Org2 ahead of the Q3 code freeze?

Flag anything risky."

STRUCTURED PROTOCOL

CONTEXT:

Object/field metadata for Store_Location__c, Product_Bundle__c, Order__c (Org 1) and Retail_Order__c, Warehouse_Transfer__c (Org 2), plus both orgs' automations

CONSTRAINTS:

Flag objects where 2+ automation types share an event context or write the same field. Exclude managed-package components from remediation, keep them in totals.

TOOLS:

Tooling API for active automation (flows, triggers, Process Builder, ValidationRule) + MetadataComponentDependency + schema diff on Order__c vs Retail_Order__c

OUTPUT:

merge-readiness-scorecard.md — ERD + risk-scored table (Object, Risk Type, Severity, Merge Blocker, Remediation). Self-check: every Blocker = Yes row must cite the field or class it flags

THE GUIDE

What the merger review just taught us

Two rules for the Guide, straight from what was just built

Rule 1 — Name the artifact, not the category

When you say “the order object” and the model has to guess which one you mean — and it'll guess wrong.
Actual API names leave nothing to guess.

FROM THIS DEMO:

```
The prompt named the exact objects and class:
Retail_Order__c, Warehouse_Transfer__c, and ProductBundleHandler.cls.
```

Rule 2 — Verify before assuming

Same name, same purpose — not always the same structure.
Verify current metadata and dependencies in each org before you map or merge.

FROM THIS DEMO:

```
Before planning the merge, it compared Order__c with Retail_Order__c.
It also checked each org’s automation instead of assuming they behaved alike.
```

DEMO 2

Scoping the Order Sync

UNSTRUCTURED PROMPT

"We're retiring OrderSyncService.cls — the point-to-point Apex integration to SAP S/4HANA — for MuleSoft Anypoint.
Before we scope the build, can you look at logic inside classes like OrderSyncService.cls & SAPCalloutHandler.cls and explain how the sync to SAP actually works end to end?"

STRUCTURED PROTOCOL

CONTEXT:

OrderSyncService.cls, SAPCalloutHandler.cls, named credential Aurora_SAP_Prod (JWT bearer), and the MuleSoft Order API OpenAPI spec

CONSTRAINTS:

Classify each call sync vs async. Flag callouts inside loops and any callout that runs after DML in the same transaction (OrderSyncService.pushTransfer). Flag missing idempotency keys. Treat every finding as a migration-checklist item.

TOOLS:

Static code search across both classes using MetaData API + + OpenAPI diff against the MuleSoft Order API spec

OUTPUT:

integration-cutover-checklist.md — sequence diagram + table (Endpoint, Current Pattern, Target Pattern, Risk, Cutover Order).

Self-check: every Risk = High row must name the limit it violates

THE GUIDE

What the integration teardown just taught us

Two more rules, straight from the cutover checklist

Rule 3 — Point at a verifiable tool, not general knowledge

Ask the model to use “your Salesforce knowledge” and it answers from memory, not from your org.
Naming the exact API or search forces it to go look.

FROM THIS DEMO:

`The prompt named the checks to run: search both Apex classes and inspect the Mule API specs.`

Rule 4 — Make every finding actionable, not descriptive

A paragraph explaining a risk is something someone has to re-read and translate into a task.
A sequence number and an owner is already the task.

FROM THIS DEMO:

`Each finding became a cutover step with a clear order and risk — not just a paragraph describing the problem.`

DEMO 3

Designing Agent Metadata in Industries Cloud org

UNSTRUCTURED PROMPT

"We keep getting requests from business admins to add new fields to Financial Account and other FSC objects, and half the time it breaks something on the next release.
Can you help design an Agentforce agent that handles these requests instead of admins going straight to Setup?"

STRUCTURED PROTOCOL

CONTEXT:

FinServ__FinancialAccount__c, FinServ__FinancialAccountRole__c, the standard ActionPlan / ActionPlanTemplate objects, and the Household record type with FinServ__ReciprocalRole__c relationships already in the org

CONSTRAINTS:

Governance policy, not a platform limit — only propose a custom object with a lookup, never a custom field on a FinServ__ object. Discover existing automation before drafting.

TOOLS:

Metadata describe scoped to the FinServ__ namespace, Tooling API to find existing Action Plan Templates and automation

OUTPUT:

metadata-change-request.md, discovery summary, proposed custom-object design, and an explicit approval flag.

Self-check: Approval defaults to Needs Approval unless sign-off appears in Context

THE GUIDE

What the FSC metadata creation just taught us

The final two rules, straight from Metadata_Copilot

Rule 5 — Forbid the model's default shortcut, not just the goal

A model defaults to the most common pattern in its training data — here, a field on the managed object. Describing the right answer isn't enough; you must rule out the wrong one.

FROM THIS DEMO:

```
The prompt blocked the tempting shortcut:
do not add a custom field to the managed
Financial Account object. Propose a custom
object with a lookup instead.
```

Rule 6 — Make the model show its discovery, not just its answer

If the output format has no slot for “what I found already exists,” you have no way to tell whether the model actually checked or just guessed.

FROM THIS DEMO:

```
Before proposing a design, the model had to
list what it discovered in the org. That
check surfaced existing Action Plan templates
the new design needed to account for.
```

Kumar Kasimala

Software Engineering Architect, Prompt Builder
Salesforce

What this demo will show

Methodology

Effective prompt engineering

By combining Context, Constraints, and Structured Output.

This systematic approach delivers the accurate, high-quality, and deterministic results you need to confidently automate critical business processes.

01 **Accurate**
Only facts on the Case

02 **Quality**
Safe to send – or hold

03 **Deterministic**
Same shape every run. A flow can act.

The complete field guide, at a glance

1

Name the artifact, not the category

Vague names force a guess. Real API names don't.

2

Verify before assuming

Same name, same purpose — not always same structure.

3

Point at a verifiable tool

Named APIs give the model explicit path to verification instead of relying solely on prior knowledge.

4

Make findings actionable

Add an owner and sequence. Turn every finding into a task.

5

Forbid the model's default shortcut

Block the wrong path — not just state the right outcome.

6

Make the model show its discovery

Require a discovery summary. If it isn't shown, it wasn't verified.


With AI, everyone has access to the same models.
The Differentiator is the architecture around it.

Coffee on us.

The first 4,000 attendees to provide feedback on this event will receive a $5 Starbucks gift card.*

1. Download the Salesforce Events mobile app.
2. Navigate to My Event, then My Surveys.
3. Complete (4) session surveys and the Event Survey.
4. Redeem your gift at Badge Pickup on September 17th.

*Restrictions apply. See terms and conditions at sforce.co/survey-terms

Session summaries available soon

A short recap of this session is on its way. You'll find it later today in the **Salesforce Events mobile app** or online on the **sessions page**.