Acumatica Virtual DevCon 2020

June 17 & 18

Integration: How It’s Done

Let’s Integrate Acumatica with the World!

Sergey Marenich

Commerce Edition Architect & Team Lead

Marenich Sergey

13 years of experience at Acumatica

- Build Engineer
- System Developer
- Product Engineer
- Solution Architect
- Team Lead
- Commerce Edition Architect

@: smarenich@acumatica.com
P: Commerce Edition Architect & Team Lead
U: http://asiablog.acumatica.com

Integration: Is there something unsaid yet?

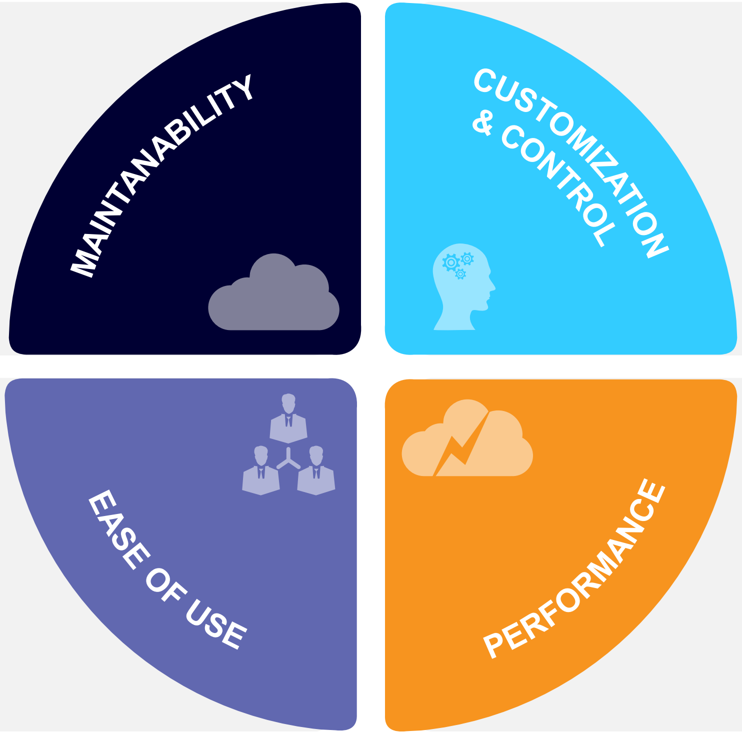
Today’s **World**

What Clients Want from “Integration”?

Reliable & Consistency

Notifications on Issues

Ability to Solve Problems



Choose what to Sync

Adjust Workflow

Conflict Resolution

Plug & Play

Develop Fast

Low Cost

Close to Realtime

Run Fast

Integration is Easy, Right? Not Quite…

Data Structures & Business Logic

Each system designed in the different ways and approaches.

This leads to challenges with:

- Data flow
- Data mapping
- Actions calling
- Custom fields handling

Versions Hell

Each system supports own version cycle which significantly increases the complicity of integration project with any addition system

API Limits & Throttling

Each system protects own resources and applies license.

As a result integration suffers from very different limit of resources.

API Technology Evolution

Each system relies on different technologies and even changes in a while:

- REST
- Graph QL
- SOAP
- gRPC
- JSON-RPC
- Thrift

Acumatica
The Cloud ERP

APIANT

tray.io

Cloud Elements

celigo

zapier

ADAPTRIS™

Jitterbit

integromat

IFTTT

Microsoft Flow

azuqua

automate.io

Informatica

wyzebulb

piesync™

elastic.io

workato

built.io® flow

ONESAAS

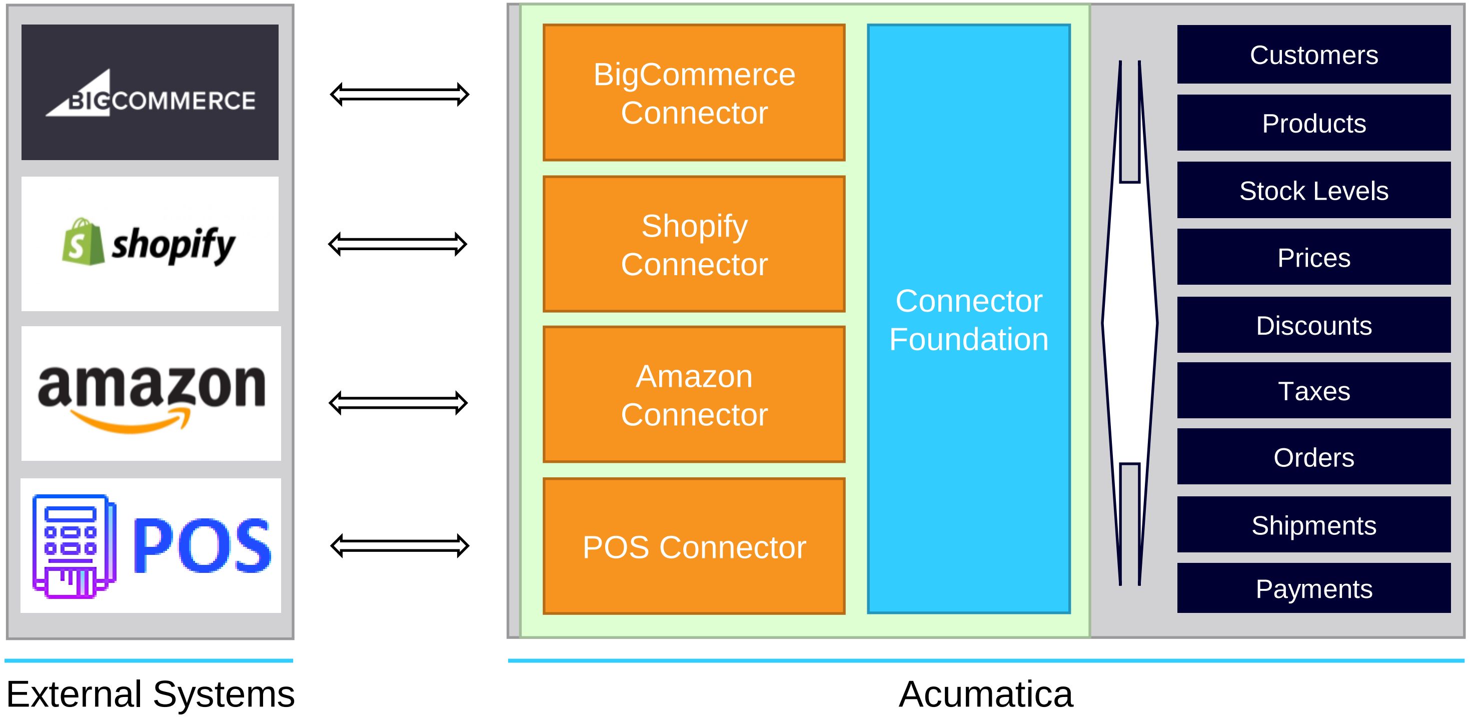
MuleSoft®

Integration Platforms – Solution?

Complicated	Complexity of implementation is shifted from developer to the engineer through configurations
API Clients	Development and Code Maintenance effort of API clients is still required
Productizing	Applying of the same use-case to multiple clients requires special deployment procedures
Upgrade	Upgrade requires abstraction level between core integration and customer specific customization
Business Logic Flow	Complicated systems like ERP with designed business flow (Order-Shipment-Invoice) adds extra complexity

Idea: Acumatica Commerce Edition

Commerce Enabled ERP – Vision



Commerce Enabled ERP – System Requirements

Plugin-like Architecture with the same Infrastructure

Interaction Abstraction on Acumatica Screens

Processing Queue to Spread out Peaks Load

Parallel Processing of the Queue

Push and Scheduled Synchronization

User Defined Fields Mapping (with Formulas)

User Define Conditions (Filters) for Synchronization

Synchronization Algorithm, Status and Conflict Resolutions

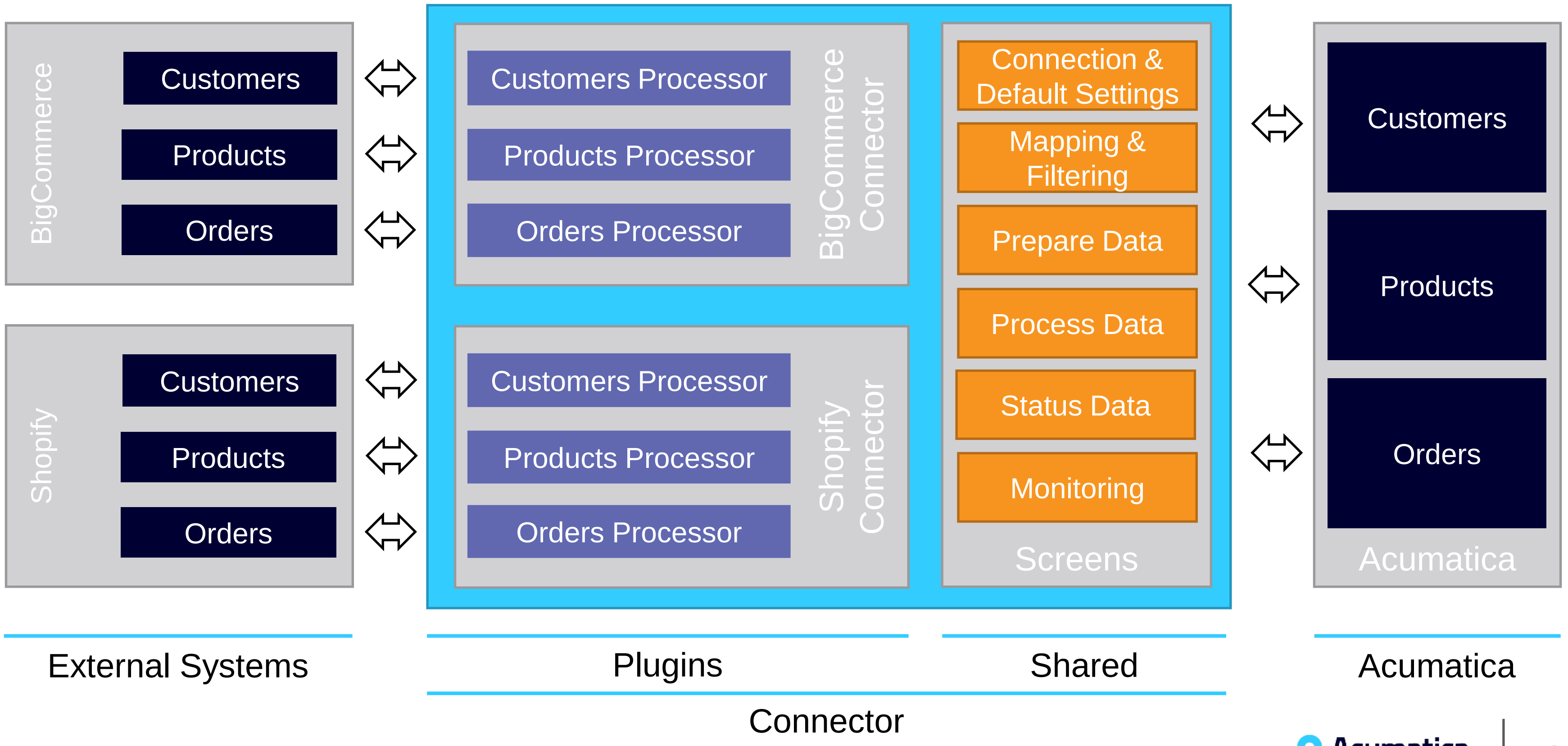
Tools: Logging, Monitoring, Issue Handling

Duplicates Merge

Specialized API Endpoints: Taxes, Availability, CC Tokens

Plugin-like Architecture

Connector Architecture



Abstraction on Acumatica Screens

Saving Data to Acumatica

Direct to Database

- Pros
  - Very Fast
  - No need to learn BQL
- Cons
  - Need to know SQL
  - Bypassing business logic
  - High change to make a mistake
  - Affected by upgrades
  - Does not work on SaaS

Graph Program API

- Pros
  - Fast
  - Flexible
- Cons
  - Need to learn BQL and Event Model
  - Logic requires emulation of user interface
  - Affected by Upgrades

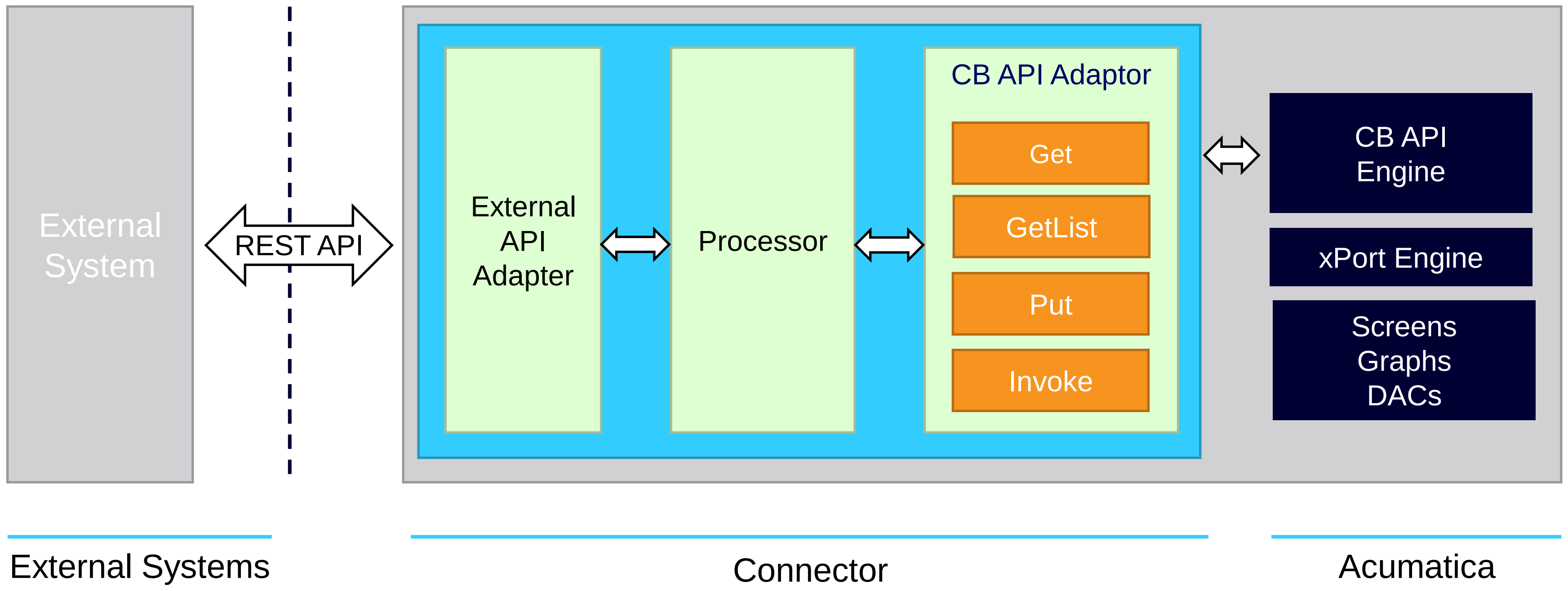
Import Scenarios with Data Provider

- Pros
  - User configurable mapping
  - Great business logic abstraction
  - No need to learn BQL
  - Less code needed
- Cons
  - Hard to implement data flow
  - Hard to do data queries
  - Hard to run in parallel
  - Hard to upgrade

Contract Base API

- Pros
  - Contracts are protected from upgrades
  - Great business logic abstraction
  - No need to learn BQL
  - Easy Data Queries
  - Easy Parallel Processing
  - Endpoint is an extension point
  - Documentation
- Cons
  - API Calls Limits

Read/Save Abstraction

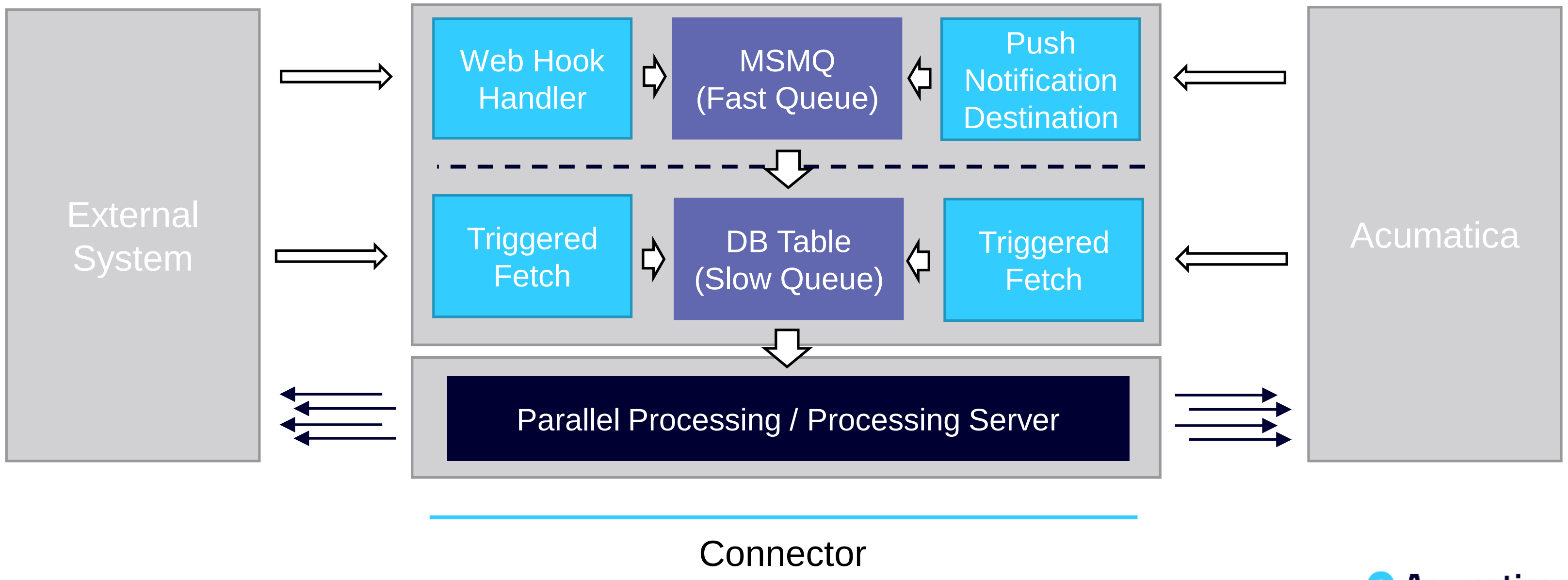


Processing Queues with Parallel Processing

Processing Queues

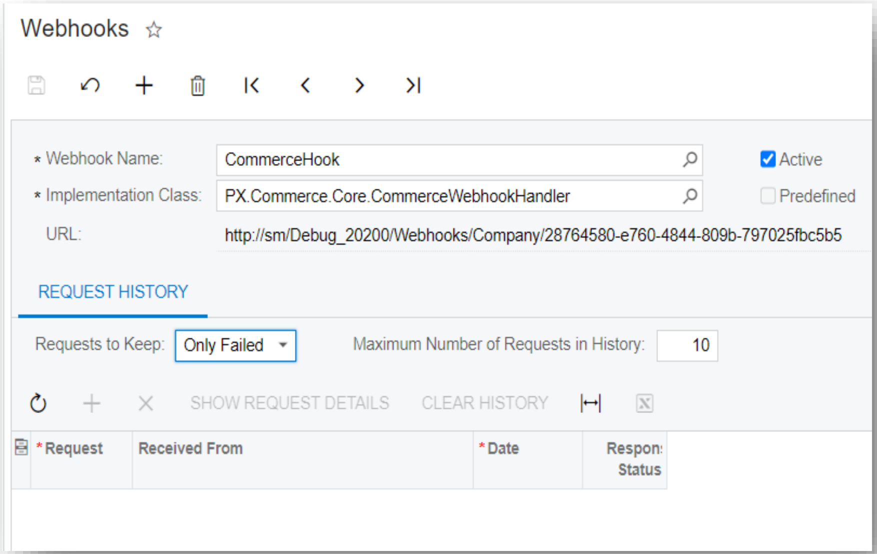
Queues help us to:

- Withstand the load
- Spread Peaks
- Splitting of fetching and saving logic
- Dedicated Processing Node



Handling of Real-Time Sync

Webhook Handler



Webhooks

- * Webhook Name: CommerceHook
- ☑ Active
- * Implementation Class: PX.Commerce.Core.CommerceWebhookHandler
- ☐ Predefined
- URL: http://sm/Debug_20200/Webhooks/Company/28764580-e760-4844-809b-797025fbc5b5

REQUEST HISTORY

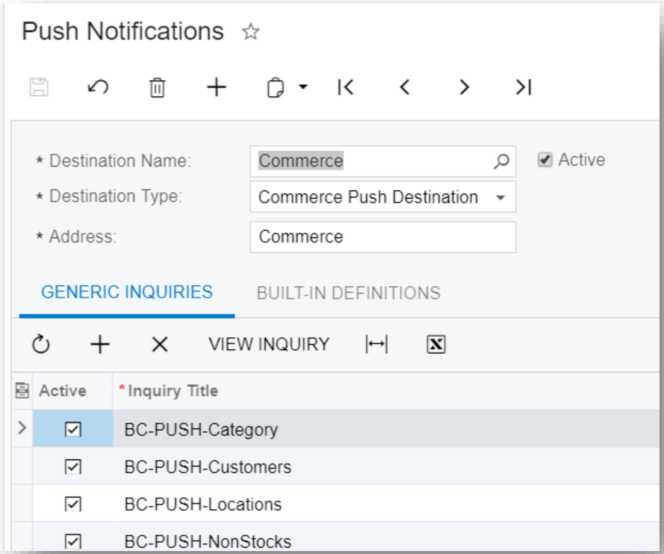
Requests to Keep: Only Failed

Maximum Number of Requests in History: 10

SHOW REQUEST DETAILS · CLEAR HISTORY

*Request	Received From	*Date	Respon: Status

Push Notification Destination



Push Notifications

- * Destination Name: Commerce
- ☑ Active
- * Destination Type: Commerce Push Destination
- * Address: Commerce

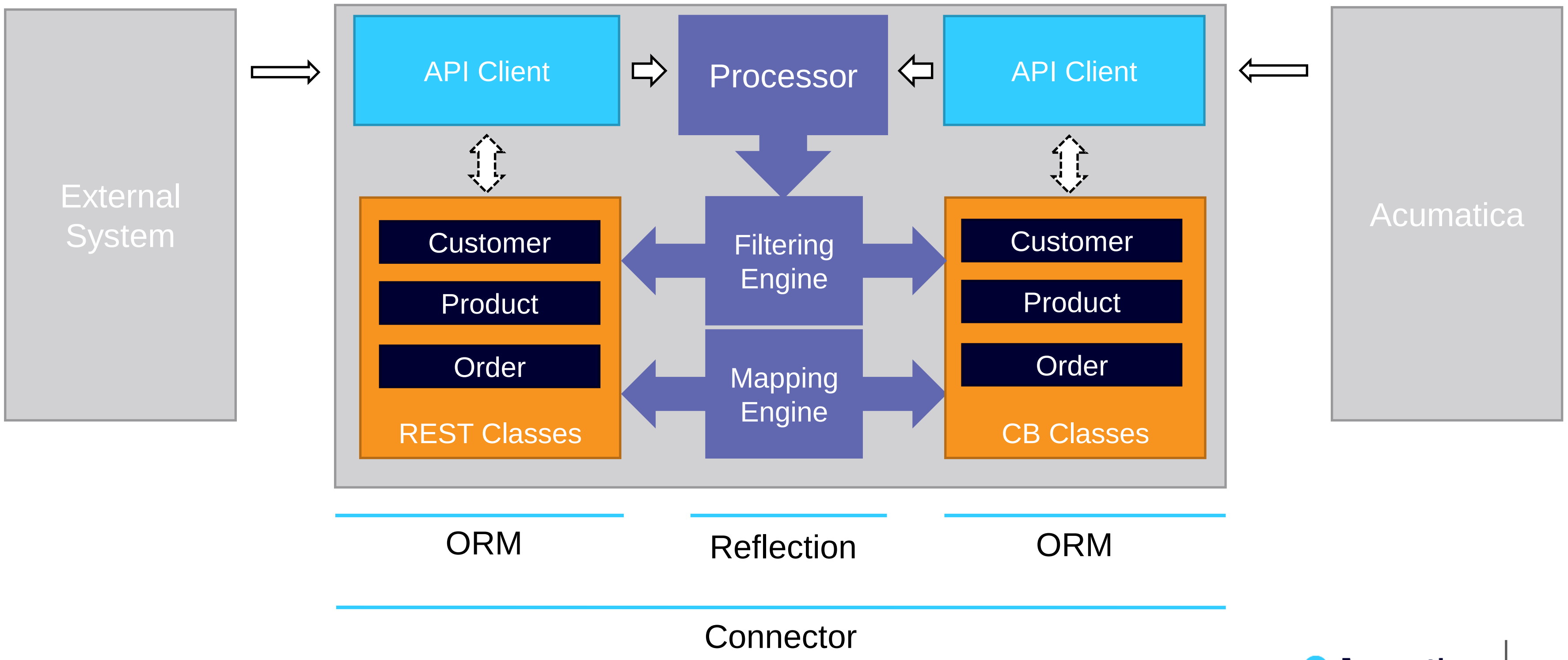
GENERIC INQUIRIES · BUILT-IN DEFINITIONS

VIEW INQUIRY

Active	*Inquiry Title
☑	BC-PUSH-Category
☑	BC-PUSH-Customers
☑	BC-PUSH-Locations
☑	BC-PUSH-NonStocks

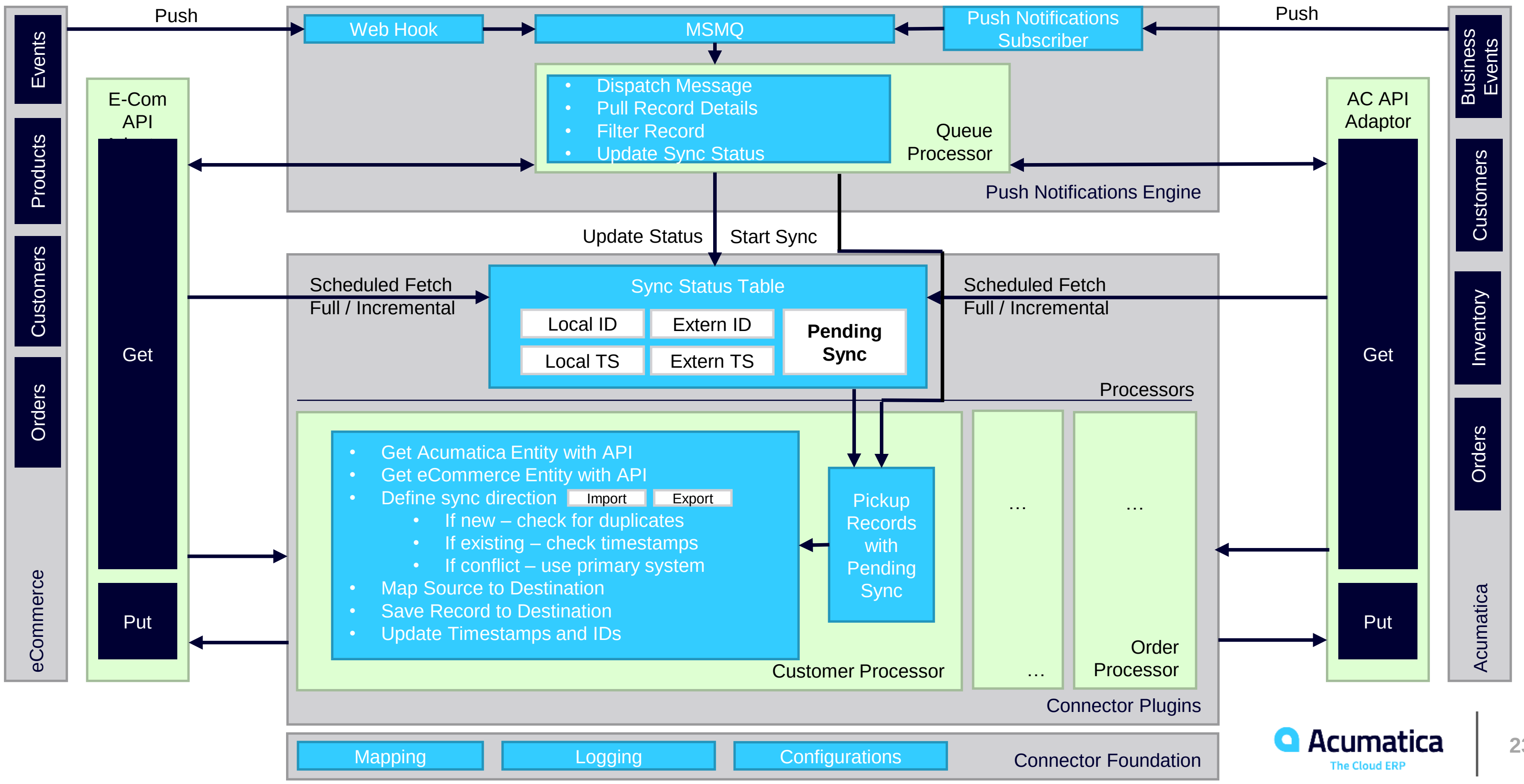
Filtering & Mapping

Filtering & Mapping



Synchronization Algorithm and Status

Synchronization Algorithm and Status



Acumatica The Cloud ERP

Synchronization Status

Sync ID	Identity, Primary Key
Status	Current Status of the record: Synchronized, Pending, Failed, Skipped, Deleted, …
Local ID	Note ID from Acumatica. Starting from Acumatica 2019R2 you can use Note ID as permanent Key for API Calls.
Local Time Stamp	Date & Time when record was Last Modified at Acumatica
External ID	ID of the record from External System
External Time Stamp	Date & Time when record was Last Modified at External System
Last Error	Last Synchronization Error if record is in Failed Status

Synchronization Algorithm

	Operation		Status	LocalID	LocalTS	ExtenID	ExternTS
1	New Customer created Externally at 1:23PM 1/1/2020		Pending			1	**1:23PM 1/1/2020**
2	Synchronization of Customer at 1:31PM 1/1/2020		Synced	5819C47C-1DCC-…	1:31PM 1/1/2020	1	1:23PM 1/1/2020
3	Customer has Updated Locally At 1:44PM 1/1/2020		Pending	5819C47C-1DCC-…	1:31PM 1/1/2020	1	1:23PM 1/1/2020
4	Synchronization of Customer at 1:49PM 1/1/2020		Synced	5819C47C-1DCC-…	**1:44PM 1/1/2020**	1	**1:49PM 1/1/2020**
5	Customer has Updated Externally At 2:01PM 1/1/2020		Pending	5819C47C-1DCC-…	1:44PM 1/1/2020	1	1:49PM 1/1/2020
6	Customer has Updated Locally At 2:07PM 1/1/2020		Pending	5819C47C-1DCC-…	1:44PM 1/1/2020	1	1:49PM 1/1/2020
7	Synchronization of Customer in favor of Primary System at 2:10PM 1/1/2020		Synced	5819C47C-1DCC-…	**2:10PM 1/1/2020**	1	**2:07PM 1/1/2020**

Acumatica
The Cloud ERP

Synchronization Algorithm

Get

- Get Object from Local System
- Get Object from External System
- Apply Filtering Conditions

Check for Duplicates

- Only if record is new, try to merge by ID

Define Direction

- Compare Time Stamps and define what record has changed since last time
- In case of conflict solve in favor for primary system

Map Data

- External to Local or Local to External Depend on Direction
- Apply User Mapping

Save Changes

- Update Time Stamps

Acumatica
The Cloud ERP

Development

Implementing of New Connector

- Libraries:
  - PX.Commerce.Core
  - PX.Commerce.Objects

- <Connector> : IConnector
  - PXGraph
  - Connection Settings
  - Navigation to Records
  - Realtime Subscription & Processing
  - Sync Processing

- <Processors> : IProcessor
  - PXGraph
  - Fetching of Records
  - Getting of Local and External Records
  - Default Mapping Logic
  - Export and Import Logic
  - Sync Processing

Implementing of New Processor

Description	Import	Export

Fetch changed records
Update BCSyncStatus
	`void GetBucketsForImport(`
	`    DateTime? lastModifiedDateTime,`
	`    PXFilterRow[] filters)`
		`void GetBucketsForExport(`
		`    DateTime? lastModifiedDateTime,`
		`    PXFilterRow[] filters)`

Get entity (with all details)	`bool GetBucketForImport(`
	`    BCSalesOrderBucket bucket,`
	`    BCSyncStatus syncstatus)`
		`bool GetBucketForExport(`
		`    BCSalesOrderBucket bucket,`
		`    BCSyncStatus syncstatus)`

Map single entity between systems	`void MapBucketImport(`
	`    BCSalesOrderBucket bucket,`
	`    IMappedEntity existing)`
		`void MapBucketImport(`
		`    BCSalesOrderBucket bucket,`
		`    IMappedEntity existing)`

Save entity to destination system (with all details)	`void SaveBucketImport(`
	`    BCSalesOrderBucket bucket,`
	`    IMappedEntity existing,`
	`    String operation)`
		`void SaveBucketImport(`
		`    BCSalesOrderBucket bucket,`
		`    IMappedEntity existing,`
		`    String operation)`

Pull primary entity only.
For push notificaitons
	`MappedOrder PullEntity(`
	`    String externID,`
	`    String jsonObject)`
		`MappedOrder PullEntity(`
		`    Guid? localID,`
		`    Dictionary<String, Object> fields)`

Acumatica
The Cloud ERP

Summary

Acumatica Integration - Expectations

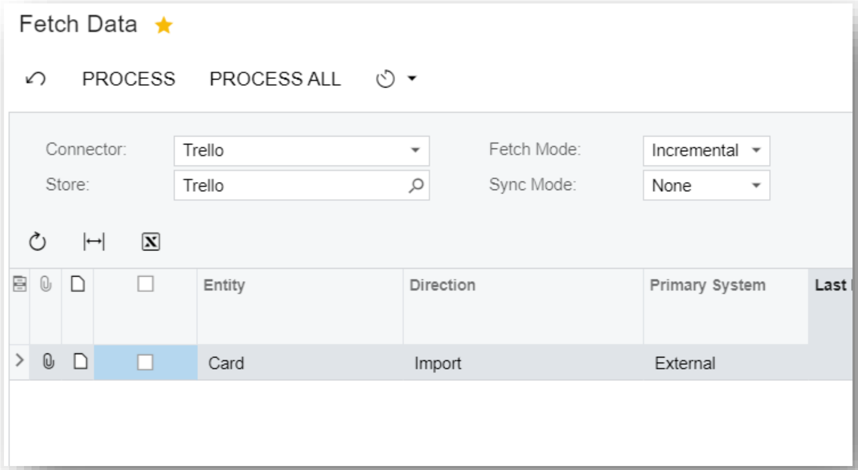
- Framework
- Requires Coding: API Clients, Mapping, Logic
- Experimental Model
- Commerce focused (Shipping Cards, POS)
- Documentation is in the Roadmap
- Time to Complete

Code Example – Trello Connector

Git Hub Project: https://github.com/smarenich/TrelloConnector

Simplifications:

- API URLs Hardcoded
- API Credentials Hardcoded
- Only Cards Import (No boards, No lists)
- No Push Notifications


Fetch Data

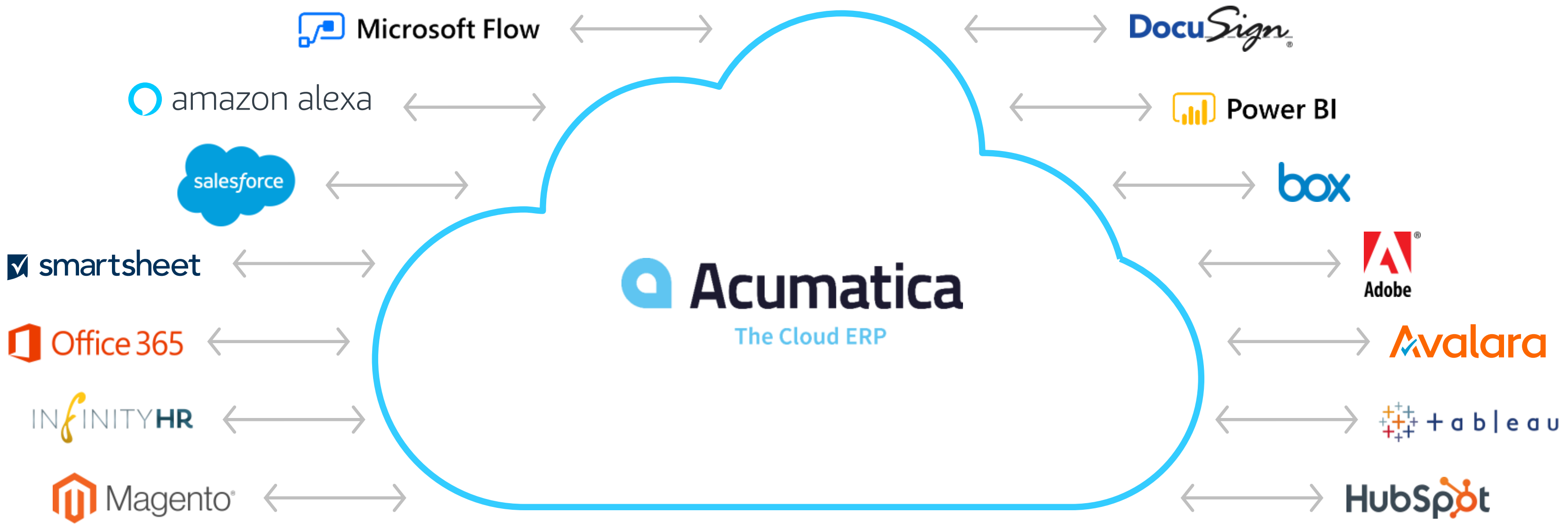
Connector: Trello
Store: Trello
Fetch Mode: Incremental
Sync Mode: None

	Entity	Direction	Primary System	Last
☐	Card	Import	External	

When?

Fall 2020

Open Architecture and Rapid Integration



Acumatica
The Cloud ERP

Thank You!

Sergey Marenich

smarenich@acumatica.com