goto; copenhagen

GOTO Copenhagen 2019
Conference Nov. 18 - 20

Building HTTP API SDKs that really are a kit

Darrel Miller

Microsoft Graph

..is a unified API to the Microsoft 365 data that describes the patterns of productivity, identity and security in an organization.

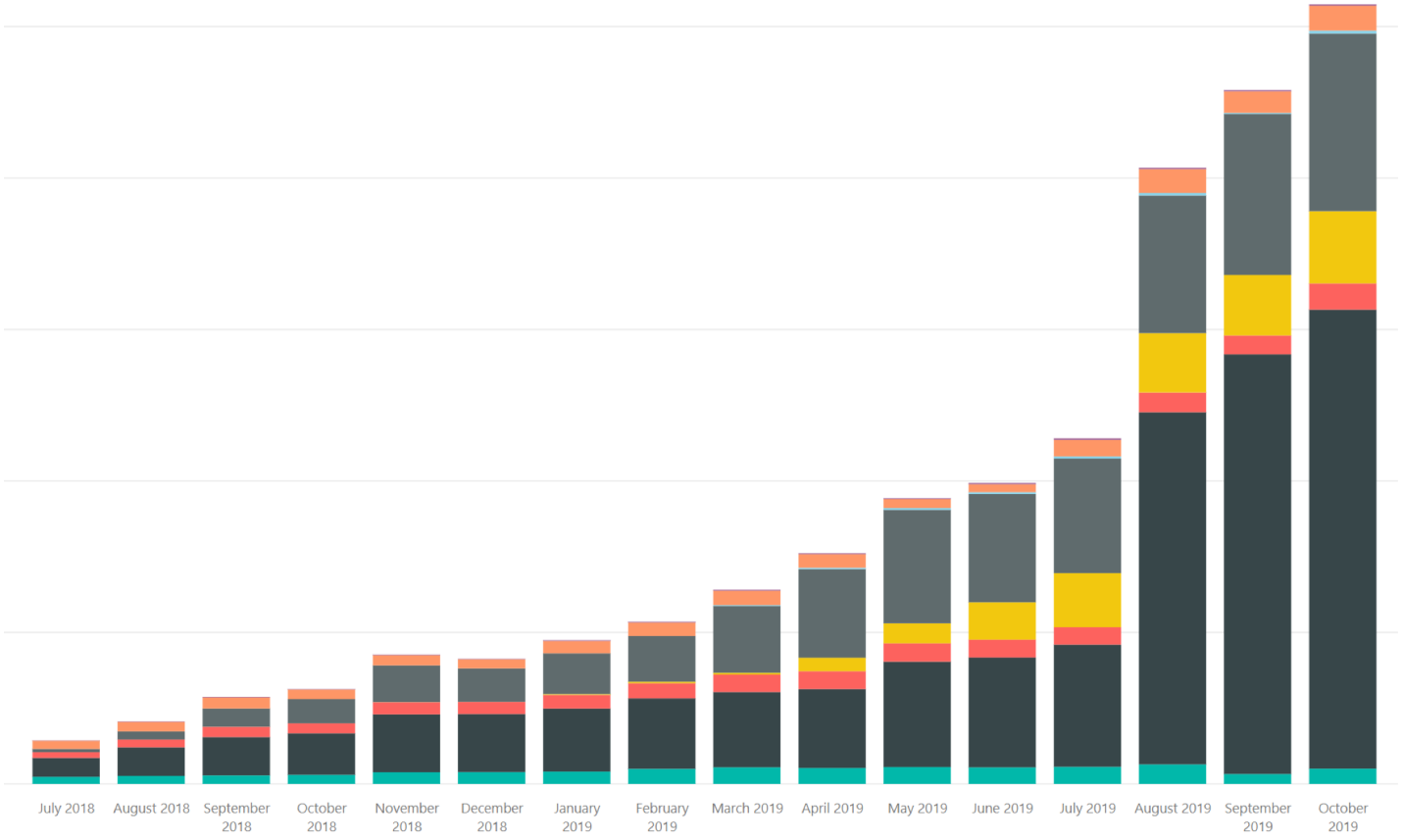
For individual users

and all the organization

- Activities
- Attachments
- Audits
- Calendar
- Categories
- Charts
- Classes
- Contacts
- Conversations
- Cross-device experiences
- Customer booking
- Device configuration
- Device management
- Domains
- Education
- Events
- Files
- Financials
- Groups
- Identity
- Lists
- Mail
- Messages
- Notes
- Notifications
- Pages
- Places
- Plans
- Reports
- Schools
- Search
- Secure score
- Security alerts
- Sites
- Social
- Subscriptions
- Tasks
- Teams
- Threat intelligence
- Users
- Workbooks

• • • and many, many more

Microsoft Graph SDK Usage Growth



Two questions:

Why isn’t everyone using the SDK?

What can be done to encourage usage?

How do you feel about SDKs for APIs?

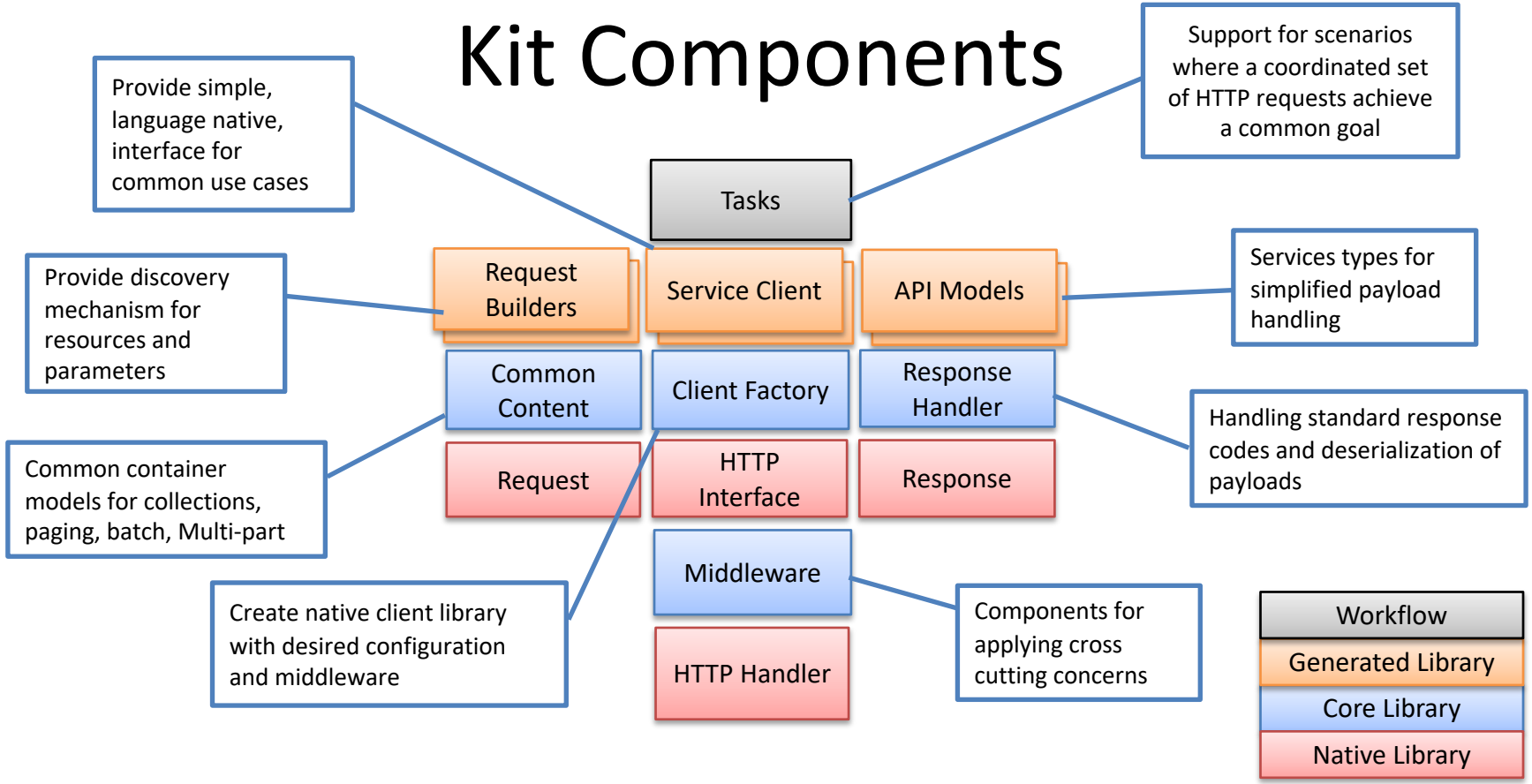
Do they make you feel like this?

TTFC

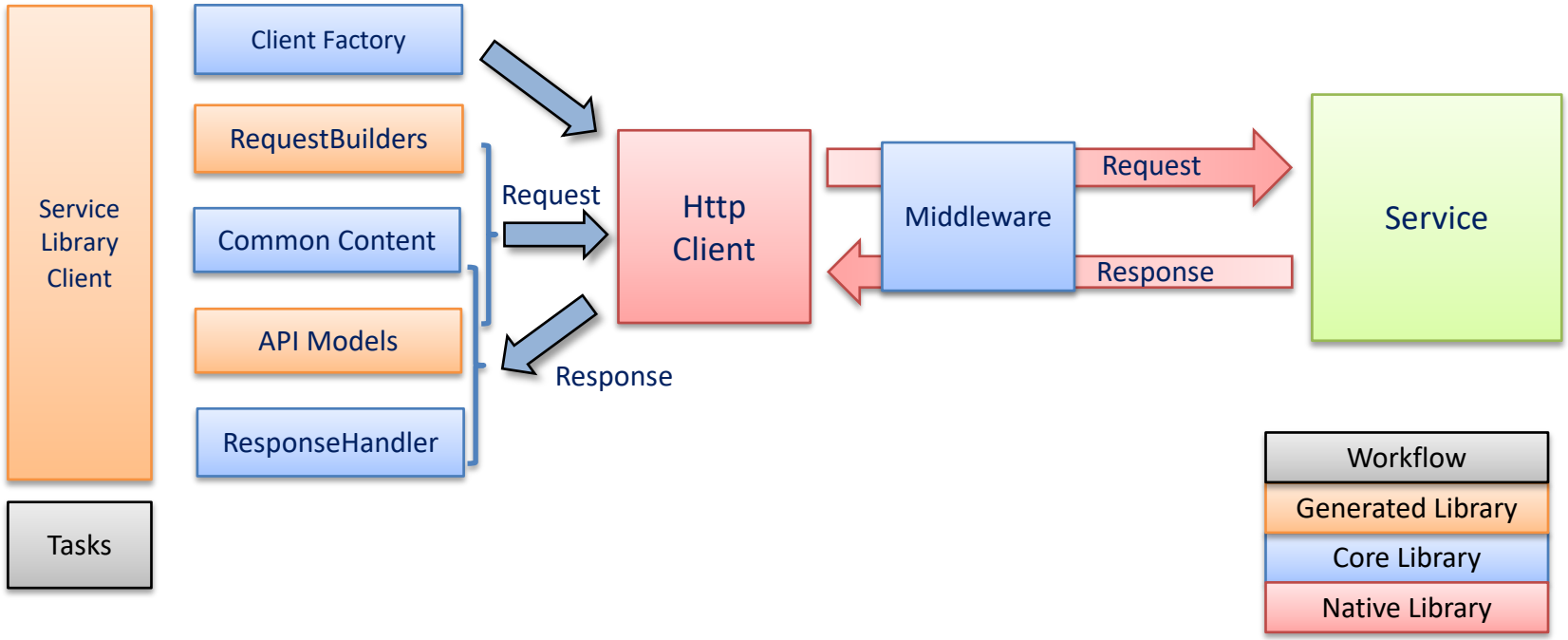
Made by a machine

This is what we need

Kit Components



Request/Response Flow



“Just Do It” Experience

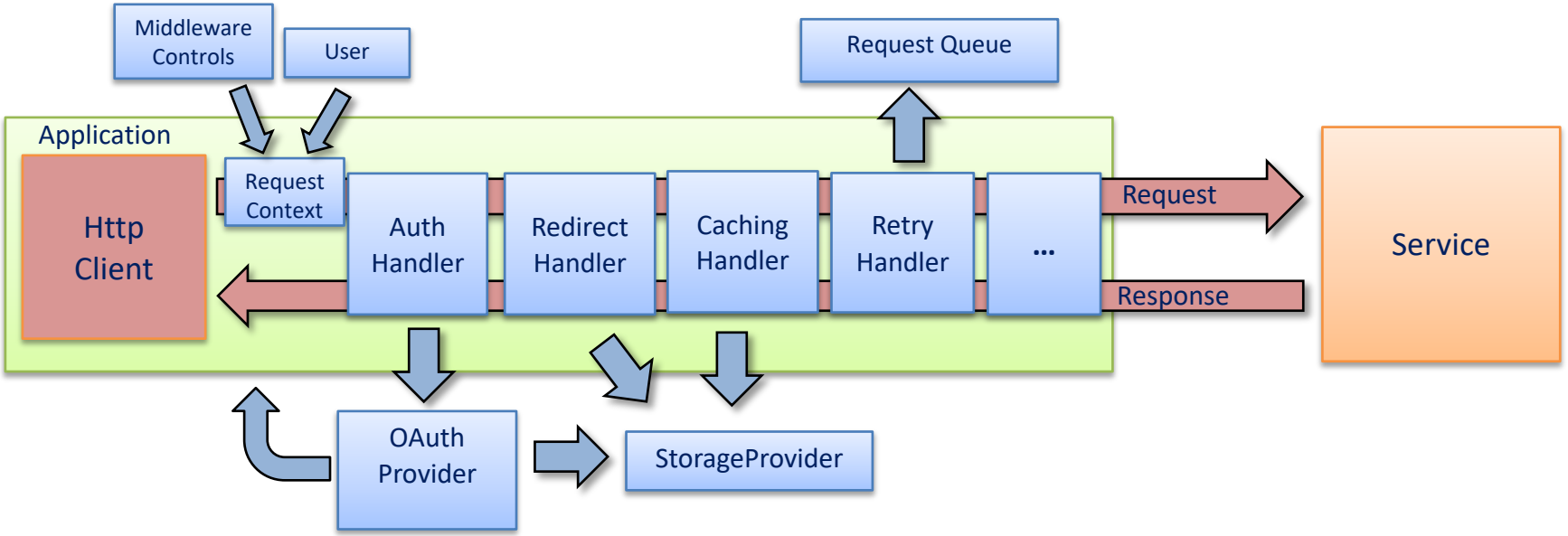
```
var graphClient = new GraphServiceClient(authProvider);

IUserMessagesCollectionPage myMessages = await
graphClient.Me.Messages.Request()
        .Top(100)
        .Select(m => m.Subject)
        .GetAsync();
```

“Just Do It” Experience

```
IGraphServiceClient graphClient = GraphServiceClient.builder()
                        .authenticationProvider(authProvider)
                        .buildClient();

IMessageCollectionPage messages =
    graphClient.me().messages().buildRequest()
         .top(100)
         .select("subject")
         .get();
```

“Do it my way” experience

```
IList<DelegatingHandler> handlers;
handlers = GraphClientFactory.CreateDefaultHandlers(authProvider);

handlers.Add(new ChaosHandler());

HttpClient httpClient = GraphClientFactory.Create(handlers);

var graphClient = new GraphServiceClient(httpClient);

IUserMessagesCollectionPage myMessages = await graphClient.Me.Messages.Request()
        .Top(100)
        .Select(m => m.Subject)
        .GetAsync();
```


Middleware Controls
User
Request Queue
Application
Http Client
Request Context
Auth Handler
Redirect Handler
Caching Handler
Retry Handler
...
Request
Response
Service
OAuth Provider
StorageProvider

Per-request middleware control

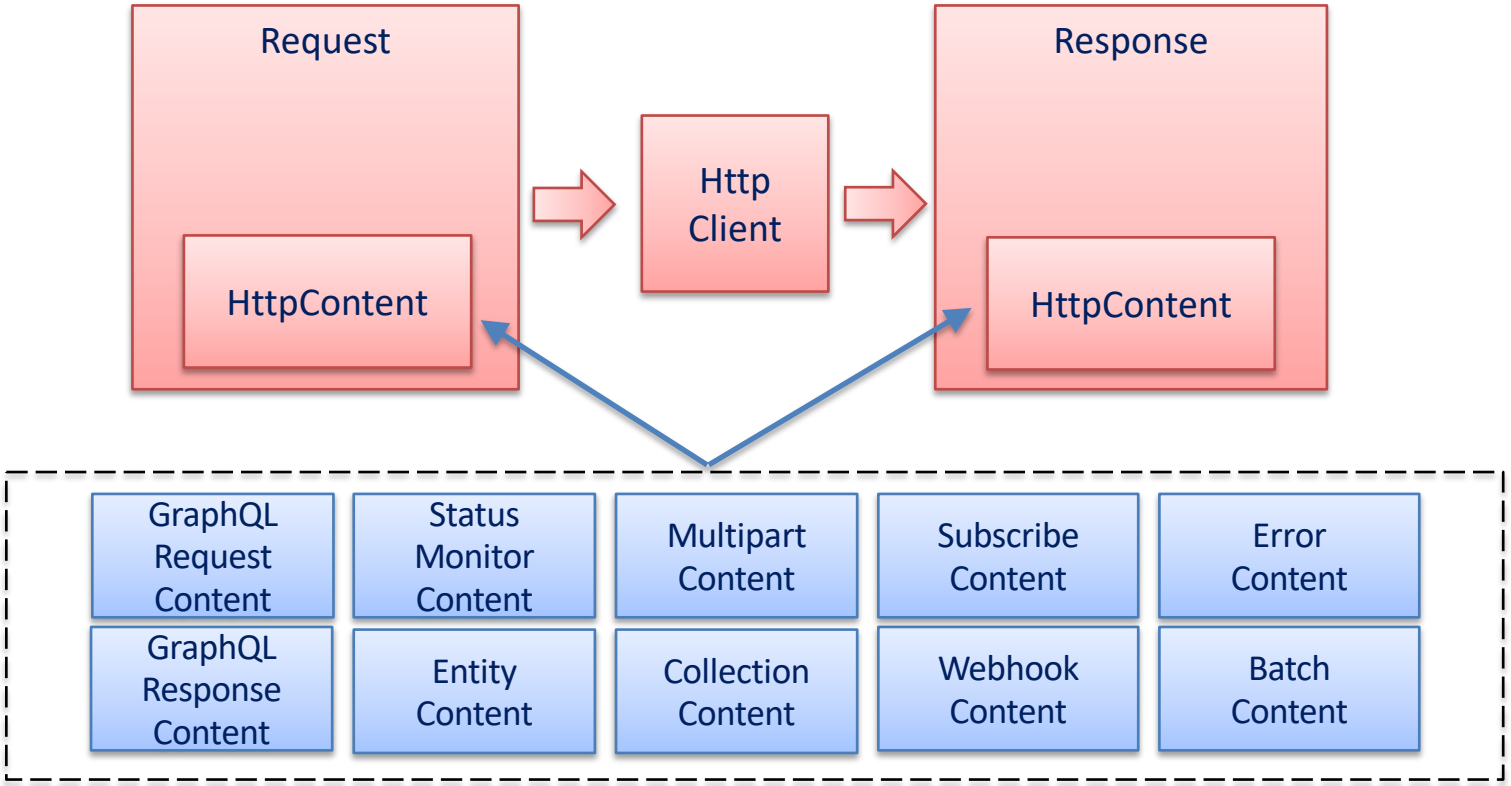
```
IUserMessagesCollectionPage myMessages = await
    graphClient.Me.Messages.Request()
        .Top(100)
        .Select(m => m.Subject)
        .WithMaxRedirects(2)
        .WithMaxRetry(new TimeSpan(0,3,0))
        .WithScopes(new string[] { "Mail.Read" })
        .WithUserAssertion(new UserAssertion(inboundToken))
        .GetAsync();
```

Native is always an option

```
IList<DelegatingHandler> handlers =
GraphClientFactory.CreateDefaultHandlers(authProvider);
HttpClient httpClient = GraphClientFactory.Create(handlers);

HttpRequestMessage requestMessage = new HttpRequestMessage()
{
    RequestUri = new Uri("/me/messages?$top=100&$select=subject", UriKind.Relative),
    Method = HttpMethod.Get
};

requestMessage.GetRequestContext().MiddlewareOptions.Add(
  typeof(RetryHandlerOption).ToString(),
  new RetryHandlerOption()
  {
    MaxRetry = 3
  });

var response = await httpClient.SendAsync(requestMessage);
```

Request/Response vs Content

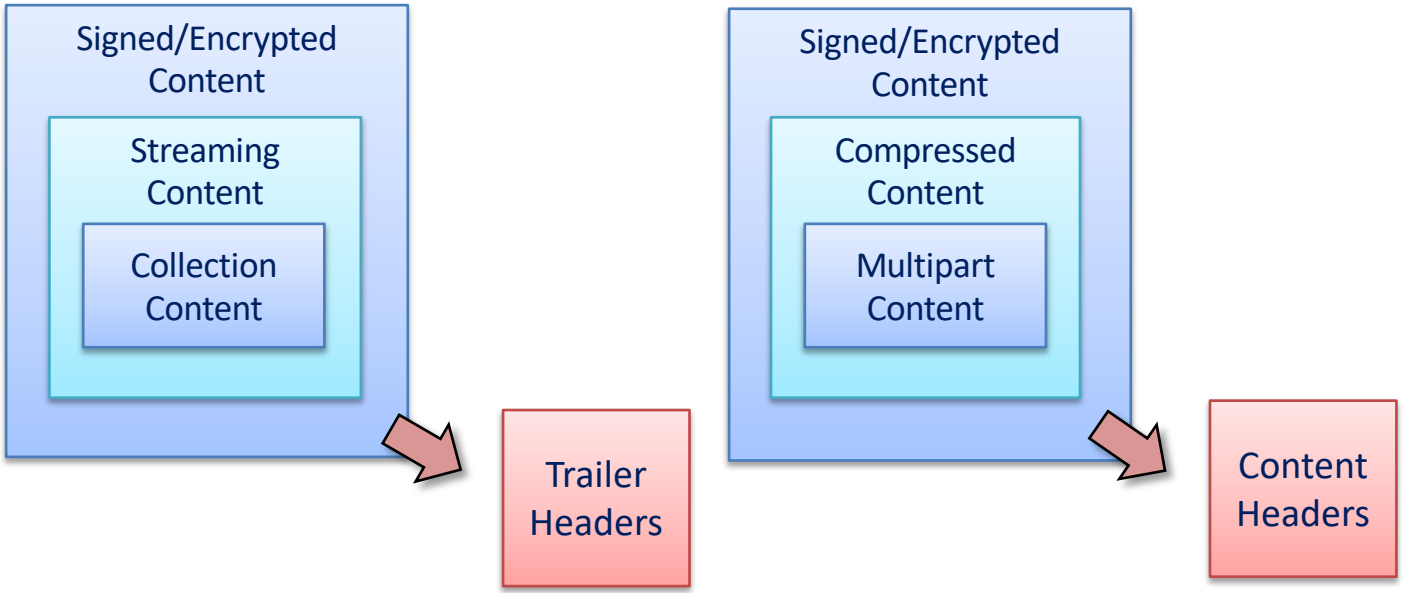
Common Content



Encapsulated/Reusable Content

```
IUserRequest userRequest = graphClient.Me.Request();
IUserEventsCollectionRequest eventsRequest = graphClient.Me.Events.Request();

BatchRequestContent batchRequestContent = new BatchRequestContent();

var userRequestId = batchRequestContent.AddBatchRequestStep(userRequest);
var eventsRequestId = batchRequestContent.AddBatchRequestStep(eventsRequest);

BatchResponseContent returnedResponse = await
    graphClient.Batch.Request().PostAsync(batchRequestContent);
```

Content is composable


Signed/Encrypted Content
Streaming Content
Collection Content
Trailer Headers
Signed/Encrypted Content
Compressed Content
Multipart Content
Content Headers

Response Handler

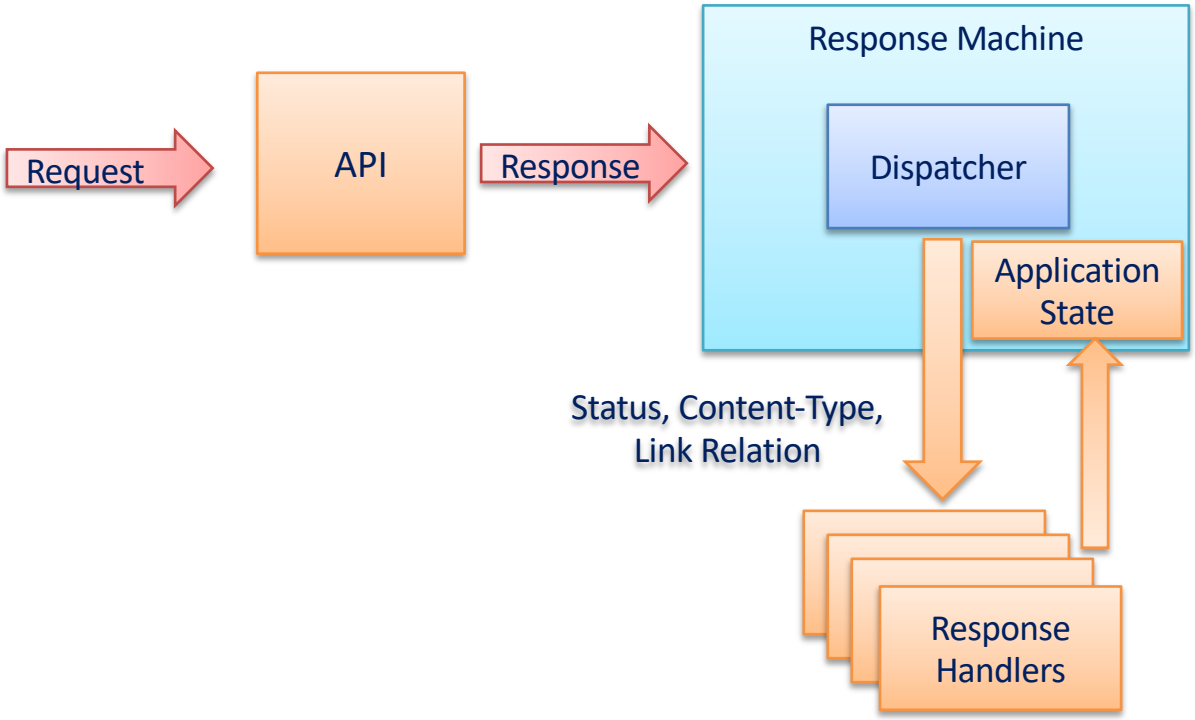
From native and back

```
HttpResponseMessage response = await httpClient.SendAsync(requestMessage);

var handler = new ResponseHandler(new Serializer());

var messagesResponse = await
   handler.HandleResponse<UserMessagesCollectionResponse>(response);
```

Speciality Handlers

```
IUserDeltaCollectionPage userDeltaCollectionPage = await
  graphClient.Users
             .Delta()
             .Request()
             .WithResponseHandler(new DeltaResponseHandler())
             .Select("displayName,givenName,surname")
             .GetAsync();
```

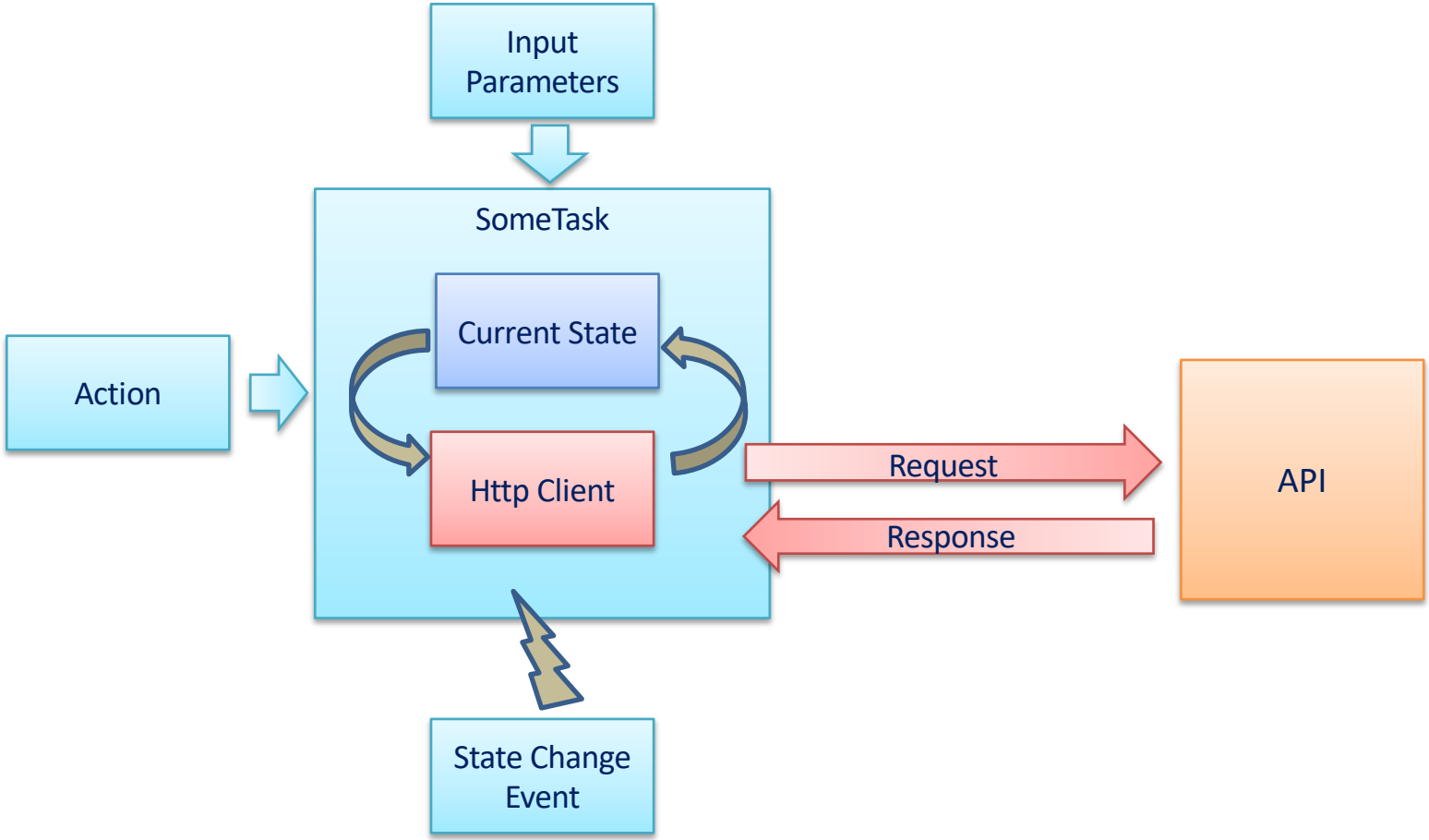
Response Machine



Maybe, one day…

```
ViewModel viewModel = new ViewModel();
viewModel.PropertyChanged += ModelPropertyChanged;

ResponseMachine responseHandler = new ResponseMachine();
responseHandler.OnSuccess<User>(u => viewModel.Me = u);
responseHandler.OnSuccess<Calendar>(cal => viewModel.Calendar = cal);
responseHandler.OnSuccess<Drive>(dr => viewModel.Drive = dr);
responseHandler.OnClientError(e => Console.WriteLine(e.Message));
responseHandler.OnServerError(e => Console.WriteLine(e.Message));

graphClient.Me.Request().SendGet(responseHandler);
graphClient.Me.Calendar.Request().SendGet(responseHandler);
graphClient.Me.Drive.Request().SendGet(responseHandler);
```

Raising the level of abstraction with Tasks


Input Parameters
SomeTask
Current State
Action
Http Client
Request
Response
API
State Change Event

PageIterator Task

```
let somePagedCollection: PageCollection = await client.api("/me/messages").get();

let callback: PageIteratorCallback = (data) => {
     console.log(data);
     return true;
};

let pageIterator = new PageIterator(client, somePagedCollection, callback);

await pageIterator.iterate();
```

Do we have answers?

Why isn’t everyone using the SDK?

Past experiences.
Inflexible options.

What can be done to encourage usage?

Make every part of your SDK optional

Thank you

@darrel_miller

https://github.com/microsoftgraph/msgraph-sdk-design

goto; copenhagen

GOTO Copenhagen 2019

Conference Nov. 18 - 20

Please

Remember to rate this session

Thank you!

Follow us @gotocph


goto;
copenhagen
Follow us @gotocph