



AIP-C01: Foundation Models and Bedrock APIs

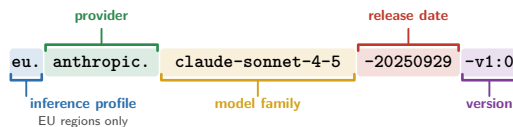
AIP-C01 Domain 1 (foundation model integration), Amazon Bedrock user guide and API reference, as of 2026 | Choosing a model, Converse vs InvokeModel, streaming, tool use, inference profiles and cross-region inference, throughput and service tiers, batch, prompt caching, quotas

AIP-C01 Domain 1 Bedrock Converse API

☉ Choose the foundation model

Domain 1 of AIP-C01 (foundation model integration, data management and compliance) is the largest at about 31 % (as of 2026, check the official exam guide). This sheet covers selecting and calling models; sheet 02 covers data management and RAG; sheet 03 prompts and customisation. Bedrock and generative-AI depth for the whole AWS course lives in this chapter.

Criterion	What to check	Example decision
Task and modality	text, code, images in or out, video, speech, embeddings	Nova Canvas or Stability for images; Titan / Cohere for embeddings
Quality	Bedrock model evaluation on your own prompts, not public leaderboards	LLM-as-a-judge run on 200 real questions
Latency	time to first token, tokens per second, latency-optimised options	small model (Haiku, Nova Micro/Lite) for chat autocomplete
Cost	input and output price per million tokens, caching, batch	route easy questions to a cheaper model
Context window	longest prompt incl. retrieved chunks and history	long contracts → long-context model or chunked RAG
Region and residency	model availability per region, cross-region profiles	EU data → eu. profile, no global.
Customisation	fine-tuning, distillation, continued pre-training support per model	only some models can be fine-tuned
Licence and provider terms	open weights vs proprietary, acceptable-use policy	Llama or Mistral if you need to self-host later



Reading a model ID: the prefix decides where requests may be processed

📖 EXAM TIP

Model choice traps: 'LOWEST cost for simple classification at high volume' → the smallest adequate model, batch inference if offline. 'Data must stay in the EU' → EU cross-region inference profile or a single EU region, not global. 'Throughput beyond the on-demand quota at peaks' → cross-region inference before Provisioned Throughput.

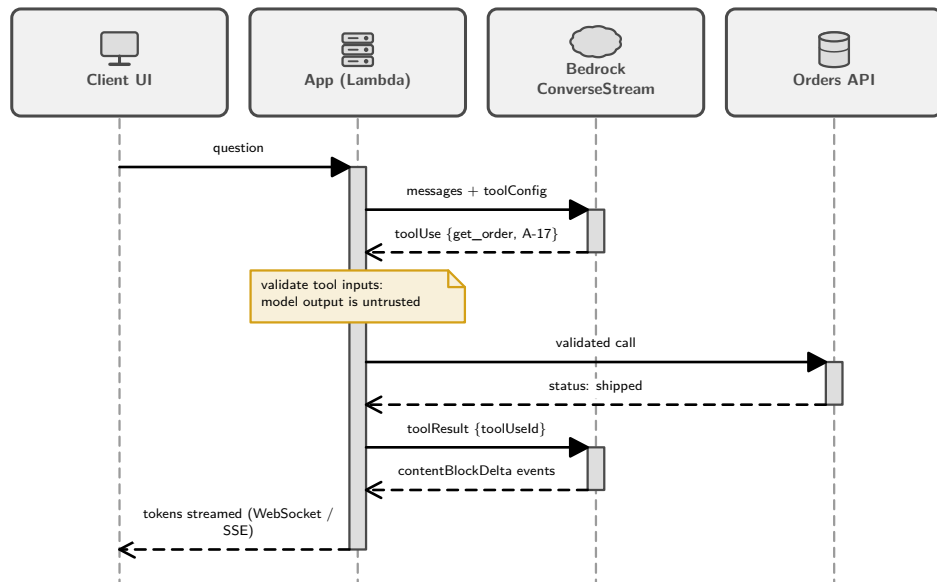
🔗 Calling models: Converse, streaming, tools

Converse with a system prompt, inference settings and a tool

```
import boto3, json
brt = boto3.client("bedrock-runtime", region_name="eu-west-3")
tools = {"tools": [{"toolSpec": {
    "name": "get_order",
    "description": "Look up an order by its id",
    "inputSchema": {"json": {"type": "object",
        "properties": {"order_id": {"type": "string"}},
        "required": ["order_id"]}}}}]}

resp = brt.converse(
    modelId="eu.amazon.nova-pro-v1:0",

    system=[{"text": "You are a support agent. Be brief."}],
    messages=[{"role": "user",
        "content": [{"text": "Where is order A-17?"}]}],
    inferenceConfig={"maxTokens": 400, "temperature": 0.2},
    toolConfig=tools)
if resp["stopReason"] == "tool_use":
    call = resp["output"]["message"]["content"][-1]["toolUse"]
    # run get_order(**call["input"]) then send a toolResult
```



Tool use loop with ConverseStream: the app, not the model, runs the tool

```

{
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "text": "Order A-17 shipped yesterday."
        }
      ]
    }
  },
  "stopReason": "end_turn",
  "usage": {
    "inputTokens": 1830,
    "outputTokens": 42,
    "totalTokens": 1872,
    "cacheReadInputTokens": 1500
  },
  "metrics": {
    "latencyMs": 640
  }
}
  
```

tool_use → run the tool; max_tokens → answer was cut
log per request for cost; cache reads are cheaper

Converse response: the fields an application reads

API

Use

<code>converse / converse_stream</code>	chat, tools, documents, images, guardrails; same shape across providers
<code>invoke_model / invoke_model_with_response_stream</code>	native body: embeddings (Titan, Cohere), image and video generation
<code>start_async_invoke</code>	long-running generation (video) written to S3
<code>create_model_invocation_job</code>	batch inference from JSONL in S3
<code>aws bedrock list-inference-profiles --type-equals SYSTEM_DEFINED</code>	cross-region profiles available in the region

③ Throughput, cost and latency options

	Recommended On-demand	Cross-region inference	Batch inference	Provisioned Throughput	Service tiers (late 2025)
Billing	per input/output token	source-region on-demand price	per token, discounted for supported models	per model unit per hour	Priority premium, Flex discount
Best for	variable traffic	peaks, resilience	offline jobs	steady high volume, custom models	latency-critical vs tolerant
Latency	standard	standard	hours	consistent	Priority fastest
Watch out	per-model RPM/TPM quotas	processing in other regions of the profile	JSONL in S3, minimum records per job	billed while it exists	model availability varies; check

Monthly token cost of a feature

FORMULA $C = R \times (T_i \times P_i + T_o \times P_o) / 1000000$

WHERE

- C = monthly cost (USD)
- R = requests per month (requests)
- T_i = average input tokens per request (system + context + question) (tokens)
- T_o = average output tokens per request (tokens)
- P_i = input price per million tokens (illustrative) (USD)
- P_o = output price per million tokens (illustrative) (USD)

WORKED

```

R = 500000
Ti = 3000
To = 400
Pi = 0.8
Po = 3.2
C = 500 000 × (3 000 × 0.8 + 400 × 3.2) / 1000000
C = 1 840 USD per month
  
```

RESULT C = 1 840 USD per month

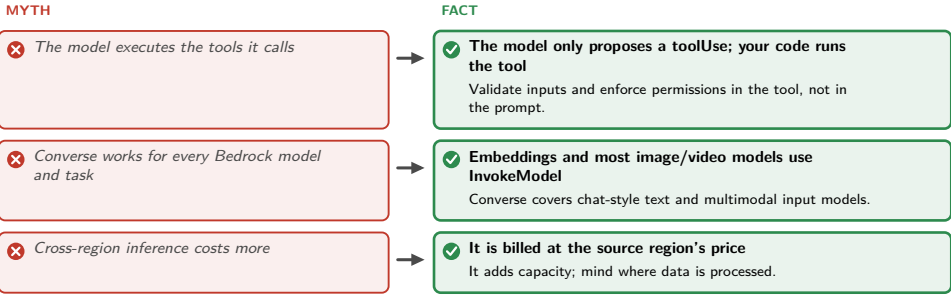
READ IT 1,200 USD input + 640 USD output: input dominates, so cache the stable prefix and retrieve fewer chunks before changing the model

MISTAKES

- Prices differ per model and region: read the Bedrock pricing page
- Agents multiply calls: one user question can cost 3-10 model invocations
- Output tokens usually cost several times input tokens: cap maxTokens

- **Prompt caching** put stable content first (system prompt, tool definitions, long documents) and a cachePoint block after it; cache lives about 5 minutes after the last hit; check cacheReadInputTokens / cacheWriteInputTokens in usage; supported models and minimum prefix sizes vary
- **Latency-optimised inference** performanceConfig={'latency': 'optimized'} on supported models and regions for lower time to first token (as of 2026, check availability)
- **Intelligent prompt routing** a prompt router sends each request to one of several models of a family by predicted quality and cost
- **Quotas** requests and tokens per minute per model per region in Service Quotas; raise them before launch; retry ThrottlingException with exponential backoff and jitter

✓ Myths, recap and self-test



Tick a box only when you could explain the point to someone else

☐	Evaluate models on your prompts quality, latency, cost, region	☐	Converse by default InvokeModel for embeddings and media
☐	Pick throughput by traffic shape on-demand, cross-region, batch, provisioned	☐	Cache stable prefixes input tokens dominate cost
☐	Plan quotas TPM/RPM, backoff, cross-region		

- Q1

Which stopReason means the answer was truncated?

A

max_tokens; raise maxTokens or shorten the request.
- Q2

API for embeddings with Titan Text Embeddings V2?

A

InvokeModel (native body with inputText, dimensions, normalize).
- Q3

Track Bedrock cost per application with tags?

A

Create an application inference profile with cost allocation tags and call it as the modelId.
- Q4

Two million offline summaries, latency irrelevant, cheapest?

A

Batch inference (create_model_invocation_job) from JSONL in S3.
- Q5

Steady high volume on a fine-tuned model?

A

Provisioned Throughput (model units), required for some custom models.
- Q6

What marks the end of the cacheable prefix?

A

A cachePoint content block in the system prompt, messages or tools.
- Q7

How should a client handle ThrottlingException?

A

Retry with exponential backoff and jitter; raise quotas or use cross-region inference.
- Q8

What does temperature 0 vs 1 change?

A

0: most deterministic output; 1: more varied sampling.