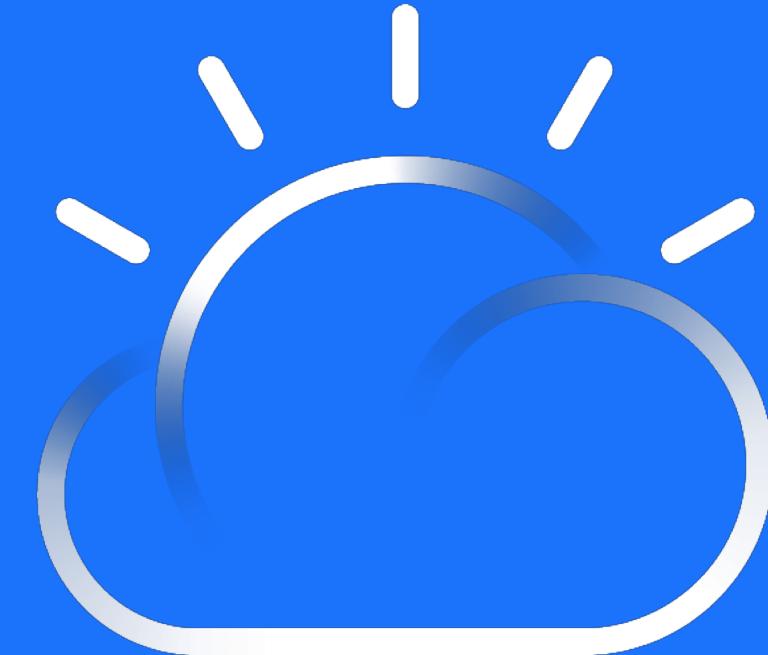


A12 Application Integration

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

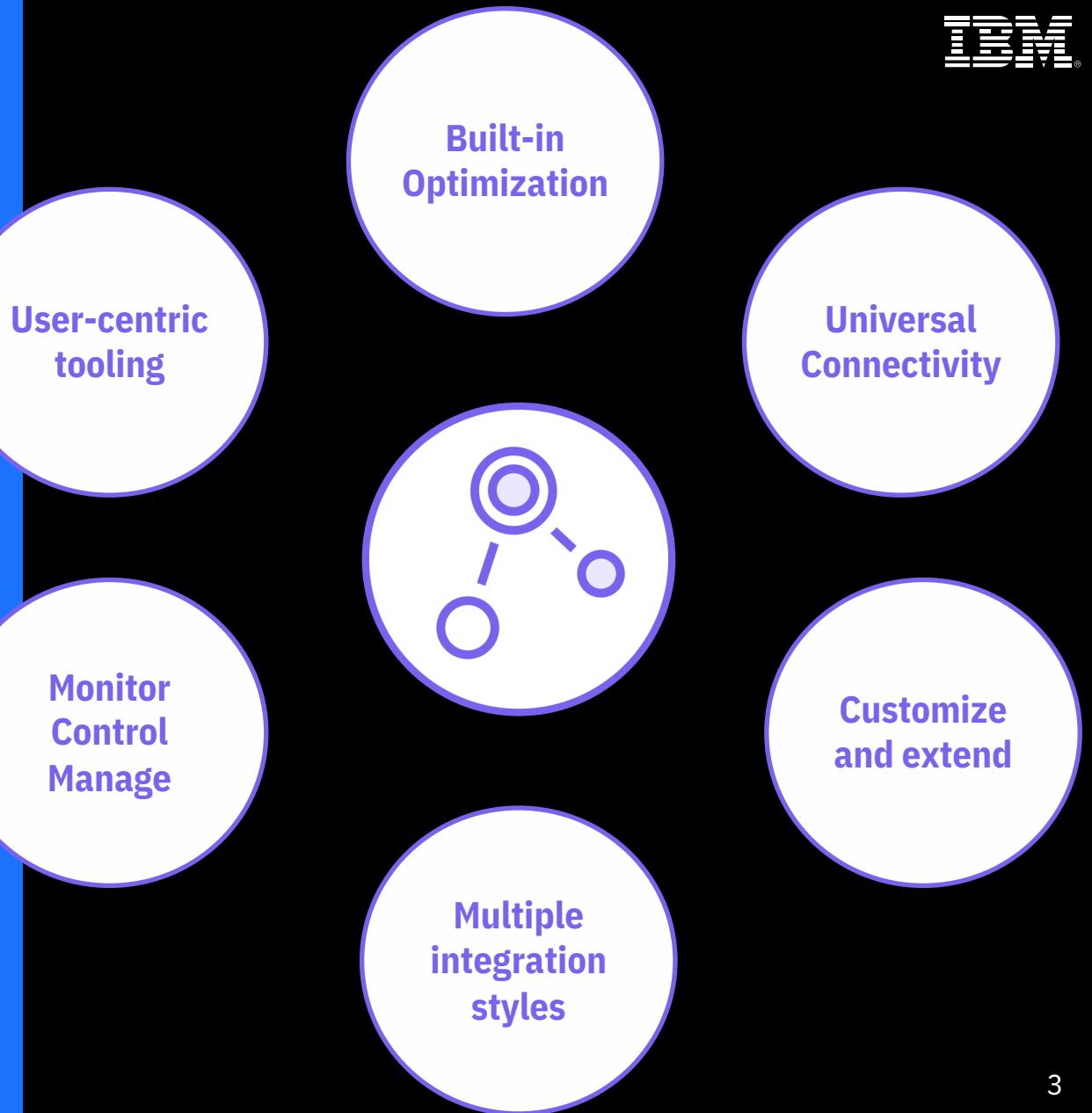
IBM App Connect



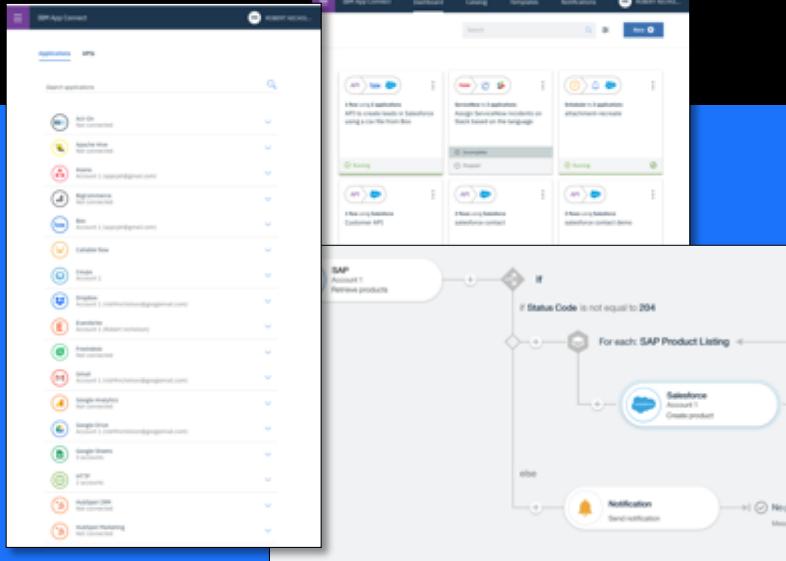
IBM App Connect brings together
the best of the existing,

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

The screenshot displays the IBM App Connect interface, specifically the 'Define' tab for a 'Product and Product Catalog API'. It shows:

- Create model:** A table for defining properties of a 'Product_catalog' model, listing fields like id, name, description, weight, cost, location, available_qty, and available.
- Operations:** A section showing pre-defined operations for 'Product_catalog' and 'Product'. For 'Product_catalog', operations include 'Create Product_catalog' (POST /Product_catalog), 'Replace or create Product_catalog by ID' (PUT /Product_catalog/{id}), and 'Select an operation to add'. For 'Product', operations include 'Create Product' (POST /Product), 'Retrieve Product by ID' (GET /Product/{id}), 'Replace or create Product by ID' (PUT /Product/{id}), and 'Delete Product by ID' (DELETE /Product/delete_product).
- Flow Diagram:** A visual representation of the API flow titled 'Product and Product Catalog API'. It shows a sequence starting with 'Request' (represented by a person icon), followed by 'IBM DB2 Account 1' (represented by a database icon), and ending with 'Response' (represented by an arrow icon). Below the flow, there's a configuration section for 'Retrieve PRODUCT records' with fields for 'Where' (e.g., PID equals id) and 'Add condition'.

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

The screenshot shows a 'Create model' interface for a 'Product' model. The 'Properties' tab is selected. Two properties are defined: 'id' (String) and 'name' (String). A button 'Select properties from applications...' is highlighted with a blue box. A modal window titled 'Manage' is open, showing a list of applications and their properties. The 'ServiceNow Account 1' application is selected, and its properties are listed below. Several properties are highlighted with red boxes: 'approval', 'approval_history', 'approval_set', 'assigned_to', 'assignment_group', and 'business_duration'. A blue arrow points from the 'Select properties from applications...' button to the 'ServiceNow Account 1' application in the modal.

Create model

Product Properties Operations ...

Add properties to your Product model

id String
name String Date Object Array of strings Array of numbers Array of booleans

+ Add property Select properties from applications...

Manage

Select properties from your applications:

IBM Watson Tone Analyzer ServiceNow Account 1 3

Asset Attachment Comments Department Incident

Select All Deselect All

active	"Active"
activity_due	"Activity due"
additional_assignee...	"Additional assignm..."
approval	"Approval"
approval_history	"Approval history"
approval_set	"Approval set"
assigned_to	"Assigned To"
assignment_group	"Assignment group"
business_duration	"Business duration"

Cancel Add properties

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.

The screenshot shows the 'Define' tab of the 'Product and Product Catalog API' dashboard. Under the 'Operations' tab for the 'Product_catalog' model, several operations are listed: 'Create Product_catalog' (POST /Product_catalog), 'Replace or create Product_catalog by ID' (PUT /Product_catalog/{id}), 'Retrieve Product_catalog by ID' (GET /Product_catalog/{id}), 'Replace or create Product_catalog with filter' (PUT /Product_catalog/filter), 'Add a custom operation' (POST /Product_catalog/custom), 'Retrieve Product by ID' (GET /Product/{id}), 'Replace or create Product by ID' (PUT /Product/{id}), and 'Delete Product by ID' (DELETE /Product/delete_product). A blue arrow points from the 'auto-generated' label in the bottom screenshot to the 'Replace or create Product_catalog by ID' operation.

The screenshot shows the 'Request' tab for the 'Retrieve Product_catalog by ID' operation. It includes a 'Request URL example' field containing '.../Product_catalog/sample_id'. A blue arrow points from the 'auto-generated' label in the bottom screenshot to the 'Request' tab.

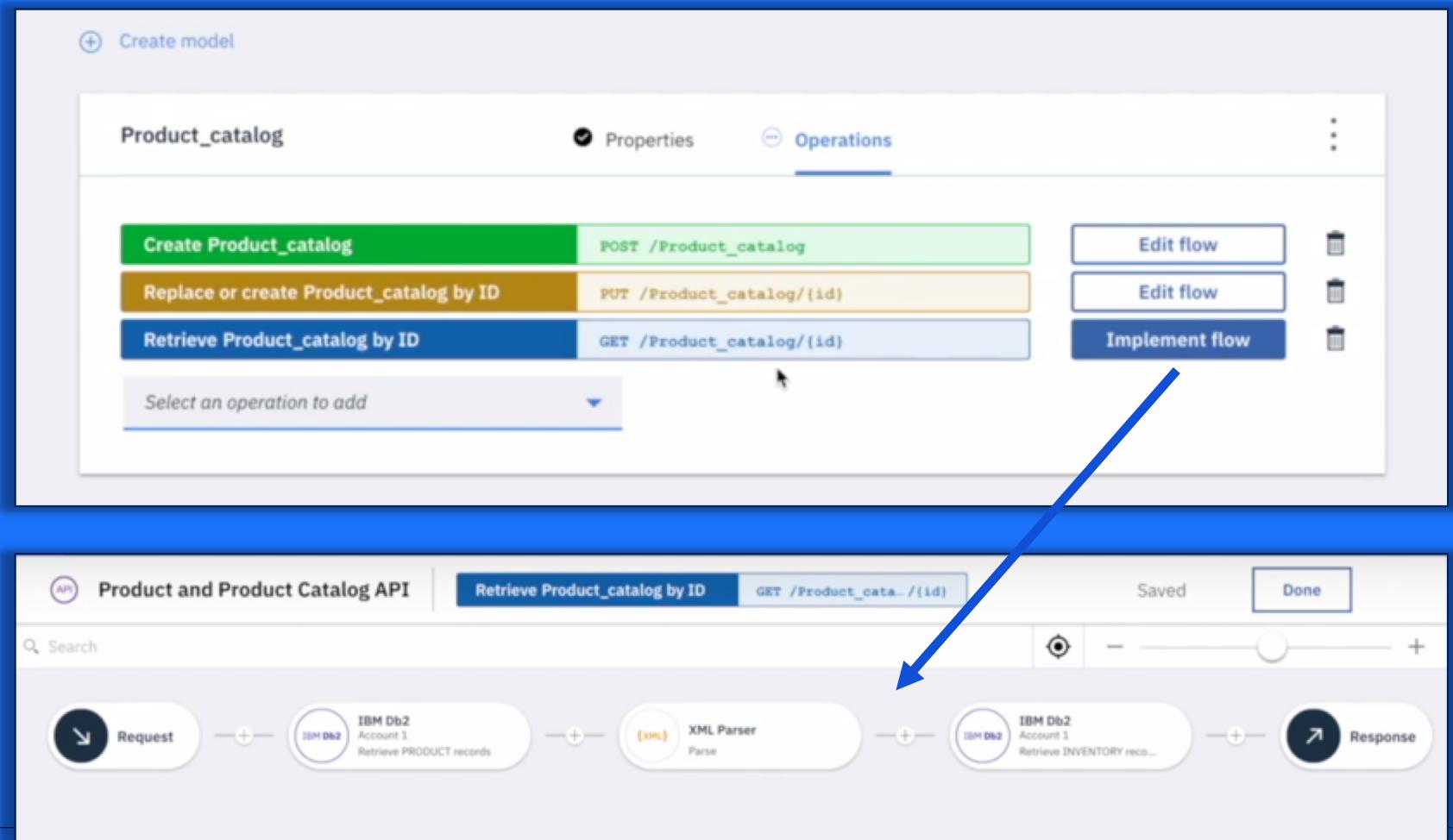
Implement business logic for API operations using flow editor



click, drag and drop
configuration
no coding

Implement business logic using any of:

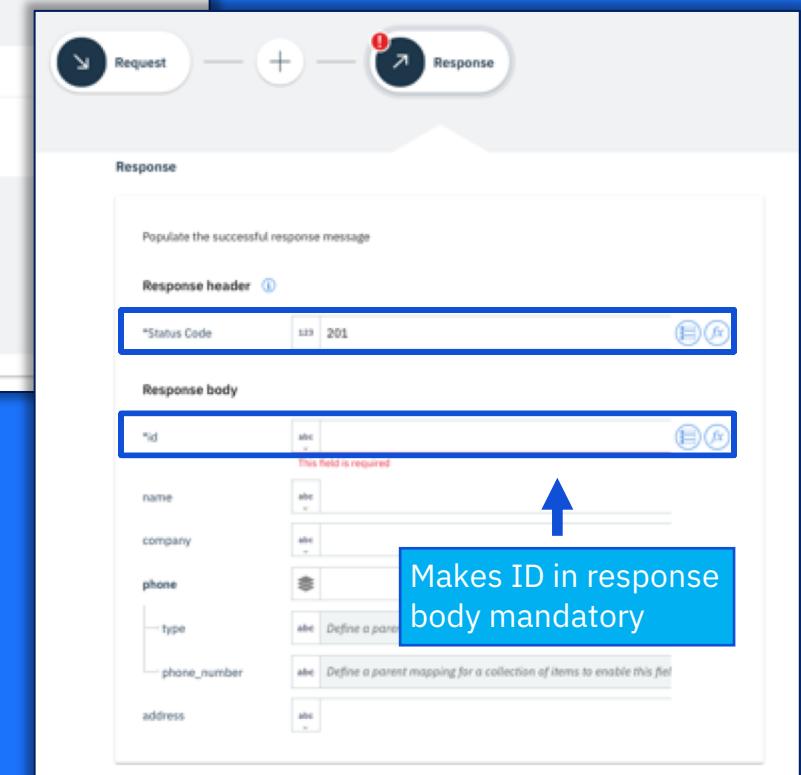
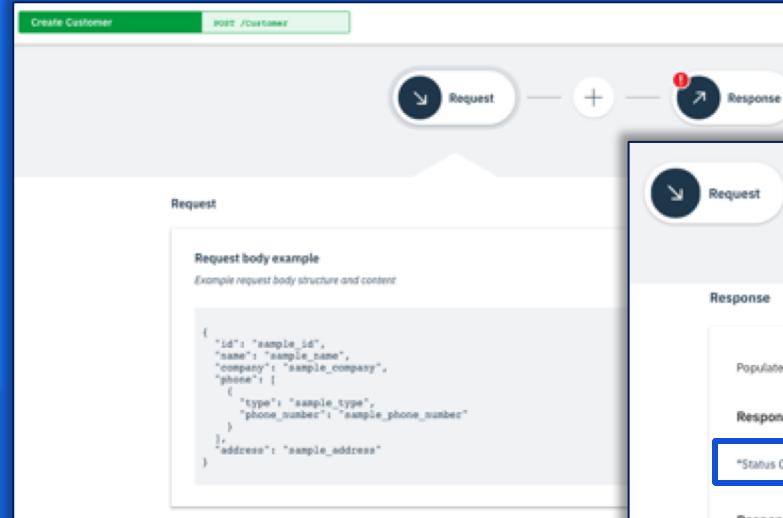
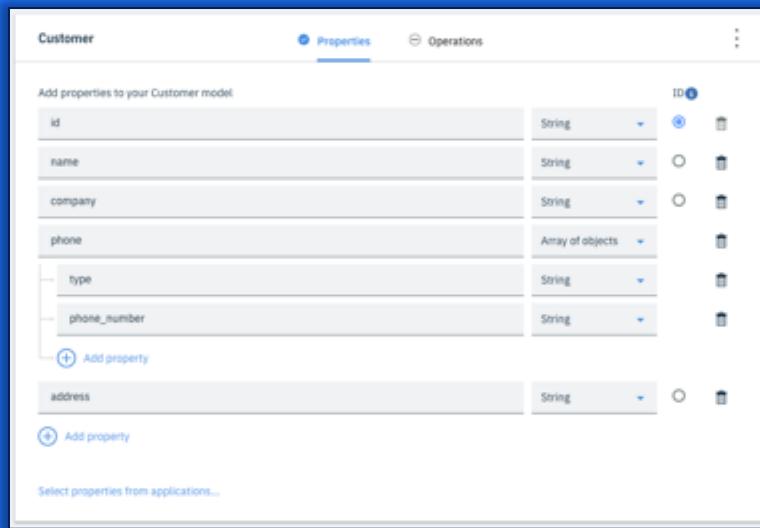
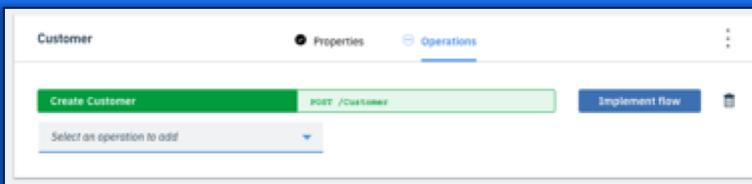
- connectors
- flow constructs
- mapper
- data transformation



POST

IBM

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



GET by ID

IBM

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

Customer

Operations

Retrieve Customer by ID

GET /Customer/{id}

Implement Flow

Select an operation to add

Customer

Properties

Add properties to your Customer model

id	String
name	String
company	String
phone	Array of objects
type	String
phone_number	String
+ Add property	
address	String
+ Add property	
Select properties from applications...	

Request

Request URL example

Example request URL content and syntax

/Customer/sample_id

Response

Populate the successful response message

Response header

Status Code 123 200

Response body

id	abc
name	abc
company	abc
phone	abc
type	abc Define a parent mapping for a collection of items to enable this field
phone_number	abc Define a parent mapping for a collection of items to enable this field
address	abc

GET by WHERE



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

The screenshot shows the Integration Techni interface. On the left, the 'Customer' model is displayed with properties: id (String, ID), name, company, phone, type, phone_number, address, and a 'Select an operation to add' section. On the right, a 'Retrieve Customer with filter' operation is configured with a GET request to '/Customer?name=SampleName&company=SampleCompany'. It includes filter parameters 'name' and 'company'.

Integration Techni

The screenshot shows a flow titled 'Retrieve Customer with filter' with a GET request to '/Customer?name=SampleName...'. The flow consists of two steps: 'IBM Db2 Account 2' (Request) and 'IBM Db2 Account 2' (Response). A callout box with a blue border and arrow points to the 'Mapper mode' dropdown in the 'IBM Db2 Account 2' step, with the text 'Change the mapper mode to access filter options'.

A second callout box with a blue border and arrow points to the 'name' field in the 'equals' condition of a 'Where' clause, with the text 'Handle filter parameters exceptions'.

A third callout box with a blue border and arrow points to the 'name' field in the 'exists' condition of a 'Where' clause, with the text 'Use filter parameters in flow logic to query your endpoint'.

Retrieve Customer with filter
GET /Customer?name=SampleName...

Request

Request URL example
Example request URL content and syntax

Filter supported:

- .../Customer?name=SampleName&company=SampleCompany
- .../Customer?name=SampleName
- .../Customer?name=SampleName&company=SampleCompany

IBM Db2 Account 2
Retrieve CONTACT records

Response

Change the mapper mode to access filter options

Advanced and filter mode

name

Handle filter parameters exceptions

Use filter parameters in flow logic to query your endpoint

GET with pagination

IBM

supports skip, limit and token

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

repository

Properties Operations

Retrieve repository with filter GET /repository?token=tokenValue&limit=

Select filter properties for repository to be retrieved:

- name
- full_name

Enable pagination for retrieved repository

Paginate results using:
 'skip' and 'limit' parameters
 'token' and 'limit' parameters

Select an operation to add

repository

Properties Operations

Add properties to your repository model

- uuid
- name
- full_name

Add property

Select properties from applications...

Integration Technical Con

Request

Request URL example

Example request URL content and syntax

Filter supported:

- .../repository
- .../repository
- .../repository

The above filter formats

```
{ "token": "tokenValue", "limit": "limitValue", "where": {} }
```

Invoke method

Populate the target fields in HTTP

*HTTP method

abc GET

*URL (fully qualified)

abc https://api.bitbucket.org/2.0/repositories?pagelen=

limit

&after=

token

Available inputs

Request header

Request body

Request filter parameters / Object

Object { }
token abc
limit 123

Map to object

NetToken

Continues at

Use pagination parameters in flow logic

pagination parameters available as source mapper value

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

The screenshot shows the 'Customer' model in the 'Operations' tab. A yellow bar at the top indicates the selected operation is 'Replace or create Customer by ID' with the URL 'POST /Customer/{id}'. Below this, there is a dropdown menu 'Select an operation to add' and a button 'Implement flow'.

The screenshot shows the 'Customer' model in the 'Operations' tab. A yellow bar at the top indicates the selected operation is 'Replace or create Customer with filter' with the URL 'POST /Customer?name={name}&company={company}'. Below this, there is a section 'Select filter properties for Customer to be replaced or created:' with fields for 'name', 'company', and 'address'. There is also a dropdown menu 'Select an operation to add' and a button 'Implement flow'.

The screenshot displays two detailed pages for the 'Replace or Create' operations. The left page is for 'Replace or Create Customer by ID' and the right page is for 'Replace or Create Customer with filter'. Both pages include sections for 'Request' and 'Response', 'Request URL example', 'Request body example', and 'Request body structure and content'. The 'Request body example' and 'Request body structure and content' sections show JSON examples. The 'Request body structure and content' section for the 'Replace or Create Customer by ID' operation shows:

```
{
  "name": "sample_name",
  "company": "sample_company",
  "phone": [
    {
      "type": "sample_type",
      "phone_number": "sample_phone_number"
    }
  ],
  "address": "sample_address"
}
```

The 'Request body structure and content' section for the 'Replace or Create Customer with filter' operation shows:

```
{
  "where": {
    "name": "sample_name",
    "company": "sample_company",
    "name": "sample_name"
  }
}
```

Both pages also mention 'Filter supported' and provide examples of how to use filters.

Build custom operations, because:



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

The screenshot shows the 'Operations' tab for a service named 'rocket'. A new operation named 'Launch Rocket' is being created. The details are as follows:

- Display name:** Launch Rocket
- Description:** This operation will trigger rocket to launch. This operation does not fit a typical CRUD pattern, hence the use of custom operation...
- HTTP verb:** POST
- Use rocket ID:** (unchecked)
- Operation name:** launch
- Query parameters:** [Add query parameter](#)
- Request body:** rocket_target
- Response body:** rocket_status

A blue arrow points from the 'Request body' field to a callout box containing the text: "auto-generates request and response body". Another blue arrow points from the 'Request body' field to the 'Request' panel on the right.

The 'Request' panel displays examples for both the URL and the body of the request message.

Request URL example:
Example request URL, content and syntax
.../rocket/:id/Launch

Request body example:
Example request body structure and content
{"id": "sample_id",
"co-ordinates": "sample_co-ordinates"}
}

The 'Response' panel displays examples for the response message, including a status code and a response body.

Response header:
Status Code: 200

Response body:
id:
target_obliterated:

Build custom operations, because:



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

The screenshot shows the 'Operations' tab for a service named 'rocket'. A green box highlights the 'Business semantics' section, which contains the operation name 'Launch Rocket' and its RESTful URL 'POST /rocket/launch'. A blue arrow points from the text 'Allows you to pick your HTTP verb' to the 'HTTP verb' dropdown, which is set to 'POST'. Another blue arrow points from the text 'Allows you to pick any combination of available models' to the 'Request body' and 'Response body' dropdowns, both currently set to 'rocket_target'.

This panel shows the 'Request' configuration. It includes fields for 'Request URL example' (e.g., '.../rocket/:id/launch') and 'Request body example' (e.g., '{"id": "sample_id", "co-ordinates": "sample_co-ordinates"}').

This panel shows the 'Response' configuration. It includes fields for 'Response header' (with 'Status Code' set to '200') and 'Response body' (with fields 'id' and 'target_obliterated' both set to 'abc').

Build custom operations, because:



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

Integration

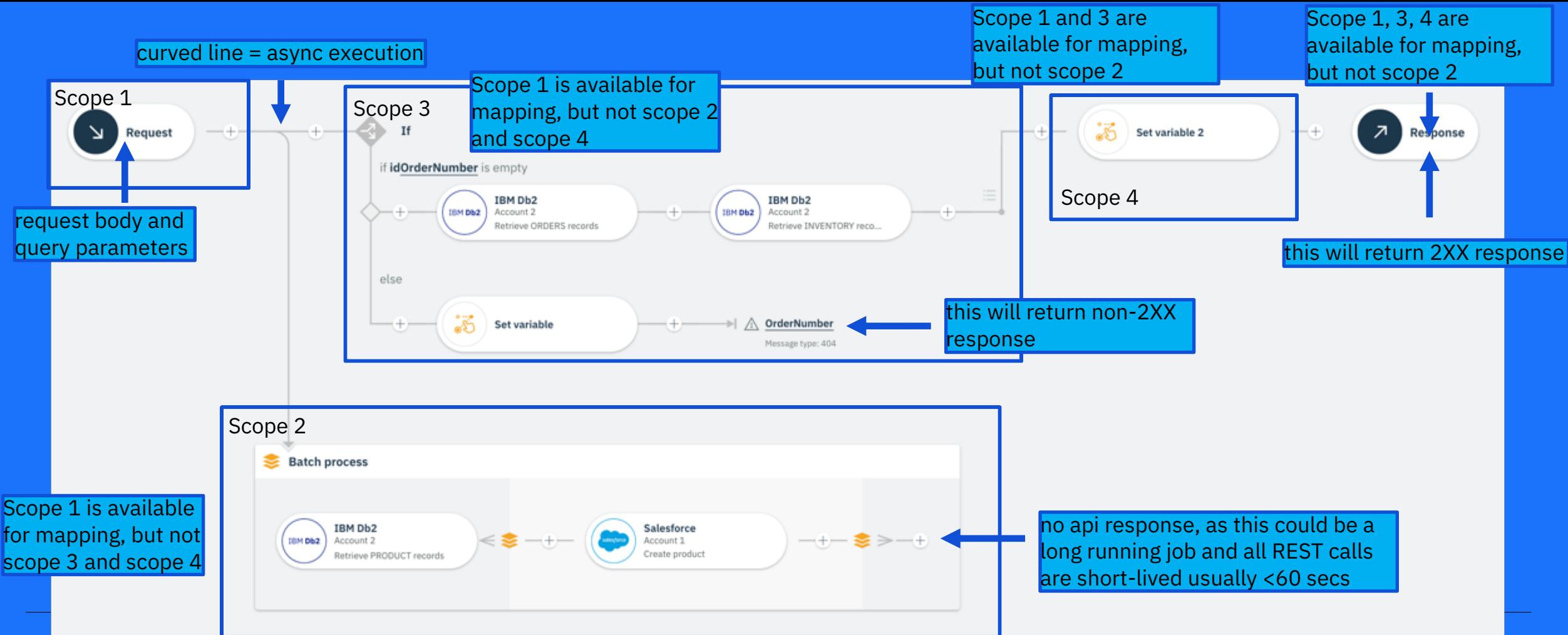
The screenshot shows the 'Operations' tab for a service named 'rocket'. A specific operation is highlighted with a green border and labeled 'Launch Rocket'. The URL for this operation is 'POST /rocket/:id/launch'. Below the URL, there is a checkbox labeled 'Use rocket ID' with a blue arrow pointing to it. A text box contains the annotation: 'This will enable instance ID in the URL'.

This screenshot shows the 'Request' configuration section. It includes fields for 'Request URL example' (containing '.../rocket/:id/launch') and 'Request body example' (containing a JSON object with 'id' and 'co-ordinates' fields).

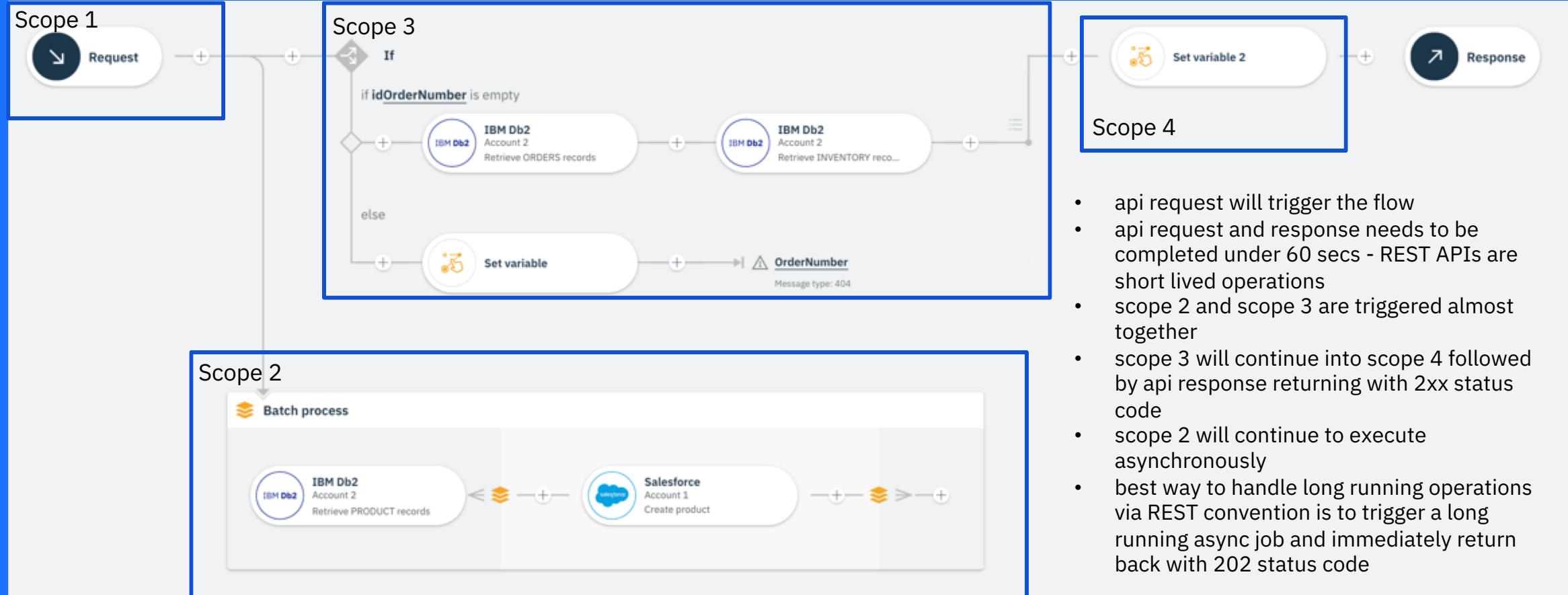
This screenshot shows the 'Response' configuration section. It includes fields for 'Response header' (with 'Status Code' set to '200') and 'Response body' (containing fields 'id' and 'target_obliterated').

Understanding the mechanics of API flow design scope of references

IBM



Understanding the mechanics of API flow design sequence of execution



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*

API Info

API definition

You can download the current API definition. It is also possible to open the API definition in API Connect, IBM's premier API management platform. If API Connect is not present within your IBM Cloud space, we will first provision the service with the free Lite plan.

Download YAML file

Download JSON file

Open in API Connect

Security and Rate Limiting

APIs

API definition

APIs

API definition

APIs

API definition

APIs

API definition

*Consume API in App Connect from
API Connect catalog*

APIs

API definition

APIs

API definition

APIs

API definition

APIs

API definition

Salesforce Account 1

New lead

reusableApI1-u3RkUQ Account 1

GET /reusableApI1/[id]

Populate the target fields in reusableApI1-u3RkUQ

Auto match fields

ID

Lead ID

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

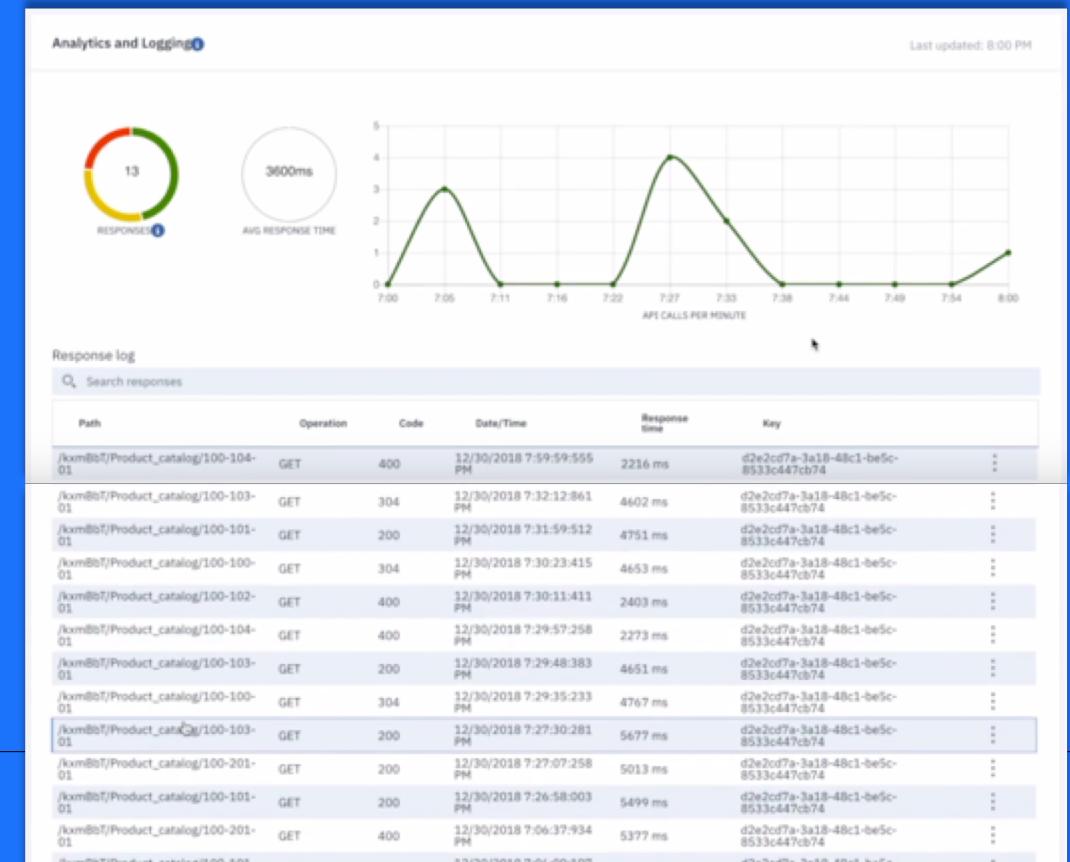
Click on manage to access embedded APIm capabilities

The screenshot shows the App Connect dashboard for the 'Product and Product Catalog API'. The 'Manage' tab is selected. Under the 'Product_catalog' section, there are three operations: 'Create Product_catalog' (POST /Product_catalog), 'Replace or create Product_catalog by ID' (PUT /Product_catalog/{id}), and 'Retrieve Product_catalog by ID' (GET /Product_catalog/{id}). Each operation has a 'View flow' button. Below this, under the 'Product' section, there are two operations: 'Create Product' (POST /Product) and 'Retrieve Product by ID' (GET /Product/{id}), also with 'View flow' buttons.

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

The screenshot shows the App Connect dashboard for the 'Product and Product Catalog API'. The 'Manage' tab is selected. A message at the top says 'To get started, scroll down to "Sharing Outside of Cloud Foundry organization" and click on "Create API key". Follow the "API Portal Link" link to explore the API in the portal and invoke it by clicking on "Try it!"'. Below this, the 'API name' is listed as 'Product and Product Catalog ...' and the 'Route' is listed as 'https://service.us.apiconnect.ibmcloud.com/gws/api/gateway/api/d7874e3029937583c7e82fce4bd3e308561b2bd8d37c6...'. There is a 'Copy' button next to the route URL.

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

API Info

API definition

You can export the current API definition to file. It is also possible to open the API definition in API Connect, IBM's premier API management platform. If API Connect is not present within your IBM Cloud space, we will first provision the service with the free Lite plan.

Security and Rate Limiting

Rate limiting

When rate limiting is enabled, API calls failing outside of the limit will be rejected and response code 429 will be returned. Given that rate limiting is on a per-key basis, application authentication must be enabled.

The leaky bucket algorithm is used to prevent sudden bursts of invocations of your API. For example, if you set your limit as 1000 calls per second, then after 10 seconds, the API will start rejecting requests.

Limit API call rate on a per-key basis

Maximum calls: 1000

Unit of time: Second

Sharing within Cloud Foundry organization

Include API in organization-level Shared APIs view

API keys created by IBM Cloud users

Name	Space	API key
No keys created by users in your Cloud Foundry organization		

Create API key

Sharing Outside of Cloud Foundry organization

API keys and portal links

Name	API key	API Portal Link
share with mobile app developers	d2e2cd7a-3a18-48c1-be5c-8533c447cb74	https://service.us.apiconnect.ibmcloud.com/fusion/devportal/corral?artifactId=accon...

Create API key

Security and Rate Limiting

Application authentication

You can require consuming applications to authenticate using API key and secret or API key alone.

Require applications to authenticate via API key

Method: API key only

Location of API key and secret: Header

Parameter name of API key: X-IBM-Client-Id

Parameter name of API secret: X-IBM-Client-Secret

OAuth user authentication

You can control access to your API through the OAuth 2.0 standard. First require an end user to log in via IBM Cloud App ID, Facebook, GitHub, or Google. Then include the corresponding OAuth token in the Authorization header of each API request. The authenticity of the token will be validated with the specified token provider. If the token is invalid, the request will be rejected and response code 401 will be returned.

Require users to authenticate via OAuth social login

Provider: IBM Cloud App ID

App ID service: Create an App ID service

CORS

Enabling cross-origin resource sharing (CORS) will allow embedded scripts in a web page to call the API across domain boundaries.

Enable CORS so that browser-based applications can call this API

Embedded API management functions



how to use developer portal to test your API

Sharing within Cloud Foundry organization
Include API in organization-level Shared APIs view

API keys created by IBM Cloud users

Name	Space
share with mobile app developers	d2e2cd7a-3a3b-48c1-be5c-8533c447cb74

Sharing Outside of Cloud Foundry organization
Create API key

Name	API key	API Portal Link
share with mobile app developers	d2e2cd7a-3a3b-48c1-be5c-8533c447cb74	https://service.us.apiconnect.ibmcloud.com/fusion/devportal/portalPart?factId=aacon...

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

Product and Product Catalog API-
kxmBbT 0.0.1

Operations by name

- Product.create
- Product.findbyId
- Product.prototype.patchAttributes
- Product.delete_product
- Product_catalog.create
- Product_catalog.findbyId
- Product_catalog.prototype.patchAttributes

Product.create
Create a new instance of the model and persist it into the data source.

Product.findbyId
Find a model instance by [[id]] from the data source.

Product_catalog.create
Create a new instance of the model and persist it into the data source.

Product_catalog.findbyId
Find a model instance by [[id]] from the data source.

developer portal will allows you to test each implemented API operation

Operations by name

Product.create

Product.findbyId

Product.prototype.patchAttributes

Product.delete_product

Product_catalog.create

Product_catalog.findbyId

Product_catalog.prototype.patchAttributes

Definitions

Product

x-any

Product_catalog

Security

X-IBM-Client-Id: apiKey located in header

Parameters

id: string, required in path

Model id

Example request

curl

```
curl -X GET https://service.us.apiconnect.ibmcloud.com/gws/apigateway/api/d7f74e3029937583c7e82fce4bd3e308561b2bdbdd37c6832126bb989b179a/kxmBbT/Product.catalog/{id}
```

Example response

```
{"id": "8586451785839552", "name": "Baby Carrot", "description": "Eet ka oka furt eh baasnuuf dup klapje be come! also", "weight": "40.00", "cost": "40.00", "location": "Burgard", "available_qty": 10, "available_cost": "40.00", "available_weight": "400.00", "promo_cost": "10.00", "promo_end_date": "2018-12-31T23:59:59Z", "promo_start_date": "2018-12-01T00:00:00Z"}

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


```

Operations by name

Product.create

Product.findbyId

Product.prototype.patchAttributes

Product.delete_product

Product_catalog.create

Product_catalog.findbyId

Product_catalog.prototype.patchAttributes

Type Headers

Accept

application/json

Parameters

id: 100-100-01

Call operation

easily create test data and call the operation

Response

Code: 400 Bad Request

Headers

```
Date: Sun, 30 Dec 2018 19:59:52 GMT
X-Powered-By: Express
X-RateLimit-Remaining: 119
X-Global-Transaction-ID: 2011034221
Content-Length: 383
X-Request-ID: b2faefc10d6fpmRpwwvPJuVGAAmtMsb
X-Backside-Transport: OK OK
Server: nginx/1.15.0
ETag: W/"17-1X.nq1UAt0zY44ByjD5GmFdu/Y"
Vary: Origin
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS
Content-Type: application/json; charset=utf-8
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
X-RateLimit-Reset: 1546200628
X-RateLimit-Limit: 120
```

{ "available": "1", "available_qty": 0, "cost": "0.99", "description": "Great for pushing or shoveling snow. Poly blade won't", "id": "100-100-01", "location": "", "name": "Snow Shovel, Basic 22 inch", "promo_cost": "7.25", "promo_end_date": "2004-12-19T00:00:00.050Z", "promo_start_date": "2004-11-19T00:00:00.050Z", "weight": "1 kg" }

Response

Code: 200 OK

Headers

```
Date: Sun, 30 Dec 2018 19:59:58 GMT
X-Powered-By: Express
X-RateLimit-Remaining: 119
X-Global-Transaction-ID: 3453024073
Content-Length: 59
X-Request-ID: 29-74621178
X-Backside-Transport: OK OK
Server: nginx/1.15.0
ETag: W/"17-1X.nq1UAt0zY44ByjD5GmFdu/Y"
Vary: Origin
Access-Control-Allow-Methods: GET, POST, PUT, DELETE, PATCH, HEAD, OPTIONS
Content-Type: application/json; charset=utf-8
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
X-RateLimit-Reset: 1546208456
X-RateLimit-Limit: 120
```

{ "error": { "statusCode": 400, "message": "No documents found" } }

Export to full API Connect



API Info

API definition
You can export the current API definition to file. It is also possible to open the API definition in API Connect, IBM's premier API management platform. If API Connect is not present within your IBM Cloud space, we will first provision the service with the free Lite plan.

Security and Rate Limiting ⓘ

Application authentication
You can require consuming applications to authenticate using API key and secret or API key alone.

Method: API key only

API definition ▾
Export YAML file
Export JSON file
Open in API Connect

Drafts

Products APIs

Add Search APIs

Title	Last Modified	Type
Blockchain_Closing_API-XfTLdIq 0.0.1	4 months ago	REST
Catalog-CplmKy 0.0.1	4 months ago	REST
Initialise Marketing Campaign-BPsbGW 0.0.1	8 months ago	REST
Product and Product Catalog API-kxmBbT 0.0.1	a minute ago	REST

Open 'Product and Product Catalog API-kxmBbT' in API Connect

API Connect is IBM's premier platform for managing, socializing, analyzing, and monetizing APIs.

After your API is transferred, it will be available in the API Connect catalog.

If API Connect is not present within your IBM Cloud space, we will first provision the service with the free Lite plan. A new browser window will open to the API Connect interface.

Launch API Connect in a new browser tab

Cancel Continue

Product and Product Catalog API-kxmBbT

All APIs Design Source Assemble

Lifecycle

Phase: Realized

Policy Assembly
Security Definitions
Security
Extensions
Properties
Paths
/Product
/Product/[id]
/Product/delete_product
/Product_catalog
/Product_catalog/[id]

Analytics
Parameters
Definitions
Product
x-any
Product_catalog

Enforced
Enforce API using a gateway

Testable
Allow the API to be tested using the developer portal test tool

CORS
Enable CORS access control

Authenticate application
Enable application authentication with certificate

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

This is a list of available APIs for use in your flows. These APIs might have been added by importing an OpenAPI document or might be shared APIs from within the IBM Cloud organization you are in. More information on this [here](#).

Search APIs

Customer API for the FUDOrg-2FDNCF Not connected

My Shared API-LgYoAI Not connected

Payments Not connected

Do you have an OpenAPI definition file? Add your API now

Import

Add an API or Web service

petstore.yaml

or enter a file URL:

Name:

Description:

Do you have an OpenAPI definition or WSDL file?

Cancel Add

19

Halifax Open Data Account 1 Imported

Connect to Halifax Open Data

Username: (optional)

The basic auth username that you use to access Halifax Open Data

Password: (optional)

basic auth

API Key: (optional)

The API key that you use to access Halifax Open Data

Override the host name and port of the Halifax Open Data (optional)

override host URL

Network name: <Not selected (optional)>

Specifies the name of the network that App Connect will use to access your system. Required only if connecting to a system in a private network.

Connect

secure gateway for private network

More info about Halifax Open Data

Account: Account 1 Imported

Halifax Open Data actions

The action is the task you want your flow to complete.

- > /atms
- > /branches
- > /business-current-accounts
- > /commercial-credit-cards
- > /personal-current-accounts
- > /unsecured-time-loans

discovers objects

Request Response

What do you want to add?

Applications APIs Tools

Search APIs

Halifax Open Data Account 1

Account: Account 1

- > /atms GET /atms HEAD /atms
- > /branches GET /branches HEAD /branches

discovers operations

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

*OrderNumber OrderNumber

OrderDescription abc

Available inputs

Halifax Open Data / GET atms

- get_atm_model { }
- response { }
- 200: Successful response with a list of 'ATM' data { }
- Strict-Transport-Security abc
- Etag abc
- Cache-Control abc

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

The screenshot shows a MuleSoft Anypoint Studio interface with a flow editor and a configuration panel.

Flow Editor: At the top, a flow diagram shows a "Salesforce Account 1 New contact" step connected to an "HTTP Account 1 Invoke method" step.

Configuration Panel: Below the flow editor, the "HTTP Account 1 Invoke method" configuration screen is displayed.

- URL:** "https://api.eu.apiconnect.ibmcloud.com/contact-mailer/sb/postmarketingpack/sendDetails". A callout notes: "URL added with user credentials in Accounts will override this field".
- Request Headers:** "Accept: application/json", "Content-type: application/json". A callout notes: "add more request header fields" with a link to "Edit properties".
- Request Body:** A JSON object: {"name": "Full Name", "address": "Mailing Street", "Mailing City": "Other Zip/Postal Code"}. A callout notes: "build your request body if you need to build structured body use set variable".
- Action Buttons:** "Continue flow (non-2xx)" (highlighted with a blue arrow).
- Available Inputs:** A list of inputs from the previous "HTTP / Invoke method" step:
 - request { } (highlighted with a blue arrow)
 - HTTP method abc
 - URL (fully qualified) abc
 - Request headers { }
 - Request body abc
 - Response headers { }A callout notes: "HTTP action node outputs which can be mapped in other nodes down-stream".

API Integration

Unlock your apps and data...

VIDEO



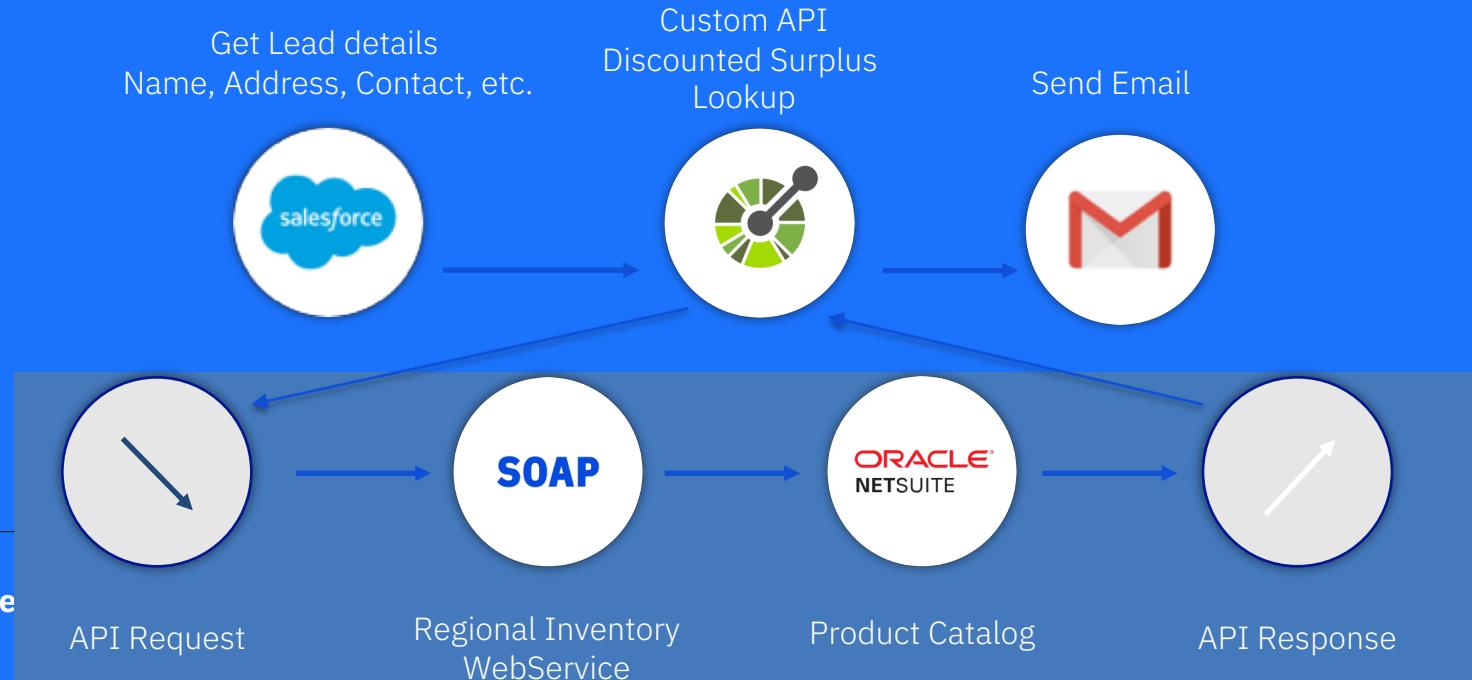
Cassie
Acme - Lead Digital
Marketing Coordinator

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.

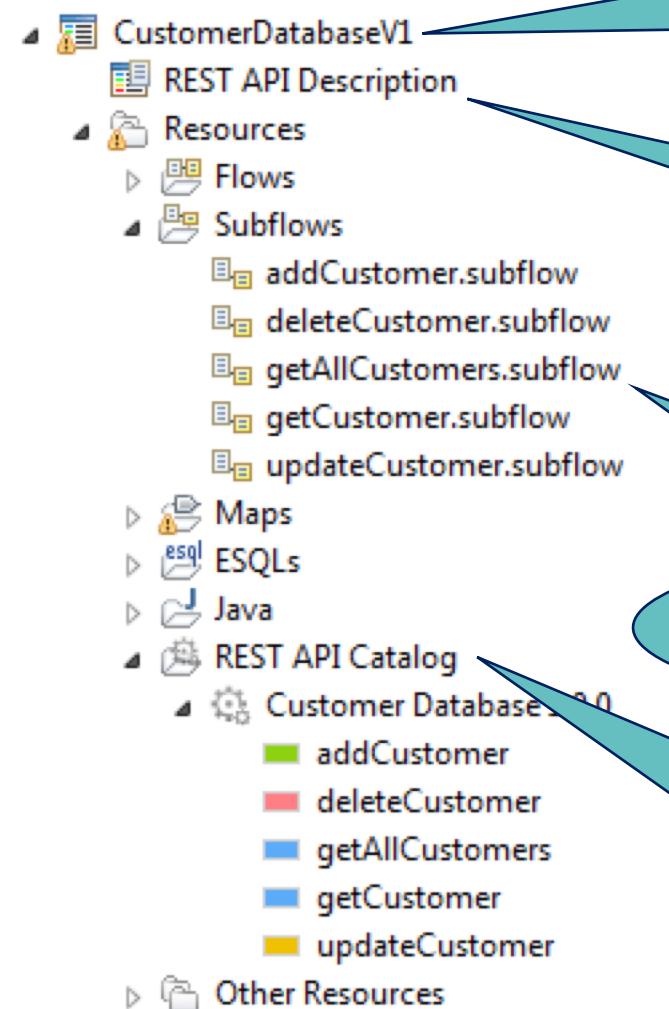


Tia
Acme – Integration
Developer



REST APIs in App Connect Enterprise

REST APIs in App Connect Enterprise



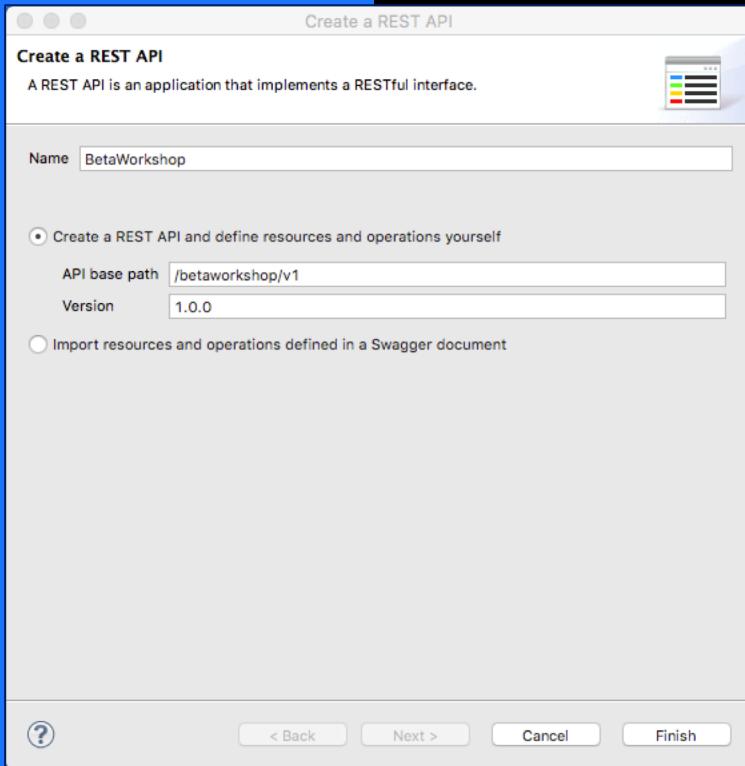
Integration Bus V10 introduced a new type of project, or container, called a REST API.

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

IBM

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

The image displays three sequential screens of a software wizard for creating a REST API:

- Step 1: Create a REST API**

A REST API is an application that implements a RESTful interface.

Name: BetaWorkshop2

Create a REST API and define resources and operations yourself

API base path: /betaworkshop/v1
Version: 1.0.0

Import resources and operations defined in a Swagger document

Buttons: ? < Back Next > Cancel Finish
- Step 2: Create REST API from definition file**

Create a REST API from an existing Swagger 2.0 document.

Select from a file system

Location: [Browse]

Select from your workspace

BetaWorkshop
 Resources
 swagger.json
CustomerDatabaseREST
 Customers
 Employees
 Orders
 PushToAPIConnectV1
 PushToAPIConnectV2
 queryParams

Open Swagger Editor in the default web browser, outside of Integration Development Toolkit

Buttons: ? < Back Next > Cancel Finish
- Step 3: Review the imported REST API definition**

Title: Customer Database
Description: This is the customer database sample Swagger document included with IBM
Base path: /customerdb/v1
Version: 1.0.0

Operation	Method	Resource
getAllCustomers	GET	/customers
addCustomer	POST	/customers
getCustomer	GET	/customers/{customerId}
deleteCustomer	DELETE	/customers/{customerId}
updateCustomer	PUT	/customers/{customerId}

Buttons: ? < Back Next > Cancel Finish

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.

The screenshot shows the IBM Integration Toolkit REST API Editor interface. The title bar reads "Integration Development - BetaWorkshop/restapi.descriptor - IBM Integration Toolkit - /Users/sstone1/IBM/IIBT10/workspace". The main window has several sections:

- Header:** REST API base URL: /betaworkshop/v1, Title: BetaWorkshop, Description: BetaWorkshop, Version: 1.0.0. A note says: "You can access the operations in the REST API by pointing your web browser to the following URL, where <hostname> is the host name and <port_number> is the port number: http://<hostname>:<port_number>/betaworkshop/v1".
- Resources:** A table with columns: Name, Array, Type, Allow null, Format, Required. A placeholder row says: "+ Enter a unique name to create a new model".
- Model Definitions:** A table with columns: Name, Array, Type, Allow null, Format, Required. A placeholder row says: "+ Enter a unique name to create a new model".
- Error Handling:** A section with a plus sign icon.
- Security:** A section with a plus sign icon.

Defining resources



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

The screenshot illustrates the process of defining a new resource in a REST API using the IBM Integration Toolkit.

Create Resource Dialog: On the left, a modal dialog titled "Create Resource" is open. It prompts the user to "Select a path segment in the existing resource structure to create a resource under it". A "Base path" tree view is shown. Below, the "Resource path relative to the selection" is set to "/users". Under "Select operations to add to the resource", the "GET" and "POST" checkboxes are checked, while others like "OPTIONS", "HEAD", "PUT", "PATCH", and "DELETE" are unchecked. At the bottom are "OK", "Cancel", and "Apply" buttons.

Integration Development Environment: On the right, the main "Integration Development" window shows the "BetaWorkshop" project. The "Header" section contains the REST API base URL: "/betaworkshop/v1", title: "BetaWorkshop", description: "BetaWorkshop", and version: "1.0.0". The "Resources" section lists the newly created resource at the path "/users". It includes two operations: "getUsers" (GET) and "postUsers" (POST). The "getUsers" operation retrieves users, has a response status of 200 ("The operation was successful."), and a schema type of "string". The "postUsers" operation inserts a user, has a request body schema type of "string", and a response status of 200 ("The operation was successful."). Both operations are marked as required. The "Model Definitions" section is also visible at the bottom.

A large green arrow points from the "Create Resource" dialog to the "Integration Development" window, indicating the flow of the process.

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

The screenshot shows the IBM API Connect interface for defining API resources. The left sidebar shows a tree structure with 'Resources' expanded, and a node for '/users/{user}' is selected, indicated by a red box.

/users/{user}

GET `getUser` Retrieve user

Name	Parameter type	Data type	Format	Required	Description
user	path	string		<input checked="" type="checkbox"/>	

Response status **Description** **Array** **Schema type** **Allow null**

200	The operation was successful.		string	<input type="checkbox"/>
-----	-------------------------------	--	--------	--------------------------

POST `postUser` Insert a user

Name	Parameter type	Data type	Format	Required	Description
user	path	string		<input checked="" type="checkbox"/>	

Request body **Schema type** **Allow null**

The request body for the operation	string	<input type="checkbox"/>
------------------------------------	--------	--------------------------

Response status **Description** **Array** **Schema type** **Allow null**

200	The operation was successful.		string	<input type="checkbox"/>
-----	-------------------------------	--	--------	--------------------------

Integration

Defining parameters

IBM

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

The screenshot shows two views of the IBM API Management interface, illustrating the process of defining parameters for a RESTful API operation.

Top View: This view shows the initial state of the operation configuration. The operation is named "getUsers" and is described as "Retrieve users". The table below lists the parameters:

Name	Parameter type	Data type	Format	Required	Description

A green arrow points from the "Description" column towards the "+" button in the header, indicating where new parameters can be added.

Bottom View: This view shows the state after adding parameters. The "maxUsers" parameter has been added and is highlighted with a red box. The "searchString" parameter is also listed. The "Description" column for "maxUsers" contains the text: "The maximum number of users to retrieve".

Name	Parameter type	Data type	Format	Required	Description
maxUsers	query	integer		<input checked="" type="checkbox"/>	The maximum number of users to retrieve
searchString	query	string		<input checked="" type="checkbox"/>	A filter string for users

In In the bottom left corner, there is a small "In" icon.

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

Model Definitions

Name	Array	Type	Allow null	Format	Required
(+) <Enter a unique name to create a new model>	<input type="checkbox"/>	object			
{...} User	<input type="checkbox"/>				

Model Definitions

Name	Array	Type	Allow null	Format	Required
(+) <Enter a unique name to create a new model>	<input type="checkbox"/>				
{...} User	<input type="checkbox"/>	object			
id	<input type="checkbox"/>	string			
firstname	<input type="checkbox"/>	string			
lastname	<input type="checkbox"/>	string			
[] tags	<input checked="" type="checkbox"/>	string	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Referencing models

IBM

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

The screenshot shows the 'postUser' operation configuration. The 'Request body' section has a dropdown menu for 'Schema type' containing a 'User' model, which is highlighted with a green arrow pointing to it.

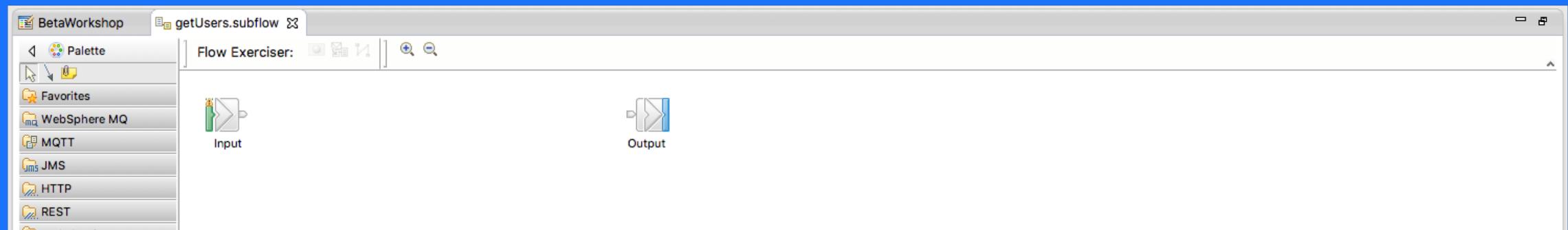
The screenshot shows the 'getUsers' operation configuration. The 'Response status' section has a dropdown menu for 'Schema type' containing a 'User' model, which is highlighted with a green arrow pointing to it.

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.

The screenshot shows the 'Resources' view in IBM Integration Workbench. A REST API endpoint for 'getUsers' is defined under the path '/users'. The 'getUsers' operation is mapped to a subflow named 'Retrieve users'. The response status is set to 200, indicating success, with the description 'The operation was successful.'

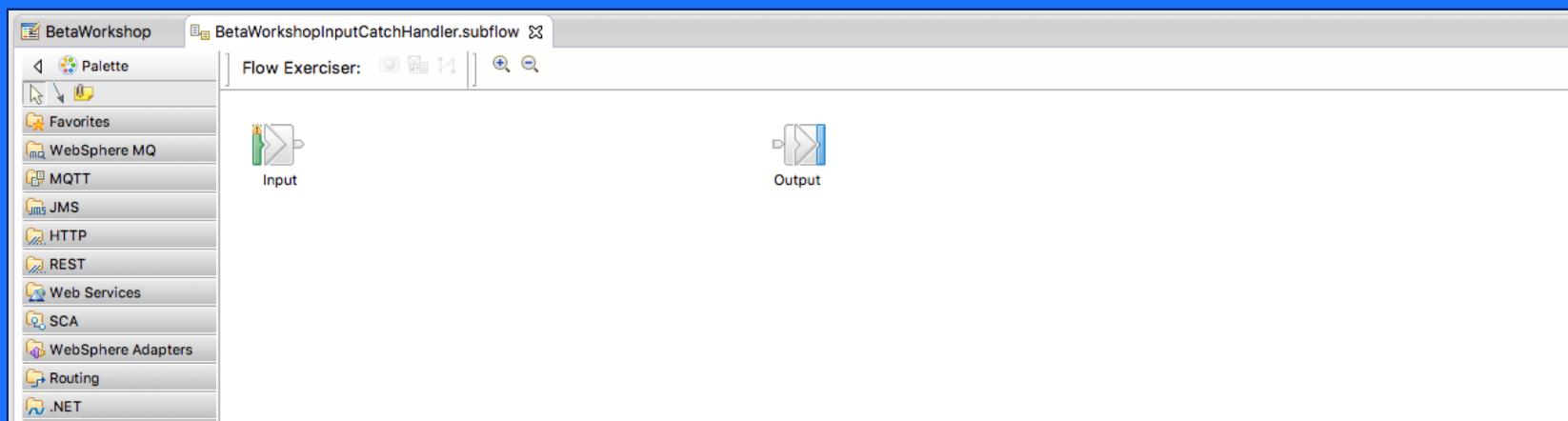


Implementing error handlers and HTTPS



- 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

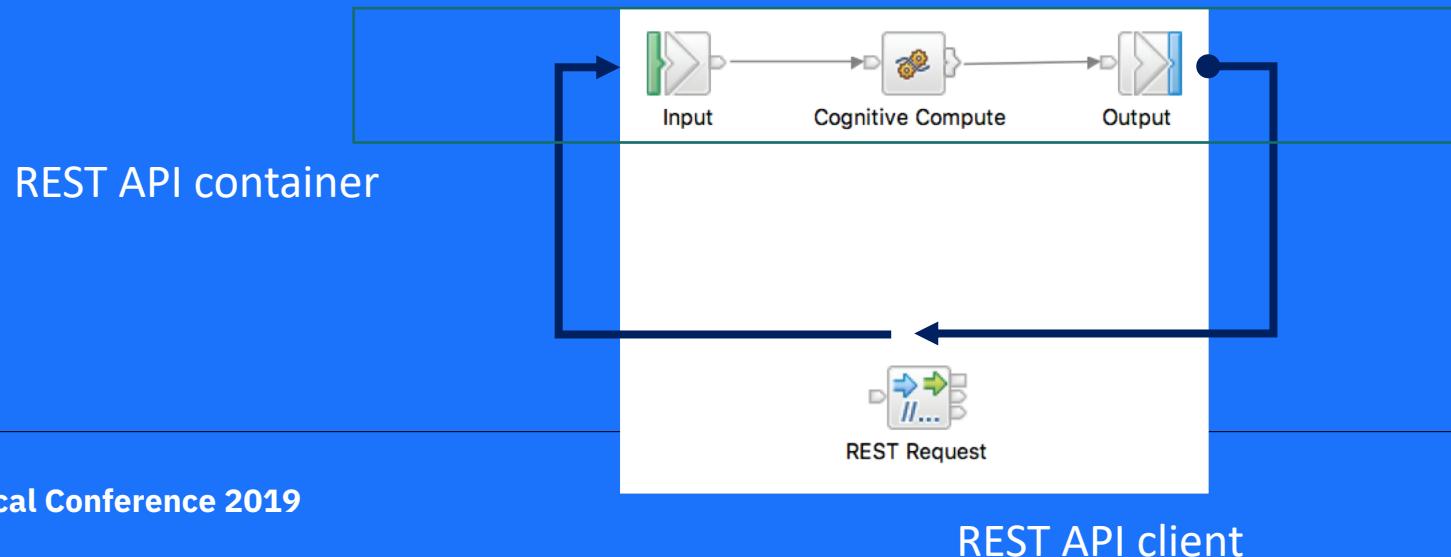
The screenshot shows the 'BetaWorkshopInputCatchHandler.subflow' in the REST API Editor. The 'Error Handling' section contains three entries: 'Implement the Catch handler', 'Implement the Failure handler', and 'Implement the Timeout handler'. The 'Security' section contains a checkbox labeled 'Enable HTTPS' with a green arrow pointing to it, and a note stating 'You can enable other security settings in the BAR file editor.'



Implementing an operation



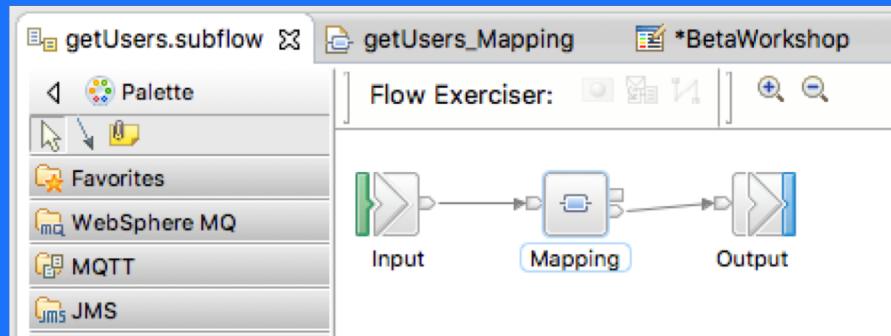
- When the operation is called by an HTTP client, a message will be routed automatically by the REST API container to the input node for the corresponding subflow for that operation.
- That message will have a JSON request body, if a body has been provided in the request. JSON is the default message 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.



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!

A screenshot of the 'getUsers_Mapping' configuration table. The table lists various parameters and their types:

Parameters	[0..1]	<Anonymous>
choice of cast items	[0..*]	
any	[1..1]	
maxUsers	[1..1]	string
param2	[1..1]	string

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:

The screenshot shows two main windows from the IBM Integration Designer interface.

Top Window (Flow Exerciser): Shows a simple flow with an "Input" port connected to a "Mapping" node, which then connects to an "Output" port.

Bottom Window (New Message Map): A dialog box titled "Specify a new message map file". It includes fields for "Container" (PetStore) and "Map name" (addPet_Mapping). The "Type of map that you want to create:" section has a radio button selected for "Message map with the input and output for REST API operation addPet", which is highlighted with a red box.

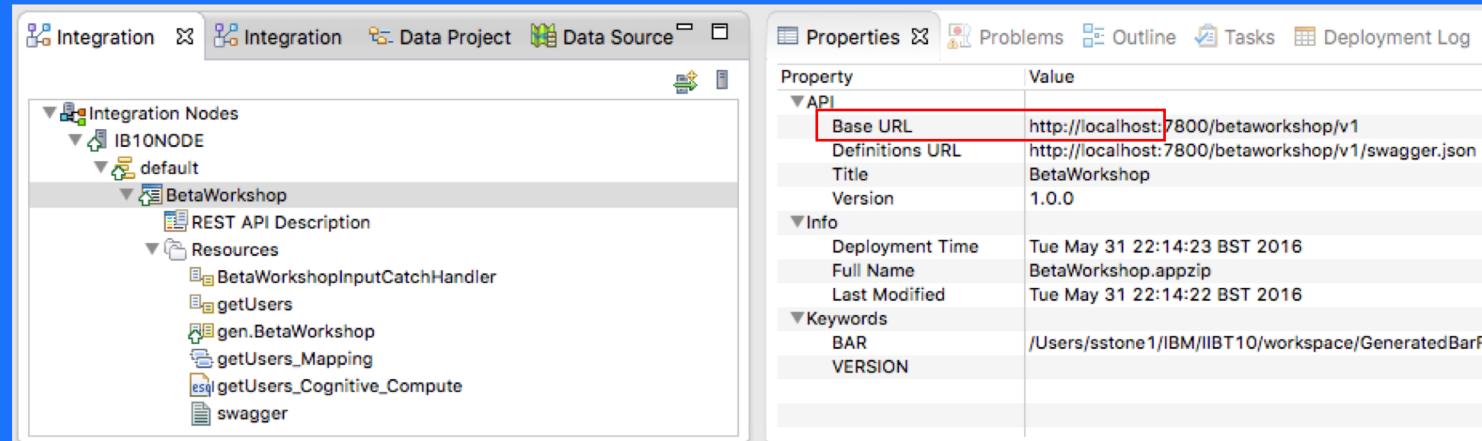
Central View (addPet_Mapping): The main workspace shows a "Transform - Task" configuration. On the left, a "Properties" tree lists various JSON schema elements: "Message Assembly" (JSON), "Properties" (PropertiesType), "LocalEnvironment" (_LocalEnvironmentType), "JSON" (JSONMsgType), "Padding" (string), "Data" (Pet), "id" (<integer>), "category" (Category), "name" (<string>), "photoUrls" (JSONArray_photoUrls), "tags" (JSONArray_tags), and "status" (<string>). On the right, there are nodes for "Task", "Move", "Insert into SAMPLE/BOA", "Insert", and "Return". A dashed green line connects the "Data" node in the properties tree to the "Insert" node.

Bottom Panel (Transform - Task): A "General" tab shows the description: "The REST API input data of operation addPet.". A "Documentation" tab provides additional information: "This Task is provided to show the location of the REST API input data of operation addPet. The path and query parameters are provided under 'Parameters'. You may need to use these as inputs to your mappings."

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.

The screenshot shows the IBM App Connect web interface. The top navigation bar includes the IBM logo, the title "IBM App Connect", and a "Started" status indicator. Below the navigation, the page title is "HR_Service_ODBC". A horizontal menu bar offers links to "Documentation", "Contents", "Properties", "Statistics", and "Other resources". The main content area displays the REST API Base URL (http://myhost.adomain.com:7600/1/HR_Services/resources) and the OpenAPI document (http://myhost.adomain.com:7600/1/HR_Services/resources/swagger.json). A search bar is present at the top of the content area. Two sections of API endpoints are listed:

- /departments**
 - GET** getDepartments
 - POST** createDepartment
- /departments/{departmentKey}**
 - GET** getDepartment
 - PUT** updateDepartment
 - DELETE** deleteDepartment

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)

The screenshot shows the Swagger UI interface for a 'Customer Database' API. The main title is 'Customer Database'. Below it, a note says 'This is the customer database sample Swagger document included with IBM Integration Bus'. The 'customers' endpoint is highlighted, showing its operations:

- GET /customers**: Description: 'Get all customers from the database'.
- POST /customers**: Description: 'Add a customer to the database'. It includes a 'Parameters' table:

Parameter	Value	Description	Parameter Type	Data Type
body	(required)	The customer to add to the database	body	Model Example Value

The 'Example Value' field contains a JSON object:

```
{  
  "id": 0,  
  "firstname": "string",  
  "lastname": "string",  
  "address": "string"  
}
```

Below the table, there's a note: 'Parameter content type: application/json'. The 'Response Messages' section shows a 200 status code with the reason: 'If the customer was successfully added to the database'. At the bottom, there are 'Try it out!' and 'Headers' buttons.

At the very bottom, there are other endpoint cards for DELETE, GET, and PUT operations on the '/customers/{customerId}' path.

[BASE URL: /customerdb/v1 , API VERSION: 1.0.0]

Developing REST APIs in App Connect Enterprise



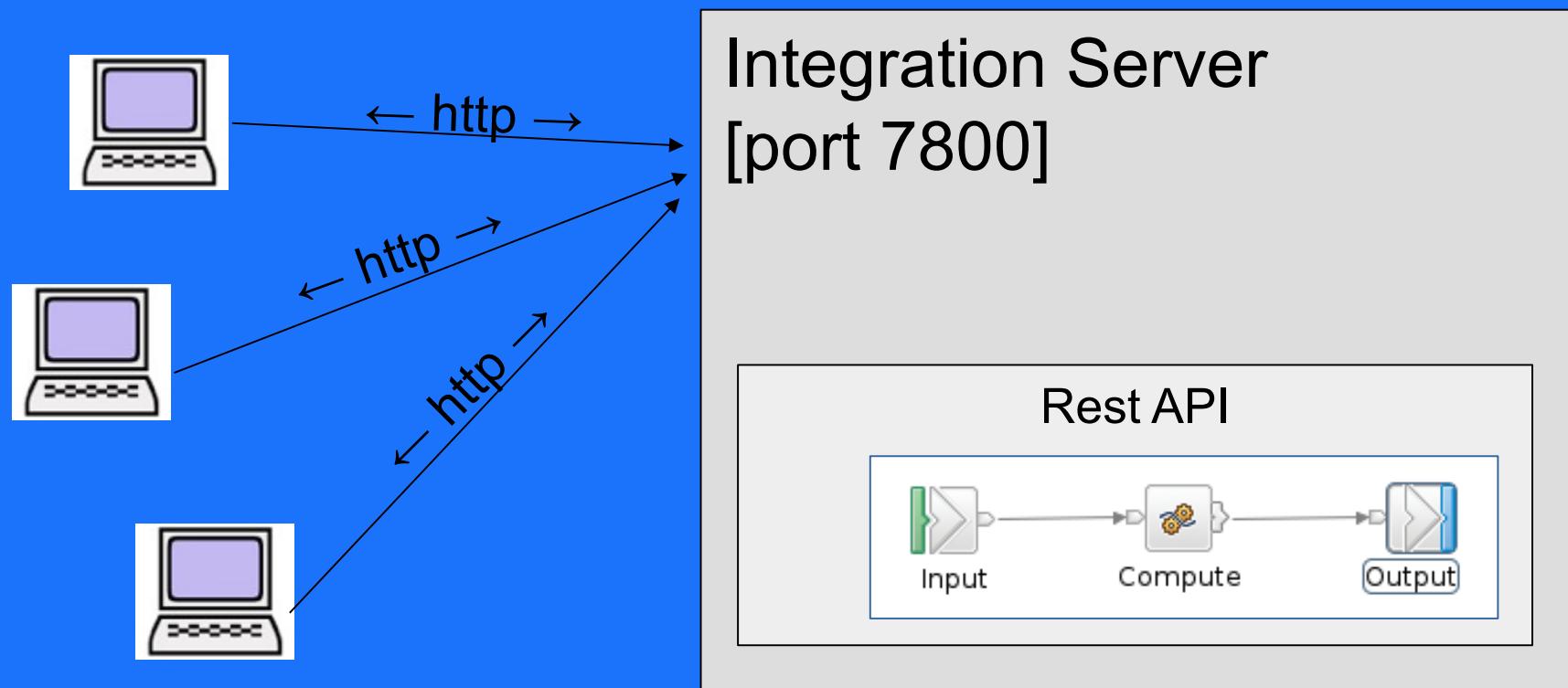
LIVE DEMO

Pushing a REST API to API Connect

REST APIs developed in IIB



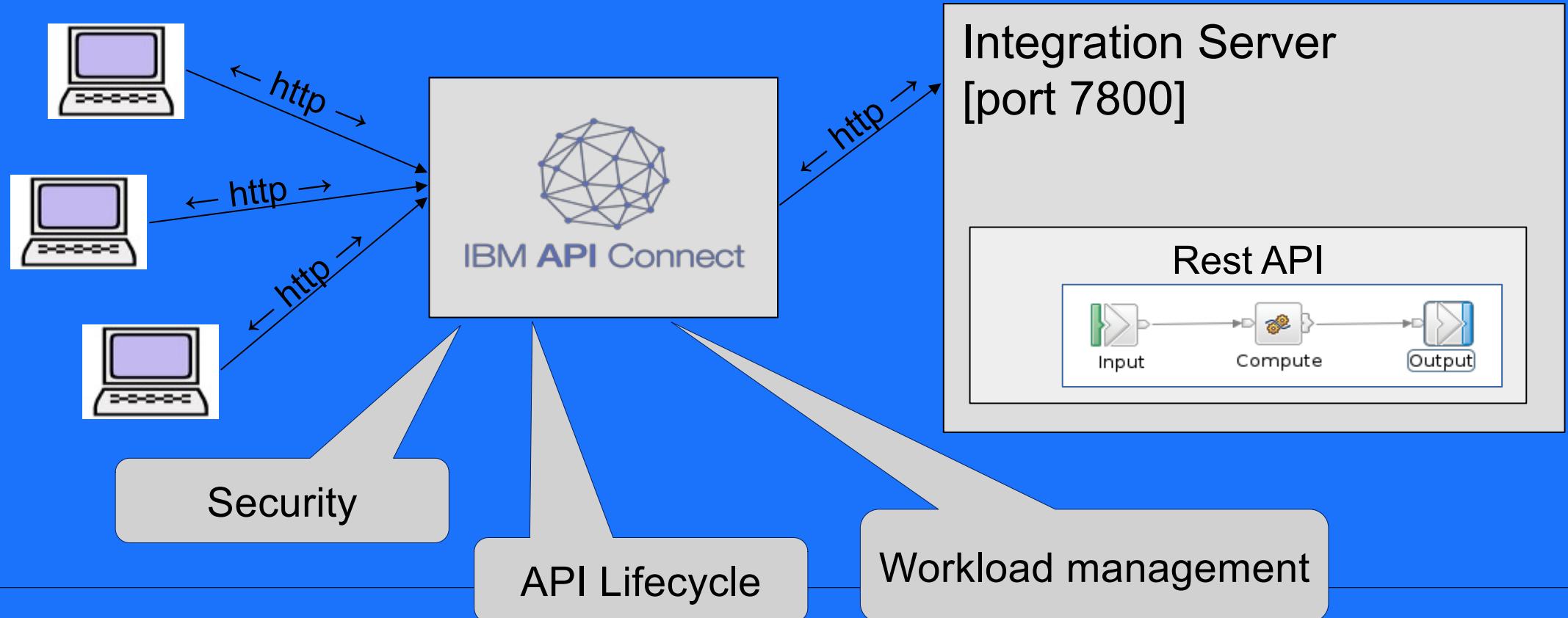
- We have just seen how REST APIs allow integrations to be exposed as a RESTful web services that can be called by HTTP clients.



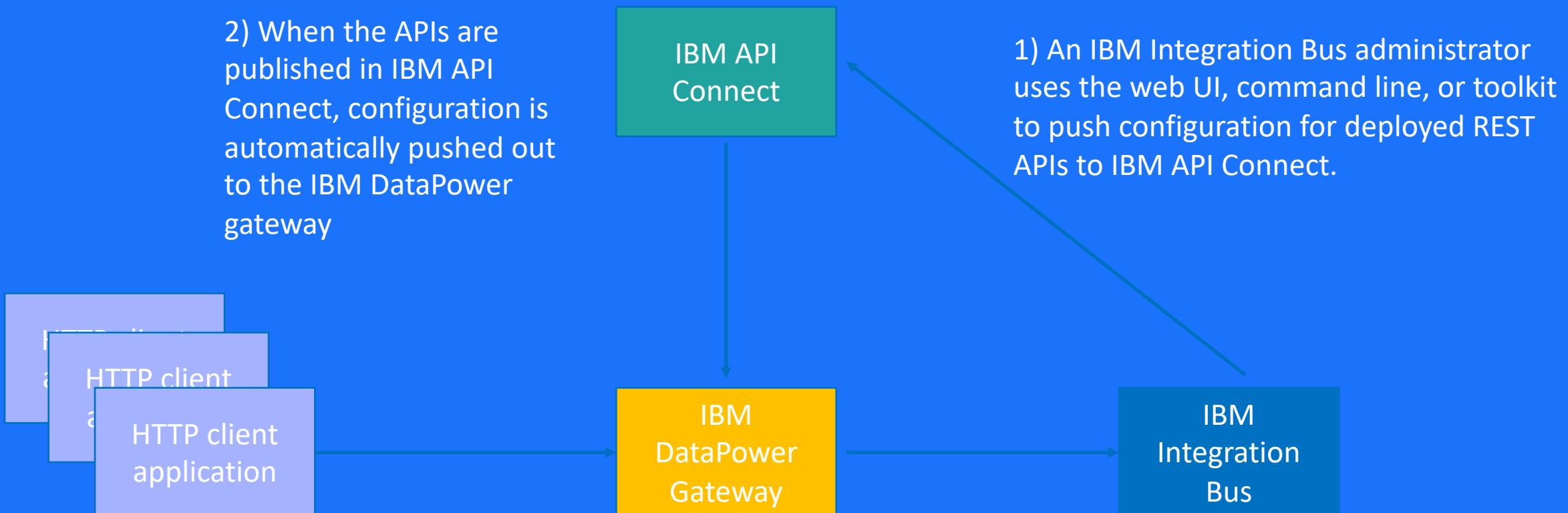
IBM API Connect



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



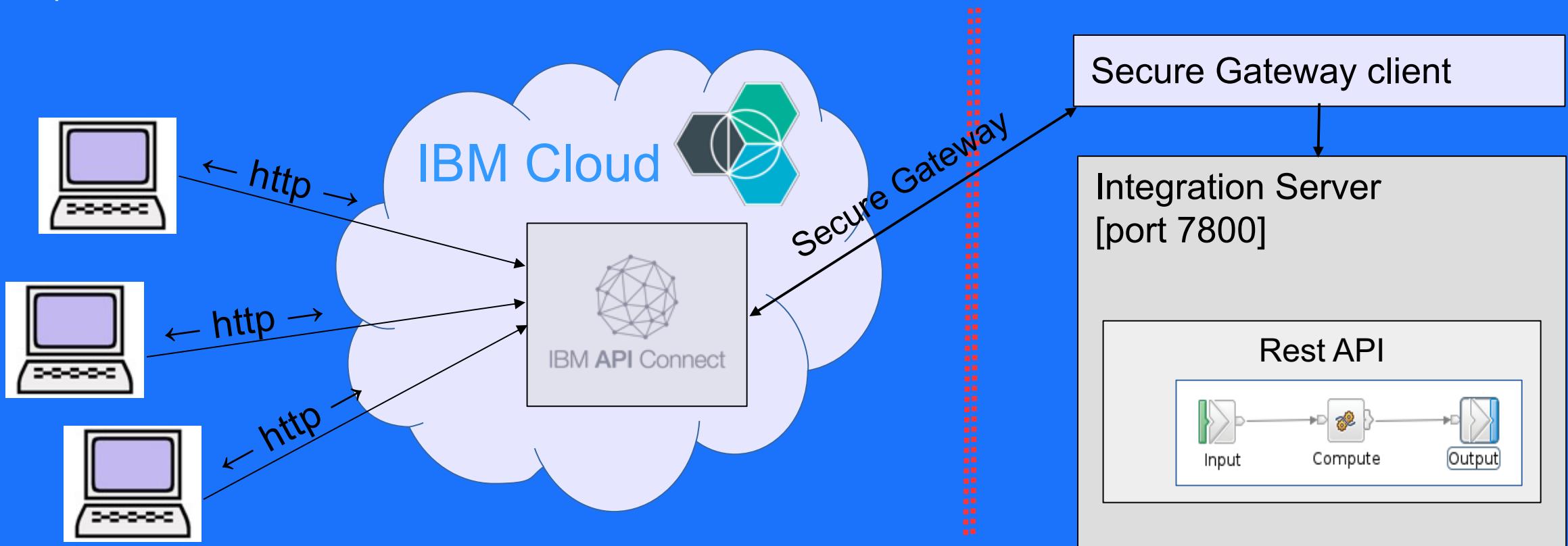
Interaction between IIB and IBM API Connect



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



1.

Create draft product

2.

Import API definition
(swagger)

3.

Add API to product

4.

Create Plan(s) for APIs

5.

Stage product in Catalog

6.

Publish Product

• Toolkit – Push API

• **Webui** - Push to API Connect
• **mqcipushapis** – command line

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



The screenshot shows the IBM Integration Bus (IIB) Web UI interface. On the left, the navigation tree is expanded to show the TESTNODE Sanjay node, specifically the default server under REST APIs. The main panel displays the Overview of the TESTNODE node. A modal dialog box is open in the center, titled "Push REST APIs to IBM API Connect". The dialog has two main sections: "Management Cluster / Server Address" and "Authentication". In the "Management Cluster / Server Address" section, the "Host" field is empty and highlighted with a blue border, while the "Port" field contains the value "443". In the "Authentication" section, both the "UserID" and "Password" fields are empty. At the bottom of the dialog is a large "Connect to IBM API Connect" button.

IBM Integration

TESTNODE

Overview

Filter Options...

TESTNODE_sanjay

Servers

default

Services

REST APIs

CustomerDatabase

Applications

Libraries

Shared Libraries

Message Flows

Subflows

Resources

Operational Policy

Data

Security

Monitoring

Business

Quick View

Node N

Version

Admin

Run Mo

Short D

Long D

Advanced P

Platform

Fixpac

Welcome

Push REST APIs to IBM API Connect

Define a connection to the IBM API Connect system

Management Cluster / Server Address

Host

Port 443

Authentication

UserID

Password

Connect to IBM API Connect

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!



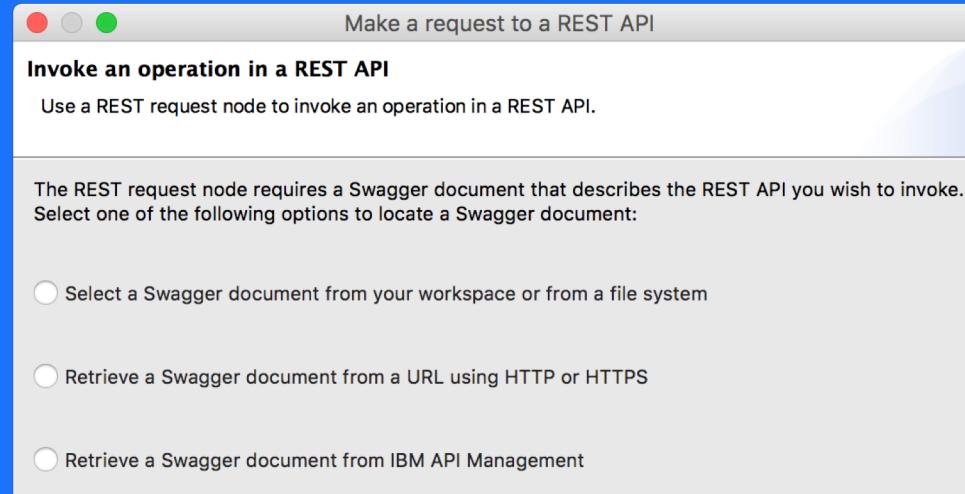
The screenshot shows the 'REST Request Node Properties - REST Request' dialog box. The 'Basic' tab is selected, displaying the 'REST API definitions file*' field set to 'swagger.json' with a 'Browse...' button. The 'Operation*' dropdown shows 'deleteCustomer - Delete a specified customer from the database'. The 'Parameters' table lists two entries:

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

REST Request node and Swagger



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

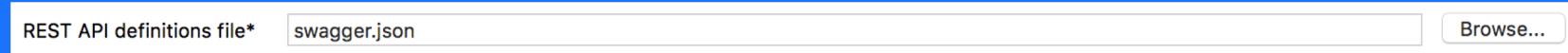


- Swagger documents supported in both available formats (JSON and YAML)

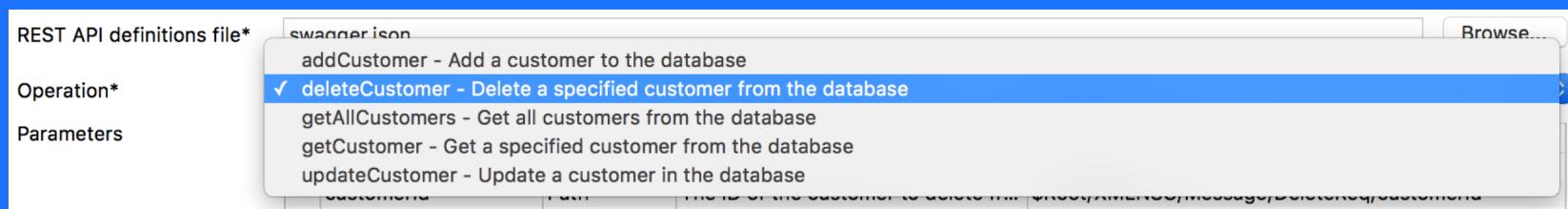
REST Request node and Swagger



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



- The REST Request node uses 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:

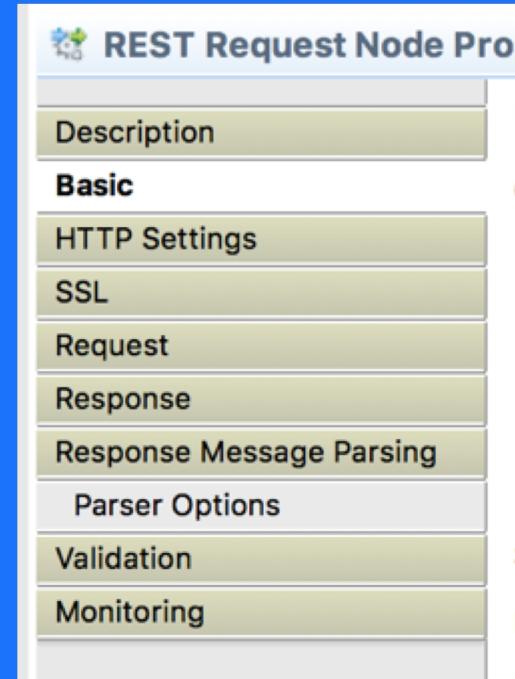
```
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.Authorization = 'suchASecretAuthKey';
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.customerId = 12345;
SET OutputLocalEnvironment.Destination.REST.Request.Parameters.clientName = 'LE override client';
```

- 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 myRETPassword -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



Pattern Specification Inst03 - Pattern Configuration

Review Specification

Information about the pattern instance.

App Connect REST Pattern

IBM App Connect is a Software as a Service (SaaS) solution that provides a simple web based experience for easily creating app-to-app connections in a few minutes. You can connect SaaS apps, so that when an event occurs in one app the other connected apps are updated automatically.

App Connect also enables users to expose a REST API flow, so that when the request is submitted (such as through a mobile or web application), the flow performs all its actions, and then returns a response. IBM App Connect is available as a [service as part of IBM Cloud](#).

App Connect RESTRequest node

IBM

App Connect REST Request Node Properties - App Connect REST Request

Description ✖ Definitions file: Value must be set.

Basic

Request

Monitoring

Operation
Specify a Swagger 2.0 file containing the definitions for the REST API that you wish to invoke, and an operation within that REST API.

Definitions file*

Operation*

Security identity <The name of an identity defined with mqsisetdbparms. Do not include the "rest://" prefix here>

Request timeout (sec)* 120

Base URL override <defa

App Connect REST Request Node Properties - App Connect REST Request

Description

Basic

Request

Monitoring

Parameters
Specify the values for the parameters to this operation by specifying XPath expressions for each parameter.
Alternatively, you can specify the value of the parameter in the LocalEnvironment message tree by creating an element named:
`LocalEnvironment.Destination.REST.Request.Parameters.<parameter name>`

Name	Type	Expression	Description

Content-Type* Determine Content-Type from input message

Input body location* \$Body

Integration Technical Conference 20

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

Pattern Specification Inst03 - Pattern Configuration

Review Specification

Information about the pattern instance.

Solution

This pattern helps accelerate users wanting to connect IBM Integration Bus to SaaS applications. Menus on the pattern let you select the SaaS application you'd like to connect with and the action you'd like to execute. The pattern generates a simple IIB message flow containing an App Connect REST Request node, and an App Connect Template. You import the Template into App Connect, and provide the required credentials to access the SaaS application. Once deployed, the IIB message flow invokes App Connect using REST (JSON over HTTPS with Basic Authentication).

The diagram illustrates the flow of data from different sources (HTTP, File, MQ) into the IBM Integration Bus. Inside the bus, the flow passes through several nodes before reaching a central connector. From this connector, the flow continues through an App Connect node to a SaaS app. The connection is secured using JSON / HTTPS with Basic Auth.

Related information

[Parameters for the pattern](#)
[Tasks to complete after generating the pattern](#)
[IBM Knowledge Center - AppConnectRESTRequest node](#)
[IBM Knowledge Center - Connecting to an App Connect REST API](#)
[IBM Knowledge Center - Configuring security credentials for connecting to an App Connect REST API](#)

Specification Configuration

App Connect REST pattern



Select SaaS Application

Select Action

Define style of IIB flow

Pattern Specification Inst03 - Pattern Configuration

Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click [here](#) to generate a pattern instance.

i Pattern workspace resources are generated successfully!

Pattern Parameters

Basic Pattern Parameters

App Connect *	Application: Slack
	Action: Create message
Create IIB Flow *	<input checked="" type="checkbox"/>
IIB Input Protocol *	HTTP
HTTP URL Path Suffix *	/Example
MQ Input Queue *	EXAMPLE.IN
MQ Output Queue *	EXAMPLE.OUT
File Input Directory *	C:\ExampleInput
File Output Directory *	C:\ExampleOutput
File Output Filename *	output.txt
IIB Transform Type *	Mapping node

Generate

Specification Configuration

App Connect REST pattern



Optionally generate an IIB flow

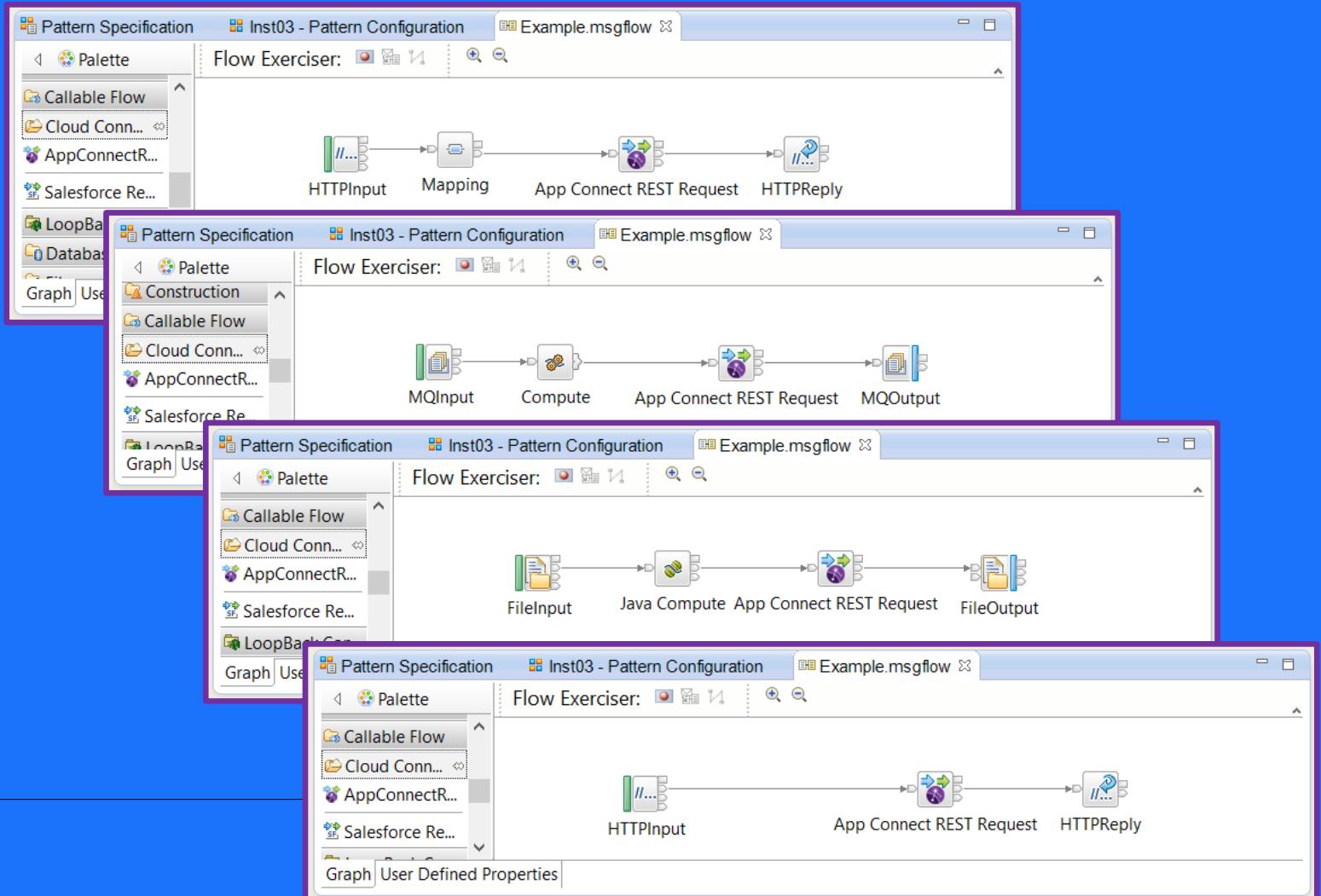
Choose IIB Input Protocol

HTTP / MQ / File

Choose IIB Transform Type

Map / ESQL / Java / None

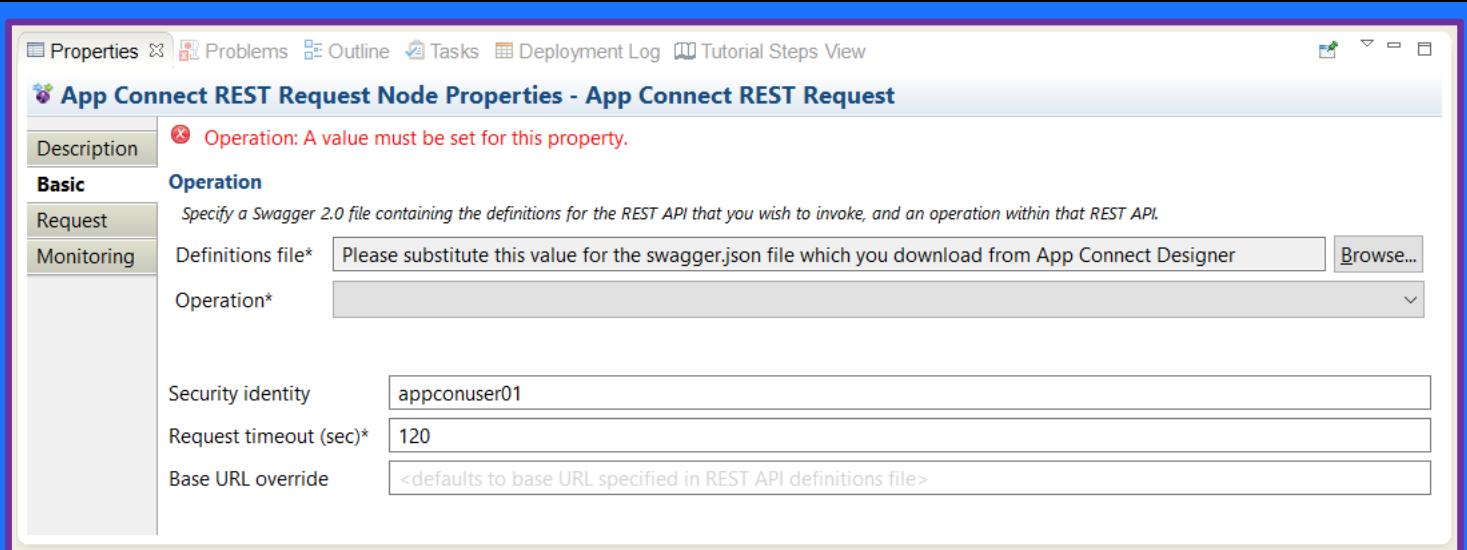
Transform node provides simple assignment of values



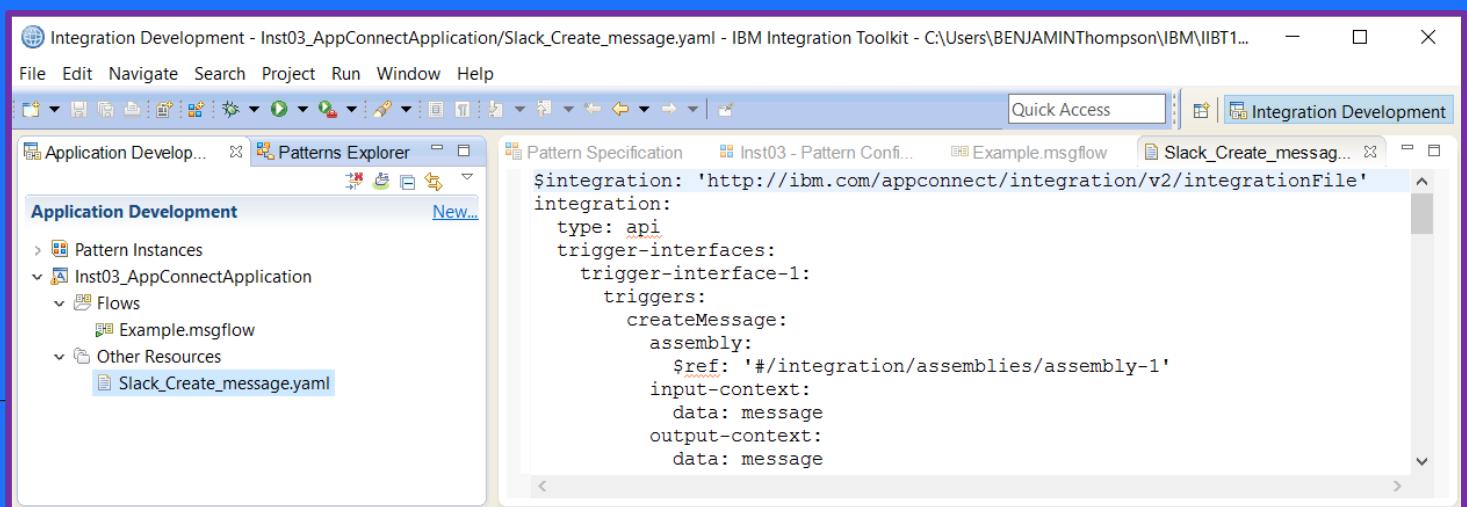
App Connect REST pattern



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

IBM

Import the generated App Connect template into the Designer dashboard

The screenshot shows two instances of the IBM App Connect Designer interface. The top instance displays the main dashboard with navigation tabs for Dashboard, Applications, Templates, and Notifications. A search bar and a 'New' button are visible. The bottom instance shows a modal dialog titled 'Import flow'. The dialog instructs users to 'Import the flow from a .yaml file or via a URL.' It features a file input field containing 'Slack_Create_message.yaml' and a 'Flow URL:' input field. At the bottom right of the dialog are 'Cancel' and 'Import' buttons. To the right of the dialog, a sidebar menu is open, listing options: 'Event-driven flow', 'Flows for an API', 'Import flow...', and 'Import a BAR file...'. The sidebar also shows the user's profile as 'Ben Thompson'.

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

The screenshot shows the IBM App Connect Designer interface. The top navigation bar includes 'IBM App Connect' and a user profile for 'Ben Thompson'. Below the bar, there are tabs for 'Dashboard', 'Applications' (which is selected), 'Templates', and 'Notifications'. A search bar labeled 'Search application library' is positioned above a list of connected applications. The list includes:

Application	Status
Asana	Not connected
Atlassian JIRA Service Desk	Not connected
BigCommerce	Not connected
Box	Not connected
Coupa	Not connected
Dropbox	Not connected
Equals 3 Lucy	Not connected
Eventbrite	Not connected
Freshdesk	Not connected
Gmail	Account 1 (appcjet@gmail.com)

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.

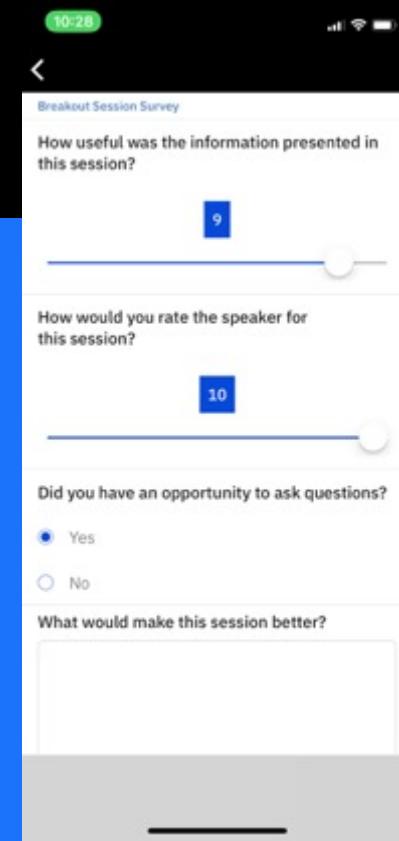
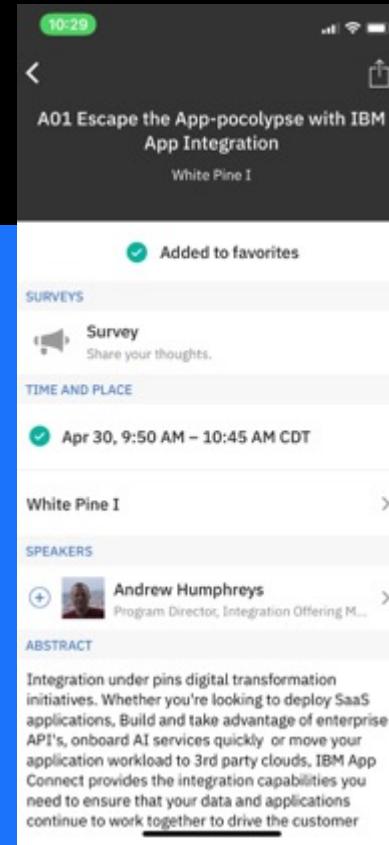
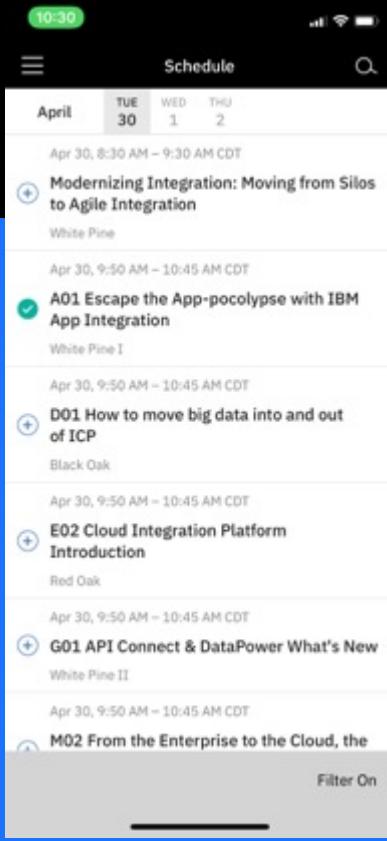
The screenshot illustrates the process of managing an App Connect REST API and configuring its request node.

Manage API Page: Shows the API base URL as <https://apis-3.appconnect.ibmcloud.com/COauCO>, Username as appconuser01, and Password as a masked string. A tooltip indicates the operation is Slack_Create_message-zYmH9X 0.0.1.

File Download Dialog: A Firefox dialog titled "Opening swagger.json" shows the file is a JSON file (1.0 KB) from a blob. It offers options to Open with TWINUI (default) or Save File, with the latter selected. There is also a checkbox for "Do this automatically for files like this from now on."

App Connect REST Request Node Properties: The "Request" tab is selected. The "Operation" section specifies a Swagger 2.0 file (Slack_Create_message-zYmH9X-0.0.1.yaml) and the operation message.create - Create a new instance of the model and persist it into the data source. The "POST /message" row is highlighted in green. Other fields include Security identity (appconuser01), Request timeout (sec) (120), and Base URL override (<defaults to base URL specified in REST API definitions file>).

Questions ?



IBM

Don't forget to fill out the survey!

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

Thank You

