# Apex Callout Framework — Technical Design

**Purpose** A configurable, extensible Apex framework for making HTTP callouts from Salesforce. Uses Custom Metadata for endpoint configuration and the Factory pattern to instantiate callout implementations. Supports headers, retries, Named Credentials, timeouts, and structured error handling.

## Contents

1. Overview
2. Sequence Diagram
3. Custom Metadata Type: API\_Resource\_Endpoint\_\_mdt
4. Apex Architecture (interfaces, factory, implementation)
5. Error Handling & Retry Strategy
6. Example Usage

## 1) Overview

* Store API connection settings in a Custom Metadata Type (CMDT).
* Use Named Credentials for authentication where possible.
* Factory reads CMDT and returns an implementation of IApiCallout.
* Implement retries and HTTP-status-aware error handling.

## 2) Sequence Diagram

A diagram with text and arrows

AI-generated content may be incorrect.

## 3) Custom Metadata Type: API\_Resource\_Endpoint\_\_mdt

| Field Label | API Name | Type | Description |
| --- | --- | --- | --- |
| Endpoint | Endpoint\_\_c | Text | Relative/full path (e.g. /v1/accounts) |
| HTTP Method | HttpMethod\_\_c | Picklist | GET/POST/PUT/DELETE/PATCH |
| Named Credential | NamedCredential\_\_c | Text | Named Credential developerName |
| Timeout (ms) | Timeout\_\_c | Number | Timeout in milliseconds |
| Retry Limit | RetryLimit\_\_c | Number | Max retry attempts (integer) |
| Headers | Headers\_\_c | Long Text | JSON string with static headers |

## 4) Apex Architecture

### Interfaces

public interface IApiCallout {  
 HttpResponse performCallout(String requestBody);  
}

### Factory

public static IApiCallout get(String configName) {

API\_Resource\_Endpoint\_\_mdt apiResource = [

SELECT

DeveloperName, Endpoint\_\_c, Method\_\_c,

NamedCredential\_\_c, Timeout\_\_c, RetryLimit\_\_c,

Headers\_\_c, ImplType\_\_c

FROM API\_Resource\_Endpoint\_\_mdt

WHERE DeveloperName = :configName

WITH SYSTEM\_MODE

LIMIT 1

];

//TODO: Implement SOAP callout

// if (apiResource.ImplType\_\_c == 'SOAP') {

// return new SoapApiCallout(apiResource);

// }

// default

return new RestApiCallout(apiResource);

}

### Concrete Implementation (REST)

public with sharing class RestApiCallout implements IApiCallout {

private API\_Resource\_Endpoint\_\_mdt cfg;

private Integer timeout;

public RestApiCallout(API\_Resource\_Endpoint\_\_mdt config) {

this.cfg = config;

}

public HttpResponse performCallout(String requestBody) {

Http http = new Http();

HttpRequest req = new HttpRequest();

timeout = cfg.Timeout\_\_c == null ? 5000 : Integer.valueOf(cfg.Timeout\_\_c);

String endpoint = (cfg.NamedCredential\_\_c != null)

? 'callout:' + cfg.NamedCredential\_\_c + cfg.Endpoint\_\_c

: cfg.Endpoint\_\_c;

req.setEndpoint(endpoint);

req.setMethod(cfg.Method\_\_c == null ? 'GET' : cfg.Method\_\_c);

req.setTimeout(timeout);

// Headers from CMDT (JSON expected)

if (String.isNotBlank(cfg.Headers\_\_c)) {

try {

Map<String, Object> headers = (Map<String, Object>) JSON.deserializeUntyped(cfg.Headers\_\_c);

for (String k : headers.keySet()) {

req.setHeader(k, String.valueOf(headers.get(k)));

}

} catch (Exception e) {

// log invalid header JSON

System.debug(LoggingLevel.ERROR, 'Invalid Headers JSON: ' + e.getMessage());

}

}

if (req.getMethod() == 'POST' || req.getMethod() == 'PUT' || req.getMethod() == 'PATCH') {

req.setHeader('Content-Type', 'application/json');

req.setBody(requestBody);

}

Integer maxRetries = cfg.RetryLimit\_\_c == null ? 0 : Integer.valueOf(cfg.RetryLimit\_\_c);

Integer attempt = 0;

HttpResponse res;

Long backoffMillis = 500; // base backoff

while (true) {

try {

res = http.send(req);

if (res.getStatusCode() >= 200 && res.getStatusCode() < 300) {

return res;

}

// status-based decision

if (shouldRetry(res) && attempt < maxRetries) {

attempt++;

exponentialBackoffSleep(backoffMillis, attempt);

continue;

}

// no retry: throw a typed exception

throw createHttpException(res);

} catch (System.CalloutException ce) {

if (attempt < maxRetries) {

attempt++;

exponentialBackoffSleep(backoffMillis, attempt);

continue;

}

throw ce;

}

}

return res;

}

private Boolean shouldRetry(HttpResponse res) {

Integer s = res.getStatusCode();

// retry on 429 (rate limit) and 5xx server errors

return (s == 429) || (s >= 500 && s < 600);

}

private void exponentialBackoffSleep(Long base, Integer attempt) {

// Apex doesn't allow Thread.sleep; use a simple pause via Limits or just increase timeout between attempts

timeout = Integer.valueOf(base \* Math.pow(2, attempt));

}

private Exception createHttpException(HttpResponse res) {

Integer s = res.getStatusCode();

String body = res.getBody();

if (s == 400) {

return new CalloutException('Bad Request: ' + body);

}

if (s == 401 || s == 403) {

return new CalloutException('Authentication/Authorization Error: ' + body);

}

if (s == 404) {

return new CalloutException('Not Found: ' + body);

}

if (s >= 500) {

return new CalloutException('Server Error: ' + s + ' - ' + body);

}

return new CalloutException('HTTP Error: ' + s + ' - ' + body);

}

public class CalloutException extends Exception {

}

}

## 5) Error Handling & Retry Strategy

* **Retry on**: 429 (rate limiting) and 5xx (server) errors, and callout exceptions (transient network issues).
* **Do not retry on**: 4xx client errors other than 429, unless business logic dictates.
* **Max retries** configured in CMDT.
* **Backoff strategy**: exponential backoff (e.g., base 500ms \* 2^(attempt-1)). Note: Apex synchronous code cannot sleep — for spaced retries use Platform Events, Queueable or Scheduled jobs to retry later.
* **Logging**: capture request/response in a secure log (Platform Event or custom object) and ship to Splunk.

## 6) Example Usage

IApiCallout callout = CalloutFactory.get('OneSource\_AddressValidation');

String body = JSON.serialize('acc');

HttpResponse res = callout.performCallout(body);

System.debug('Response: ' + res.getBody());

## Appendix: Best Practices

* Keep secrets in Named Credentials or a secure store.
* Use CMDT only for non-sensitive configuration.
* Unit test callouts with HttpCalloutMock and cover success/failure/retry scenarios.
* Document each CMDT record with clear owner and usage notes.