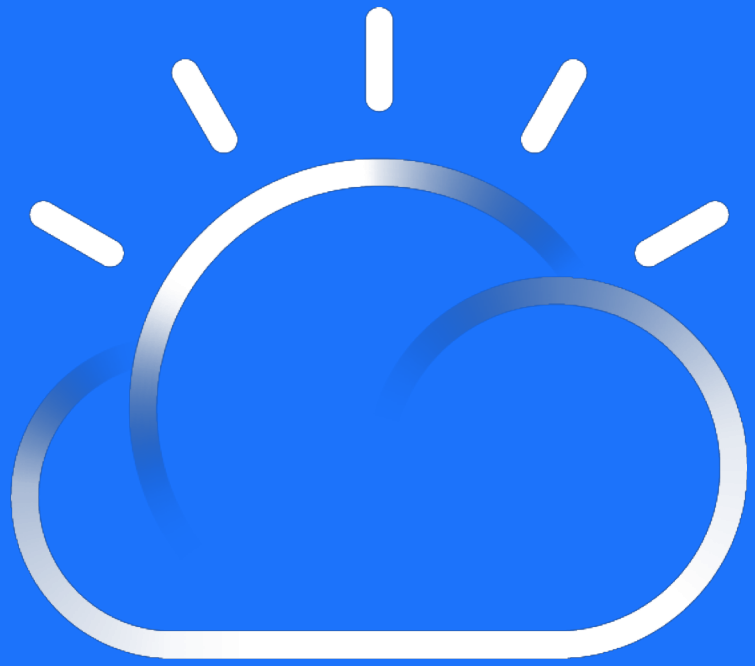
Integration Technical Conference 2019

A12 Application Integration

Ready for REST? Using App Connect with REST APIs

David Coles
dcoles@uk.ibm.com



IBM **Cloud**

IBM

Agenda

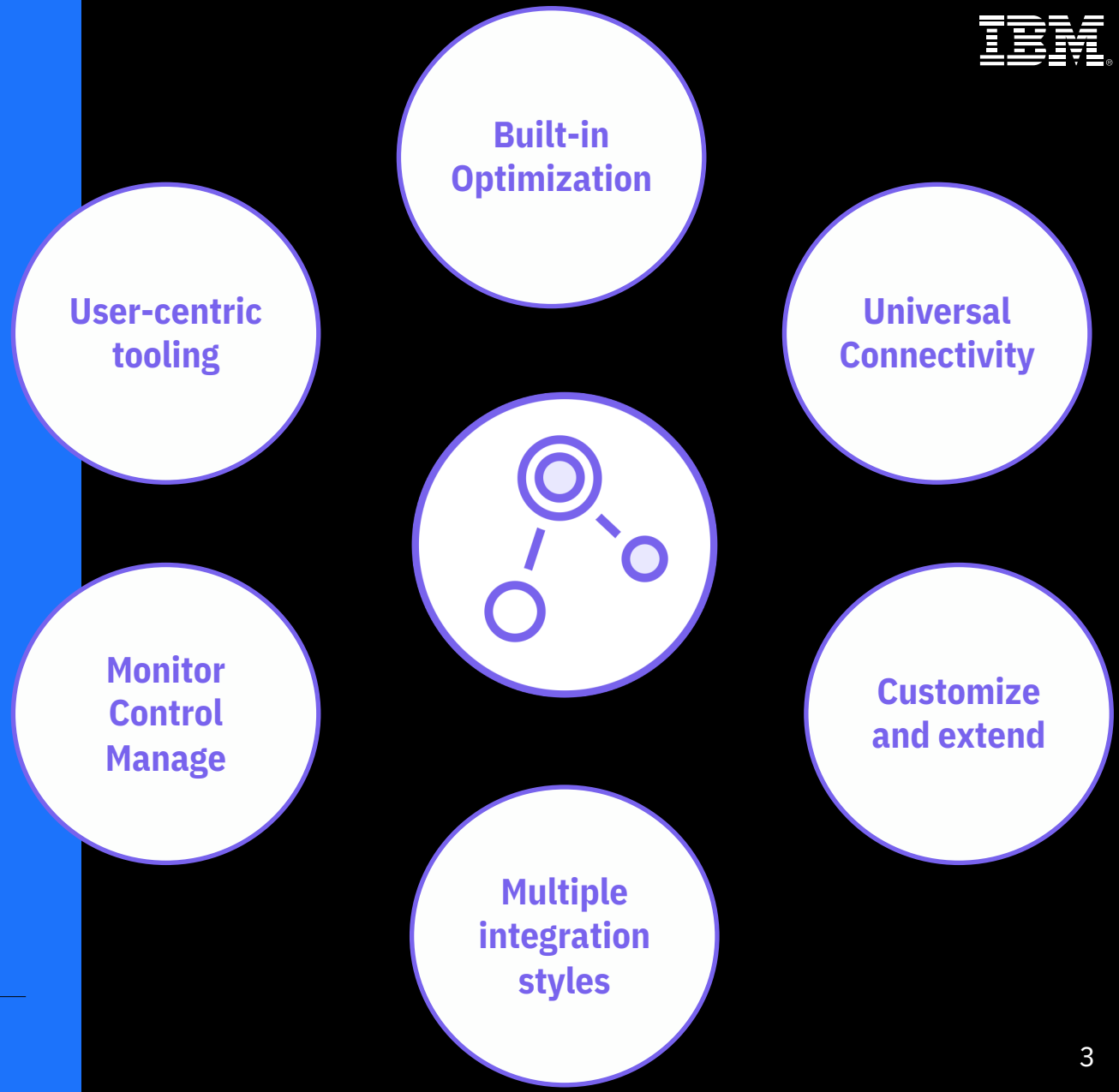
- IBM App Connect
- REST Apis in App Connect Designer
- REST Apis in App Connect Enterprise

IBM App Connect

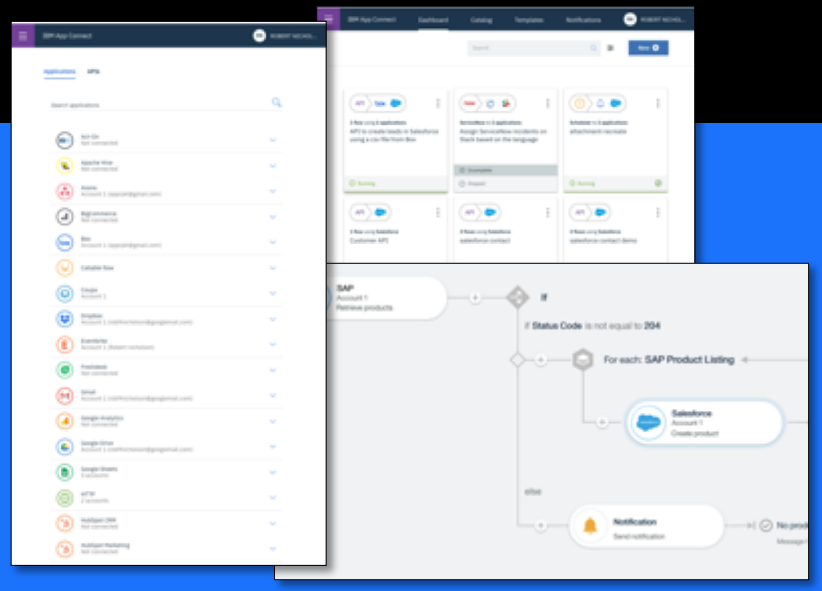
IBM App Connect brings together the best of the existing,

industry-trusted IBM integration technology with new IBM Cloud native technologies

to deliver a platform that supports a breadth of integration needs across a modern digital enterprise



Two complimentary development experiences



The web one It let's you do integration with zero training

The eclipse one for experts to build out the hard stuff

App Connect Designer
High Productivity

- Award winning, browser-based design
- Configuration based, model-driven tooling for rapid outcomes
- Supports a broad range of use cases

1. call shared assets
2. intermingle functions
3. fully blended capabilities

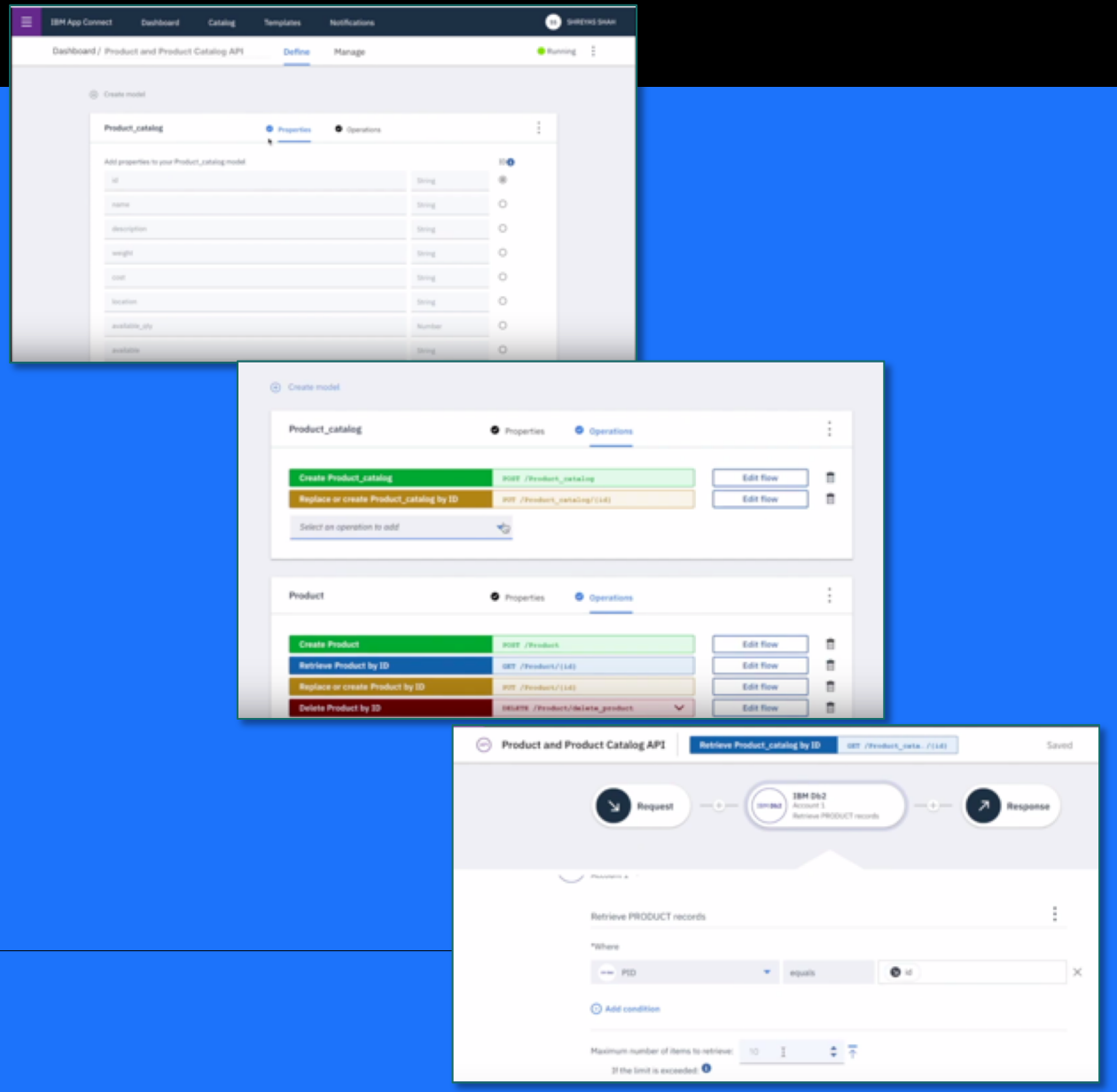
App Connect Toolkit
High Control

- Powerful integration tooling
- Build and manage integrations for any requirement
- Utilize with Designer to enable greater collaboration

REST APIs in App Connect Designer

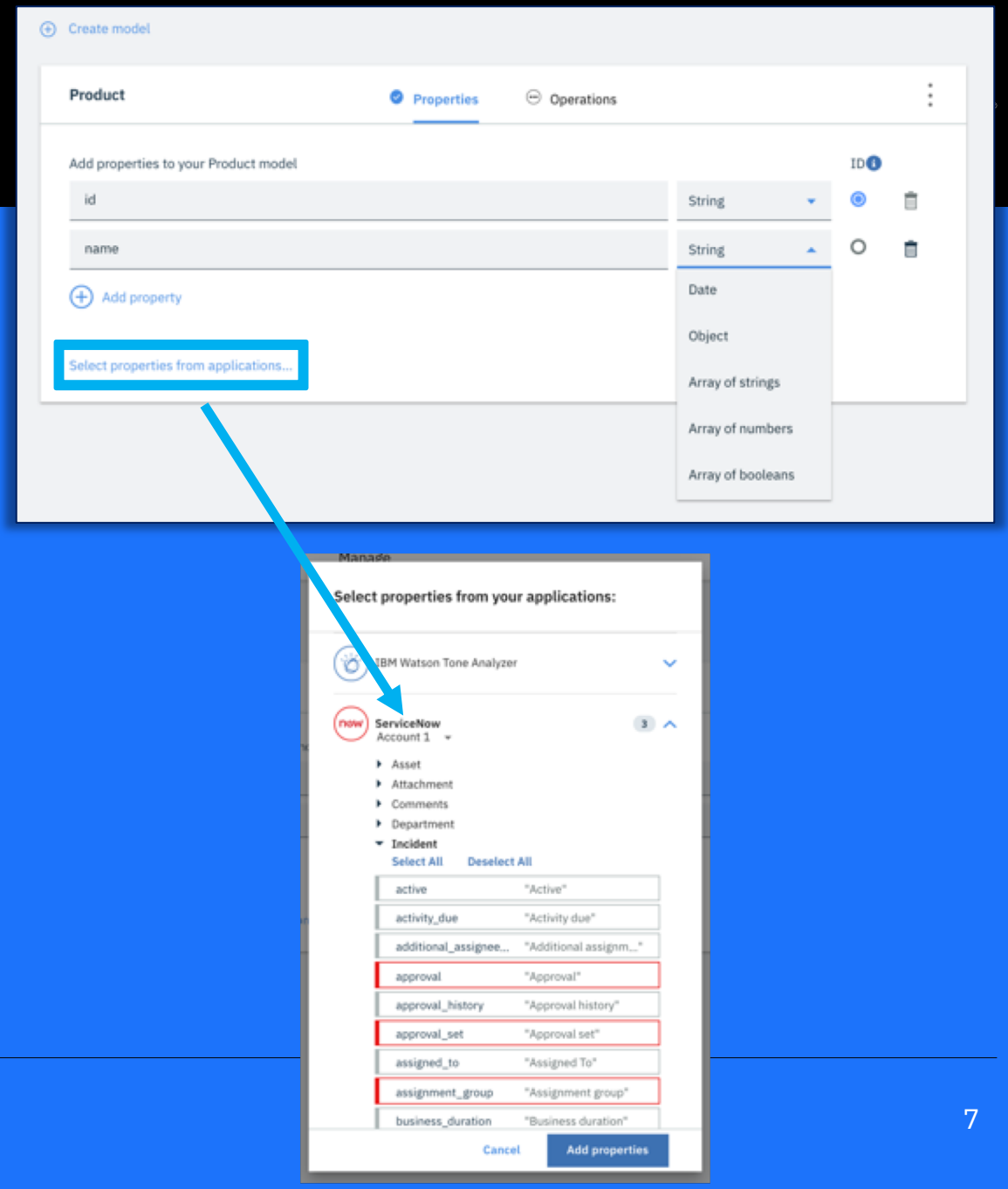
We have implemented it so you don’t have to

- Resource-centric API development
- Guided experience to generate REST semantics
- Auto generate URIs and URLs depending upon selected semantics
- Auto generates industry level OpenAPI v2.0 document
- Import user-defined OpenAPI docs and schemas
- Auto generate request and respond body for your operation



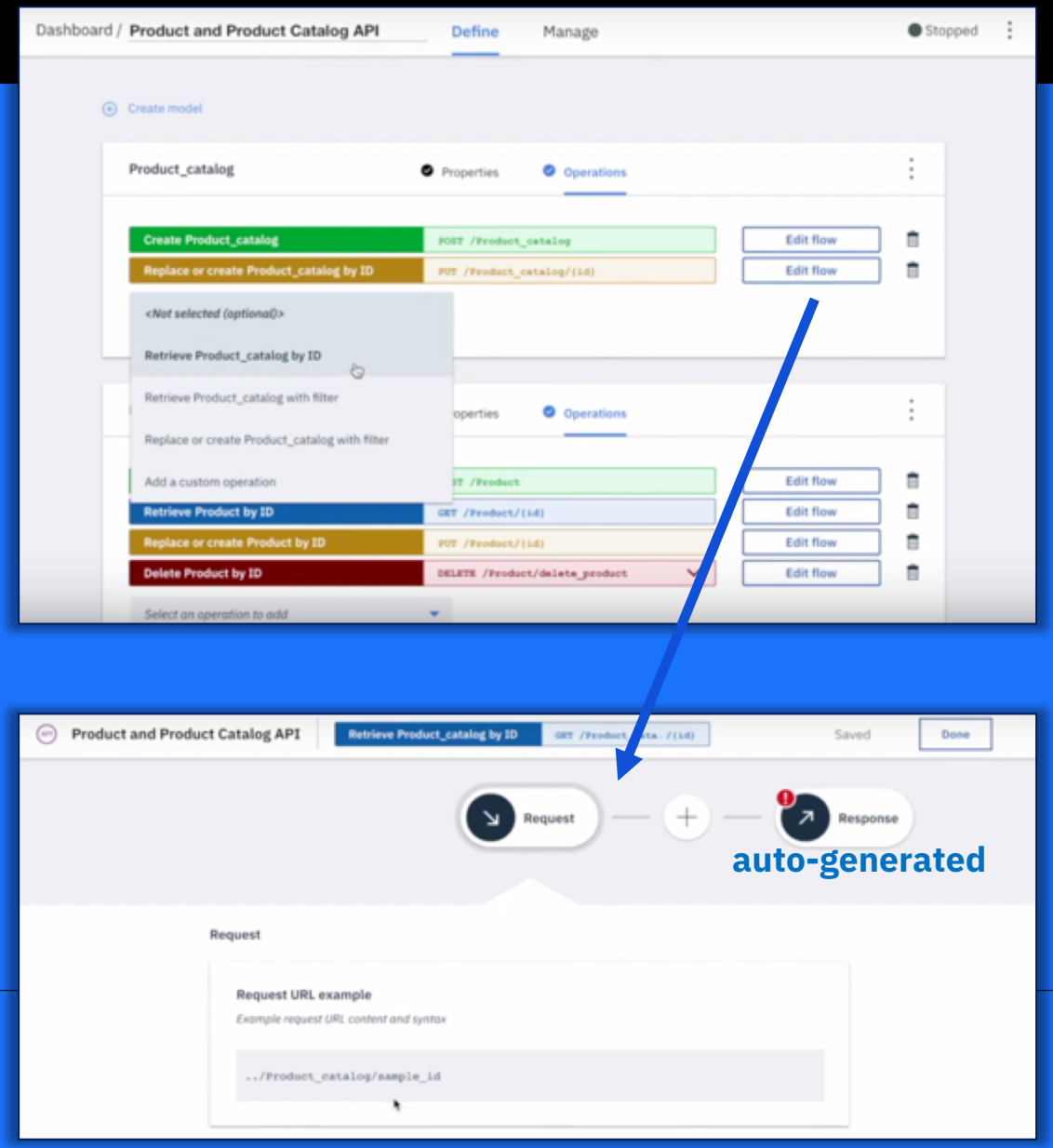
You always start with models

- Models are the resources for your API operations
- Create 1 or more models per API
  - Build model using a form based UI builder
  - or import properties from your business app
  - Model properties can be any of simple or complex data type
    - string, boolean, date, number, object, array of simple types, array of complex types
- Creating models by importing OpenAPI doc or schema



Define operations for your models

- Choose from business semantics
- Auto implements REST semantics
  - Create - POST
  - Retrieve - GET
  - Replace or Create - PUT
  - Delete - DELETE
- Implement filter parameters into your operations:
  - by ID, by WHERE, pagination, etc.

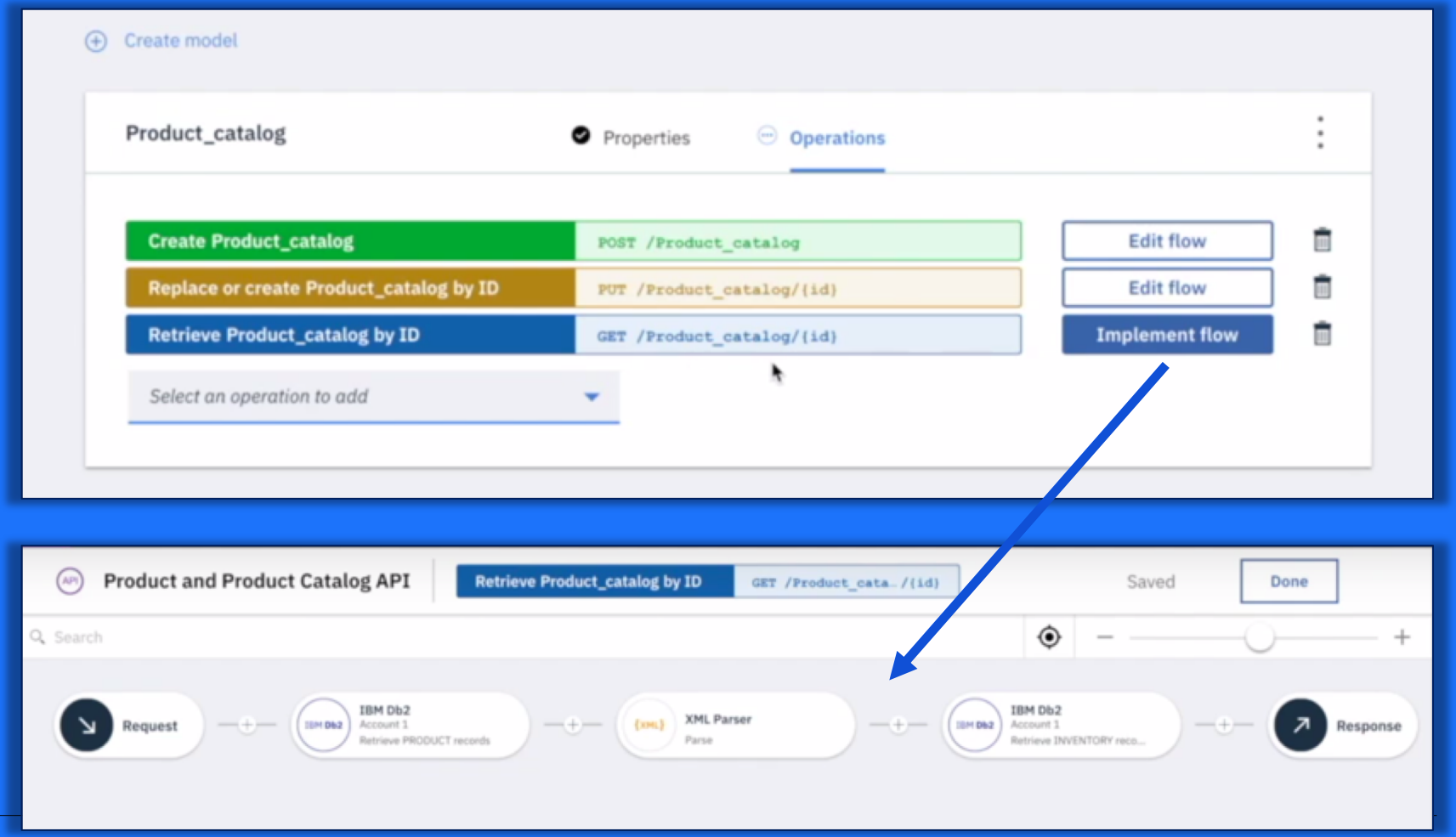


Implement business logic for API operations using flow editor

click, drag and drop configuration
no coding

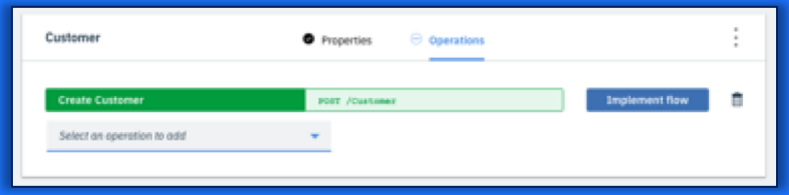
Implement business logic using any of:

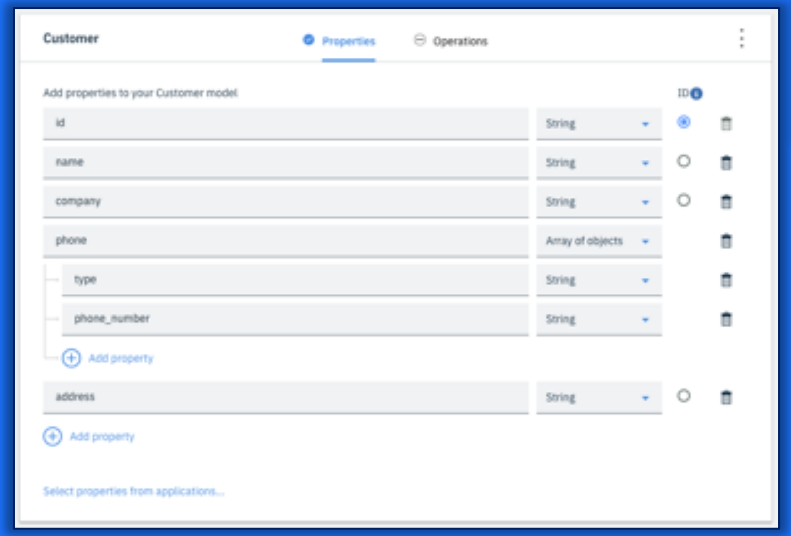
- connectors
- flow constructs
- mapper
- data transformation

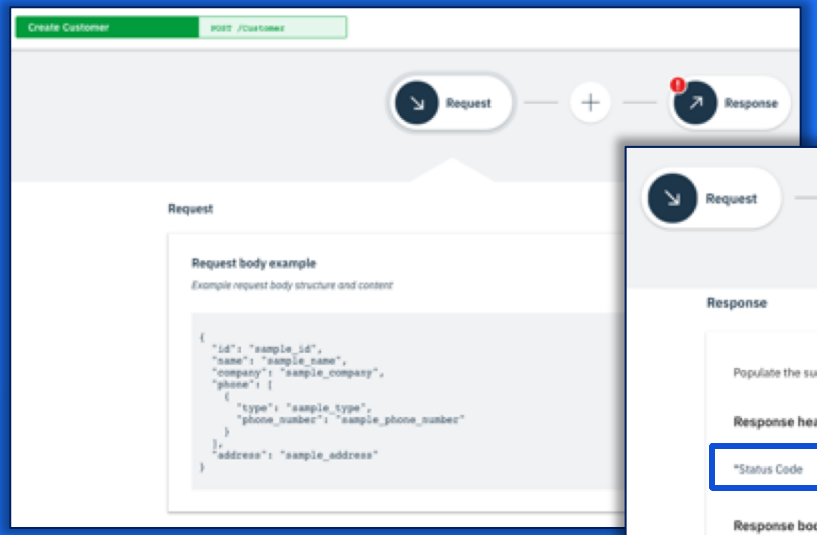


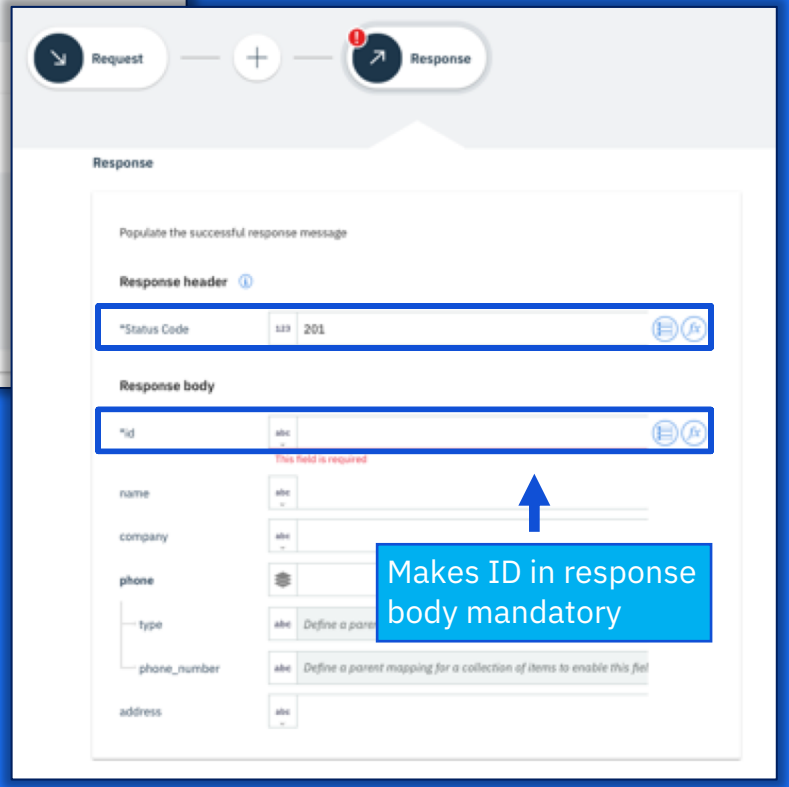
POST

generate a sample request body
auto-generate response body, status code
offer guidance to support best practices



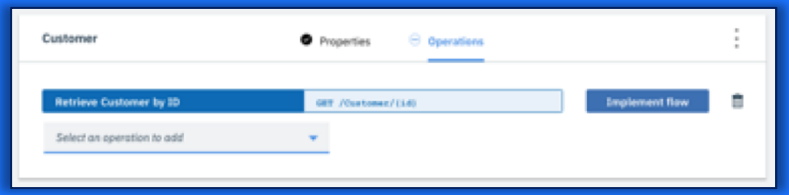


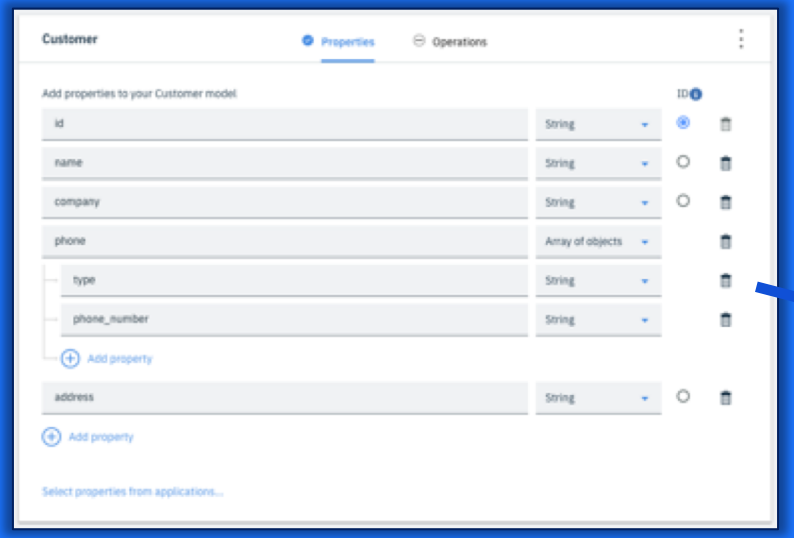


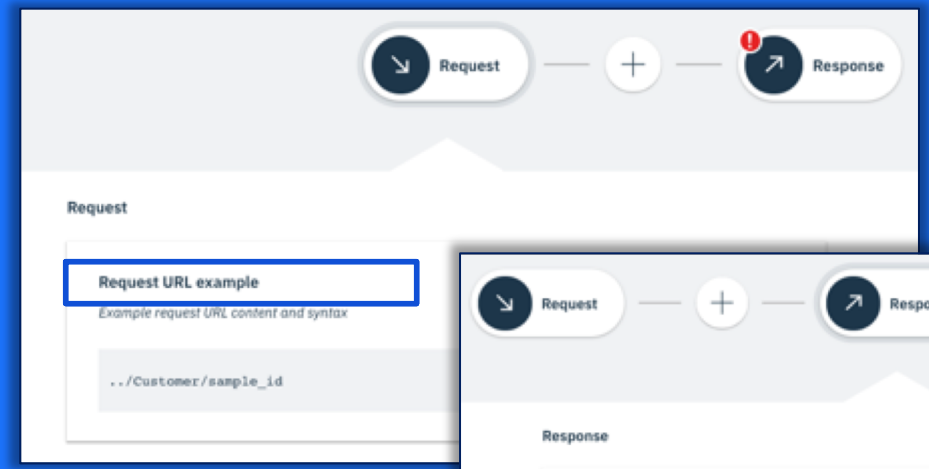


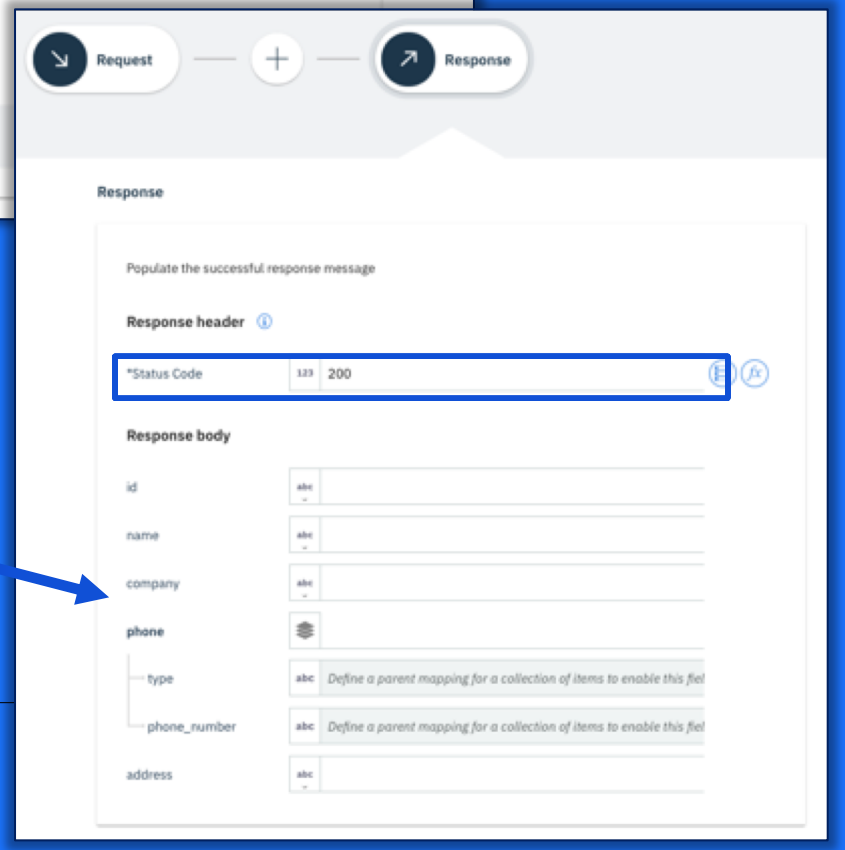
GET by ID

auto generate REST response URL
auto generate response body derived from model



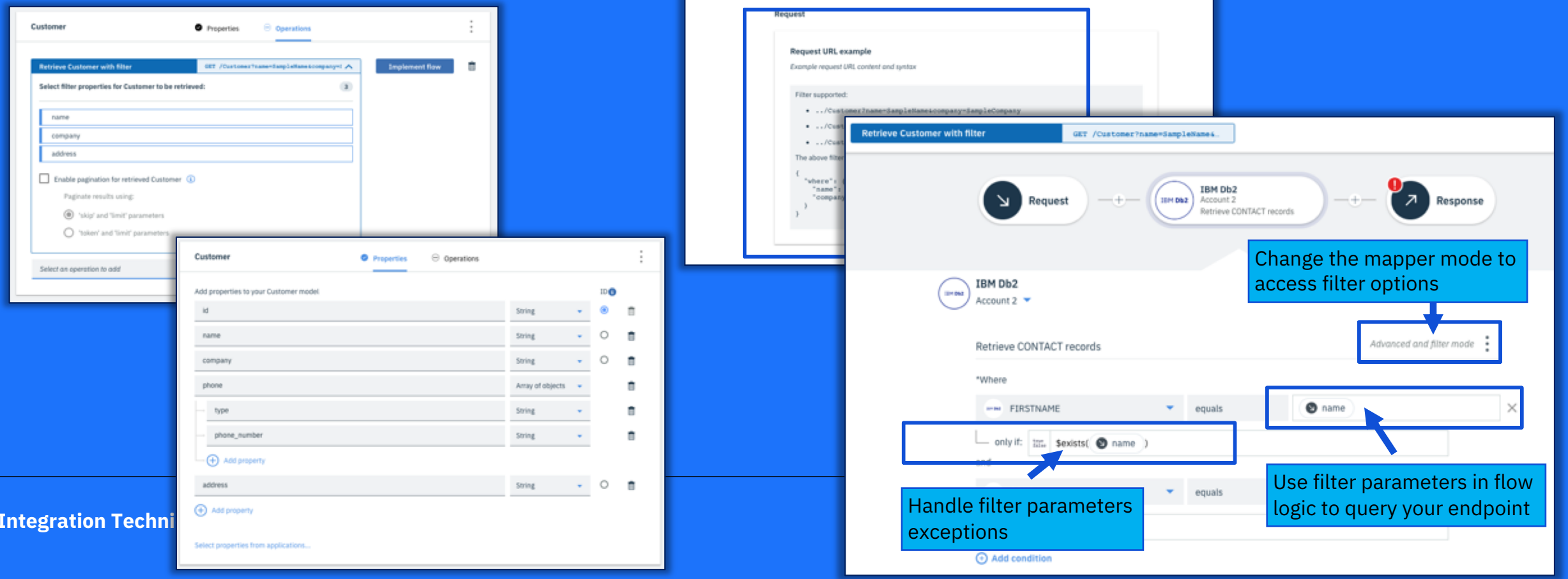






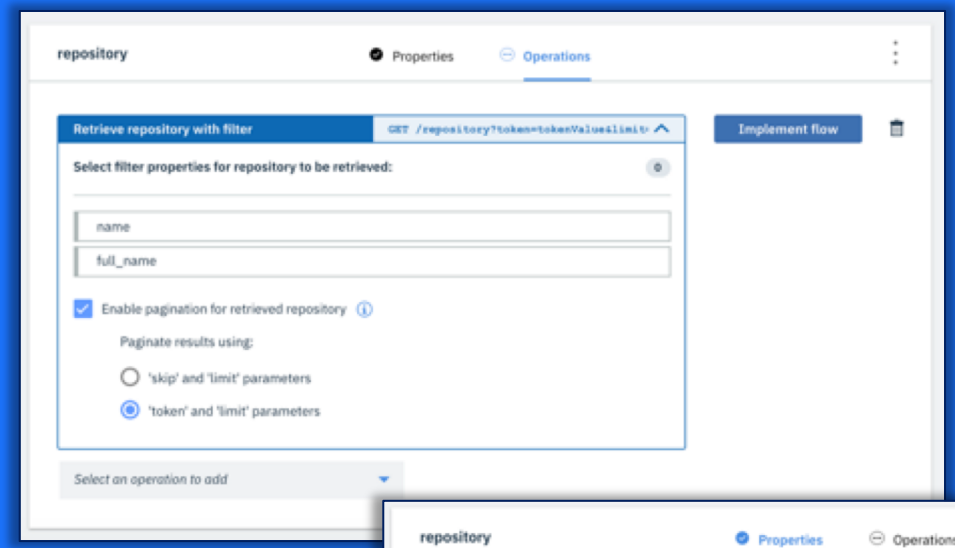
GET by WHERE

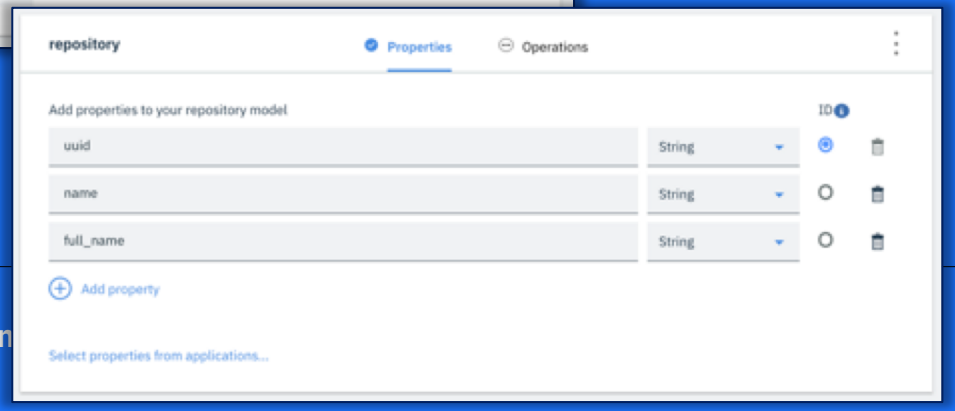
allows to pick non-ID fields as query parameters
auto-generates request URL and syntax
handle filter parameters within flow logic

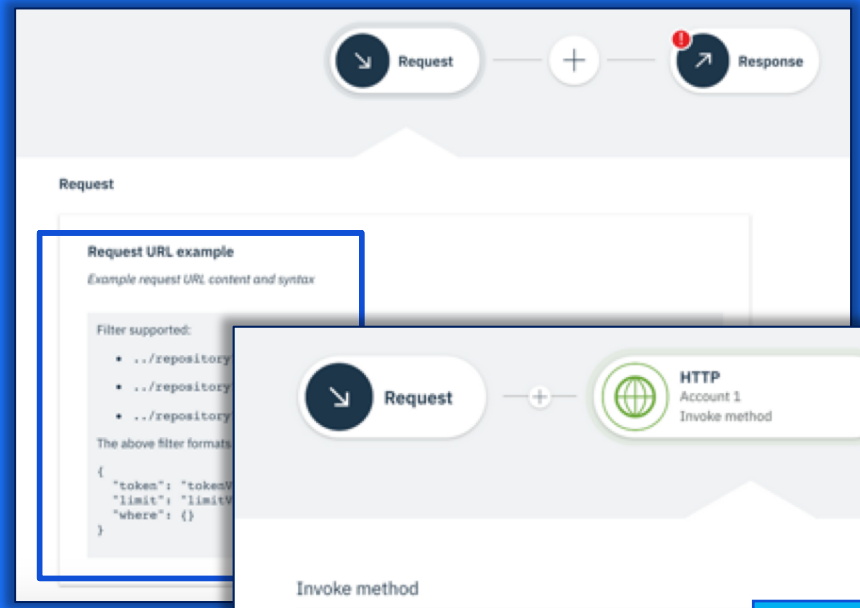


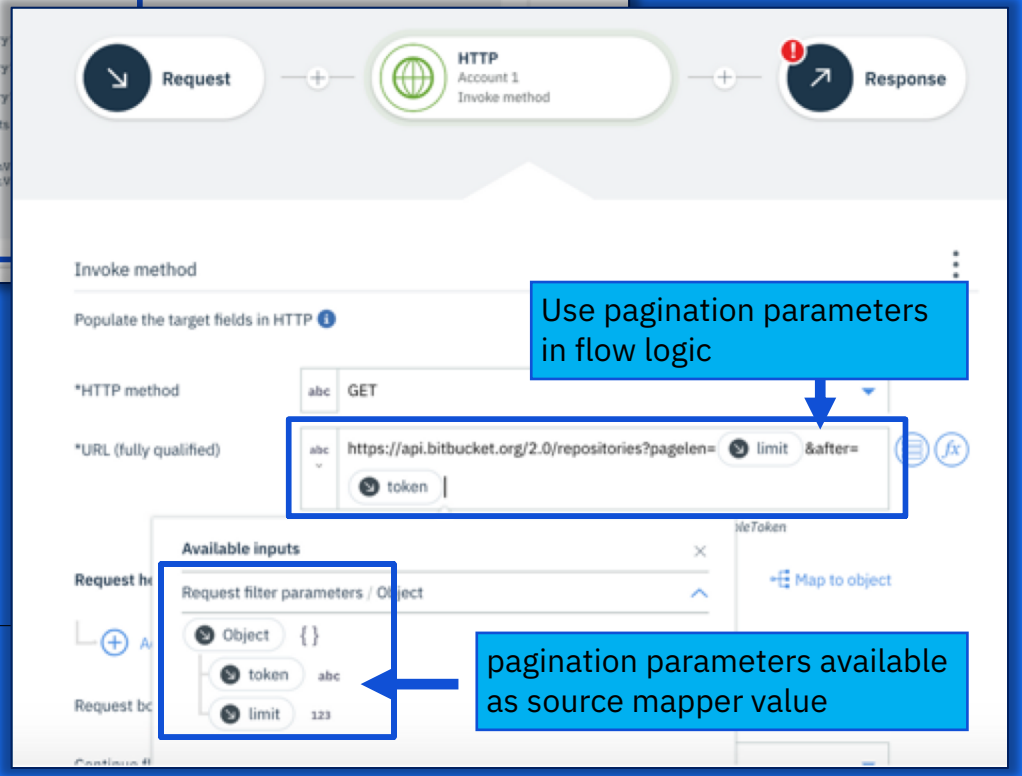
GET with pagination

supports skip, limit and token
auto-generate query parameter, URL and REST syntax
handle pagination parameters in flow logic



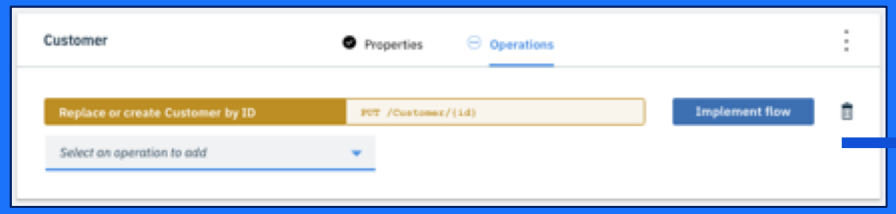


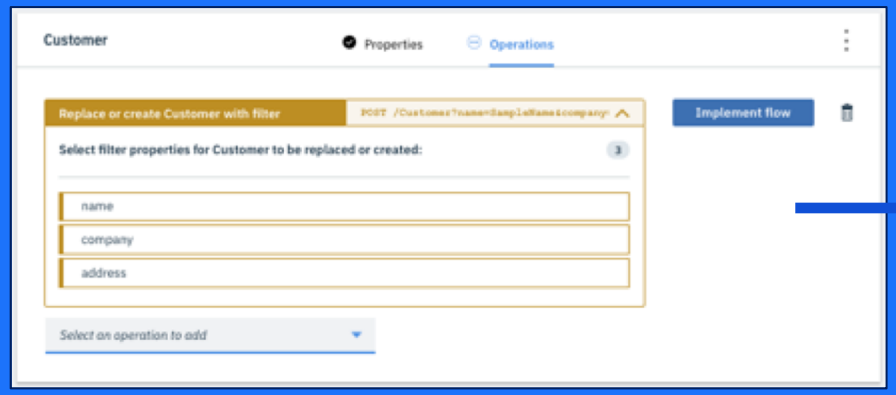


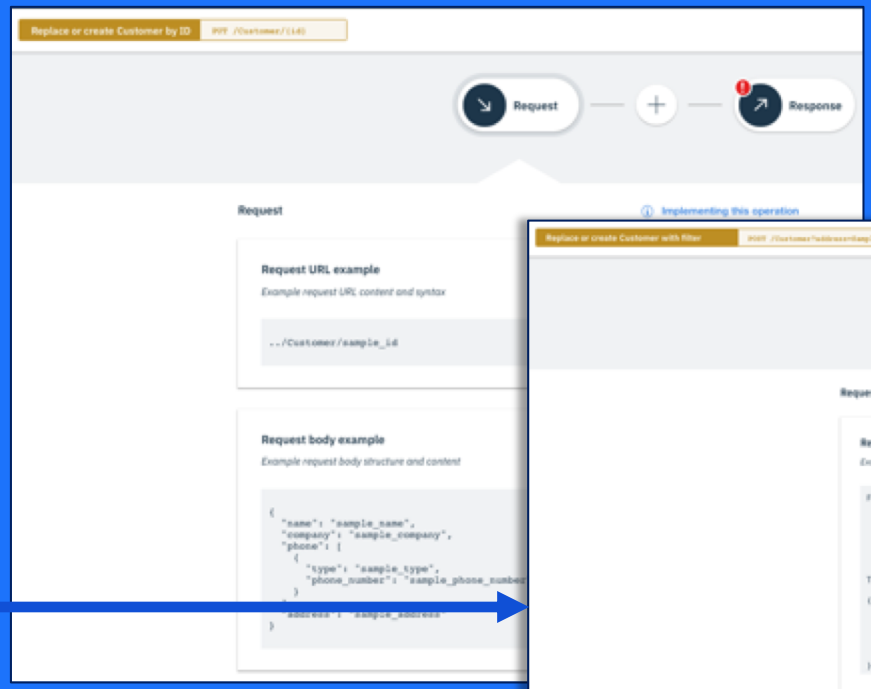


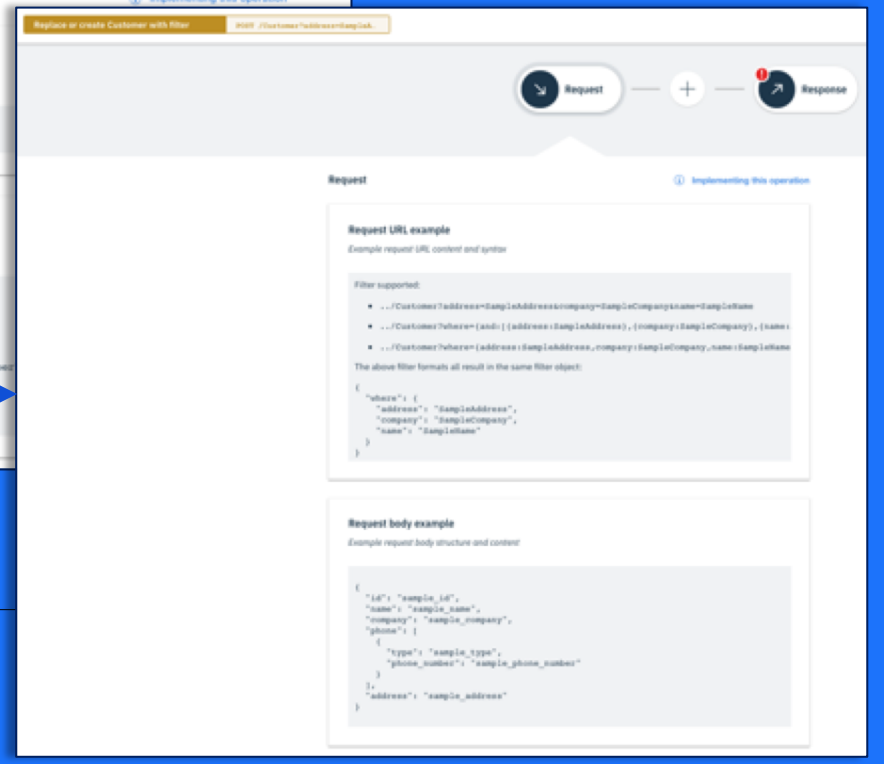
PUT

supports by ID, and by WHERE similar to GET operation
PUT will strictly behave as REPLACE, if you miss out on mapping/provide value for an optional field, it will be replaced with null
Quick tip: make all fields in model as mandatory



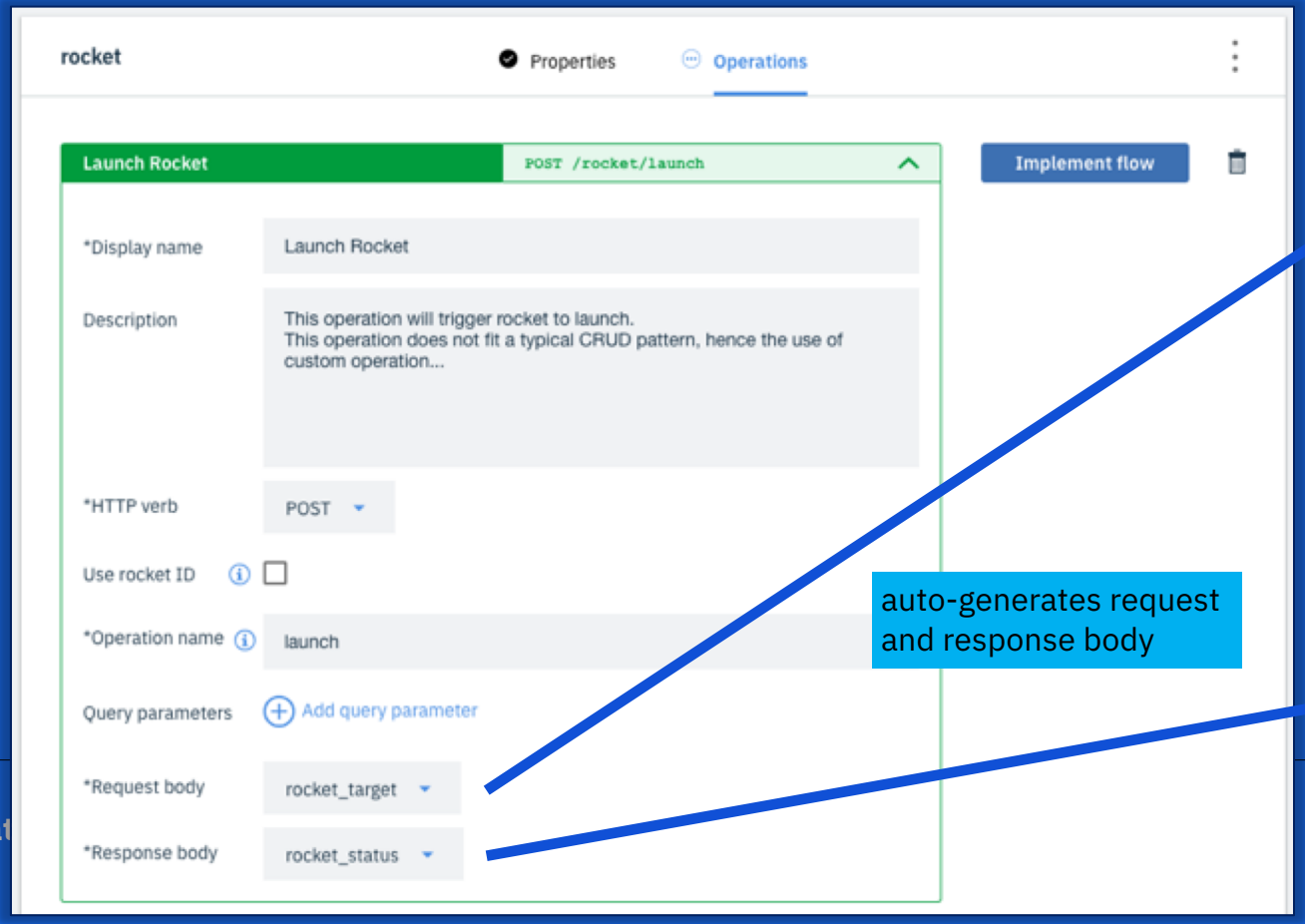




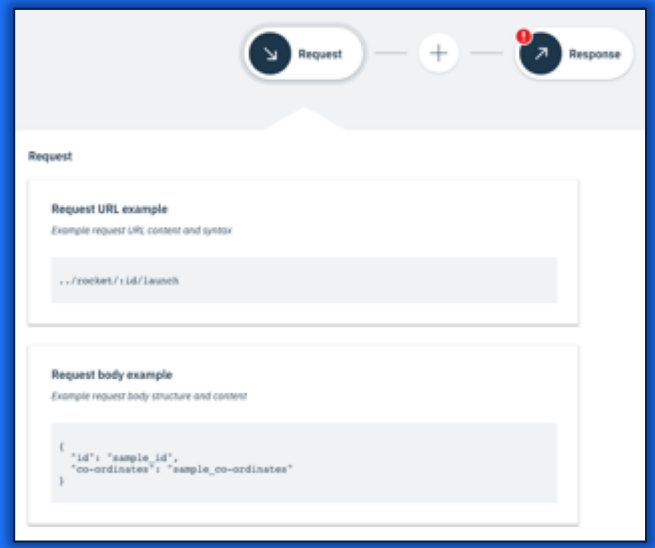


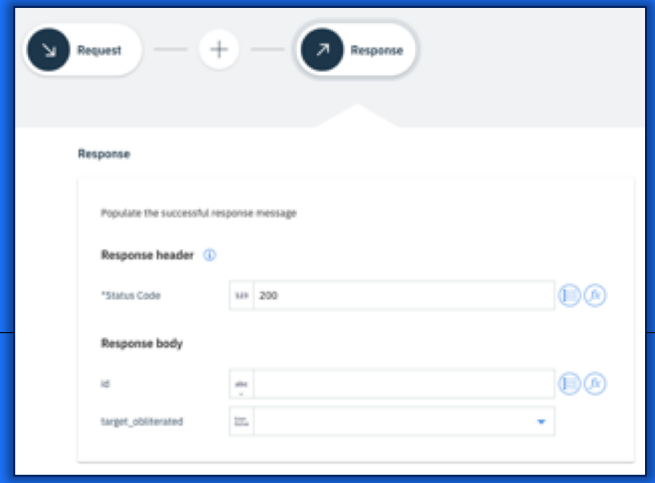
Build custom operations, because:

your operation does not fit CRUD
you do not agree with IBM convention
or you are feeling cowboy-ish!!



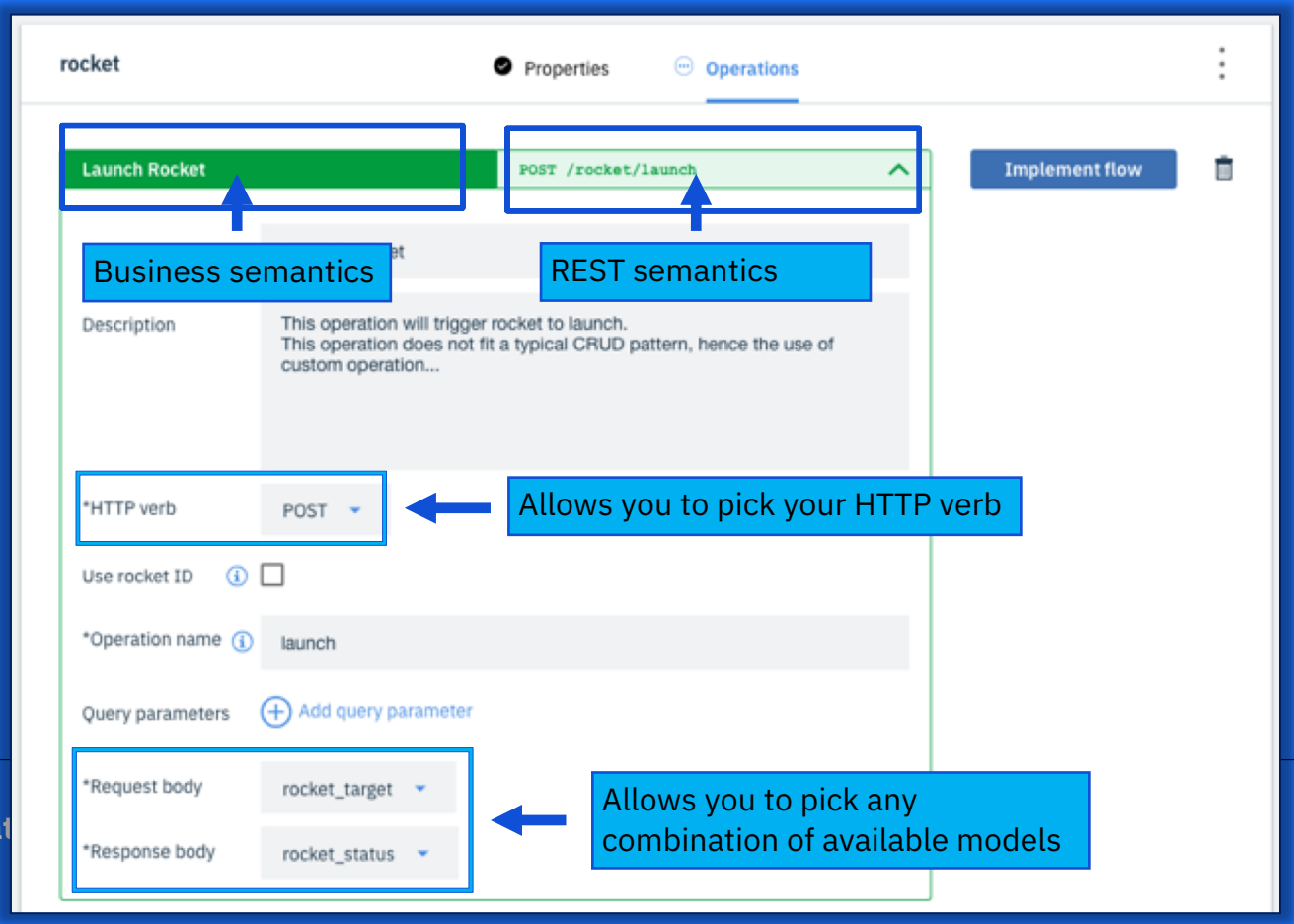
auto-generates request and response body

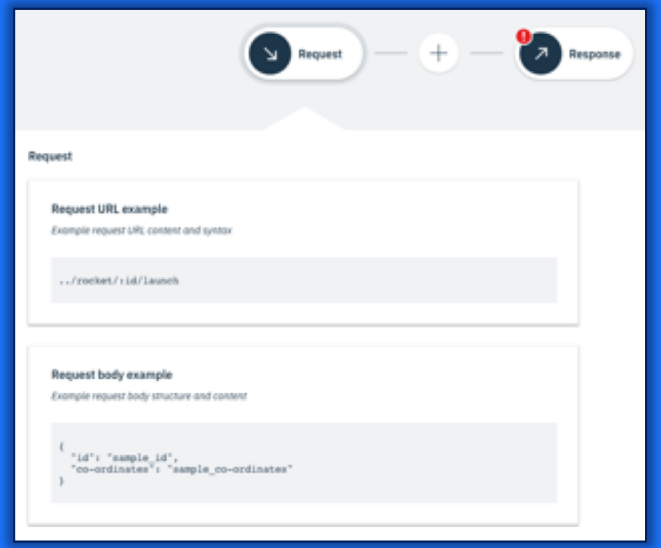


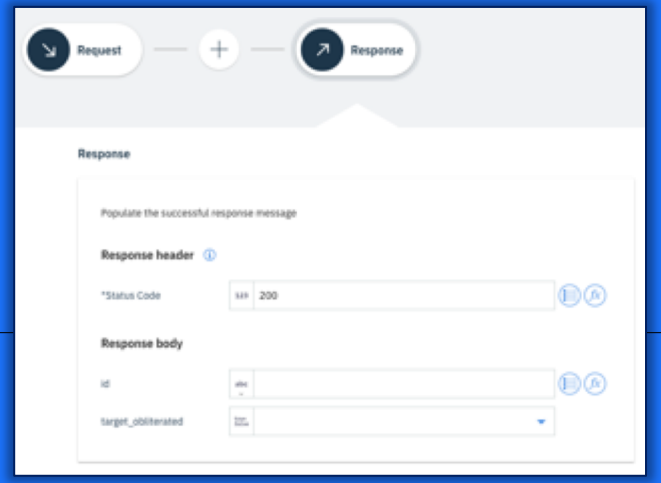


Build custom operations, because:

your operation does not fit CRUD
you do not agree with IBM convention
or you are feeling cowboy-ish!!

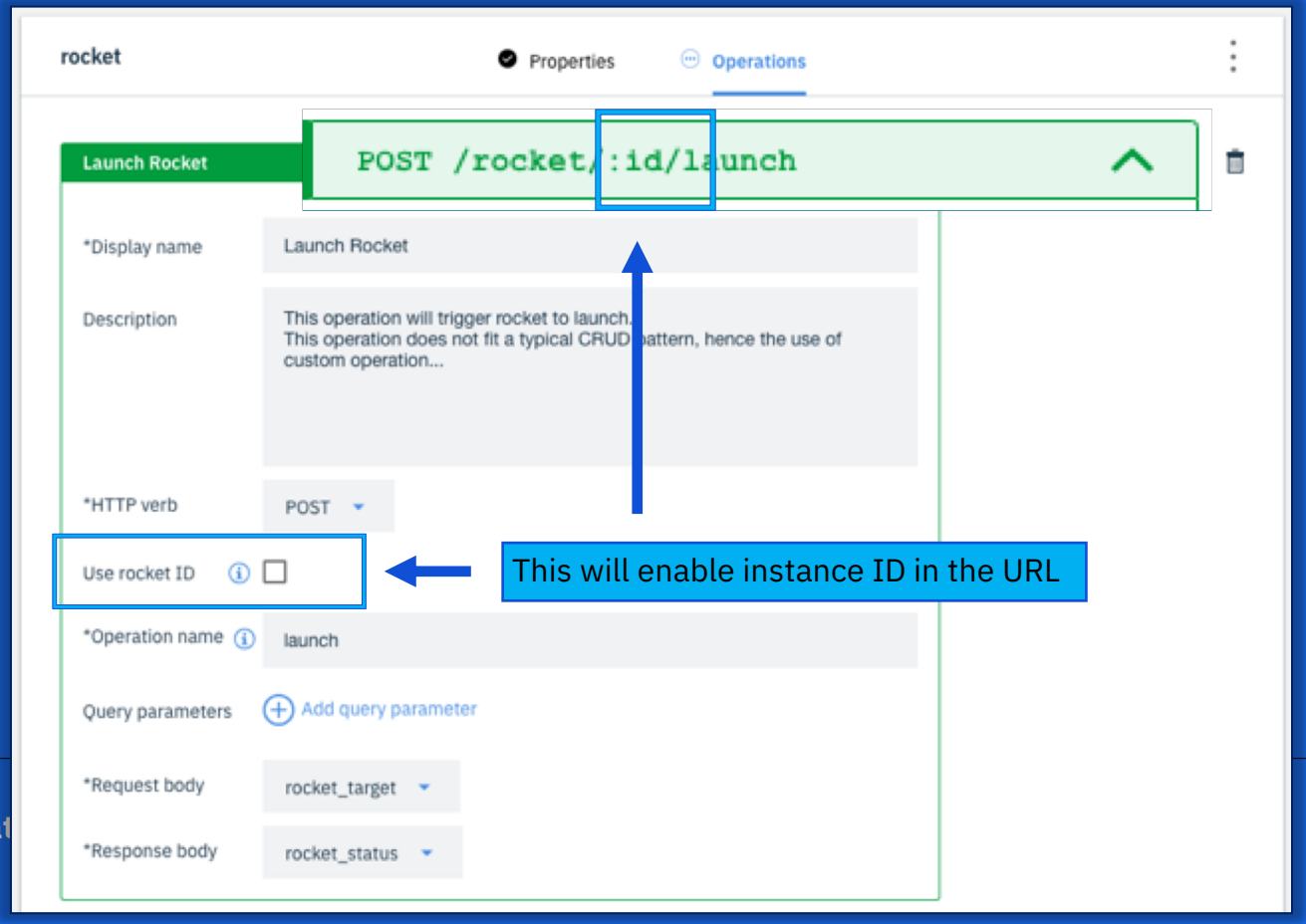


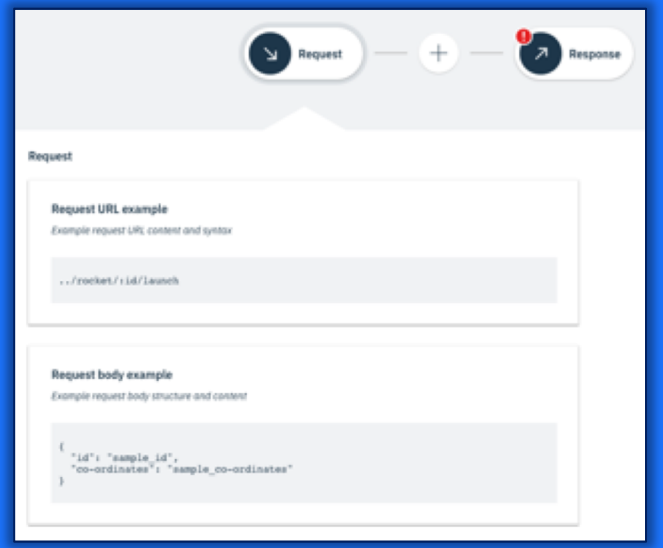


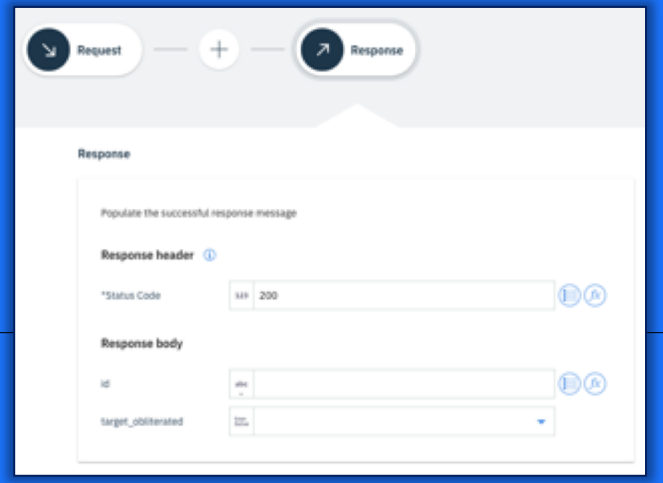


Build custom operations, because:

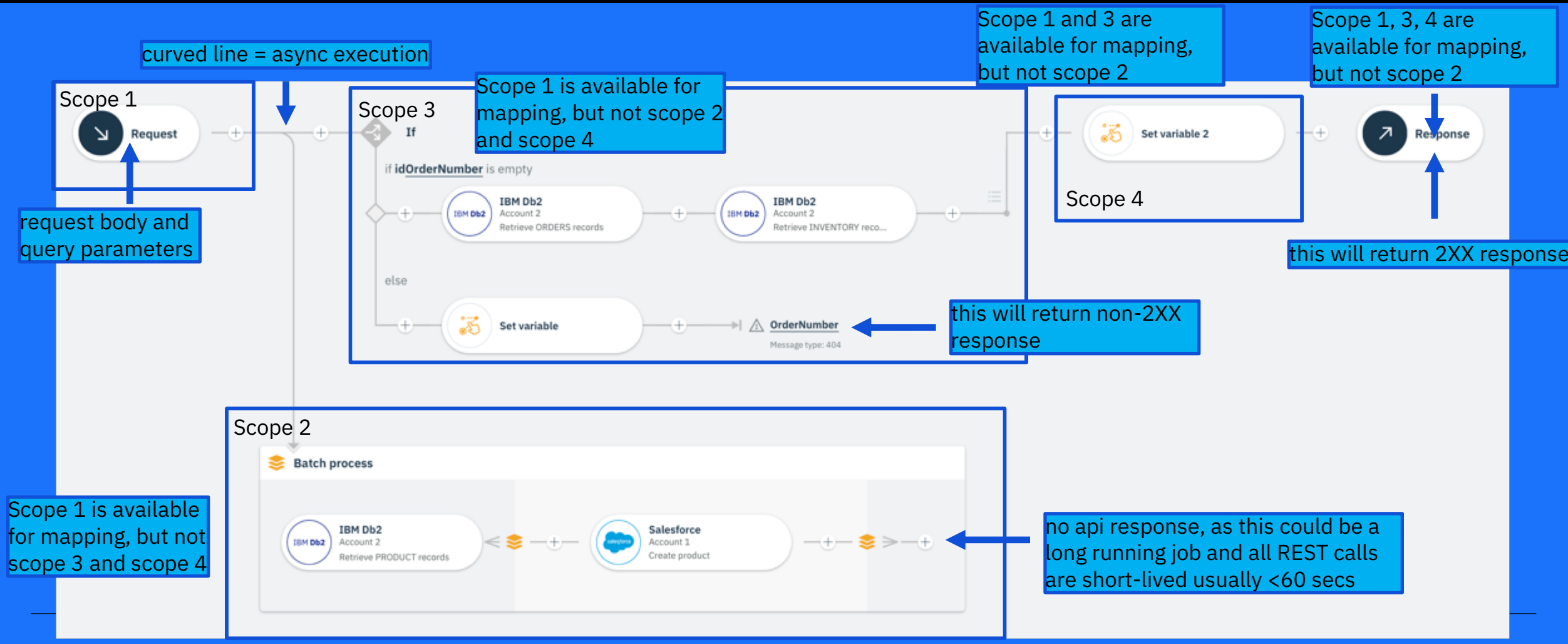
your operation does not fit CRUD
you do not agree with IBM convention
or you are feeling cowboy-ish!!



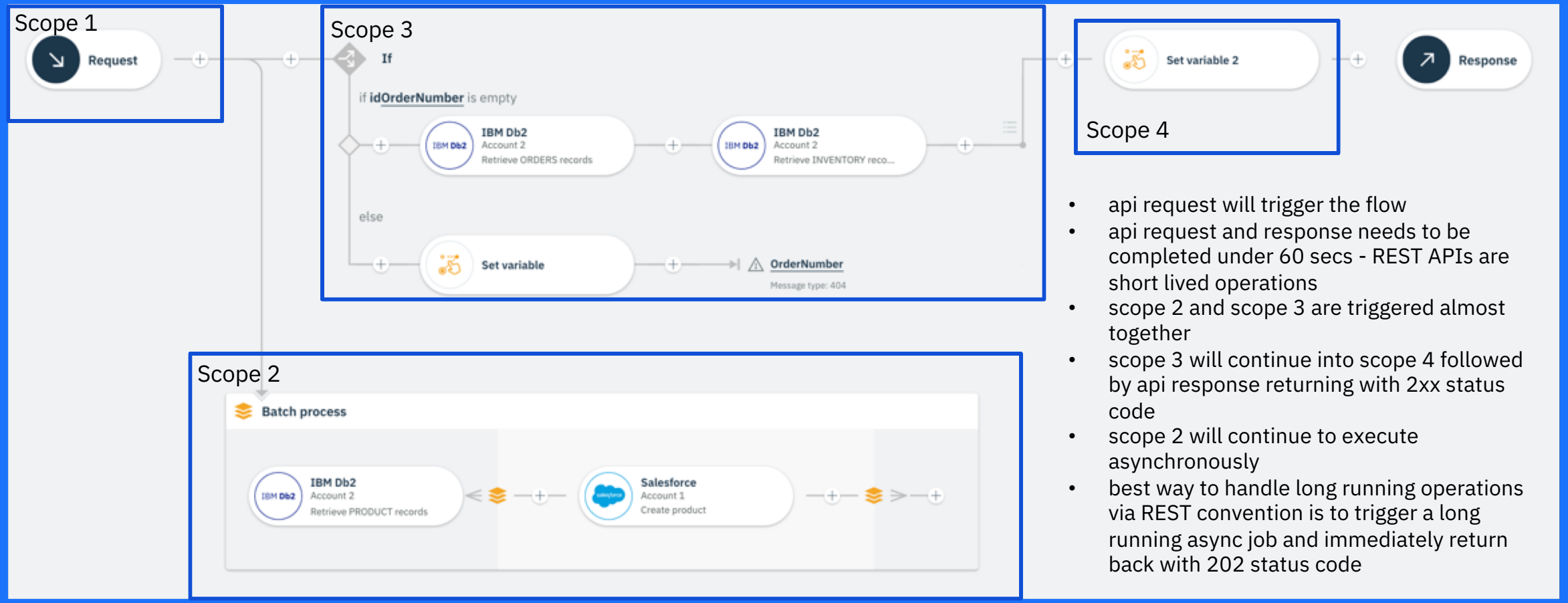




Understanding the mechanics of API flow design scope of references



Understanding the mechanics of API flow design sequence of execution



- api request will trigger the flow
- api request and response needs to be completed under 60 secs - REST APIs are short lived operations
- scope 2 and scope 3 are triggered almost together
- scope 3 will continue into scope 4 followed by api response returning with 2xx status code
- scope 2 will continue to execute asynchronously
- best way to handle long running operations via REST convention is to trigger a long running async job and immediately return back with 202 status code

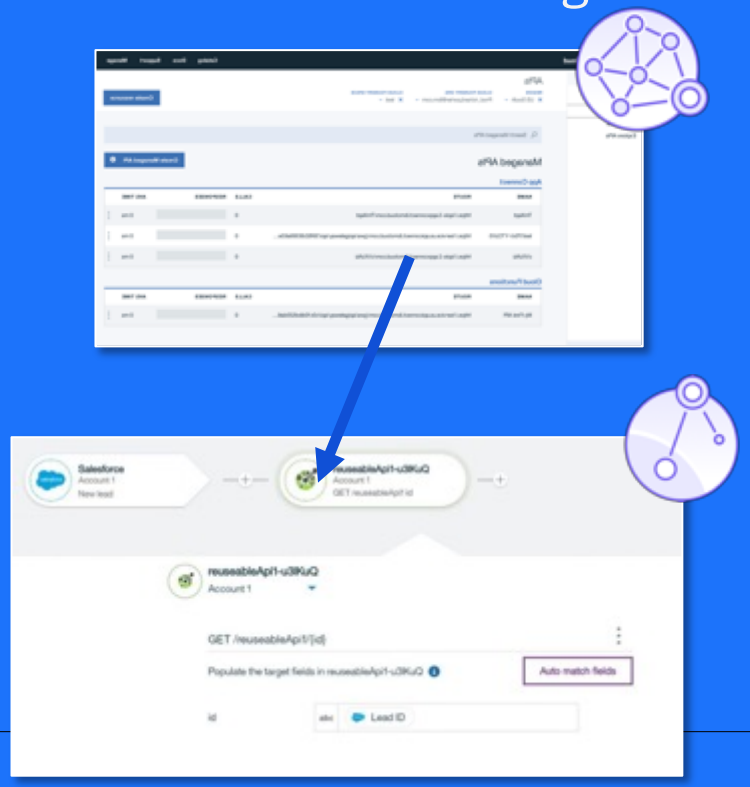
API Connect with REST APIs in App Connect Designer

Seamless integration between App Connect and API Connect

Use App Connect to develop REST APIs / Use API Connect to manage, socialise and monetise APIs

Expose APIs in App Connect manage in API Connect

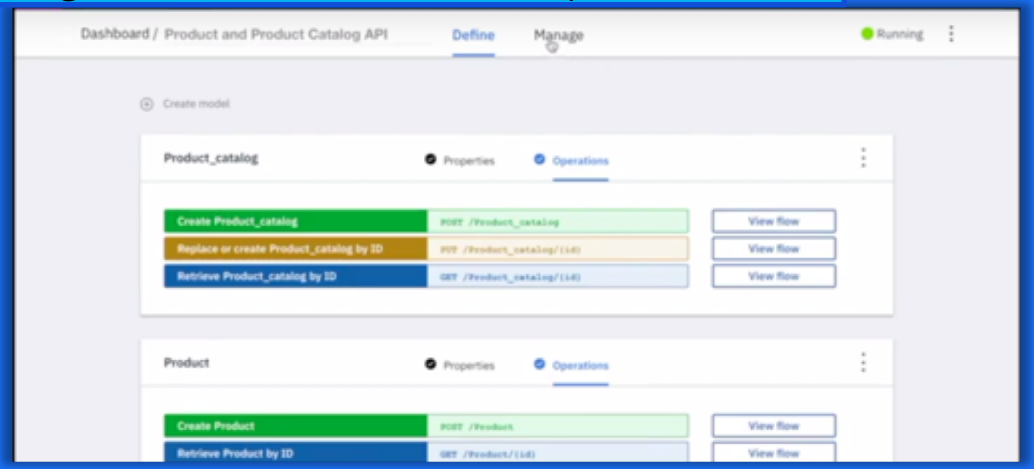
Consume API in App Connect from API Connect catalog



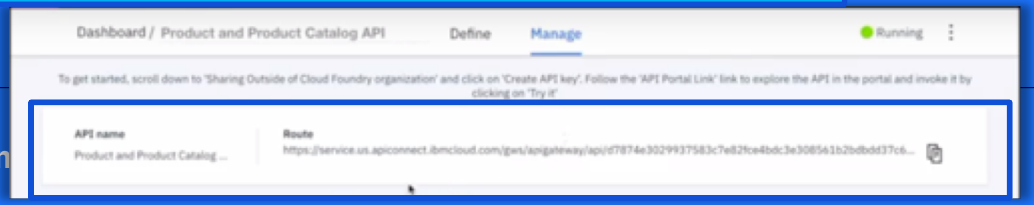
Embedded API management functions

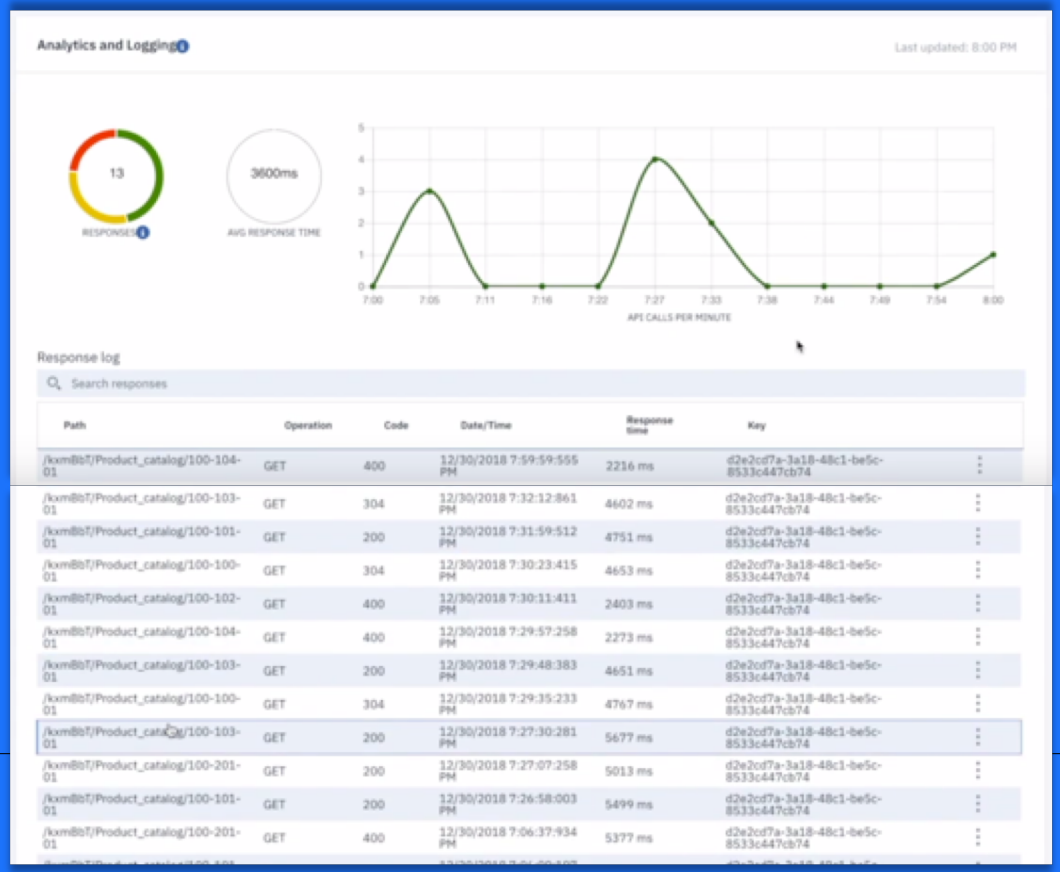
App Connect supports embedded API Connect Lite that offers limited cloud API management functions and capabilities for FREE
cloud API gateway, API security, rate limit, developer portal, analytics and logging

Click on manage to access embedded APIm capabilities



Provides a cloud API gateway
automatically generates API name and API route, also available with small URLs



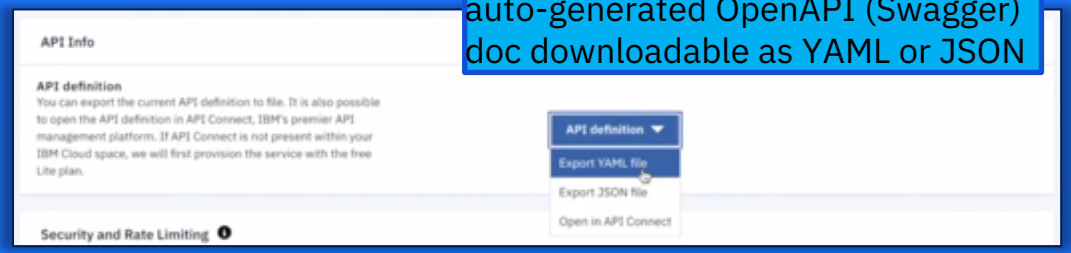


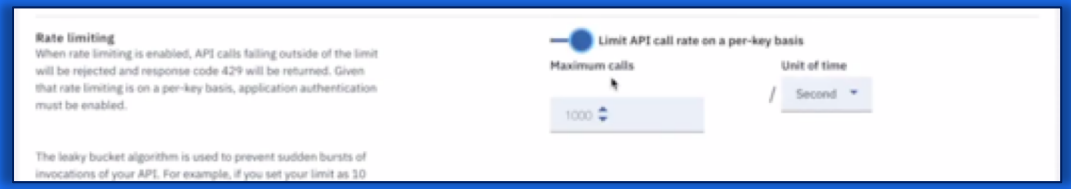
Integration

Embedded API management functions

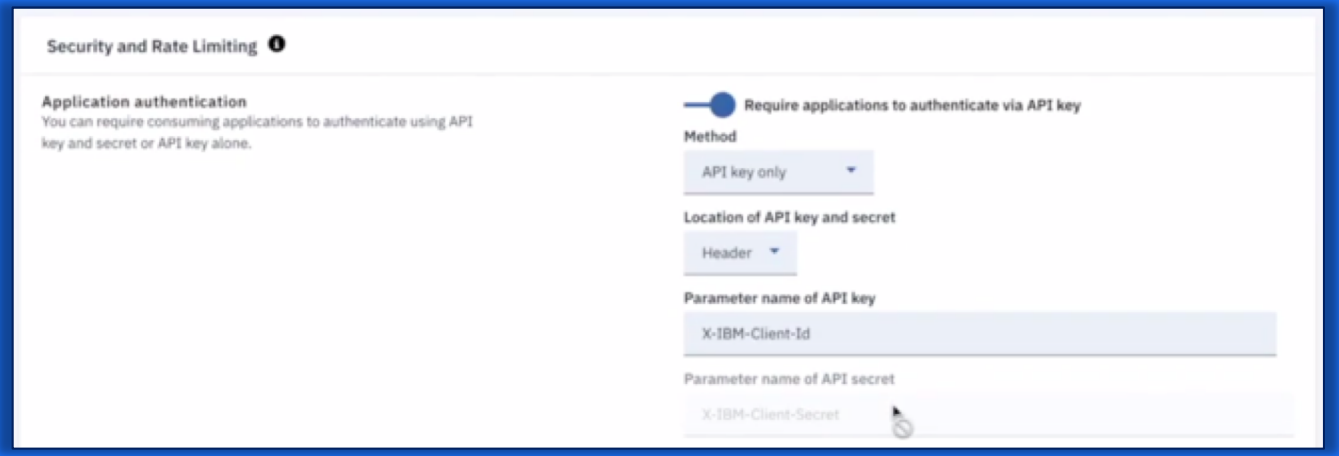
App Connect supports embedded API Connect Lite that offers limited cloud API management functions and capabilities for FREE
cloud API gateway, API security, rate limit, developer portal, analytics and logging

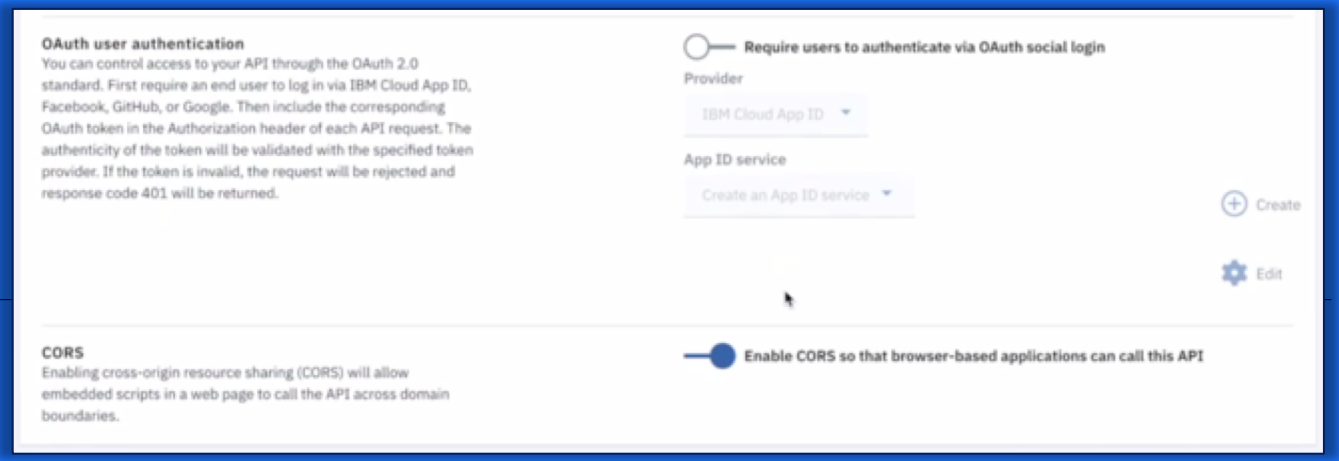
auto-generated OpenAPI (Swagger) doc downloadable as YAML or JSON





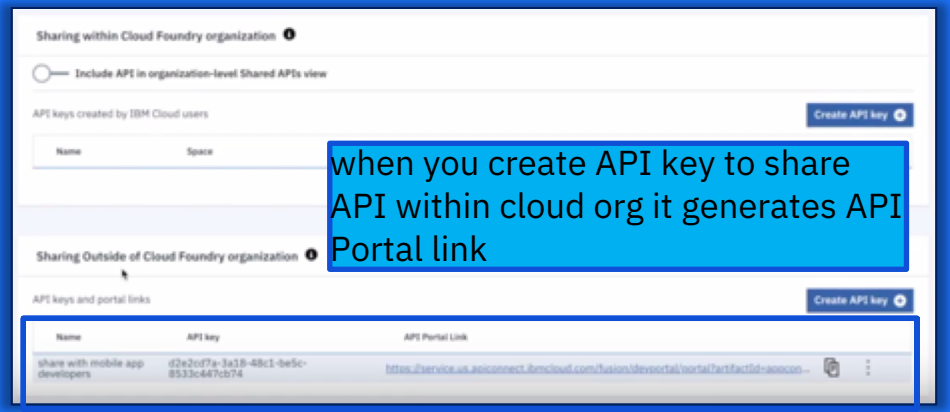
securely publish API within cloud org and outside it



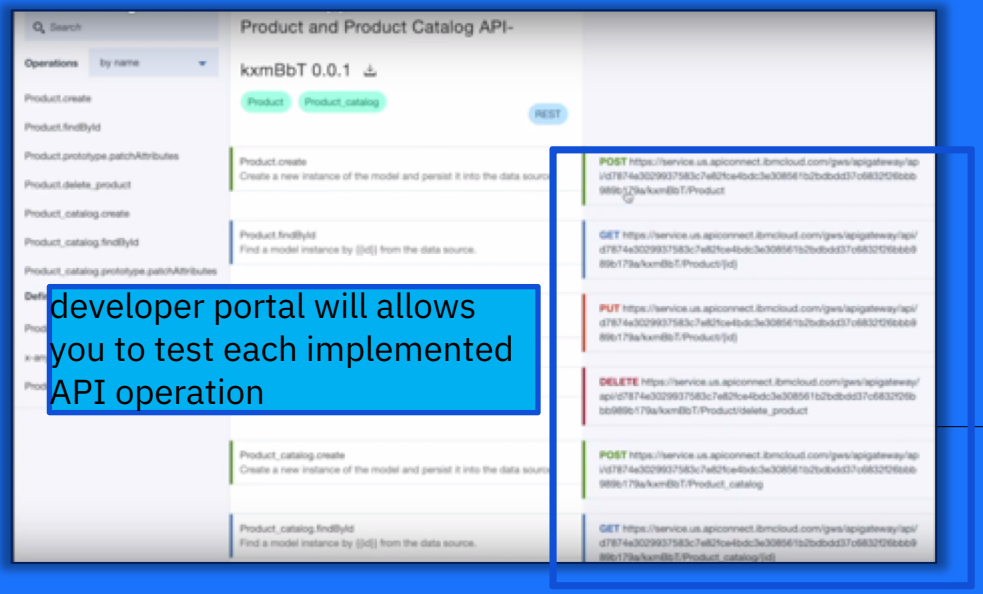


Embedded API management functions

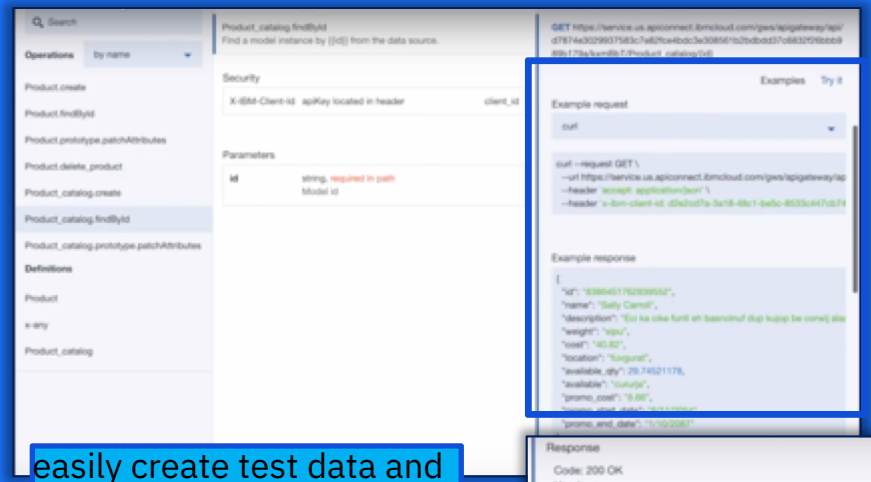
how to use developer portal to test your API



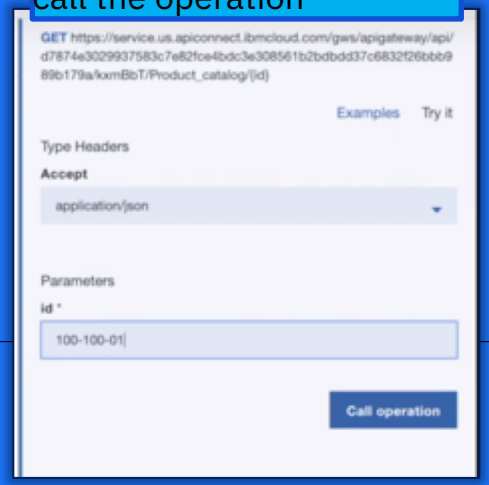
when you create API key to share API within cloud org it generates API Portal link



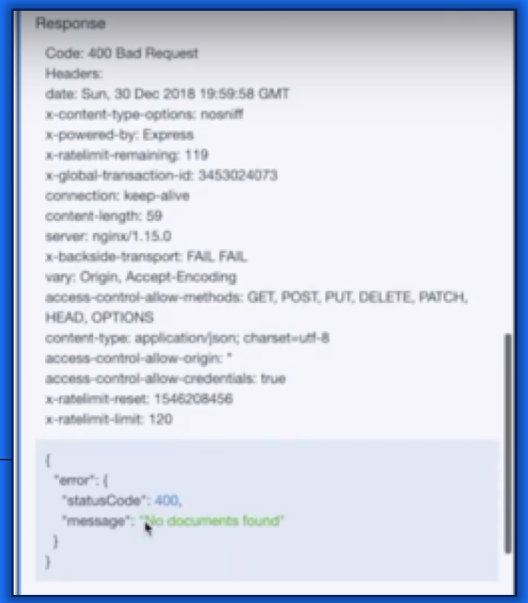
developer portal will allows you to test each implemented API operation



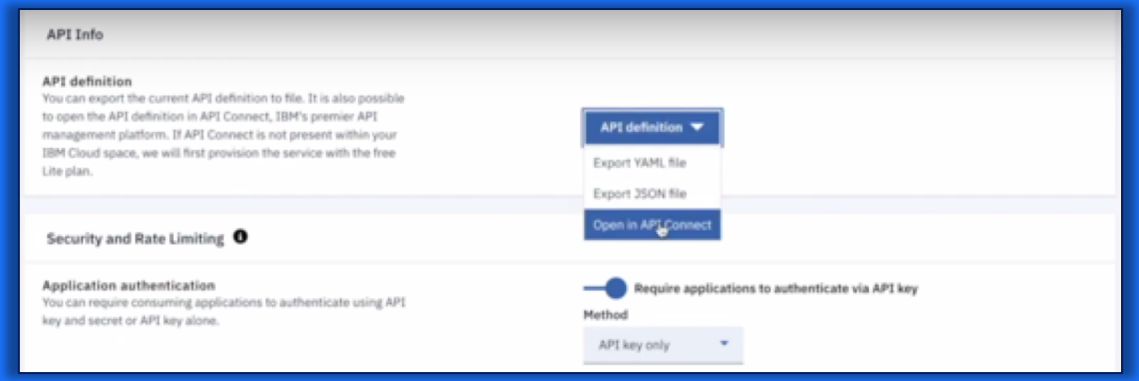
easily create test data and call the operation

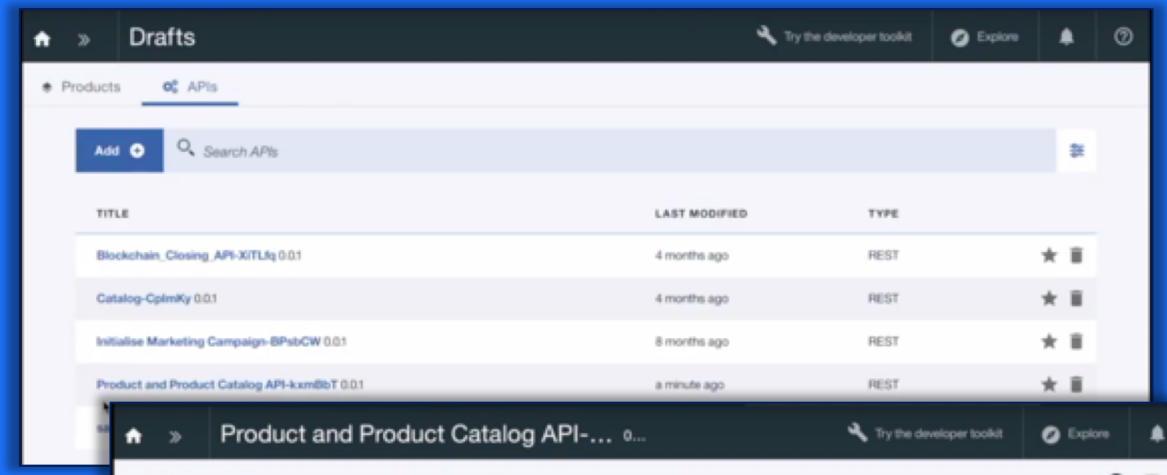


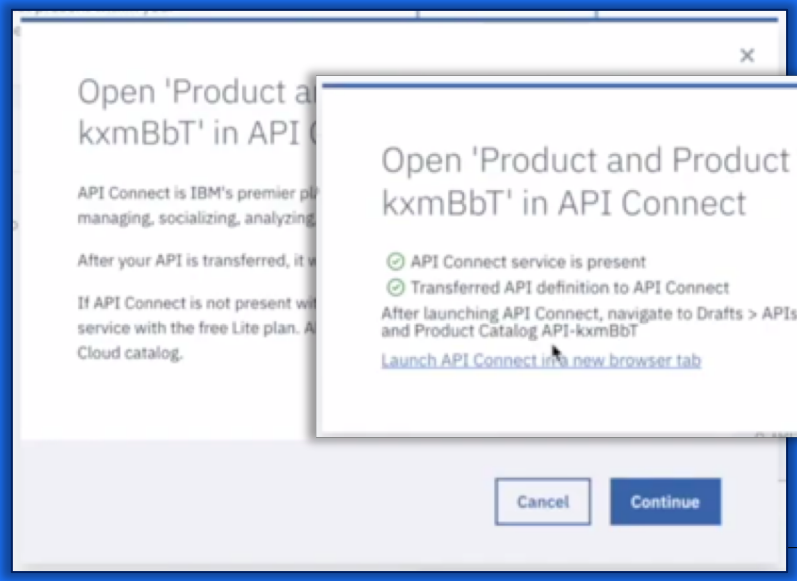
easily create test cases with client request in programming language of choice:
curl, ruby, python, java, php, node, go, swift

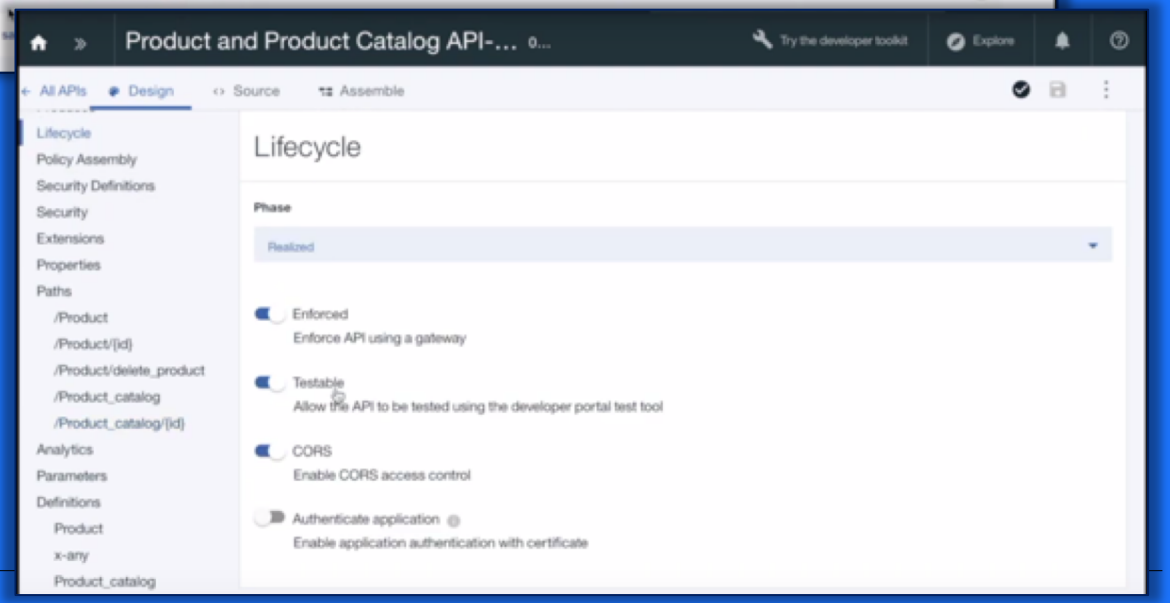


Export to full API Connect





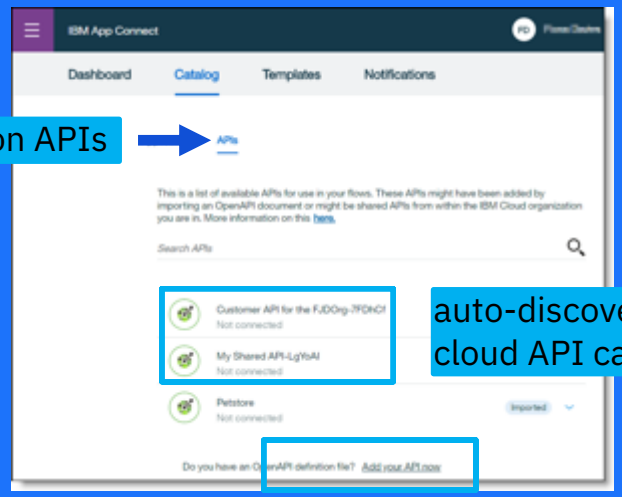




Consuming REST APIs in App Connect Designer

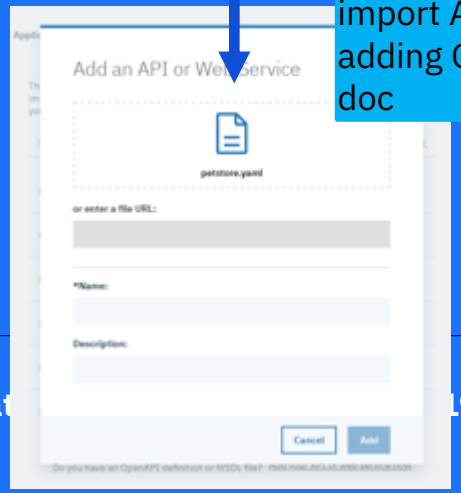
How to connect REST endpoints?

OpenAPI (formerly Swagger) connector / supports OpenAPI v2.0 / supports both public and private APIs

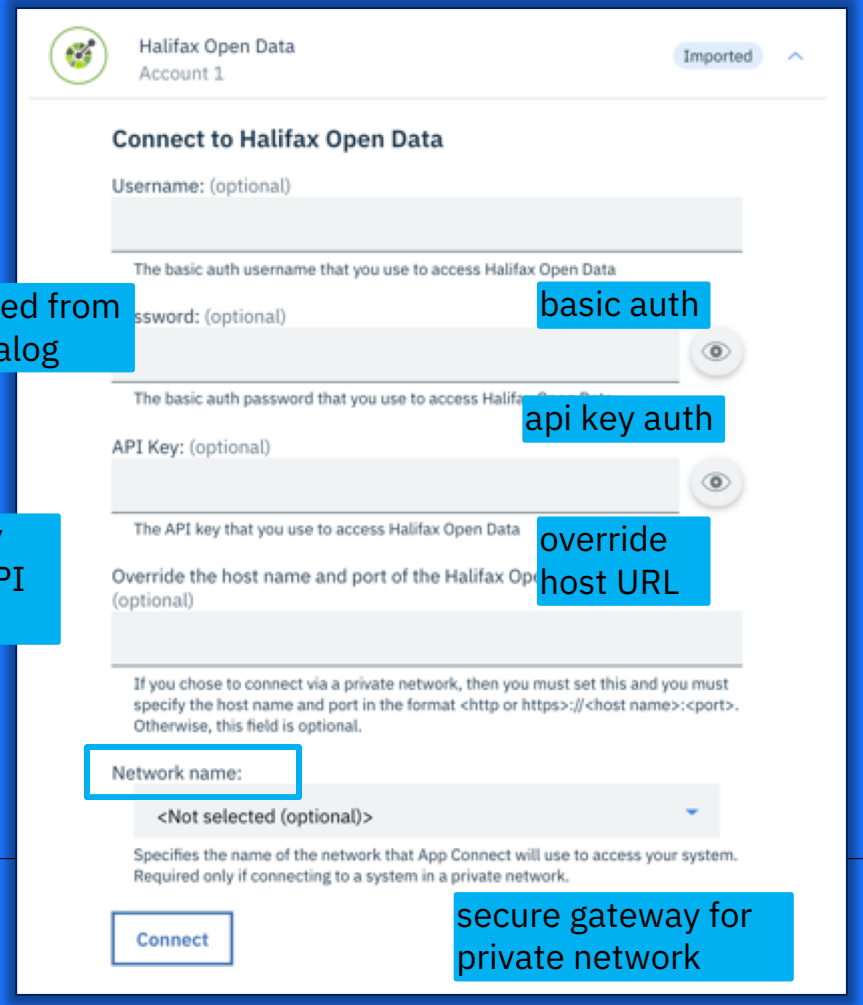


click on APIs

auto-discovered from cloud API catalog



import APIs by adding OpenAPI doc



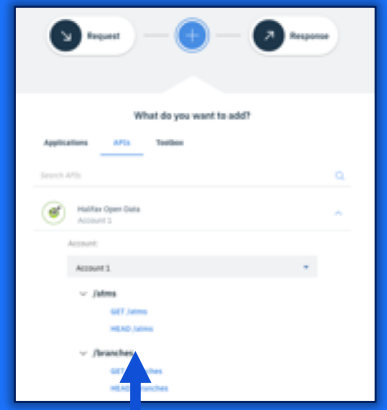
basic auth

api key auth

override host URL

secure gateway for private network

discovers objects



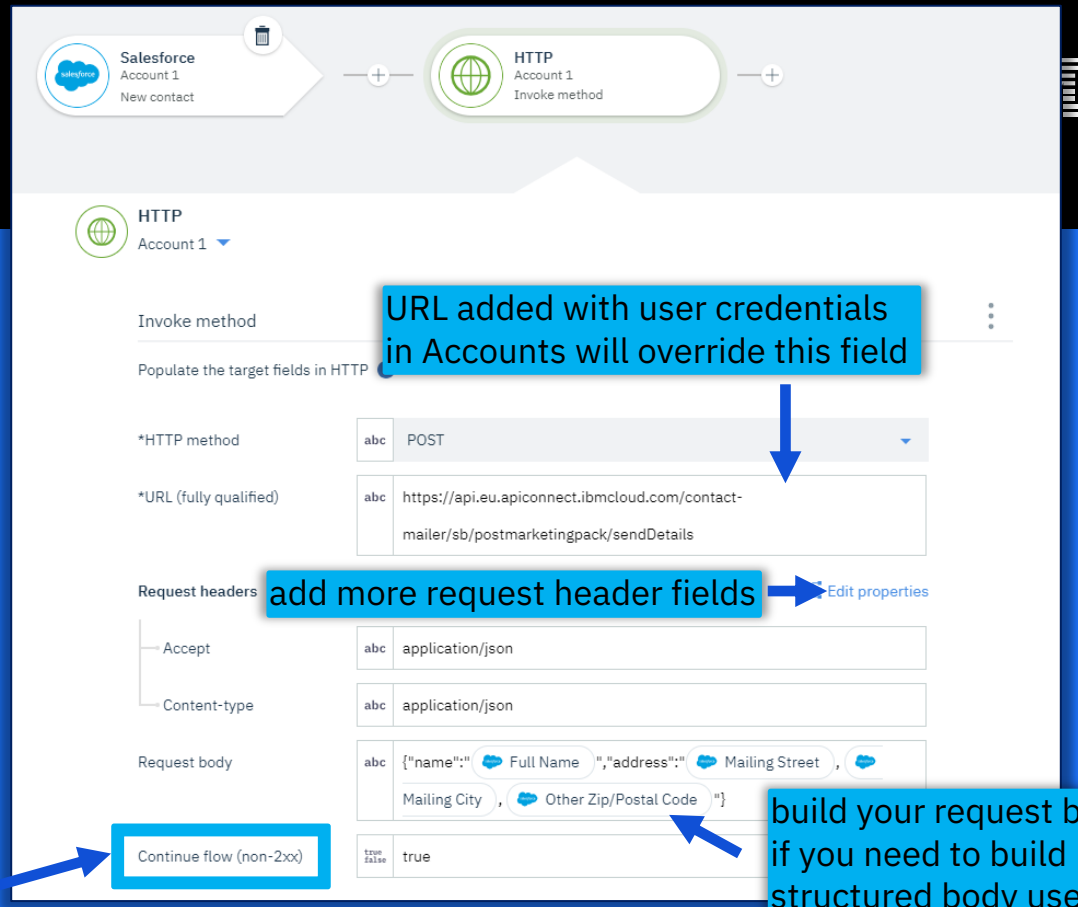
discovers operations

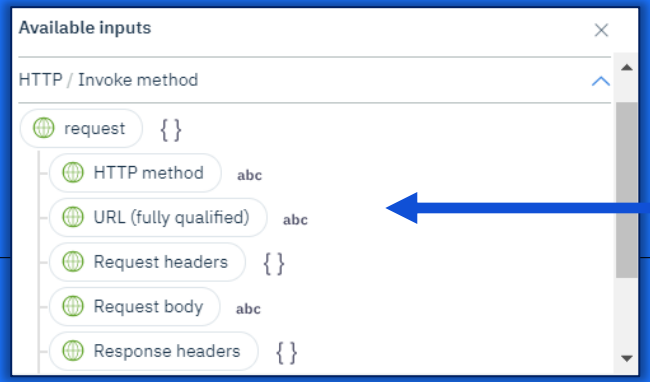
Discovers all the objects, data structure and available operations from the OpenAPI document

object data structure as defined by OpenAPI document

How to connect REST endpoints?

- generic HTTP connector
- invoke any public or private endpoint
  - private endpoint connected via secure gateway
- invoke any supported HTTP methods GET, POST, PUT, PATCH, DELETE
- supports basic authentication
  - enter user credentials during account creation step
  - support for other authentication types (on roadmap)
- allows error handling by implementing flow logic
  - option to continue flow upon non-2XX status
  - response status code is available as mapper input
  - build business logic to handle non-2XX status





API Integration
Unlock your apps and data...

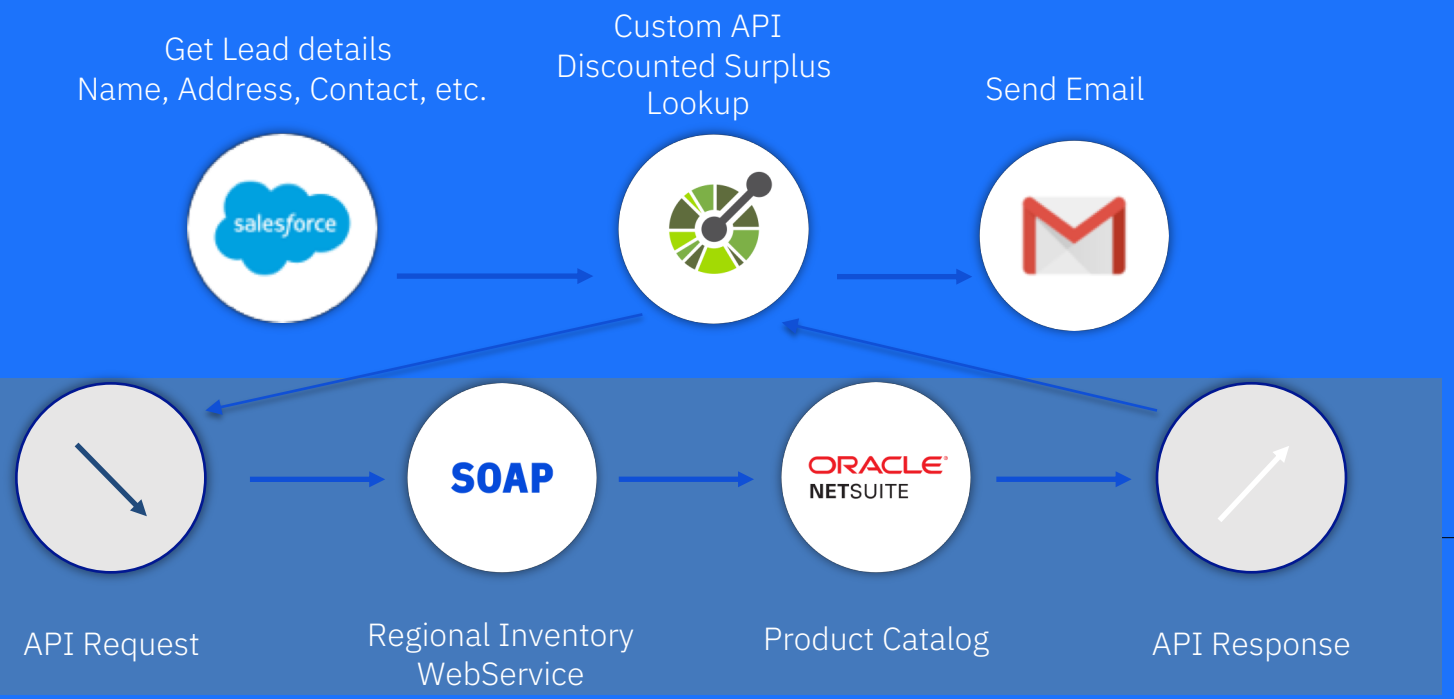
VIDEO

Cassie
Acme - Lead Digital Marketing Coordinator

Now that we are sending people an e-mail as soon as they are entered on the system, can we send them discounts and offers.

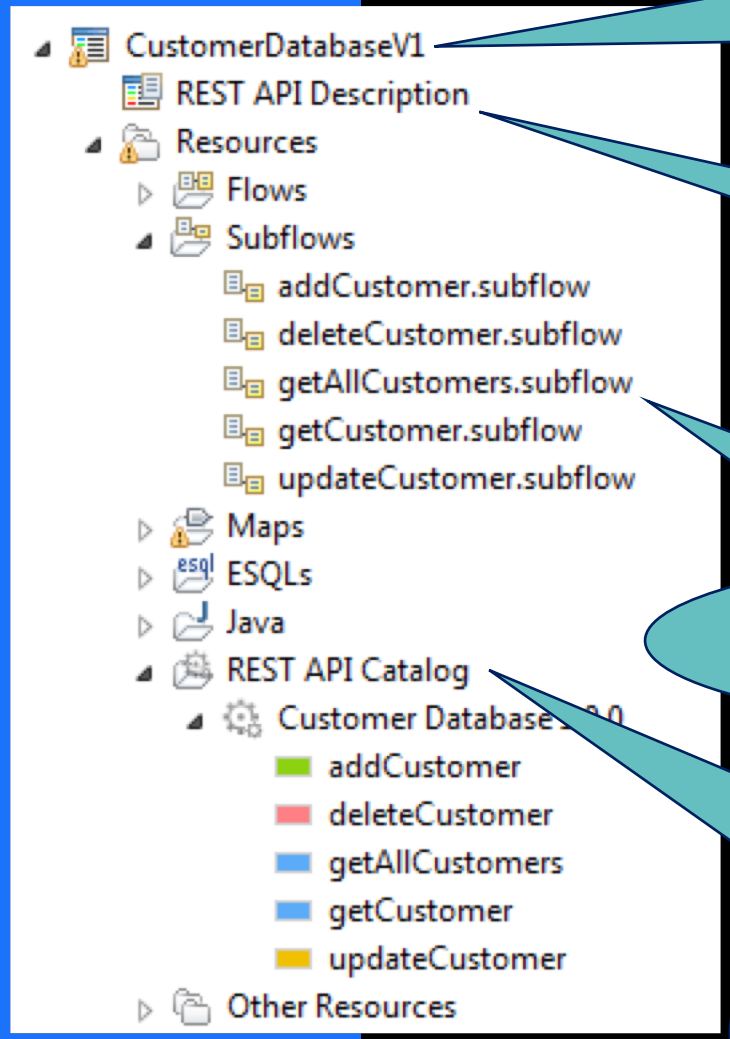
Easy. I can use App Connect to create an integration that uses the API I built last month for querying surplus stock in the regional warehouses and offer that to your lead at a discount.

Tia
Acme – Integration Developer



REST APIs in App Connect Enterprise

REST APIs in App Connect Enterprise



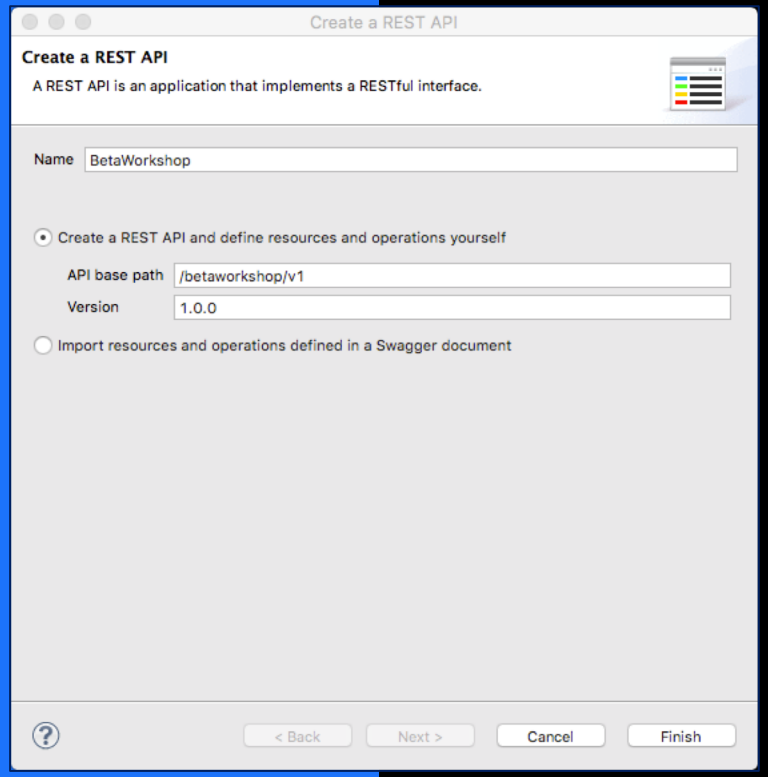
Integration Bus V10 introduced a new type of project, or container, called a REST API.

Clicking on the REST API Description opens the Editor.

Operations defined in the REST API are implemented as subflows.

The “REST API Catalog” is a category that appears in the REST API project and shows all REST APIs and operations that are available to be invoked.

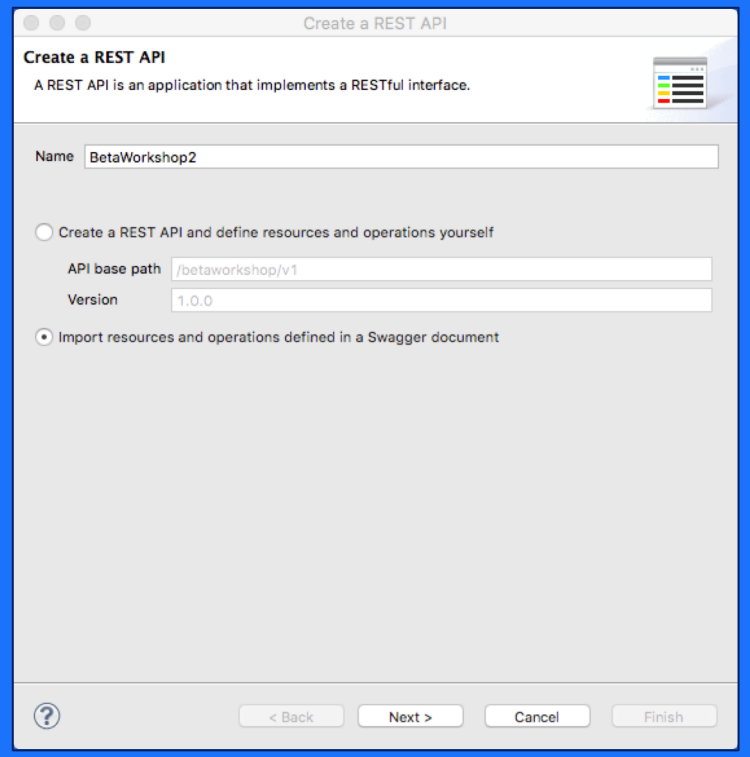
New project wizard

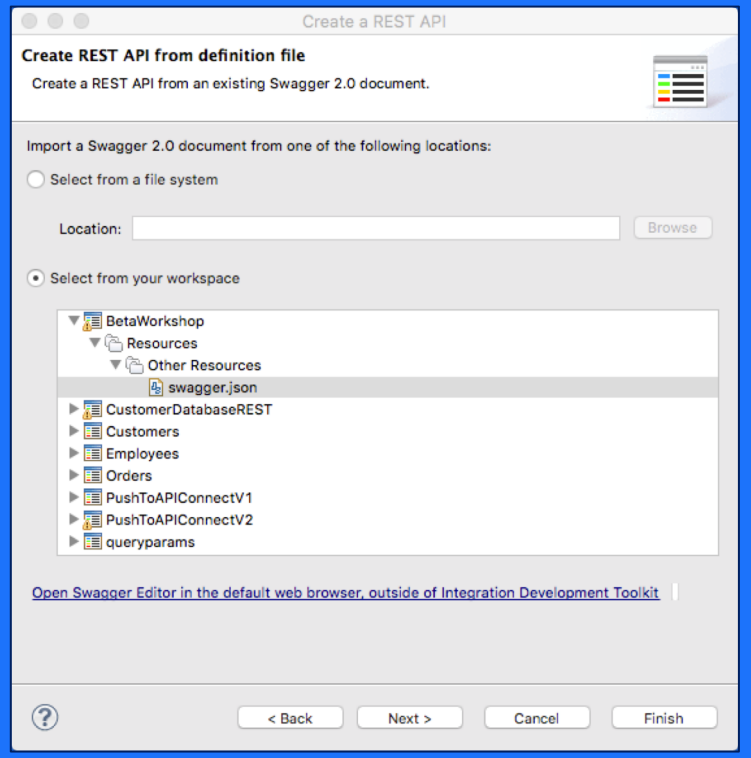


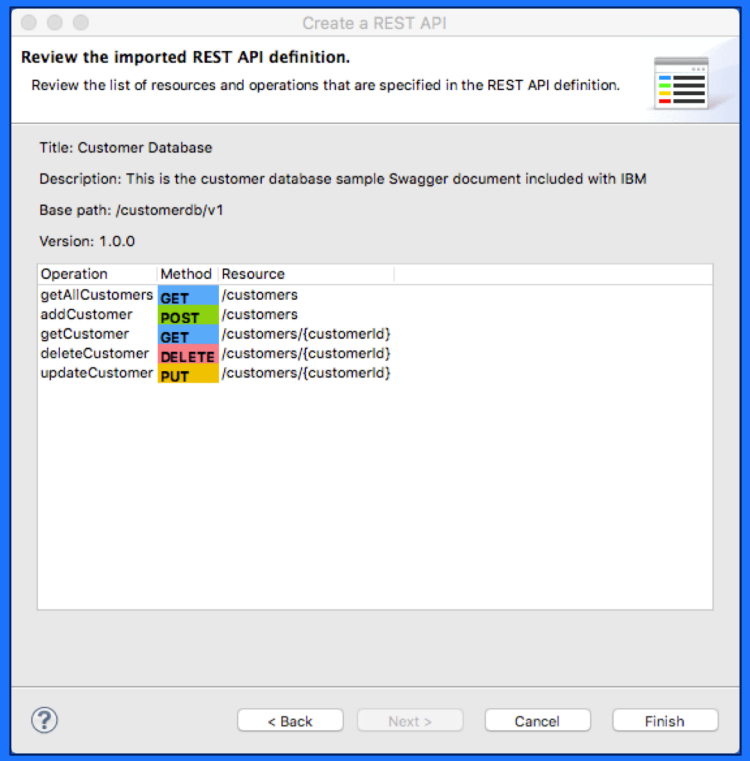
- The first option lets you graphically build a REST API using the toolkit editor without needing to create a Swagger document.
- You can define resources, operations, parameters, and JSON Schema for your REST API without ever writing a line of Swagger!
- The toolkit editor automatically generates a Swagger document for you under the covers.

New project wizard

- If you already have a Swagger document, then the second option lets you build a REST API by importing that Swagger document.

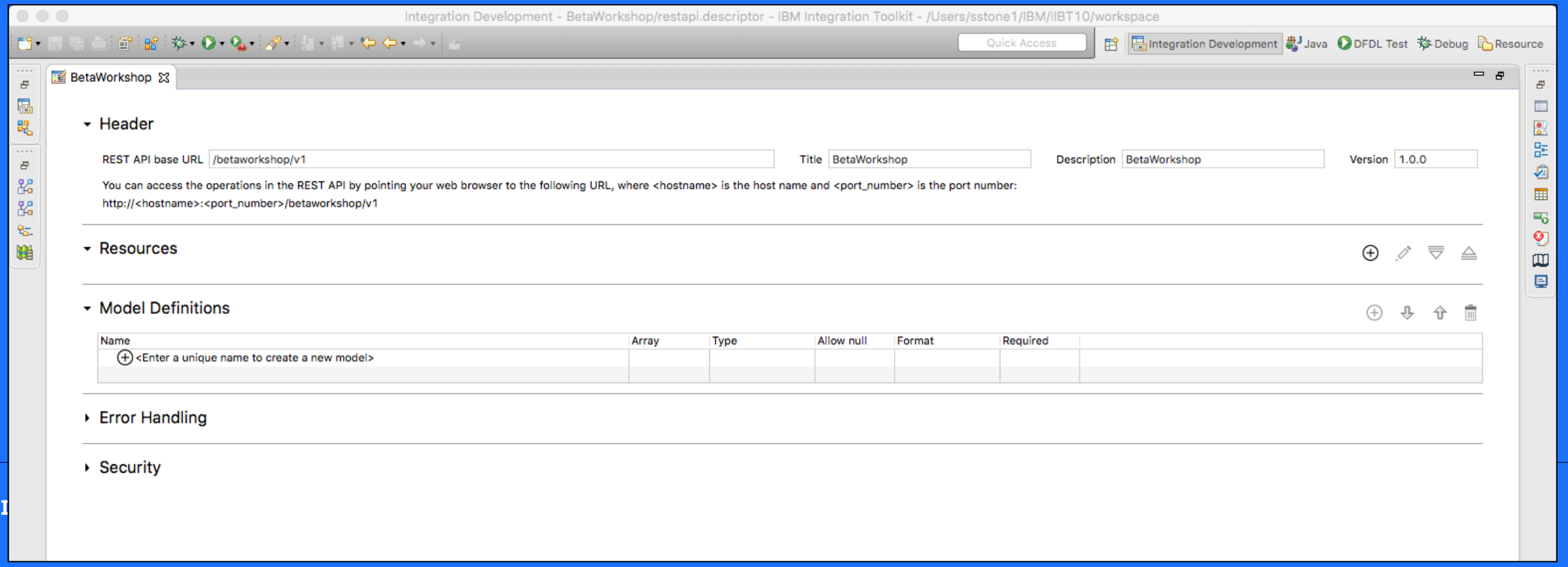






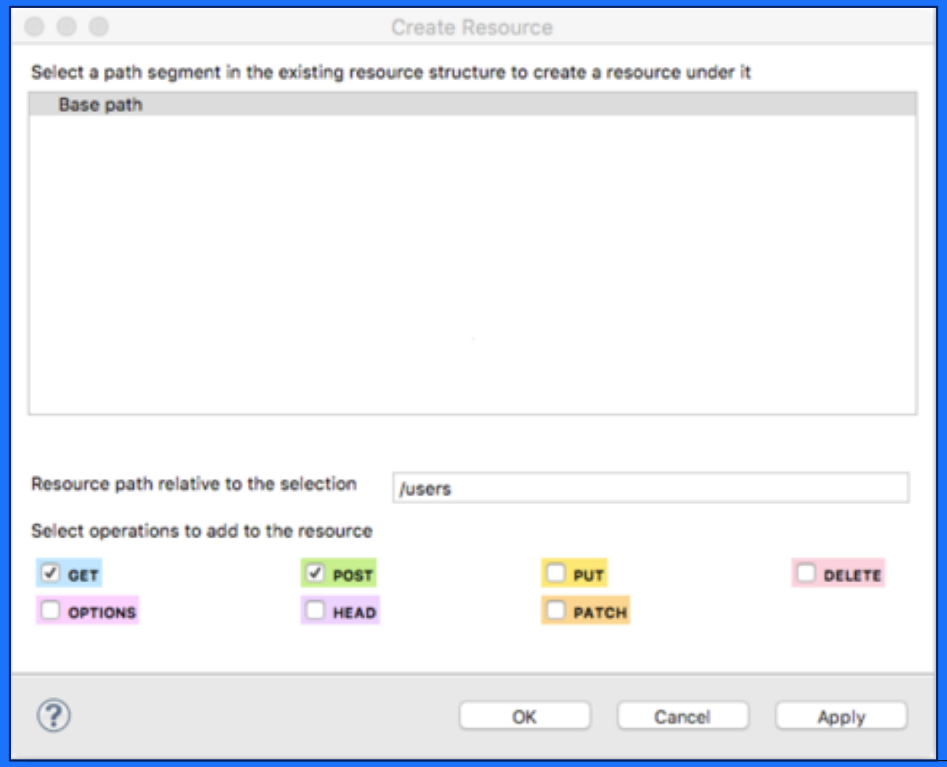
REST API Editor

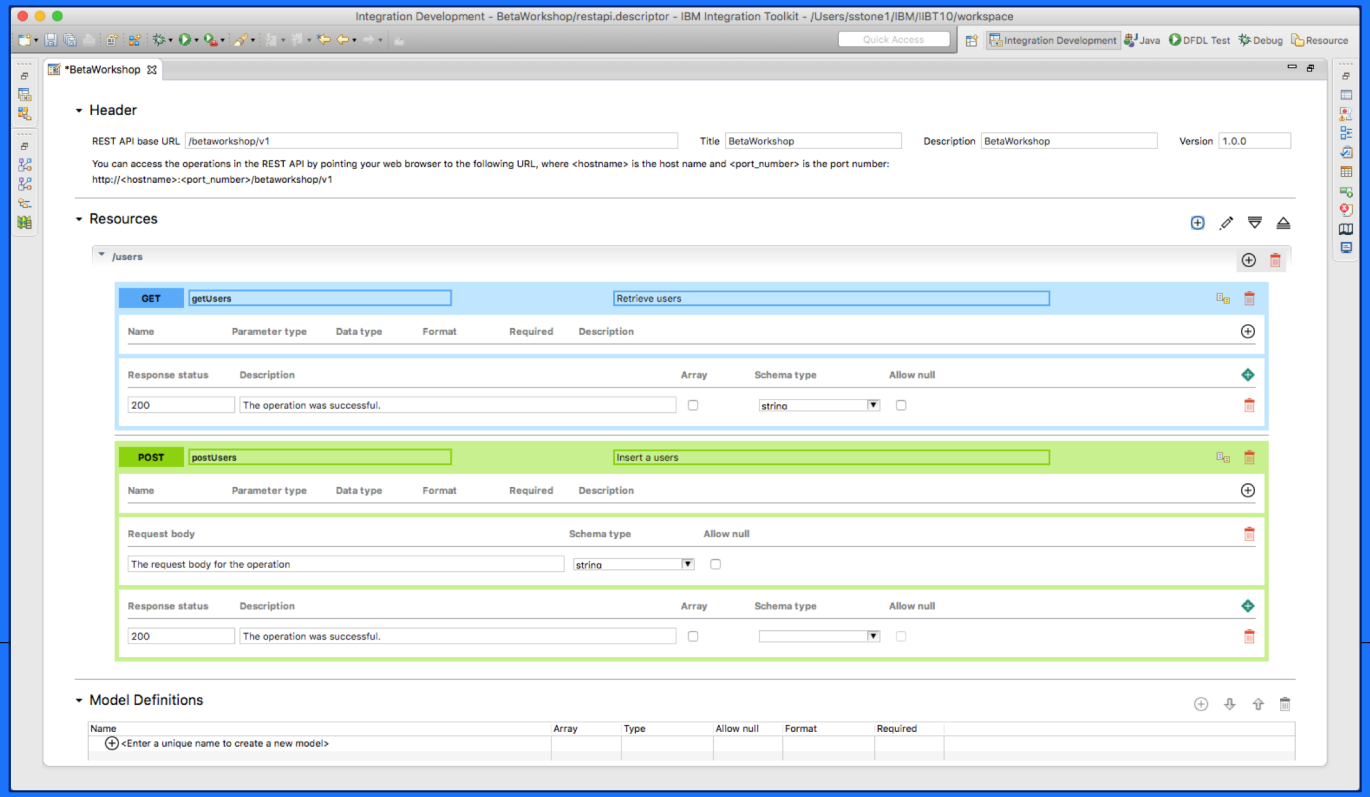
- The REST API Editor is the starting point for a newly created REST API.
- From this point, you can see all of the resources, operations, parameters and JSON Schemas defined in the REST API.



Defining resources

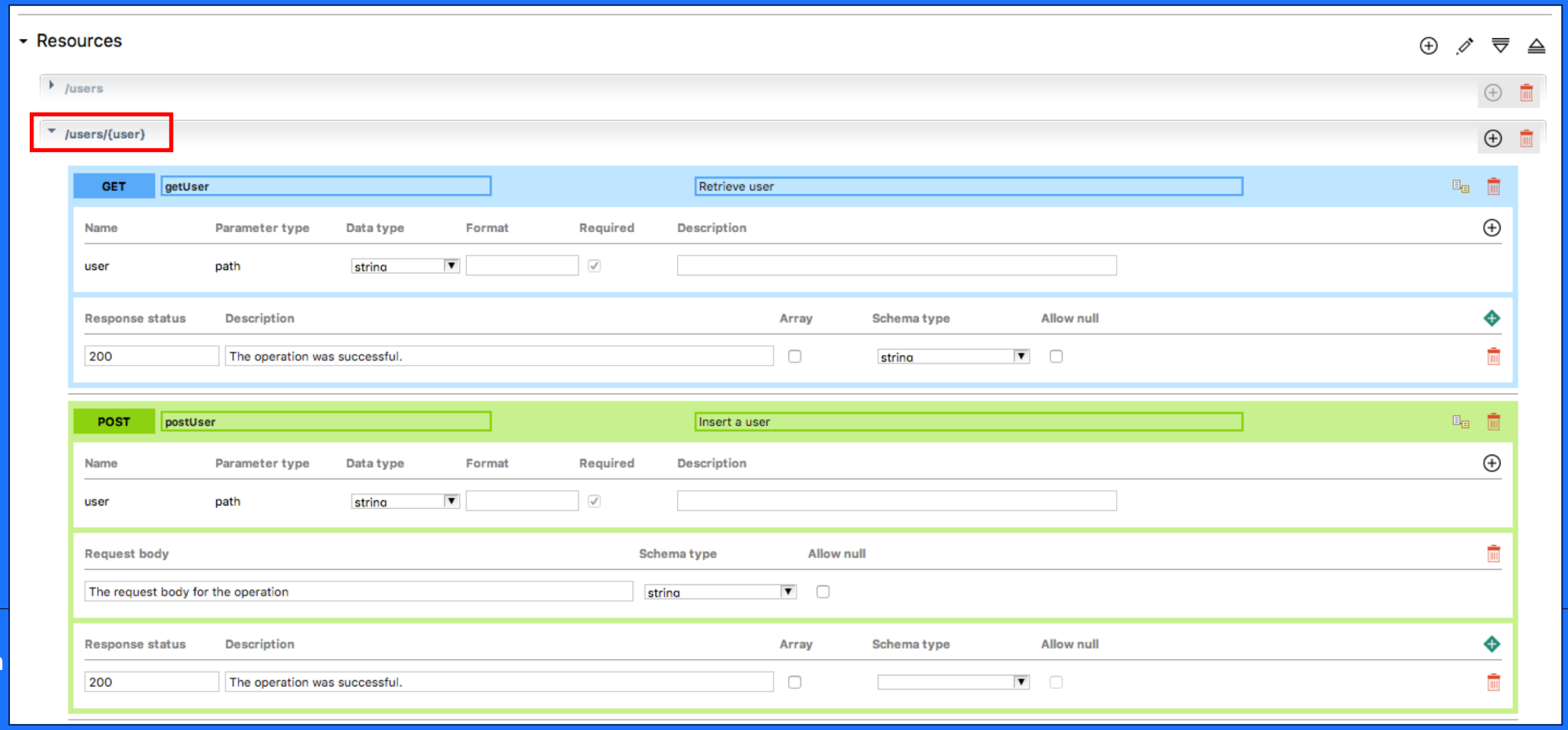
- Add new resources into the REST API by clicking the (+) symbol on the resources section





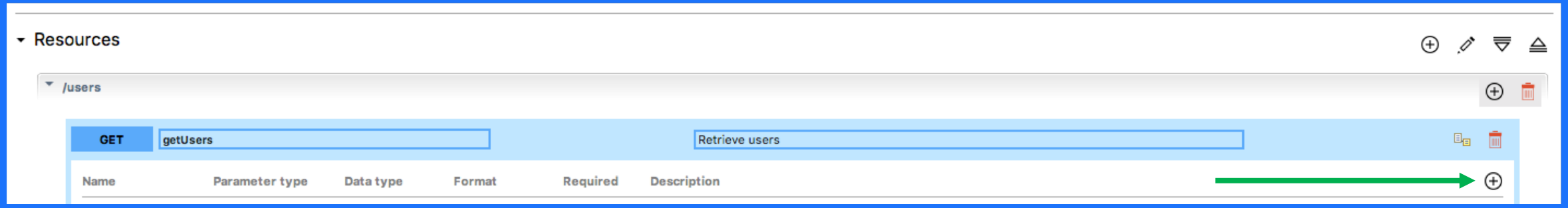
Defining parameters

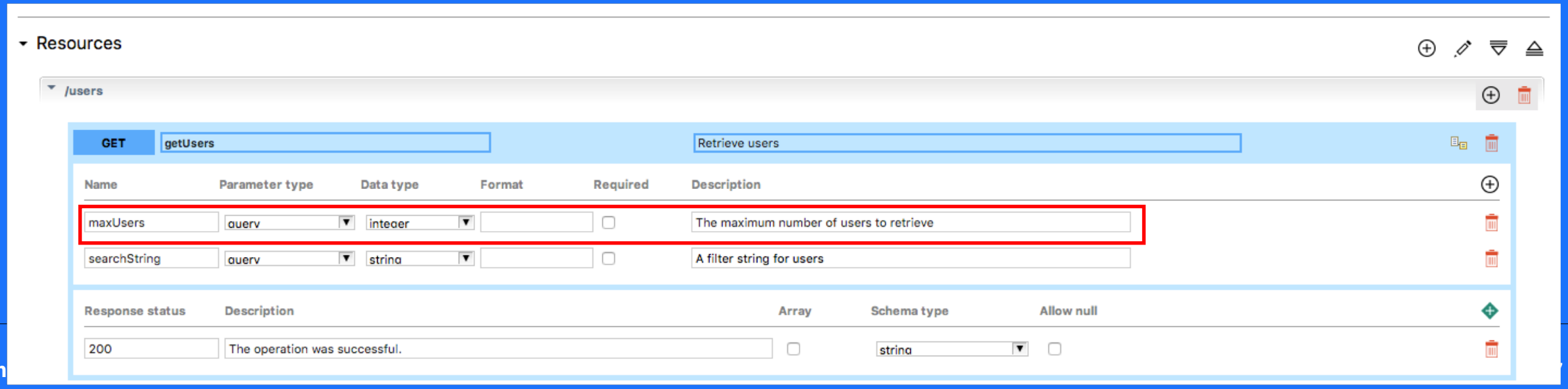
- Path parameters are automatically added to operations when defined as part of the path, by specifying {param} in the path when you add a new resource



Defining parameters

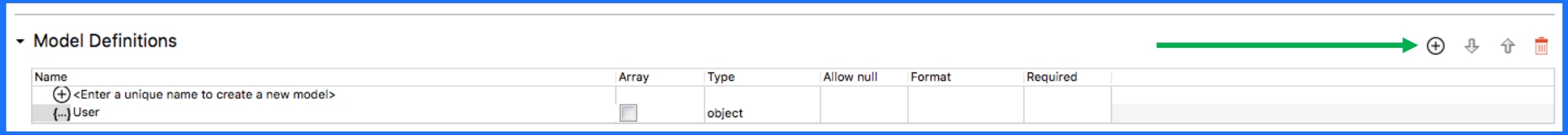
- If your operations accept query or header parameters, then add them in using the (+) button on the operation

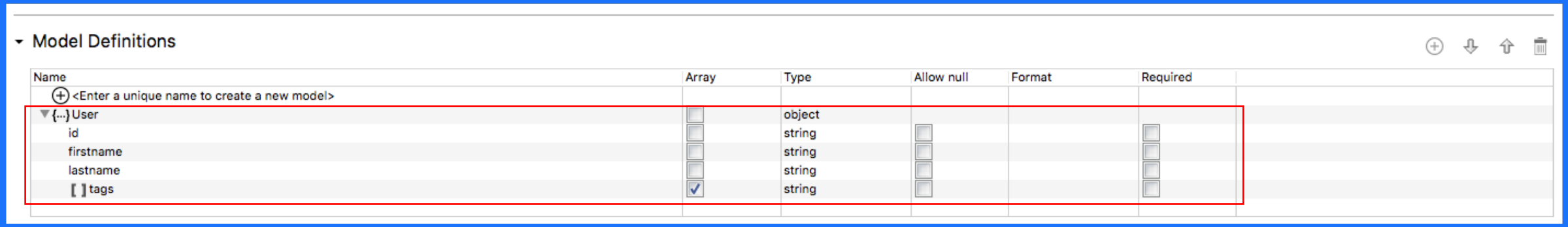




Defining models

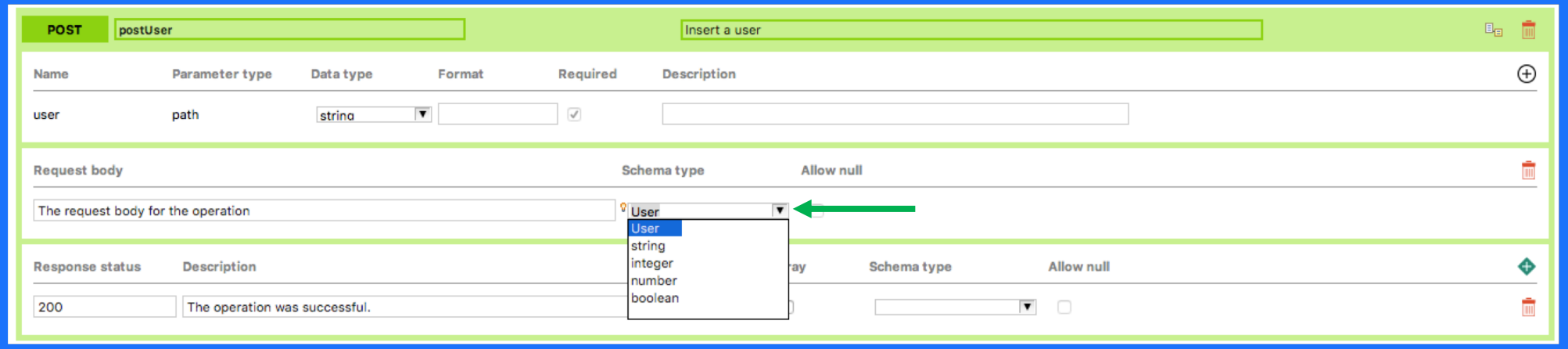
- Models describe the JSON request or response bodies. You can build JSON Schema that describes that JSON by adding new models to the REST API

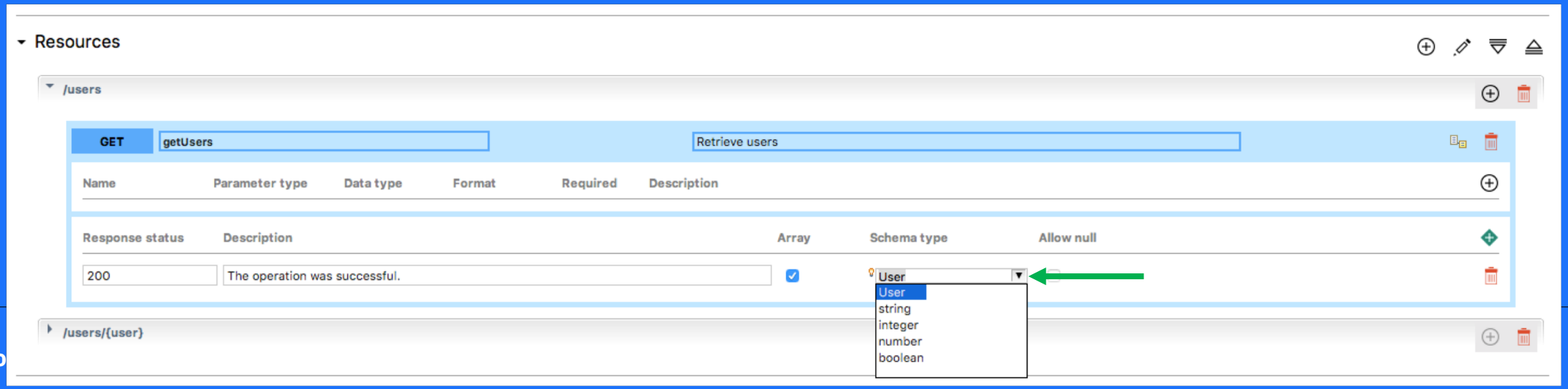




Referencing models

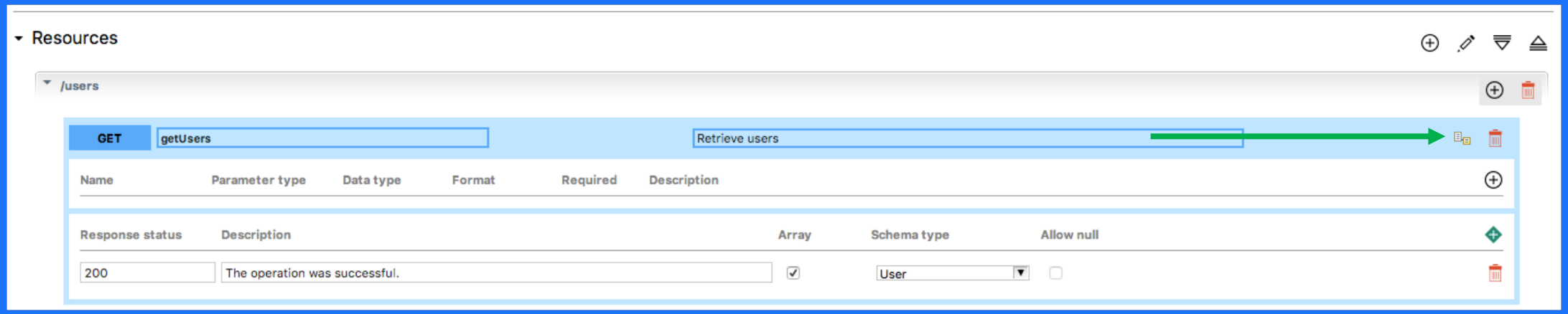
- Once defined, you can reference models in the request or response bodies for an operation

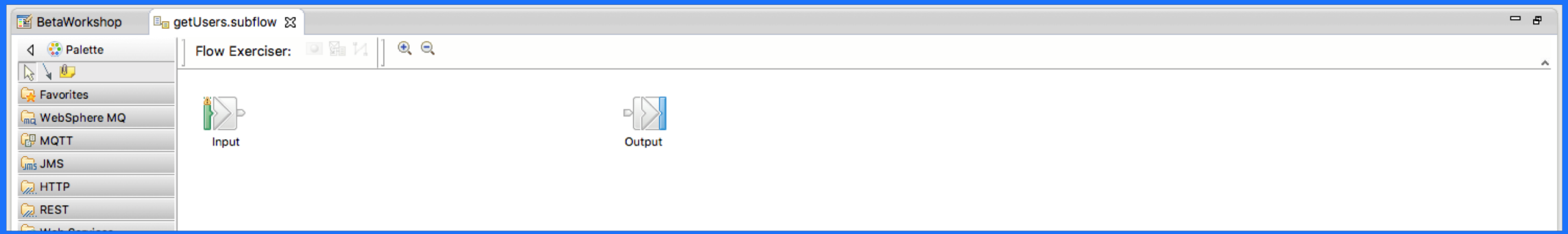




Implementing operations

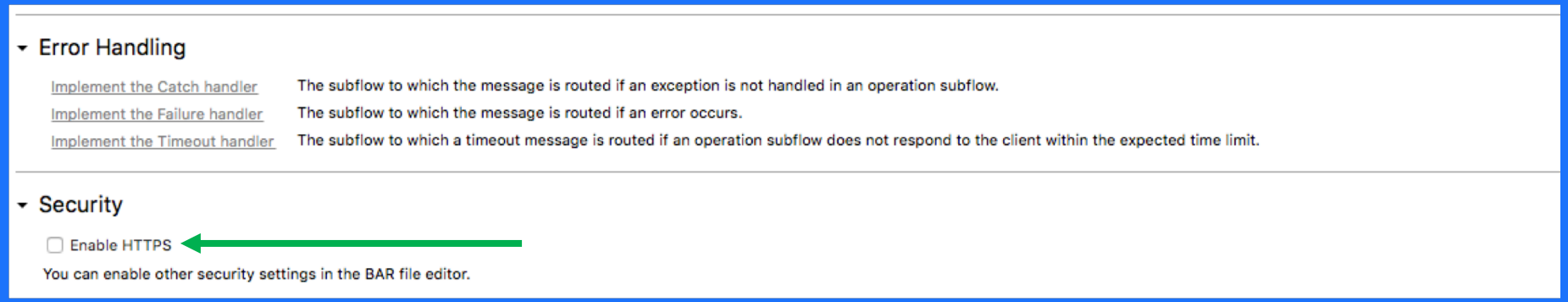
- Clicking on an unimplemented operation automatically generates an empty subflow for that operation, and creates the underlying links between the REST API and that subflow.

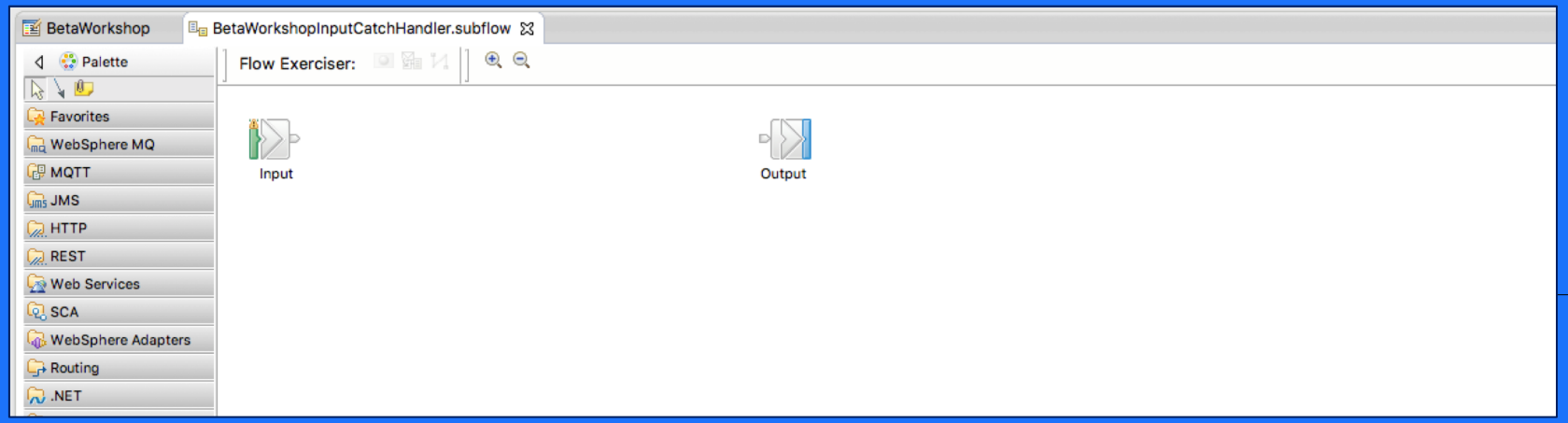




Implementing error handlers and HTTPS

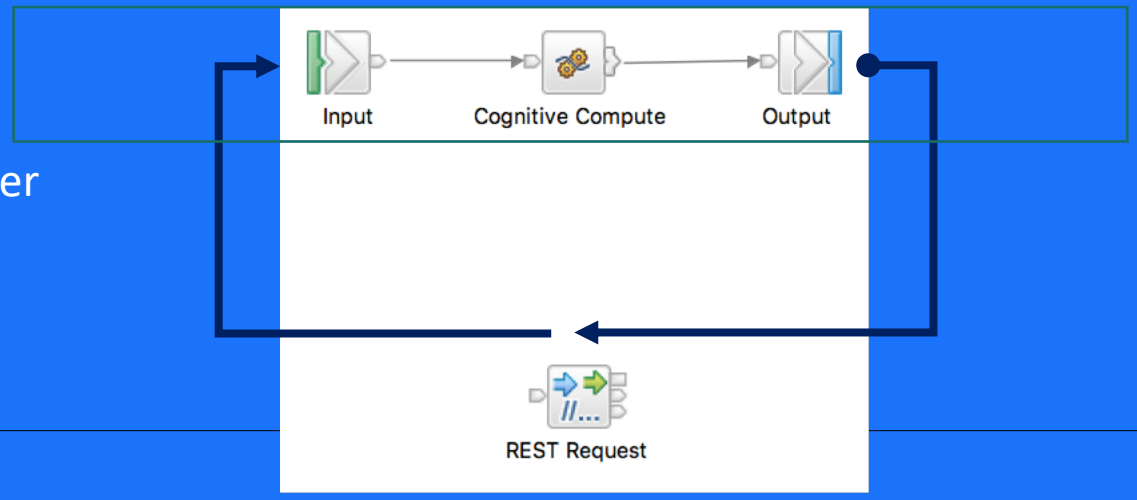
- You can also use the REST API Editor to implement error handling for a REST API. Error handlers are additional subflows that can be used to handle errors and exceptions that are not handled by the subflow for an operation.
- Finally, you can also enable or disable HTTPS





Implementing an operation

- When the operation is called by an HTTP client, a message will be routed automatically by the REST API container to the input node for the corresponding subflow for that operation.
- That message will have a JSON request body, if a body has been provided in the request. JSON is the default message domain for REST APIs, but you can also use other message domains - such as XMLNSC or DFDL.
- When the subflow completes and passes a message to the Output node of the subflow, the response is sent back to the HTTP client.

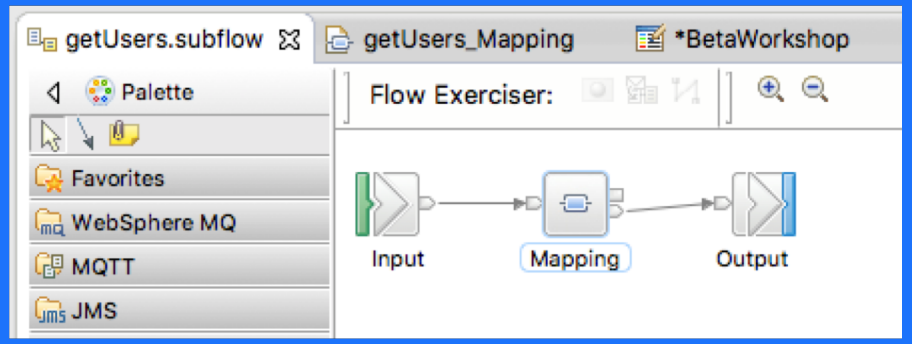


REST API container

REST API client

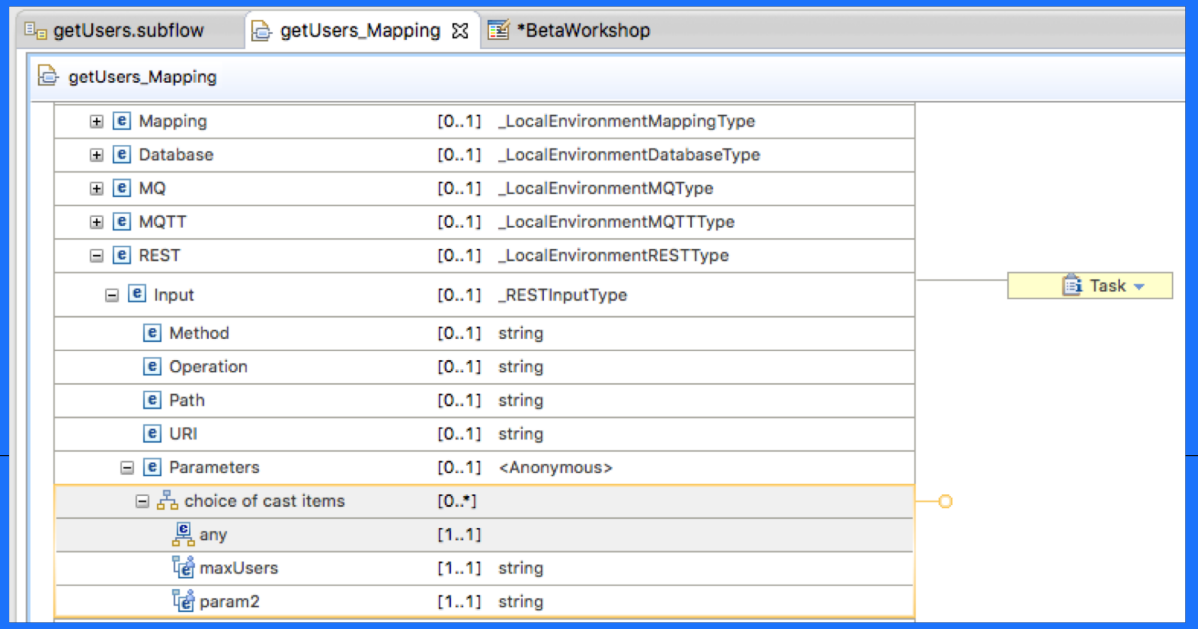
Accessing parameter values

- The message that arrives on the Input node of the subflow will contain the request body and the values of any parameters passed to that operation.



- All of the parameters (path, query, and header parameters) defined by that operation are automatically extracted from the HTTP request and placed into the LocalEnvironment tree – if they have been specified!

- If you use a graphical mapping node in the subflow, then the all of the parameter values are automatically defined in the LocalEnvironment structure for you.



Accessing parameter values

- You can also access the extracted parameter values placed into the LocalEnvironment tree from the message flow nodes by using any of the available programming languages (ESQL, Java, or .NET).

ESQL:

```
DECLARE max INTEGER -1;
IF FIELDTYPE(InputLocalEnvironment.REST.Input.Parameters.max) IS NOT NULL THEN
  SET max = InputLocalEnvironment.REST.Input.Parameters.max;
END IF;
```

Java:

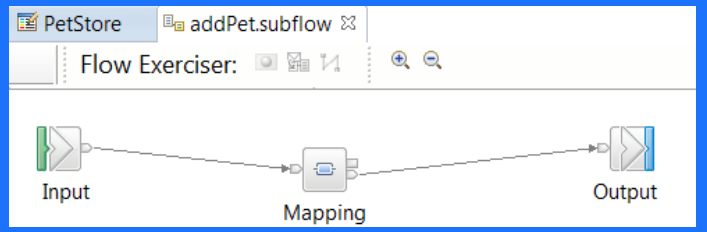
```
MbElement maxElement = inLocalEnvironment.getRootElement().getFirstElementByPath("/REST/Input/Parameters/max");
int max = -1;
if (maxElement != null) {
  max = Integer.valueOf(maxElement.getValueAsString());
}
```

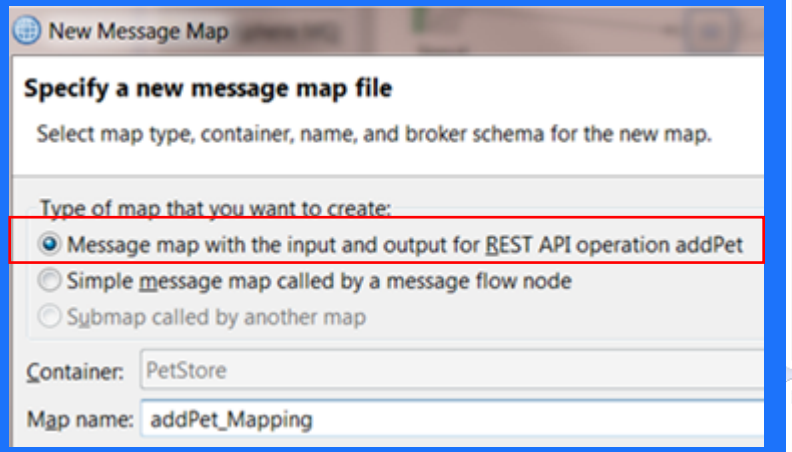
.NET:

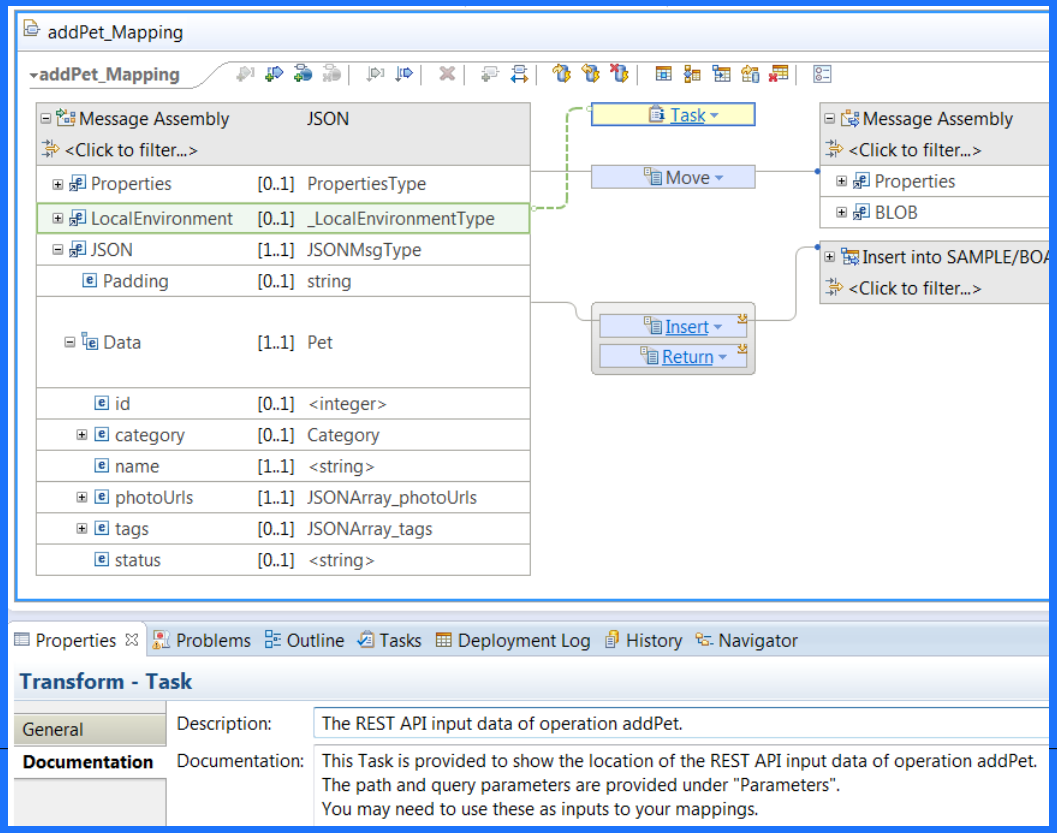
```
NBElement maxElement = inLocalEnvironment.RootElement["REST"]["Input"]["Parameters"]["max"];
int max = -1;
if (max != null) {
  max = (int) maxElement;
}
```

Mapping JSON request/response bodies

- Graphical mapping enhancements introduce support for JSON Schema. The mapper ”knows” that it is in a REST API, and can automatically pick the models for you:

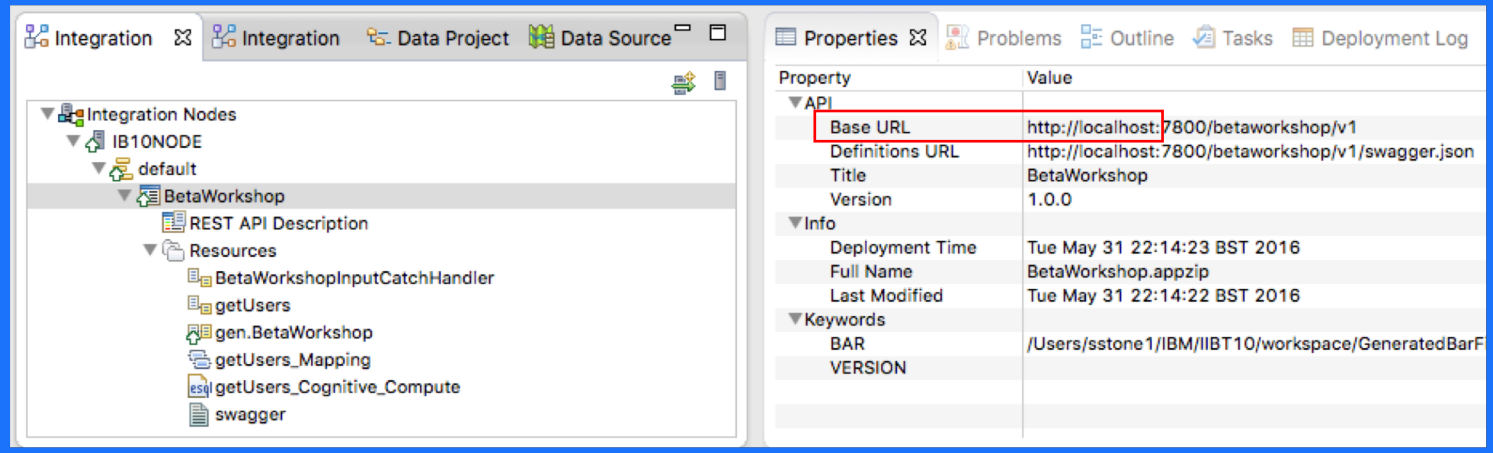






Packaging and deployment

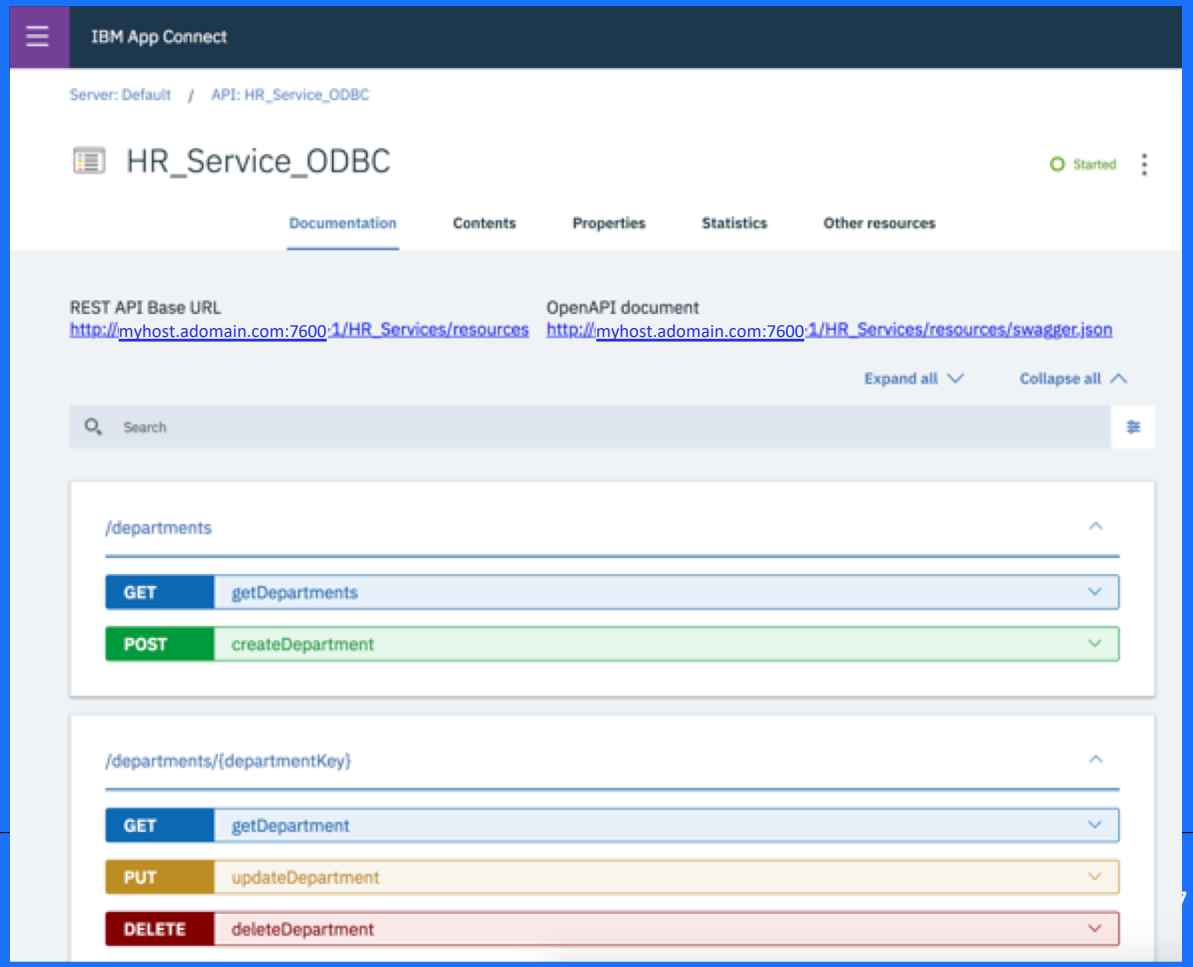
- REST APIs can be packaged into a BAR file and deployed to an integration server using any of the standard mechanisms – either the Integration Toolkit, the command line, or the Integration Java API.
- Once deployed, a REST API appears in the Integration Toolkit and web administration interface as a REST API, under a new REST APIs category.



- The base path of the REST API can be used to isolate a REST API from other REST APIs, in a similar way to a context root for a J2EE application.
- The base path can also be used to isolate multiple versions of the same REST API on a single integration server – for example, you could have /customerdb/v1 and /customerdb/v2.

Administration

- All administrative and operational controls that are available for applications are also available for REST APIs.
- The command line programs that work with applications will also work with REST APIs.
- There is support in the administrative REST API and Integration API (v10) for programmatically interacting with deployed REST APIs.
- The web user interface has also been extended to provide information about the resources, operations, and parameters that are available in a deployed REST API.



Deployed Swagger

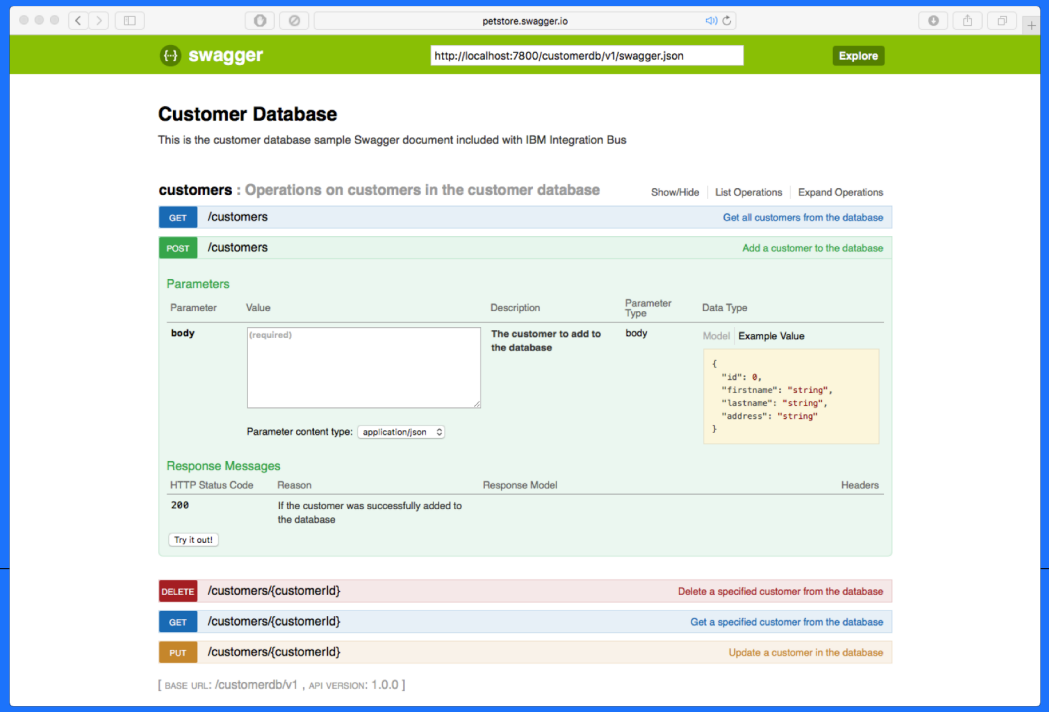
- When a REST API is deployed, the Swagger document for that REST API is automatically made available over HTTP from the same server and port that the REST API is hosted in. The URL for the deployed Swagger document is available from the Integration Nodes view in the Integration Toolkit, as well as the web user interface.
  - http://localhost:7800/customerdb/v1/swagger.json
  - http://localhost:7800/customerdb/v1/swagger.yaml

- The deployed Swagger document is automatically updated to reflect the server, port, and HTTP/HTTPS details for the deployed REST API. You do not have to update it with the correct details before deployment.

- This means that you can easily use the deployed Swagger document in combination with open source Swagger tooling, to explore and interact with a deployed REST API.

Swagger UI

- You can pass the URL of the deployed Swagger document to Swagger UI, which allows you to explore and test a deployed REST API. Swagger UI is available from:
  - http://petstore.swagger.io/ (live demo, but can also be used)
  - https://github.com/swagger-api/swagger-ui (source for download)
- In order for Swagger UI to work, you must enable Cross-Origin Resource Sharing (see A03 yesterday)



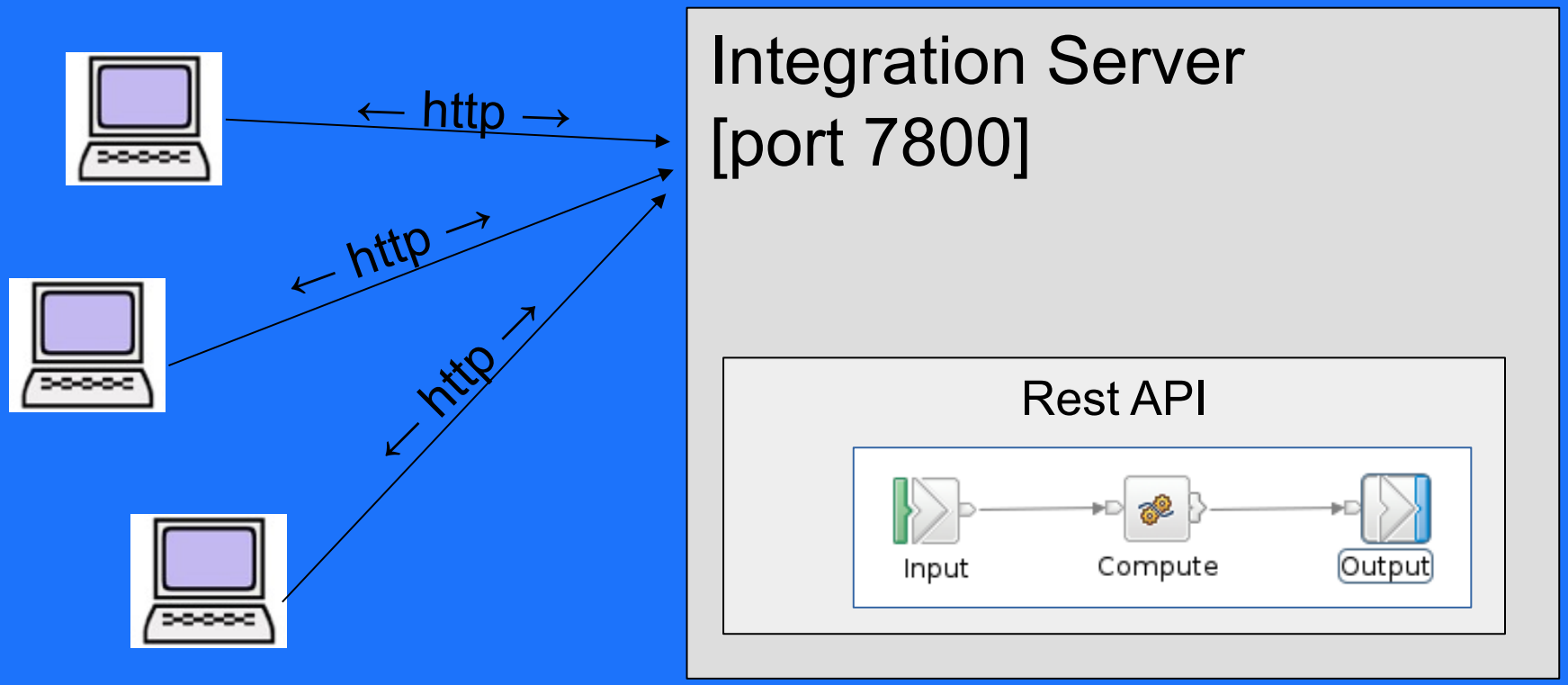
Developing REST APIs in App Connect Enterprise

LIVE DEMO

Pushing a REST API to API Connect

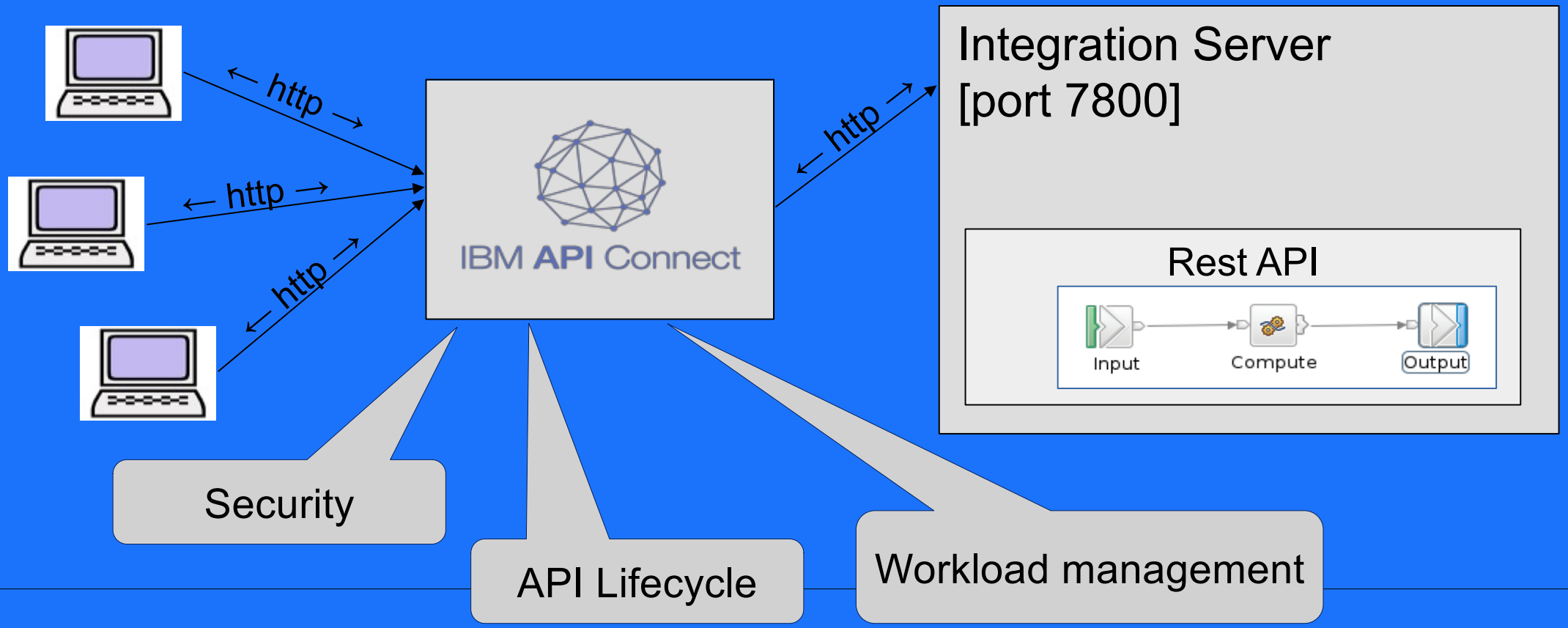
REST APIs developed in IIB

- We have just seen how REST APIs allow integrations to be exposed as a RESTful web services that can be called by HTTP clients.

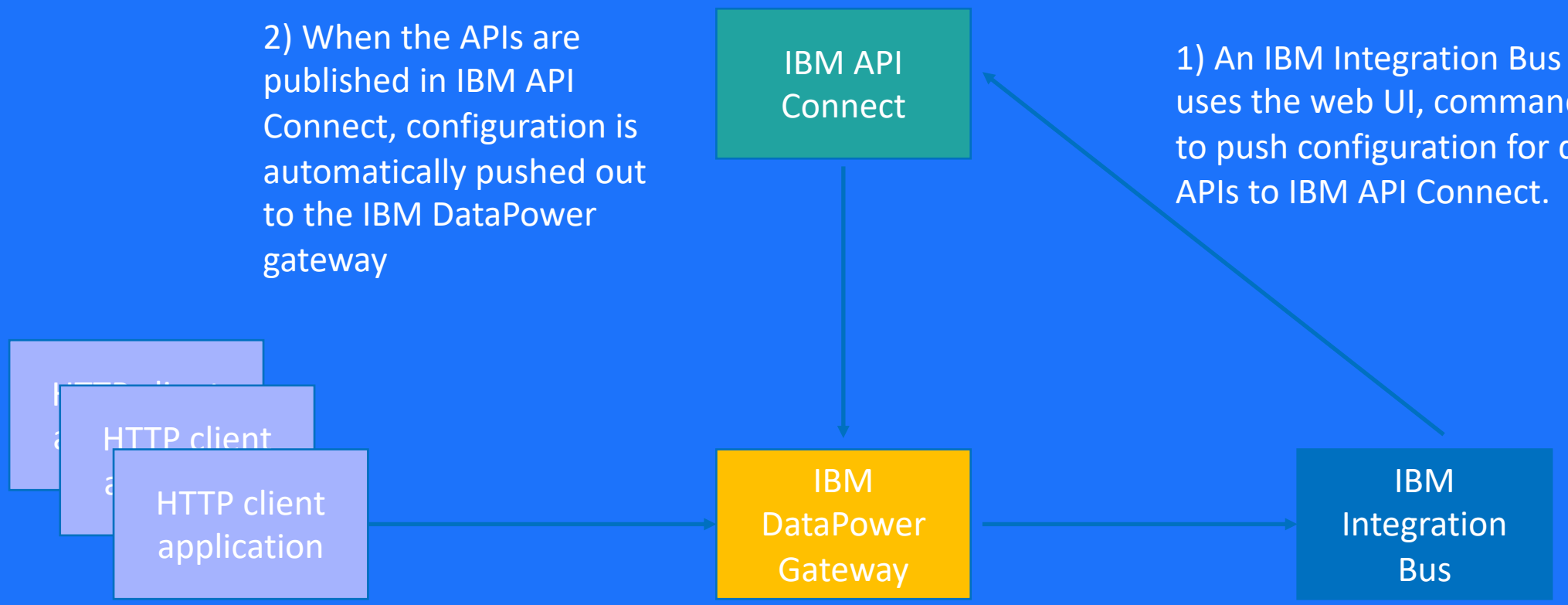


IBM API Connect

- IBM API Connect is an API management solution
- IBM API Connect provides a layer of security and manageability to your REST APIs.



Interaction between IIB and IBM API Connect

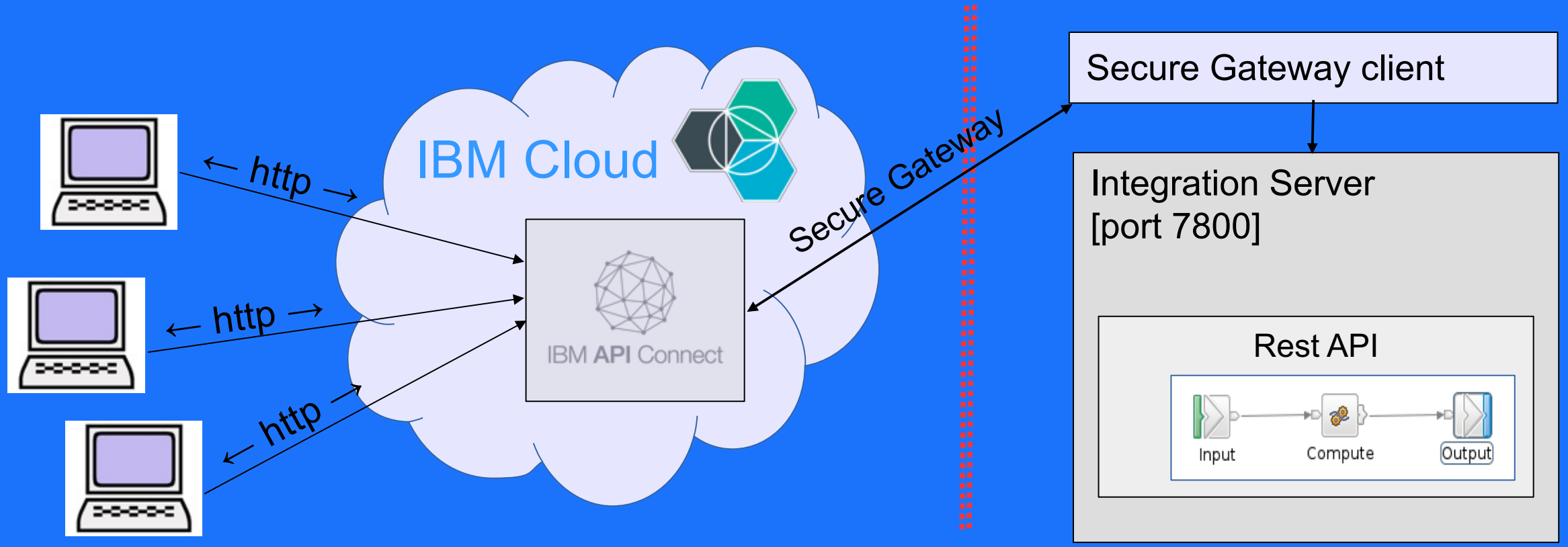


3) HTTP client applications wishing to make an HTTP request to the REST API deployed on IBM Integration Bus make an HTTP request to the IBM DataPower Gateway.

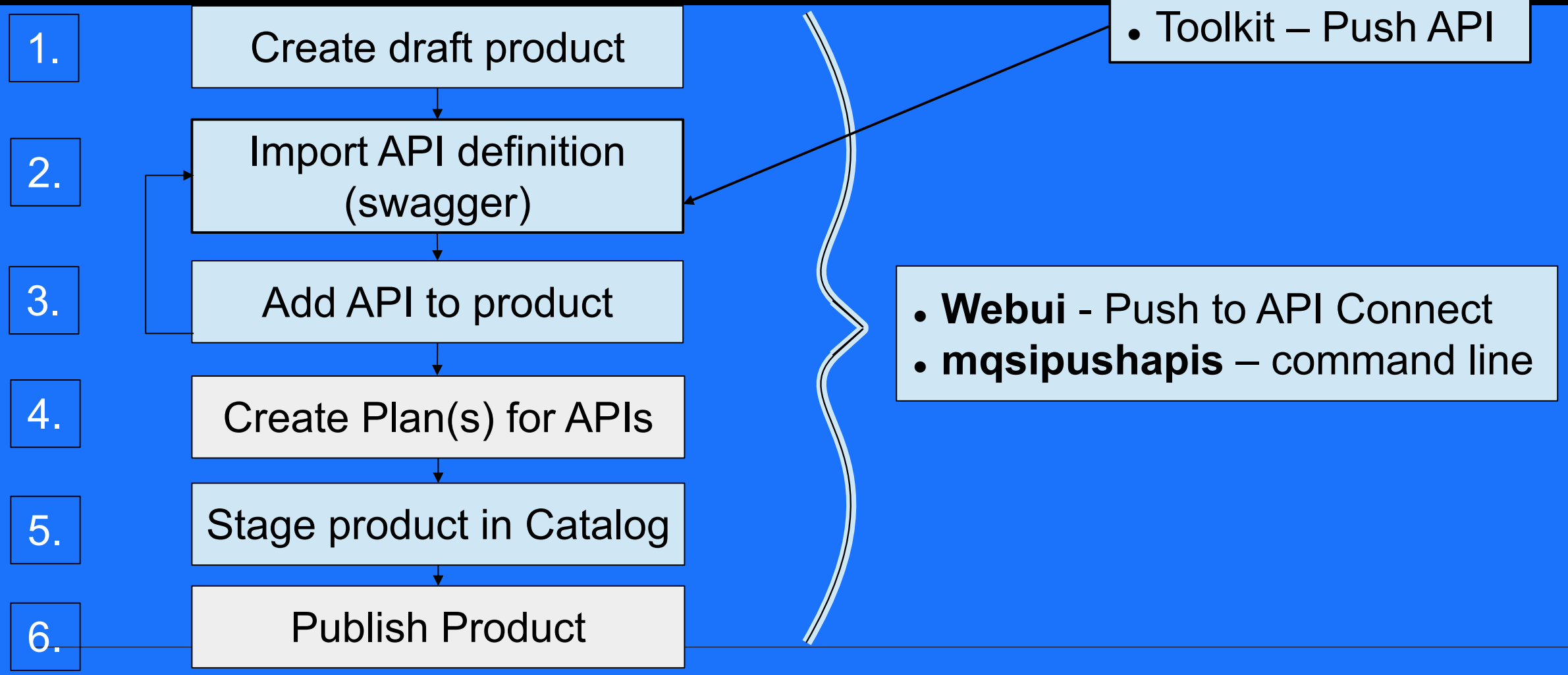
4) The IBM DataPower Gateway enforces access control, rate limiting, and other policies before it proxies the HTTP request to IBM Integration Bus.

IBM API Connect in IBM Cloud

- API Connect can fully reside in the Cloud, and utilise the Bluemix Secure Gateway component to bridge to on-premise IIB installation.



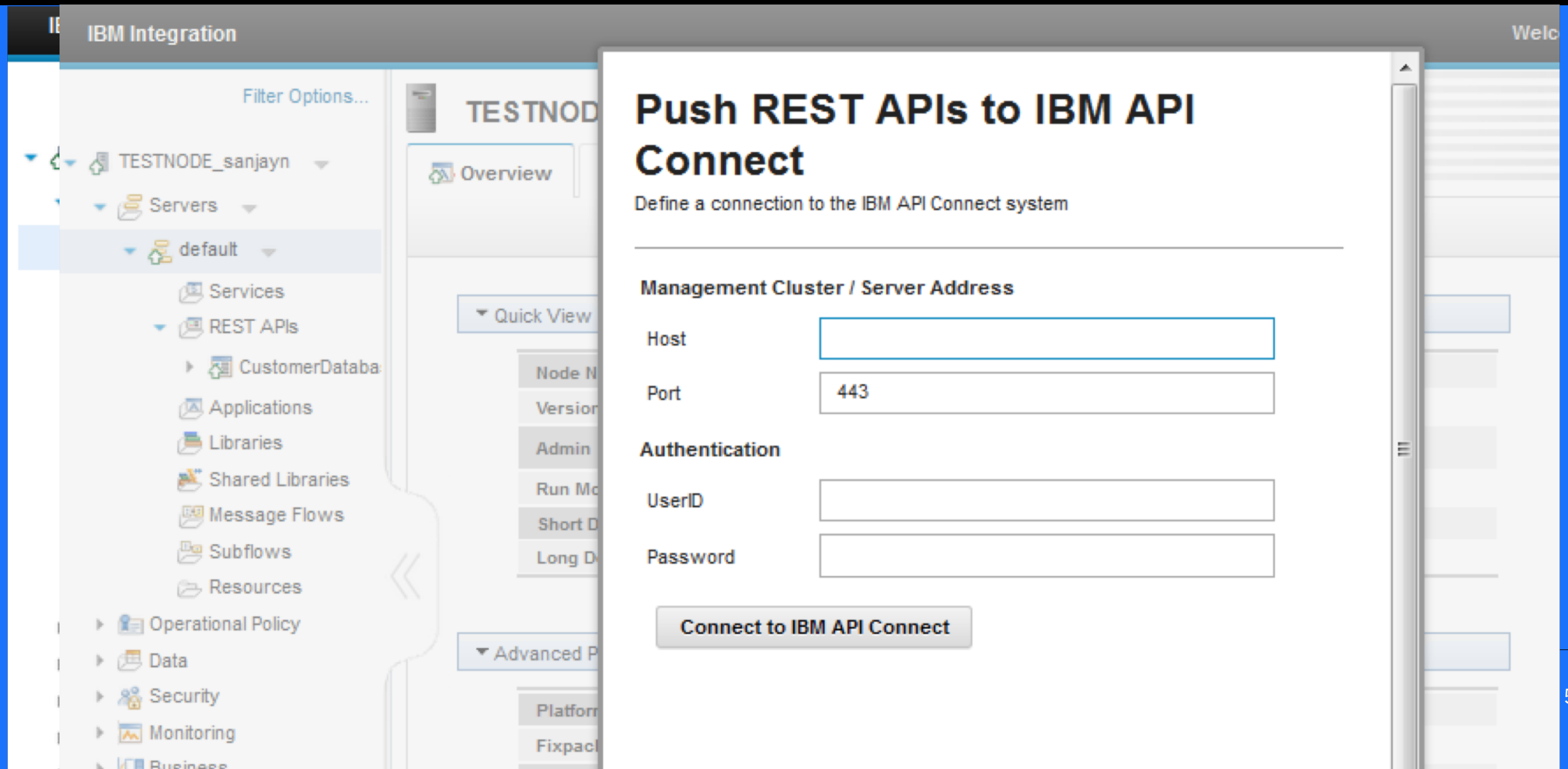
Exposing an API in IBM API Connect



Pushing API's from IIBv10 to IBM API Connect

- Using Toolkit
  - Can only push single APIs. Does not create a product or stage the product in the catalog.
  - Works with IBM API Management v4 and IBM API Connect v5
- Using WebUI
  - Push multiple APIs to IBM API Connect
  - Available from Context menu on Integration Server
  - Creates Product and optionally stages in a Catalog
  - Works only with IBM API Connect v5
- Using 'mqsipushapis' command
  - Same operation as offered using WebUI
  - Works only with IBM API Connect v5

Pushing API's from IIBv10 to IBM API Connect: Web UI



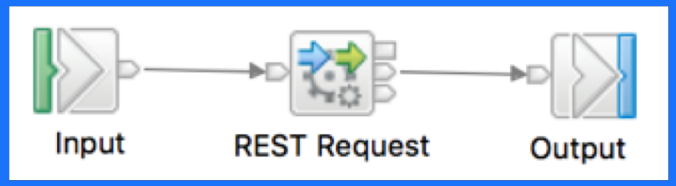
Pushing API's from IIBv10 to IBM API Connect: Command Line

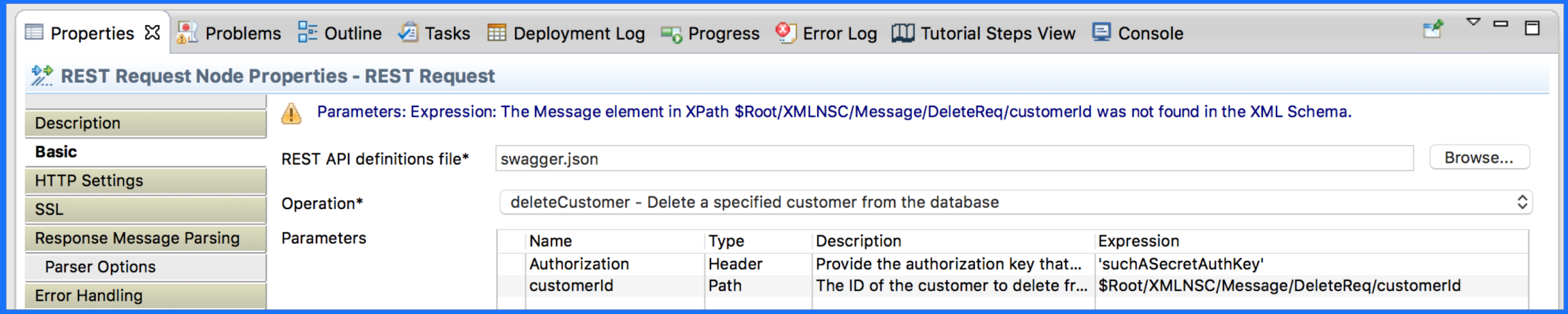
```
mqsipushapis integrationNodeSpec -e integrationServerName -t apiConnectHost -g apiConnectPort
        -u apiConnectUser -a apiConnectPassword -o apiConnectOrganization
        [-c apiConnectCatalogTitle] -r apiConnectProductTitle [-d apiConnectProductName]
        [-s apiConnectProductVersion] -k restApis [-x httpInboundProxyHost]
        [-y httpsInboundProxyHost] [-v traceFileName] [-w timeoutSecs]

Command Options:
'-t apiConnectHost' The host name of the IBM API Connect system.
'-g apiConnectPort' The port of the IBM API Connect system.
'-u apiConnectUser' The user name that is being used to connect to IBM API Connect.
'-a apiConnectPassword' The password that is being used to connect to IBM API Connect.
'-o apiConnectOrganization' The name of the API Connect Organization where the APIs will be pushed.
'-c apiConnectCatalogTitle' The title of the API Connect Catalog where the product will be staged.
'-r apiConnectProductTitle' The title of the IBM API Connect Product to create or update
'-d apiConnectProductName' The name of the IBM API Connect Product to create or update.
'-s apiConnectProductVersion' The version of the IBM API Connect Product to create or update.
'-k restApis' A list of API names, separated by colons (:), that will be pushed to IBM API Connect.
'-x httpInboundProxyHost' The host name and port of a proxy that receives the inbound HTTP request.
'-y httpsInboundProxyHost' The host name and port of a proxy that receives the inbound HTTPS request.
'-v traceFileName' Send verbose internal trace to the specified file. This is optional.
'-w timeoutSecs' Maximum number of seconds to wait for the integration node to respond.
```

Consuming REST APIs in App Connect Enterprise

Introducing the REST Request node

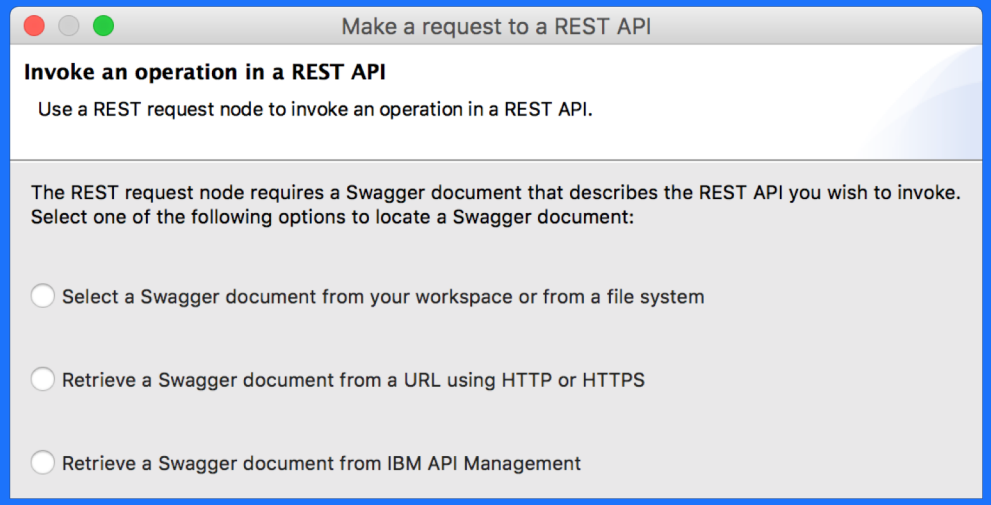
- The REST Request makes it much simpler for an ACE flow developer to make a call to a REST API
- Point it at a swagger file which defines the API
- Choose an operation and away you go!





REST Request node and Swagger

- The REST Request node requires a Swagger document.
- Swagger documents can come from many different sources:

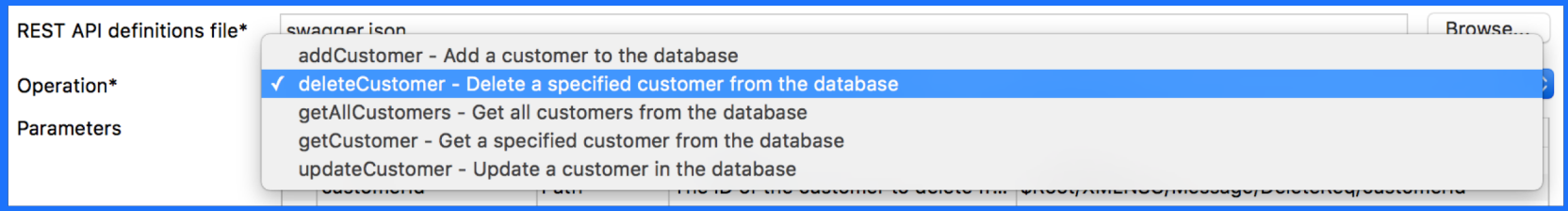


- Swagger documents supported in both available formats (JSON and YAML)

REST Request node and Swagger

- As before, you supply the REST Request node with a Swagger file:

- The REST Request node uses then uses the information stored in the Swagger file to determine what HTTP call to make:
- What URL to use (host name, port, SSL/TLS?)
- What path to use
- How to encode parameters (path, query, header?)
- You simply pick which operation you want to call and it does the rest!



REST Request node configuration

- Well… almost! If the operation requires one or more parameters, then you will need to pass these into the REST Request node.
- These can be supplied as XPath or ESQL expressions on the “Parameters” table:

Parameters

Name	Type	Description	Expression
Authorization	Header	Provide the authorization key that...	'suchASecretAuthKey'
customerId	Path	The ID of the customer to delete fr...	$Root/XMLNSC/Message/DeleteReq/customerId
clientName	Query	Provide the authorization key that...	LocalEnvironment.Variables.CLIENT_NAME

- You can specify hard-coded literals (strings, numbers, or booleans).
- You can also extract information from the input message (message, local environment, environment, exception list).

REST Request node configuration

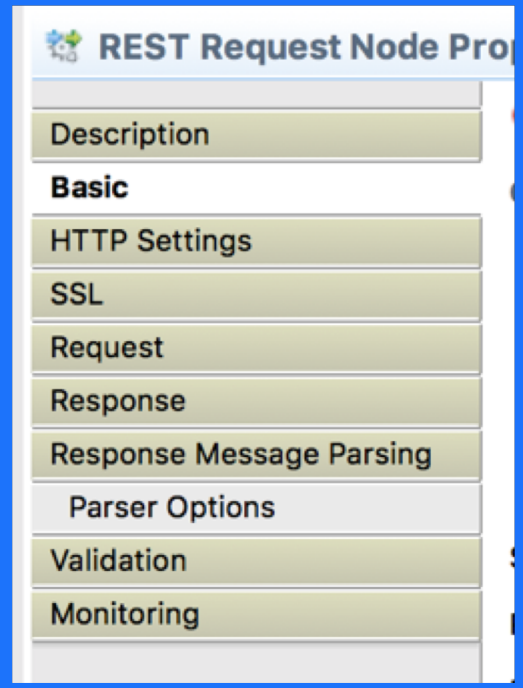
- You do not have to fill in the expressions on the “Parameters” table.
- Even if the parameter is marked as required in the Swagger document!
- As an alternative, you can specify the parameter values in the Local Environment – there is a new folder for REST Request node overrides:

```
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.Authorization = 'suchASecretAuthKey';
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.customerId = 12345;
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.clientName = 'LE override client';
```

- It is an error to not provide a value for a parameter that is marked as required in the Swagger document:
- BIPxxxxE: No value could be found for the required parameter 'clientName'.

REST Request node configuration

- The other node options are familiar:
  - HTTP settings
  - SSL/TLS settings
  - Parser choice and parser options
  - Input/output message tree locations
  - Validation
  - Monitoring
- Looking suspiciously like an HTTP Request node?
  - It is under the covers
  - Irrelevant/legacy options have been removed



REST Request node request message body

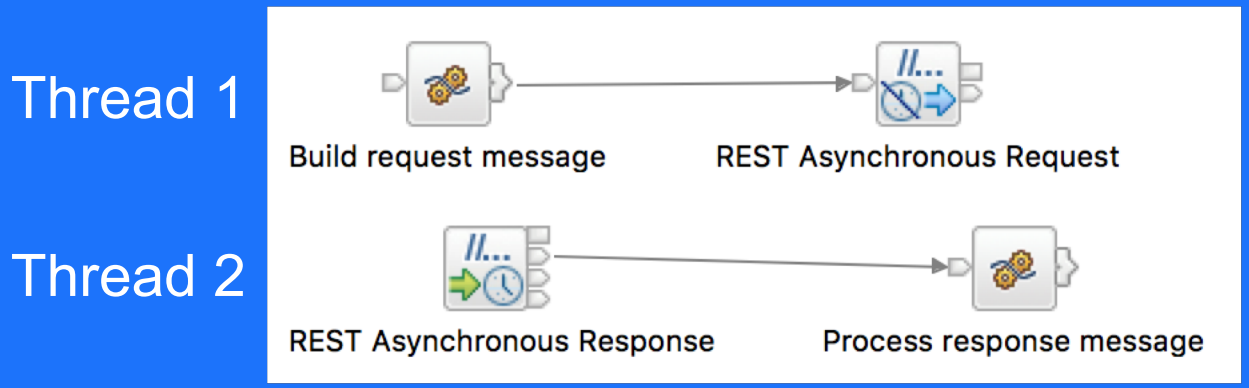
- The request body will be sent from the input message as per any other request node.
  - The Content-Type header defaults to a sensible one for the parser (application/json for JSON, text/xml for XML/XMLNS/XMLNSC).
  - The Swagger document can specify supported Content-Type values.
  - Possible additional property to choose the correct Content-Type from the Swagger document that represents the request message.
- Can modify the request body location – for example, to send a JSON request that has been placed into the Environment tree.

REST Request node response message body

- The response body will currently be parsed using the parser specified on the REST Request node.
  - Can enable ‘Automatic Content-Type handling’ for the node to pick the correct parser based on the value of the Content-Type header in the response.
- Swagger document can specify a list of supported Content-Type values for the responses – for example, REST APIs can come with the choice of JSON or XML responses:
  - The client (us!) specifies which one they want via the Accept header.
  - Can specify the desired response Content-Type
- Can modify the response body location – for example, to place a JSON response in the Local Environment tree.

Asynchronous REST Request nodes

- REST Async Request/Response nodes allows splitting or request and response processing into separate threads of execution



- Request and response processing is handled as separate transactions.

Outbound security for the REST Request node

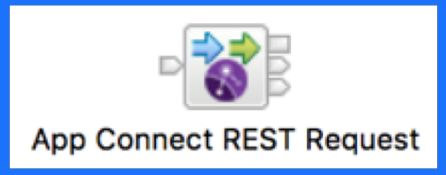
- REST APIs are most likely going to be secured:
  - HTTP Basic Authentication
  - API authorization key passed as header or query parameter
  - SSL/TLS client or mutual authentication
- Security identity property takes a mqsisetdbparms reference which can be either username/password and/or API Key

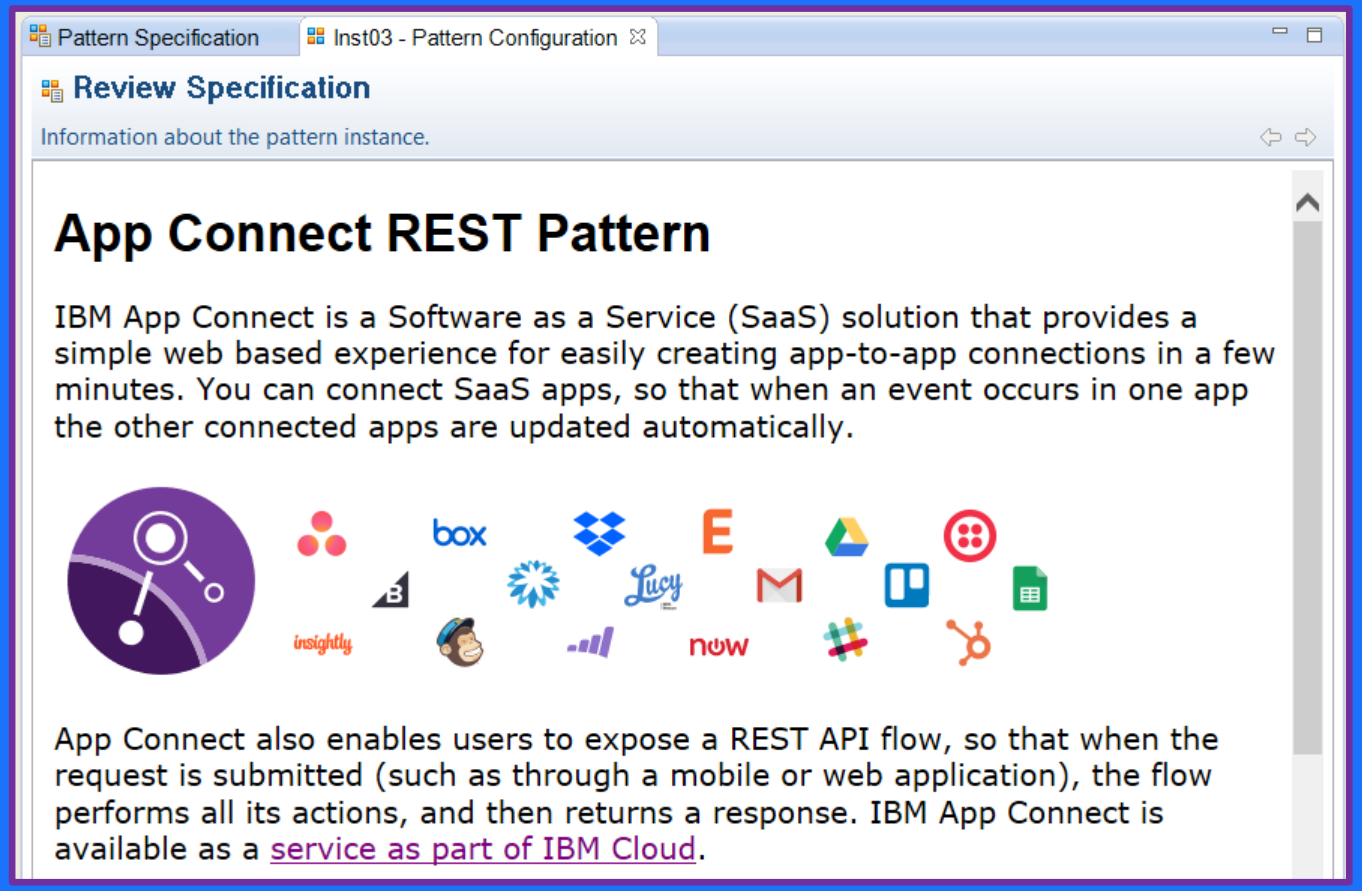
  IBNODE -n rest::mySecurityIdentity -u myRESTUserID -p myRESTPassword -k myRESTAPIkey
- REST API swagger will define if the API needs an basic authentication or API Key which the node will then use
- HTTPS configured as per normal HTTPRequest node

Consuming App Connect REST API in App Connect Enterprise

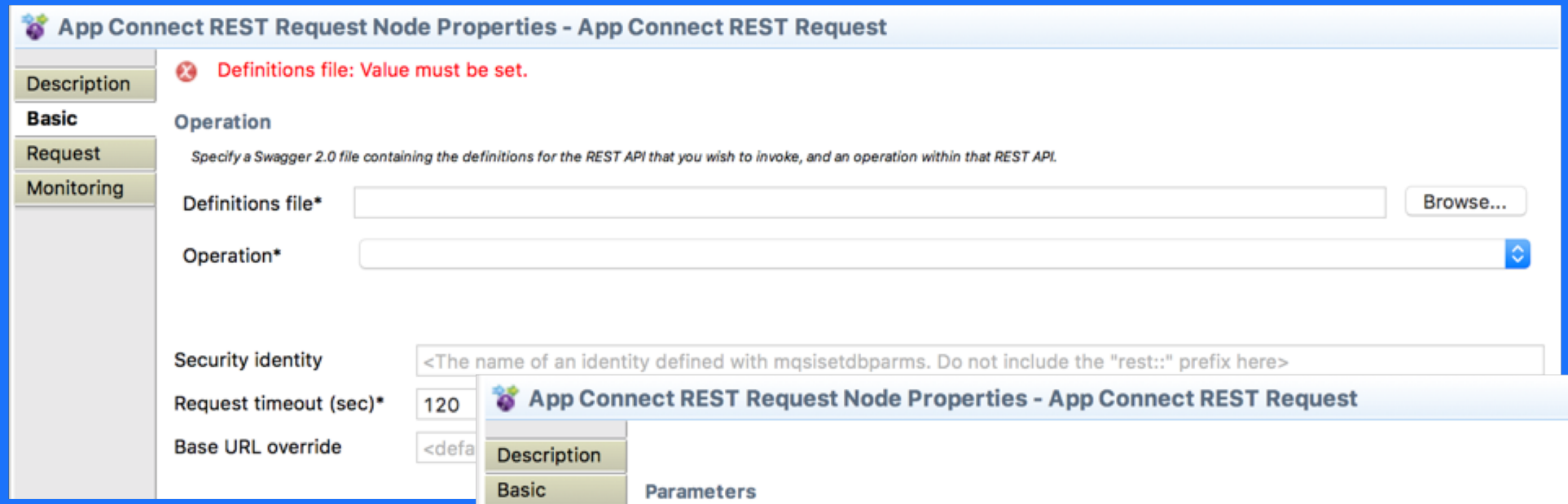
App Connect RESTRequest node + pattern

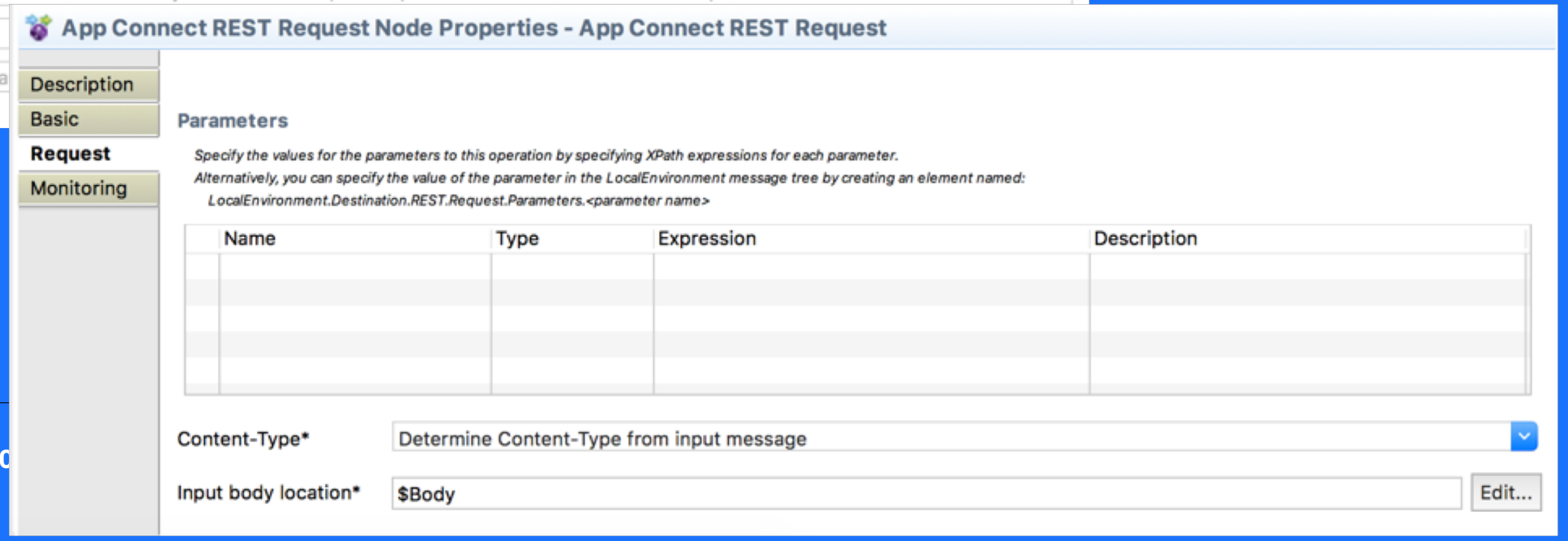
- The App Connect REST Request node is available to issue requests to an IBM App Connect REST API
- The node uses an imported Swagger document (in either JSON or YAML format), which defines the REST API and the operations that are available
- App Connect REST Pattern available to help accelerate flow generation





App Connect RESTRequest node





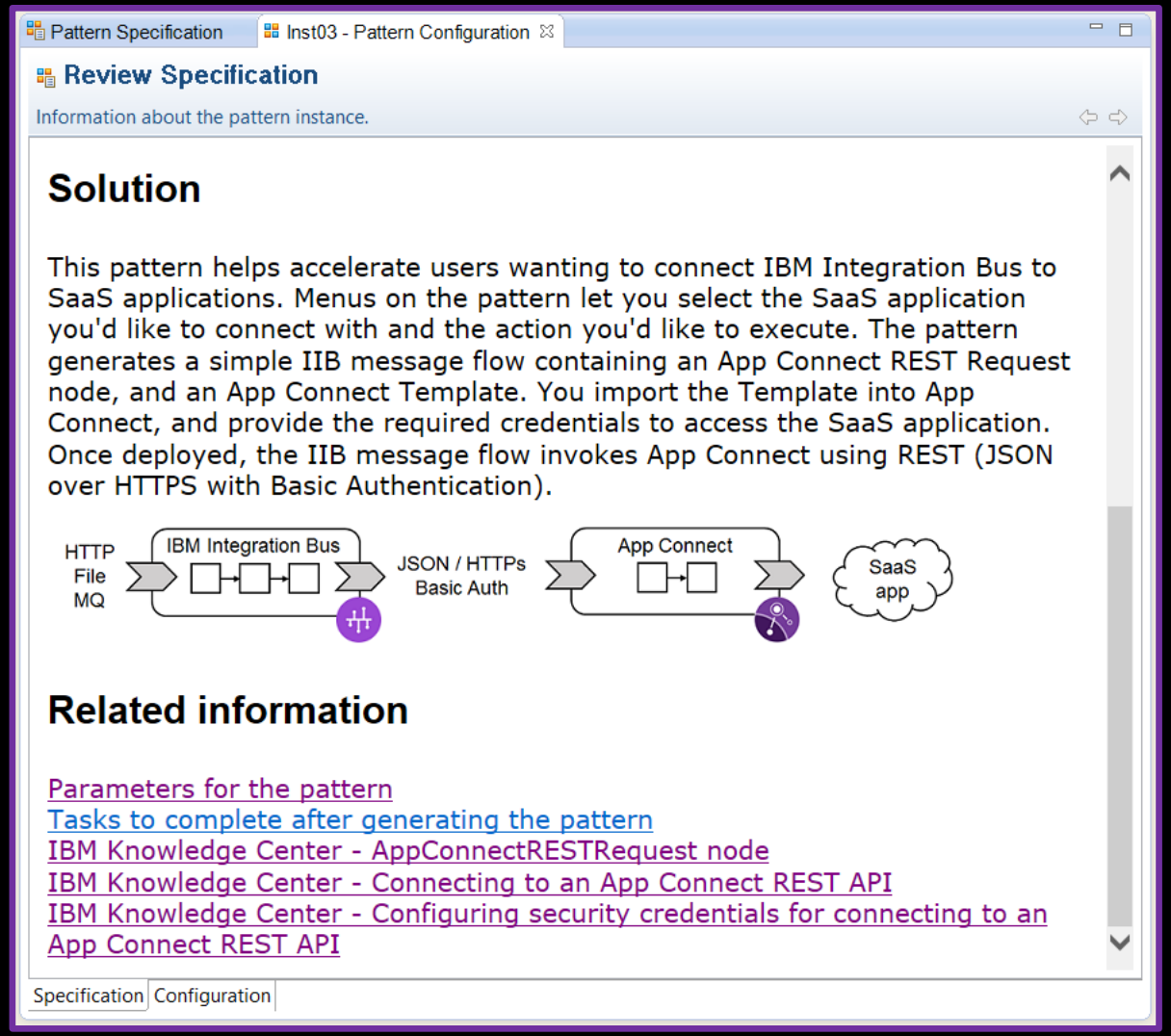
Reduced set of parameters compared to normal RESTRequest node as we already know some of the answers!

App Connect REST pattern

Available in toolkit pattern explorer

Pattern allows you to choose a transport to wrapper the RESTRequest node with

An App Connect flow template is then generated for you to import into App Connect to build out the SaaS connectivity

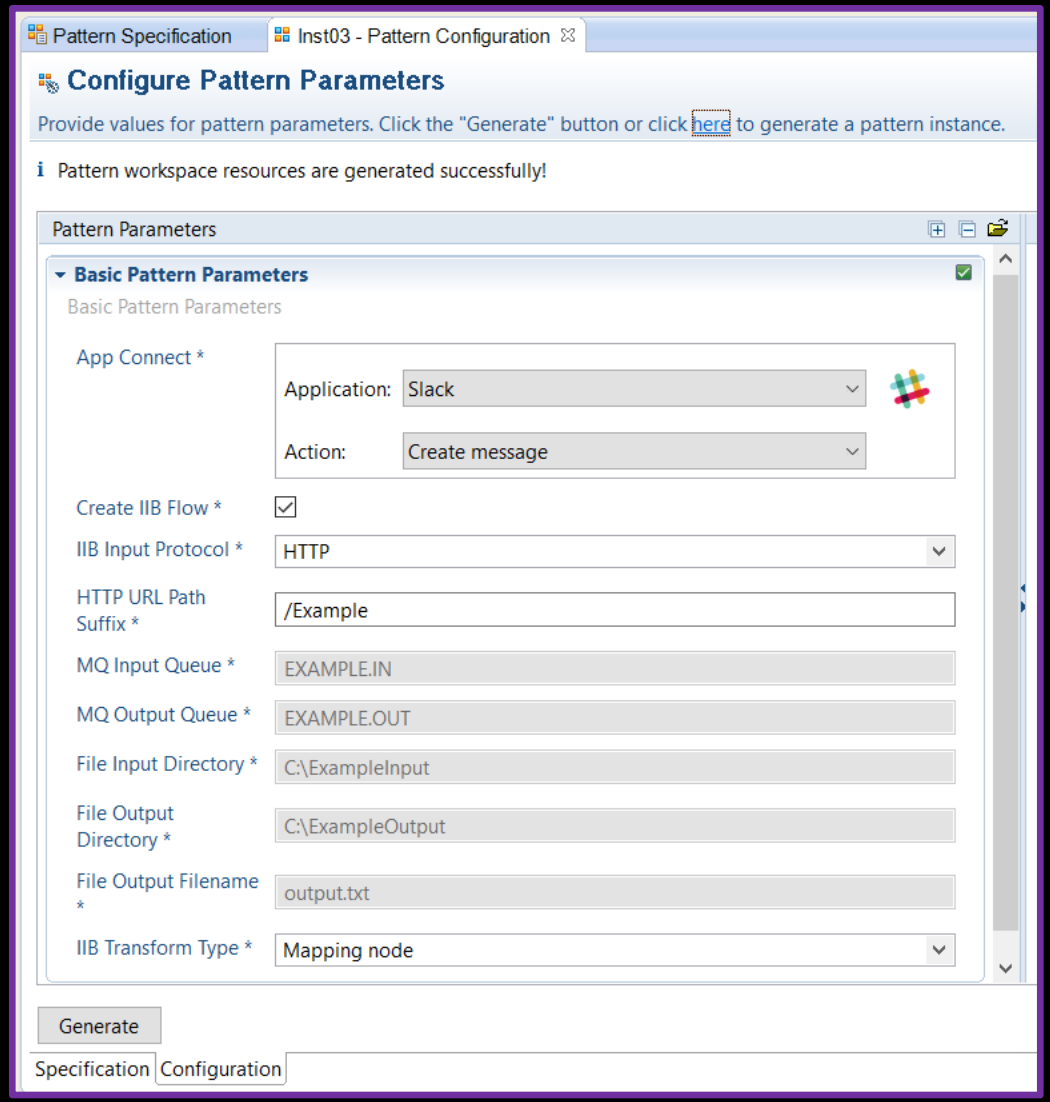


App Connect REST pattern

Select SaaS Application

Select Action

Define style of IIB flow



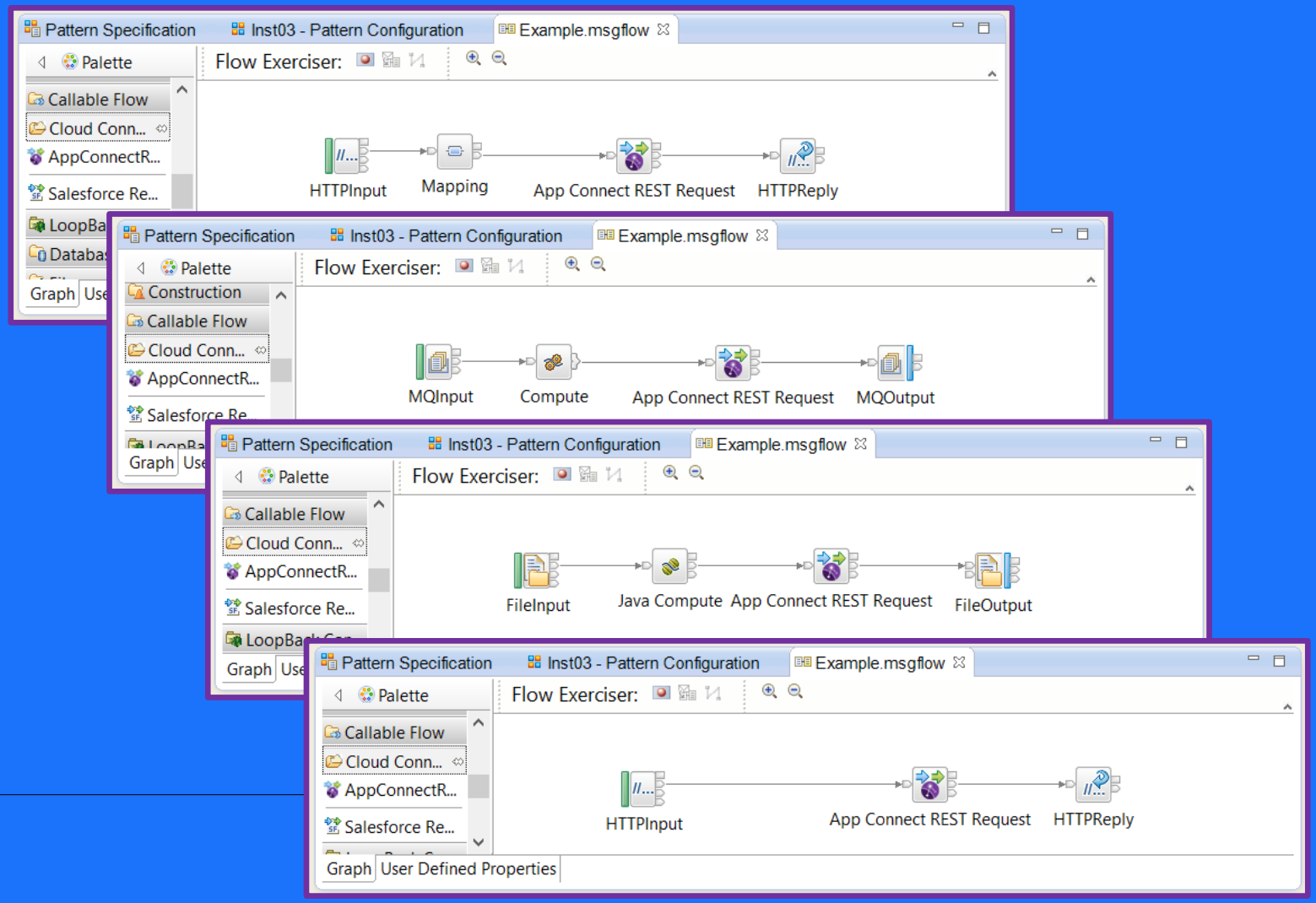
App Connect REST pattern

Optionally generate an IIB flow

Choose IIB Input Protocol
HTTP / MQ / File

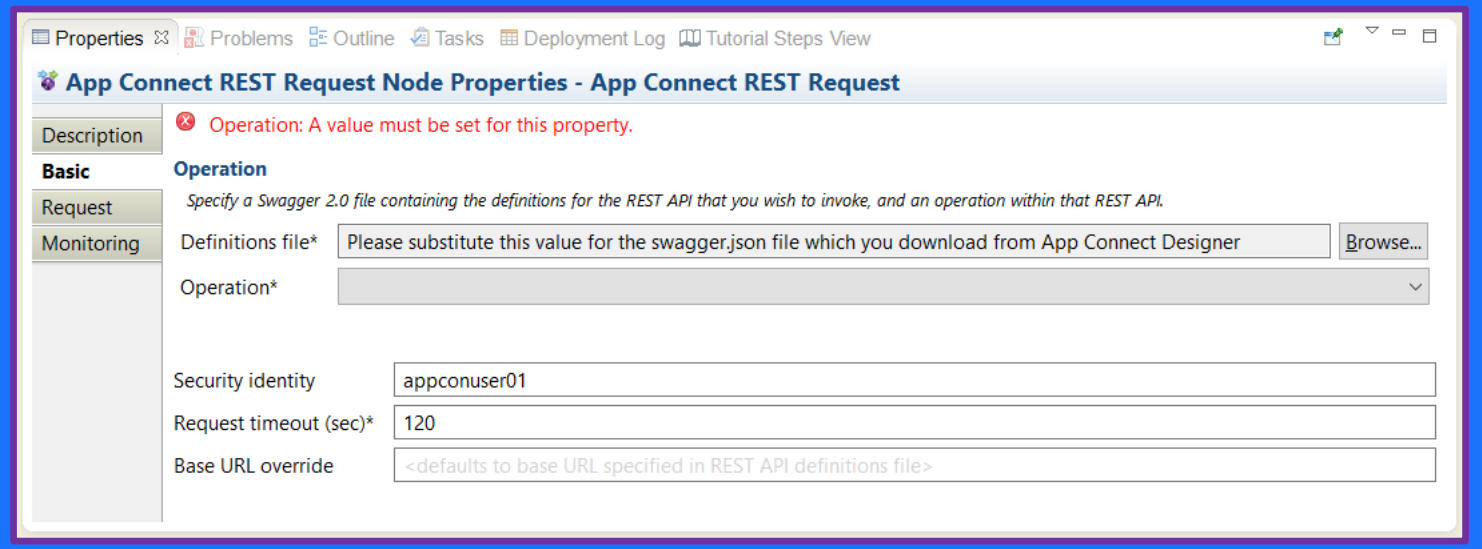
Choose IIB Transform Type
Map / ESQL / Java / None

Transform node provides simple assignment of values

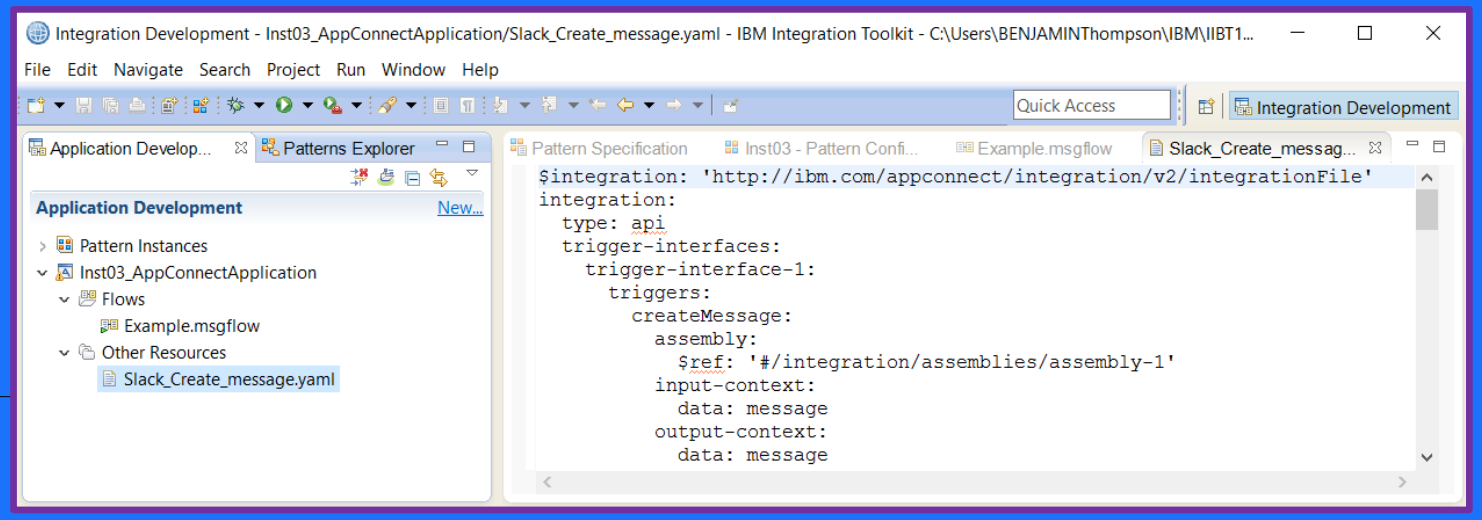


App Connect REST pattern

The App Connect REST Request node in the generated flow needs a Swagger file from App Connect

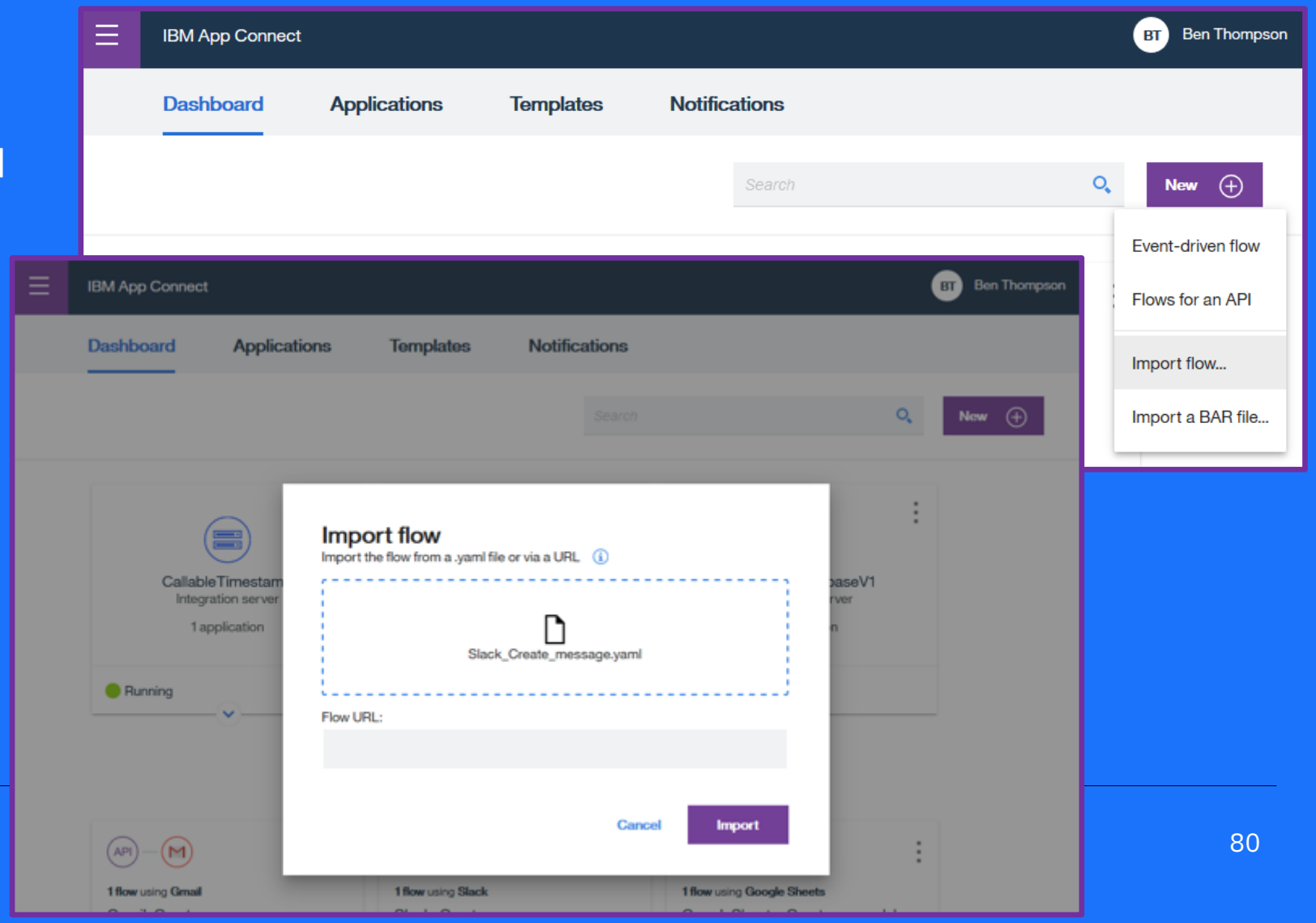


An App Connect template yaml is generated in the IIB Toolkit



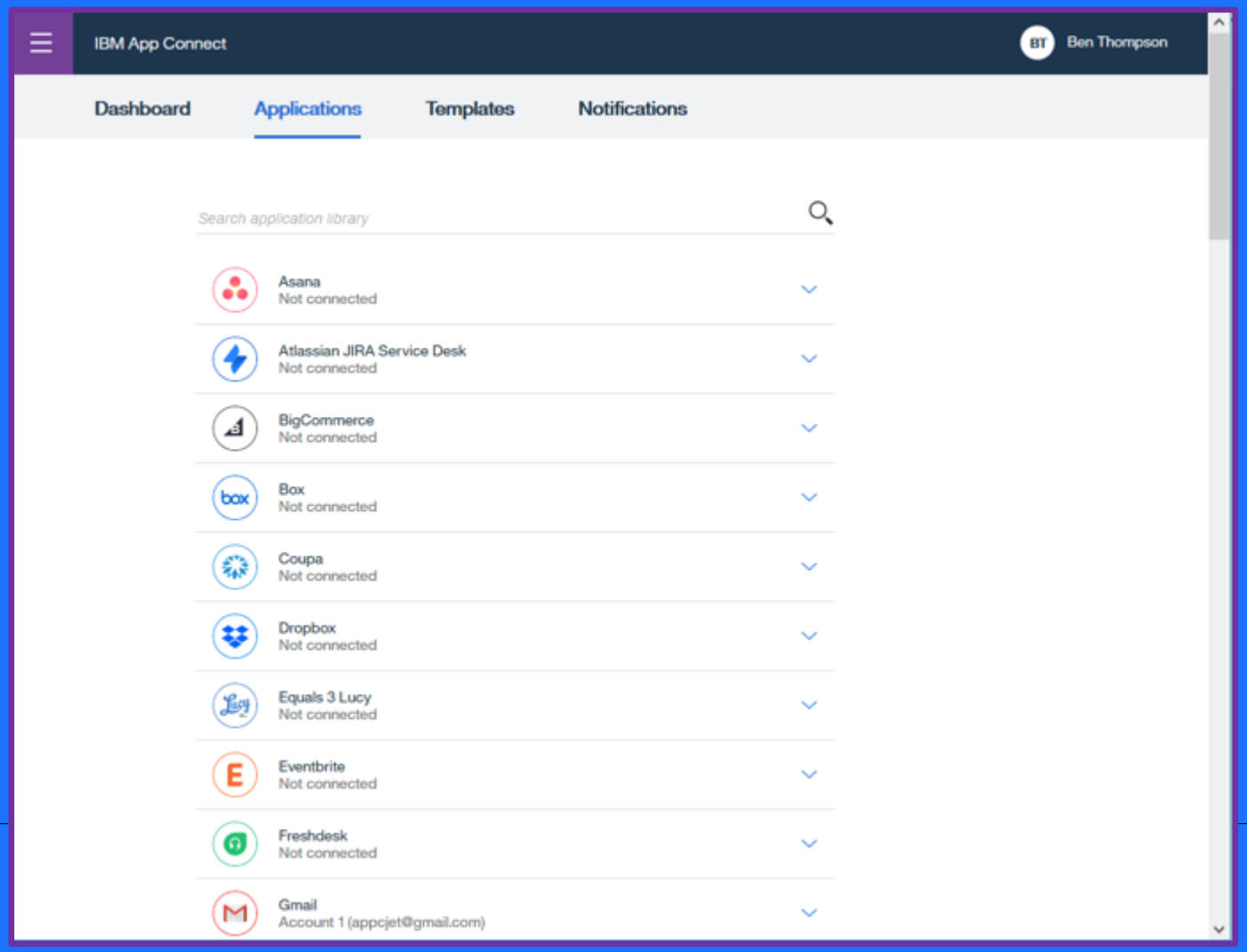
App Connect REST pattern

Import the generated App Connect template into the Designer dashboard



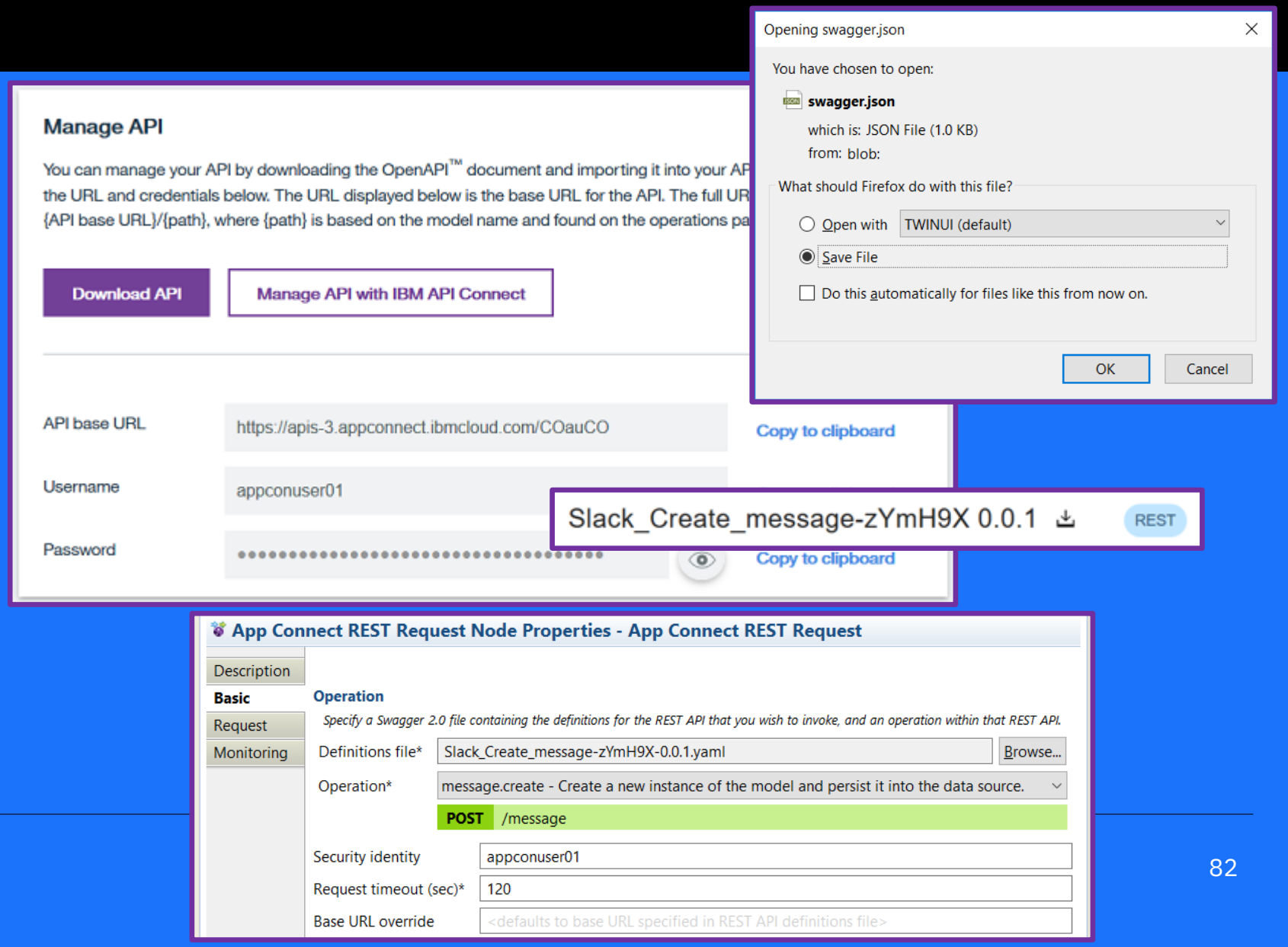
App Connect REST pattern

Use the Applications section of the Designer interface to specify the account credentials for the SaaS application you are connecting to

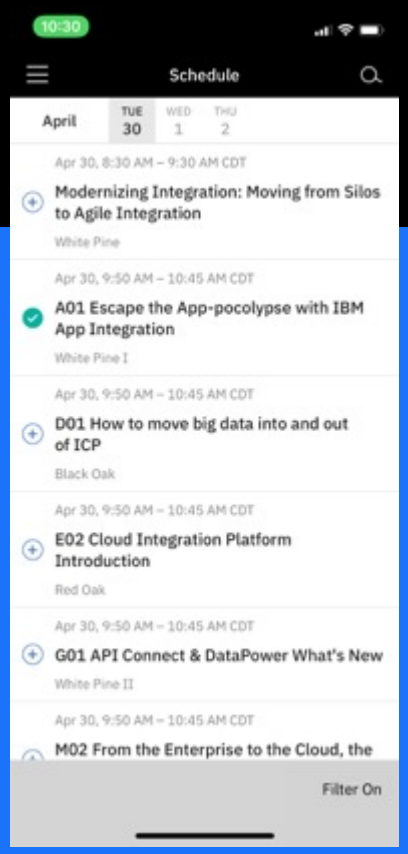


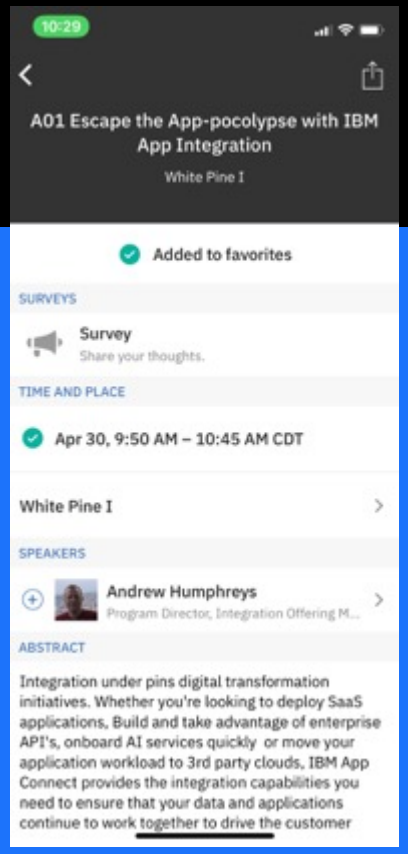
App Connect REST pattern

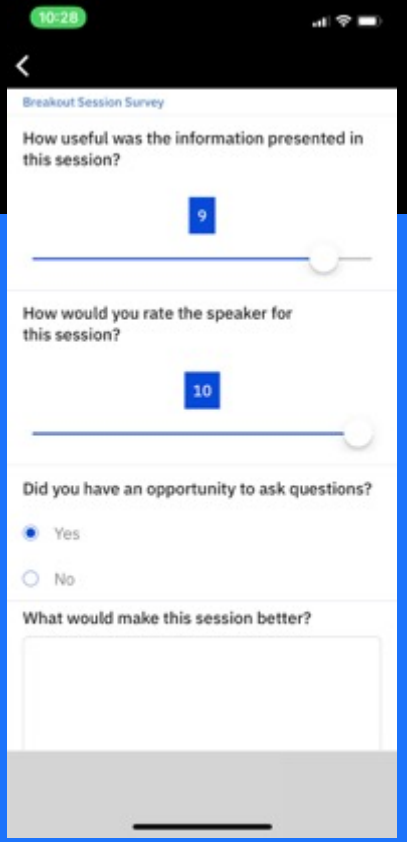
Download the swagger.json interface describing the App Connect REST API, copy it into your IIB Toolkit workspace and associate it with the App Connect REST Request node.



Questions ?







Don’t forget to fill out the survey!

Select your session, select survey, rate the session and submit!

Thank You