HHI-2111
IBM Integration Bus and REST APIs

InterConnect
2017

Sanjay Nagchowdhury

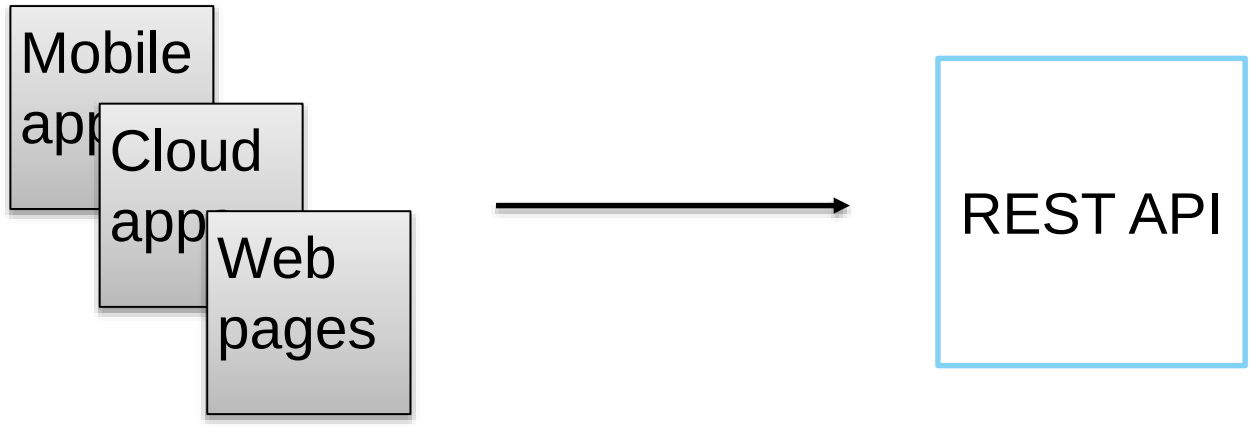
IBM Integration Bus
sanjay_nagchowdhury@uk.ibm.com

Agenda

- Introduction
- Developing a REST API in IIB
  - ➢ Demo
- Pushing a REST API to API Connect
  - ➢ Demo
- REST Request, REST Async Request, REST Async Response nodes
  - ➢ Demo
- Summary

REST APIs

- A REST API is a lightweight web service API based on HTTP, and is a much simpler alternative to SOAP based web services.
- A REST API describes a set of resources, and a set of operations that can be called on those resources.
- Those operations can be called from any HTTP client - there are HTTP clients available for most programming languages nowadays.
- Operations in a REST API can easily be called from JavaScript code running in a web browser, or application code running on a mobile device.



REST APIs - resources

- A REST API has a base path – the root from which all of the resources and operations are available. An example base path might be:
  - **http://mycompany.com:7843/customerdb/v1**
- Each resource in a REST API has a path, relative to the base path, which identifies that resource. Example resources might be:
  - **/customers** - all of the customers in the database
  - **/customers/12345** - customer #12345
  - **/customers/12345/orders** - all orders for customer #12345
  - **/customers/12345/orders/67890** - order #67890 for customer #12345

REST APIs - operations

- Each resource in a REST API has a set of operations.
- An operation in a REST API has a name, and an HTTP method, such as GET, POST, PUT, or DELETE.
- The combination of the path of the HTTP request and the HTTP method identifies which resource and operation is being called.
- Example operations on the resource /customers/12345 might be:
  - **GET getCustomer** - retrieve the customer details from the database.
  - **PUT updateCustomer** - update the customer details in the database.
  - **DELETE deleteCustomer** - delete the customer from the database.
- In order to call the updateCustomer operation, the HTTP client must make a HTTP PUT request to:
  - **http://mycompany.com:7843/customerdb/v1/customers/12345**

REST APIs - parameters

- Each operation defined in a REST API can also specify a set of parameters. Parameters can be used to pass information in to the operation.
- These parameters are in addition to the body passed in the HTTP request.
- Integration Bus supports three different types of parameters:
  1. **Path parameters** – one or more parts of the path for a resource can be defined as a variable. For example, the customer ID in the previous examples is a path parameter:
     - /customers/{customerId}/orders/{orderId}
     - /customers/**12345**/orders/**56789**
  2. **Query parameters** – one or more parameters can be specified in the URL following the path:
     - /customers?**maxResults=5**&**name=2**
  3. **Header parameters** – one or more parameters can be specified in the headers of the HTTP request:
     - **Api-Client-Id: ff6e2c5d-42d5-4026-8f7f-d1e56da7f777**

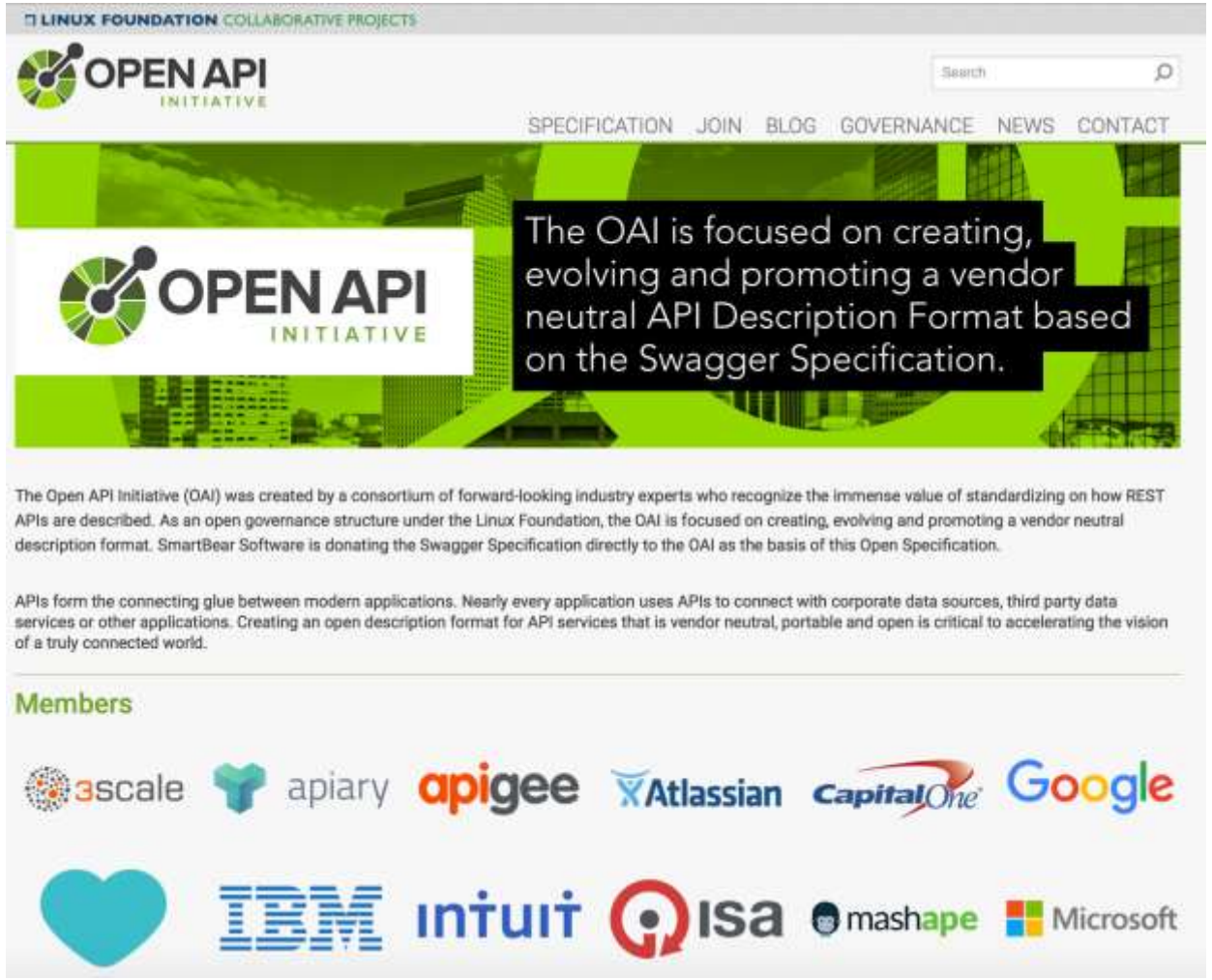
REST APIs - Swagger

- Swagger is an open standard for defining a REST API:
  - **http://swagger.io/**
- Along with the specification, there is a set of open source tooling that can be used to interact with Swagger documents and the REST APIs that they describe.
- A Swagger document includes definitions of the resources, operations, and parameters in a REST API. It can also include JSON Schema that describes the structure of the request and response bodies to an operation.
- A Swagger document can be thought of as the REST API equivalent of a WSDL document for a SOAP web service.
- Integration Bus supports Swagger 2.0. The specification for Swagger 2.0 can be found at:
  - **https://github.com/swagger-api/swagger-spec/blob/master/versions/2.0.md**
- REST APIs in Integration Bus are described by a Swagger 2.0 document.
- You can build a REST API by importing a Swagger document.
- Alternatively, you can now build a REST API from scratch using the toolkit!

REST APIs – Open API Initiative

“SAN FRANCISCO, November 5, 2015 – The Linux Foundation, the nonprofit organization dedicated to accelerating the growth of Linux and collaborative development, today is announcing the Open API Initiative. Founding members of the Open API Initiative include 3Scale, Apigee, Capital One, Google, IBM, Intuit, Microsoft, PayPal, Restlet and SmartBear.

The Initiative will extend the Swagger specification and format to create an open technical community within which members can easily contribute to building a vendor neutral, portable and open specification for providing metadata



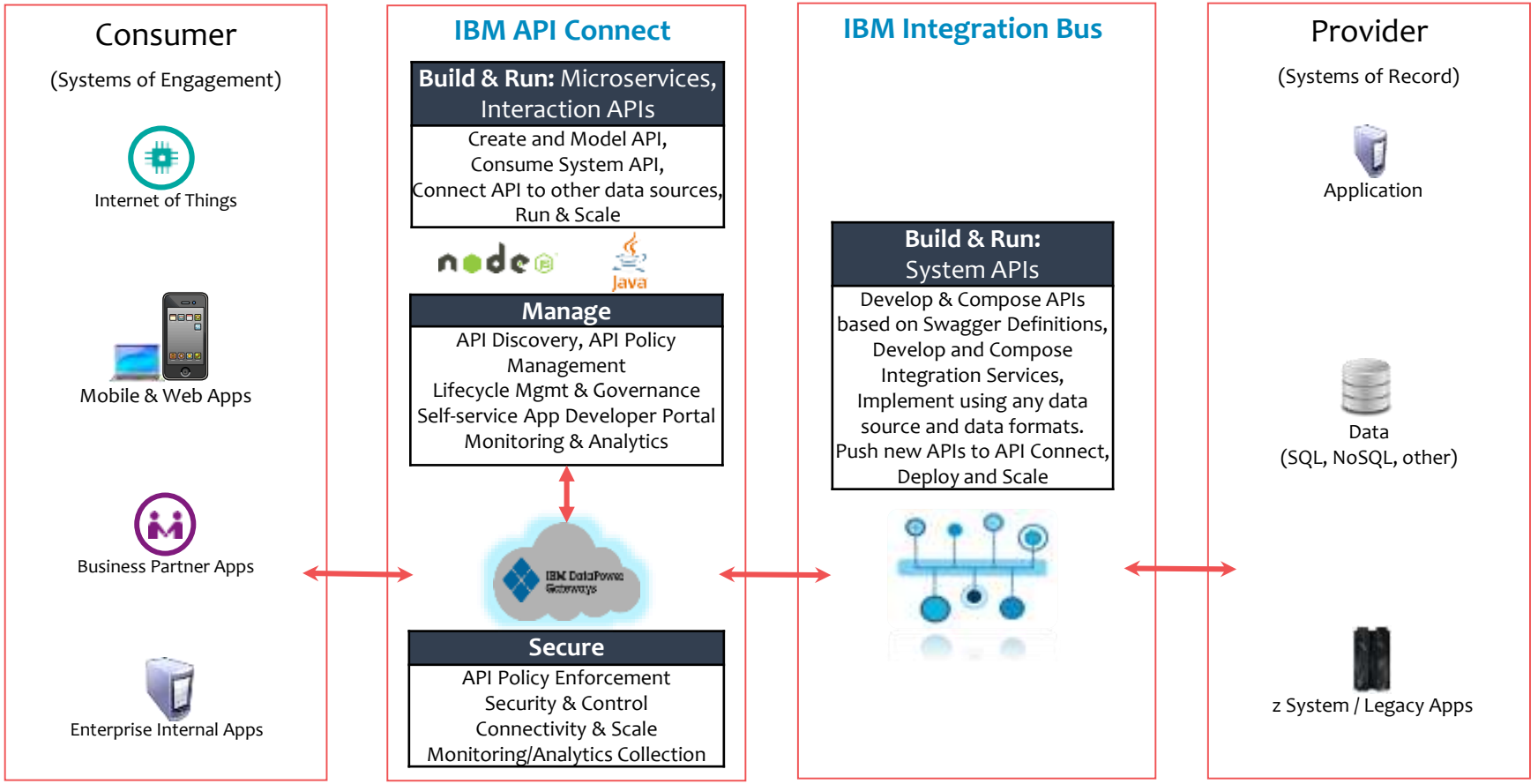
https://openapis.org/

REST APIs – current trends

- Everyone is building REST APIs!
- The API Economy is encouraging businesses to expose their applications and data via REST APIs, so that developers can consume those APIs in order to build new applications.

- One current trend is towards building applications as a collection of micro services. Micro services can expose their functionality as a REST API.
- For example, a weather micro service might provide a REST API with a single operation for accessing weather information for a supplied location.

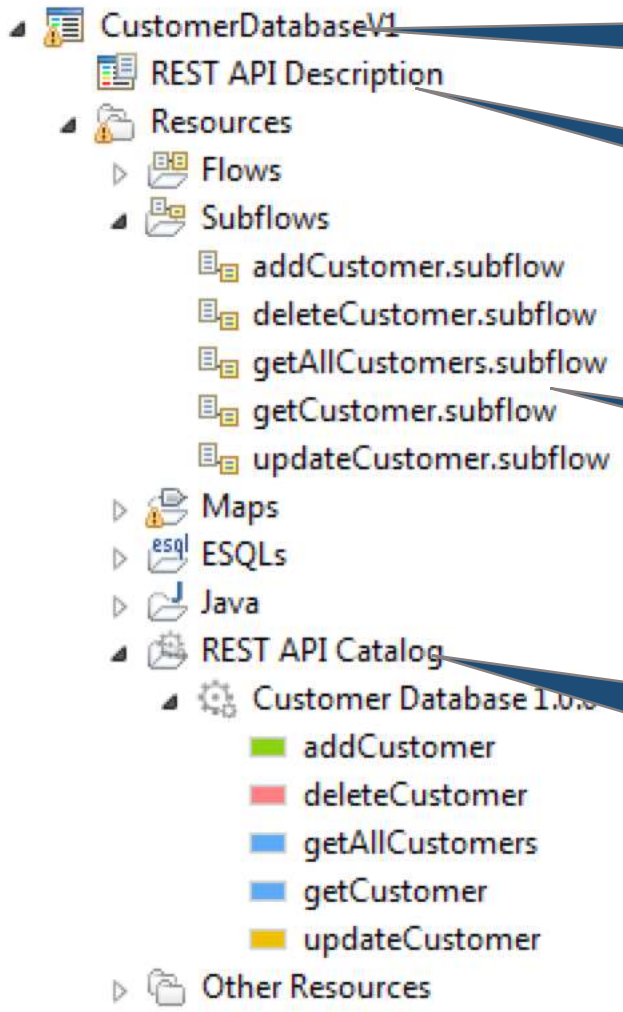
REST APIs – what's IBM doing?



REST APIs – IIB functionality

Version	Functionality
10.0.0.0	First-class support for building REST APIs and handling inbound requests.
10.0.0.2	Push to API Connect v4 from Toolkit.
10.0.0.4	Build a REST API from scratch using the Graphical Editor in the Toolkit. Graphical mapping support for JSON Schema defined in .json format
10.0.0.5	Push to API Connect v5 from WebUI and command line.
10.0.0.6	First-class support for handling outbound requests using a REST Request node. Graphical mapping support for JSON Schema defined in .yaml format
10.0.0.8	Support for passing User Context between REST Async Request/Response nodes. Support for allOf, one of, anyOf in graphical mapper for JSON schema

REST APIs – REST APIs in Integration Bus



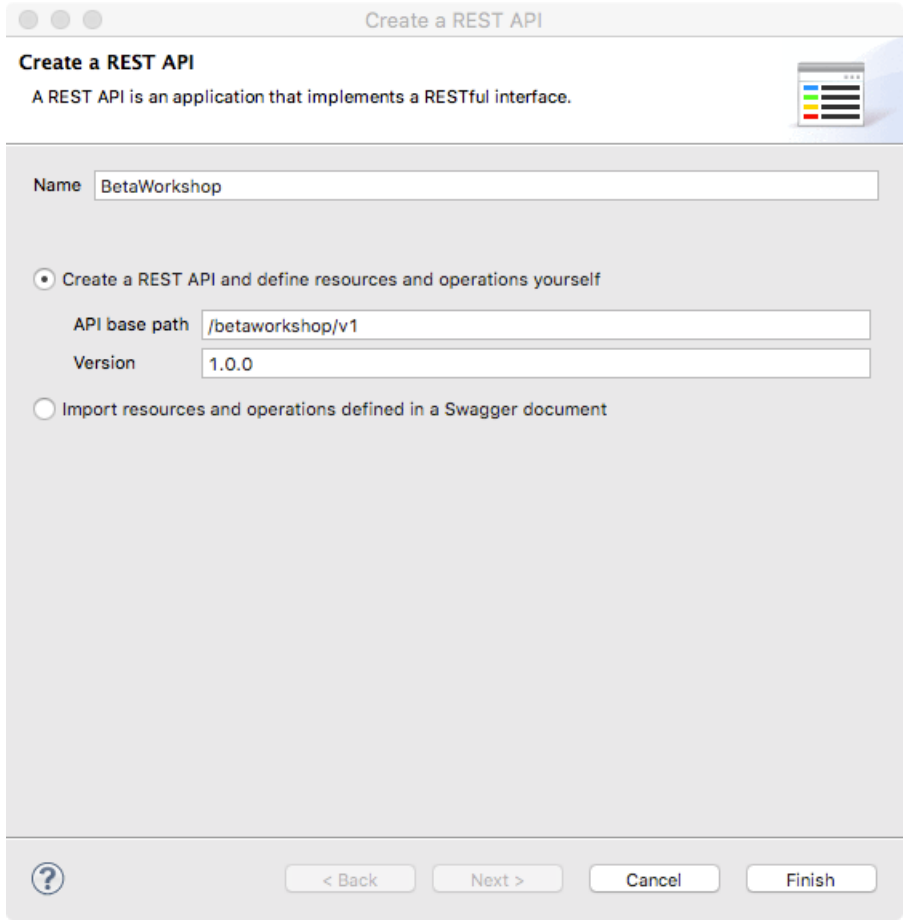
For Integration Bus V10, we have introduced a new type of project, or container, called a REST API.

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 new category that appears in the REST API project and shows all REST APIs and operations that are available to be invoked.

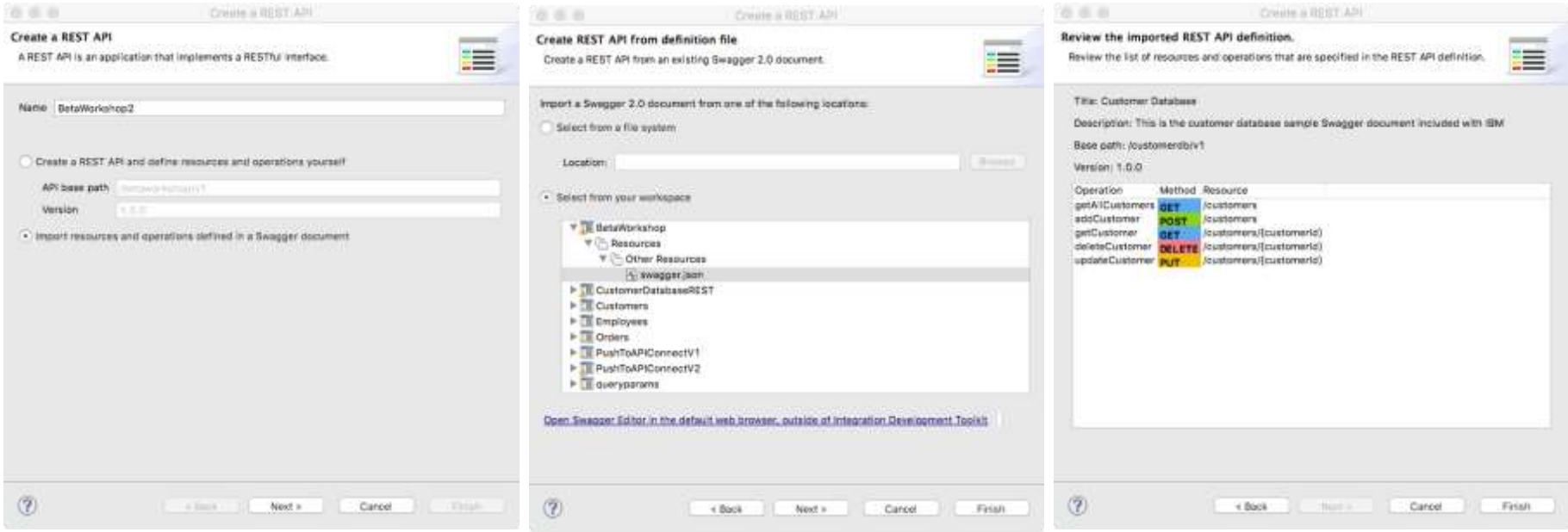
REST APIs - new project wizard



- 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.

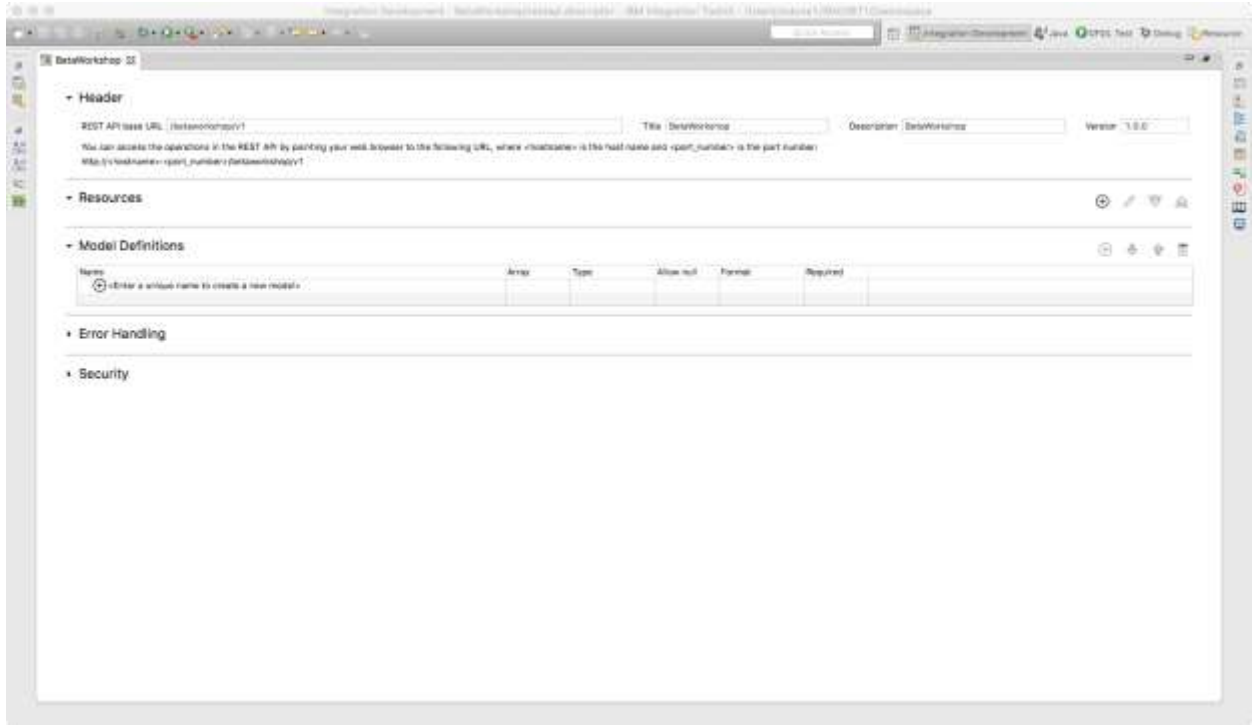
REST APIs – new project wizard

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



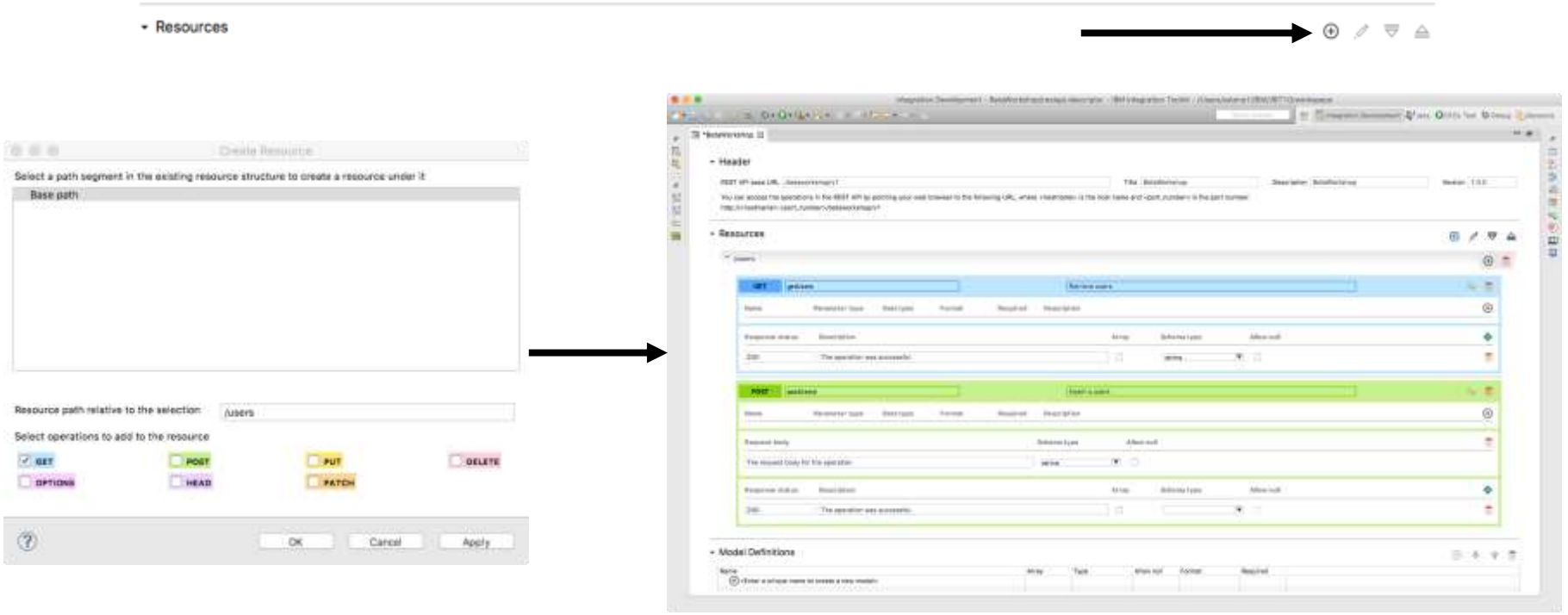
REST APIs – REST API Editor

- 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.



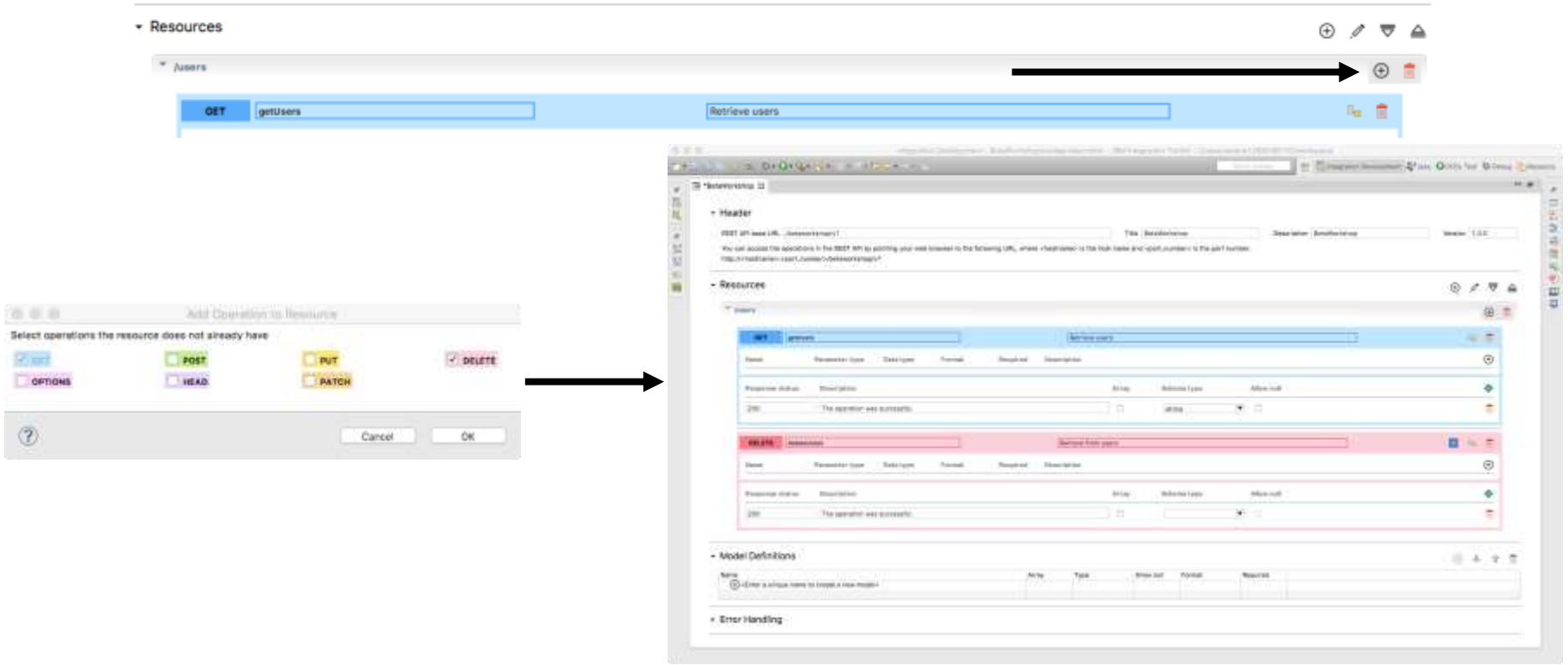
REST APIs – defining resources

- We can add new resources into the REST API by clicking the (+) symbol on the resources section:



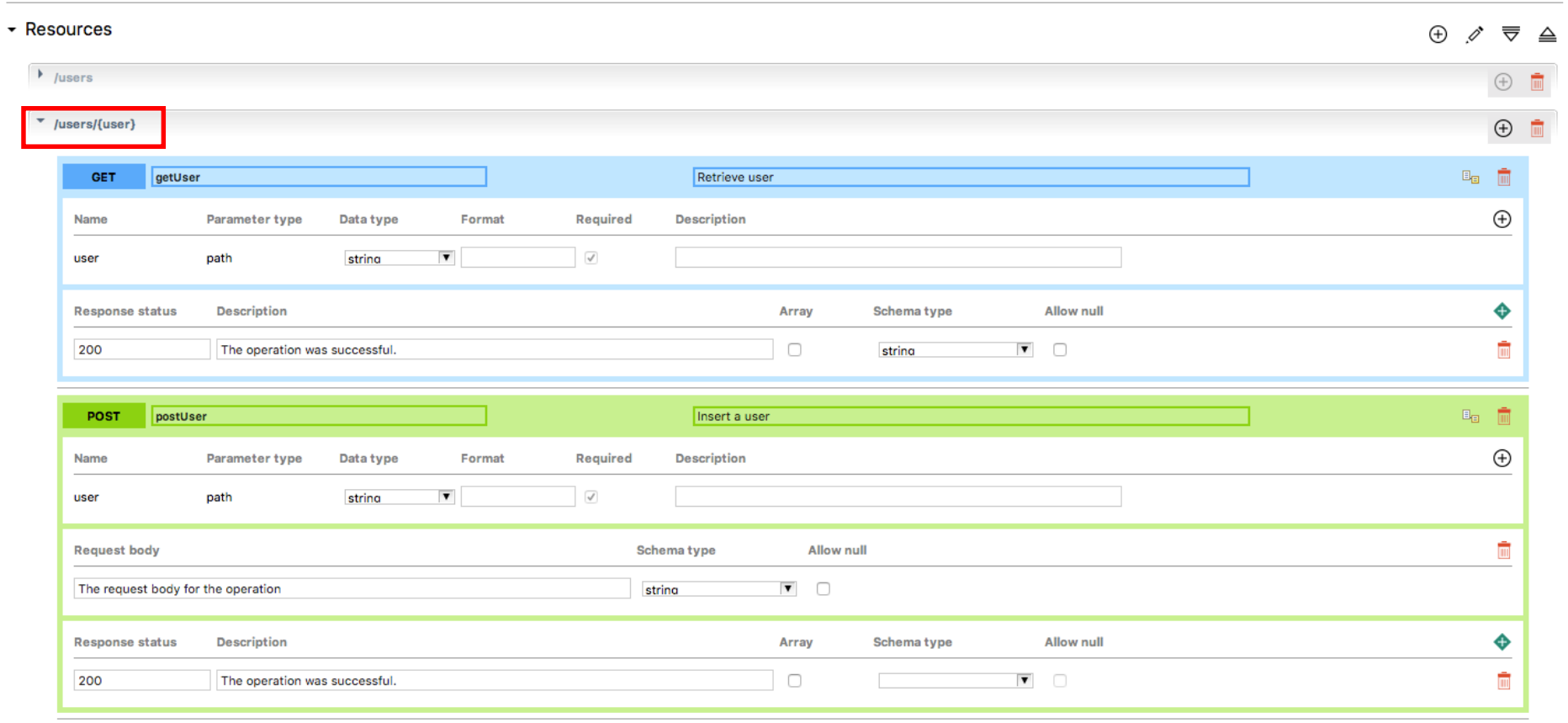
REST APIs – defining operations

- The new resource dialog adds in default operations for you, but you can add new ones:



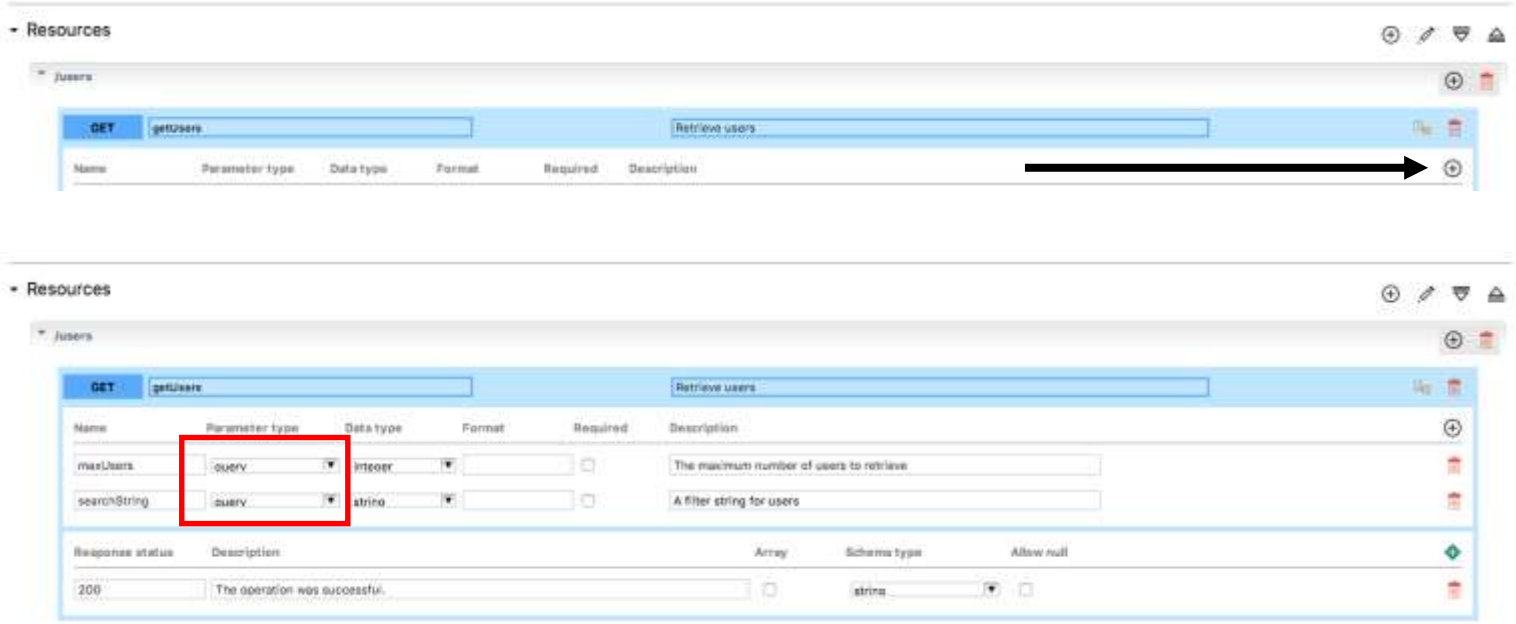
REST APIs – defining parameters

- 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:



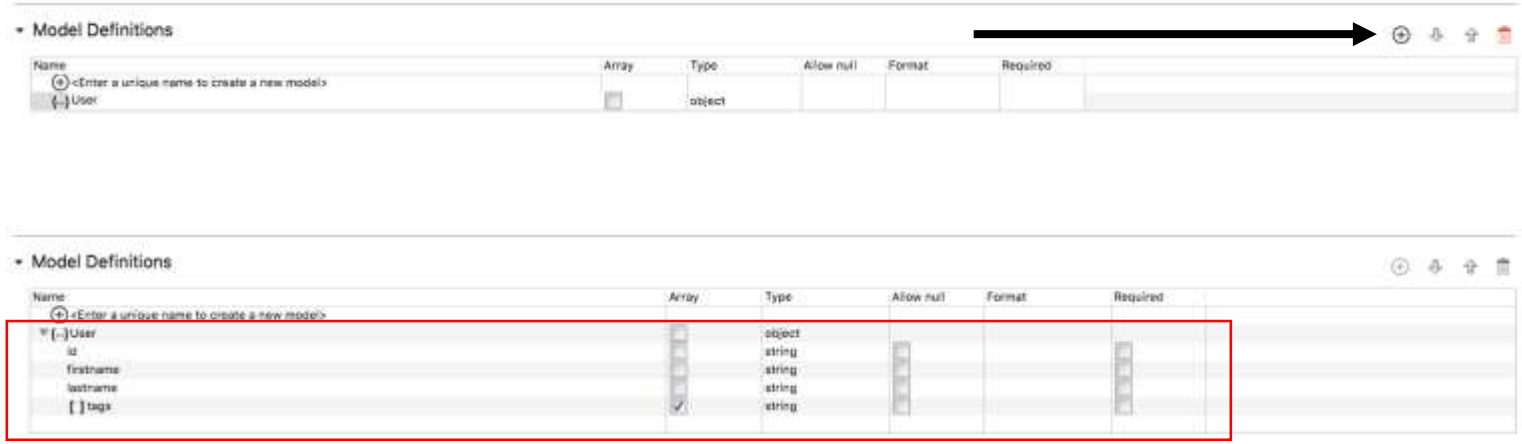
REST APIs – defining parameters

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



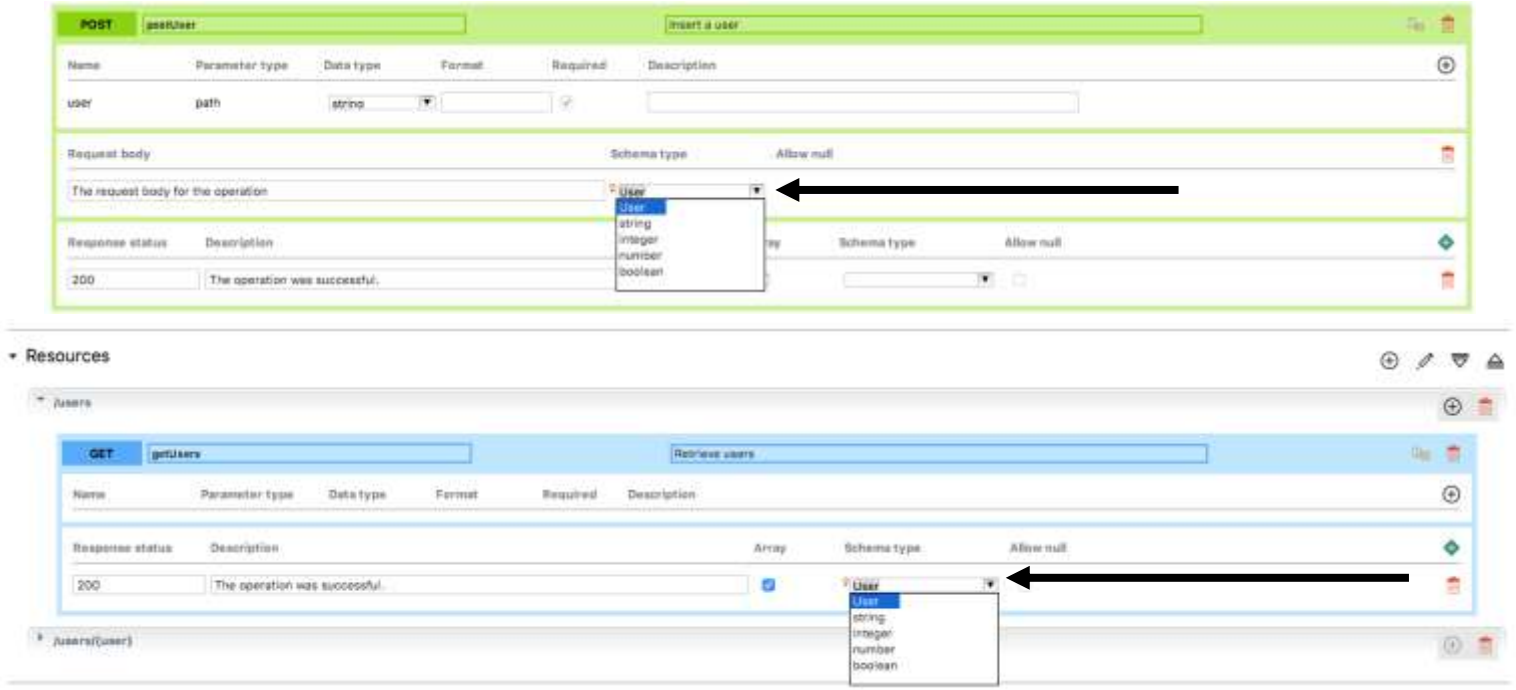
REST APIs – defining models

- 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:



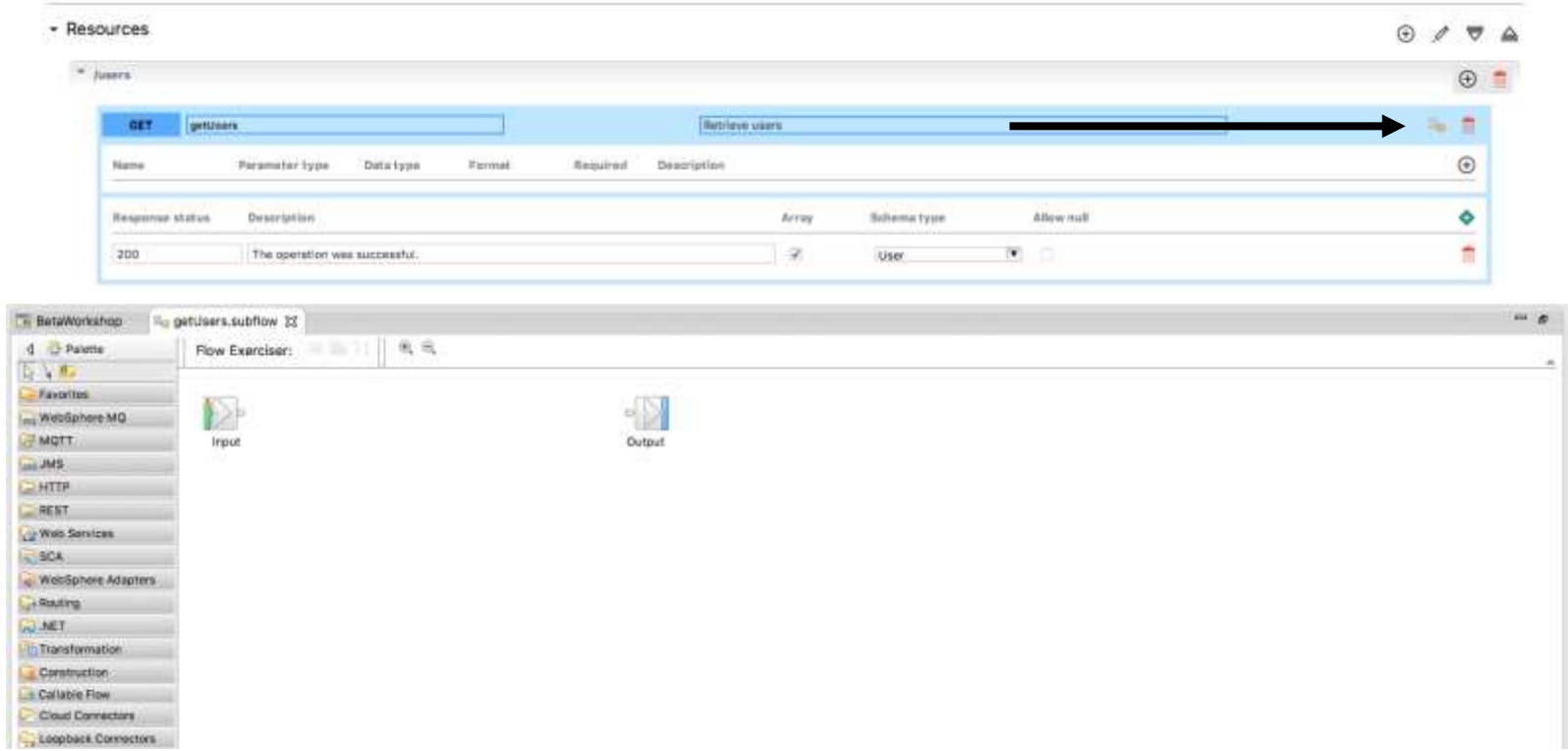
REST APIs – referencing models

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



REST APIs – 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.

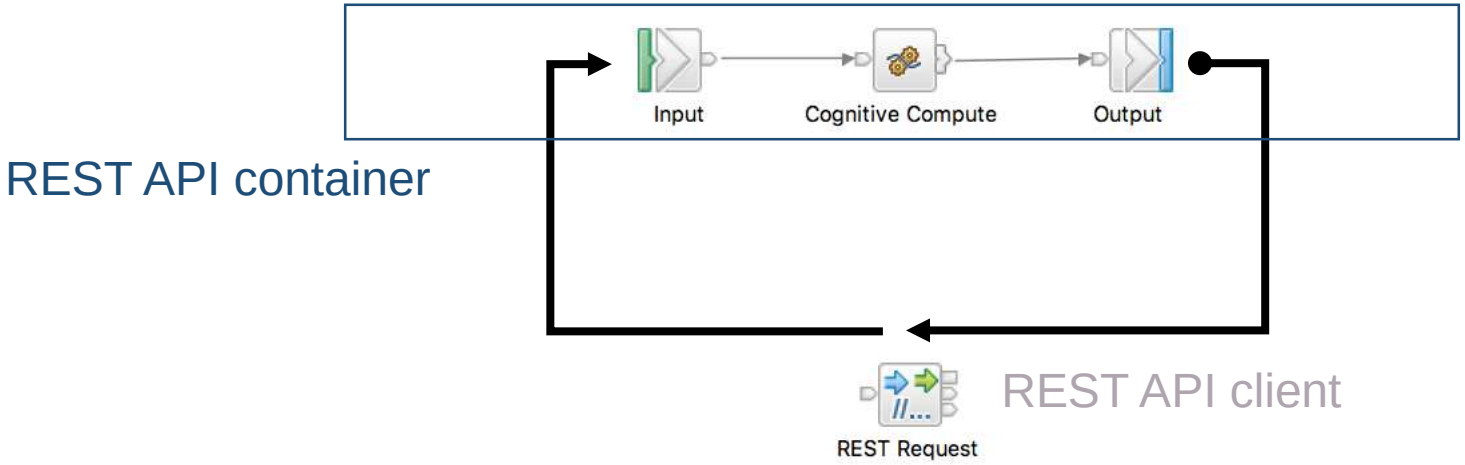


REST APIs – implementing error handlers and HTTPS

- 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:

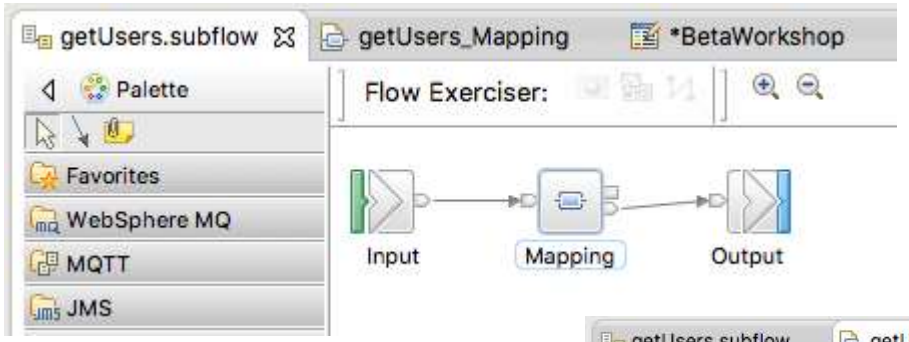
REST APIs – implementing an operation

- 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.



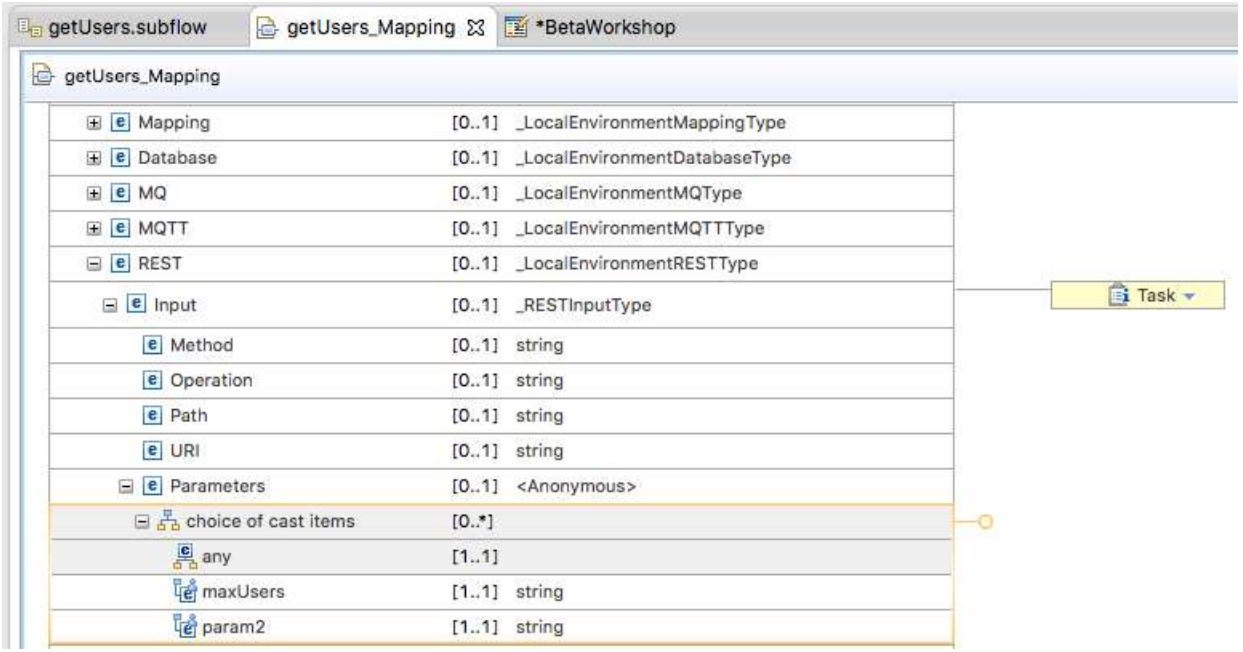
REST APIs – accessing parameter values

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.



REST APIs – 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 programming languages included in Integration Bus (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:

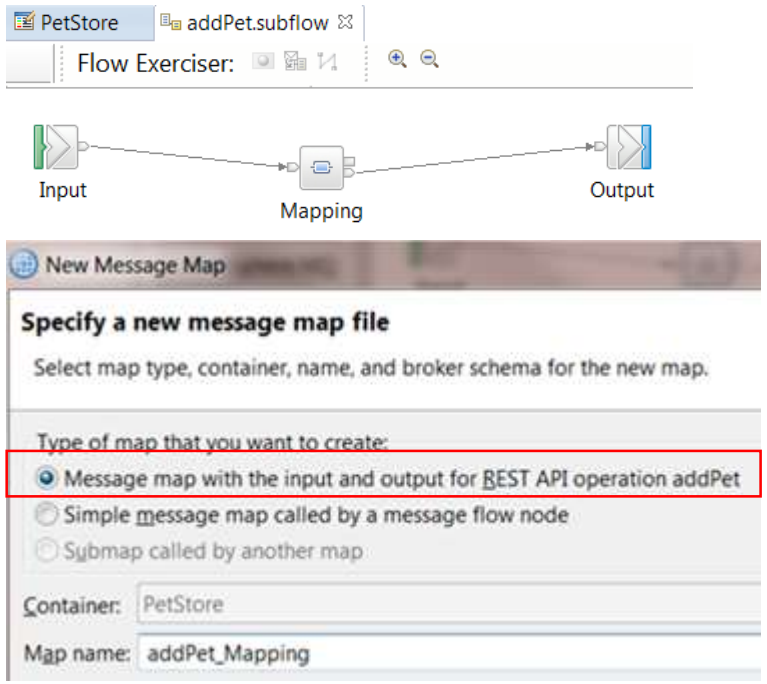
```
MbElement maxElement = inLocalEnvironment.getRootElement().getFirstElementByPath("/REST/Input/Parameters/max");
int max = -1;
if (maxElement != null) {
  max = Integer.valueOf(maxElement.getValueAsString());
}
```

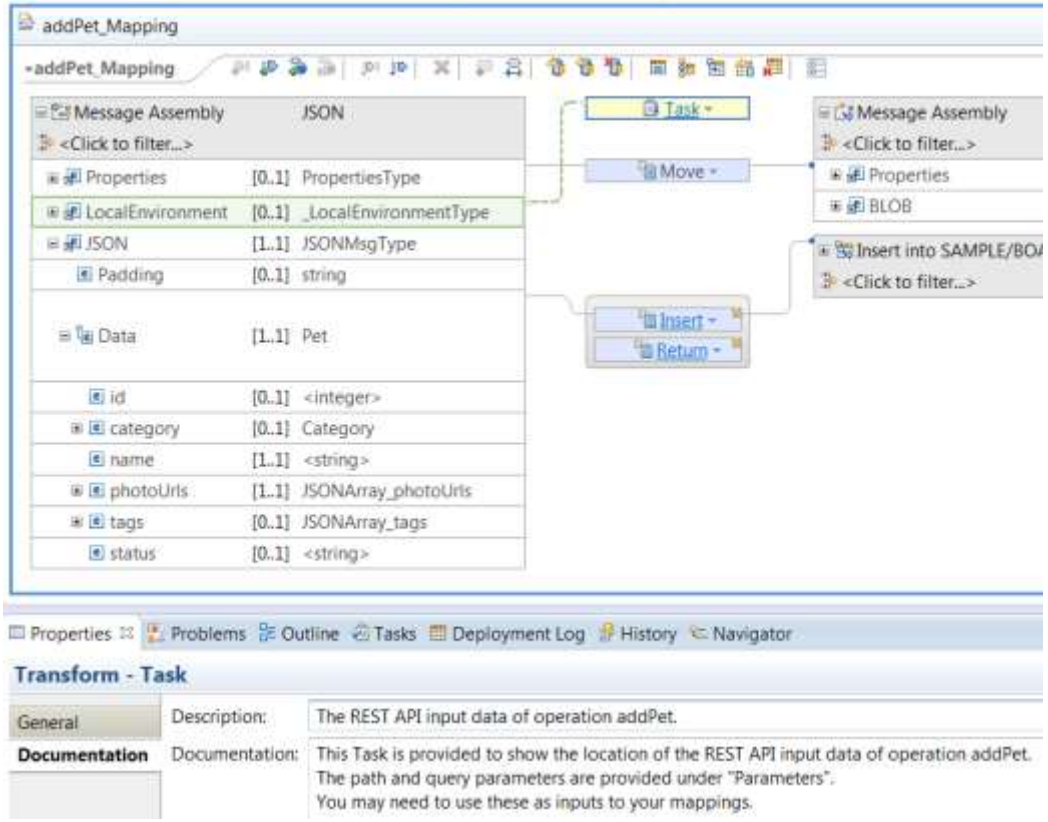
.NET:

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

REST APIs – mapping JSON request/response bodies

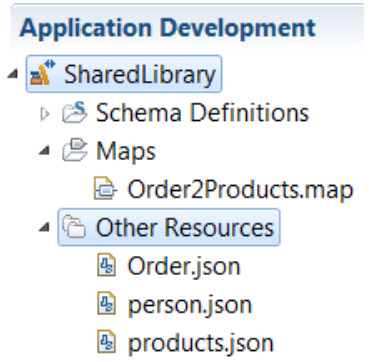
- Graphical mapping enhancements introduce support for JSON Schema. The mapper ”knows” that it is in a REST API, and can automatically pick the models for you:

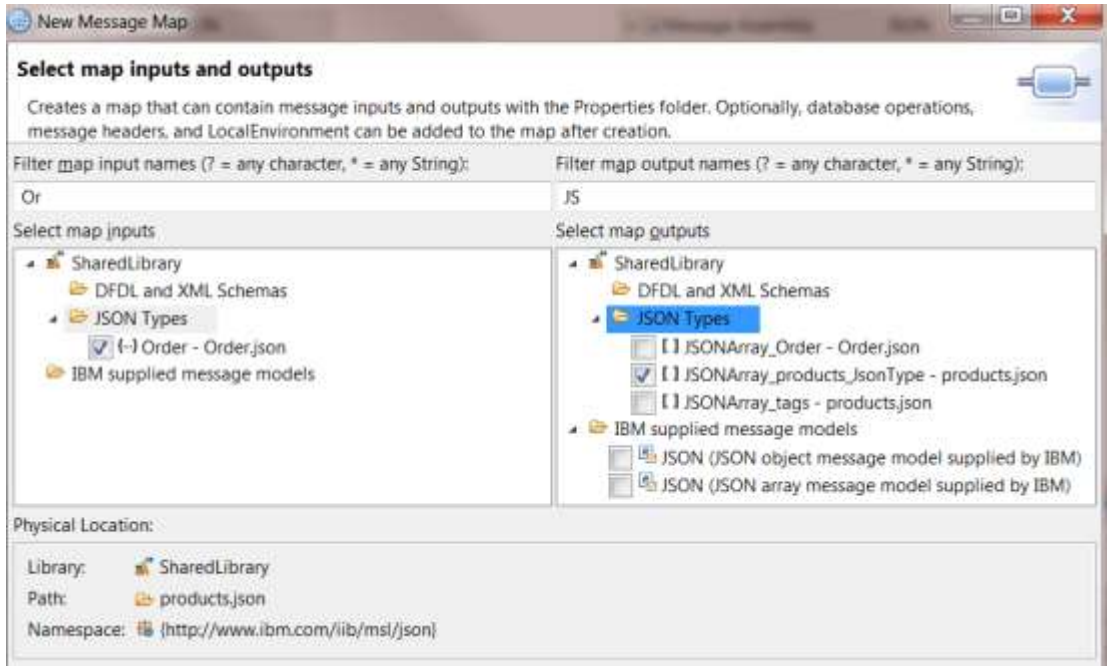




REST APIs – general mapping enhancements

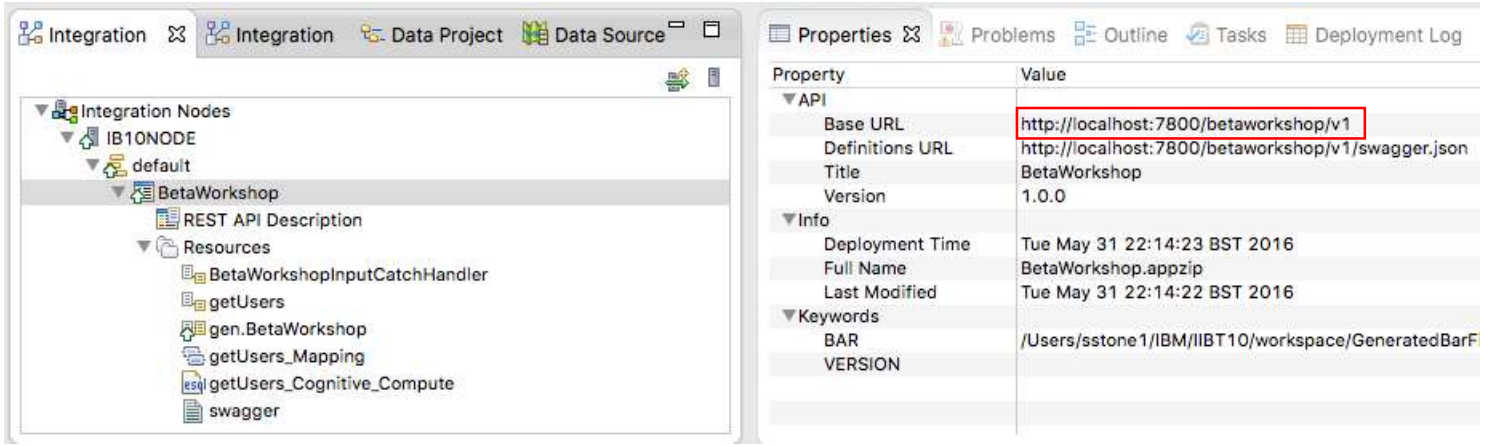
- The JSON Schema support in the mapper can also be used outside of a REST API.
- At present, the JSON Schema and the map **must** be col-located in the same container





REST APIs – 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.

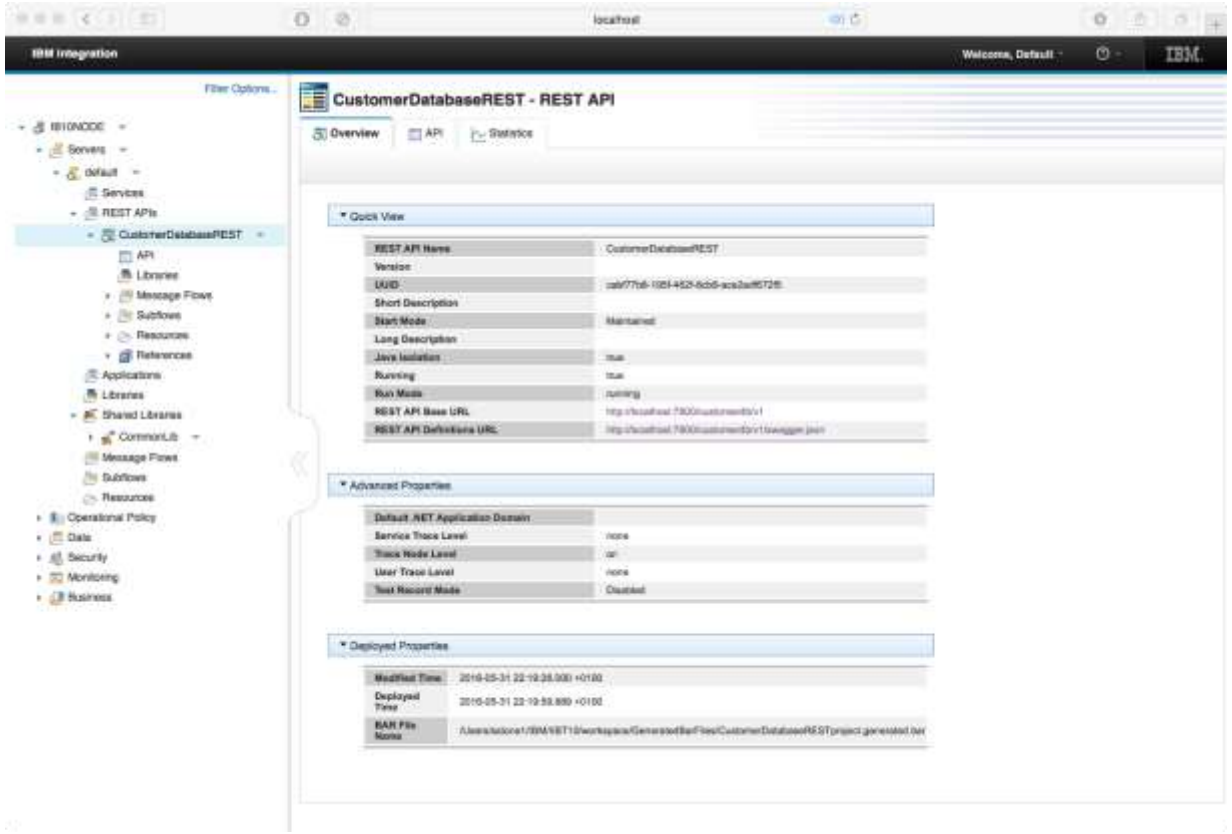


- The base path of the REST API can be used to isolate a REST API from other REST APIs, in a similar way to a context root for a J2EE application.

- 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.

REST APIs - administration

- All administrative and operational controls that are available for applications in Integration Bus 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 Integration API and administrative REST API for programatically interacting with deployed REST APIs.

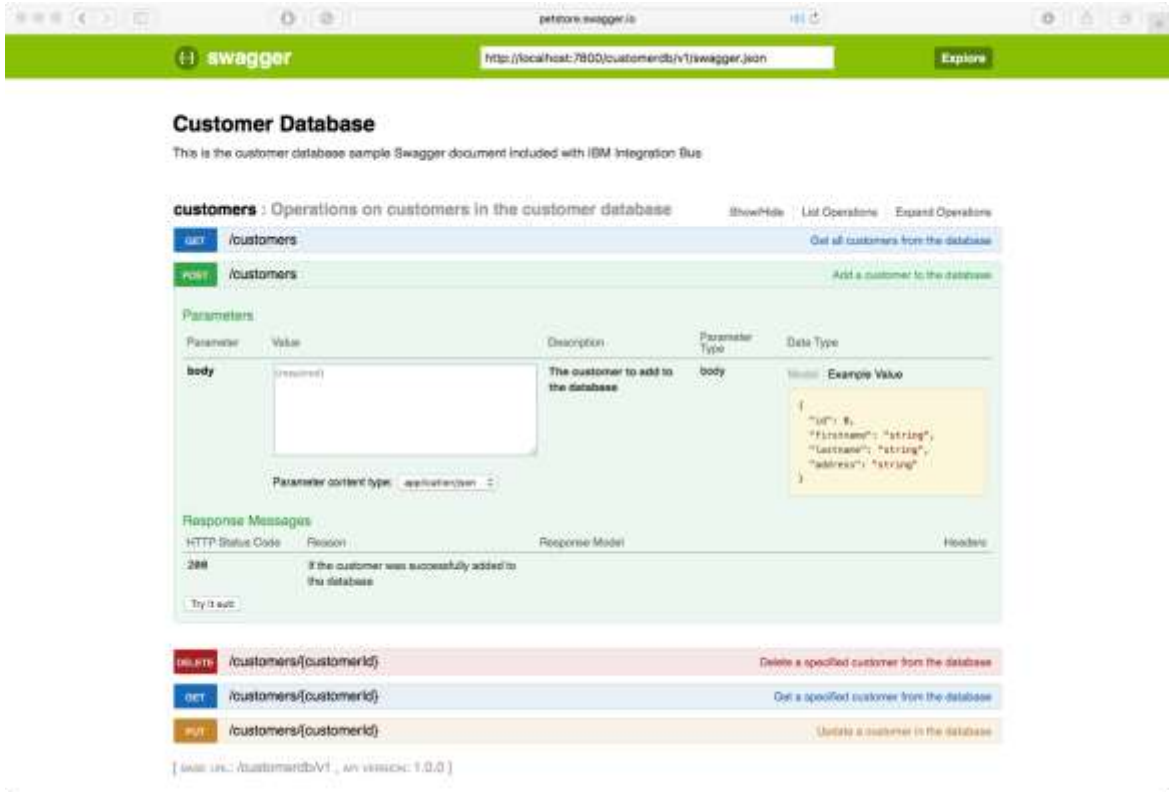


- The web user interface has also been extended to provide information about the resources, operations, and parameters that are available in a deployed REST API.

REST APIs – 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.

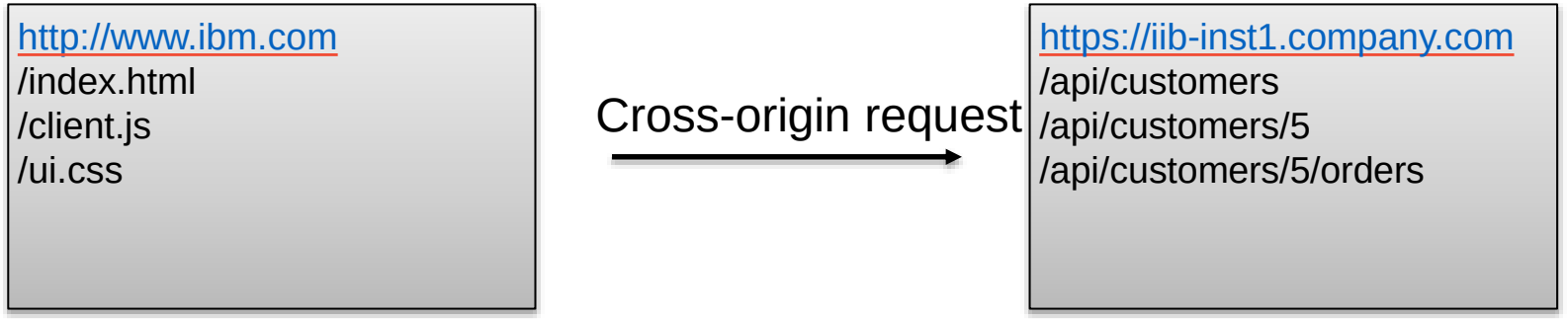
REST APIs – Swagger UI



- 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 (more on that shortly!).

REST APIs – Cross-Origin Resource Sharing (CORS)

- A web page has an origin. The origin of a web page is the scheme, host, and port of the web server that is hosting the web page.
  - For example, the origin of https://localhost:8080/test/index.html is https://localhost:8080.
- When a web page makes a request for another web page or resource on the same origin, a same-origin request is made.
- However, when a web page makes a request for another web page or resource on a different origin, a **cross-origin request** is made. This requires special support from the HTTP server that the cross-origin request is being made to. Without this support, the request is rejected by the web browser.



REST APIs – Cross-Origin Resource Sharing (CORS)

- If you are making an HTTP request from a web page to a REST API or other HTTP service deployed to Integration Bus, it is likely that a cross-origin request will need to be made.

- Integration Bus V10 now includes built in support for CORS. At V10 GA, this support was limited to the HTTP listener for the integration server, but as of V10 FP1 is available for the HTTP listener for the integration node as well.

- The support is disabled by default, but can be easily enabled with a single command:
  - `mqsichangeproperties IB10NODE -e default -o HTTPConnector -n corsEnabled -v true`
  - `mqsichangeproperties IB10NODE -b httplistener -o HTTPConnector -n corsEnabled -v true`

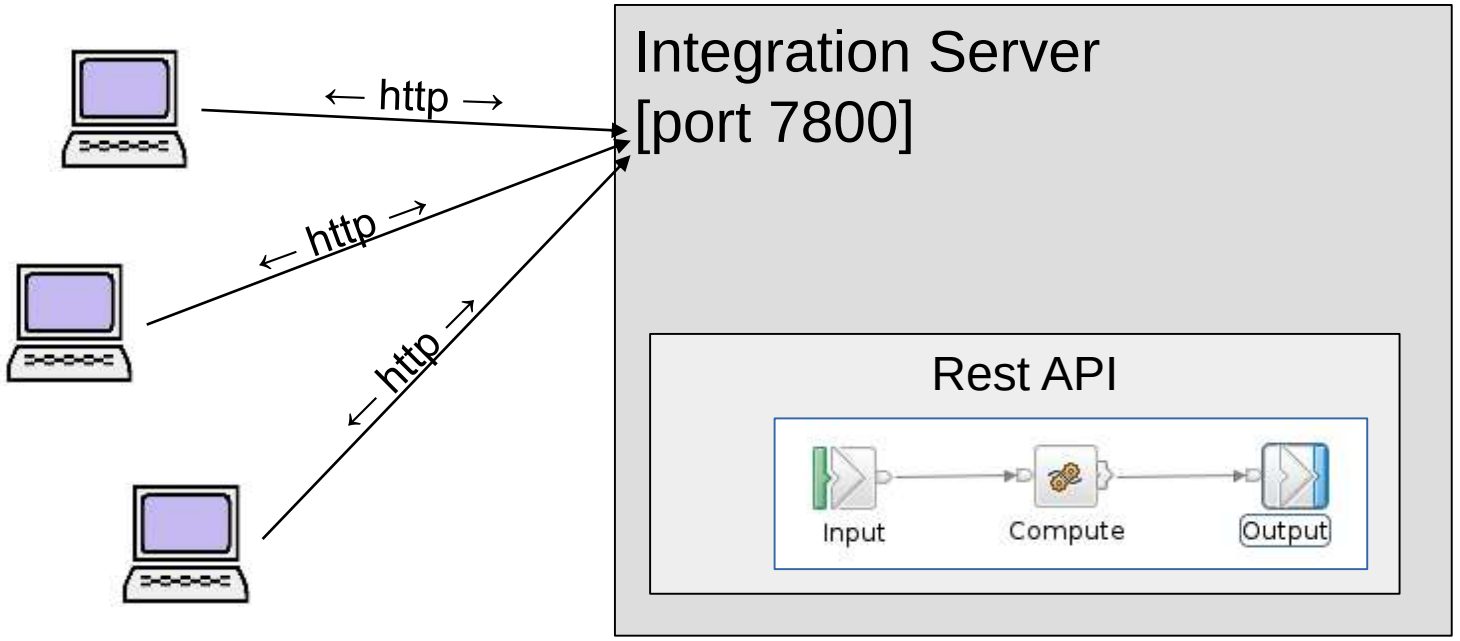
- When CORS settings are modified, the changes are effective immediately – there is no need to restart the integration server or node.

Demo: Develop a
REST API in IIB

Pushing a REST API to API Connect

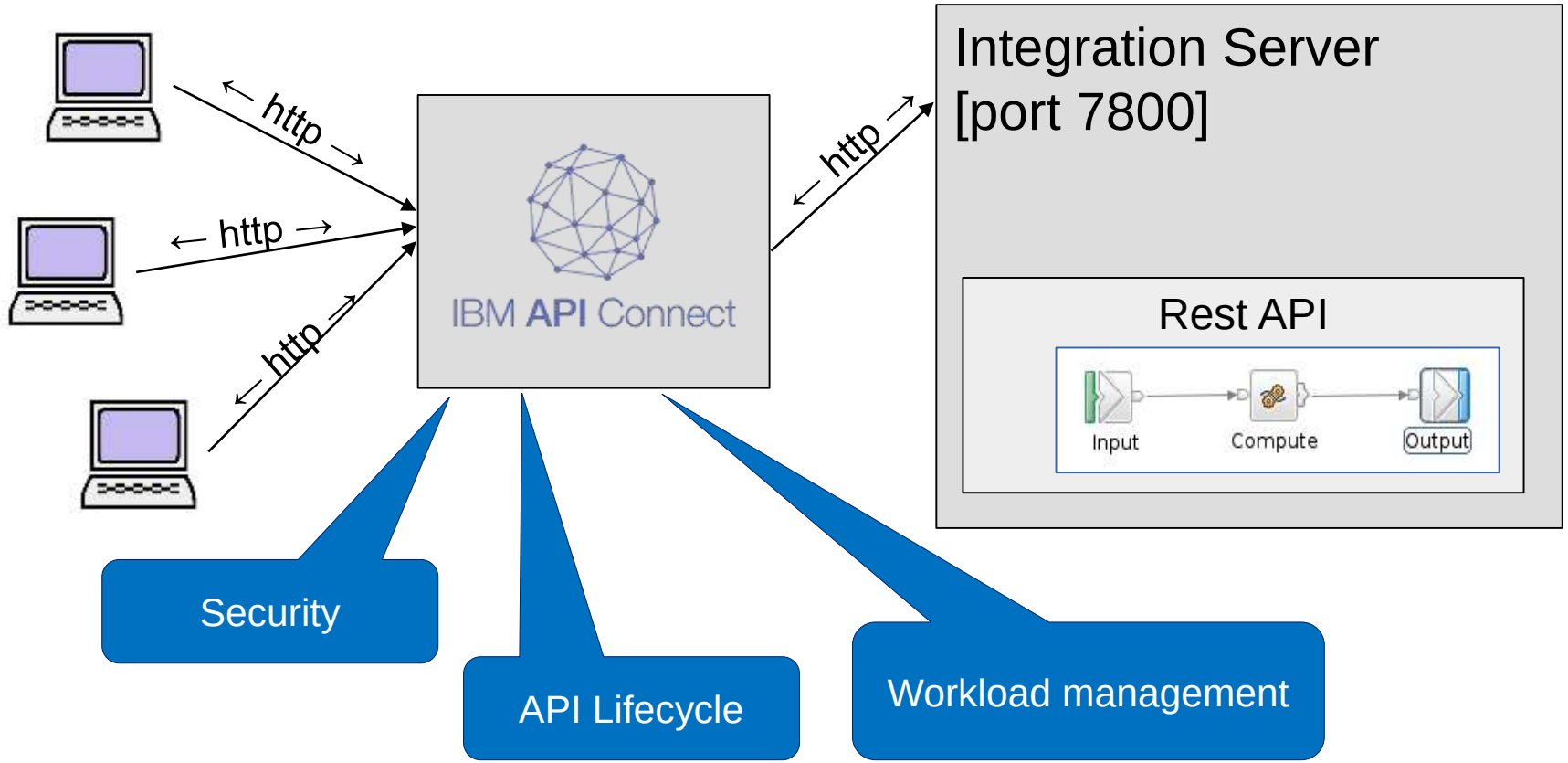
REST APIs developed in IIB

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



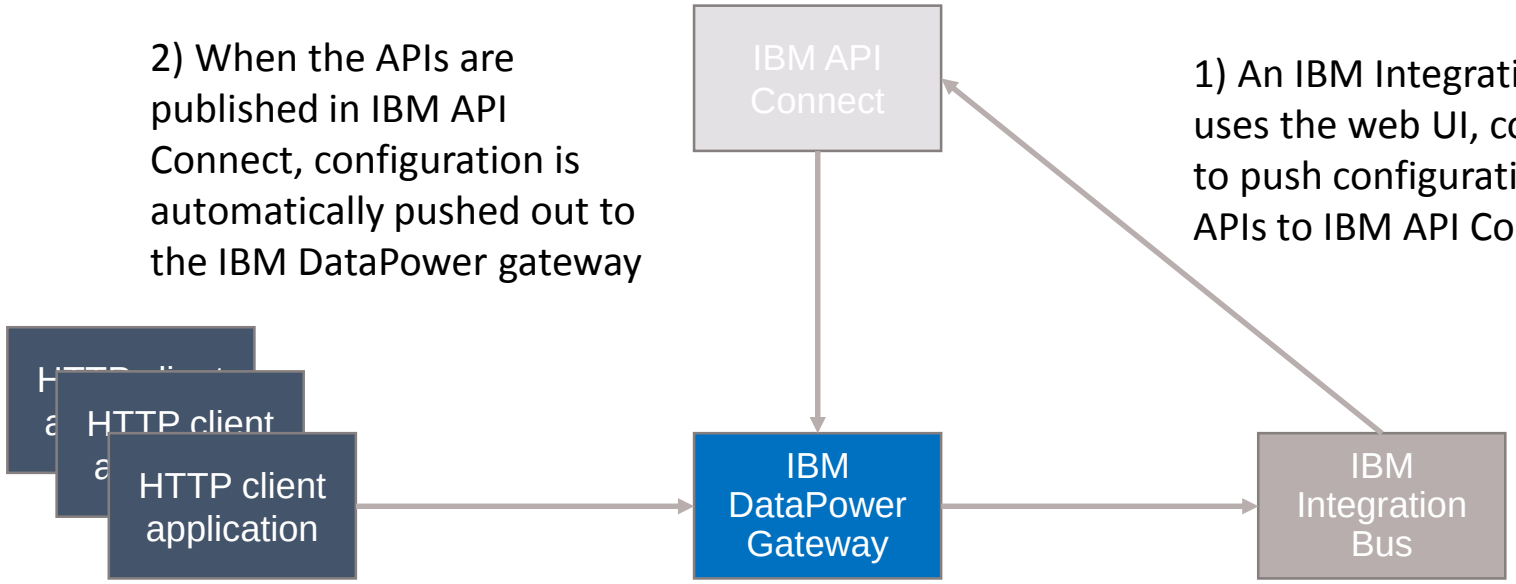
IBM API Connect and IIB

- 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

2) When the APIs are published in IBM API Connect, configuration is automatically pushed out to the IBM DataPower gateway



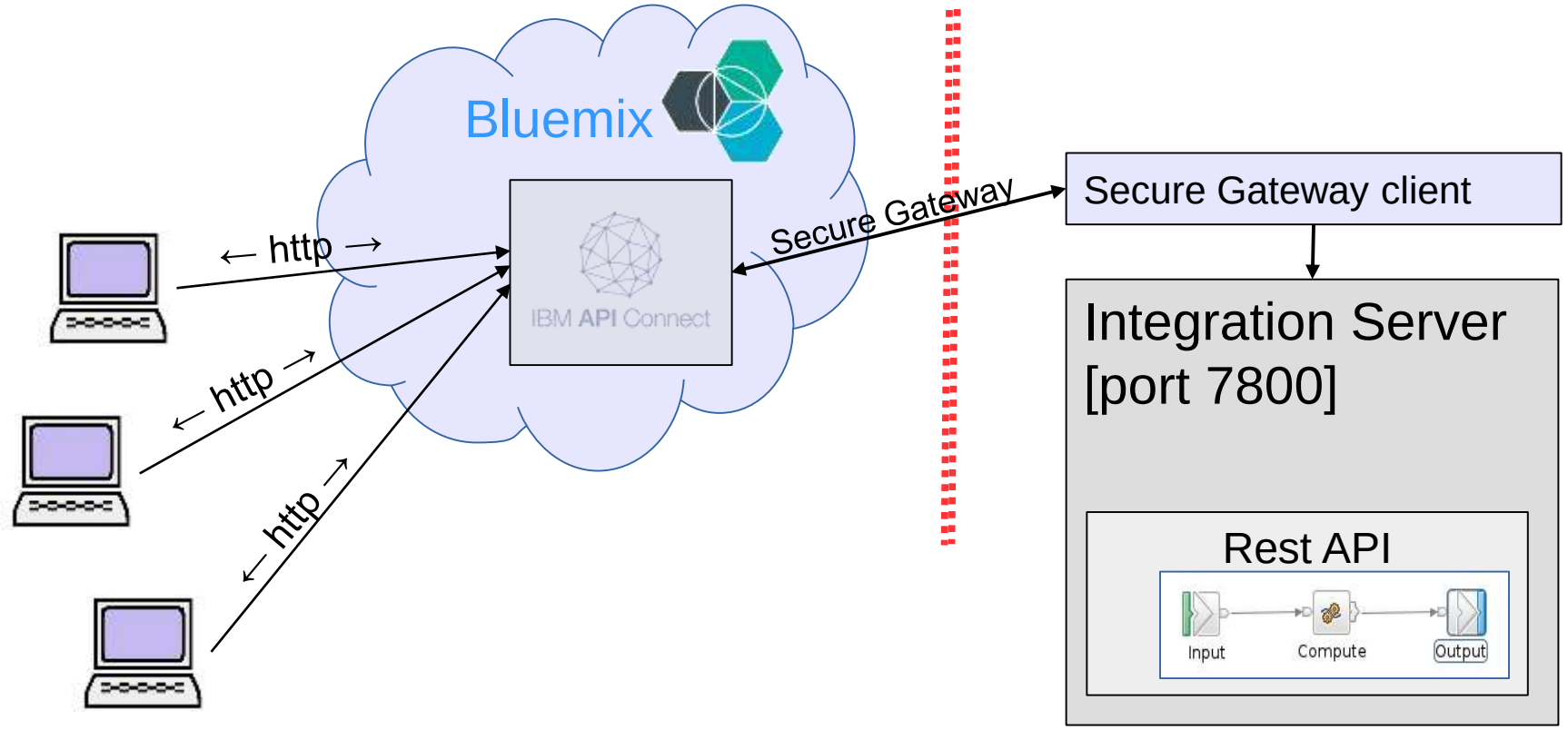
1) An IBM Integration Bus administrator uses the web UI, command line, or toolkit to push configuration for deployed REST APIs to IBM API Connect.

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 Bluemix

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

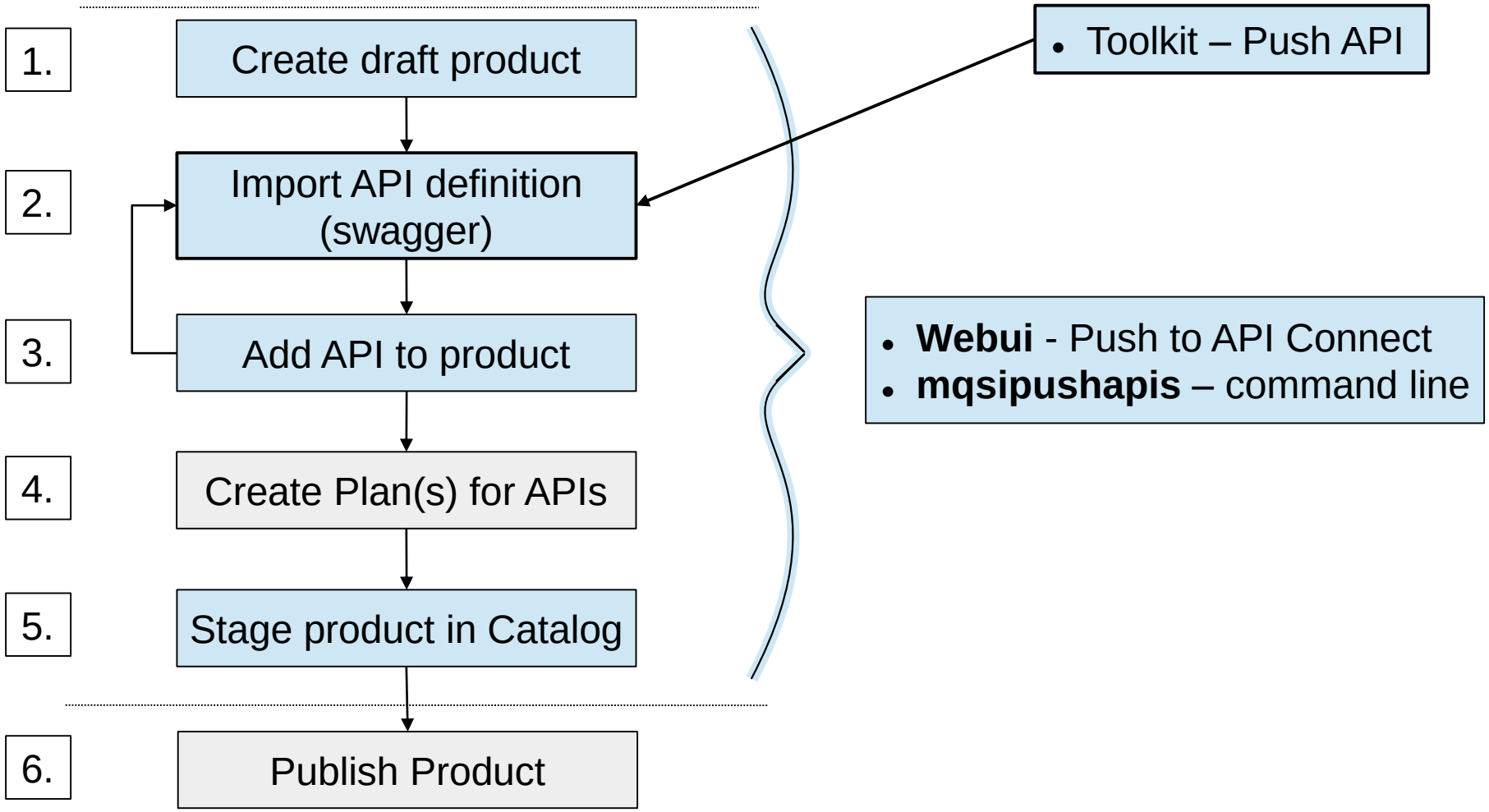


IBM API Connect Concepts

- APIs
  - An API is a collection of http operations. An API is identified by an API name and version.
- Plan
  - A plan defines the operational limits (rate, authorisation) of an API or APIs.
- Product
  - A product is a collection of APIs and Plans. A Product is identified by a Product name and version.
- Catalog
  - Products must be staged to a catalog before they can be invoked. A catalog behaves as a logical partition of the Gateway and the Developer Portal.

    (A 'Sandbox' catalog is useful when developing APIs, allowing APIs to updated and deleted without the deprecating / retiring the Product first.)

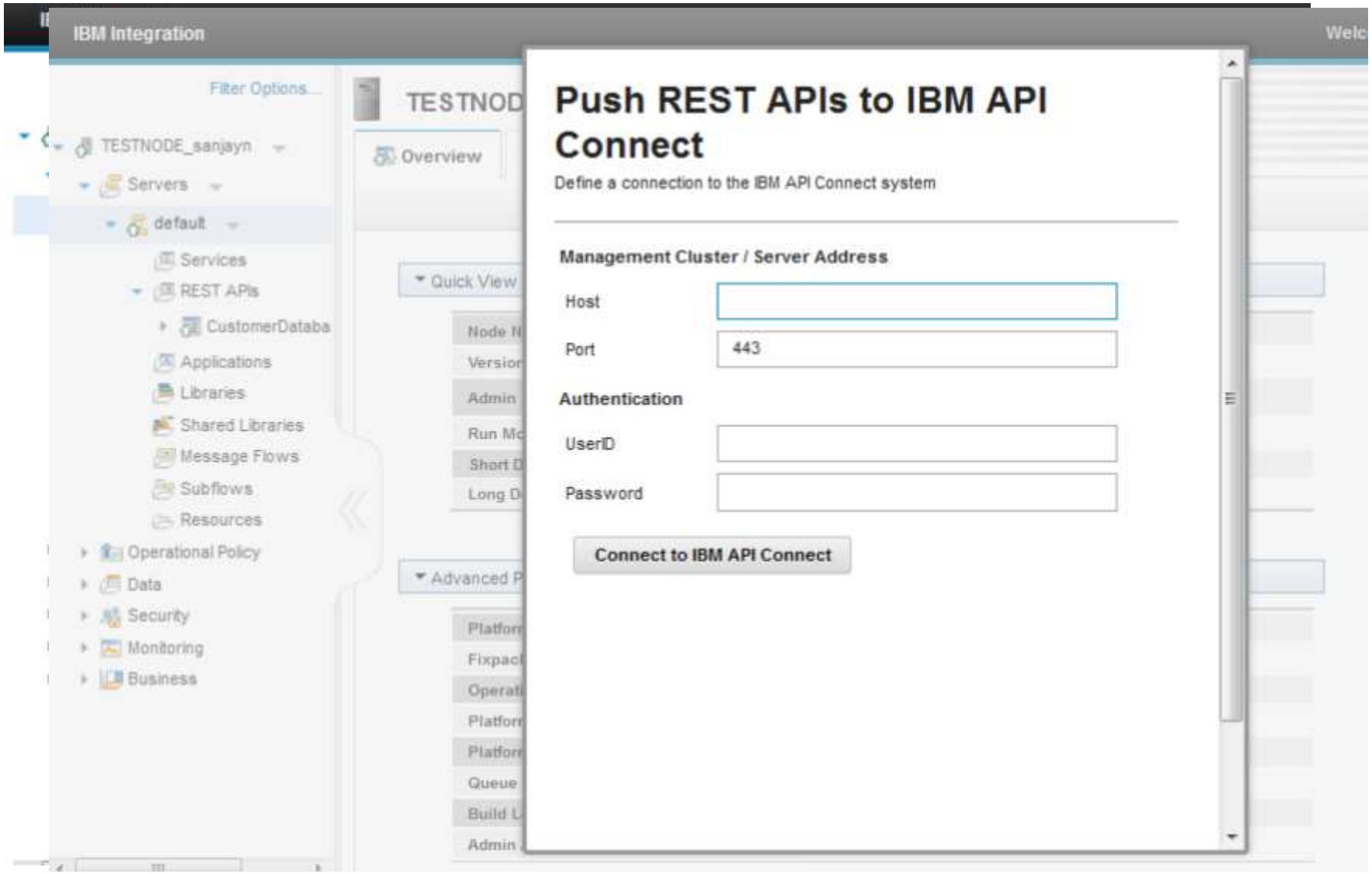
Exposing an API in IBM API Connect



Pushing API's from IIB 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

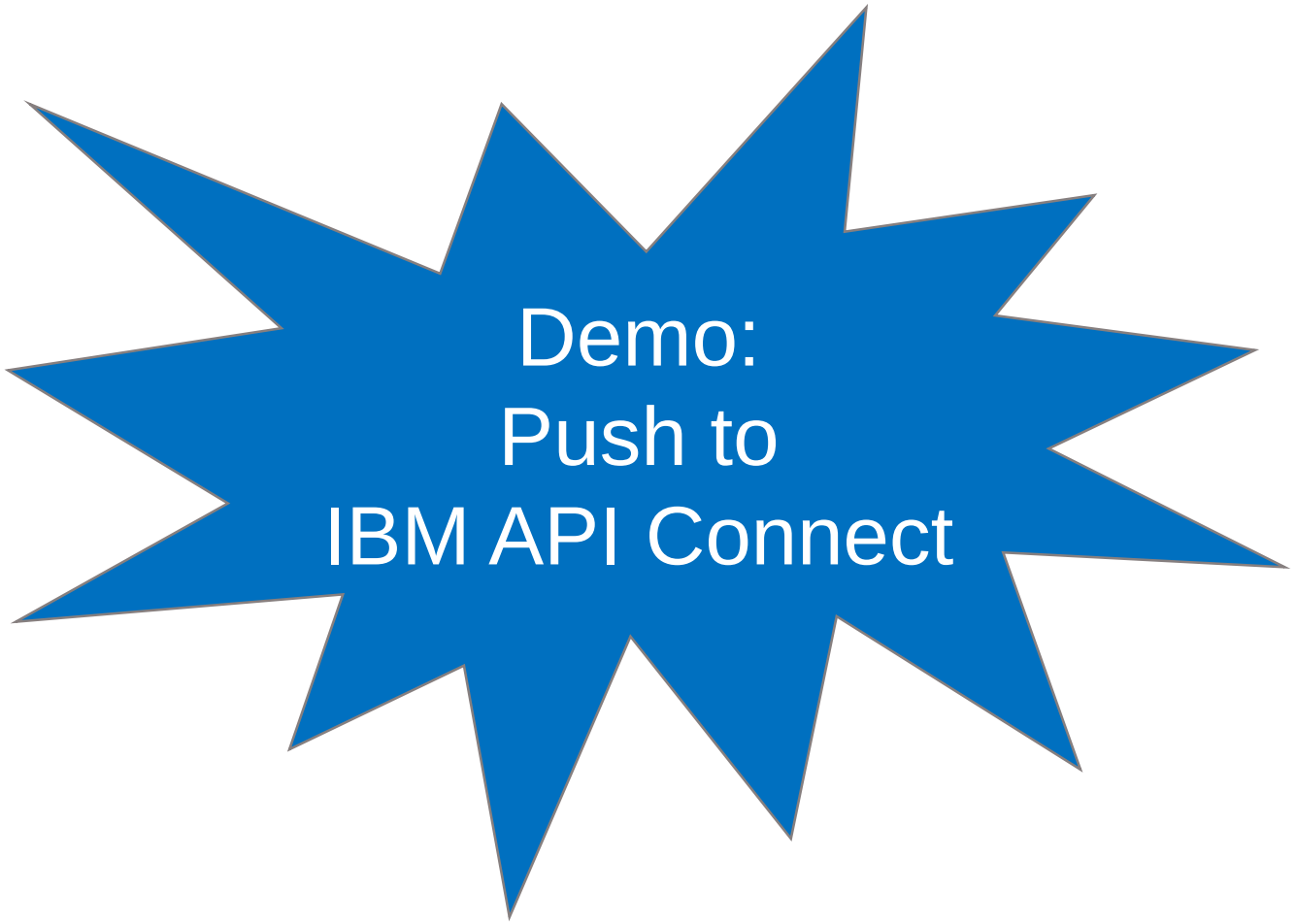
Pushing API's from IIB to IBM API Connect: Web UI



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

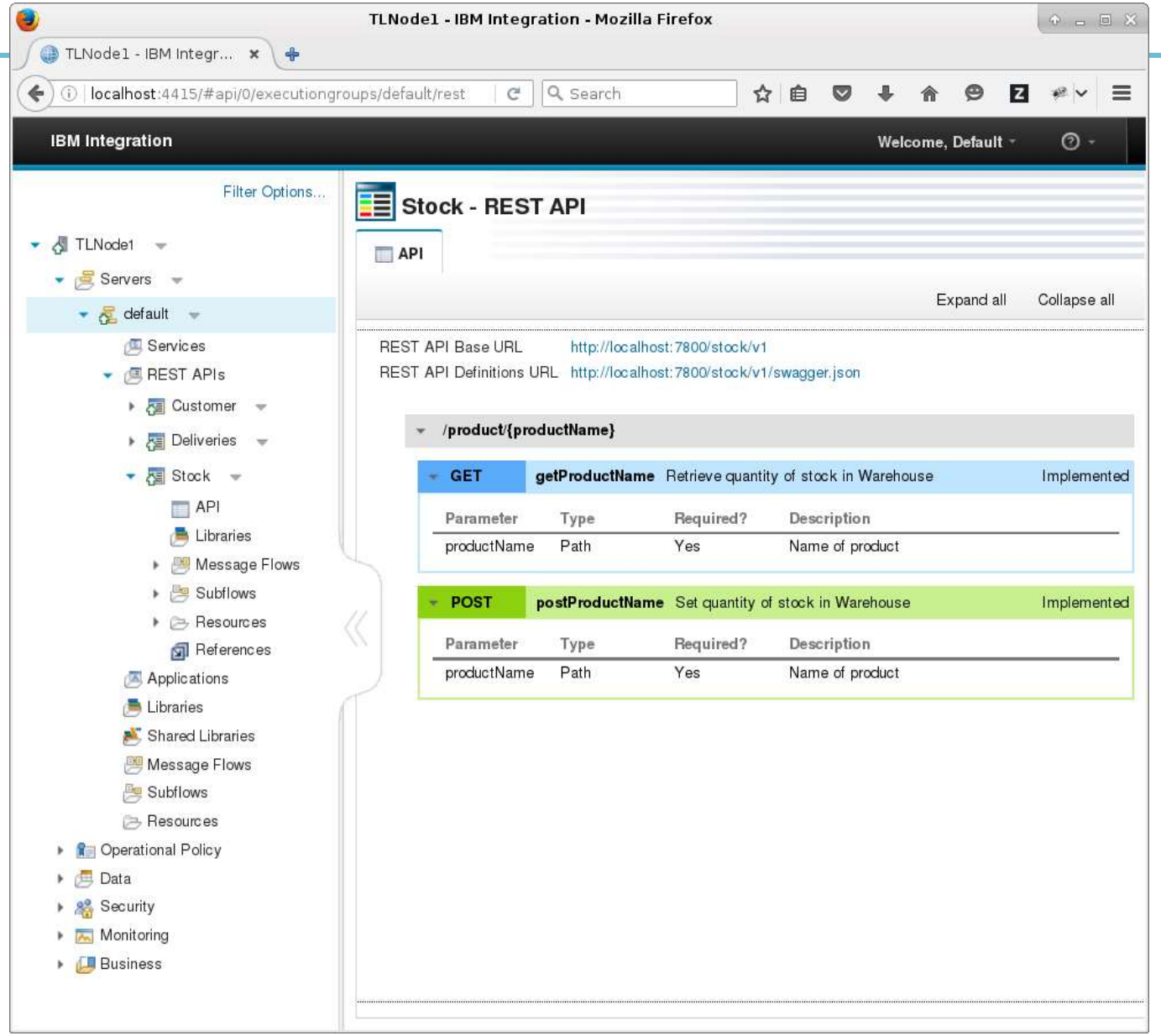
```
mqsipushapis integrationNodeSpec -e integrationServerName -t apiConnectHost -g apiConnectPort
        -u apiConnectUser -a apiConnectPassword -o apiConnectOrganization
        [-c apiConnectCatalogTitle] -r apiConnectProductTitle [-d apiConnectProductName]
        [-s apiConnectProductVersion] -k 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.
```


Demo:
Push to
IBM API Connect

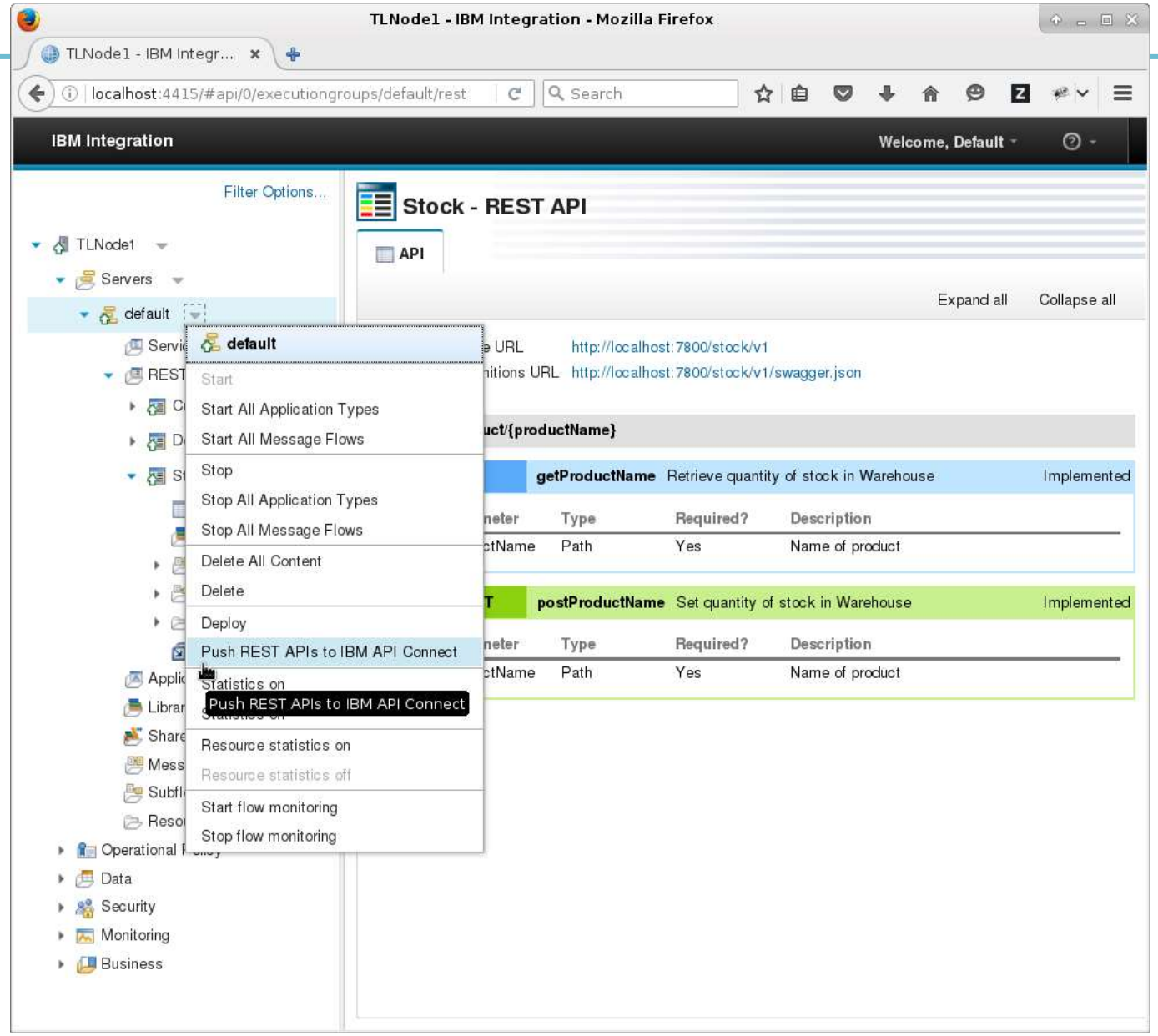
Integration Server contains a number of APIs

Demo – IIB WebUI



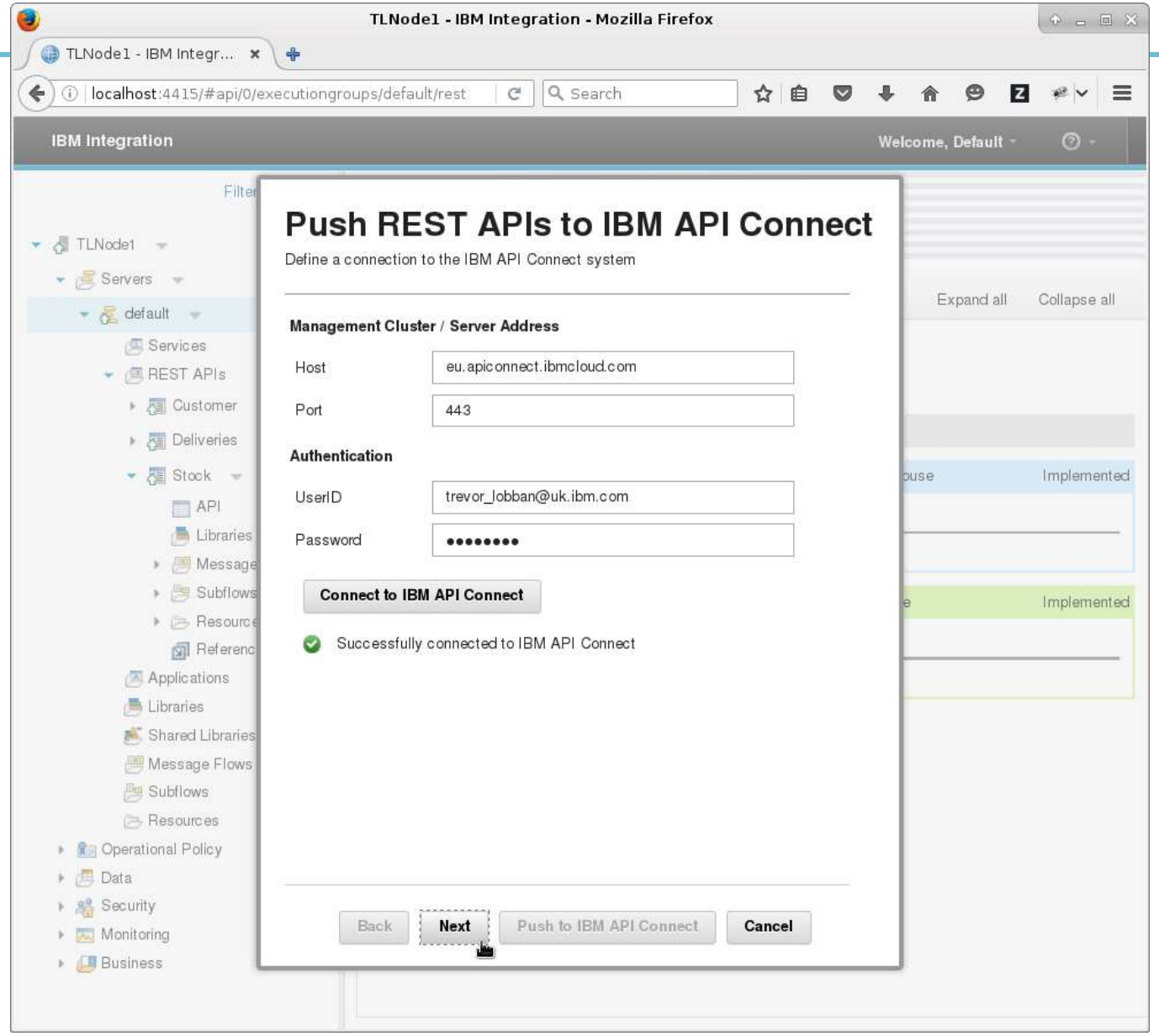
Select “Push REST APIs to IBM API Connect”

Demo – IIB WebUI



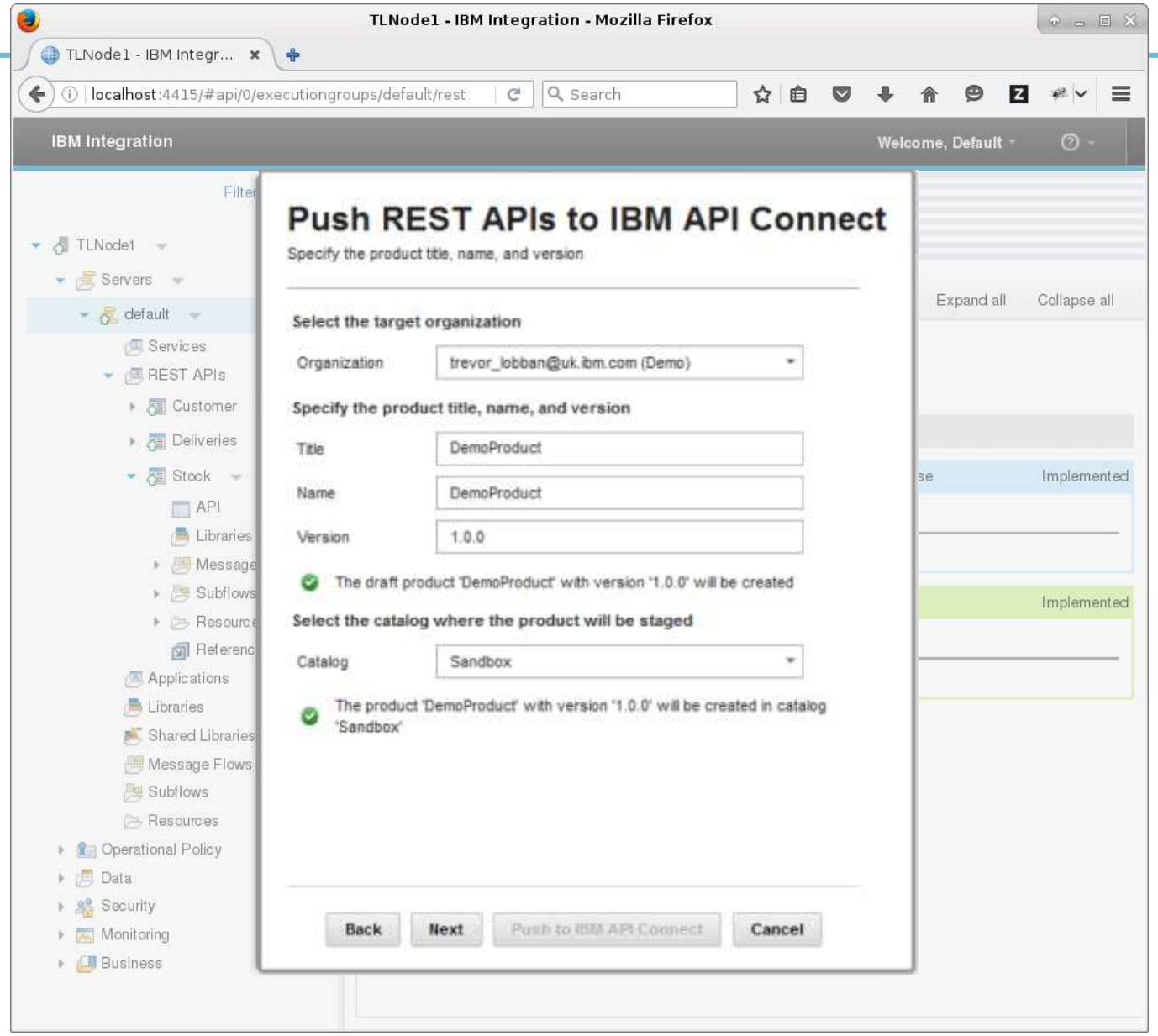
Enter IBM API Connect credentials

Demo – IIB WebUI



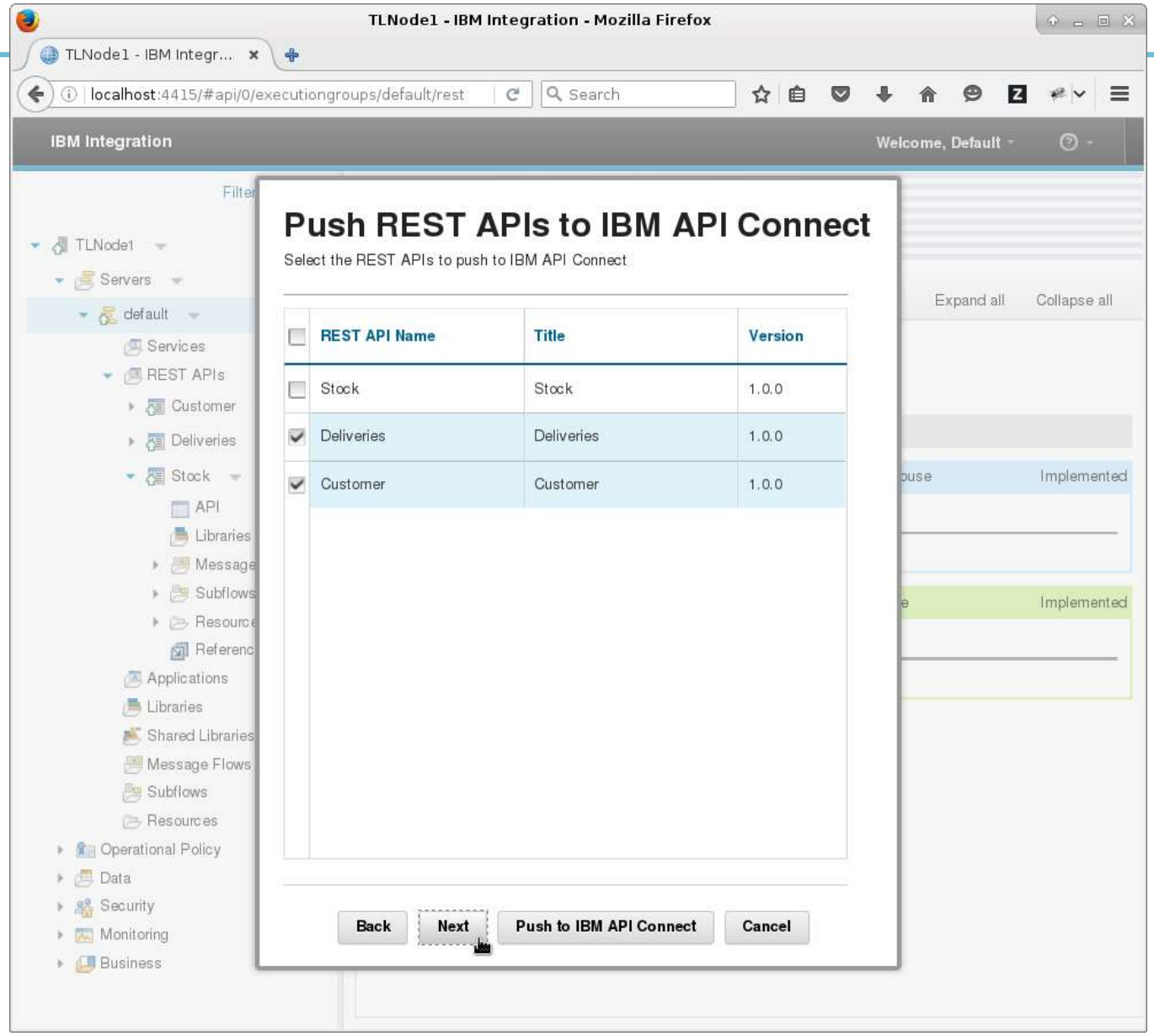
Enter IBM API Connect credentials

Demo – IIB WebUI



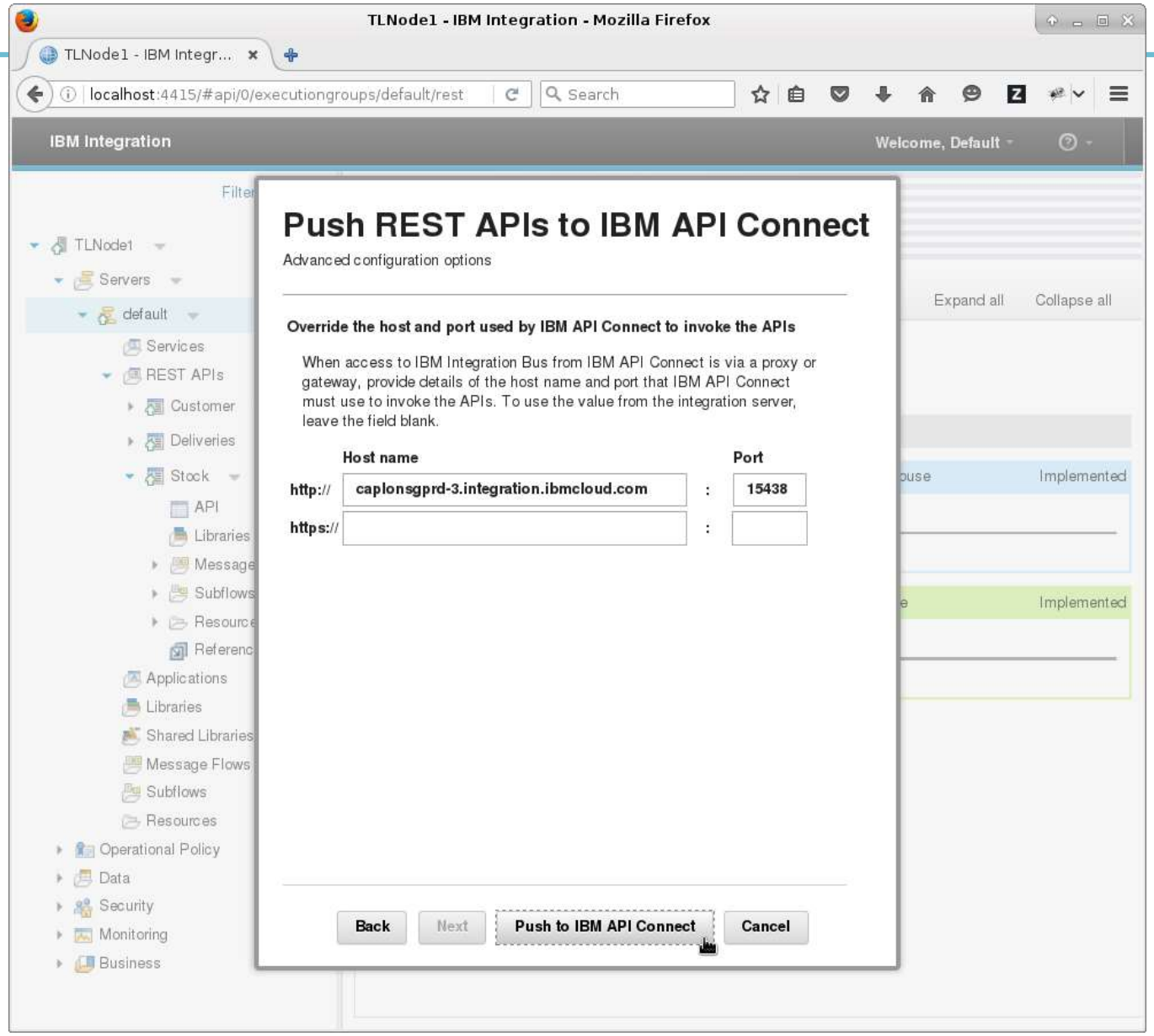
Select APIs to push

Demo – IIB WebUI



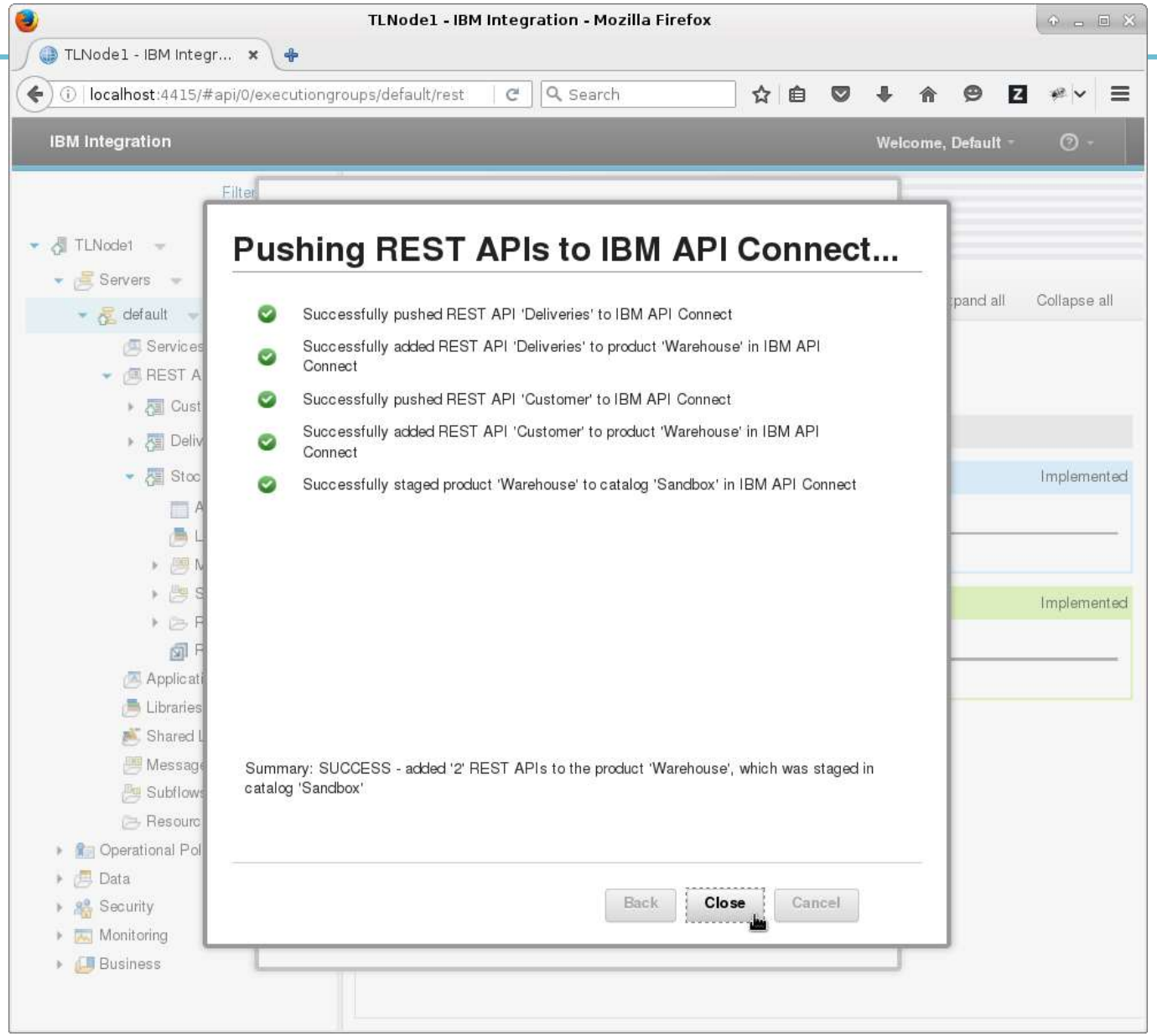
Optionally provide Gateway details

Demo – IIB WebUI



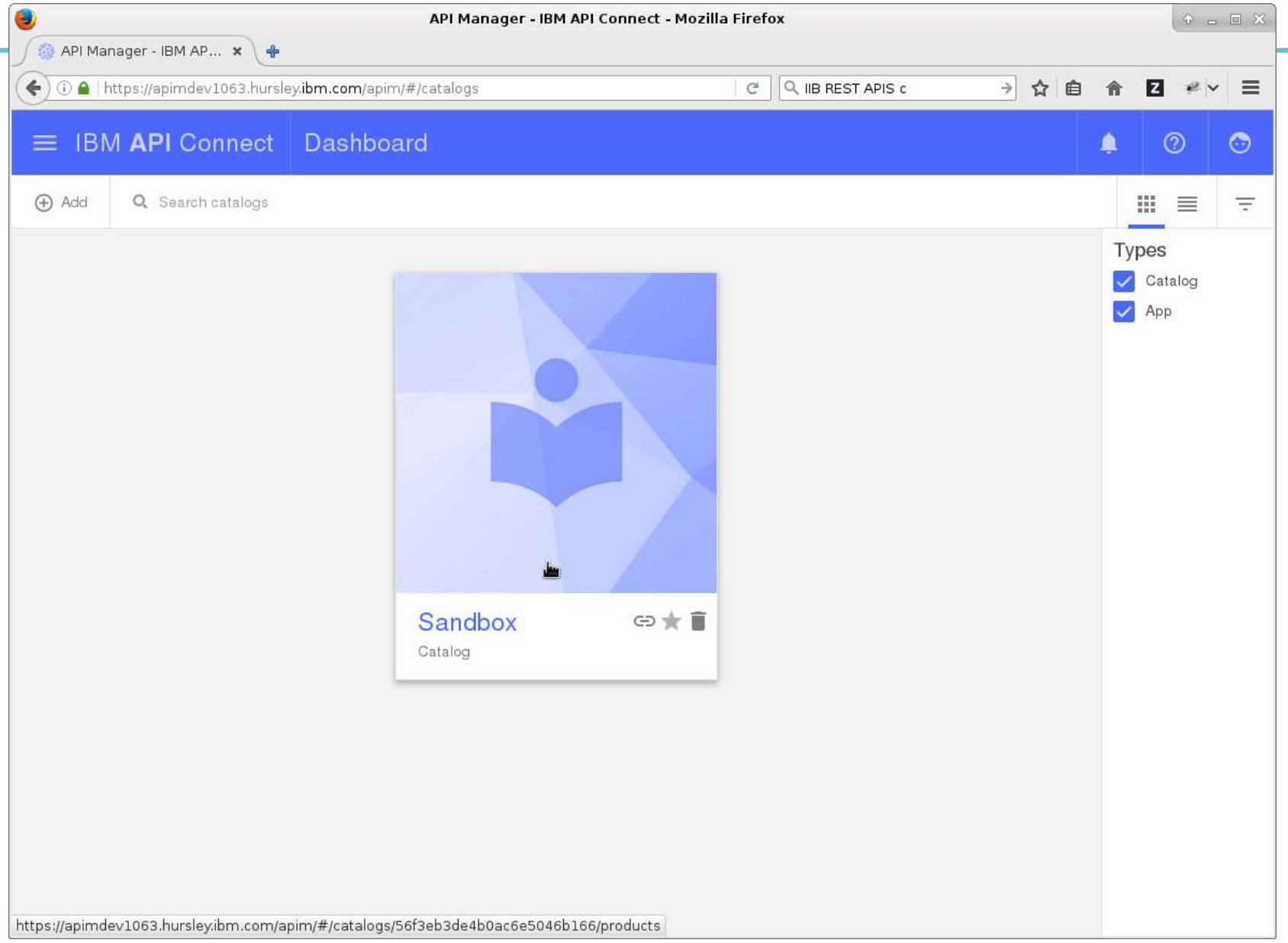
Push definitions to IBM API Connect

Demo – IIB WebUI



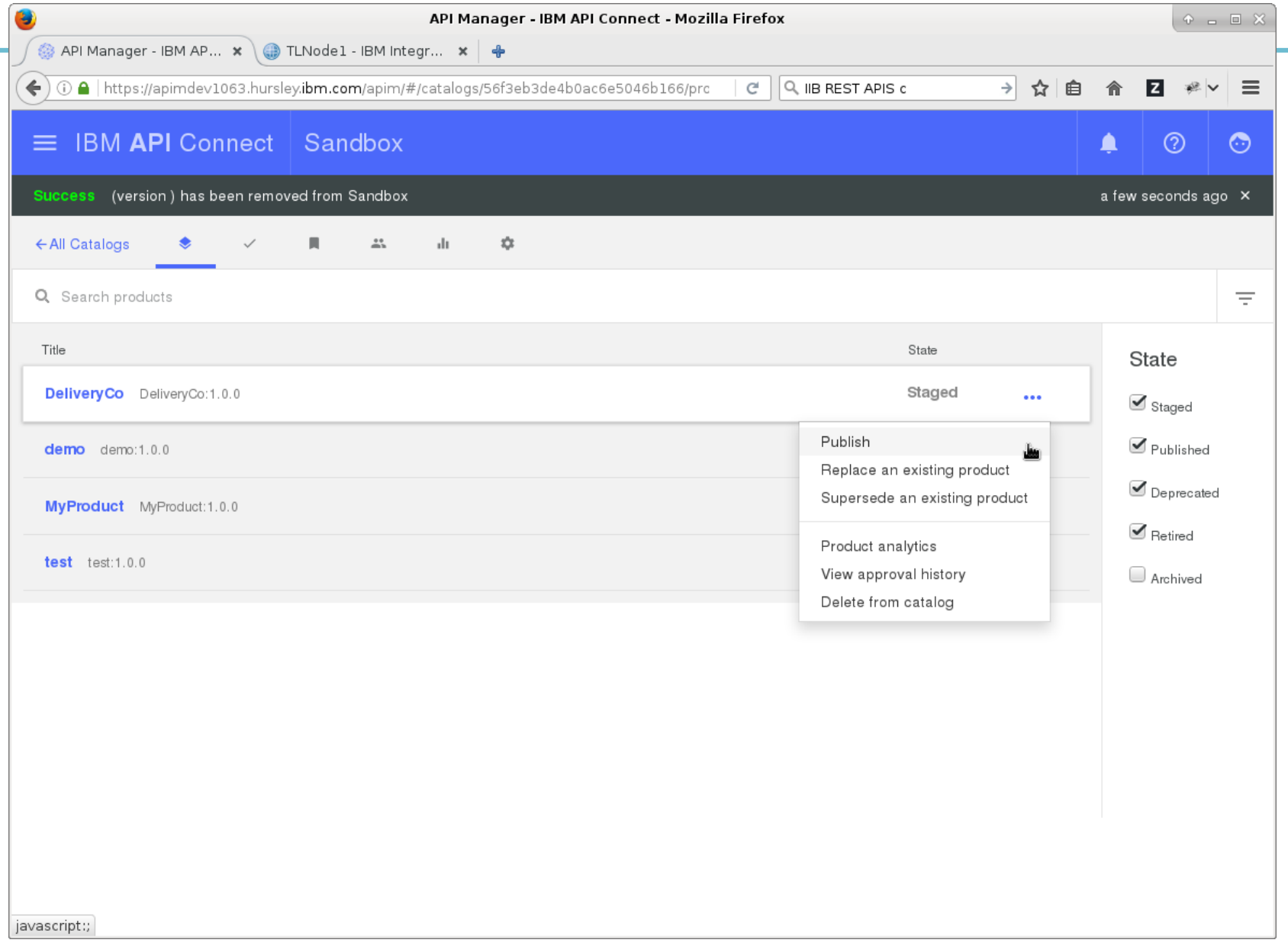
Push definitions to IBM API Connect

Demo – API Manager



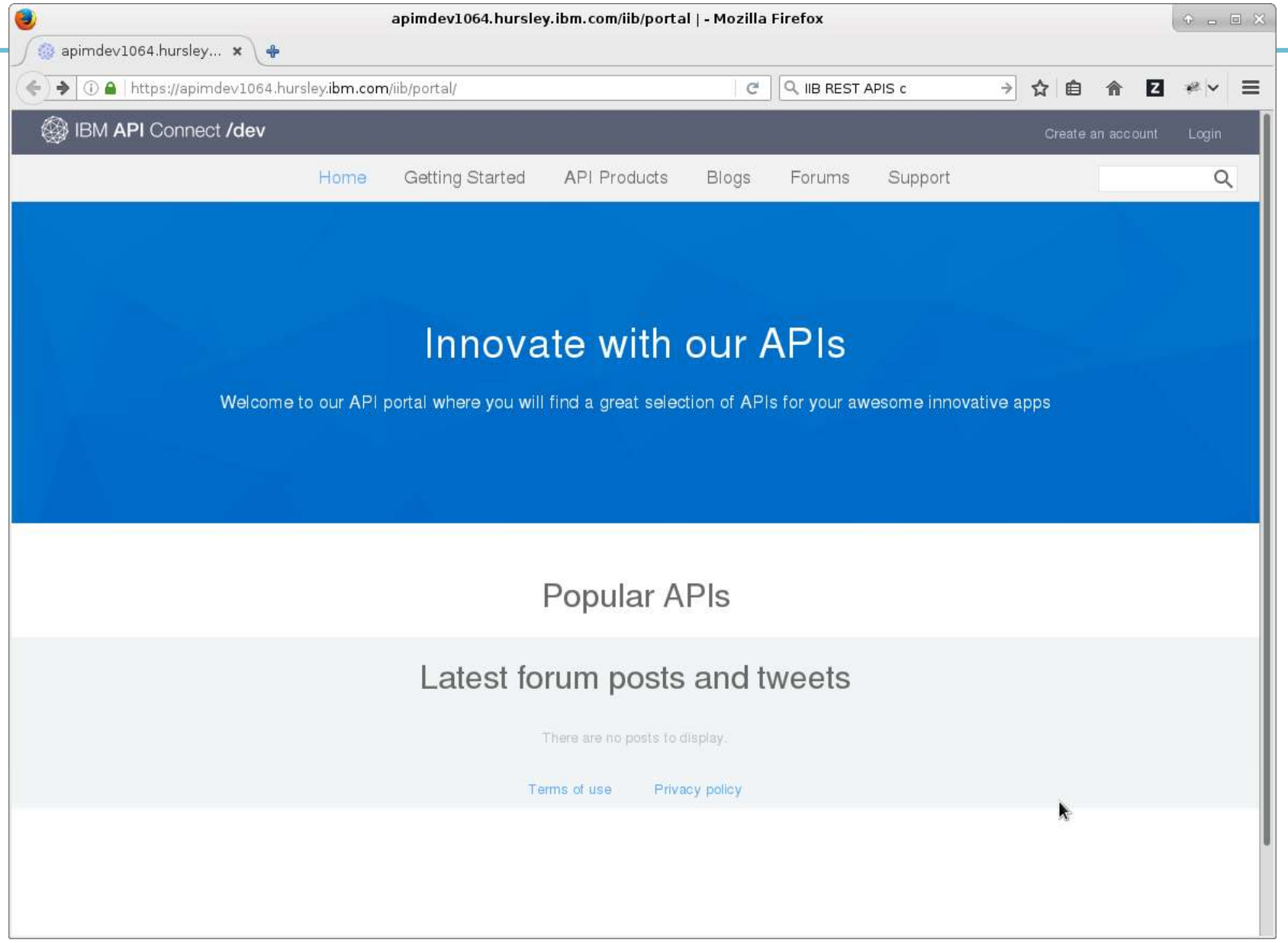
Push definitions to IBM API Connect

Demo – Publish API



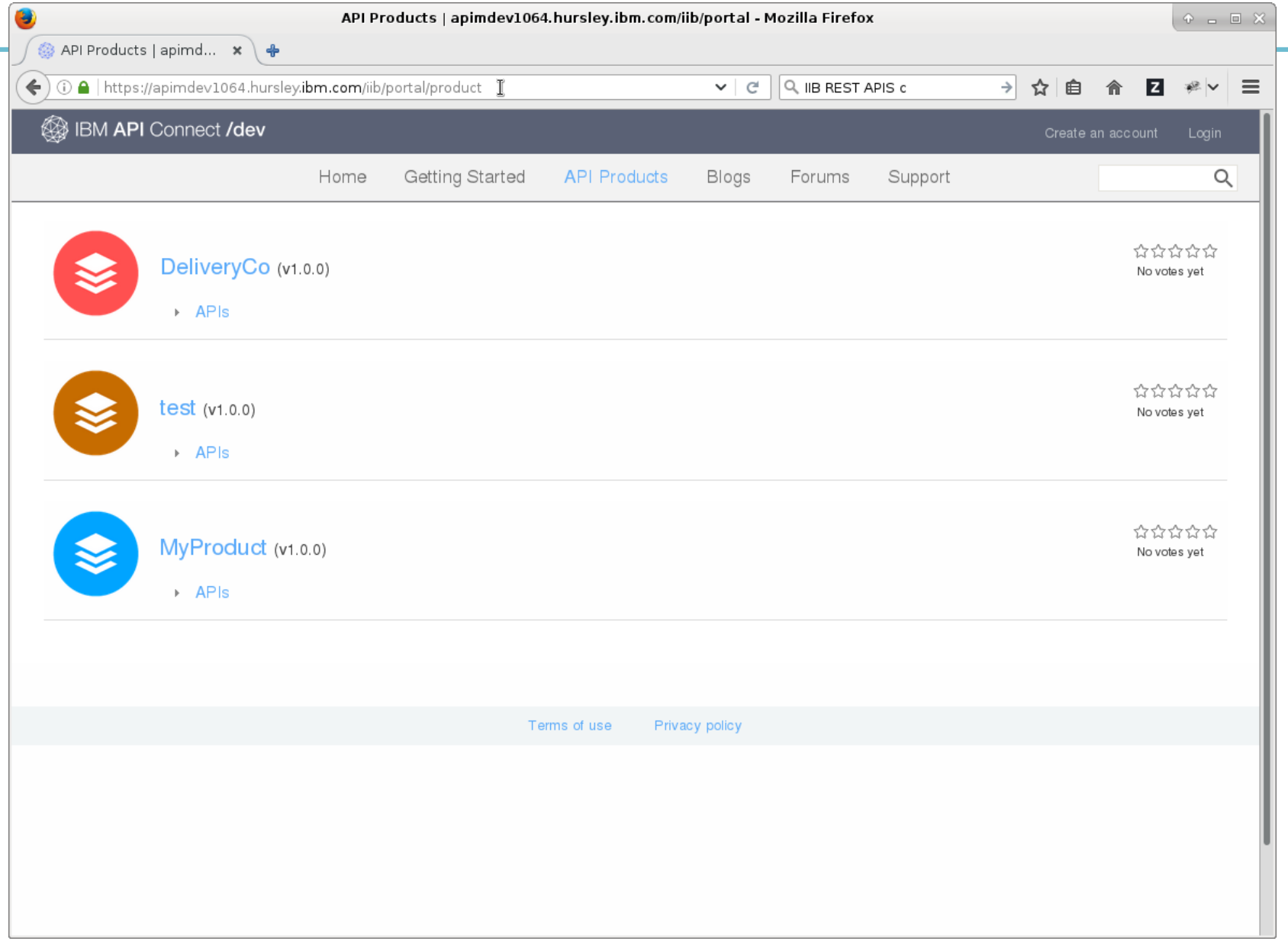
Push definitions to IBM API Connect

Demo – Developer Portal



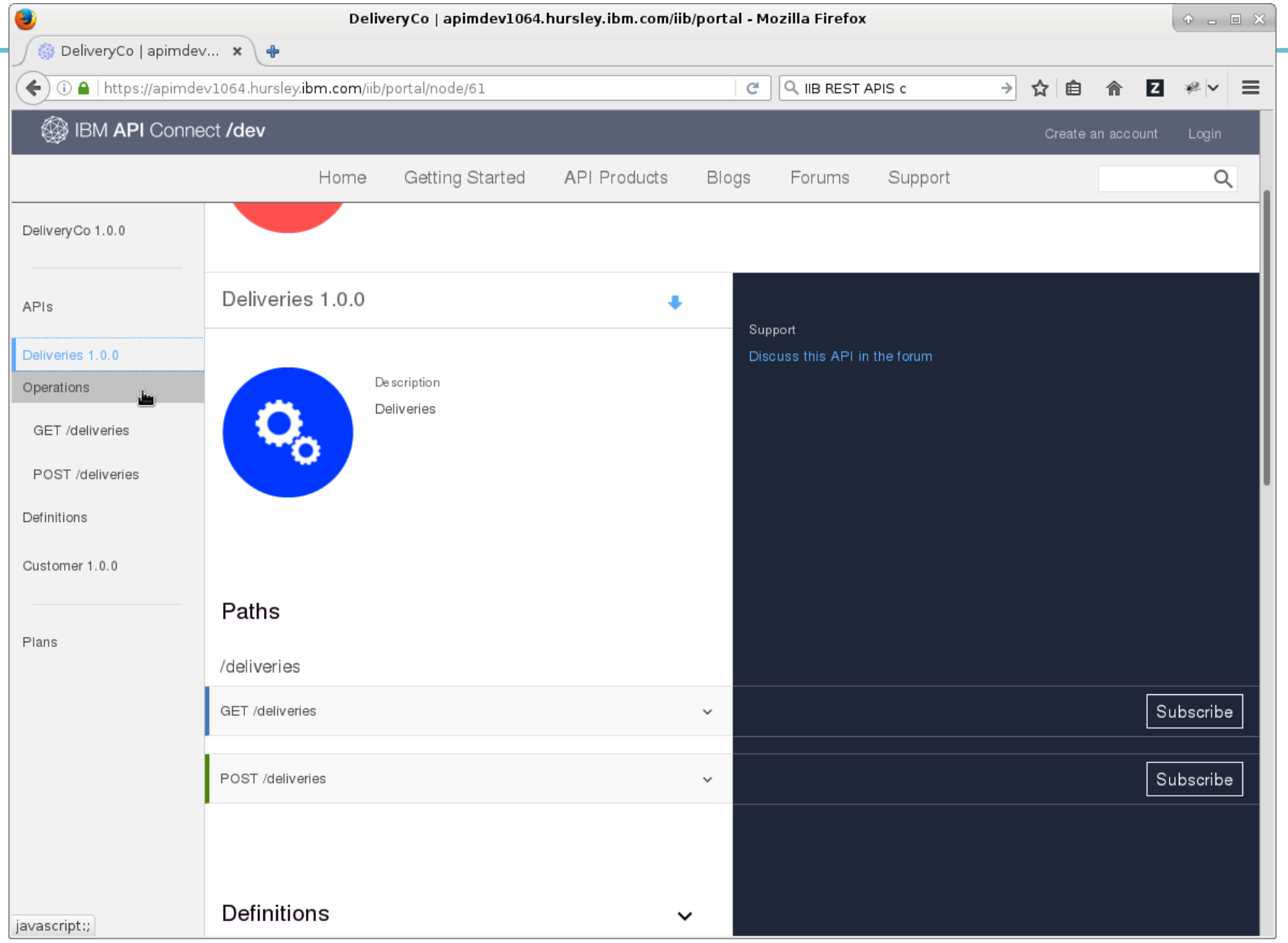
Push definitions to IBM API Connect

Demo – Developer Portal



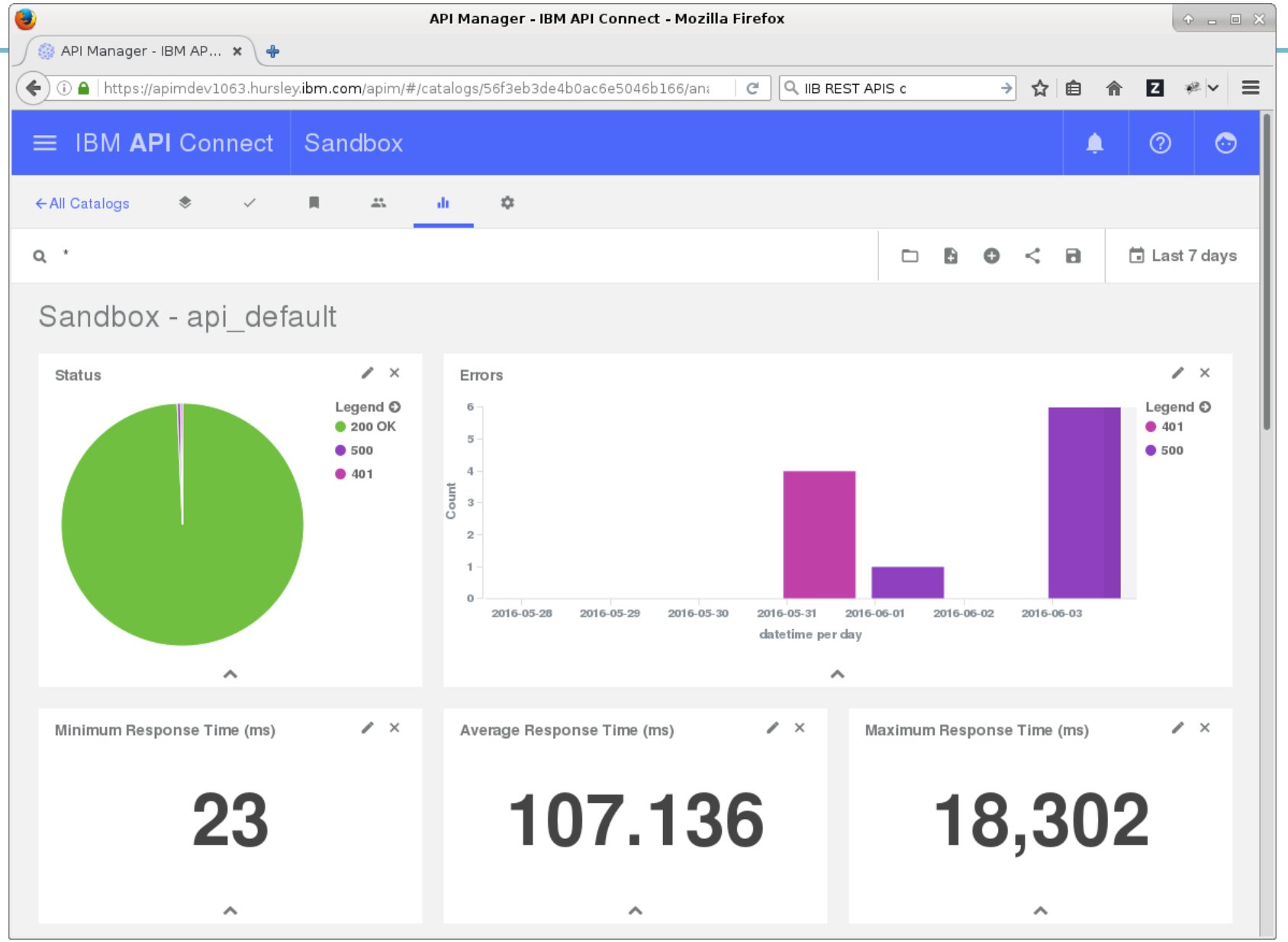
Push definitions to IBM API Connect

Demo – Developer Portal



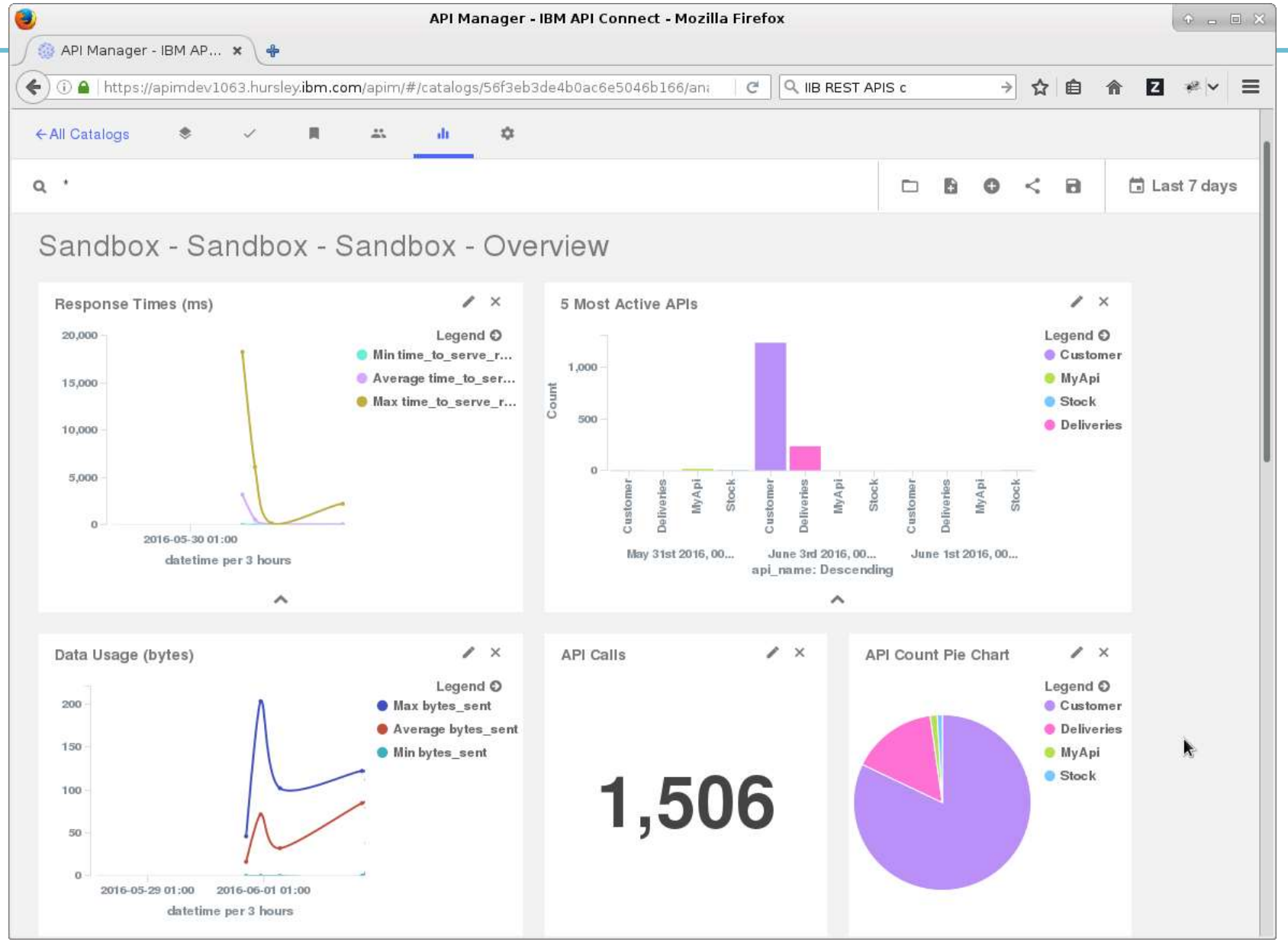
Push definitions to IBM API Connect

Demo - Analytics



Push definitions to IBM API Connect

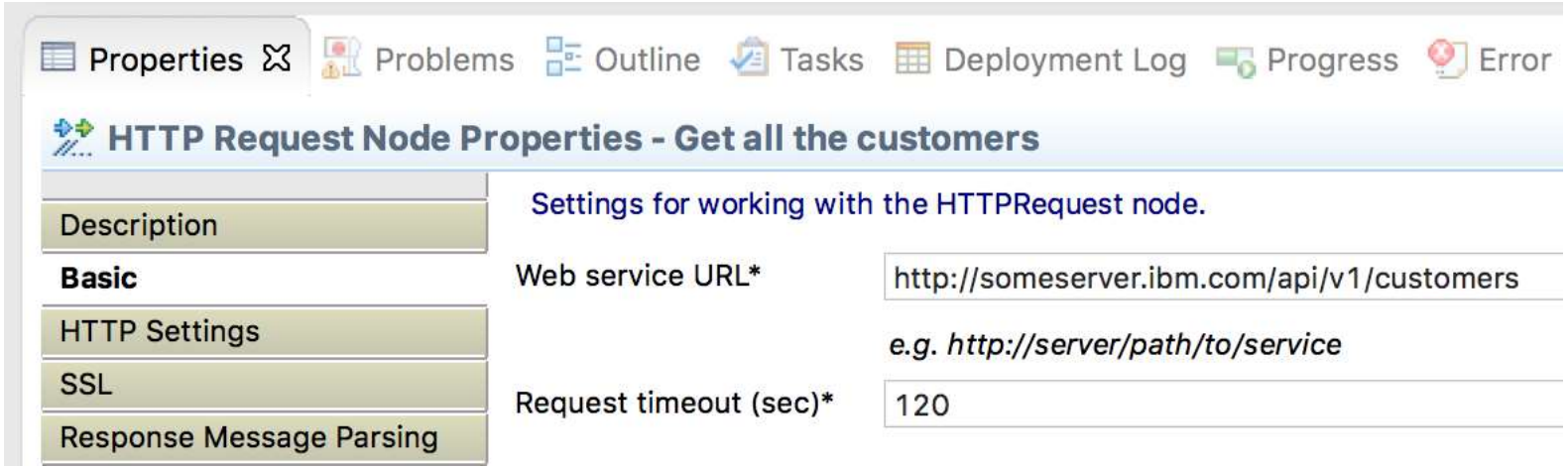
Demo - Analytics



REST Request Node

Calling REST APIs using HTTP Request nodes

- You can call a REST API today in IIB using HTTP Request node.
  - It's only standard HTTP under the covers!
- This is pretty simple for static URLs:

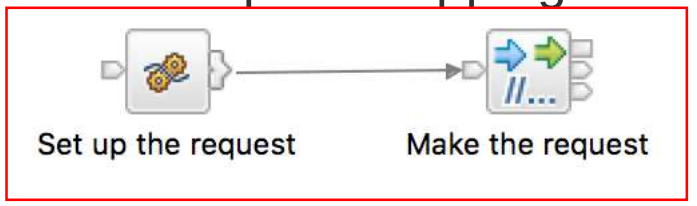


Calling REST APIs using HTTP Request nodes

- But for dynamic URLs (for example, those with path parameters), it's a bit more involved and usually requires custom code:

```
SET OutputLocalEnvironment.Destination.HTTP.RequestLine.Method = 'DELETE';
SET OutputLocalEnvironment.Destination.HTTP.RequestURL =
    'http://someserver.ibm.com/api/v1/customers/' ||
    customerId ||
    '/orders/' ||
    orderId ||
    '/items/' ||
    itemId;
SET OutputLocalEnvironment.Destination.HTTP.QueryString.forceDelete = TRUE;
SET OutputLocalEnvironment.Destination.HTTP.QueryString.areYouSure = 'really sure';
SET OutputLocalEnvironment.Destination.HTTP.QueryString.lastChance = 'do it';
```

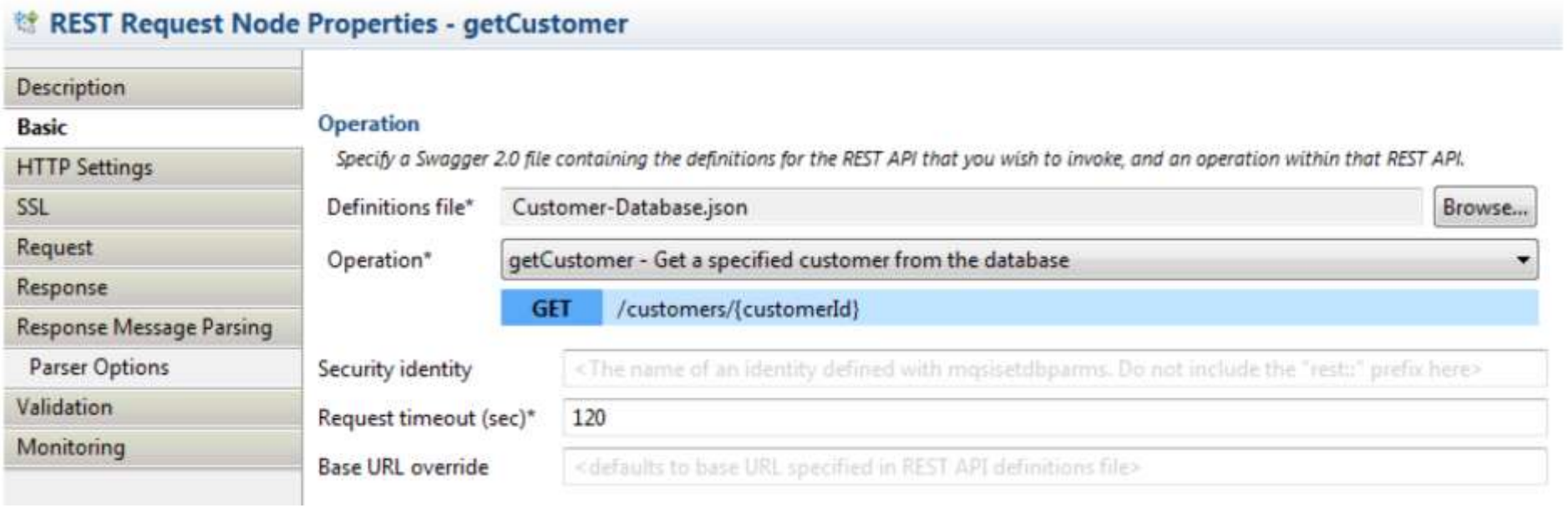
- Most of the time, you're going to **need** a Compute/Mapping node before the HTTP Request node:



Introducing the REST Request node

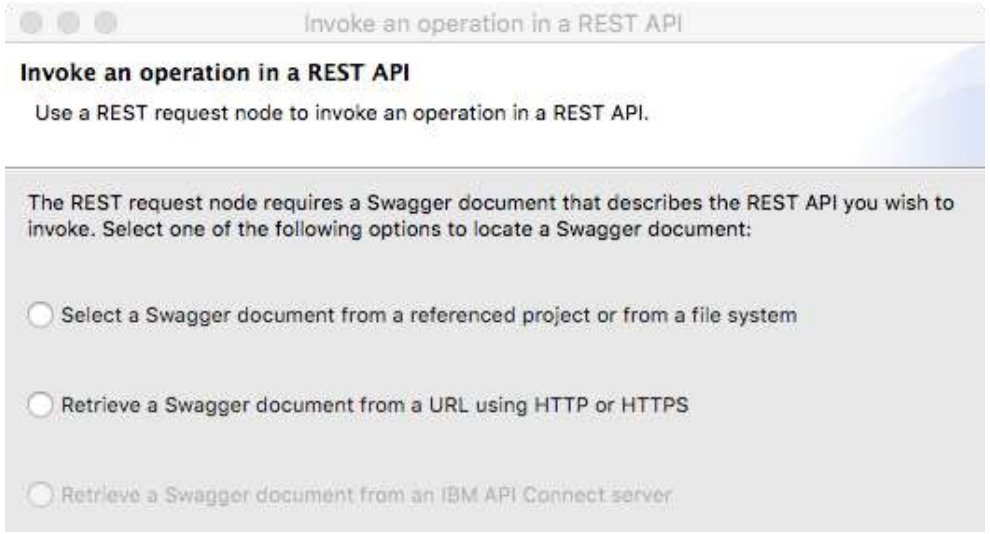
- We have provided a new REST Request node to make it much simpler for an IIB flow developer to make a call to a REST API.

REST Request



REST Request node and Swagger

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

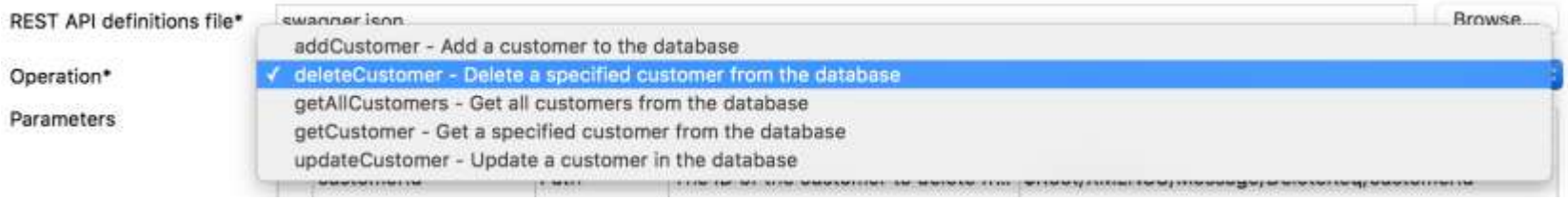


- Swagger documents also come in two formats (JSON and YAML) - we now support YAML format since 10.0.0.6

REST Request node configuration

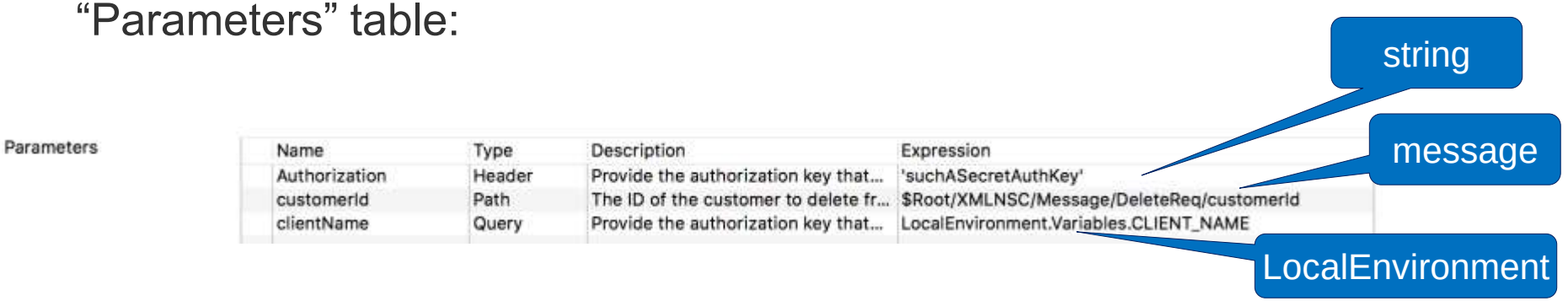
- As before, you supply the REST Request node with a Swagger file:

- 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:



- 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
  - Request/response message tree locations
  - Parser choice and parser options
  - Validation
  - Monitoring

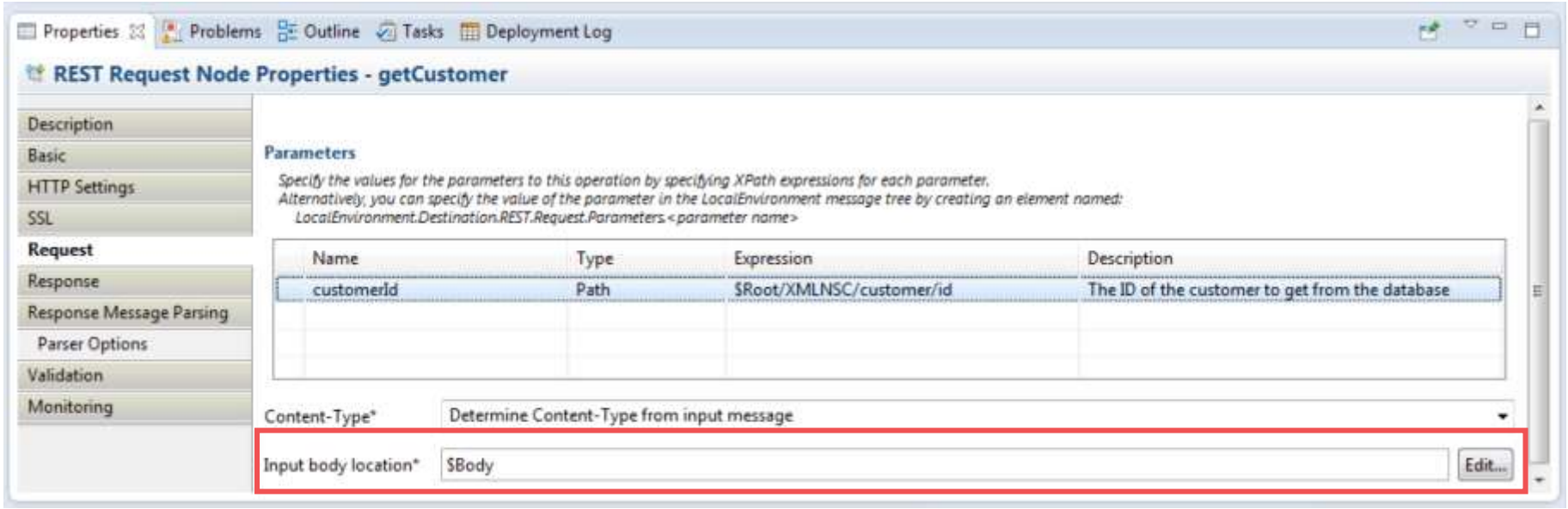
REST Request node request body

- The request body will be sent from the input message as per any other request node.
- Can modify the request body location – for example, to send a JSON request that has been placed into the Environment tree.

Input body location* $Body

Or, for example, you can specify a location in the Environment tree.

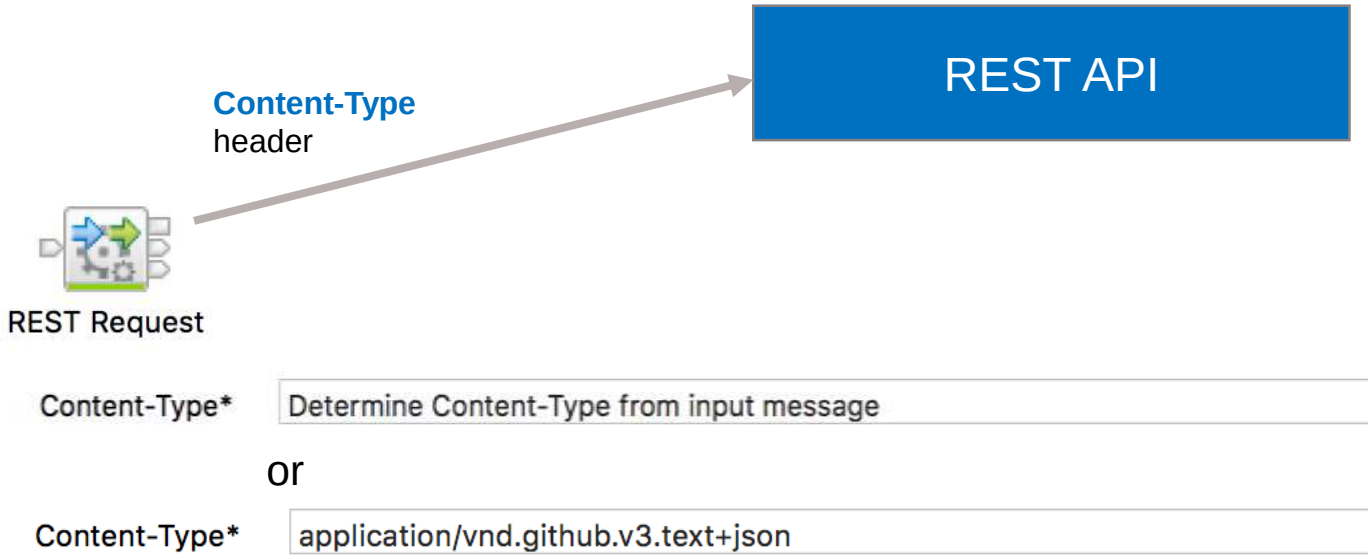
Input body location* $Environment/Variables/Request1

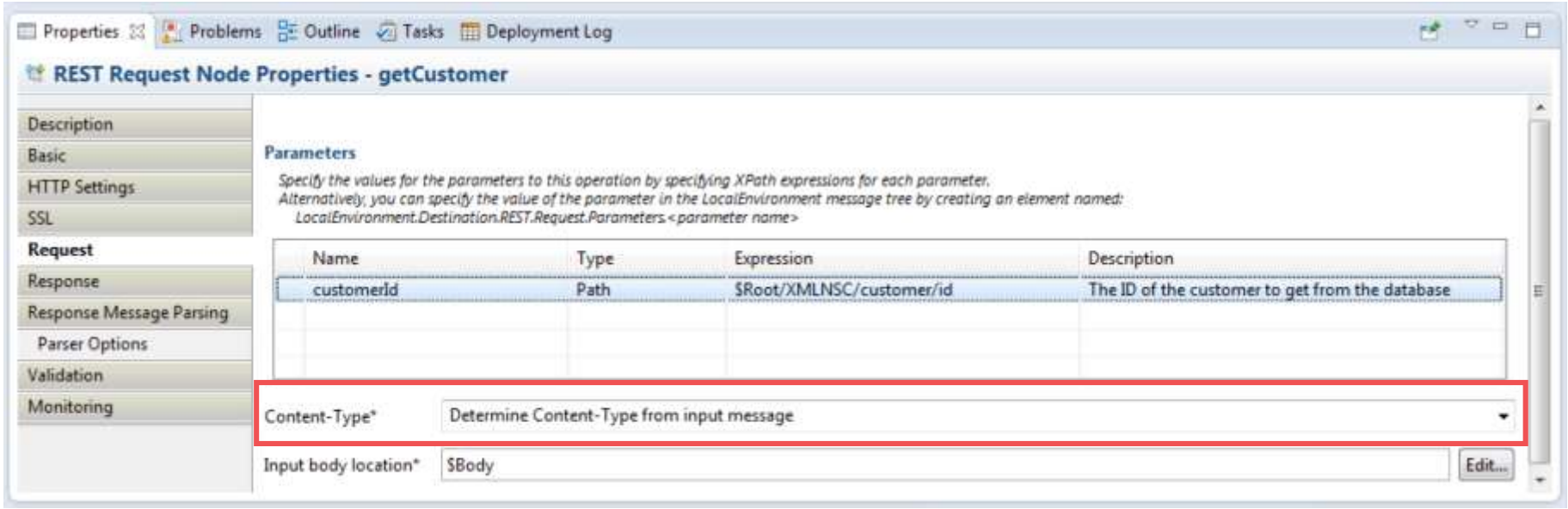


REST Request node request headers

- A **Content-Type** header is sent in the request to describe the type of the data in the request body:

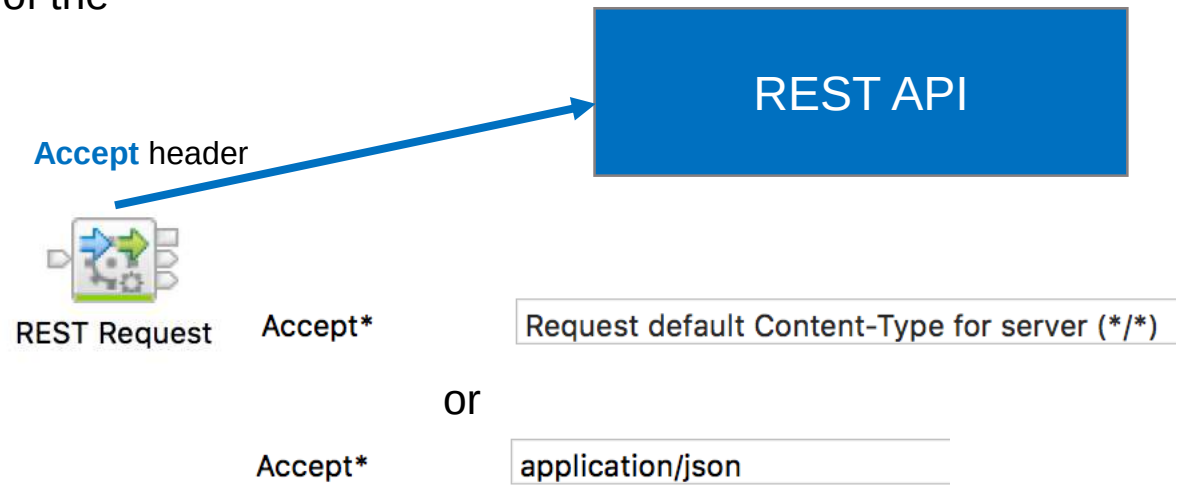
- Defaults to a sensible one for the parser in use (application/json for JSON, application/xml for XML/XMLNS/XMLNSC).
- The Swagger document can specify supported Content-Type values.
- You can select one of these, or type in your own value.

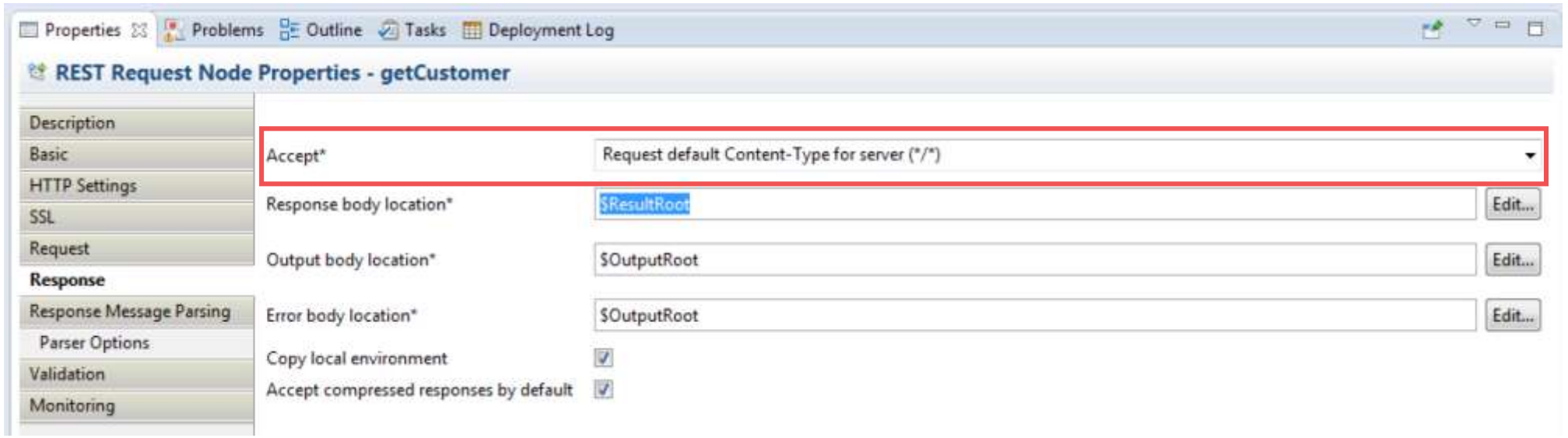




REST Request node request headers

- An **Accept** header is sent in the request to describe the type of the data that we want in the response body from the server.
- Defaults to */* which is “let the server decide the default”.
- The Swagger document can specify supported Accept values.
  - The same operation might return either XML or JSON.
  - You specify which one you want by setting the Accept header.





REST Request node response body

- By default the whole of the response body is placed into the output message at the specified location.
- However you can also select a specific part of the response body to place into the location specified in the output message.
- You can use this if your REST API returns a large amount of data, but you only want a specific section of that data.

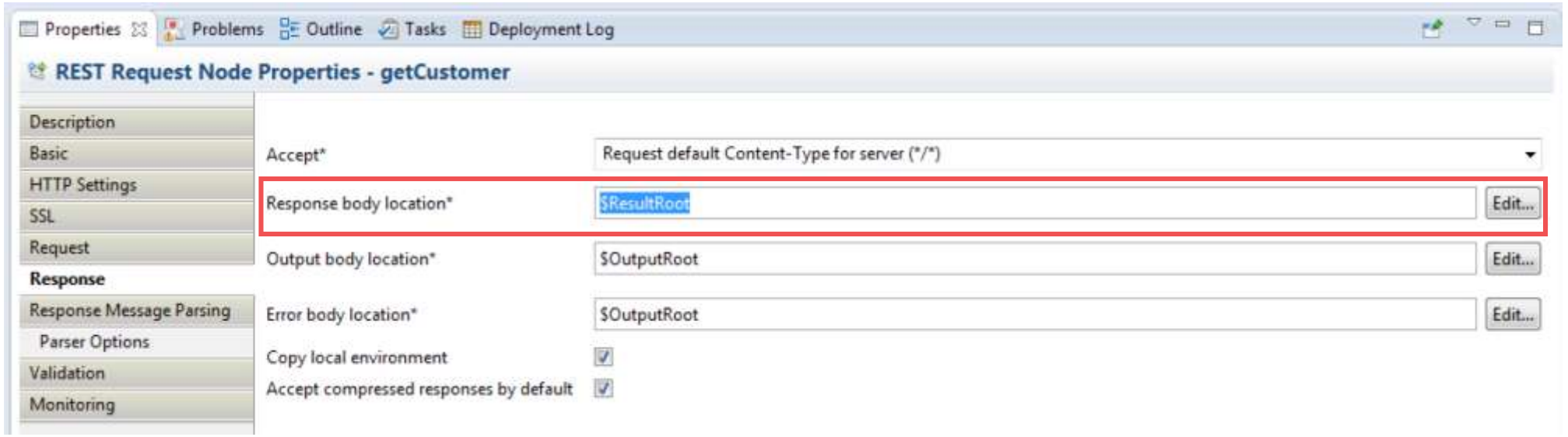
The entire response from the REST API is used.

Response body location* $ResultRoot

or

Response body location* $ResultRoot/JSON/Data/Customers/Item[2]/id

Only part of the response from the REST API is used.



REST Request node output body

- By default, the node will replace the input message with an output message containing the response body.
- Can modify the output body location – for example, to place a JSON response in the Local Environment tree. You can use this to combine the input message with the output message.

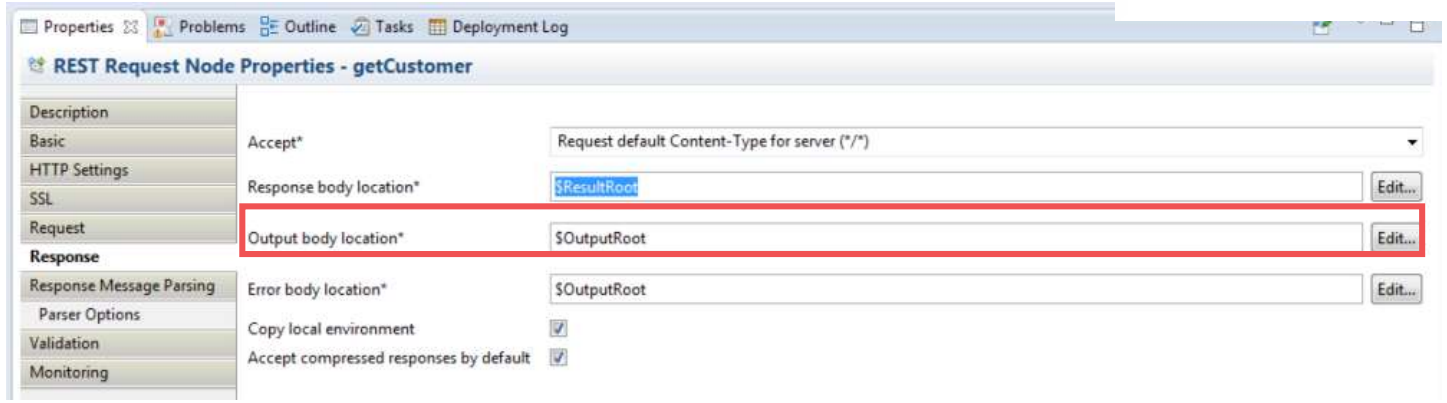
Response from the REST API is placed in the output message that is propagated from the REST Request node.

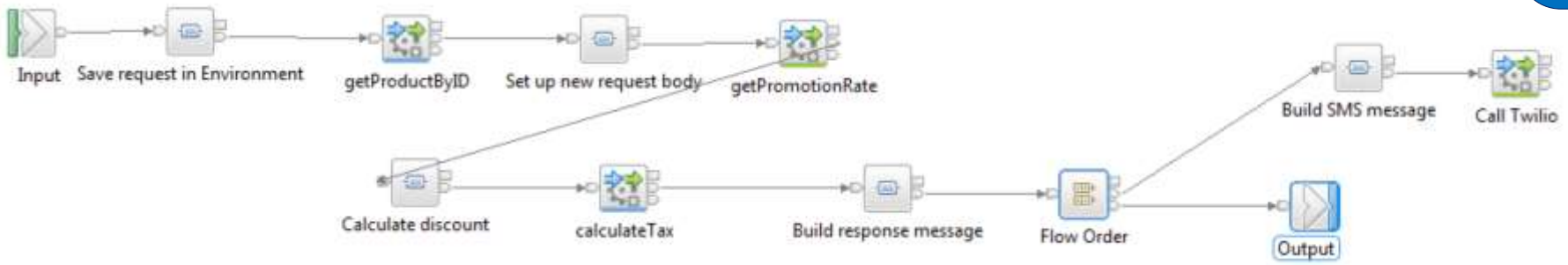
Output body location* $OutputRoot

or

Output body location* $OutputLocalEnvironment/Variables/Response1

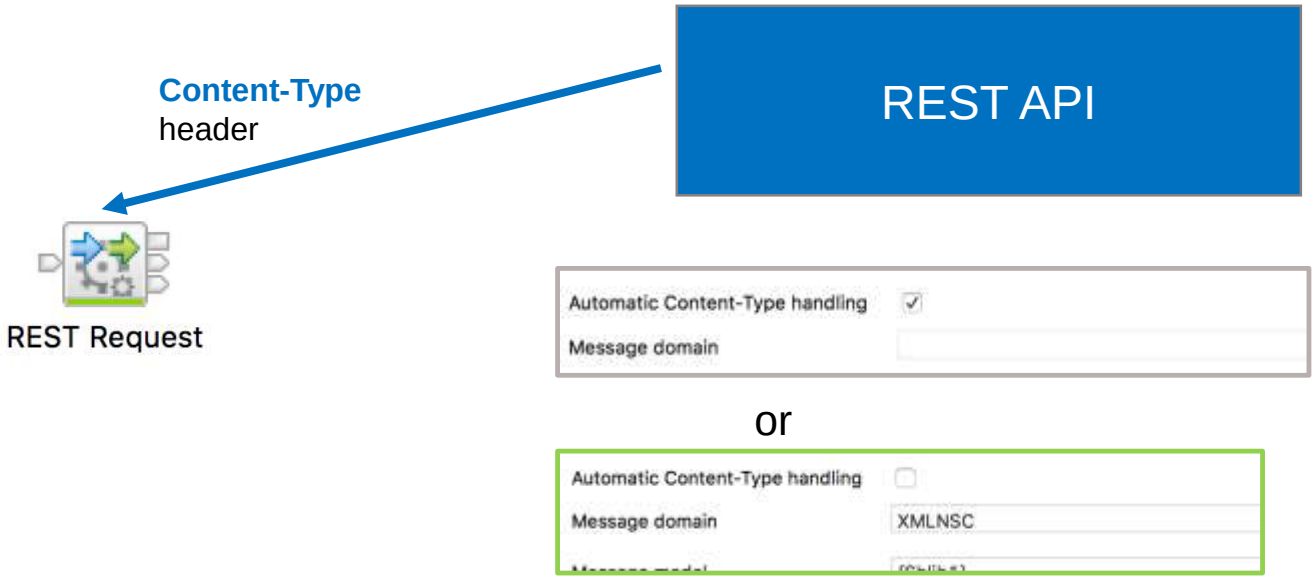
Response from the REST API is placed in the Local Environment so that it can be accessed later on and combined with other data.

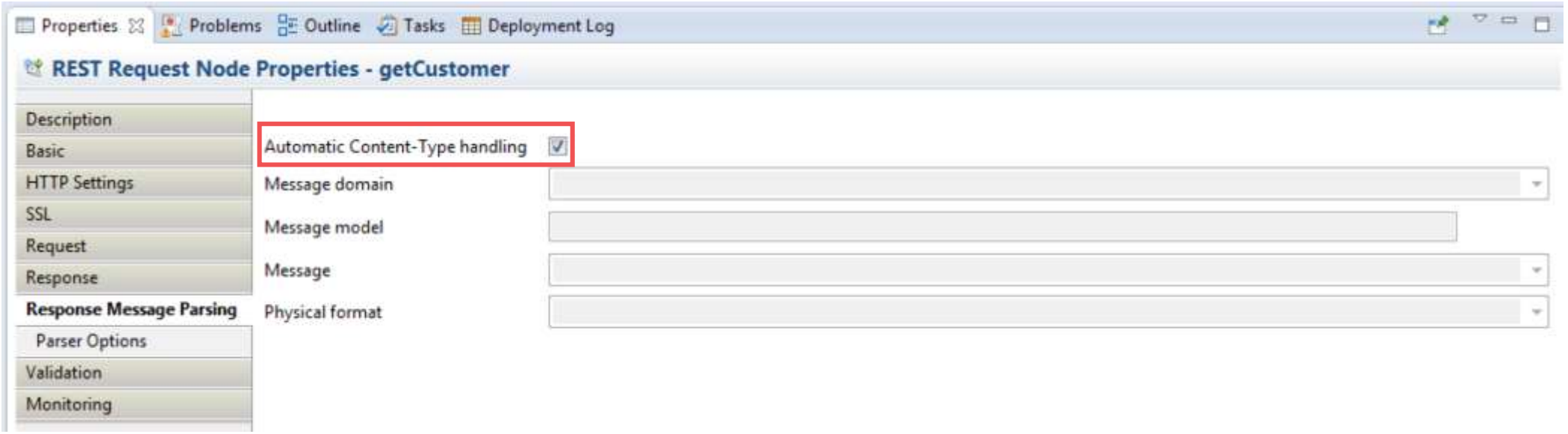




REST Request node response headers

- The response from the REST API should contain a Content-Type header that describes the type of the data in the response body.
- By default, the node will choose the message domain (parser) based on the value of the Content-Type header in the response:
  - For application/json, it will pick JSON
  - For application/xml or text/xml, it will pick XMLNSC
  - Anything not recognized by IIB, it will pick BLOB
- You can disable this functionality and manually specify the parser.

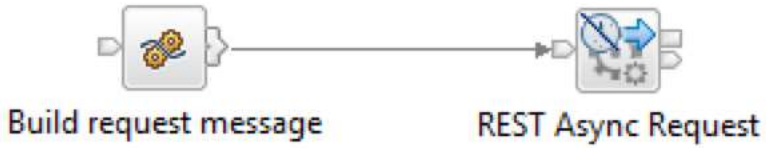


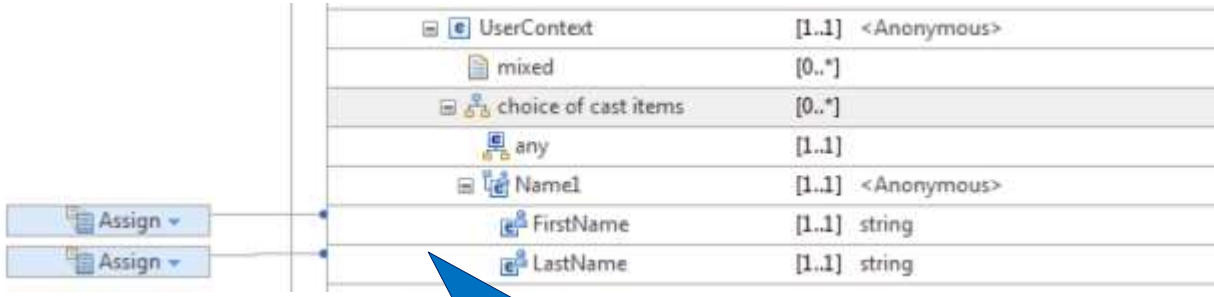


REST Request node **and friends!**

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

Thread 1





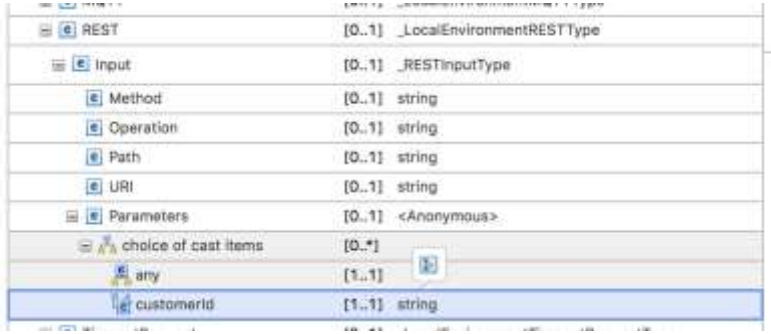
Thread 2

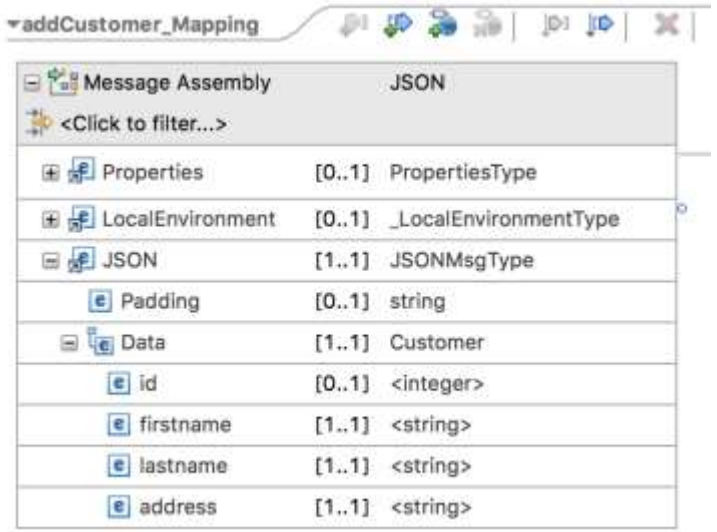
Share User Context between REST Async Request/Response nodes

- Request and response processing is handled as separate transactions.

REST Request node and the graphical mapper

- The graphical mapper already supports using parameter information and JSON Schema from Swagger documents as part of our REST API support.
- Functionality enhanced so that the mapper can provide assistance for mapping request/response data for the REST Request node:





- JSON types from both JSON schema and REST Swagger documents are available for performing mapping casts, enabling you to work with REST request nodes that place data in the environment tree.

New allOf, oneOf, anyOf support in graphical mapper

- IIBv10.0.0.8 adds support for three further JSON constructs: allOf, anyOf and oneOf. These combination keywords facilitate the reuse of defined structures in your JSON schema.

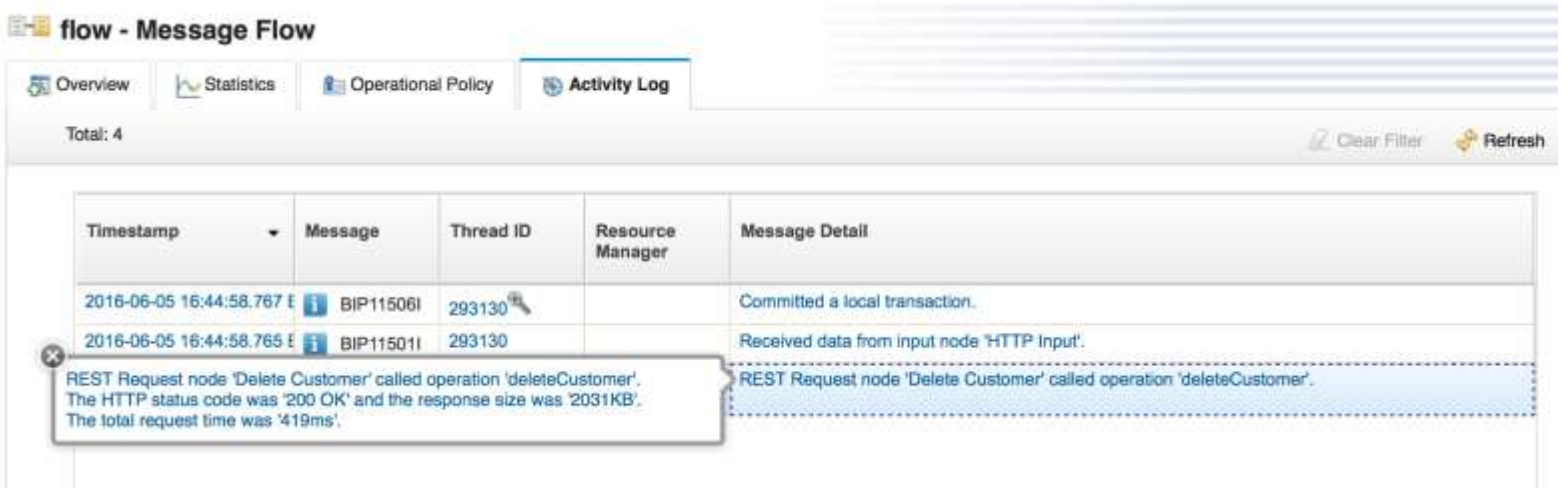
- The allOf keyword lets you extend a structure with additional fields.
- The oneOf keyword lets you specify a structure which is valid if it matches *exactly* one of the defined options.
- For example, consider the example JSON schema definition below which has an array of vehicles, each of which is either a car or a bike.
- The anyOf keyword is similar to oneOf, but specifies that one *or more* of the contained structures must validate against the instance value.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {
    "car": {
      "type": "object",
      "properties": {
        "transmissionType": {"type":"string"},
        "engineSizeCC": {"type":"string"}
      },
      "required": ["transmissionType", "engineSizeC
    },
    "bike": {
        "type": "object",
        "properties": {
          "style": {"type":"string"},
          "numberOfGears": {"type":"number"}
        },
        "required": ["style", "numberOfGears"]
    },
    "vehicle": {
        "oneOf": [
                  { "$ref": "#/definitions/car" },
                  { "$ref": "#/definitions/bike" }
                ]
    },
    "vehicleList": {
        "type": "array",
        "items": {"$ref": "#/definitions/vehicle"}
    }
  }
}
```

Message Assembly		JSON
<Click to filter...>		
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType
Padding	[0..1]	string
Data	[1..1]	JSONArray_vehicleList
choice of cast items	[0..*]	
Item	[1..1]	vehicle
Item	[1..1]	vehicle_bike
style	[1..1]	<string>
numberOfGears	[1..1]	<double>
Item	[1..1]	vehicle_car
transmissionType	[1..1]	<string>
engineSizeCC	[1..1]	<string>

Activity logging for the REST Request node

- REST Request node will have activity logging.
- All requests will be logged to the activity log for the message flow, along with the HTTP status code, response size, and total request time.
- Visible through command line, web UI, and Integration API.

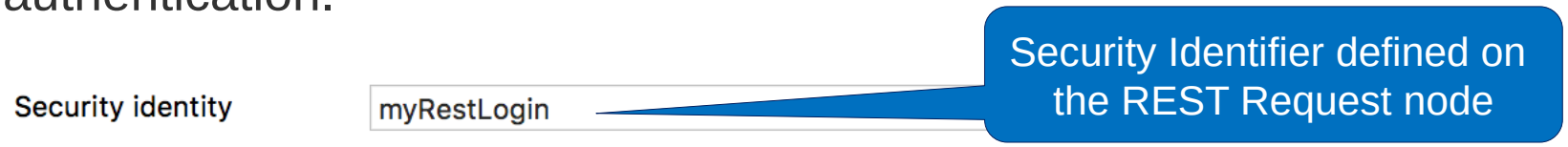


Outbound security for the REST Request node

- A lot of REST APIs are going to be secured:
  - HTTP Basic Authentication (username + password)
  - API authorization key passed as header or query parameter
  - OAuth2
  - SSL/TLS client or mutual authentication
- HTTP Basic Authentication is not currently a simple experience for the IIB flow developer when using an HTTP Request node:
  - Place user ID/password in Properties tree
  - And, use default propagation policy set (which is not on by default!)
  - Or, create your own Authorization header

Outbound security for the REST Request node

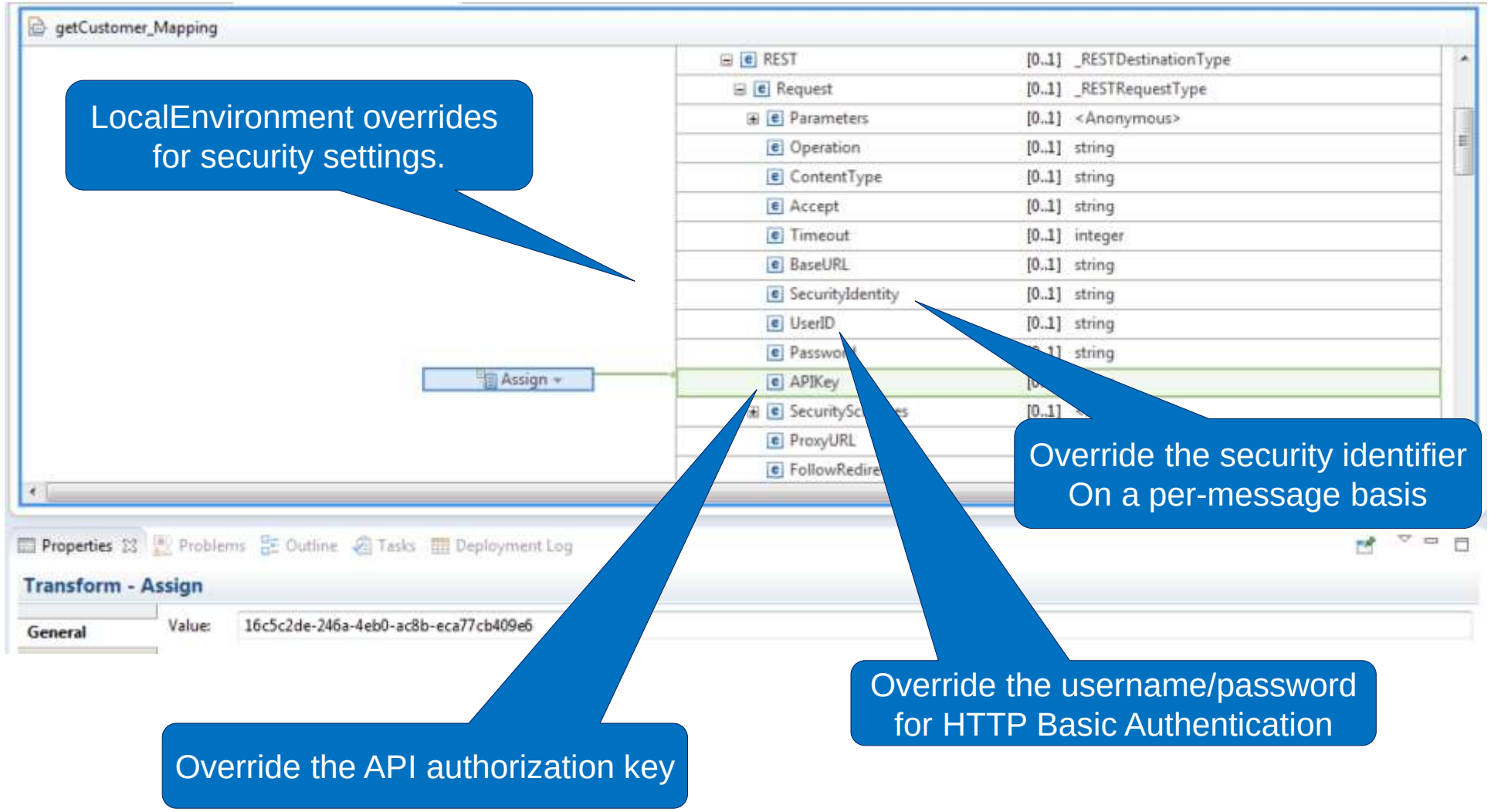
- We are going to make it easier to use the following security mechanisms:
  - HTTP Basic Authentication (username + password)
  - API authorization key passed as header or query parameter
- We **do not** support OAuth2 at the moment.
- There will be a new security identity property on the node.
- You can use mqsisetdbparms to set the username + password and/or API authorization key and the node will take care of the authentication.



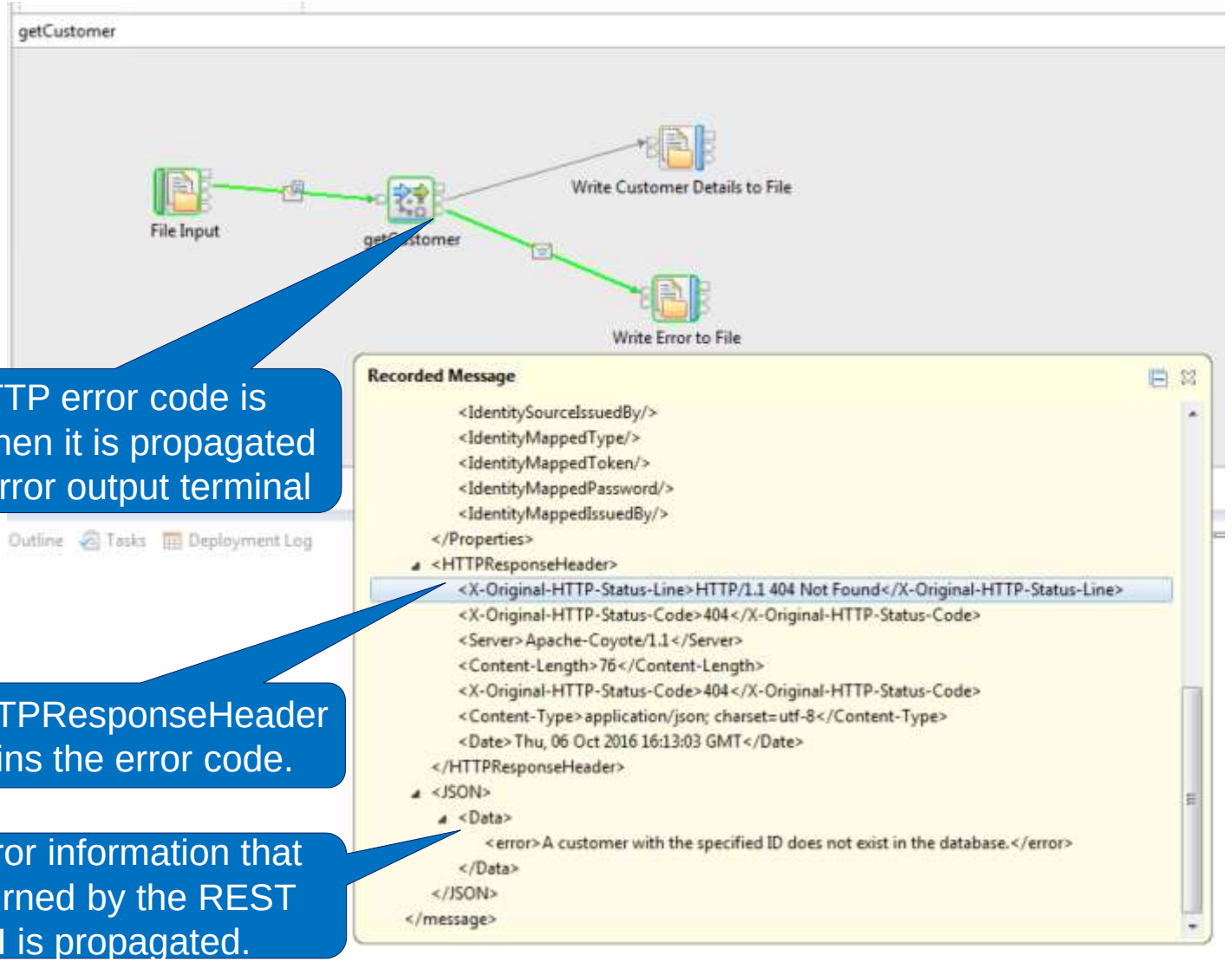

```
$ mqsisetdbparms IB10NODE -n rest::myBasicAuth -u sstone1 -p Passw0rd
$ mqsisetdbparms IB10NODE -n rest::myApiKey -k F3F7F28D-6B99-4720-9E96-3EE038AEEF57
$ mqsisetdbparms IB10NODE -n rest::myBasicAuthAndApiKey -u sstone1 -p Passw0rd -k F3F7F28
```

Security Identifier used by mqsisetdbparms

Outbound security for the REST Request node

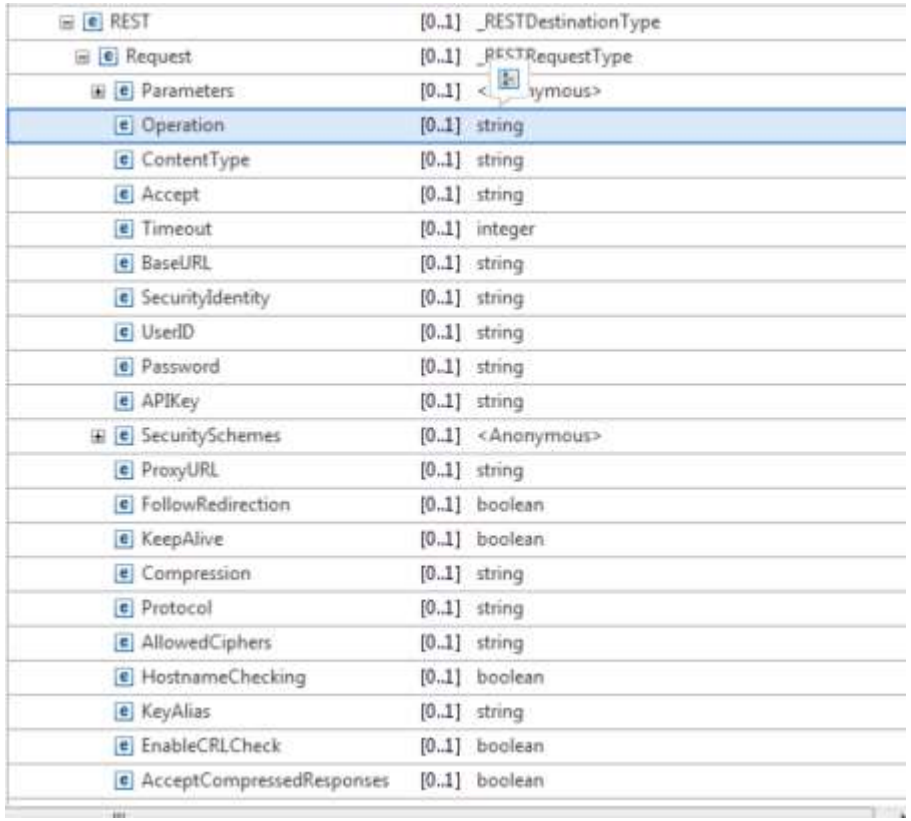


Error handling



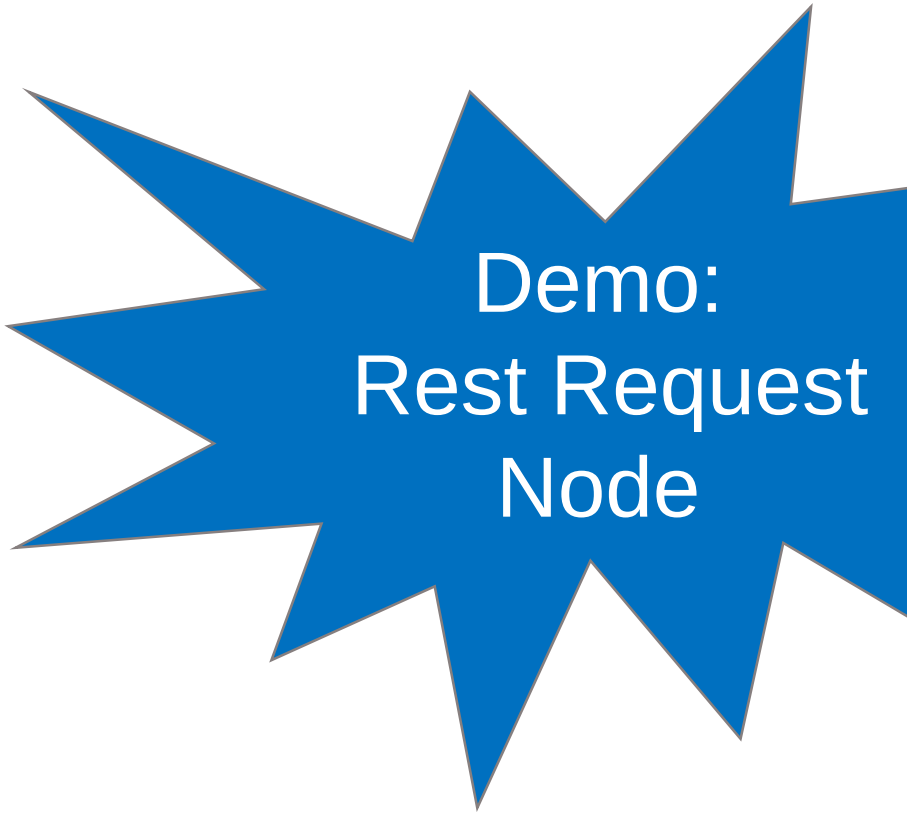
Local Environment

- The REST request nodes support an extensive set of Local Environment overrides that can be used to dynamically change the configuration of the REST request node at runtime.
- These Local Environment overrides include:
  - The operation name.
  - The values for parameters to the operation.
  - The values of the “Content-Type” and “Accept” node properties.
  - The value of the “Security identity” node property.
  - Usernames, passwords, and API keys can also be specified.
  - The value of the “Base URL” node property.



Local Environment - WrittenDestination

- The REST request nodes write a set of properties to the Local Environment output by the node.
- These output properties include:
  - The operation name, HTTP method, and URL used to call the REST API.
  - The size of the HTTP request headers and body sent to the REST API.
  - The size of the HTTP response headers and body received from the REST API.
  - The HTTP status code from the REST API.
  - The total time spent waiting for the response from the REST API.


Demo:
Rest Request
Node

Links

Articles:

IIB V10.0.0.5 Push to API Connect
https://developer.ibm.com/integration/blog/2016/05/31/expose-your-integrations-to-your-organization-as-rest-apis-using-ibm-integration-bus-and-ibm-api-connect/

IIB V10.0.0.5 Push to API Connect using Secure Gateway
https://developer.ibm.com/integration/blog/2016/06/06/pushing-rest-apis-to-ibm-api-connect-provisioned-on-bluemix-and-accessing-them-through-a-secure-gateway-service/

10.0.0.6 REST Request :
https://developer.ibm.com/integration/blog/2016/09/27/consuming-rest-apis-by-using-the-new-rest-request-nodes-in-ibm-integration-bus/

Links

Videos:

10.0.04: REST, Graphical Mapper & Salesforce
https://youtu.be/XIK6QvNSHdY

10.0.0.5: Pushing REST APIs from IIB to API Connect
https://youtu.be/KTP94zIHKyI

10.0.0.5: Pushing REST APIs from IIB to API Connect on Bluemix and using Secure Gateway
https://youtu.be/JSgXum-iJnk

10.0.0.6 REST Request node: https://www.youtube.com/watch?v=r1EWkSWjnR8

10.0.0.6 Using REST Request node with security:
https://www.youtube.com/watch?v=C_6gPlrCHZQ

Summary

- IIB provides first class support for developing REST APIs to receive inbound requests and send back responses.
- The REST APIs can be created from scratch or a swagger (json or yaml) can be imported.
- The REST API catalog is shown in the project explorer where you can see at a glance the operations and their type (GET/POST etc)

- REST APIs can be pushed to API Connect and staged in a product using toolkit/webui/command line.

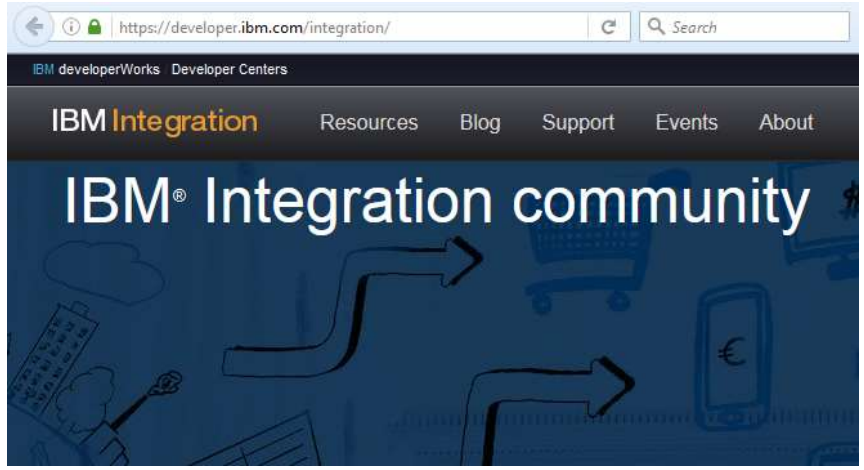
- IIB provides first class support for sending outbound requests and receiving responses to REST APIs using the REST Request, REST Async Request and REST Async Response nodes.
- Easy to configure parameters for operations on the REST Request node.
- Easier way to configure security on REST Request node.

IIB Sessions at Interconnect 2017

Session	Who	Time
2110A What's New in IBM Integration Bus	BT	Monday 16:15 – 17:00
2141A IBM Integration Bus Futures and Strategy **(Inner Circle only)**	BT	Tuesday 11:30 – 12:15
2158A Technical Introduction to IBM Integration Bus	GG	Tuesday 13:30 – 14:15
2118A Developing Integrations for IBM Integration Bus on Cloud	GG	Tuesday 14:30 – 15:15
2144A IBM Integration Bus Customer Roundtable	BT	Tuesday 15:45 – 16:30
2121A Docker and IBM Integration Bus	GG	Wednesday 09:00 – 09:45
2151A Effective Administration of IBM Integration Bus	SN	Wednesday 10:15 – 11:00
2144B IBM Integration Bus Customer Roundtable	BT	Wednesday 16:15 – 17:00
2124A Operational and Business Monitoring with IBM Integration Bus	SN	Thursday 09:30 – 10:15
2111A IBM Integration Bus and REST APIs	SN	Thursday 10:30 – 11:15
2166 IBM Integration Bus Version 10 Hands-On Scheduled Lab	GG+SN	Monday 13:00 – 14:45
9402 IBM Integration Bus Version 10 Hands-On Open Lab	None	Any Open Lab Session

In case powerpoint isn't your thing …

- https://developer.ibm.com/integration
- Lots of Blog entries, regular updates and links to product demo videos! All our recent enablement material is on youtube



IIB and Kibana dashboards	https://youtu.be/sCPrT2dHKSs
Running IIB in Bluemix Container Service	https://youtu.be/ybGOiPZO3sY
IIB and Kibana dashboards	https://youtu.be/sCPrT2dHKSs
IIB and Hybrid Connect	https://youtu.be/gWbxIooq3_g
IIB and LDAP	https://youtu.be/HrqY9MyfzNs
IIB LoopBack Request node	https://youtu.be/rUK_OQ5-Anw
Using IIB to integrate with MongoDB and Cloudant	https://youtu.be/Is1pphngUlM
Using IIB for REST, Graphical Mapping & Salesforce:	https://youtu.be/XIK6QvNSHdY
IIB, Kafka and Twilio SMS:	https://youtu.be/7mCQ_cfGGtU
Using Kafka with IIB	https://youtu.be/kYv0crxL86Y
Consuming REST APIs using the IIB REST Request node	https://youtu.be/C_6gPlrCHZQ
Easy demo of an IIB App Connect node	https://youtu.be/StwPbOiFKzk

Notices and disclaimers

Copyright © 2017 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular**, **purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli® Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

InterConnect
2017