



TIBCO Cloud Events Workshop

Virtual Workshop via Zoom

TIBCO Cloud Events – Asynchronous API Workshop

21st May 2020

CONFIDENTIALITY & DISCLAIMER

The information in this document is confidential information of TIBCO Software Inc. and/or its affiliates. Use, duplication, transmission, or republication for any purpose without the prior written consent of TIBCO is expressly prohibited.

This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release and availability dates for TIBCO products and services. This document is provided for informational purposes only and its contents are subject to change without notice. TIBCO makes no warranties, express or implied, in or relating to this document or any information in it, including, without limitation, that this document, or any information in it, is error-free or meets any conditions of merchantability or fitness for a particular purpose.

The material provided is for informational purposes only, and should not be relied on in making a purchasing decision. The information is not a commitment, promise or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

During the course of this presentation, TIBCO or its representatives may make forward-looking statements regarding future events, TIBCO's future results or our future financial performance. These statements are based on management's current expectations. Although we believe that the expectations reflected in the forward-looking statements contained in this presentation are reasonable, these expectations or any such forward-looking statements could prove to be incorrect and actual results or financial performance could differ materially from those stated herein. TIBCO does not undertake to update any forward-looking statement that may be made from time to time or on its behalf.

About TIBCO

TIBCO Software unlocks the potential of real-time data for making faster, smarter decisions. Our Connected Intelligence Platform seamlessly connects any application or data source; intelligently unifies data for greater access, trust, and control; and confidently predicts outcomes in real time and at scale. Learn how solutions to our customers' most critical business challenges are made possible by TIBCO at www.tibco.com.

Agenda & Timings

Time	Length	Subject	Presenter
13:00	15 Minutes	Welcome & Introduction to the team	
13:15	30 Minutes	Driving agile innovation across the Enterprise with the TIBCO Cloud Platform and our Event Based and API-led connectivity capabilities: Opportunities, challenges of connectivity and integration in today's reality	
14:00	30 mins	Product Roadmap	
14:30	2 Hours	Hands-on Labs	
15:30	10 Minutes	Break	
	30 Minutes	Discussion & Wrap-up	

Virtual Delivery / Digital Etiquette



Breakout Session - In Progress X

Breakout Session 1 Join

- Danny Mariscal
- John

Breakout Session 2 Join

- Brandon (not joined)
- Kim

Stop All Sessions

- We'll be using Video - but don't feel pressured to do so yourself!
- We'll be using a single Zoom session for the majority of the time. Breakout rooms will be available for individual help during the Labs
- Do Mute yourself when not speaking as this helps with the audio experience
- Use the "Raise Hand" button if you want to ask a question, but want to wait for a convenient point
- Put a question in the Chat window
 - Send it to everyone if you think the other participants would benefit
- Or just Unmute yourself and ask!





Mark Mussett

Senior Solutions Engineer

Veteran IT specialist with a career spanning 21 years in the Integration industry. Joined TIBCO in 2005, with 11 years in Professional Services as Technical Architect and 3 years in Pre-Sales organisations working with our Connected portfolio of products.

Expertise in API (Mashery), Integration (BusinessWorks, Flogo), Messaging (JMS, FTL, and RV), and Golang programming.



Dave Winstone

Senior Solutions Engineer

I have had the benefit of working for TIBCO for the last 14 years, and that has allowed me to work with a wide variety of customers – from the largest Banks and Telco's in the world, to the smallest innovators and disruptors.

I specialise in our “Connect” portfolio, allowing me to work with our customers on ensuring the right solutions and architectures are employed to provide the base for Digital Transformation.



Bryn Garton

Senior Solutions Engineer

Except for a brief spell as an Electrical Engineer my entire career has been spent working for software companies. For the majority of that time I have worked in the field of middleware across its many generations, with roles in Consulting, Software Development, Product Management and most recently Presales.

Since joining TIBCO in 2005 I have worked with TIBCO's customers across many industry verticals including Finance, Energy, Logistics, Utilities and Government.



Rishikesh Palve

Senior Product Manager

I am a Senior Product Manager at TIBCO. I have over 9 years of experience working in various roles across different groups such as P&T, Customer Engineering, PreSales and Product Management.

I take keen interest in APIs, Microservices, Event-driven architectures and Cloud-Native ecosystem. At TIBCO, I am responsible for building product roadmap and vision for our Cloud-Native, Event-driven product capabilities that are part of the TIBCO Connected Intelligence platform.

Customer Challenge

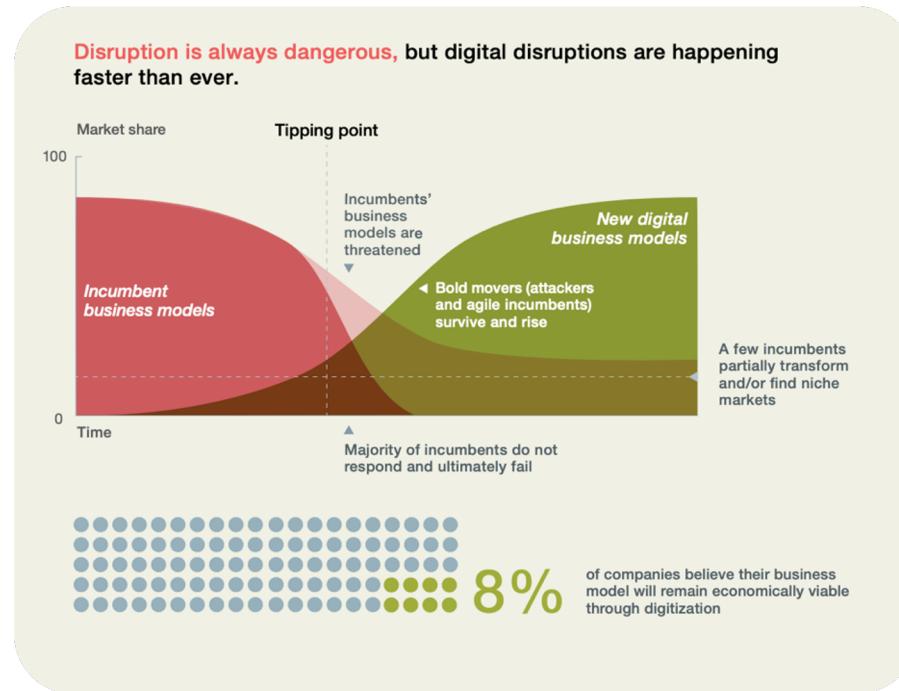
How Do You Create & Accelerate Value in the Digital World

TIBCO's Value To Our Customers

Delivering Connected Intelligence through our Digital Business Platform

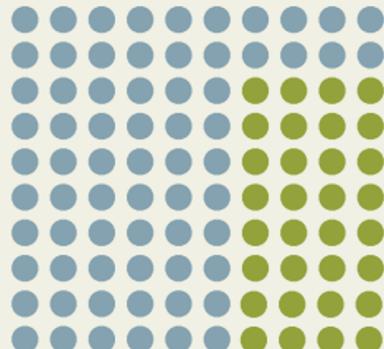
Digital disruption is happening *faster than ever*

- Incumbent Models are under attack
- Agile Movers Survive & Rise
- New Models Created
- <8% believe “their biz model will remain eco viable through digitalization”



Digital disrupts = Companies need to change how they work & play.. Ecosystems

By 2025, almost a third of total global sales will come from **ecosystems**.



68% Traditional economy

- Platforms facilitate Ecosystem Business Models
 - = Business Value
 - = Agility
- Businesses must be Open, API(s) will create the new models assisting new models & opportunities

How Are TIBCO's Customers Innovating for the Connected Intelligence Era?



\$600,000

Anticipated OR cost savings PA

"In real time, we are able to alert our rapid response team to go to the bedside of a patient who is likely to go into cardiac arrest. So far, we've successfully reduced the number of cardiac arrests in the hospital by an estimated 15 to 20%."

Chief Enterprise Architect, Christine Watts

Use Case

Integrated systems and centralized data to more effectively measure services and predict risks

<https://www.tibco.com/customers/scottish-environment-protection-agency>
<https://www.tibco.com/customers/university-chicago-medicine>
<https://www.tibco.com/formula-one-competitive-advantage>



60%

Saving of time compared with previous technologies used

"The software's ability to cover the [full data science development lifecycle](#): data retrieval, preparation, transformation, analysis, and final interactive dashboard presentation, meant it could easily meet the aspirations of our proposed agile informatics process," Informatics Head of Unit SEPA, Mark Hallard

Use Case

Evidence-based decisions and efficient and automated analytics



6 Years

Time to deliver own visualizations vs. 2 months

The driver changes gears about 100 times a lap. Every time there's a gear change, we collect around 1,000 data points. Those gear changes can be categorized into how fast they were, and how much stress they put on the gears. When you visualize that data, you can actually make the gear box last longer, or more importantly, make harsher gear changes. You can then find that if you put the gear box into one particular mode, it's roughly 50 milliseconds faster per lap. Cars that were fourth and fifth in qualifying were one one-thousandth of a second different, so 50 milliseconds matters. So for us to be able to make those harsher gear changes is important. We spent six years creating our own visualizations. TIBCO turned up, and in a matter of two months, did better work." Head of IT, Matt Harris

Use Case

Real Time Car Dynamics Changing, Continuous Improvement

Only TIBCO can guarantee its customers digital sovereignty by providing:

Comprehensive Platform

Capabilities for Digital Execution

Innovate and collaborate without constraints

Accelerated Innovation

Simplification, Automation & Agility

Deploying anywhere and everywhere at pace

Freedom

Choice & Scale

Operational Excellence From experimentation to mission critical

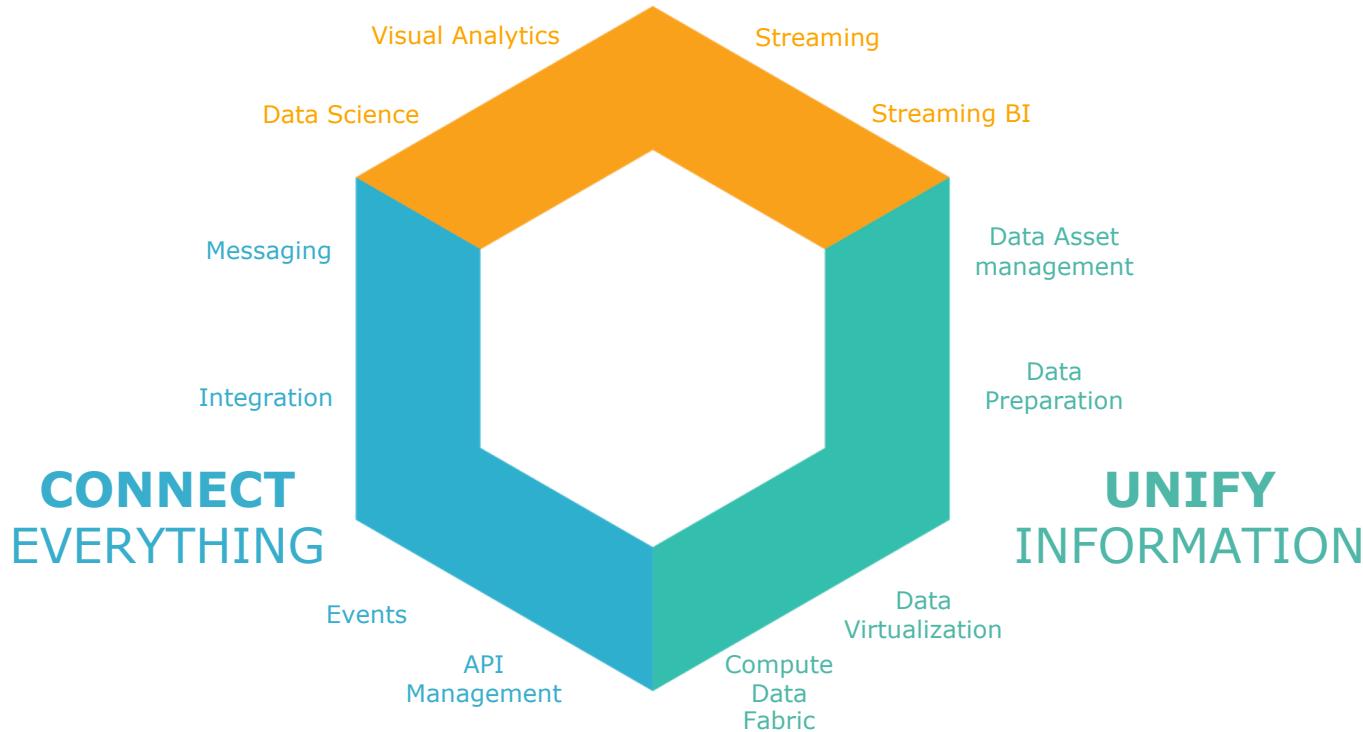


Digital is Built On This Foundation..

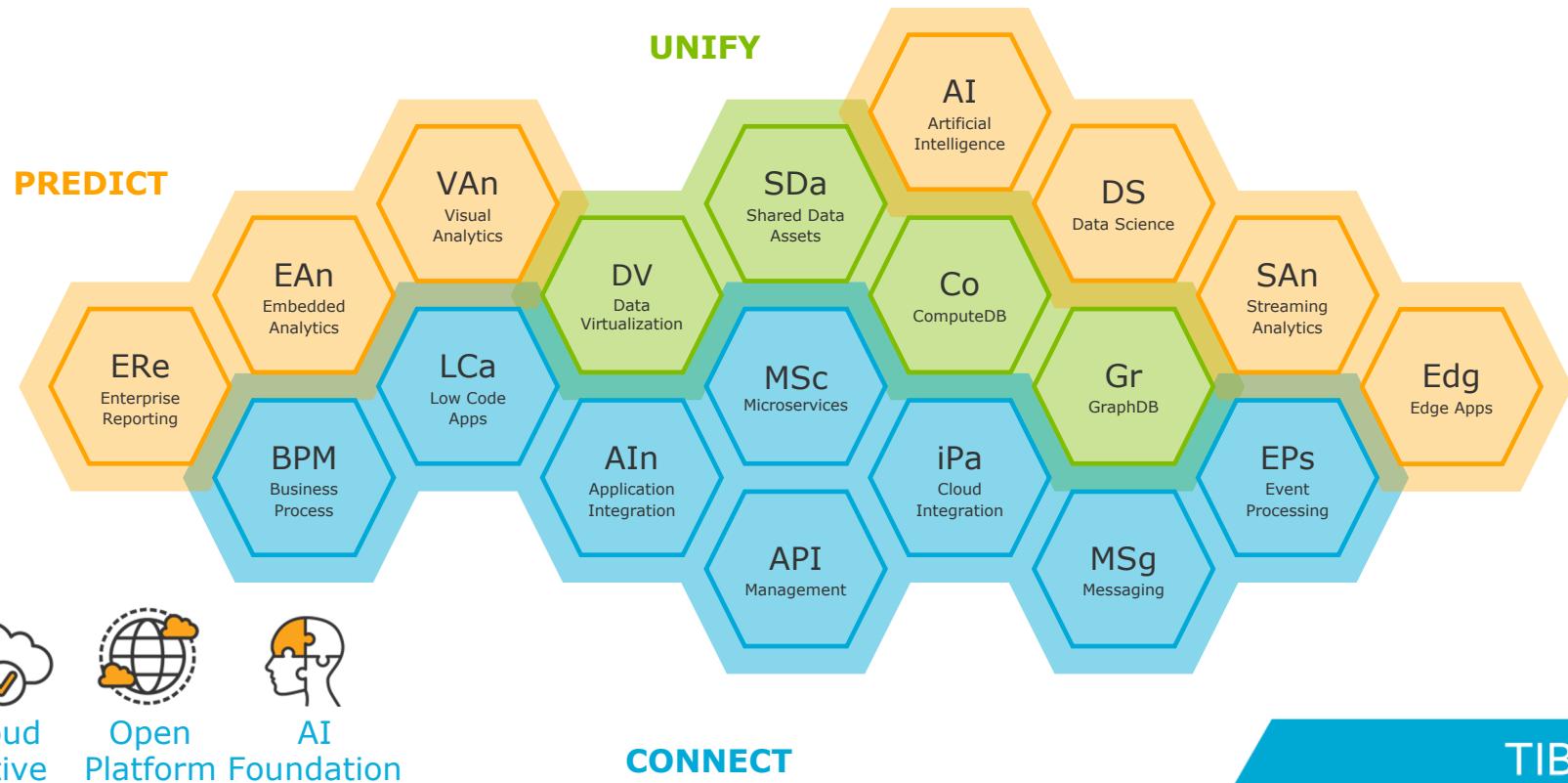


Best-In-Class, Open and Independent

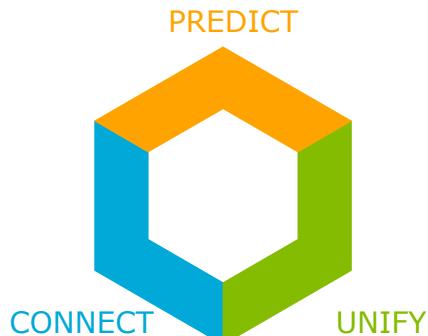
PREDICT



How TIBCO delivers the Connected Intelligence Vision

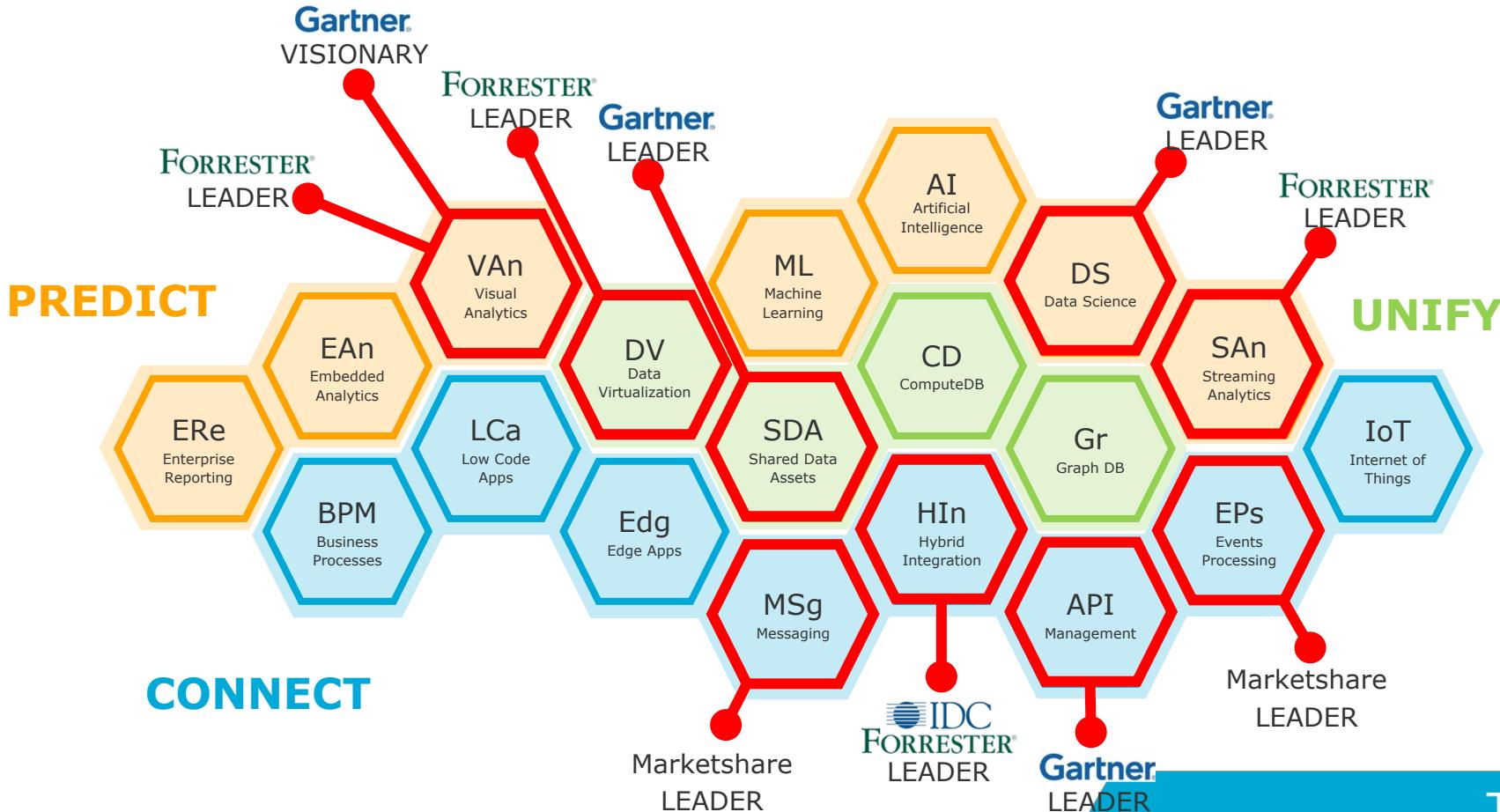


Built on Strong Core Principles



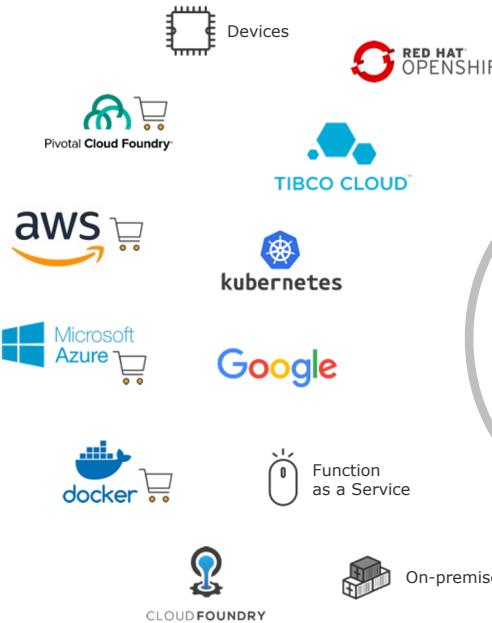
- | | |
|--|---|
| 
Cloud Native | <ul style="list-style-type: none">• Re-architected core capabilities as cloud-native 'engines'• Deliver the same core cloud-native components across workloads• Significantly reduced or removed any proprietary dependencies• Devops/Cloudops/Dataops support delivered as open source |
| 
Open Platform | <ul style="list-style-type: none">• Provide customers the same tools used internally to extend capabilities• Open source delivery of new key capabilities like Project Flogo• Continued adoption of enterprise and emerging open source (e.g. Kafka)• Increasingly active engineering participation in larger scale OSS projects |
| 
AI Foundation | <ul style="list-style-type: none">• Application of AI principles and technology across capabilities to enhance workflow and user productivity• Integration to leading AI frameworks both for real-time projects (Flogo with TensorFlow) as well as Data Science modeling and design• Increasing use of AI as a platform for developing new capabilities such as adaptive business logic and execution |

Market Leadership Across The Platform



Available anywhere, everywhere

TIBCO Connected Intelligence



marketplaces proposing
TIBCO Cloud services

Open Source

Give developers easy access to core capabilities of the Connected Intelligence Cloud

SaaS

Quickly get up and running with a fully TIBCO hosted and managed service

Hybrid

With all options for self-host, multi-cloud, or edge deployments

Dynamic Cloud

Optimize cost by dynamically scaling on cloud infrastructure such as Amazon Web Services

API-Led Strategy Is the Connective Tissue of Digital Business

App Dev & Integration

Internal Agility
Standardize services



Cloud-native
Microservices & microgateways



Integration
SaaS and Hybrid Services



Digital Business Platform

Channel Expansion
New Distribution



B2B & B2C Innovation
Partner Collaboration



New Business Models
Becoming an ecosystem platform



API Platform

Enterprise Data, Services,
and Applications

TIBCO Connected Intelligence Cloud

Capabilities

PREDICT



Data Management



Advanced Data Visualization



Streaming Analytics



Shared services



Billing



SSO



Account Management



Customer Servicing



Service Registry



Integration and
API
Management



Messaging and
Events
Processing



Digital
Process
Automation

**INTERCONNECT
EVERYTHING**



TIBCO CLOUD™ Platform

TIBCO®

TIBCO Cloud in a Nutshell



ACME Parent Production ...

Oregon



Todd B

Home

Team Members

Subscriptions

Settings

Downloads

Welcome to your TIBCO Cloud™



Discover a new way
to get started with
the power of TIBCO
Cloud



TIBCO®

Capabilities available on TIBCO Cloud

- TIBCO Cloud Nimbus
- TIBCO Cloud Live Apps
- TIBCO Cloud Integration
- TIBCO Cloud Mashery
- TIBCO Cloud AuditSafe
- TIBCO Cloud Events
- TIBCO Cloud Messaging
- TIBCO Cloud Spotfire



TIBCO Cloud Services

At-a-Glance



High Availability



Disaster Recovery



**Security, Privacy,
& Compliance**



PKI Management



Role-based Access



Auditability



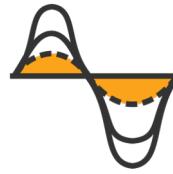
**Service Level
Agreement**



Reporting



**Operational
Dashboard**

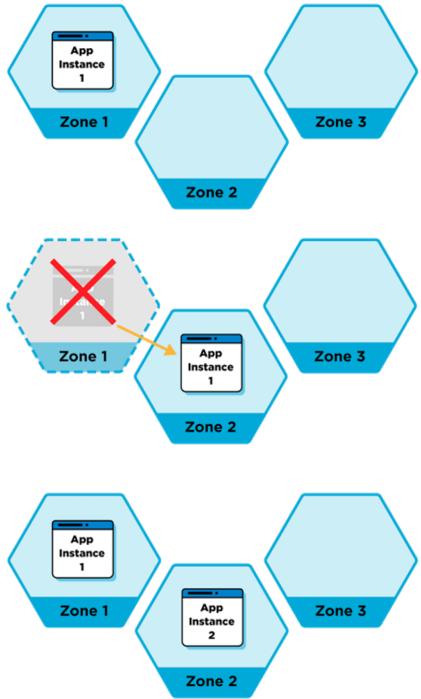


**Transaction
Tracking**



Support

High Availability

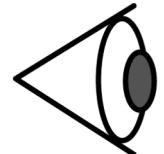


- Each application is executed in an availability zone automatically selected based on available resources.
- Automatic switch-over of apps to new zone if they go down or stop reporting status.
- Scale apps across multiple zones for high availability - load is distributed evenly by the system.

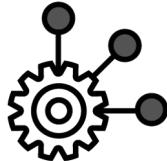
Security, Privacy and Compliance



Customer traffic, secrets
and encryption



Authentication and
Authorization

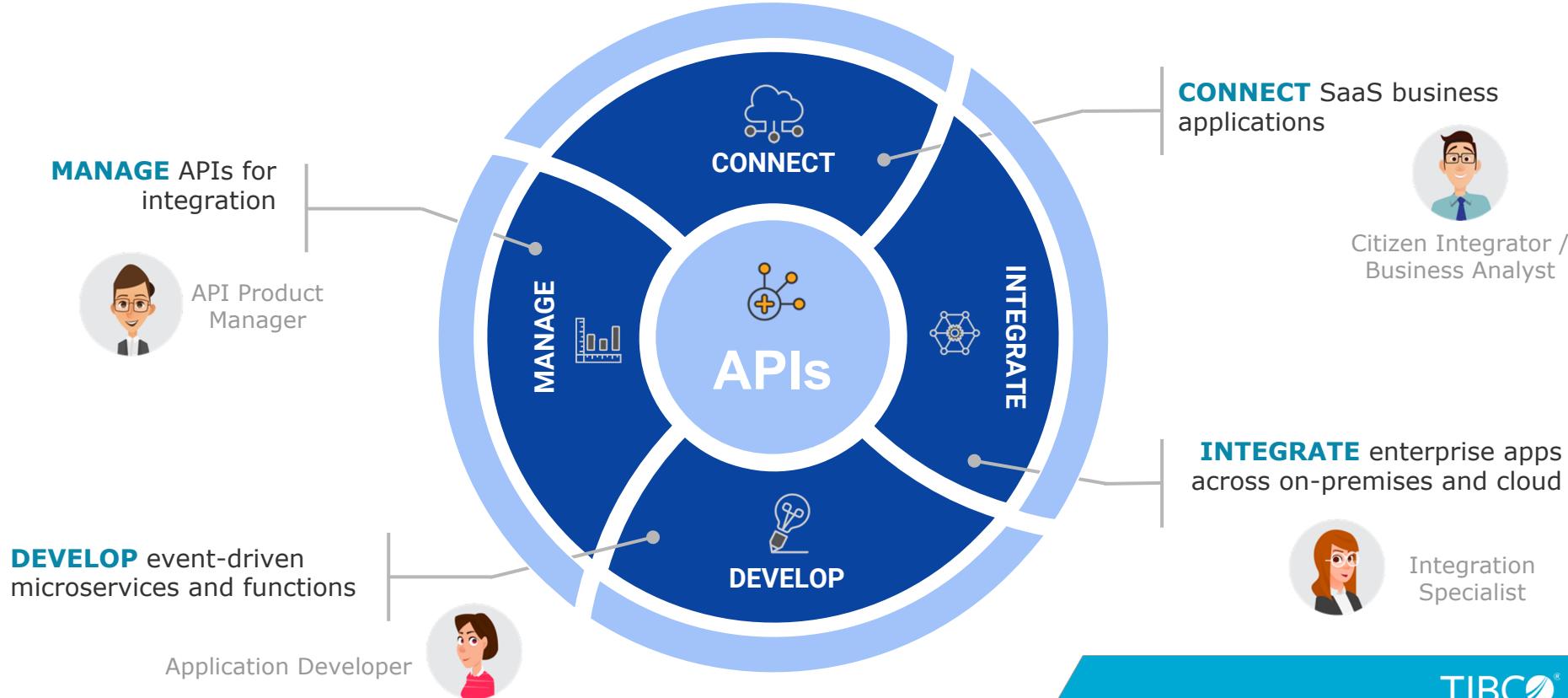


Secure API's for Mobile
and Command line
interface

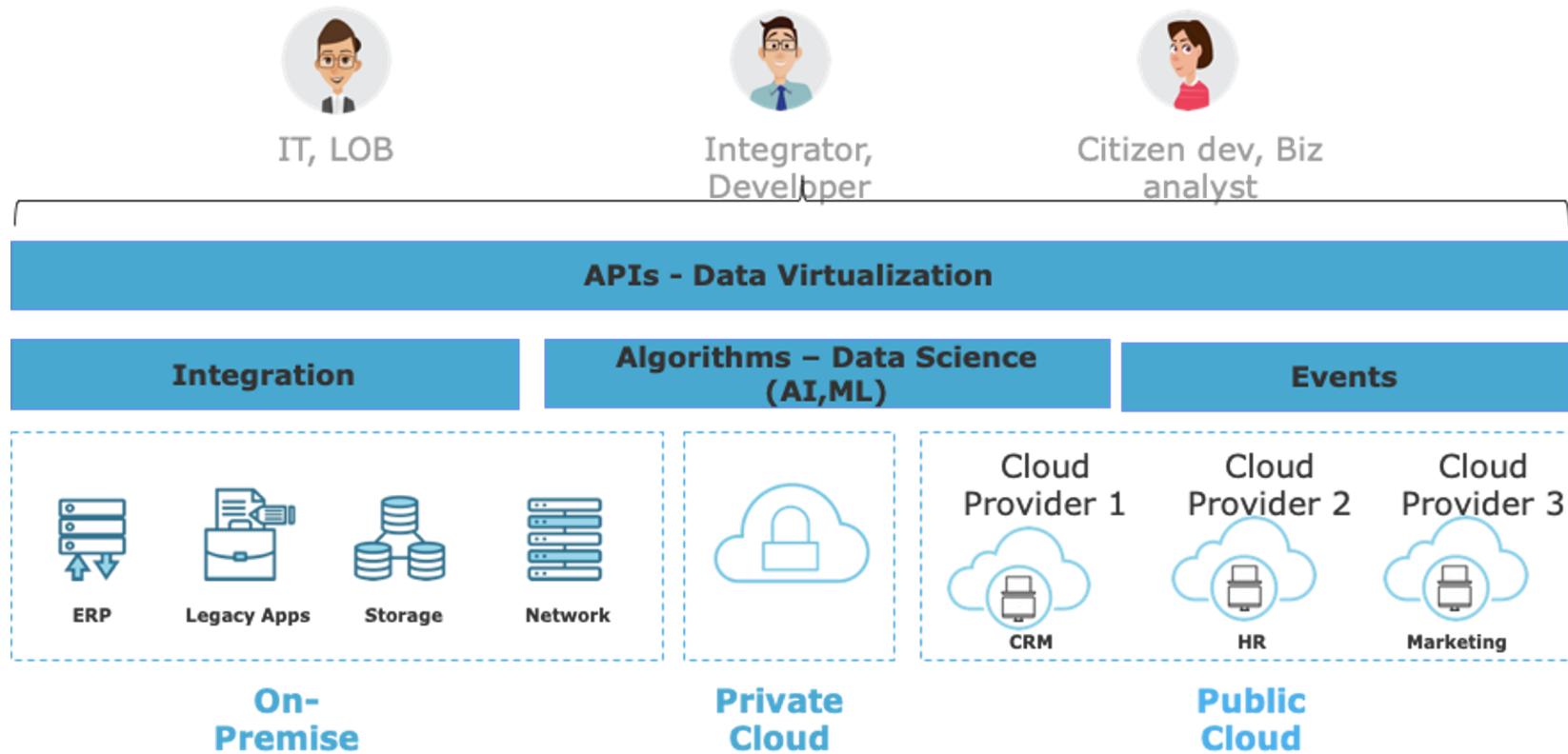
- Multi-tenant platform with logical separation of customers.
- Built with privacy and security in mind.
- Encrypted network traffic and data storage.
- Hybrid Connectivity

Accelerate Integrations Across Your Business

Enable Collaborative Integration for Everyone through Tailored User Experiences



Desired state for enterprises

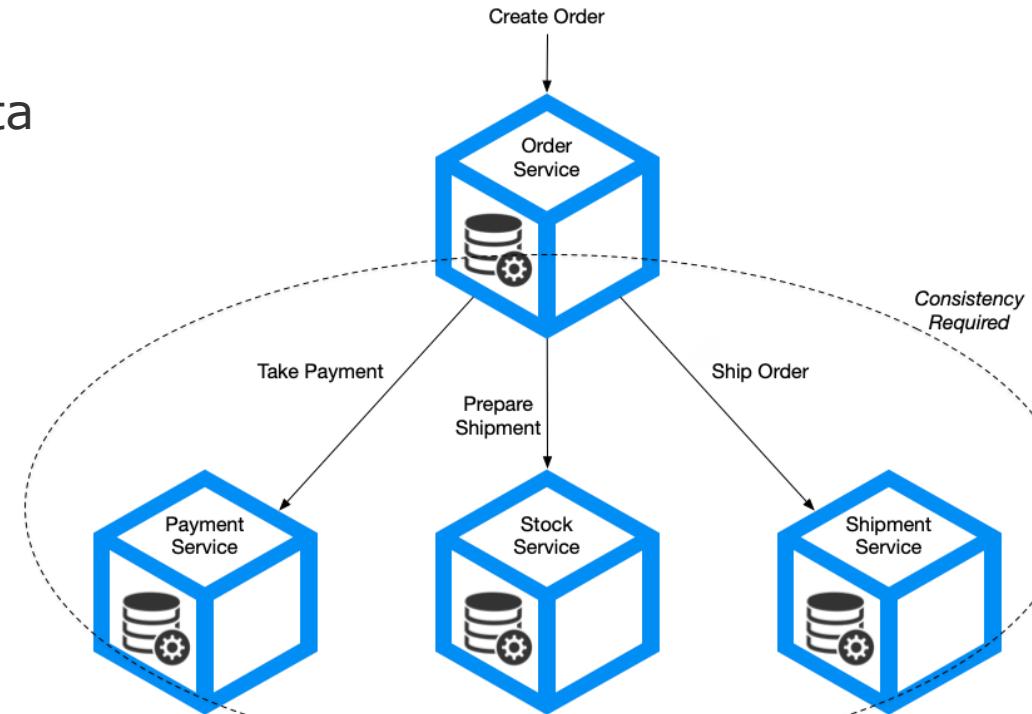


The Labs



Business Transactions in a Microservice Architecture

- Need to maintain data consistency across bounded context
- Saga Pattern solves the challenge of consistency through the use of Choreography or Orchestration of events across boundaries



Saga Pattern

- Pattern used to maintain data consistency in a Microservice architecture.
- Relies on the ability of every Microservice being able to maintain coherent and consistent state.
- Pattern utilises an approach of coordinating each step in the business transaction.
- By using messaging we can maintain transactional flow between participants.

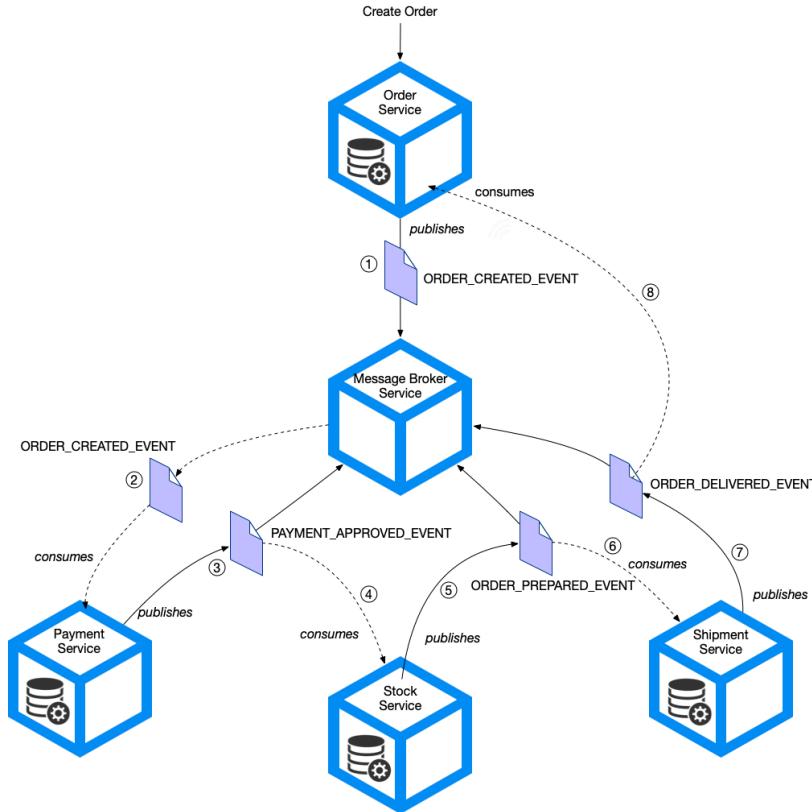
Use Case – Create Order

1. Order Service - Create an Order in PAYMENT_PENDING state.
2. Payment Service - Authorize payment for Order.
3. Order Service - Update Order to PAYMENT_APPROVED and AWAIT_PICKING state.
4. Stock Service - Prepare Order Items for dispatching.
5. Order Service - Update Order to ITEMS_PICKED state.
6. Shipment Service - Create Delivery of Order.
7. Order Service - Update Order to ORDER_SHIPPED state.

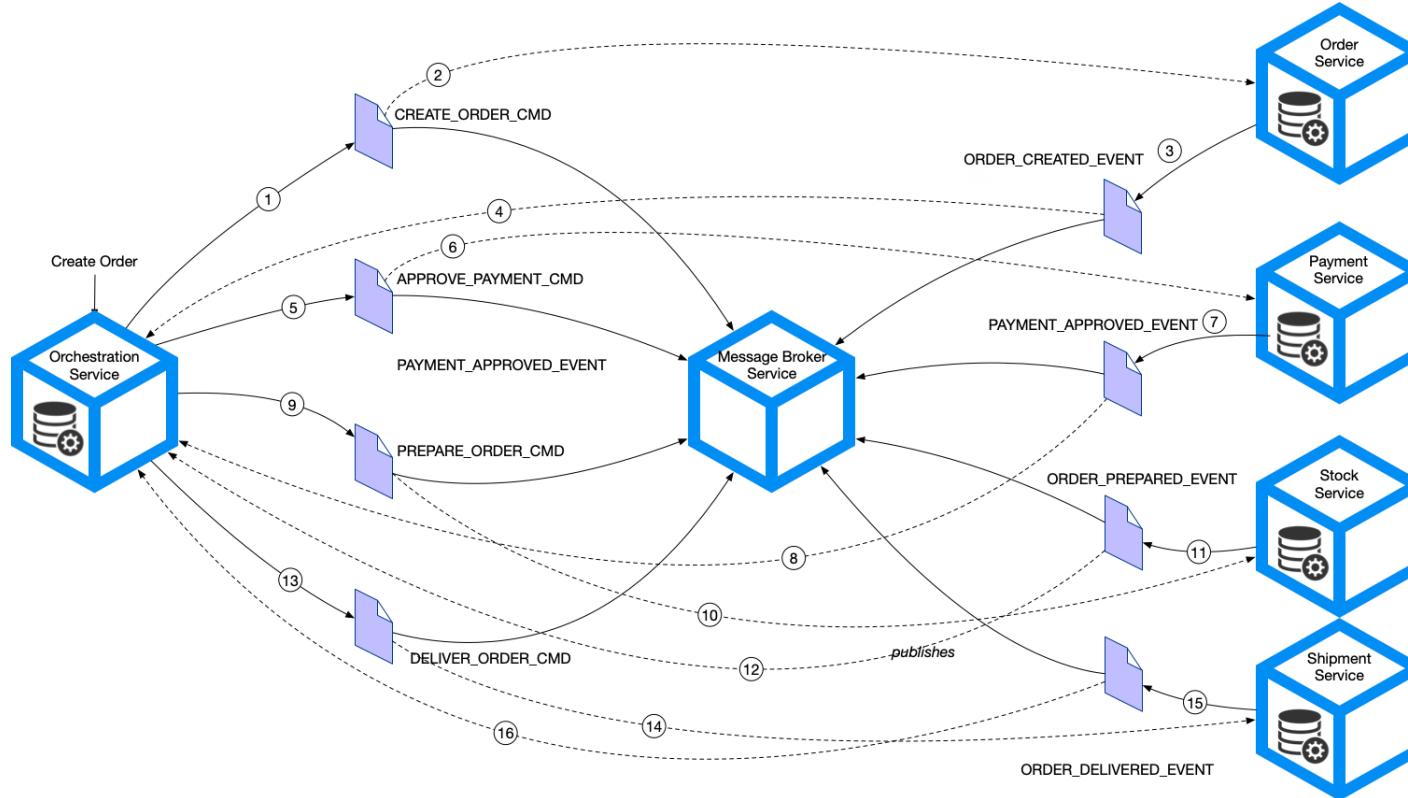
Coordinating Transactions

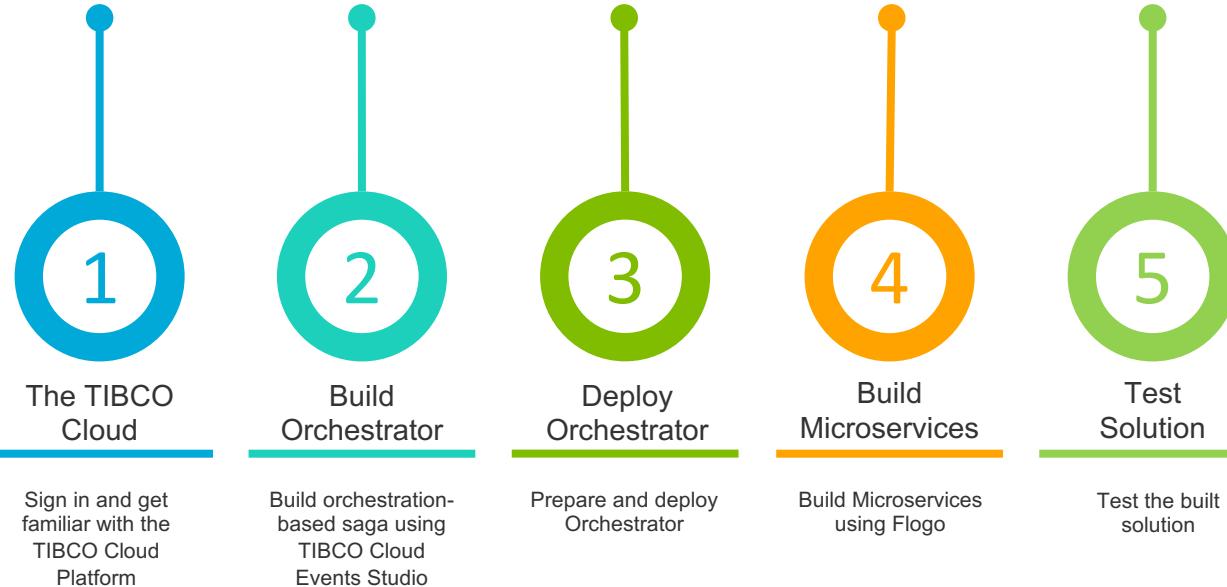
1. Choreography
2. Orchestration

Choreography-based Saga



Orchestration-based Saga







The TIBCO Cloud

Sign in and get familiar with the TIBCO Cloud Platform



Build Orchestrator

Build orchestration-based saga using TIBCO Cloud Events Studio



Deploy Orchestrator

Prepare and deploy Orchestrator



Build Microservices

Build Microservices using Flogo



Test Solution

Test the built solution

Key Resources

 TIBCO CLOUD™ <http://cloud.tibco.com>



Download and extract the Hands On Lab : <https://github.com/TIBCOUK/AsyncAPIs>

TIBCO / AsyncAPIs

No description, website, or topics provided.

Manage topics

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Branch: master New pull request Create new file Upload files Find file Clone or download

remusset Updated docs

doc Updated docs

src Latest Fligo Apps

.gitignore first commit

README.md Updated docs

README.md

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/TIBCOUK/AsyncAPIs>

Open in Desktop Download ZIP 4 months ago

TIBCO Cloud Asynchronous APIs Workshop - Business Transactions

Note: Before you start the labs, download the repo (Clone or download > Download ZIP), and unzip it to a local directory.

1. Setup Your Environment

- Sign in to TIBCO Cloud™
- Download TIBCO Business Studio™
- Download TIBCO Cloud™ Events Studio
- Download Postman
- Download github repository



TIBCO Community :

- <https://community.tibco.com/products/tibco-cloud-events>
- <https://community.tibco.com/products/tibco-cloud-integration>
- <https://community.tibco.com/products/tibco-activematrix-businessworks>



Docs TIBCO : <https://docs.tibco.com/>

Sign Up



The TIBCO
Cloud

- Go to cloud.tibco.com
 - <https://cloud.tibco.com>
 - Register
 - Login: your email address
 - Password: your password
 - Region: **EU (Ireland)**
 - Check your email for activation email
 - And Junk folder!

The screenshot shows the TIBCO Cloud homepage. At the top right, there is a 'TRY NOW' button with a green arrow pointing to it from the left. Below the button, there is a section titled 'TIBCO Cloud™ Events' with a description of the service. To the right of this section is a 'FREE TRIAL' button. Further down the page, there is a diagram illustrating the three pillars of TIBCO Cloud: CONNECT, UNIFY, and PREDICT, represented by three overlapping circles in blue, green, and orange respectively. A purple banner at the bottom of the page reads 'The power of TIBCO, in the cloud. Get started today with these services on TIBCO Cloud™'.

Add Subscriptions to the Trial



TIBCO CLOUD™

Oregon

Subscriptions

Home Team Members Subscriptions Settings Downloads

Subscriptions in Oregon

Integration TRIAL

Named users **Unlimited** Expires in **1** MONTH

AVAILABLE

[Details](#) [Upgrade](#) [Launch](#)

Mashery® TRIAL

Named users **Unlimited** Expires in **1** MONTH

AVAILABLE

[Details](#) [Upgrade](#) [Launch](#)

Add subscription

+ Add subscription



2 0

TRIAL ACTIVE

Add subscription

Add Integration & Messaging



TIBCO CLOUD™

Add to your TIBCO CLOUD™ Services in Ireland

Integration View plans <small>1</small> Quickly model and test APIs. Connect cloud and on-premise services and applications. Free trial Add or upgrade	Live Apps View plans <small>1</small> Automate and digitize your processes. Document and visualize communication. Build smart apps in minutes. Free trial Add or upgrade	Messaging View plans <small>1</small> Enhance your mobile, web or stand-alone applications with real-time communication powered by TIBCO's messaging as a service. Free trial Add or upgrade
Spotfire® View plans <small>1</small> TIBCO Cloud™ Spotfire® is a smart visual analytics solution with built-in data wrangling, geo-analytics and predictive analytics. Free trial Add or upgrade	Nimbus® View plans <small>1</small> Map your way to a better understanding of your business. Free trial Add or upgrade	AuditSafe View plans <small>1</small> Connect your app and start logging critical business audit trail. Free trial Add or upgrade

Download and Install TIBCO Cloud Events Studio



<https://eu.events.cloud.tibco.com/#!/downloads>

TIBCO CLOUD™ Events

Ireland Mark M

WebStudio

Downloads

Apps VPN Connection Downloads

TIBCO Cloud™ Events Command Line Interface

TIBCO Cloud™ Events Studio

Here are the tools to get started!

Mac Current version Download

TIBCO Cloud™ Events Studio v2.1.0.064

TIBCO Cloud Events Studio is an Eclipse platform based design-time IDE where you design and test your TIBCO Cloud Events applications. You may then also deploy your application to TIBCO Cloud from the IDE.

Download for another platform

Windows Download

Linux Download

Download and Install TIBCO Business Studio for BusinessWorks



<https://eu.integration.cloud.tibco.com/download>

The screenshot shows the 'Downloads' section of the TIBCO Cloud Integration website. At the top, there's a navigation bar with tabs: Apps, API Specs, Connections, Extensions, VPN Connections, and Downloads (which is currently selected). Below the navigation, there's a heading 'Downloads' and a sidebar with links to 'TIBCO® Cloud - Command Line Interface' and 'TIBCO Business Studio for BusinessWorks™'. A green arrow points from the sidebar link to the 'TIBCO Business Studio for BusinessWorks™' section. This section contains a sub-headline 'Here are the tools to get started!', a note about license requirements, and download links for 'Mac', 'Windows', and 'Linux'. The 'Windows' link is highlighted with a green box and a green arrow pointing to its 'Download' button. Below each platform section, there's a brief description of the tool and its current version.

TIBCO Business Studio for BusinessWorks™

Current version 2.5.2
Mar 05 2020

Download

TIBCO Business Studio for BusinessWorks™

TIBCO Business Studio for BusinessWorks is the design-time IDE (based on Eclipse) where you create and test TIBCO BusinessWorks Cloud processes. You use TIBCO Business Studio for end-to-end API development. You can create new APIs, orchestrate resources, and integrate cloud-based applications in a short time. A model-driven development approach is supported, with a rich set of palettes for process design. These palettes can be used to visually create and test processes.

Download for another platform

Windows

Current version 2.5.2
Mar 05 2020

Download

Linux

Current version 2.5.2
Mar 05 2020

Download

Download TIBCO Cloud Messaging Java SDK



<https://eu.messaging.cloud.tibco.com/tcm/ui/downloads>

The screenshot shows the TIBCO Cloud Messaging interface. At the top, there's a navigation bar with tabs for Status, Download SDKs (which is currently selected), and Authentication Keys. Below the navigation bar, there are two tabs: eFTL (selected) and FTL. On the left, there's a sidebar with instructions for downloading the SDK, including a link to the eFTL download page and a 'Compare messaging services' button. The main content area is titled 'TIBCO Cloud™ Messaging SDKs for TIBCO eFTL™'. It lists several language options with 'Start coding' links and 'Download' buttons. A green arrow points to the 'Java / Android' download button.

[eFTL download SDK page](#)
[Compare messaging services](#)

Instructions

Download the software development kit (SDK) for your app by choosing a language and clicking the download button. The downloaded archive file has all the resources needed to start coding.

After you have downloaded an SDK, click on the corresponding Start Coding button. This will take you to a set of tutorials for your chosen language.

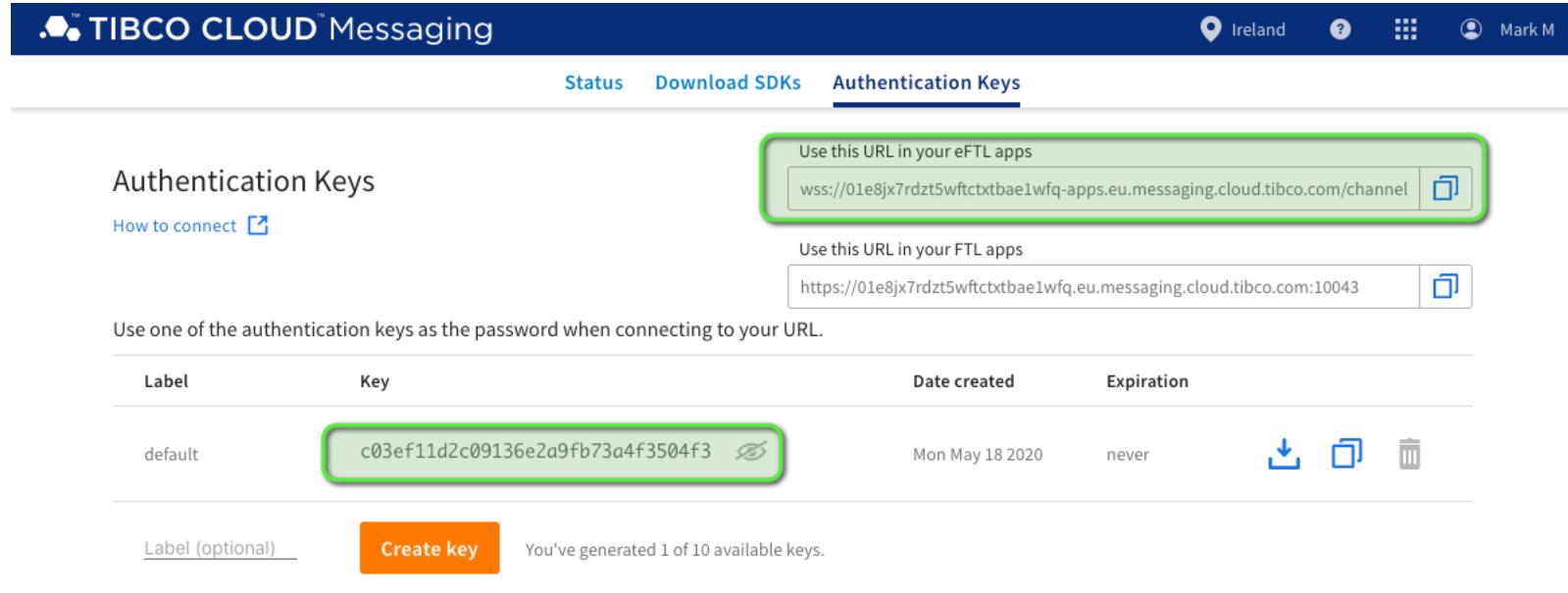
The tutorials require you to have a URL and an authentication key to make a connection. You'll find both on the [Authentication Keys](#) page.

TIBCO Cloud™ Messaging SDKs for TIBCO eFTL™

Language	Action
Go	Start coding Download
JavaScript / Node.js	Start coding Download
Objective-C / iOS	Start coding Download
Java / Android	Start coding Download
C#	Start coding Download

Record TCM Information

<https://eu.messaging.cloud.tibco.com/tcm/ui/admin>



TIBCO CLOUD™ Messaging

Ireland ? Mark M

Status Download SDKs Authentication Keys

Authentication Keys

How to connect ↗

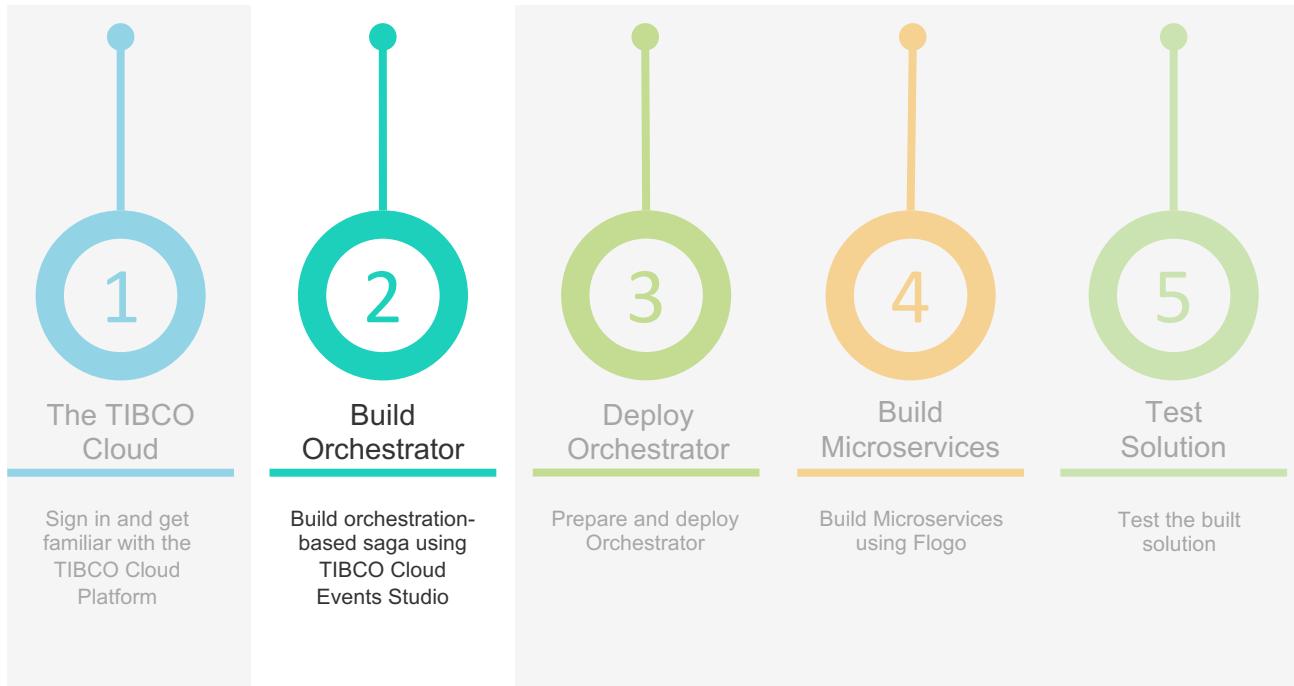
Use this URL in your eFTL apps
wss://01e8jx7rdzt5wftctxbae1wfq-apps.eu.messaging.cloud.tibco.com/channel [Copy](#)

Use this URL in your FTL apps
https://01e8jx7rdzt5wftctxbae1wfq.eu.messaging.cloud.tibco.com:10043 [Copy](#)

Use one of the authentication keys as the password when connecting to your URL.

Label	Key	Date created	Expiration	
default	c03ef11d2c09136e2a9fb73a4f3504f3 Copy	Mon May 18 2020	never	Download Edit Delete

Label (optional) [Create key](#) You've generated 1 of 10 available keys.





Build
Orchestrator

Lab Objectives

In this lab, you will create and deploy a TIBCO Cloud™ Events solution that will implement an orchestration-based saga based on a very simple Order use-case:

1. Take Payment - implemented by Payment Microservice
2. Prepare Order - implemented by Stock Microservice
3. Ship Order - implemented by Delivery Microservice

Getting Started with TIBCO Cloud™ Events Studio - Create Project

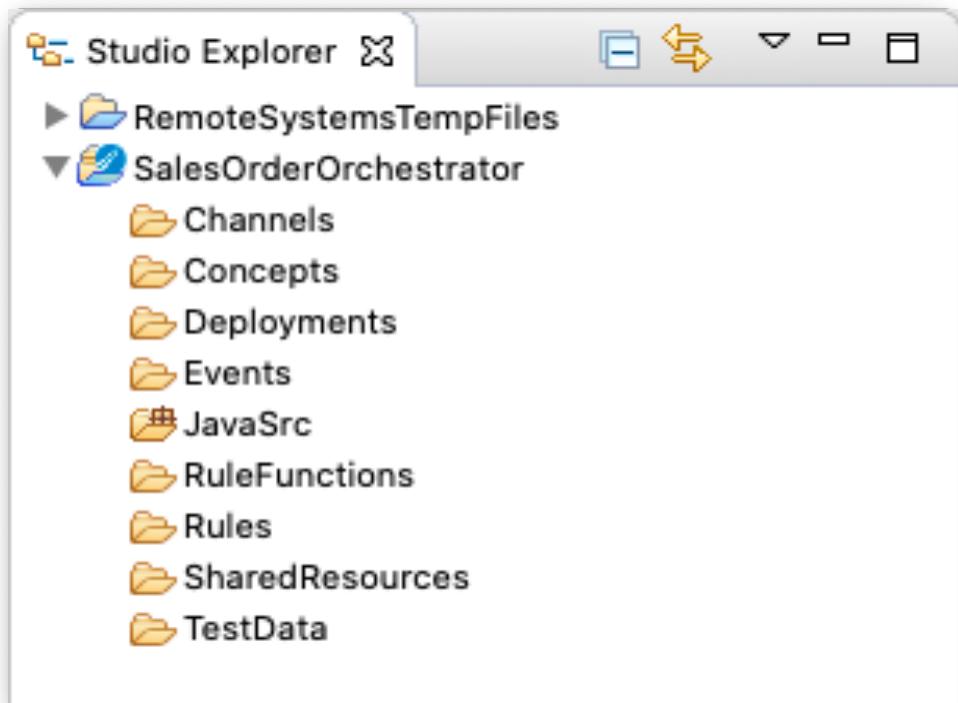


Let's start, open TIBCO Cloud™ Events Studio and create a new Workspace, your screen should look like this:

Right click in the Studio Explorer view, select New->Studio Project...

Enter 'SalesOrderOrchestrator' in the Project name field and click 'Finish' button.

Expand the Project list within the Studio Explorer view to show your newly created project...



Create Concept Definitions



Within TIBCO Cloud Events, a Concept definition is used to define objects and their properties. Concepts are similar to object-oriented concept of a class in that they represent class-level information.

At runtime, the instance of a concept are called objects and are persisted in memory or database.

For our use-case we are going to create 4 concepts:

1. **Order** - holds the Order properties (Order ID , Customer ID, email, and Order State)
2. **Order Item** - holds the Order Items (Item Unit Price, Quantity Ordered, Line Price)
3. **Shipping Address** - holds the Order Shipping properties (Address Lines, City, Country, and Postcode)
4. **InFlightOrders** - tracks Orders ID

Create Shipping Address Concept

2

Build
Orchestrator

Right click in the Studio Explorer view, select New
>Concept...

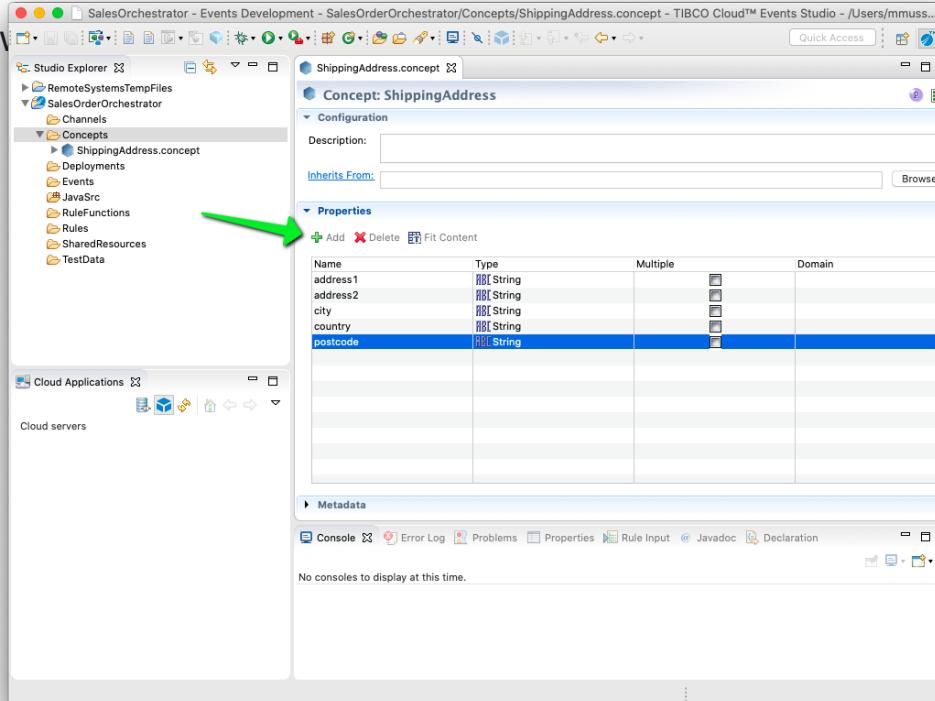
A dialog box will appear for you to create a new
concept.

Enter 'ShippingAddress' in the concept name
field. Click 'Finish' button.

Using the Concept Properties editor, click +
and add the following 5 properties:

- address1 - Type=String
- address2 - Type=String
- city - Type=String
- country - Type=String
- postcode - Type=String

Click 'Save' icon on the toolbar.



Create Order Item Concept



Build
Orchestrator

Now that you've created your first concept, go ahead and create Order Item concept.

This time use 'OrderItem' as the Concept name.

Add the following 3 properties to the Order Item concept:

- item_name - Type=String
- price - Type=double
- quantity - Type=int

Click 'Save' icon on the toolbar.

The screenshot shows the 'OrderItem.concept' configuration screen. The 'Properties' section contains three properties:

Name	Type	Multiple	Domain
item_name	String	<input type="checkbox"/>	
price	double	<input type="checkbox"/>	
quantity	int	<input type="checkbox"/>	

At the bottom, there is a 'Metadata' tab.

Create In-Flight Order Concept



Build Orchestrator

You should be a 'dab-hand' now at creating concepts, go ahead and create In-flight Order Concept.

This time use 'InFlightOrders' as the Concept name.

Add the following property to the In-Flight Order concept:

- order_id - Type=String

Click 'Save' icon on the toolbar.

Create Order Concept



Build
Orchestrator

Create Order Concept. This time use 'Orders' as the Concept name.

Add the following property to the In-Flight Order Concept:

- order_id - Type=String
- customer_id - Type=String
- email - Type=String
- items - Type=ContainedConcept,
Resource=OrderItem, Multiple=enabled
- address - Type=ContainedConcept,
Resource=ShippingAddress

The screenshot shows the 'Order.concept' configuration screen in the Build Orchestrator. The 'Properties' section contains the following table:

Name	Type	Multiple	Domain
order_id	ABC String	<input type="checkbox"/>	
customer_id	ABC String	<input type="checkbox"/>	
email	ABC String	<input type="checkbox"/>	
items	/Concepts/OrderItem	<input checked="" type="checkbox"/>	
address	/Concepts/ShippingAddress	<input type="checkbox"/>	
state	ABC String	<input type="checkbox"/>	

Click 'Save' icon on the toolbar.

Create Connection Shared Resource Definitions



Build
Orchestrator

We need to get data in and out of TIBCO Cloud Events, to do this we need to create two connections:

- HTTP Connection - this will be used by TIBCO Cloud Events to receive the initial Order JSON request.
- TCM Connection - this will be used by TIBCO Cloud Events to send & receive asynchronous messages with.

Shared Resources can be created in TIBCO Cloud Events and used by components of the solution.

Shared Resources configure the connection and contain properties such as host address, URL, or passwords.

Create HTTP Connection



Build
Orchestrator

Right click in the Studio Explorer view, select New->Other...

Enter 'HTTP' in the Wizards field, click 'Next >' button.

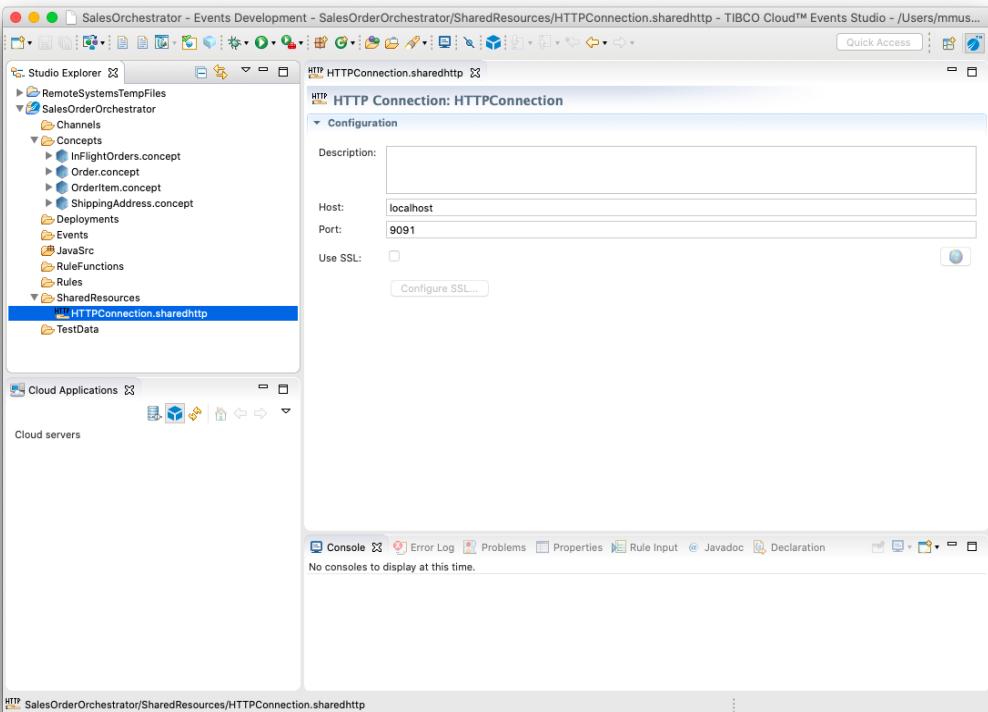
Select 'SharedResources' folder.

Enter 'HTTPConnection' in the Filename field, click 'Finish'.

Enter 'localhost' in the Host field in the HTTP Connection view.

Enter '9091' in the Port field in the HTTP Connection view.

Click 'Save' icon on the toolbar.



Create TCM Connection



Build
Orchestrator

Right click in the Studio Explorer view, select New->Other...

Enter 'TCM' in the Wizards field, click 'Next >' button.

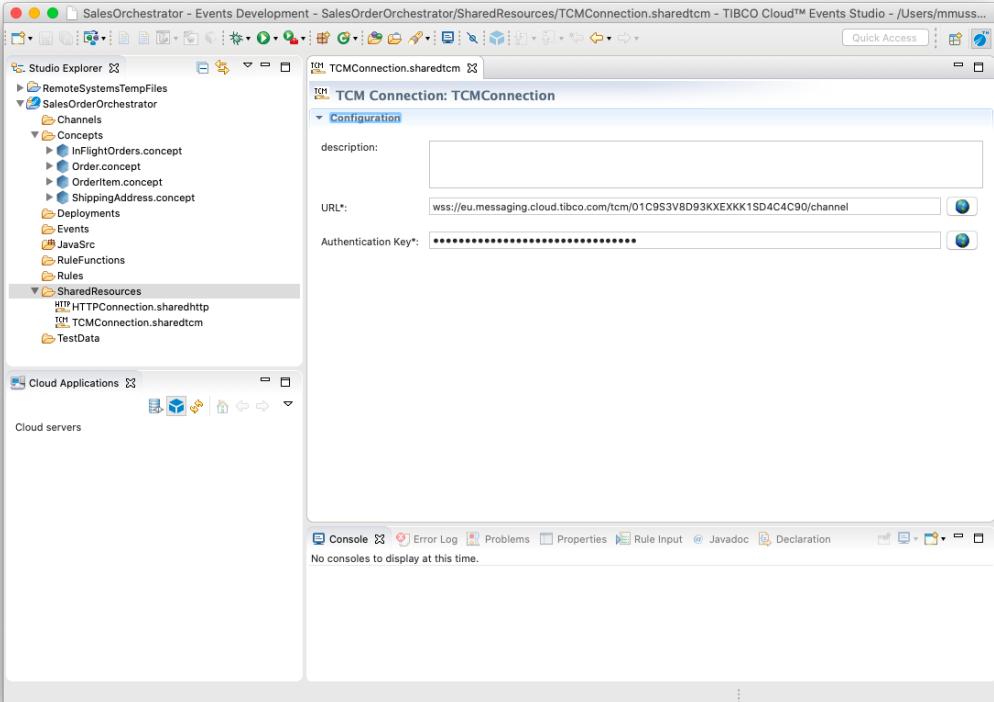
Select 'SharedResources' folder.

Enter 'TCMConnection' in the Filename field, click 'Finish'.

Paste in the TCM URL (make sure to over-write what's there) in the URL field in the TCM Connection view.

Paste in the TCM Authentication Key in the Authentication Key field in the TCM Connection view.

Click 'Save' icon on the toolbar.



Create Channels & Destinations



Build
Orchestrator

Channels are the messaging transport where events enter or exit TIBCO Cloud Events.

A channel uses connection resource.

A channel may have one or more destination, which represents a source of messages from that connection resource as well as a sink of messages that are to be sent to the outside.

Whenever we wish to interact with participants either sending or receiving messages we do so through Channels and Destinations.

For our use-case we are going to create 4 channels:

- **Order** - HTTP Channel used for sending in an Order request.
- **Payment** - TCM Channel used to communicate with the Payment Microservice using async messaging.
- **Stock** - TCM Channel used to communicate with the Stock Microservice using async messaging.
- **OrderDelivery** - TCM Channel used to communicate with the Delivery Microservice using async messaging.

Create Order Channel



Build
Orchestrator

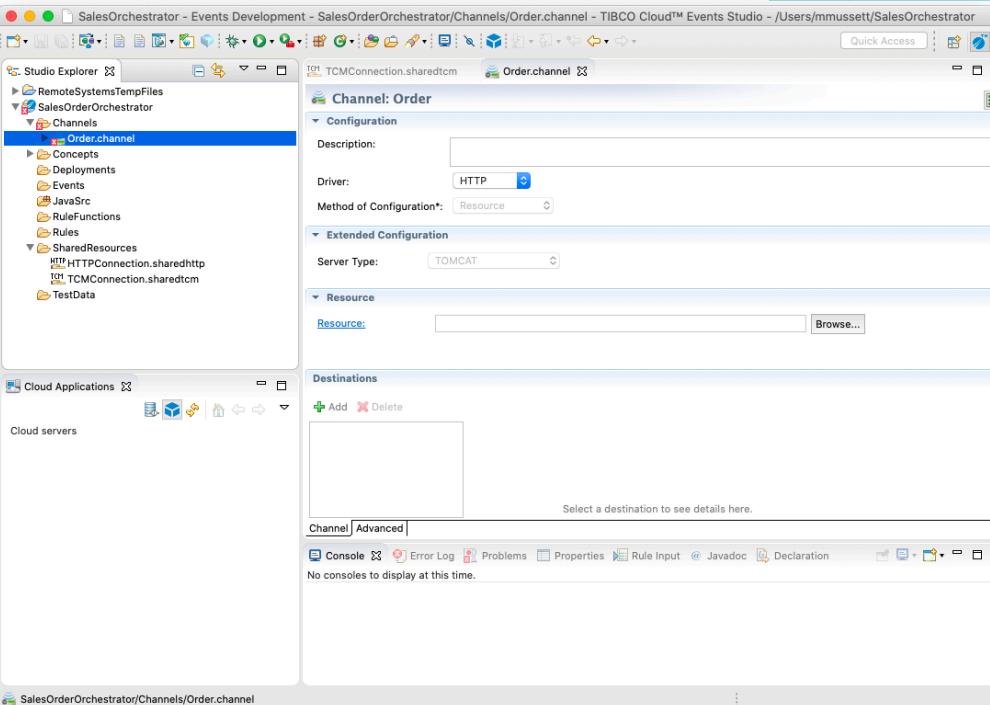
Right click in the Studio Explorer view, select New->Channel...

Enter '**Order**' in the Channel name field, select 'HTTP' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the HTTPConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.



Create Destinations on Order Channel



Build
Orchestrator

Click '+' in the Destinations panel...

A new destination will be created called 'NewDestination_0'.

Set the Name field to '**OrderDest**'.

Set the Default Event to '/Events/OrderEvent'.

Click 'Save' icon on the toolbar.

The screenshot shows the 'Destinations' configuration screen. On the left, there's a tree view with a single node labeled 'orderDest'. The main panel contains the following fields:

Name*:	orderDest
Description:	
Default Event*:	/Events/OrderEvent
Serializer/Deserializer:	com.tibco.cep.driver.http.serializer.RESTMessageSerializer
Is JSON Payload:	<input type="checkbox"/>
Include Event Type:	When Serializing and Deserializing
Type:	Default
Context Path:	
Action RuleFunction:	

Create Payments Channel



Build
Orchestrator

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter '**Payment**' in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on Payments Channel



Build
Orchestrator

We need 3 destinations for the **Payment** Channel.

Each destination will serve to send or receive asynchronous messages to the Payment microservice over TIBCO Cloud Messaging:

- **ExecutePayment** - used to send payment request.
- **ExecuteRefund** - used to send refund of payment request.
- **PaymentExecuted** - used to receive acknowledgement of payment.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
ExecutePayment	/Events/PaymentEvent	payment		Standard Durable	Yes
ExecuteRefund	/Events/RefundEvent	refund		Standard Durable	Yes
PaymentExecuted	/Events/PaymentExecutedEvent	paymentexecuted		Standard Durable	Yes

Create Destinations on Payments Channel



Build
Orchestrator

Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 3 destinations.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
ExecutePayment	/Events/PaymentEvent	payment		Standard Durable	Yes
ExecuteRefund	/Events/RefundEvent	refund		Standard Durable	Yes
PaymentExecuted	/Events/PaymentExecutedEvent	paymentexecuted		Standard Durable	Yes

The screenshot shows the TIBCO Studio interface. On the left, the 'Destinations' panel lists three items: 'ExecutePayment', 'ExecuteRefund', and 'PaymentExecuted'. The 'PaymentExecuted' item is selected. To the right, a detailed configuration dialog is open for this destination. The fields are as follows:

- Name*: PaymentExecuted
- Description: /Events/PaymentExecutedEvent
- Default Event*: /Events/PaymentExecutedEvent
- Serializer/Deserializer: com.tibco.cep.driver.tcm.serializer.TCMSerializer
- TCM Destination: paymentexecuted
- Client Id:
- Durable Name: paymentexecuted
- Durable Subscription Type: Shared Durable
- Is JSON Payload:

Below this, there is a preview section for 'Cloud servers' and a 'Channel' section. The 'Channel' section shows a single entry with the same values as the main dialog. At the bottom, there are tabs for 'Console', 'Error Log', 'Properties', 'Rule Input', 'Javadoc', and 'Declaration', with 'Console' being the active tab. A status message at the bottom says '0 items selected'.

Create Stock Channel



Build
Orchestrator

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter '**Stock**' in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on Stock Channel



Build
Orchestrator

We need 2 destinations for the **Stock Channel**.

Each destination will serve to send or receive asynchronous messages to the Stock microservice over TIBCO Cloud Messaging:

- **PrepareOrder** - used to send order preparation request.
- **OrderPrepared** - used to receive acknowledgement that the order has been prepared.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
PrepareOrder	/Events/PrepareOrderEvent	prepare		Standard Durable	Yes
OrderPrepared	/Events/OrderPreparedEvent	prepared		Standard Durable	Yes

Create Destinations on Stock Channel



Build
Orchestrator

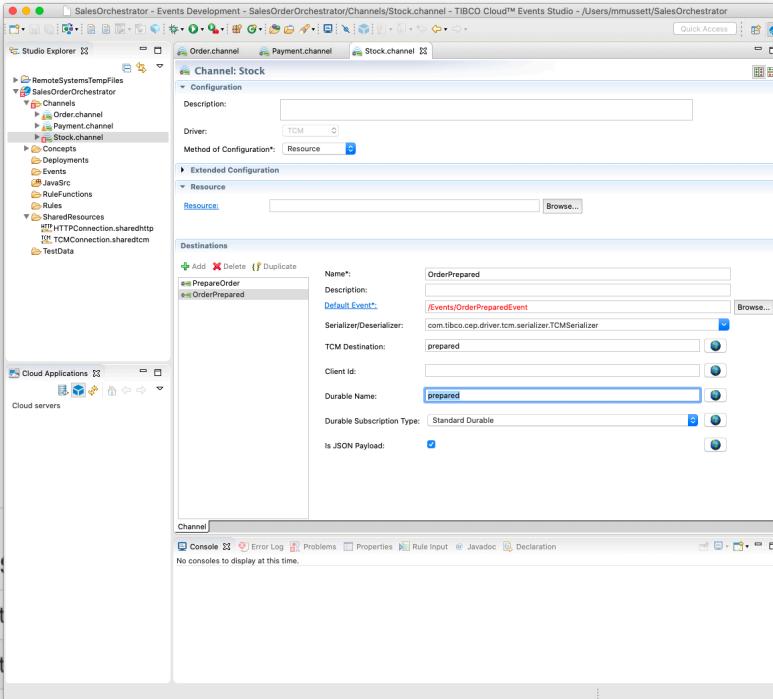
Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 2 destinations.

Name	Default Event	TCM Destination	Durable Name	Subscription Type
PrepareOrder	/Events/PrepareOrderEvent	prepare		Standard Durable
OrderPrepared	/Events/OrderPreparedEvent	prepared		Standard Durable



The screenshot shows the TIBCO Cloud Events Studio interface for configuring a Stock channel. The main window displays the 'Stock.channel' configuration under the 'SalesOrderOrchestrator' project. The 'Destinations' panel on the left lists two entries: 'PrepareOrder' and 'OrderPrepared'. The 'OrderPrepared' entry is currently selected, and its detailed configuration is shown in the central pane. The configuration includes fields for Name, Default Event, Serializer/Deserializer, TCM Destination, Client ID, Durable Name, Durable Subscription Type, and Is JSON Payload. The 'Resource' section also contains a 'Resource' input field with a 'Browse...' button. The 'Configuration' section includes fields for Description, Driver (set to TCM), Method of Configuration (set to Resource), and Extended Configuration. The bottom of the dialog shows tabs for 'Console', 'Error Log', 'Problems', 'Properties', 'Rule Input', 'Javadoc', and 'Declaration', with the 'Console' tab selected. A message at the bottom states 'No consoles to display at this time.' To the right of the dialog, there is a vertical toolbar with several icons and a 'Browse...' button, and below it, a list of five global variables with their current values.

Create OrderDelivery Channel



Build
Orchestrator

Right click in the Studio Explorer view, select New->Channel...

A dialog box will appear for you to create a new channel. Enter '**OrderDelivery**' in the Channel name field, select 'TCM' as the Driver Type and click 'Finish' button.

Hit the Resource 'Browse...' button, a Select Resource dialog will be shown.

Select the TCMConnection Shared Resource, then click 'OK'.

Click 'Save' icon on the toolbar.

Create Destinations on OrderDelivery Channel



Build
Orchestrator

We need 2 destinations for the **OrderDelivery** Channel.

Each destination will serve to send or receive asynchronous messages to the Delivery microservice over TIBCO Cloud Messaging:

- **DeliverOrder** - used to send order delivery request.
- **OrderDelivered** - used to receive acknowledgement that the order has been delivered.

Name	Default Event	TCM Destination	Durable Name	Durable Subscription Type	Is JSON Payload
DeliverOrder	/Events/DeliverOrderEvent	deliver		Standard Durable	Yes
OrderDelivered	/Events/OrderDeliveredEvent	delivered		Standard Durable	Yes

Create Destinations on OrderDelivery Channel



Build
Orchestrator

Click '+' in the Destinations panel

Configure each field with the correct row information.

Click 'Save' icon on the toolbar.

Repeat the above steps using the information provided in the table to create the 2 destinations.

Name	Default Event	TCM Destination	Durable Name	Start Date
DeliverOrder	/Events/DeliverOrderEvent	deliver		Start
OrderDelivered	/Events/OrderDeliveredEvent	delivered		Start

The screenshot shows the TIBCO Cloud Events Studio interface. On the left, the 'Destinations' panel lists two entries: 'DeliverOrder' and 'OrderDelivered'. On the right, the 'Channel: OrderDelivery' configuration pane is open. The 'Resource' field is set to '/SharedResources/TMCConnection.sharedtmc'. The 'Destinations' section shows the same two entries as the Destinations panel. The 'Driver' is set to 'TCM' and the 'Method of Configuration' is 'Resource'. The 'Resource' dropdown also points to '/SharedResources/TMCConnection.sharedtmc'. Other configuration fields include 'Default_Event' (set to '/Events/DeliverOrderEvent'), 'Serializer/Deserializer' (set to 'com.tibco.ceo.driver.tcm.serializer.TCMSerializer'), 'TCM Destination' (set to 'deliver'), 'Client Id', 'Durable Name', 'Durable Subscription Type' (set to 'Standard Durable'), and 'Is JSON Payload'. The status bar at the bottom indicates 'SalesOrderOrchestrator/Channels/OrderDelivery channel'.

Create Events



Build
Orchestrator

An event is an object representing some occurrence or point of time. In our use-case events represent business activities such as Take Order Payment or Deliver Order.

When messages arrive on channel destinations we assign these messages to events.

Channel destinations either emit or consume events from external participants.

Events model state similar to concept properties.

Create Events



Build
Orchestrator

For our use-case we are going to create 9 events which represent the following interactions:

- **OrderEvent** - A new Order arriving.
- **ReplyOrderEvent** - Acknowledgement for a new Order.
- **PaymentEvent** - A payment for an Order is needed.
- **RefundEvent** - A refund on an Order is needed.
- **PaymentExecuteEvent** - Payment confirmation for an Order.
- **PrepareOrderEvent** - Prepare Order for shipment.
- **OrderedPreparedEvent** - An Order is ready for shipment.
- **DeliverOrderEvent** - Dispatch Order to the customer.
- **OrderDeliveredEvent** - An Order has been delivered to the customer.

Create Order Event



Build
Orchestrator

Right click in the Studio Explorer view, select New->Simple Event...

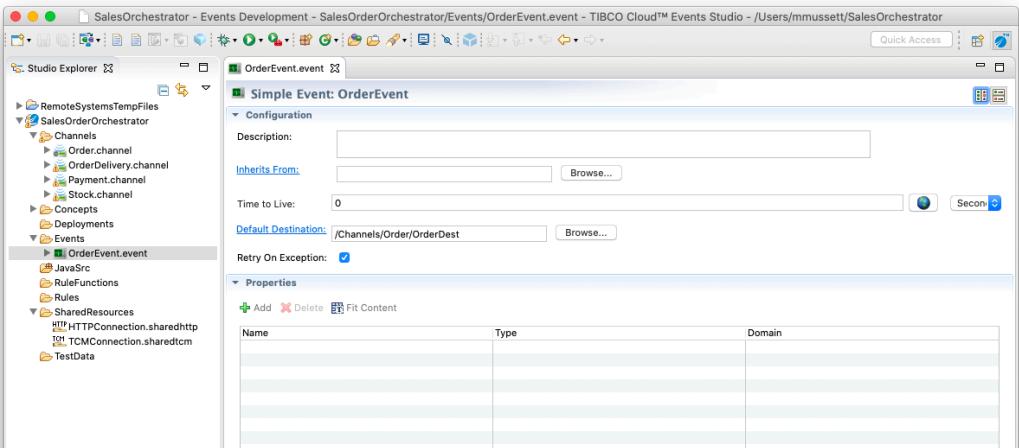
A dialog box will appear for you to create a new Simple Event.

Select 'Events' folder. Enter 'OrderEvent' in the Simple Event name field.

Click 'Finish' button.

Click 'Browse...' button for Default Destination.

A dialog box will appear for you to select a Channel Destination. Select the 'OrderDest' destination on the 'Order' channel and click 'OK' button.



Create Order Event



Build
Orchestrator

Click the 'Advanced' tab on the Properties section.

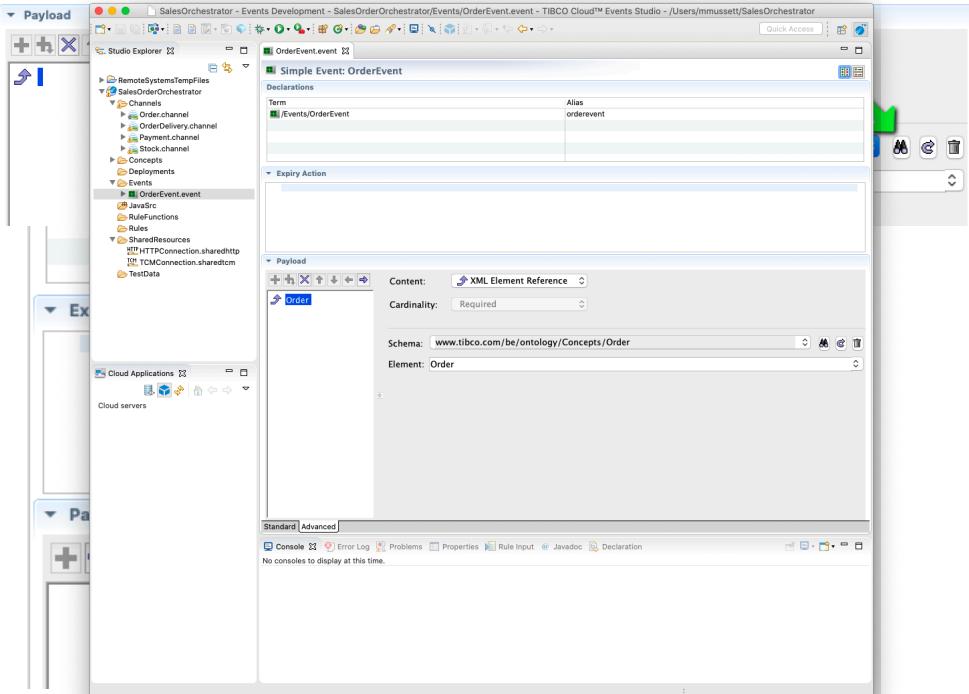
Click the '+' symbol in the 'Payload' section...

Click the 'Content' dropdown arrow and select 'XML Element Reference'...

Click the binocular icon next to the 'Schema' box...

'Select a Resource...' dialog box will be shown.
Navigate the Concepts folder and select the 'Order' concept.
Click 'OK'.

Click 'Save' icon on the toolbar.



Create ReplyOrder Event



Build
Orchestrator

Right click in the Studio Explorer view, select New->Simple Event...

A dialog box will appear for you to create a new Simple Event.

Select 'Events' folder. Enter '**ReplyOrderEvent**' in the Simple Event name field.

Click 'Finish' button.

Click 'Browse...' button for Default Destination.

A dialog box will appear for you to select a Channel Destination. Select the '**OrderDest**' destination on the '**Order**' channel and click 'OK' button.

Create ReplyOrder Event



Build
Orchestrator

Click the 'Advanced' tab on the Properties section.

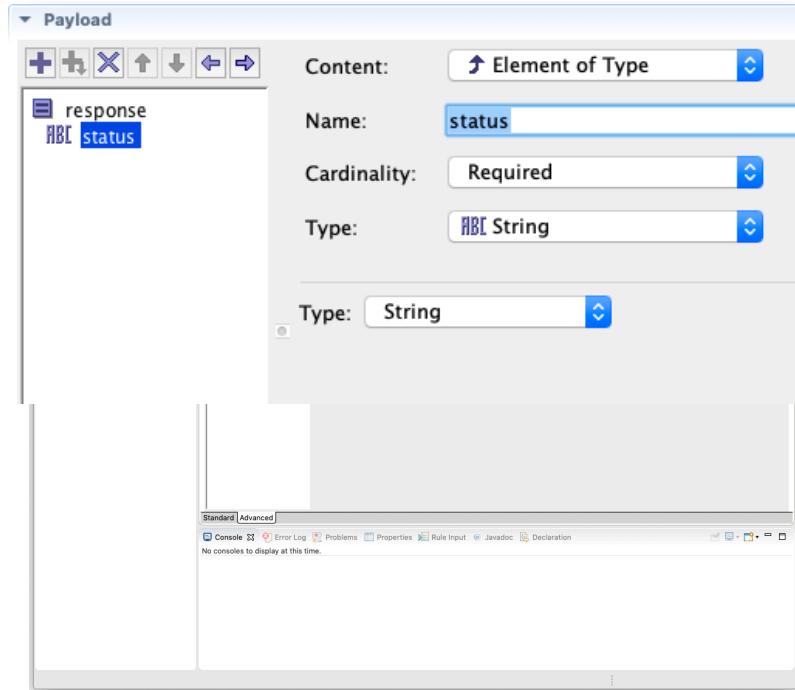
Click the '+' symbol in the 'Payload' section.

Click the '+' symbol again.

Select 'root' Complex Element Content and change the Name field from 'root' to 'response'.

Select 'param' Element and change the Name field from 'param' to 'status'.

Click 'Save' icon on the toolbar.



Create Payment Event



Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**PaymentEvent**'.

Set the Default Destination set to '**Payment**' channel and '**ExecutePayment**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- email - Content=Element of Type String
- amount - Content=Element of Type String

The screenshot shows the 'Payload' configuration screen. On the left, there is a tree view under the 'Payload' heading with the following structure: payload > order_id > customer_id > email > amount. To the right of the tree view are four configuration panels:

- Content:** Element of Type
- Name:** order_id
- Cardinality:** Required
- Type:** String

Below these four panels, there is another panel labeled 'Type:' with a dropdown menu set to 'String'.

Create PaymentExecuted Event



Build
Orchestrator

Create the Simple Event named '**PaymentExecutedEvent**'.

Set the Default Destination set to '**Payment**' channel and '**PaymentExecuted**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String

The screenshot shows a configuration interface for creating a payload structure. On the left, under the 'Payload' section, there is a toolbar with icons for adding (+), editing (blue +), deleting (X), moving up (up arrow), moving down (down arrow), and a right-pointing arrow. Below the toolbar is a list of elements:

- payload** (highlighted in blue)
- ABC order_id
- ABC customer_id
- ABC status

On the right, there are configuration fields:

- Content:** Complex Element
- Name:** payload
- Cardinality:** Required

Create Refund Event



Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**RefundEvent**'.

Set the Default Destination set to '**Payment**' channel and '**ExecuteRefund**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- email - Content=Element of Type String
- amount - Content=Element of Type String

The screenshot shows the 'Payload' configuration screen. On the left, there's a tree view under 'Payload' with nodes: payload, order_id, customer_id, email, and amount. To the right, there are four configuration panels corresponding to these elements:

- Content:** Element of Type
- Name:** order_id
- Cardinality:** Required
- Type:** String

Below these, there is another panel for 'Type' with a dropdown set to String.

Create Prepare Order Event



Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**PrepareOrderEvent**'.

Set the Default Destination set to '**Stock**' channel and '**PrepareOrder**' destination.

On the 'Advanced' tab, create the Properties structure with a XML Element Reference to the 'Order' Concept.

The screenshot shows the 'Payload' configuration for a Simple Event. The 'Content' is set to 'XML Element Reference'. The 'Cardinality' is 'Required'. The 'Schema' is 'www.tibco.com/be/ontology/Concepts/Order' and the 'Element' is 'Order'. A tree view on the left shows a single node labeled 'Order'.

Create OrderPrepared Event

2

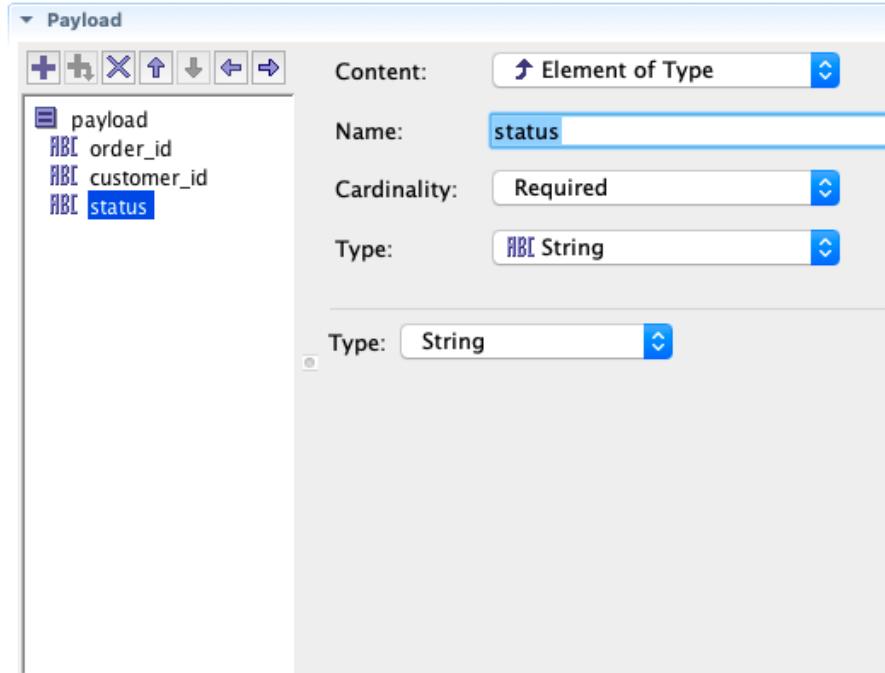
Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**OrderPreparedEvent**'.

Set the Default Destination set to '**Stock**' channel and '**OrderPrepared**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String



Create DeliverOrder Event



Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**'DeliverOrderEvent'**'.

Set the Default Destination set to '**'OrderDelivery'**' channel and '**'DeliverOrder'**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 4 child elements named:

- order_id - Element of Type=String
- customer_id - Element of Type=String
- email - Element of Type=String
- ShippingAddress - XML Element Reference=Concepts/ShippingAddress

The screenshot shows the 'Payload' configuration screen. On the left, there's a tree view of the payload structure. At the top level is 'payload'. Below it are four child elements: 'order_id', 'customer_id', 'email', and 'ShippingAddress'. To the right of the tree view are three configuration fields: 'Content' set to 'Complex Element', 'Name' set to 'payload', and 'Cardinality' set to 'Required'.

Create OrderedDelivered Event



Build
Orchestrator

Follow the same steps previously but create the Simple Event named '**OrderDeliveredEvent**'.

Set the Default Destination set to '**OrderDelivery**' channel and '**OrderDelivered**' destination.

On the 'Advanced' tab, create the Properties structure with a Complex element named 'payload' and 3 child elements named:

- order_id - Content=Element of Type String
- customer_id - Content=Element of Type String
- status - Content=Element of Type String

The screenshot shows the configuration interface for creating a payload structure. On the left, under the 'Payload' section, there is a tree view with a root node 'payload' expanded to show three child nodes: 'order_id', 'customer_id', and 'status'. To the right of the tree, there are four configuration fields:

- Content:** Set to 'Element of Type'.
- Name:** Set to 'status'.
- Cardinality:** Set to 'Required'.
- Type:** Set to 'String'.

Below these fields, there is another row with a 'Type:' dropdown set to 'String'.

Create Rules & Rule Functions



Build
Orchestrator

Rules are the logic behind the events.

Rules are executed whenever an Event is created or modified within TIBCO Cloud Events.

Rules allow us to codify our business logic using Java.

We use Rules and Events to determine the next action to take.

The TIBCO Cloud Events engine will determine which rules to execute based on predicated logic

e.g. Payment=APPROVED AND Order Status=PREPARED



Build
Orchestrator

Create OrderPreProcessor Rule Function

Right click in the Studio Explorer view, select New->Rule Function...

A dialog box will appear for you to create a new Rule Function.

Select 'RuleFunctions' folder.

Enter '**OrderPreProcessor**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderPreProcessor.rulefunction text file located in src/TCE/SalesOrderOrchestrator/RuleFunctions github repository.

Click 'Save' icon on the toolbar.

Create CalculateOrderAmount Rule Function



Build
Orchestrator

Right click in the Studio Explorer view, select New->Rule Function...

A dialog box will appear for you to create a new Rule Function. Select 'RuleFunctions' folder.

Enter '**CalculateOrderAmount**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the CalculateOrderAmount.rulefunction text file located in
src/TCE/SalesOrderOrchestrator/RuleFunctions github repository.

Click 'Save' icon on the toolbar.

Create Order Rule



Build
Orchestrator

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**OrderRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.

Create PaymentExecute Rule



Build
Orchestrator

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**PaymentExecutedRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the `PaymentExecutedRule.rule` text file located in `src/TCE/SalesOrderOrchestrator/Rule` github repository.

Click 'Save' icon on the toolbar.

Create OrderPrepared Rule



Build
Orchestrator

Right click in the Studio Explorer view, select New->Rule...

A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**PaymentExecutedRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderPreparedRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.

Create OrderDelivered Rule



Build
Orchestrator

Right click in the Studio Explorer view, select New->Rule...

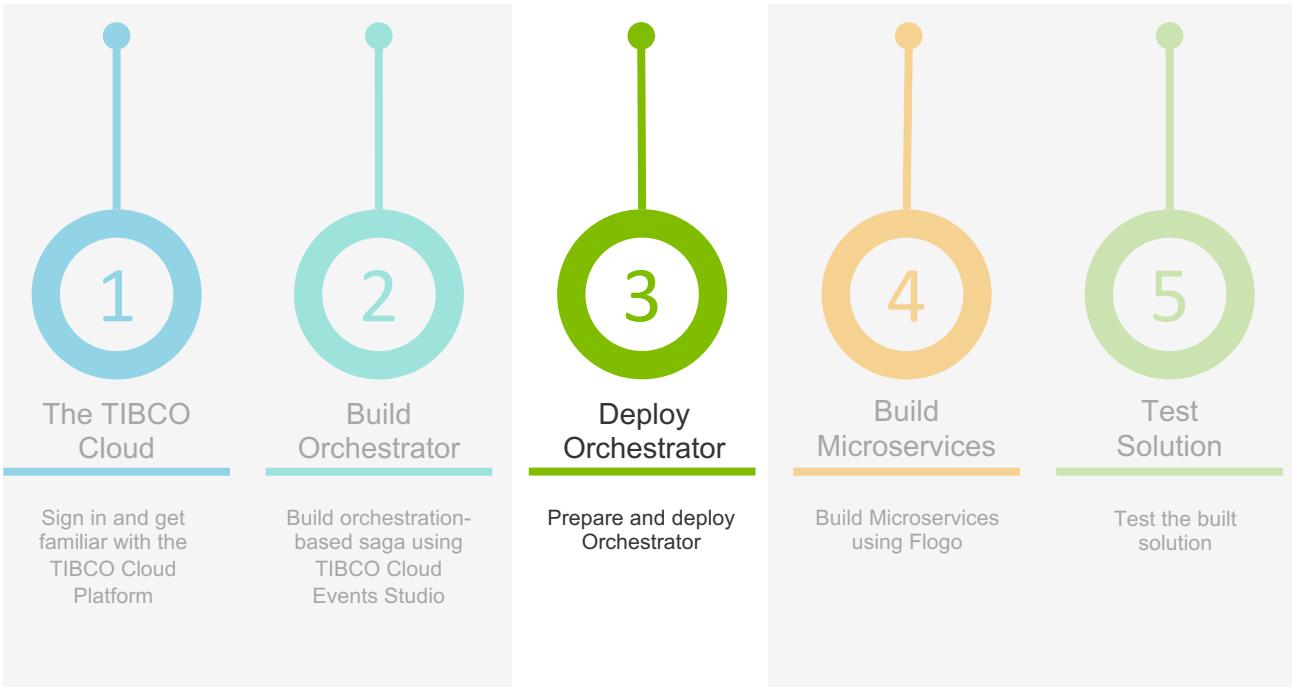
A dialog box will appear for you to create a new Rule. Select 'Rule' folder.

Enter '**OrderDeliveredRule**' in the Rule Function name field.

Click 'Finish' button.

Paste the contents of the OrderDeliveredRule.rule text file located in src/TCE/SalesOrderOrchestrator/Rule github repository.

Click 'Save' icon on the toolbar.



Import Cluster Deployment Descriptor



To run the TIBCO Cloud Events project we need to create a Cluster Deployment Descriptor file.

Import the file 'Default.cdd' located in src/TCE/SalesOrderOrchestrator folder of the github repository.

You can import directly in to the project by dragging and dropping the Default.cdd file using either your Mac Finder or Windows Explorer.

Eclipse will prompt you how it should import the files into the project, make sure to choose 'Copy Files'.

Import Swagger



To run the TIBCO Cloud Events project we need to create Swagger API file.

Import the file 'swagger.json' located in src/TCE/SalesOrderOrchestrator folder of the github repository.

You can import directly in to the project by dragging and dropping the swagger.json file using either your Mac Finder or Windows Explorer.

Eclipse will prompt you how it should import the files into the project, make sure to choose 'Copy Files'.

Build Enterprise Archive



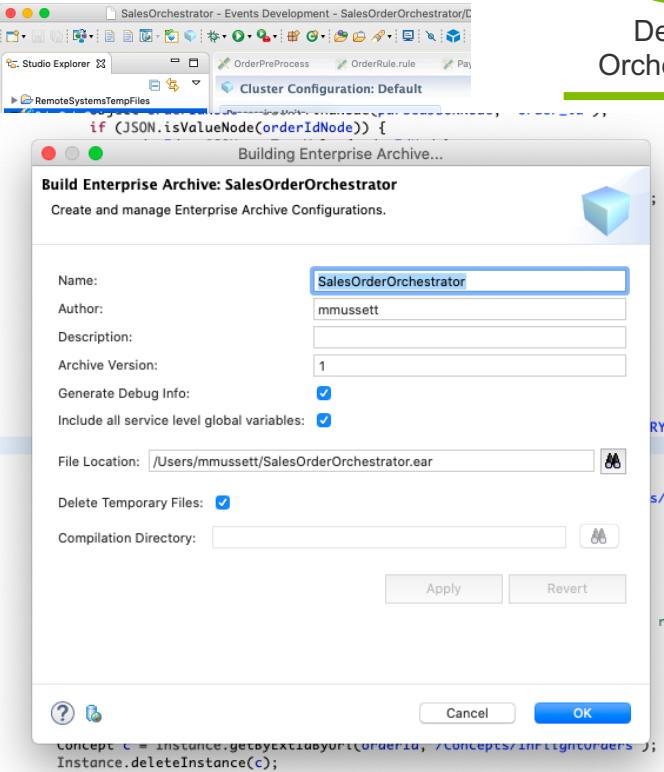
Deploy
Orchestrator

Right click the SalesOrderOrchestrator project in the Studio Explorer view.

Click 'Build Enterprise Archive...' menu item.

Set the File Location to where you wish the Enterprise Archive file to be created.

Click 'OK'



Deploy Enterprise Archive



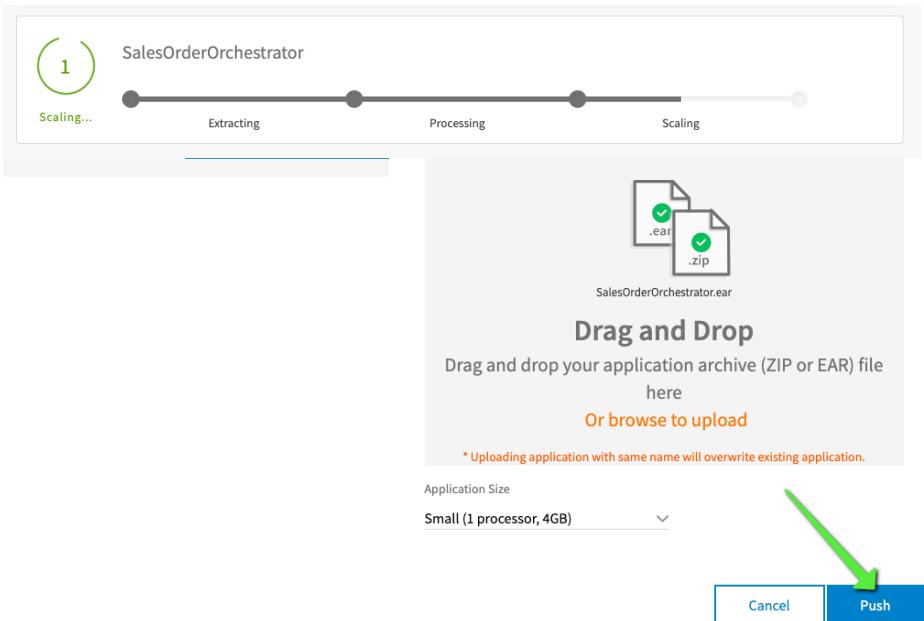
Open browser to <https://eu.events.cloud.tibco.com>

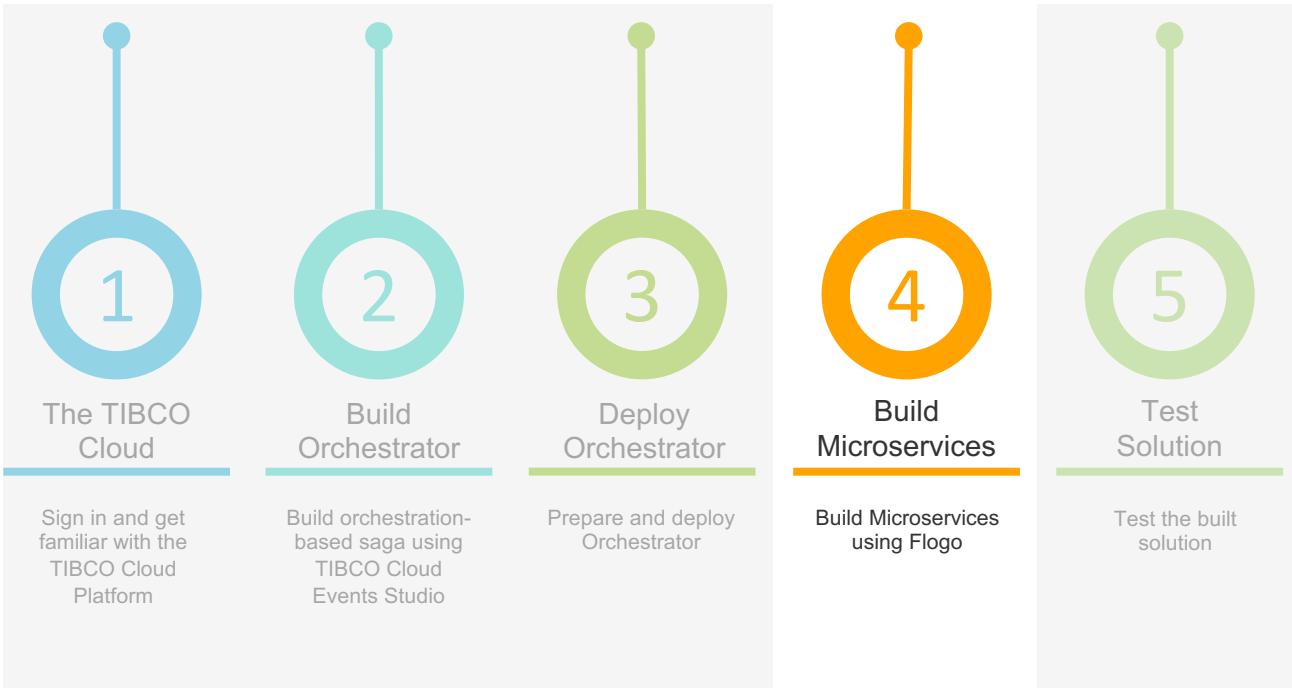
Click upload icon

Drag EAR file into dialog window

Click 'Push' button

The TCE Application will begin to deploy on to
TIBCO Cloud





TIBCO Cloud Messaging



Build
Microservices

Navigate to TIBCO Cloud Messaging Status Page [here](#)

Verify the service is running, if not start it.

Navigate to TIBCO Cloud Messaging Authentication Keys Page [here](#)

Make a note of the URL and pre-created authentication Key labelled 'default'

The screenshot shows the 'Authentication Keys' section of the TIBCO Cloud Messaging interface. At the top, there's a status bar with icons for location (Ireland), help, and user profile (Mark M). Below the status bar, the navigation menu includes 'Status', 'Download SDKs', and 'Authentication Keys' (which is underlined, indicating it's the active tab).

In the main content area, there's a placeholder text 'Add this URL in your app.' followed by a URL input field containing 'wss://01DXXG1Y0P5RPVB4ZYJH69D05B-apps.eu.messaging.cloud.tibco.com/channel'. Below this, a section titled 'Authentication Keys' displays a single key entry:

Label	Key	Date created	Date expires	Delete
default	4e0c145c583e8f09b9fe333f5fc58ba	Mon Jan 06 2020	never	

At the bottom of the key list, there are buttons for 'Label (optional)', 'Generate key', and a message stating 'You've generated 1 of 10 available keys.' Below the key list are three tabs: 'Current Connections' (highlighted in teal), 'Peak Connections' (highlighted in purple), and 'Total Messages' (highlighted in blue).

Create TIBCO Cloud Messaging Connector



Build
Microservices

Navigate to TIBCO Cloud Integration Connections page [here](#)

Enter 'TIBCO' in the search box and hit the magnifying glass symbol.

Click the 'TIBCO Cloud Messaging Connector'.

Paste the TIBCO Cloud Connection URL in to the Connection URL field.

Paste the TIBCO Cloud Authentication URL in to the Authentication Key field.

Click 'Save'

The screenshot shows the TIBCO Cloud Integration interface. On the left, there's a sidebar with a 'TIBCO CLOUD' logo and a search bar containing 'Add a connection...'. Below the search bar is a list item for 'TCM' with a small icon and the version '1.1.1'. On the right, a modal window titled 'TIBCO Cloud Messaging Connector' is open. It contains the following fields:

- Connection Name: TCM
- Description: (empty)
- Connection URL: `wss://01DXXG1Y0P5RPVB4ZYJH69D05B-apps.eu.messaging.cloud.tibco.com/channel`
- Authentication Key: (redacted)
- Timeout (in seconds): 2
- AutoReconnectAttempts: 5
- AutoReconnectMaxDelay (in seconds): (empty)

At the bottom right of the modal are 'Cancel' and 'Save' buttons.

Create Payments Microservice



Build
Microservices

Navigate to TCI Flogo Apps Page [here](#)

Click 'Create' button.

Enter 'PaymentService' for the app name.

Click 'Create' button.

Click 'Create a TIBCO Flogo App'.

Click '+ Create' to create Flogo Flow.

Enter 'ExecutePaymentFlow' for the Flow Name and click 'Create' button.

Create flow

Start from scratch

New flow

Start with

Swagger Specification

GraphQL Schema

Name *

ExecutePaymentFlow

Description

ex: Run this at the end of each pay period

Cancel Create

Create Payments Microservice



Build
Microservices

We're going to create a trigger that looks like this:

Messaging/tibco-messaging-tcm-trigger
MessageSubscriber

This trigger receives a message from TIBCO Cloud M...

Trigger Settings

Handler Settings ▾

Connection: MM-TCE

Durable Subscriber: False

Destination: payment

Content Matcher: Add row

Name	Type	Value	Actions
No Rows To Show			

Output Settings

Map to Flow Inputs

Sync X

Discard **Save**

Messaging/tibco-messaging-tcm-trigger
MessageSubscriber

This trigger receives a message from TIBCO Cloud M...

Trigger Settings

Message Schema ▾

```
1 - {  
2   "payload" : ""  
3 }
```

Output Settings

Map to Flow Inputs

Sync X

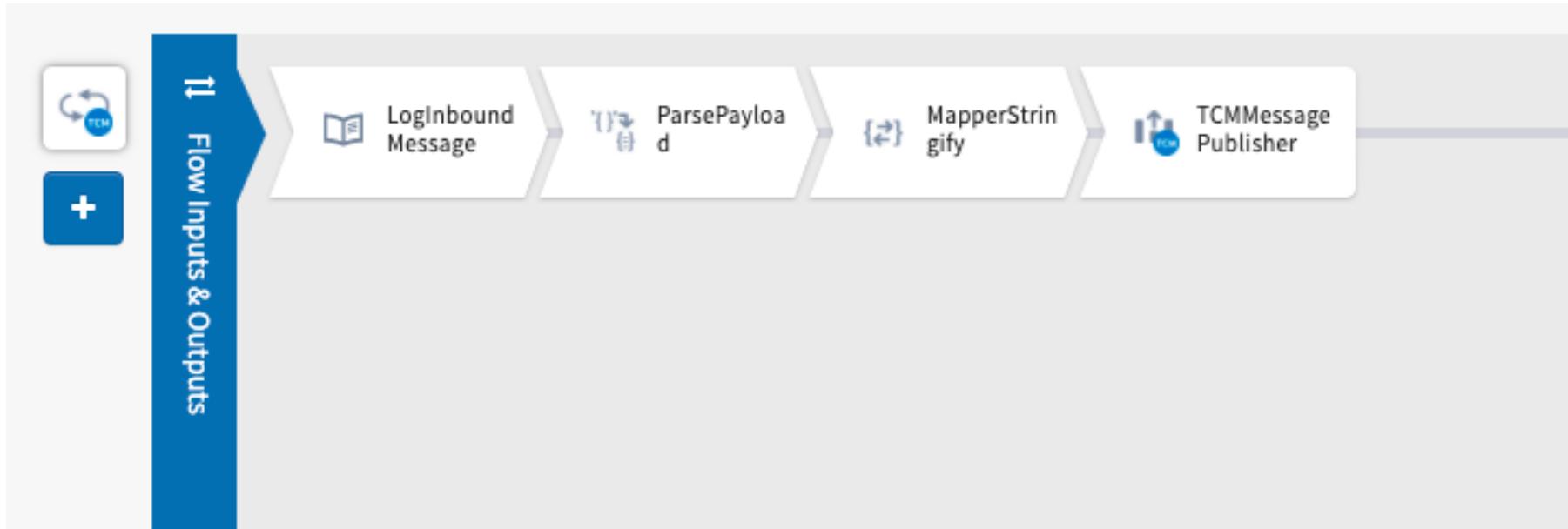
Discard **Save**

Create Payments Microservice

4

Build
Microservices

We're going to create a flow that looks like this:



Create Payments Microservice - Trigger

4

Build
Microservices

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

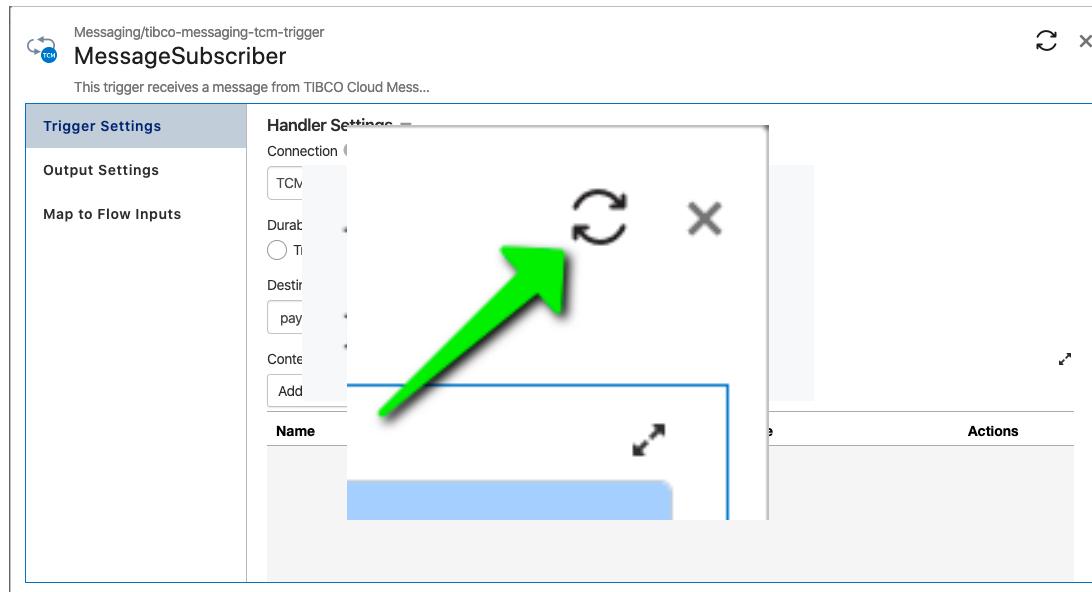
Click 'Just add the trigger' button

Click the TCM trigger symbol

Enter 'payment' into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON in to the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box



Create Payments Microservice - Trigger

4

Build
Microservices

Click 'Map to Flow Inputs' in LHS panel.

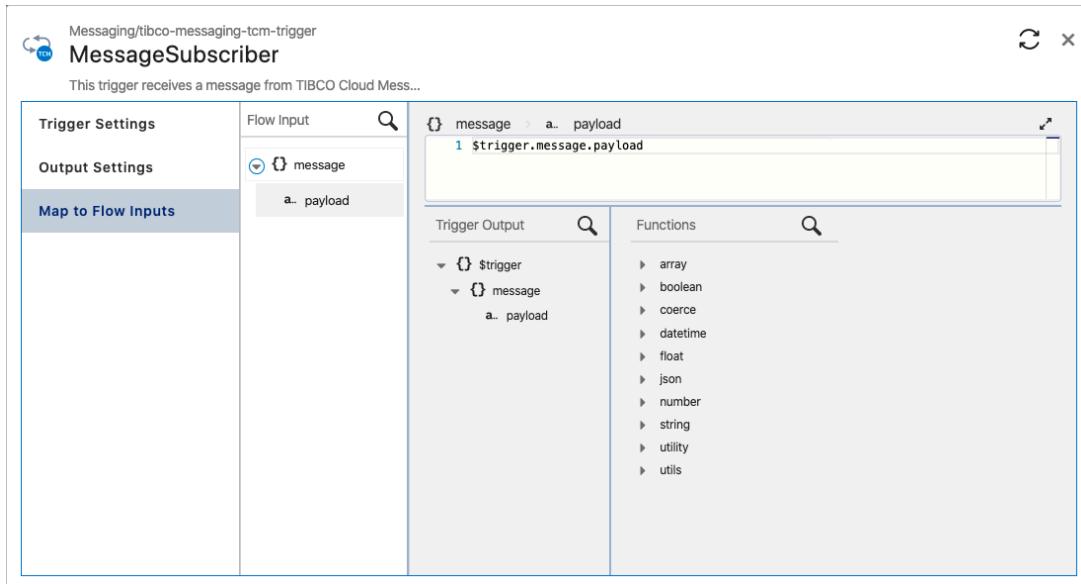
We need to map the Flow Input to the Trigger Output.

In the Flow Input pane navigate using the arrows to the payload field (message->payload).

In The Trigger Output panel navigate using the arrows to the payload field (\$trigger->message->payload).

Click the 'payload' field.

Close the Message Subscriber dialog box.



Create Payments Microservice - Flow



Build
Microservices

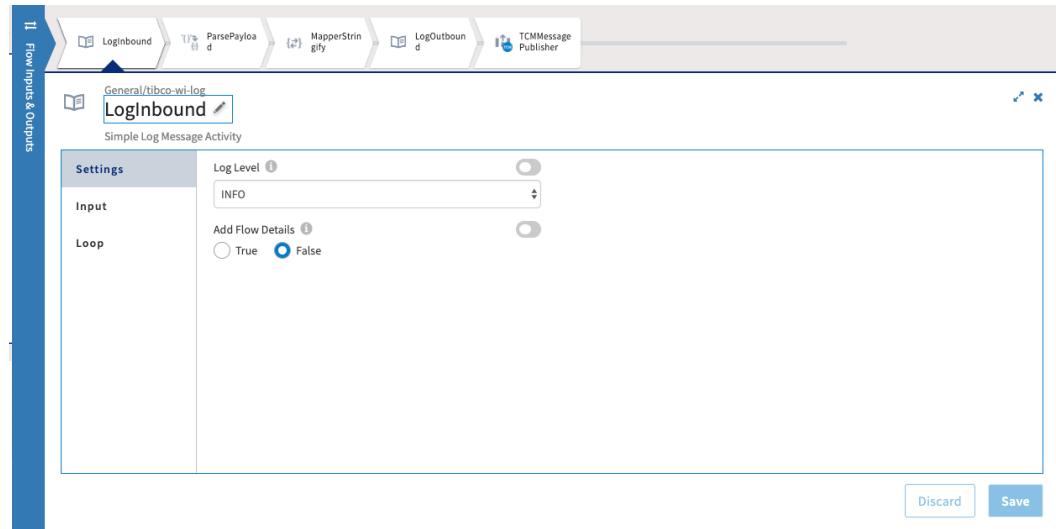
Hover over the Activity Line, and click the first free box to add activity. The Add Activity dialog will be shown.

In the LHS panel, scroll to 'General' and click. The activities panel will show you the general activities available to place at this step. Go ahead and click 'Log Message'

Click the 'LogMessage' activity title field and rename to 'LogInbound'

In the LHS panel of the LogMessage activity click 'Input' then click the 'a... * message' in the Activity input panel and enter the following in the message field:

Close the activity.



Create Payments Microservice - Flow



Build
Microservices

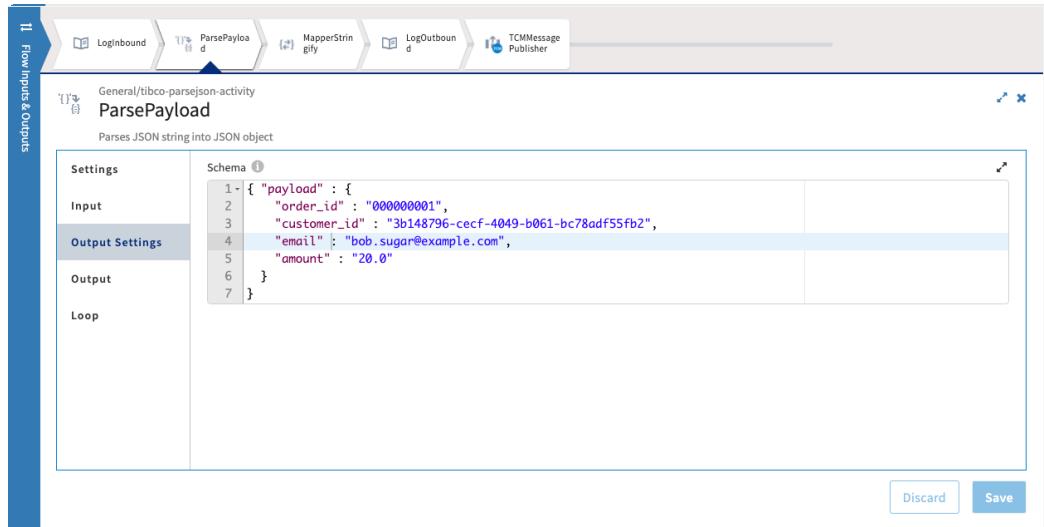
Add a Parse JSON Activity after the previous Log activity.

Click the 'ParseJSONActivity' activity title field and rename to 'ParsePayload'.

In the Input panel, map the jsonString of the Activity Input to '\$flow.message.payload'.

Click 'Output Settings' in LHS panel and enter the following JSON in to the Schema:

Close the activity.



Create Payments Microservice - Flow



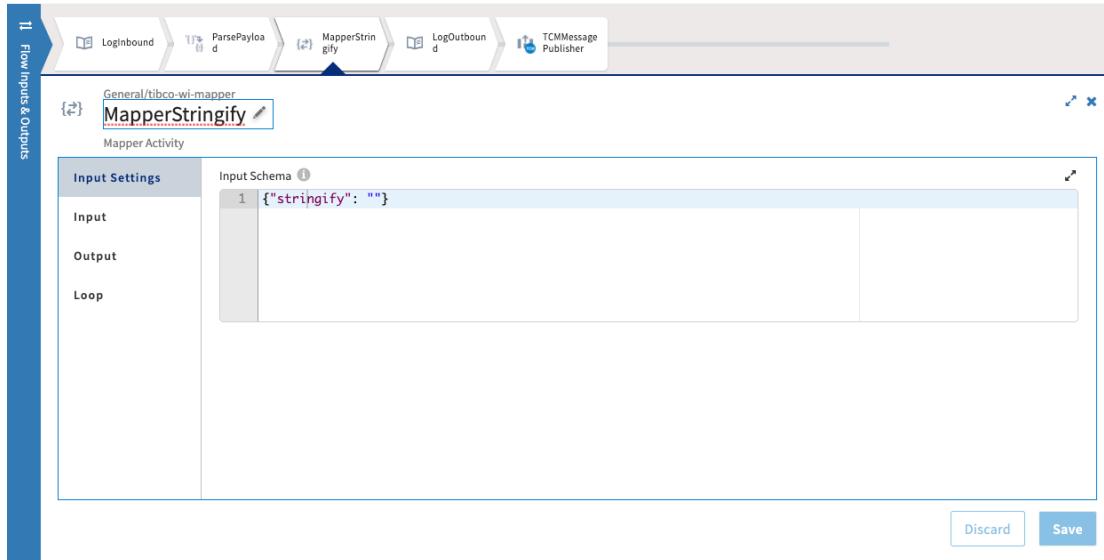
Build
Microservices

Add a Mapper Activity after the previous Parse JSON activity.

Click the 'Mapper' activity title field and rename to 'MapperStringify'.

In the 'Input Settings' enter the following JSON in to the Input Schema:

```
{"stringify":""}
```



Create Payments Microservice - Flow

4

Build
Microservices

In the Input panel, enter the follow expression in to the stringify field:



```
string.trim(string.concat("{ \"payload\" : { \"order_id\" :  
\"\",$activity[ParsePayload].jsonObject.payload.order_id,\"\", \"customer_id\" :  
\"\",$activity[ParsePayload].jsonObject.payload.customer_id,\"\", \"status\" : \"SUCCESS\"  
}}))
```

Close the activity.

Discard Save

Create Payments Microservice - Flow

4

Build
Microservices

Add a Logger Activity after the previous Mapper activity.

Click the 'Logger' activity title field and rename to 'LoggerOutbound'.

In the 'Input' enter the following

Close the Logger activity.

The screenshot shows a TIBCO Business Studio interface with a flow editor. At the top, there's a horizontal sequence of activities: LogInbound, ParsePayload, MapperStringify, LogOutbound, and TCMessagePublisher. Below this, a specific activity is selected: 'General/tibco-wi-log LogOutbound'. This is a 'Simple Log Message Activity'. The configuration dialog for this activity is open. In the 'Activity Input' section, the 'Input' tab is selected, showing 'a.. * message'. In the 'Upstream Output' section, there's a search bar with 'a.. message' and a value entry field containing '\$activity[MapperStringify].output.stringify'. To the right of the input fields is a 'Functions' dropdown menu listing various utility functions such as array, boolean, coerce, datetime, float, json, number, string, utility, and utils. At the bottom right of the dialog are 'Discard' and 'Save' buttons.

Create Payments Microservice - Flow



Build
Microservices

Add a TCM Message Publisher Activity after the previous Mapper activity.

In Settings panel, select the TCM Connection.

In the 'Input Settings' enter the following JSON in to the Input Schema

In the 'Input' enter the following in to the 'destination' field.

In the 'Input' enter the following in to the 'message payload' field.

Close the activity.

The screenshot shows the TIBCO Flow Designer interface. At the top, a flow diagram is displayed with several activities: Loginbound, TCMMessagePublisher (selected), ParsePayload, MapperStringify, and LogOutbound. Below the diagram, the activity details for 'TCMMessengerPublisher' are shown. The 'Settings' tab is selected. In the 'Activity Input' section, the 'message' input is selected, and the 'payload' field contains the JSON path '\$activity[MapperStringify].output.stringify'. The 'Upstream Output' section lists 'MapperStringify', 'ParsePayload', and '\$flow'. The 'Functions' section provides a list of available functions: array, boolean, coerce, datetime, float, json, number, string, utility, and utils. At the bottom right, there are 'Discard' and 'Save' buttons.

Create Payments Microservice



Build
Microservices

Map the 'stringify' field from the MapperStringify activity to the Activity Input of the message payload

Messaging/tibco-messaging-tcm-pub

TCMMessagePublisher

This activity sends a message to TIBCO Cloud Messaging se...

Settings	Activity Input	
Input Settings	a.. destination	
Input	message	message > a.. payload 1 \$activity[MapperStringify].output.stringify
Loop		
Retry on Error	a.. payload	

Upstream Output

- MapperStringify
- output
- a.. stringify
- ParsePayload
- \$flow

Functions

- array
- boolean
- coerce
- datetime
- float
- json
- number
- string
- utility
- utils

Create Payments Microservice



Build
Microservices

Close the Message Publisher activity.

Click 'Push App'.

TIBCO CLOUD™ Integration

EMEA North PreSales Ireland Mark M

Apps API Specs Connections Extensions VPN Connections Downloads

PaymentsService ExecutePaymentFlow Add flow's description

Start testing Revert to last push Push app

Flow Inputs & Outputs

```
graph LR; A[Loginbound] --> B[ParsePayload]; B --> C[MapperStringify]; C --> D[LogOutbound]; D --> E[TCMMessage Publisher]
```

Create Stock Microservice

4

Build
Microservices

We're going to create a trigger that looks like this:

Messaging/tibco-messaging-tcm-trigger

MessageSubscriber

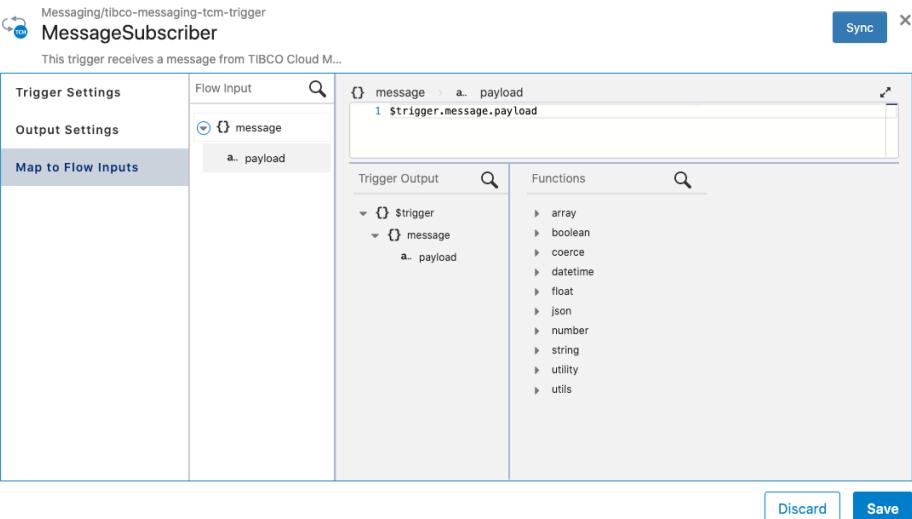
This trigger receives a message from TIBCO Cloud M...

Sync X

Trigger Settings	Flow Input <input type="text"/> message	Trigger Output <input type="text"/> \$trigger.message.payload
Output Settings	<input checked="" type="checkbox"/> message	<input type="checkbox"/> array
Map to Flow Inputs	a. payload	<input type="checkbox"/> boolean
		<input type="checkbox"/> coerce
		<input type="checkbox"/> datetime
		<input type="checkbox"/> float
		<input type="checkbox"/> json
		<input type="checkbox"/> number
		<input type="checkbox"/> string
		<input type="checkbox"/> utility
		<input type="checkbox"/> utils

Functions

Discard Save

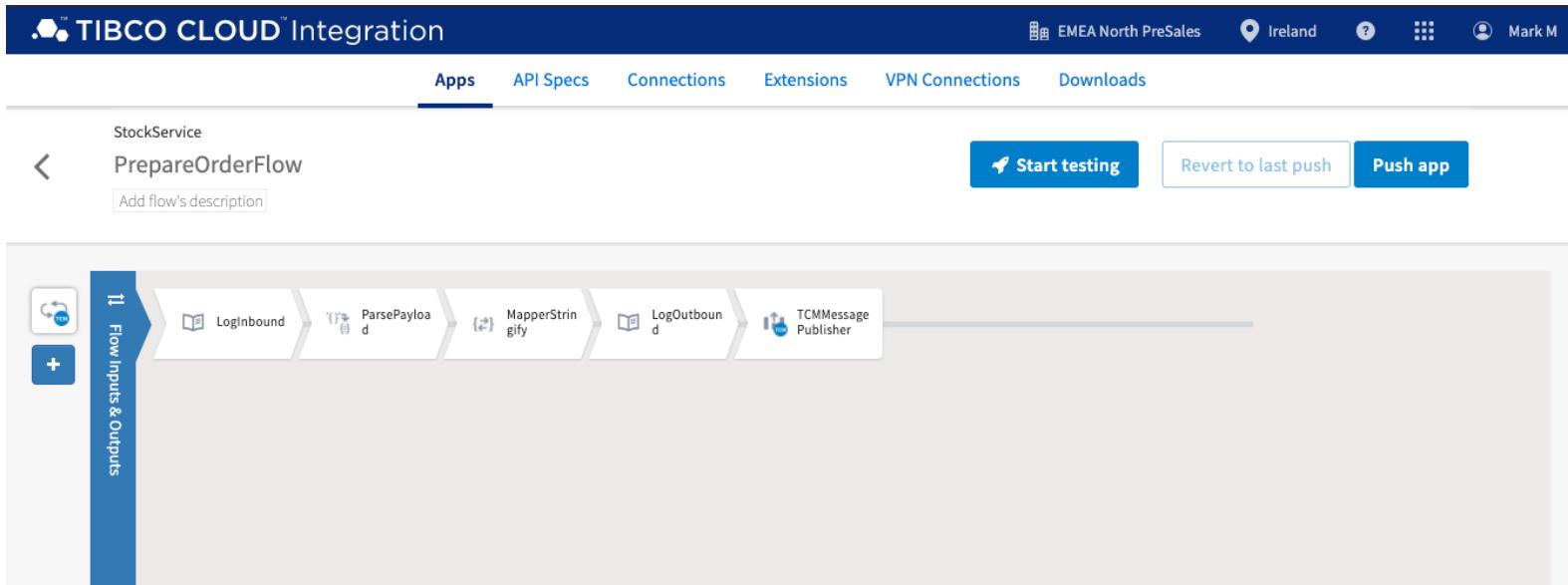


Create Stock Microservice

4

Build
Microservices

We're going to create a flow that looks like this:



Create Stock Microservice - Trigger



Build
Microservices

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

Click 'Just add the trigger' button

Click the TCM trigger symbol

Enter 'prepare' into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON into the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box

Close the Trigger

The screenshot shows the configuration of a 'MessageSubscriber' trigger. The title bar indicates it's a 'Messaging/tibco-messaging-tcm-trigger'. The main area has tabs for 'Trigger Settings' (selected), 'Output Settings' (highlighted in blue), and 'Map to Flow Inputs'. The 'Output Settings' tab contains a 'Message Schema' section with the following JSON:

```
1 - {  
2   "payload": ""  
3 }
```

At the bottom right of the dialog box are 'Discard' and 'Save' buttons.

Create Stock Microservice - Flow



Build
Microservices

Add Log Activity and configure

The screenshot shows a flow diagram with five steps: Loginbound, ParsePayload, MapperStringify, LogOutbound, and TCMessage Publisher. The 'Log inbound' activity is selected, and its configuration pane is displayed.

Activity Input:

- Input:** a. * message
- Upstream Output:** a. payload

Functions:

- \$flow
- message
- payload
- array
- boolean
- coerce
- datetime
- float
- json
- number
- string
- utility
- utils

Buttons:

- Discard
- Save

Create Stock Microservice - Flow



Build
Microservices

Add Parse Activity and configure

Schema ⓘ

```
1 {  
2   "Order" : {  
3     "order_id" : "000000107",  
4     "customer_id" : "abc123",  
5     "email" : "bob.sugar@example.com",  
6     "items" : [ {  
7       "attributes" : {  
8         "Id" : "75",  
9         "type" : "www.tibco.com/be/ontology/Concepts/OrderItem"  
10      },  
11      "item_name" : "Gummy Bears",  
12      "price" : "2.0E0",  
13      "quantity" : "10"  
14    } ]  
15  }  
16 }
```

Create Stock Microservice - Flow



Build
Microservices

Add Mapper Activity and configure

The screenshot shows the TIBCO Business Studio interface with a flow editor. On the left, a vertical toolbar has 'Flow' selected. The main area displays a flow with five steps: Loginbound, ParsePayload, MapperStringify (highlighted with a blue arrow), LogOutbound, and TCMessage Publisher. The 'MapperStringify' step is expanded, showing its configuration details.

MapperActivity
General/tibco-wi-mapper
MapperStringify

Input Settings

	Activity Input
<input type="radio"/> Input	a.. stringify
<input type="radio"/> Output	
<input type="radio"/> Loop	

Activity Input

input > a.. stringify

```
1 string.trim(string.concat("{ \"payload\" : { \"order_id\" : \"", $activity[ParsePayload]
 .jsonObject.Order.order_id, "customer_id\" : \"", $activity[ParsePayload]
 .jsonObject.Order.customer_id, "\", \"status\" : \"SUCCESS\" }}"))
```

Upstream Output

ParsePayload
\$flow

Functions

- array
- boolean
- coerce
- datetime
- float
- json
- number
- string

Buttons

Discard Save

Create Stock Microservice - Flow



Build
Microservices

Add Log Activity and configure

The screenshot shows a flow editor with a vertical toolbar on the left labeled "Flow Inputs & Outputs" containing the number "11". The main area displays a sequence of five activities: Loginbound, ParsePayload, MapperStringify, LogOutbound, and TCMMessage Publisher. The "LogOutbound" activity is currently selected, highlighted by a blue border.

The configuration pane for the "LogOutbound" activity is open, titled "General/tibco-wi-log LogOutbound Simple Log Message Activity". It has three tabs: "Settings" (selected), "Input", and "Loop".

In the "Settings" tab, there is a "Activity Input" section with a search bar and a dropdown menu showing "a.. message". Below it is an "Upstream Output" section with a search bar and a dropdown menu showing "a.. messageify", "output", "stringify", "ParsePayload", and "\$flow".

In the "Input" tab, there is a single input field labeled "a.. message" containing the expression "\$activity[MapperStringify].output.stringify".

In the "Loop" tab, there is no visible content.

On the right side of the configuration pane, there is a "Functions" list with a search bar and a scrollable list of functions:

- array
- boolean
- coerce
- datetime
- float
- json
- number
- string
- utility
- utils

At the bottom right of the configuration pane are two buttons: "Discard" and "Save".

Create Stock Microservice - Flow



Build
Microservices

Add a TCM Message Publisher Activity.

Save and Push Application.

The screenshot shows the TIBCO Business Studio interface for creating a flow. On the left, there's a sidebar titled "Flow Inputs & Outputs". The main area displays a flow diagram with several activities: "Loginbound", "ParsePayload", "MapperStringify", "LogOutbound", and "TCMMessengerPublisher". Below the flow diagram, the "TCMMessengerPublisher" activity is selected, and its configuration pane is open. The pane has tabs for "Settings", "Input Settings", "Input", "Loop", and "Retry on Error". The "Input" tab is currently active. It contains sections for "Activity Input" and "Upstream Output". In the "Activity Input" section, the "message" input is selected, and the payload is set to "\$activity[MapperStringify].output.stringify". In the "Upstream Output" section, the "MapperStringify" output is selected. To the right of the configuration pane, there's a "Functions" list containing various utility functions like array, boolean, coerce, datetime, float, json, number, string, utility, and utils. At the bottom right of the configuration pane are "Discard" and "Save" buttons.

Create Delivery Microservice



Build
Microservices

We're going to create a trigger that looks like this:

The screenshot shows the configuration of a 'MessageSubscriber' trigger in the TIBCO Cloud Platform. The title bar indicates the trigger is for 'Messaging/tibco-messaging-tcm-trigger'. The main area is divided into sections: 'Trigger Settings', 'Flow Input', 'Output Settings', 'Map to Flow Inputs', 'Trigger Output', 'Functions', and buttons for 'Discard' and 'Save'.

Trigger Settings: Shows the trigger type as 'MessageSubscriber'.

Flow Input: Set to 'message' with a mapping rule: `1 $trigger.message.payload`.

Output Settings: Set to 'a payload'.

Map to Flow Inputs: This section is currently empty.

Trigger Output: Set to '\$trigger'.

Functions: A list of available functions including array, boolean, coerce, datetime, float, json, number, string, utility, and utils.

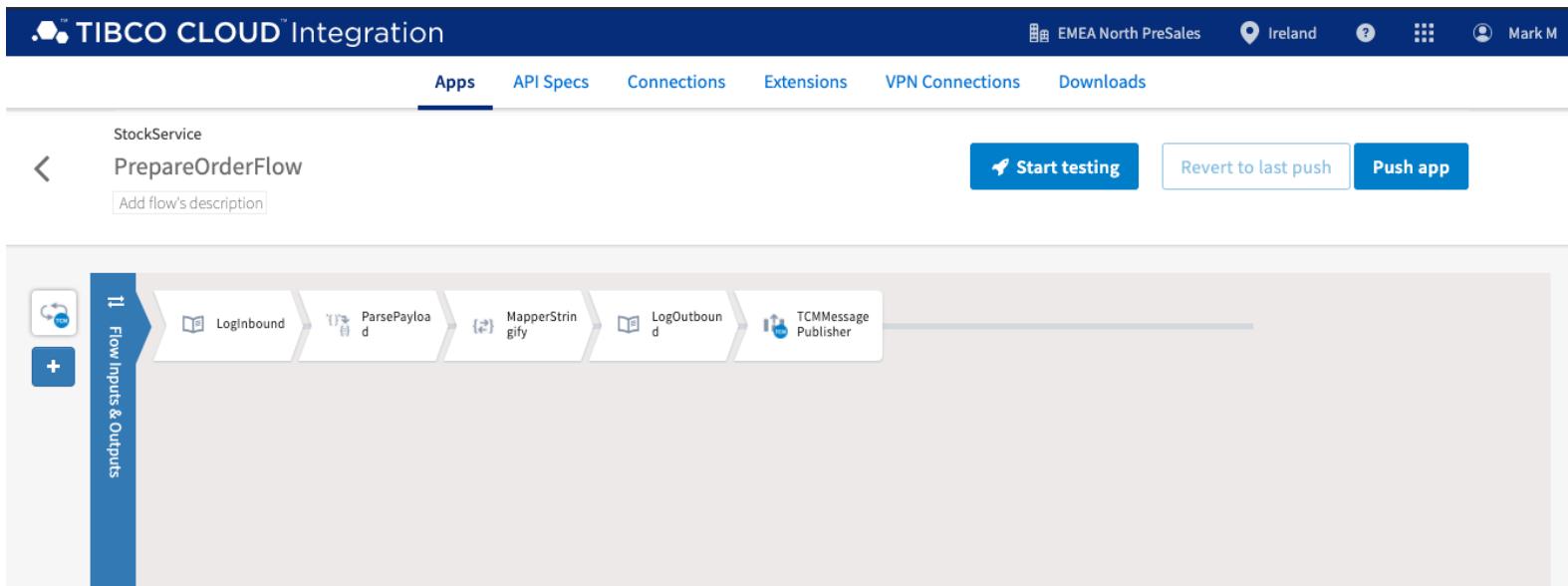
Buttons: 'Discard' and 'Save' buttons at the bottom right.

Create Delivery Microservice

4

Build
Microservices

We're going to create a flow that looks like this:



Create Delivery Microservice - Trigger



Build
Microservices

Click 'Start with a trigger'.

Click 'Message Subscriber' trigger.

Use the Connection dropdown box and choose the TCM Connector you created earlier and click 'Continue' button.

Click 'Just add the trigger' button

Click the TCM trigger symbol

Enter 'prepare' into the Destination field.

Click 'Output Settings' in LHS panel and enter the following JSON into the Message Schema

Click the sync flow input and output symbol in the top right corner of the dialog box

Close the Trigger

The screenshot shows the 'MessageSubscriber' configuration screen. The title bar indicates it's for 'Messaging/tibco-messaging-tcm-trigger'. The main area has tabs for 'Trigger Settings' (selected), 'Output Settings' (highlighted in blue), and 'Map to Flow Inputs'. The 'Output Settings' tab contains a 'Message Schema' section with the following JSON:

```
1 - {  
2   "payload": ""  
3 }  
4 }
```

At the bottom right are 'Discard' and 'Save' buttons.

Create Delivery Microservice - Flow



Build
Microservices

Add Log Activity and configure

The screenshot shows a flow editor at the top with five sequential steps: LogInbound, ParsePayloa, MapperStringify, LogOutbound, and TCMMessage Publisher. Below the flow, a specific activity is selected: **General/tibco-wi-log LogInbound**. This is described as a **Simple Log Message Activity**.

The configuration dialog for this activity is open, divided into several sections:

- Settings**: Contains tabs for **Input** and **Loop**. The **Input** tab shows an **Activity Input** section with a placeholder "a.. * message".
- Activity Input**: Shows the variable "message" with the expression "1 coerce.toString(\$flow.message)".
- Upstream Output**: Shows the variable "\$flow".
- Functions**: A list of available functions:
 - array
 - boolean
 - coerce
 - datetime
 - float
 - json
 - number
 - string
 - utility
 - utils

At the bottom right of the dialog are two buttons: **Discard** and **Save**.

Create Delivery Microservice - Flow



Build
Microservices

Add Parse Activity and configure

Schema ⓘ

```
1 {  
2   "payload" : {  
3     "order_id" : "000000109",  
4     "customer_id" : "abc123",  
5     "email" : "bob.sugar@example.com",  
6     "ShippingAddress" : {  
7       "address1" : "Flat 4",  
8       "address2" : "103 Royal Oak Road",  
9       "city" : "Warwick",  
10      "country" : "United Kingdom",  
11      "postcode" : "CV4 7ES"  
12    }  
13  }  
14 }
```

Create Delivery Microservice - Flow



Build
Microservices

Add Mapper Activity and configure

The screenshot shows a flow diagram and its configuration details. The flow consists of the following steps: Loginbound → ParsePayload → MapperStringify → LogOutbound → TCMMessage Publisher. The 'MapperStringify' activity is currently selected.

General/tibco-wi-mapper
MapperStringify
Mapper Activity

Input Settings

	Activity Input
Input	<input checked="" type="checkbox"/> input a.. stringify
Output	
Loop	

Activity Input

a.. stringify

```
1 string.trim(string.concat("{\"payload\" : { \"order_id\" : \"", $activity[ParsePayload].jsonObject.payload.order_id, "\", \"customer_id\" : \"", $activity[ParsePayload].jsonObject.payload.customer_id, "\", \"status\" : \"SUCCESS\" }}"))
```

Upstream Output

ParsePayload
\$flow

Functions

- ▶ array
- ▶ boolean
- ▶ coerce
- ▶ datetime
- ▶ float
- ▶ json
- ▶ number
- ▶ string
- ▶ utilities

Buttons

Discard Save

Create Delivery Microservice - Flow



Build
Microservices

Add Log Activity and configure

The screenshot shows a TIBCO Business Studio interface. At the top, there is a flow diagram with several nodes: Loginbound, ParsePayload, MapperStringify, LogOutbound, and TCMMessagePublisher. Below the flow, a specific activity is selected: General/tibco-wi-log LogOutbound. This activity is described as a "Simple Log Message Activity". The configuration dialog is open, divided into several sections:

- Settings**: Contains the activity's name.
- Activity Input**: Shows the input message as \$activity[MapperStringify].output.stringify.
- Input**: Shows the input message again.
- Loop**: An empty section for loops.
- Upstream Output**: Shows the upstream output message as \$activity[MapperStringify].output.stringify.
- Functions**: A list of available functions:
 - array
 - boolean
 - coerce
 - datetime
 - float
 - json
 - number
 - string
 - utility
 - utils

At the bottom right of the configuration dialog are two buttons: **Discard** and **Save**.

Create Delivery Microservice - Flow



Build
Microservices

Add a TCM Message Publisher Activity.

Save and Push Application.

The screenshot shows a TIBCO Flow Designer interface. At the top, a process flow is displayed with several activities: Loginbound, ParsePayload, MapperStringify, LogOutbound, and TCMMessagePublisher. The TCMMessagePublisher activity is highlighted. Below the flow, a modal window is open for configuring the TCMMessagePublisher activity.

Messaging/tibco-messaging-tcm-pub
TCMMessagePublisher

This activity sends a message to TIBCO Cloud Messaging se...

Settings

Input Settings

Input (selected)

Loop

Retry on Error

Activity Input

a.. destination

Upstream Output

message > a.. payload

1 \$activity[MapperStringify].output.stringify

Functions

- array
- boolean
- coerce
- datetime
- float
- json
- number
- string
- utility
- utils

Discard **Save**

Start Flogo Apps



Build
Microservices

Verify the 3 Flogo Applications are started

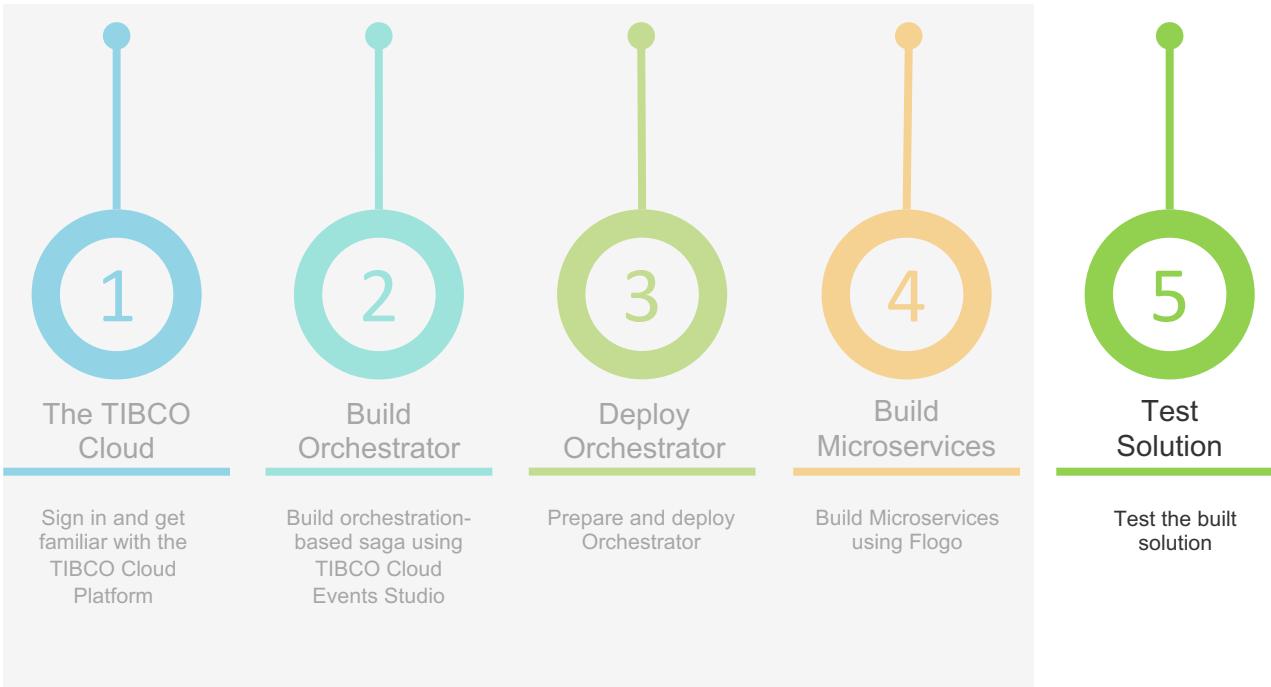
TIBCO CLOUD Integration

Apps API Specs Connections Extensions VPN Connections Downloads

Last modified Create

Flogo App Mock Node.js BusinessWorks

App Name	Status	Last Modified	Actions
PaymentService	Running	Modified on 13 January 2020 12:03 pm Greenwich Mean Time	More
StockService	Running	Modified on 13 January 2020 11:54 am Greenwich Mean Time	More
DeliveryService	Running	Modified on 13 January 2020 11:54 am Greenwich Mean Time	More



Use Swagger to Test Solution



Test
Solution

Click 'View and Test' endpoints and select the swagger link

Click 'Try it out'

Paste the content of 'SalesOrderOrchestrator.json' in to the body (the file is in the root directory of the git repo)

Click 'Execute'

The employee to json to create.

Example Value Model

```
{ "order": { "order_id": "000000001", "customer_id": "3b1a879c-c9cf-4849-b061-bc78adfc55fb", "email": "bob.super@example.com", "items": [ { "item_name": "Gummy Bears", "price": 2.00, "quantity": 10 } ], "shipping_address": { "address1": "Flat 4", "address2": "103 Royal Oak Road", "city": "Worwick", "country": "United Kingdom", "postcode": "CV4 7RE" } }, "id": 4, "payload": { "repository": { "type": "www.tibco.com/be/ontology/Events/ReplyOrderEvent" }, "status": "ORDER ACCEPTED" } }
```

Download

Response headers

```
cache-control: max-age=0, no-cache, no-store, must-revalidate
content-length: 219
content-type: application/json; charset=UTF-8559-1
date: Wed, 20 May 2020 13:32:15 GMT
expires: 0
pragma: no-cache
strict-transport-security: max-age=31536000; includeSubDomains; preload
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block
```

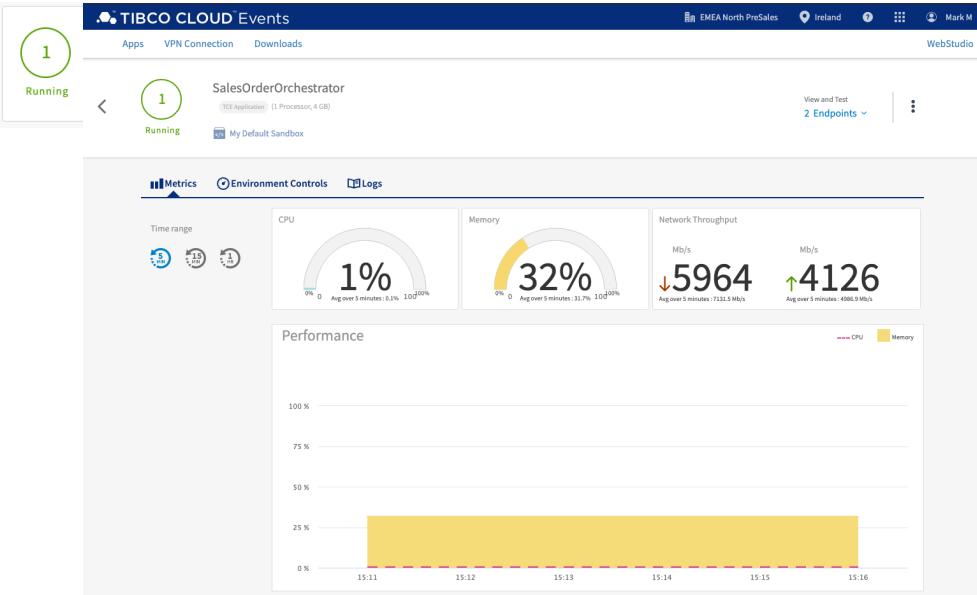
Responses

Code	Description
200	200 success
400	400 Bad Request

View Application Metrics



Click SalesOrderOrchestrator application to show Metrics page



View Application Logs

5

Test
Solution

Click ‘Logs’ Link to view the application logs.

TIBCO CLOUD Events

Apps VPN Connection Downloads

EMEA North PreSales Ireland WebStudio

SalesOrderOrchestrator

TCE Application - [1 Processor, 4 GB]

My Default Sandbox

1 Running

2 Endpoints

Metrics Environment Controls Logs

Find... 20 of 20 results in real time.

Raw Table Download

Time range

Real time Real time displays log messages generated within last 2 hours.

Level

- Error
- Warning
- Info
- Debug

2020-May-20 16:07:50.350 [user] 2020 May 20 - [inference-class] #### OrderDeliveredRule.rule - ORDER 000000123 DELIVERY STATUS = SUCCESS

2020-May-20 16:07:50.350 [user] 2020 May 20 - [inference-class] #### ORDER STATE = COMPLETED

2020-May-20 16:07:50.350 [user] 2020 May 20 - [inference-class] #### OrderDeliveredRule.rule - Received Order Delivered Event with payload - ... Show More

2020-May-20 16:07:50.349 [user] 2020 May 20 - [inference-class] >>> OrderDeliveredRule.rule - Order Delivered Rule fired

2020-May-20 16:07:50.324 [user] 2020 May 20 - [inference-class] >>> OrderPreparedRule.rule - Order Prepared Rule fired

2020-May-20 16:07:50.324 [user] 2020 May 20 - [inference-class] #### ORDER STATE = PENDING_PREPARE

2020-May-20 16:07:50.324 [user] 2020 May 20 - [inference-class] #### OrderPreparedRule.rule - Received Order Prepared Event with payload - ... Show More

2020-May-20 16:07:50.324 [user] 2020 May 20 - [inference-class] #### OrderPreparedRule.rule - ORDER 000000123 PREPARED STATUS = SUCCESS

2020-May-20 16:07:50.312 [user] 2020 May 20 - [inference-class] #### PaymentExecuteRule.rule - ORDER 000000123 PAYMENT STATUS = SUCCESS

2020-May-20 16:07:50.312 [user] 2020 May 20 - [inference-class] #### ORDER STATE = null

2020-May-20 16:07:50.312 [user] 2020 May 20 - [inference-class] #### PaymentExecuteRule.rule - Payment approved for Order #000000123

2020-May-20 16:07:50.311 [user] 2020 May 20 - [inference-class] >>> PaymentExecuteRule.rule - Payment Executed Rule fired

2020-May-20 16:07:50.311 [user] 2020 May 20 - [inference-class] #### PaymentExecuteRule.rule - Received Payment Executed Event with payload - ... Show More

2020-May-20 16:07:50.231 [user] 2020 May 20 - [inference-class] #### OrderRule.rule - Gummy Bears - £2.0 (X 10)

2020-May-20 16:07:50.231 [user] 2020 May 20 - [inference-class] #### OrderRule.rule - Order Value £20.0

2020-May-20 16:07:50.231 [user] 2020 May 20 - [inference-class] #### OrderRule.rule - Order processed: 000000123 Customer:abc123

2020-May-20 16:07:50.230 [user] 2020 May 20 - [inference-class] >>> OrderRule.rule - Order Rule fired

2020-May-20 16:07:50.230 [user] 2020 May 20 - [inference-class] Order Id - 000000123

2020-May-20 16:07:50.226 [user] 2020 May 20 - [inference-class] #### Order PreProcessor received Request with payload - {"order": {"attribut... Show More

View Application Logs



TIBCO Cloud
Events Receives
Order request over
HTTP



Send Order



Order PreProcessor ->
Order Rule ->



TIBCO Cloud
Events sends
Payment Event to
Payment Service



*Payment
Microservice*



All Hands International Presales Call - Google Slides	
2020-May-20 16:07:50:226	[user] 2020 May 20 - [inference-class] ##### Order PreProcessor received Request with payload - { "order" : { "attributes" : ["type" : "www.tibco.com/be/ontology/Concepts/Order"], "order_id" : "000000123", "customer_id" : "abc123", "email" : "bob.sugar@example.com", "items" : [{ "item_name" : "Gummy Bears", "price" : "2.0", "quantity" : "10" }], "address" : { "address1" : "Flat 4", "address2" : "103 Royal Oak Road", "city" : "Warwick", "country" : "United Kingdom", "postcode" : "CV4 7ES" }] } Show Less
2020-May-20 16:07:50:226	[user] 2020 May 20 - [inference-class] >>> Order PreProcessor fired
2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] ##### OrderRule.rule - Gummy Bears - £2.0 (X 10)
2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] ##### OrderRule.rule - Order Value £20.0
2020-May-20 16:07:50:231	[user] 2020 May 20 - [inference-class] ##### OrderRule.rule - Order processed: 000000123 Customer:abc123
2020-May-20 16:07:50:230	[user] 2020 May 20 - [inference-class] >>> OrderRule.rule - Order Rule fired
2020-May-20 16:07:50:230	[user] 2020 May 20 - [inference-class] Order Id - 000000123

View Application Logs



TIBCO Cloud
Events Receives
Payment Received
Event from
Payments Service



Payment Executed
Rule ->



TIBCO Cloud
Events sends
Prepare Order
Event to Stock
Service



**Payment
Microservice**



**Payment
Taken**

	2020-May-20 16:07:50:312	[user] 2020 May 20 - [inference-class] ##### PaymentExecuteRule.rule - Payment approved for Order #000000123
	2020-May-20 16:07:50:311	[user] 2020 May 20 - [inference-class] >>> PaymentExecuteRule.rule - Payment Executed Rule fired
	2020-May-20 16:07:50:311	[user] 2020 May 20 - [inference-class] ##### PaymentExecuteRule.rule - Received Payment Executed Event with payload - { "payload" : { "attributes" : [{ "type" : "www.tibco.com/be/ontology/Events/PaymentExecutedEvent" }, { "order_id" : "000000123", "customer_id" : "abc123", "status" : "SUCCESS" }] } Show Less



**Stock
Microservice**



**Fulfill
Order**

View Application Logs



TIBCO Cloud
Events Receives
Order Prepared
Event from Stock

Service



Order Prepared
Executed Rule ->



TIBCO Cloud
Events sends
Prepare Delivery
Event to Delivery
Service



**Stock
Microservice**



	2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] >>> OrderPreparedRule.rule - Order Prepared Rule fired
	2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] ##### ORDER STATE = PENDING_PREPARE
	2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] ##### OrderPreparedRule.rule - Received Order Prepared Event with payload - { "payload" : { "attributes" : { "type" : "www.tibco.com/be/ontology/Events/OrderPreparedEvent" }, "order_id" : "000000123", "customer_id" : "abc123", "status" : "SUCCESS" } } Show Less
	2020-May-20 16:07:50:324	[user] 2020 May 20 - [inference-class] ##### OrderPreparedRule.rule - ORDER 000000123 PREPARED STATUS = SUCCESS



**Delivery
Microservice**



View Application Logs

5

Test
Solution

TIBCO Cloud
Events Receives
Order Delivered
Event from Delivery
Service



*Delivery
Microservice*

Order
Delivered

Order Delivered
Service



Order Delivered
Executed Rule ->

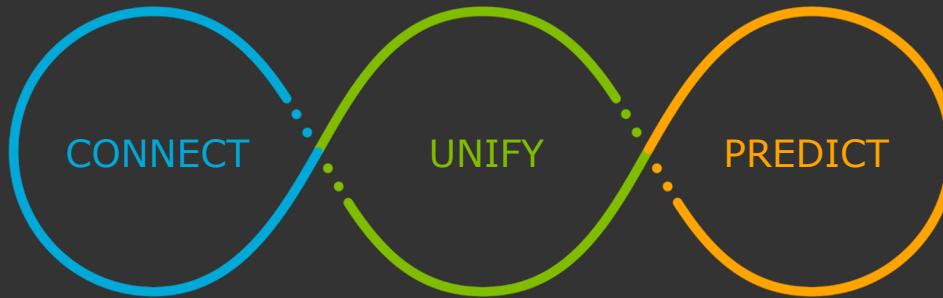


TIBCO Cloud
Events sends
Prepare Delivery
Event to Delivery
Service

 2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] ##### OrderDeliveredRule.rule - ORDER 000000123 DELIVERY STATUS = SUCCESS
 2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] ##### ORDER STATE = COMPLETED
 2020-May-20 16:07:50:350	[user] 2020 May 20 - [inference-class] ##### OrderDeliveredRule.rule - Received Order Delivered Event with payload - .. Show More
 2020-May-20 16:07:50:349	[user] 2020 May 20 - [inference-class] >>> OrderDeliveredRule.rule - Order Delivered Rule fired

Your journey of innovation does not stop here...

TIBCO provides the most optimal path to digital innovation



A single platform takes you from **connected** to **intelligent**.

Thank you

