

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра информационных систем

РЕКОМЕНДОВАНО К ЗАЩИТЕ
В ГЭК И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ

Заведующий кафедрой

д.т.н., профессор

И.Н. Глухих

2017 г.



МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

СРАВНЕНИЕ МЕТОДОВ АВТОМАТИЧЕСКОЙ ИНТЕГРАЦИИ ДАННЫХ

09.04.03 Прикладная информатика

Магистерская программа «Прикладная информатика в экономике»

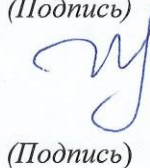
Выполнил работу
Студент 2 курса
очной формы обучения



(Подпись)

Баширу
Абдуллахи
Олавале

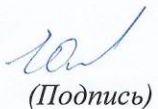
Научный руководитель
к.физ-мат.н., доцент



(Подпись)

Григорьев
Андрей
Викторович

Рецензент
к.т.н., доцент кафедры ИБ
ФГАОУ ВО «ТюмГУ»



(Подпись)

Оленников
Евгений
Александрович

Тюмень 2017

TABLE OF CONTENTS

| | |
|--|----|
| ABSTRACT/ANNOTATION | 3 |
| Chapter 1: INTRODUCTION AND PROBLEMS | 4 |
| Chapter 2: HOW DATA SHOULD SYNC | 6 |
| 2.1 System to integrate (Salesforce and Magento)..... | 6 |
| 2.2 Objects that need to be synchronized..... | 7 |
| 2.3 Fields to sync | 7 |
| Chapter 3: INPUTTING YOUR DATA INTO AN INTEGRATION SYSTEM | 8 |
| 3.1 Core ESB Features | 8 |
| 3.2 The Advantages of ESB | 9 |
| Chapter 4: DEVELOPING INTEGRATIONS IN MULESOFT | 11 |
| 4.1 Anypoint Studio | 11 |
| 4.1.1 Package Explorer | 12 |
| 4.1.2 Canvas | 12 |
| 4.1.3 Mule Palette | 14 |
| 4.1.4 Mule Message | 15 |
| 4.2 Designing a Mule Application | 15 |
| Chapter 5: MAP YOUR SYSTEMS, OBJECTS AND FIELDS | 19 |
| 5.1 Mapping Magento Customer Fields to Salesforce Contact Fields..... | 22 |
| 5.2 Mapping Magento Customer Fields to Salesforce Account Fields..... | 23 |
| 5.3 Mapping Magento Orders Attributes with Salesforce Opportunity Fields | 24 |
| Chapter 6: TASKS FOR EXECUTION | 27 |
| 1. HTTP-Magento-Salesforce-Customer | 27 |
| 2. HTTP-Salesforce-Magento-Customer | 33 |
| 3. HTTP-Magento-Salesforce-Order | 37 |
| 4. POLL-Salesforce-Magento-Customer | 41 |
| 5. POLL-Magento-Salesforce-Customer | 47 |
| 6. HTTP-Salesforce-Magento-Order | 52 |
| 7. POLL-Magento-Salesforce-Order | 56 |
| 8. POLL-Salesforce-Magento-Order | 61 |
| CONCLUSION | 66 |
| REFERENCES | 67 |

ABSTRACT/ANNOTATION

This thesis discusses the problems of integration of different systems that are managed by events, using service buses of enterprises of different manufacturers as an infrastructure layer.

Salesforce to Magento can be used to create data sharing relationship. Salesforce is a natively supported feature of the Force.com platform, and easily enables two trading partners to share relevant data records between orgs; it shows how to create connections between two different environments, and how to use the connection to share data.

Magento offers quick and easy integration with market-leading CRM, Salesforce, allowing the rapid and seamless flow of data between the two systems:

- Pull customer lists from Salesforce directly into Magento
- Build stronger account profiles from additional data in CRM
- Automatically update database changes in Salesforce
- Create leads, orders in Salesforce with new registration data
- Real-time overview of event history in Salesforce

Chapter 1: INTRODUCTION AND PROBLEMS

Data transformation is one of the most common problems facing systems integrators as source data is often in an inconsistent format or structure for systems wanting to use that data. This requires integrators to implement code for the mapping operations required to convert the data from one form to another e.g. from one XML document format to another. The code to do this is often tedious to write, consisting typically of pages of C++, Java, or XSLT code, and, as a result, tends to be error prone.

Data transformation problems are challenging to implement for large, complex datasets. I describe an approach for specifying data mapping transformations between XML schema using a combination of automated schema analysis agents and selective user interaction. A graphical tool visualizes parts of the two schemas to be mapped and a variety of agents analyze all or parts of the schema, voting on the likelihood of matching subsets. The user can confirm or reject suggestions, or even allow schema matches to be automatically determined, incrementally building up a fully-mapped schema. An implementation of the mapping specification can then be generated from the various inter-schema matches.

Some of the agents are listed below with a brief description of their input, their heuristic technique, i.e. things they look for in schema or data XML structures, and the “quality” of resultant mapping correspondence suggestions.

Exact Name Matcher- This agent compares element names in one schema to those in another, suggesting mappings when two have the same tag name. This works well when tag names are the same and unique across each document e.g. *LastName* in both schema.

Partial Name Matcher- This looks for a substring that matches in each name, e.g. *Price* to *Unit Price*.

Element Type Matcher- This compares data type names of elements e.g. *CustomerID: Integer* and *ID: String*.

Record Type Matcher- This compares record types (sets of elements) rather than leaf element types (single types). For example; *Name: LastName & FirstName* and

Name: name(s) may correspond if the complex (multi-valued record types) Name (Contact) and Name (Account) are the same or can be converted.

Synonym Matcher- This can be applied to element tag names or element type names. The Synonym Matcher compares names, or parts of names, to see if they are synonyms of each other e.g. *DOB* and *DateOfBirth* are likely to correspond in some way.

Exact Data Value Matcher- This looks at XML data records rather than schema and identifies a correspondence between a single source and target element if their values are the same.

Partial Data Value Matcher- This looks at XML data values from one or multiple elements and computes a likelihood match, similar to the Name Closeness Matcher for element and type names.

Figure below shows parts of two XML schema representing information about our two (2) systems:

```
<?xml version="1.0"?>
<layout>
  <customer_account>
    <account number="12345">
      <accountname name="John Doe">
        <template>stackexchange_customer/form/edit.phtml</template>
      </accountname>
    </account>
  </customer_account>
  <customer_account_create>
    <account number="12345">
      <accountname name="John Doe">
        <template>stackexchange_customer/register.phtml</template>
      </accountname>
    </account_create>
  </customer_account_create>
</layout>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<sObjects xmlns="http://www.force.com/2009/06/asynccapi/dataload">
  <sObject>
    <Name>Xytrex Co.</Name>
    <Description>Industrial Cleaning Supply Company</Description>
    <Account Number>ABC15797531</Account Number>
  </sObject>
  <sObject>
    <Name>Watson and Powell, Inc.</Name>
    <Description>Law firm. New York Headquarters</Description>
    <Account Number>ABC24689753</Account Number>
  </sObject>
</sObjects>
```

Figure 1: Example schema mapping between our 2 systems

Chapter 2: HOW DATA SHOULD SYNC

2.1 System to integrate (Salesforce and Magento)

Magento is a feature-rich eCommerce platform built on open-source technology that provides online merchants with unprecedented flexibility and control over the look, content and functionality of their eCommerce store. Magento's intuitive administration interface features powerful marketing, search engine optimization and catalog-management tools to give merchants the power to create sites that are tailored to their unique business needs. Magento offers companies the ultimate eCommerce solution.

Salesforce integration into magento platform allows companies to monitor customer's behavior not only by products purchased but also increase repeat purchases, brand loyalty and conversion rates, social media engagements in an integrated database.

Magento Salesforce CRM Integration is a powerful tool that helps store-owners to synchronize data automatically between Magento site and Salesforce CRM.

Features and Benefits of Salesforce and Magento

- Allows synchronizing and updating Customers from Magento store into Salesforce CRM Leads, Contacts, Accounts;
- Allows synchronizing Orders from Magento store into Salesforce CRM Orders;
- Allows synchronizing Promotions from Magento store into Salesforce CRM Campaigns;
- Allows synchronizing and updating Products from Magento store into Salesforce CRM Products;
- Allows synchronizing Product Categories from Magento store into Salesforce CRM Price Book;
- Allows creating Custom Fields in and synchronizing Custom Invoices from Magento store into Salesforce CRM Custom Invoices;
- Allows synchronizing Custom Invoice Items;
- Allows creating Custom Fields in, synchronizing and updating Custom Customers from Magento store into Salesforce CRM Custom Customers;

- Allows creating Custom Fields in, synchronizing and updating automatically Custom Products from Magento store into Salesforce CRM Custom Products;
- Allows deleting customer and product's records in Magento once they are auto deleted in Salesforce CRM Leads, Contacts, Accounts, Custom Customer and Custom Product;
- Allows admins to select conditions to transfer old data from Magento store into Salesforce CRM;
- Reports about synchronizing old data;
- Supports mapping smartly and manually between Magento attributes and Salesforce fields or custom fields;
- Allows admins define the mapping fields in backend with ease;
- Admin can view the log to see what is synchronized between two apps.

2.2 Objects that need to be synchronized

What is synced between Magento and Salesforce?

Lead, Account, Opportunity, Contact, etc...

All the Entities of the objects listed above that needs to be synchronize will fully discussed under the chapter 'mapping'.

2.3 Fields to sync (where the data lives)

We sync most standard fields in SFDC and any custom field that the sync user has permission to see.

In contrast, Magento users only have four fields available for synchronization:

- First name
- Last name
- Email address
- ID

Chapter 3: INPUTTING YOUR DATA INTO AN INTEGRATION SYSTEM (Choosing the right system for you)

Integration of information systems of various classes remains an urgent topic for many companies and government organizations. To increase efficiency and transparency of activities, it is important to organize end-to-end business processes and ensure the interaction of various information systems.

The solution was the approach based on the use of services. Information systems are divided into functionally complete, independent components (services), each of which is designed to perform a certain. To execute a business process, you must ensure that the services are called in the right sequence. The IT architecture, based on the allocation and interaction of services, was called SOA (service-oriented architecture). One of the most common ways to implement services in SOA is to use web services.

The combination of service concepts, business process management and integration servers led to the creation of a new class of integration solutions - Enterprise Service Bus. Currently, ESB is the most advanced tool for performing complex and large-scale integration projects.

When integrating, especially when it comes to complex and large-scale projects, the key issue can be called the choice of the optimal platform, which will ensure reliability and speed in the joint operation of several systems.

ESB - an approach to building distributed corporate information systems. Typically, it includes middleware, which provides the interconnection between different applications for various communication protocols.

3.1 Core ESB Features

There are a number of different ESB products available on the market today. Some, such as WebSphere Message Broker or TIBCO BusinessWorks, are traditional EAI products that have been re-factored to offer ESB-like functionality, but still function in a broker-like manner.

Others, such as MuleSoft's Mule ESB, are designed from the ground up using open messaging and integration standards to implement the ESB model.

Integration between database tables of various applications will be implemented on the basis of the Mule service bus, which allows you to receive and transmit data in a specific format, and the terminal modules are adapters to application databases.

Mule ESB is an easy and flexible platform easily adaptable to the existing infrastructure, as well as reliable to ensure the smooth operation of the largest and most demanding enterprise SOA implementers.



3.2 The Advantages of ESB

Here's a look at the advantages offered by an ESB approach to application integration:

- **Lightweight:** because an ESB is made up of many interoperating services, rather than a single hub that contains every possible service, ESBs can be as heavy or light as an organization needs them to be, making them the most efficient integration solution available.
- **Easy to expand:** If an organization knows that they will need to connect additional applications or systems to their architecture in the future, an ESB allows them to integrate their systems right away, instead of worrying about whether or not a new system will not work with their existing infrastructure. When the new application is ready, all they need to do to get it working with the rest of their infrastructure is hook it up to the bus.

- **Scalable and Distributable:** Unlike broker architectures, ESB functionality can easily be dispersed across a geographically distributed network as needed. Additionally, because individual components are used to offer each feature, it is much simpler and cost-effective to ensure high availability and scalability for critical parts of the architecture when using an ESB solution.
- **SOA-Friendly:** ESBs are built with Service Oriented Architecture in mind. This means that an organization seeking to migrate towards an SOA can do so incrementally, continuing to use their existing systems while plugging in re-usable services as they implement them.
- **Incremental Adoption:** At first glance, the number of features offered by the best ESBs can seem intimidating. However, it's best to think of the ESB as an integration "platform", of which you only need to use the components that meet your current integration needs. The large number of modular components offers unrivaled flexibility that allows incremental adoption of an integration architecture as the resources become available, while guaranteeing that unexpected needs in the future will not prevent ROI.

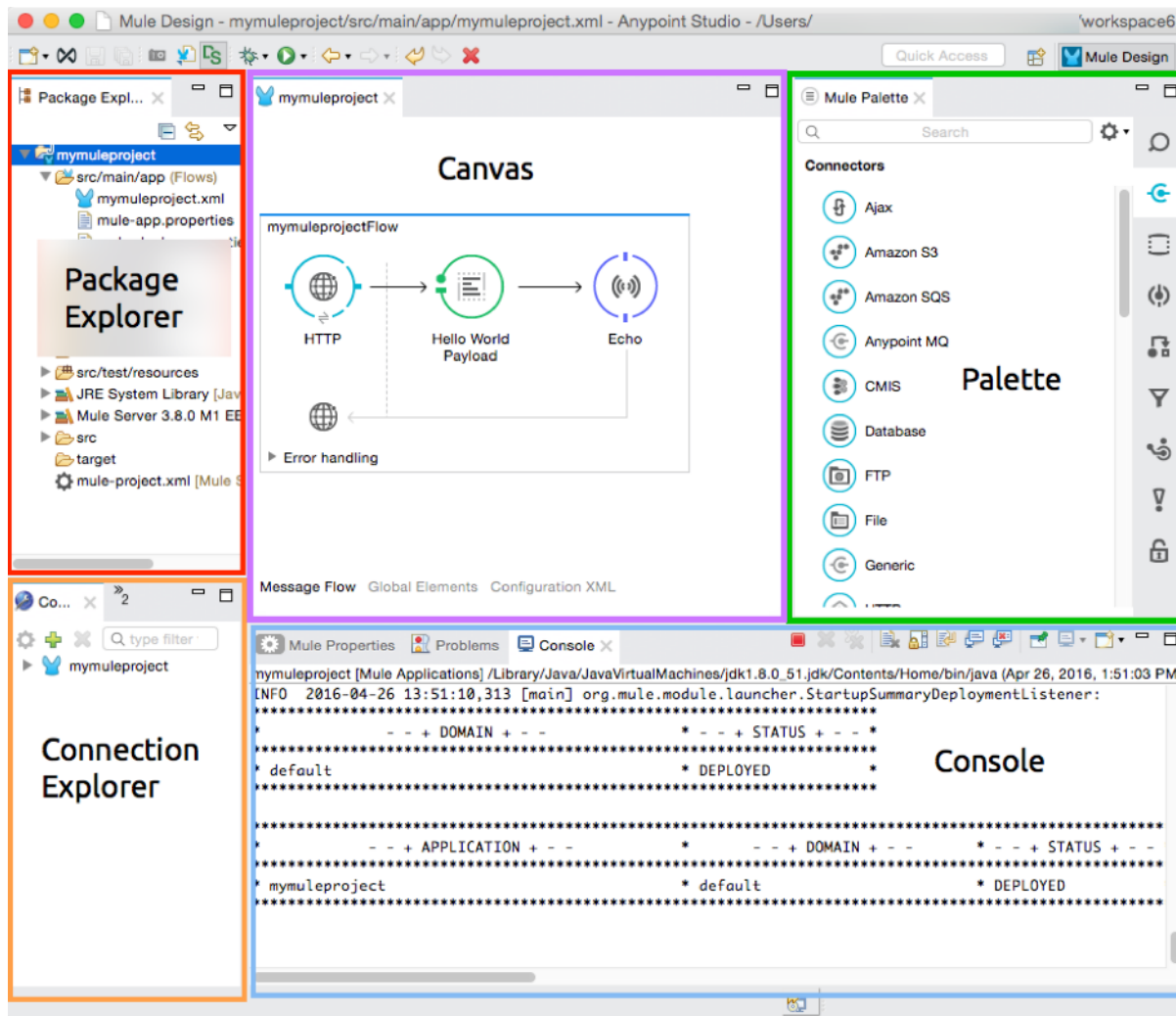
Chapter 4: DEVELOPING INTEGRATIONS IN MULESOFT

For us to integrate data between magento and salesforce, we will make use of MULE ESB.

In this chapter we will quickly see in brief the tools available for developing the integration application in MuleSoft

4.1 Anypoint Studio

Anypoint Studio is the graphical editor that is used to develop the Mule projects. This is built on top of the Eclipse IDE and hence the entire editor would give a very familiar feel if you have been acquainted with Eclipse or Eclipse based editor before. The editor has a drag and drop canvas on which the flows are designed. All the flows designed using the drag and drop canvas is internally stored as XML files. Hence we can also use the XML editor of the Anypoint Studio to design the projects.



Graphical representation of a mule studio

Let's look into the various components that are available in the studio.

4.1.1 Package Explorer

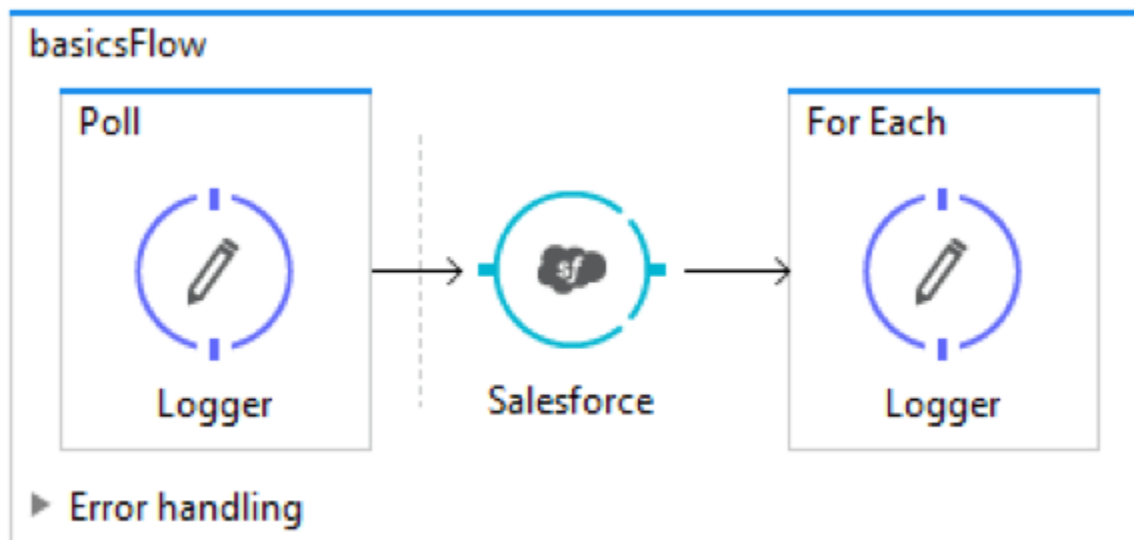
As shown above, on the left hand side of the canvas in the Anypoint Studio is the Package Explorer. It contains all the content of a mule application that we develop. Whenever we create a new Mule project, a default flow is created.

4.1.2 Canvas

The Canvas is the area where we design the flow. It is a graphical editor where in we can drag and drop the elements from the **Mule Palette** from the right side. The palette contains the basic building blocks of a flow. It contains various predefined components which can be used. With a closer look at the canvas we can see that there are three tabs at the bottom, namely:

1. **Message Flow:** Drag and drop interface to build flows.
2. **Global Elements:** Contains the elements which can be reused. It is generally a good practice to keep all the connection related attributes and configurations of the entire project in a single global elements flow.
3. **Configuration XML:** In addition to the graphical editor, the studio also provides an XML editor. All the building blocks that are placed on to the canvas are represented by their equivalent XML structure in the `configuration.xml` file.

As shown in the below two snippets, the XML is the exact equivalent of the flow that is dragged and dropped. So, we can edit either in the canvas or the xml editor.



Canvas

```
basics x
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <mule xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc" xmlns="http://www.mulesoft.org/schema/mule/core" xmlns:dc
4   xmlns:spring="http://www.springframework.org/schema/beans"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6   xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/sprir
7 http://www.mulesoft.org/schema/mule/core http://www.mulesoft.org/schema/mule/core/current/mule.xsd
8 http://www.mulesoft.org/schema/mule/sfdc http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd">
9   <sfdc:config name="Salesforce_Basic_Authentication" username="${sfdc.username}" password="${sfdc.password}" secu
10 <flow name="basicsFlow">
11   <poll doc:name="Poll">
12     <logger message="Start of the flow" level="INFO" doc:name="Logger"/>
13   </poll>
14   <sfdc:query config-ref="Salesforce_Basic_Authentication" query="dsql:Select id,Name from Account" doc:name='
15 <foreach doc:name="For Each">
16   <logger message="Account name is #[payload.name]" level="INFO" doc:name="Logger"/>
17 </foreach>
18 </flow>
19 </mule>
20
```

Configuration.xml

4.1.3 Mule Palette

The studio comes with a set of predefined building blocks that can be dragged on to the canvas to build the applications. These building blocks range from a simple **File Connector** to Enterprise connectors such as **SAP, PeopleSoft**, etc. The components on the Mule Palette are classified into the following types:

1. **Connectors** - used to interact with the third party APIs or Systems
2. **Scopes** - used to define the area or boundary until which the flow and the messages in the flow are visible
3. **Components** - used to execute the business logic
4. **Transformers** - used to modify or massage the data
5. **Filters** - used to conditionally pass the data
6. **Flow control** - used to route or broadcast the same message to more than one processor component.
7. **Error handling** - used to handle Exceptions
8. **Security** - used to provide secure access to information, applications and services

Now that we have understood the various parts of the Anypoint Studio, let's try to understand a bit about how the data moves around in Mule.

4.1.4 Mule Message

Mule message is the data that passes through the application via one or more flows. The message is composed of the following:

1. **Header** - contains the metadata about the message.
2. **Payload** - contains the actual data that would be acted upon. The entire mule message is encapsulated in an object called as the *Mule Message object*. This object contains the following:
 - Mule message
 - **Header**
 - Inbound properties
 - Outbound properties
 - **Payload**
 - Variables
 - Attachments
 - Exception payload

The Inbound properties are the immutable ones and are set by the message sources, whereas the outbound properties are set in the flow and these might act as inbound properties for the next connector.

The variables like in any other programming language are used to store some useful data. The variables are further classified into the following types:

1. **Flow variables** - specific to a flow
2. **Session variables** - available to all the flows within the app
3. **Record variables** - used in case of records of a batch context

4.2 Designing a Mule Application

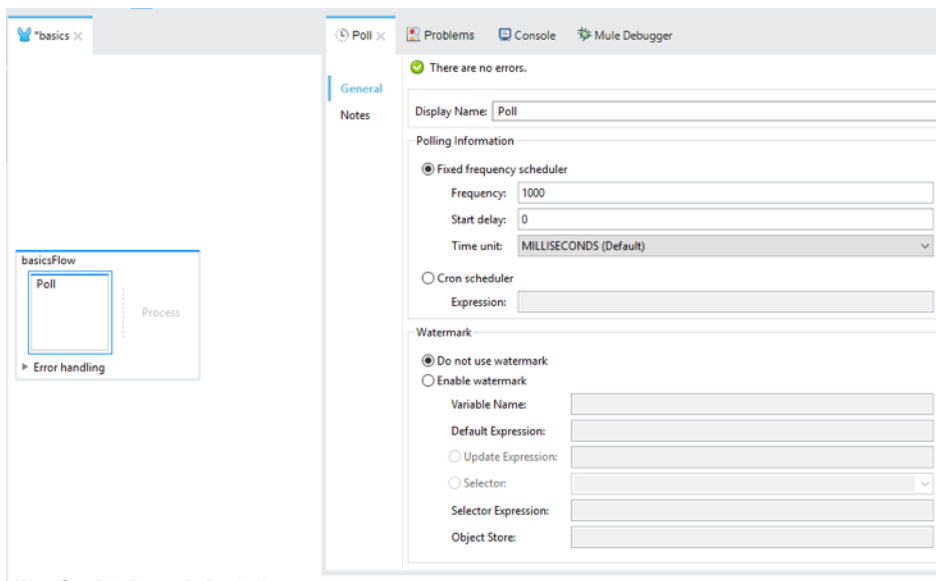
The development of a Mule application starts with a Flow. Flow is the logical component which acts as a starting point of an application. The flow kicks in when it receives an input or an incoming message. Flows can be configured to be either *Synchronous* or *Asynchronous*.



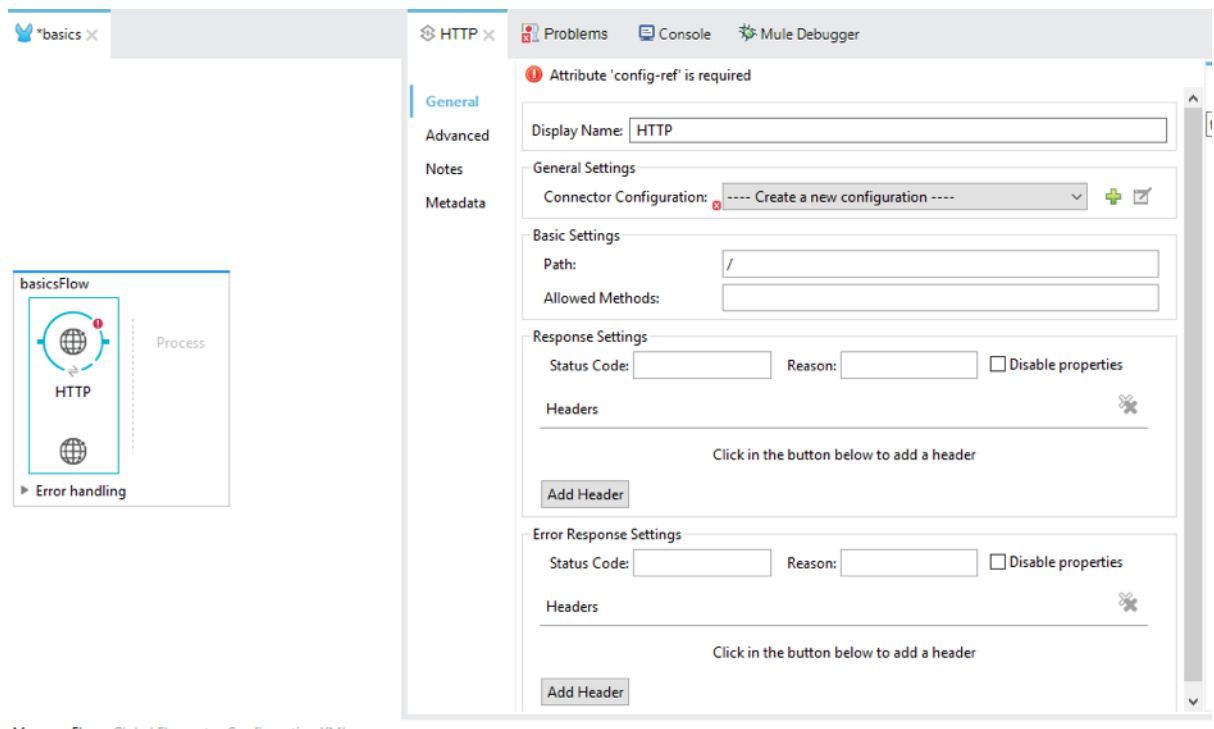
As can be seen from the above figure, the Flow contains two parts. A message **Source** and a **Processing** unit. The flows can also be configured to have an error handling mechanism. There can be any number of flows defined in an application, and we can separate them on a logical basis, e.g. consider a flow which processes the data from a REST source, another one which processes data from a SOAP endpoint, one from database and so on. There are unit names that can be used as flow reference and invoke one flow from another. Flows without a specific message source are termed as **Private** flows.

There are a couple of ways in which we can invoke or kick in the process:

1. **Polling:** We can define the Mule to poll an endpoint at regular intervals and then receive the data and process accordingly.



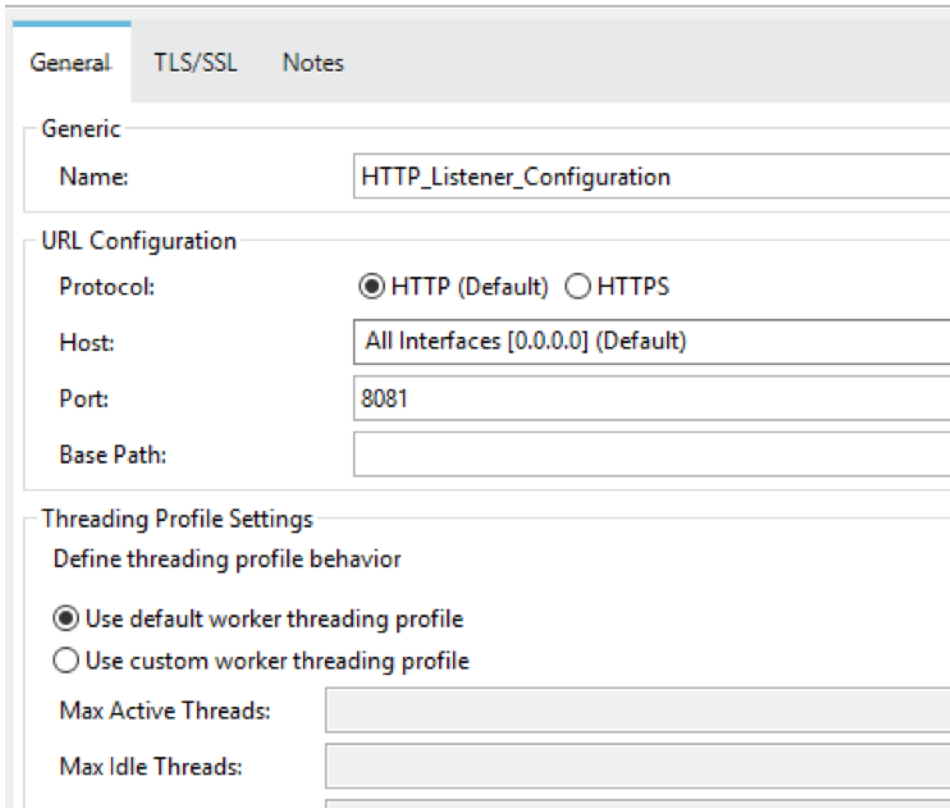
2. **HTTP Listener:** The mule flow can be made to listen on to a specific port and host. So whenever a request is received on the specific port, the mule flow starts.



The HTTP Component requires a connector configuration that defines the Hostname, the Proxy details, use of TLS etc. as shown below

HTTP Listener Configuration

Create reusable HTTP listener



General | TLS/SSL | Notes

Generic

Name: HTTP_Listener_Configuration

URL Configuration

Protocol: HTTP (Default) HTTPS

Host: All Interfaces [0.0.0.0] (Default)

Port: 8081

Base Path:

Threading Profile Settings

Define threading profile behavior

Use default worker threading profile

Use custom worker threading profile

Max Active Threads:

Max Idle Threads:

Commonly used building blocks:

The following section defines some of the commonly used connectors for simple integrations.

1. Salesforce Connector

- Authentication Methods:
 - Basic Authentication
 - OAuth 2.0 JWT Bearer
 - OAuth 2.0 SAML Bearer
 - OAuth v2.0
- Salesforce actions:
 - Insert / Upsert/ Update / Delete / Query
 - Getting Session ids

- Streaming API
- Calling apex services
- Getting batch job details
- Converting lead
- Creating metadata, Deploying metadata

2. Database

The database connector can be used to connect to the databases and perform operations such as select, update, insert.

3. Set Payload

There can be instances when we would want to modify the payload as per the need, or set the payload to a specific value. In such cases we can use the set payload to define the payload to a new value. In general whenever we use any of the transformers, the payload is usually modified. Also, when an outbound connector is used, the response received from the connector becomes the new payload.

4. HTTP Request

The http request configuration can be used in case of REST API services. This connector is used to invoke the outbound APIs. Whatever is defined as the payload prior to this step becomes the body of the API call in case of POST methods. The parameters such as Query parameters, URI parameters, and Headers can all be configured.

Chapter 5: MAP YOUR SYSTEMS, OBJECTS AND FIELDS

With CRM, you need to know Salesforce is the best CRM for Magento platform. Salesforce is full-featured CRM software for all types of businesses. As all-in-one software, Salesforce offers everything you need to find and keep customers, close sales and grow your business.

In order to synchronize data fields from Magento to your Salesforce CRM account correctly, you need to create the mapping for these fields first.

There are 4 objects that can be synchronized: Contact, Account, Order, and Opportunity.

MAGENTO (Contact & Order)

Contact

The **customerCustomerEntityToCreate** content is as follows:

| Name | Type | Description |
|-------------|-------------|---|
| Email | String | Customer email |
| Firstname | String | Customer first name |
| Lastname | String | Customer last name |
| Website_id | Int | Website ID |
| Store_id | Int | Store ID |
| Group_id | Int | Group ID |
| Prefix | String | Customer prefix (optional) |
| Suffix | String | Customer suffix (optional) |
| Dob | String | Customer date of birth(optional) |
| Tax vat | String | Customer tax/VAT number (optional) |
| Gender | Int | Customer gender: 1- Males, 2 –Female (optional) |
| Middlename | String | Customer middle name/initial (optional) |

Order

Allows you to retrieve the list of orders.

The **salesOrderEntity** content is as follows:

| Name | Type | Description |
|--------------|-------------|--------------------|
| Increment_id | String | Increment ID |
| Parent_id | String | Parent ID |

| | | |
|----------------------|--------|-------------------------------------|
| Store_id | String | Store ID |
| Created_at | String | Date of creation |
| Updated_at | String | Date of updating |
| Is_active | String | Defines whether the order is active |
| Customer_id | String | Customer ID |
| Tax_amount | String | Tax amount |
| Shipping_amount | String | Shipping amount |
| Discount_amount | String | Discount amount |
| Subtotal | String | Subtotal sum |
| Grand_total | String | Grand total sum |
| Total_paid | String | Total paid |
| Total_refunded | String | Total refunded |
| Shipping_description | String | Shipping description |
| Customer_email | String | Email address of the customer |
| Customer_firstname | String | Customer first name |
| Customer_lastname | String | Customer last name |

SALESFORCE (Account & Opportunity)

Account

| Field Name | Type |
|-------------------|-------------|
| ID | Id |

| | |
|----------------------|-----------|
| Account Number | String |
| Owner ID | Reference |
| Billing Street | Text area |
| Billing City | String |
| Billing State | String |
| Billing Postal Code | String |
| Billing Country | String |
| Created by Id | Reference |
| Created Date | Date time |
| Phone | Phone |
| Shipping Street | Text |
| Shipping City | String |
| Shipping State | String |
| Shipping Postal code | String |
| Shipping Country | String |
| Website | url |

Opportunity

| Field Name | Type |
|--------------------------|-----------|
| Account ID | Reference |
| Amount | Currency |
| Close Date | Date |
| IsClosed | Boolean |
| CreatedByID | Reference |
| CreatedDate | Date time |
| Description | Textarea |
| LastModifiedById | Reference |
| LastModifiedDate | Date time |
| Name | String |
| TotalOpportunityQuantity | Double |
| StageName | Picklist |
| Type | Picklist |

5.1 Mapping Magento Customer Fields to Salesforce Contact Fields

Automatically create a contact in Salesforce each time there is a new customer in Magento

The following fields from Magento Customers and default billing address are migrated to Salesforce Contacts:

- Id
- First Name
- Last Name
- Email
- Mailing Postal Code
- Mailing Street
- Mailing City
- Mailing State
- Mailing Country

5.2 Mapping Magento Customer Fields to Salesforce Account Fields

Automatically create an account oldest contact in Salesforce each time there is a new customer in Magento.

| Magento Customer Account Fields | Salesforce Account Fields |
|--|----------------------------------|
| Name | Name |
| Email | Email_c |
| BillingStreet | BillingStreet |
| BillingCity | BillingCity |
| BillingState | BillingState |
| BillingCountry | BillingCountry |
| ZipPostalCode | BillingPostalCode |
| ShippingStreet | ShippingStreet |
| ShippingCity | ShippingCity |

5.3 Mapping Magento Orders Attributes with Salesforce Opportunity Fields

Whenever a customer places an order, the information will be synced in Salesforce's Orders and Opportunities.

| Magento Order Fields | Salesforce Opportunity Fields |
|----------------------|-------------------------------|
| Name | Name |
| Status | StageName |
| Amount | Amount |
| TotalQty | TotalOpportunityQuantity |
| Probability | Probability |
| Web | LeadSource |
| Close date | CloseDate |
| Order Number | OrderNumber_c |

Integrating Magento with Salesforce consists of web service calls utilizing XML request/response setup over an HTTPS connection. The technical details of this connection such as request headers, error handling, HTTPS connection, etc. are all abstracted from the user to make implementation quick and easy.

Here are the metadata that were used in making the integration a success.

I. <sfmc:create-metadata>

Create metadata: Adds one or more new metadata components to your organization

XML Sample

```
<sfmc:create-metadata config-  
ref "mySalesforceConfig" type "Account">  
  <sfmc:objects>  
    <sfmc:object ref "#[payload]" />  
  </sfmc:objects>  
</sfmc:create-metadata>
```

II. <sfmc:list-metadata>

Retrieves property information about metadata components in your organization

XML Sample

```
<sfmc:list-metadata config-ref="mySalesforceConfig" type="Account"/>
```


This call retrieves property information about metadata components in your organization

III. `<sfdc:upsert-metadata>`

Creates or updates one or more metadata components in your organization

XML Sample

```
<sfdc:upsert-metadata config-ref "mySalesforceConfig" type "Account">
  <sfdc:objects>
    <sfdc:object ref "#[payload]" />
  </sfdc:objects>
</sfdc:upsert-metadata>
```

IV. `<sfdc:create>`

Adds one or more new records to your organization's data.

Take the CloseDate of an Opportunity as an example, if you set that field to a string of value "2011-12-13" it will be sent to Salesforce as a string and operation will be rejected on the basis that CloseDate is not of the expected type.

The proper way to actually map it is to generate a Java Date object, you can do so using Groovy expression evaluator as `#[groovy:Date.parse("yyyy-MM-dd", "2011-12-13")]`.

XML Sample

```
<sfdc:create config-ref "Salesforce3" type "Account">
  <sfdc:objects>
    <sfdc:object>
      <sfdc:inner-object key "BillingStreet"> </sfdc:inner-object>
      <sfdc:inner-object key "BillingCity"> </sfdc:inner-object>
      <sfdc:inner-object key "BillingCountry"> </sfdc:inner-object>
      <sfdc:inner-object key "BillingState"> </sfdc:inner-object>
      <sfdc:inner-object key "Name"> </sfdc:inner-object>
      <sfdc:inner-object key "BillingPostalCode"> </sfdc:inner-object>
    </sfdc:object>
  </sfdc:objects>
</sfdc:create>
```

V. `<sfdc:create-single>`

Adds one new record to your organization's data.

XML Sample

```
<sfdc:create-single config-ref "mySalesforceConfig" type "Account">
  <sfdc:object>
    <Name>          </Name>
    <BillingStreet>          </BillingStreet>
    <BillingCity>          </BillingCity>
    <BillingState> </BillingState>
    <BillingPostalCode> </BillingPostalCode>
    <BillingCountry> </BillingCountry>
  </sfdc:object>
</sfdc:create-single>
```

VI. <sfdc:query>

Executes a query against the specified object and returns data that matches the specified criteria.

This operation can potentially return a large amount of records that might exceed memory capacity.

To prevent this from being a problem, the output of this operation is automatically paginated into an iterable collection of objects. Regardless of the page-size, the iterator will be pushing out registries one at a time and fetching next pages on demand. If you wish to take advantage of the pagination, you must process the output through elements that can handle collections, such as a ForEach scope or DataMapper. In this way, Mule will execute the entire set of registries one at a time, but processing only a batch at a time and thus keeping memory usage from going over limits.

XML Sample

```
<sfdc:query config-ref "mySalesforceConfig" query "SELECT Id FROM Account"/>
```

Chapter 6: TASKS FOR EXECUTION

System consists of two cloud software: **Magento** and **Salesforce CRM**. You should develop the integration application to synchronize data between these systems.

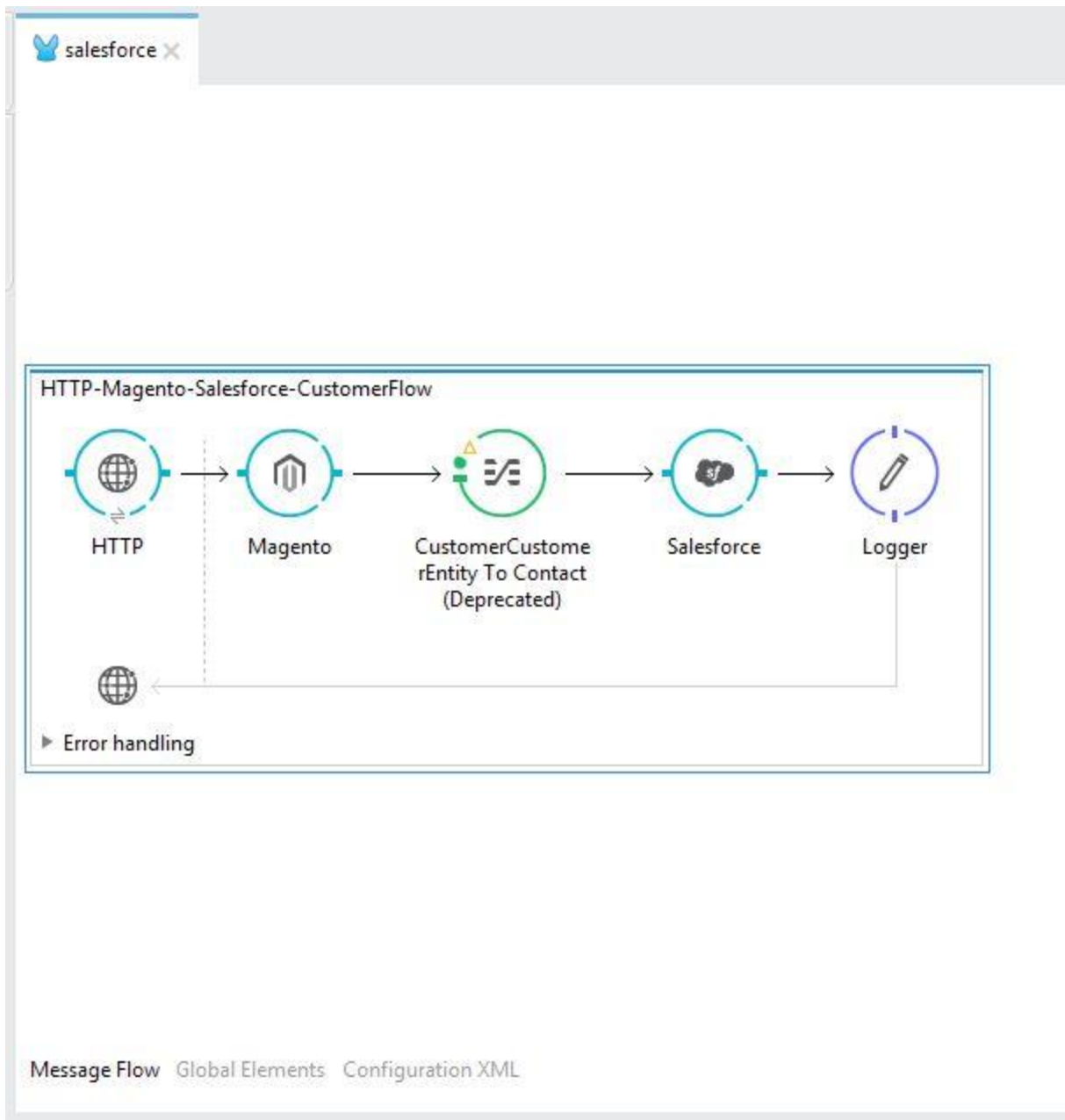
Implement following scenario for the customer synchronization:

1. Write data to salesforce/ magento
 - i. HTTP-Magento-Salesforce-Order
 - ii. HTTP-Magento-Salesforce-Customer
 - iii. HTTP-Salesforce-Magento-Order
 - iv. HTTP-Salesforce-Magento-Customer
 - v. POLL-Magento-Salesforce-Order
 - vi. POLL-Magento-Salesforce-Customer
 - vii. POLL-Salesforce-Magento-Order
 - viii. POLL-Salesforce-Magento-Customer

SOLUTION

1. HTTP-Magento-Salesforce-Customer

The purpose of this flow is create customer in magento and simultaneously been created in salesforce.



The first element, an HTTP Inbound Endpoint, listens on localhost port 8081 (the default) for incoming GET requests. Hitting the listener triggers the flow. Requests to the HTTP Inbound Endpoint must take the form:

`http://localhost:8081?<query>`

The **<query>** part of the request consists of the parameters accepted by the REST API. When the HTTP Inbound Endpoint receives the HTTP request, the **<query>** part of the URL becomes a set of inbound properties. The HTTP

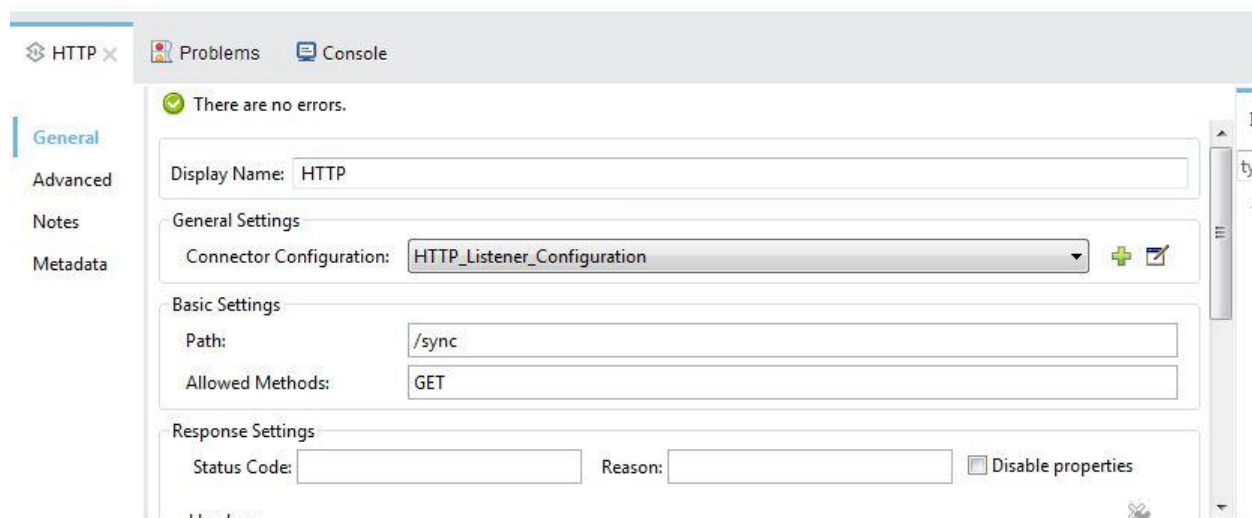
listener passes the message to the next element in the flow, Magento. The Magento uses a set of simple MEL expressions to extract the query parameters from the message, and to construct the full URL for the remote API, including the query parameters.

The Magento passes the JSON (results) it received from the API to a DataMapper Transformer configured to synchronize data to the next flow (salesforce). The transformer sends this object, which contains the JSON data as key=value pairs, to the last element in the flow, a database connector. This connector uses an SQL query with embedded Mule Expression Language expressions to extract specific values from the JSON and insert them into Salesforce database.

How the Configuration was done

I. HTTP Inbound Endpoint

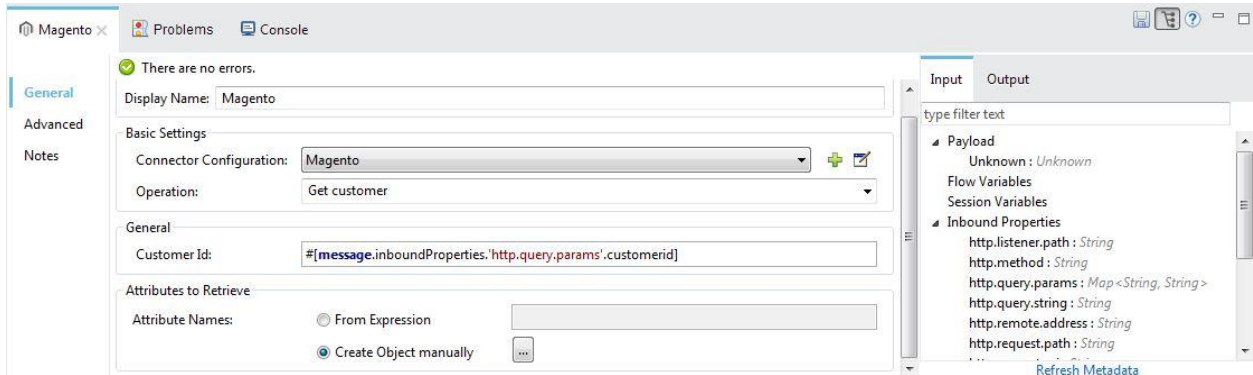
The HTTP service listens on a specific port on our mule server or our local machine. When a request is hit on this address, the Mulesoft flow kicks in, and the flow gets started. As shown above, the flow is started whenever the HTTP service is hit with a GET request on the address specified in the HTTP connector configuration. In this case, since the service is hosted on the localhost, if we host the mule on a web server, then the server's ip can be used to reach the service.



Path: /sync

Method: GET

II. Magento Configurations



Magento is configured as shown above

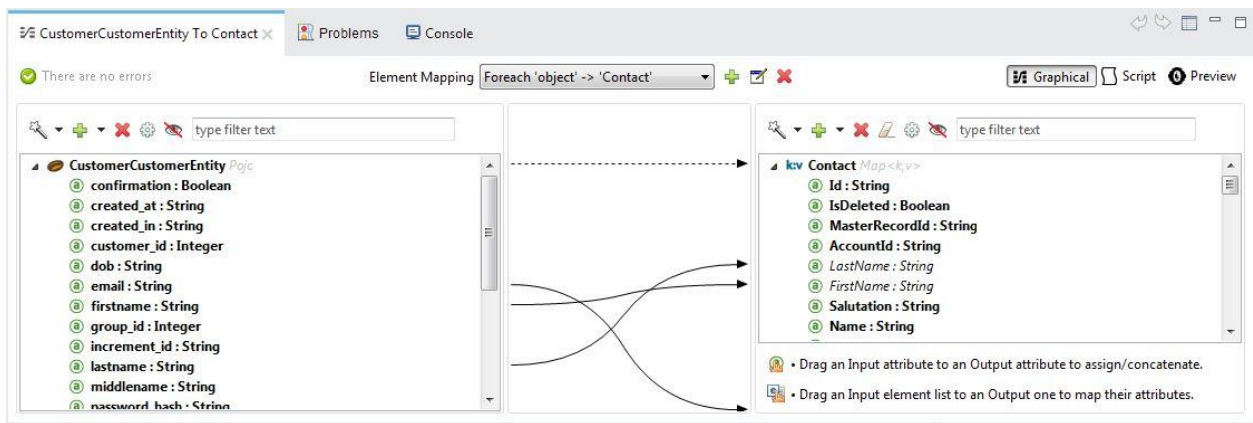
Operation: Get Customer

Customer Id: #[message.inboundProperties.'http.query.param'.customerid]

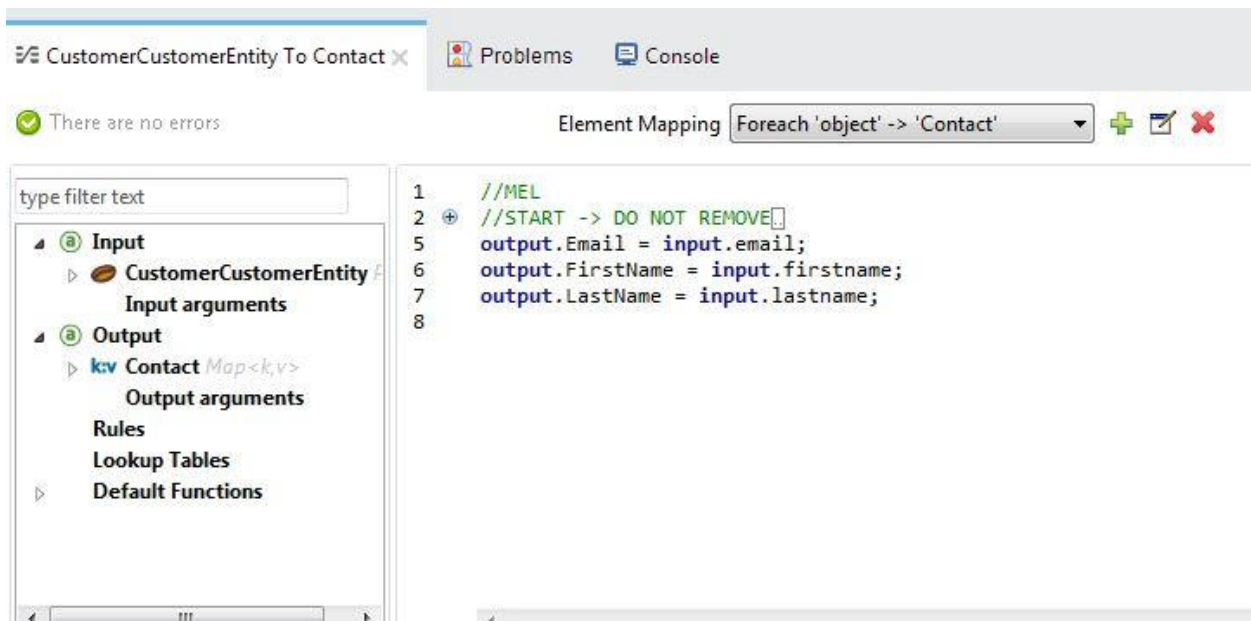
Attribute name is configured to create object manually.

III. DATAMAPPER

Is the process of converting data from one format (e.g. a database file, XML document, or Excel sheet) to another, because data often resides in different locations and formats across the enterprise, data transformation is necessary to ensure data from one application or database is intelligible to other applications and databases, a critical feature for application integration.

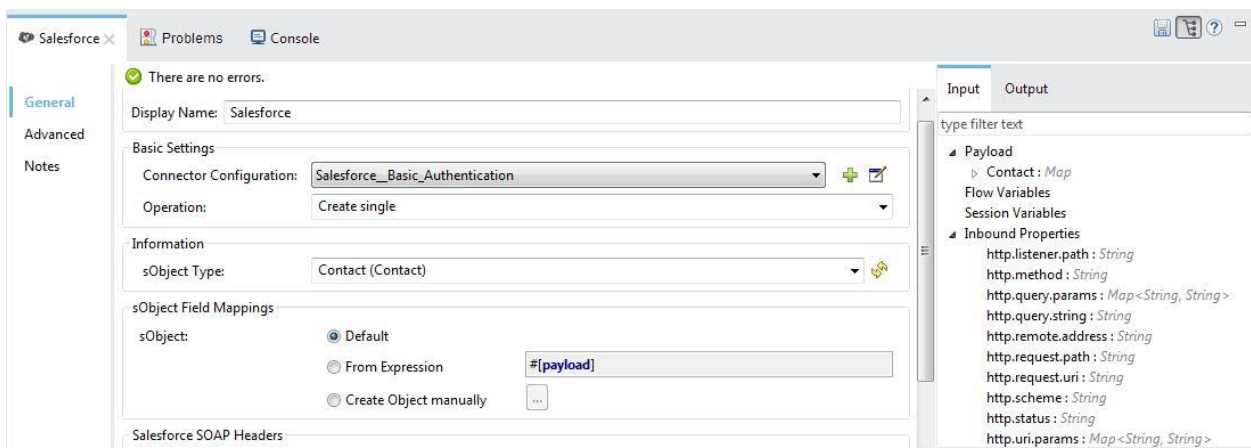


The mapping looks like:



IV. Salesforce configuration

In this salesforce connector, it is configured to create single as an operation from contact as the object type and the field mapping will read from payload meaning, it will get it data from datamapper, the data that datamapper stored from magento when it was queried, the results from datamapper will be added to the database of salesforce.



Running This Example

To trigger the flow in this application, use a Web browser or an HTTP client such as the curl command-line utility to hit the HTTP Inbound Endpoint on localhost port 8081.

I used POSTMAN call my endpoint. (*localhost:8081/sync*)

After that we logged into Salesforce and verify whether the contacts have been created, all the 30 contacts that we have in Magento have been added to Salesforce contacts making it larger.

Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:data-mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:metadata="http://www.mulesoft.org/schema/mule/metadata"
xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:dw="http://www.mulesoft.org/schema/mule/ee/dw"
xmlns:batch="http://www.mulesoft.org/schema/mule/batch"
xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking"
xmlns:json="http://www.mulesoft.org/schema/mule/json"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/json
http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.mulesoft.org/schema/mule/ee/tracking
http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd
http://www.mulesoft.org/schema/mule/batch
http://www.mulesoft.org/schema/mule/batch/current/mule-batch.xsd
http://www.mulesoft.org/schema/mule/ee/dw
http://www.mulesoft.org/schema/mule/ee/dw/current/dw.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
```



```

    <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle2"
securityToken="WJultmwDP0mBzW4mwwLZ8S13" doc:name="Salesforce: Basic
Authentication"/>
    <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0"
port="8081" doc:name="HTTP Listener Configuration"/>
    <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
    <data-mapper:config name="CustomerCustomerEntity_To_Contact"
transformationGraphPath="customercustomerentity_to_contact.grf"
doc:name="CustomerCustomerEntity_To_Contact"/>
    <flow name="BulkApiFlow1">
        <http:listener config-ref="HTTP_Listener_Configuration" path="/sync"
doc:name="HTTP" allowedMethods="GET"/>
        <magento:get-customer config-ref="Magento" doc:name="Magento"
customerId="#[message.inboundProperties.'http.query.params'.customerId]">
            <magento:attribute-names>
                <magento:attribute-name>firstname</magento:attribute-name>
                <magento:attribute-name>lastname</magento:attribute-name>
                <magento:attribute-name>email</magento:attribute-name>
                <magento:attribute-name>customerid</magento:attribute-name>
            </magento:attribute-names>
        </magento:get-customer>
        <data-mapper:transform config-ref="CustomerCustomerEntity_To_Contact"
doc:name="CustomerCustomerEntity To Contact"/>
        <sfdc:create-single config-ref="Salesforce__Basic_Authentication"
type="Contact" doc:name="Salesforce"/>
        <logger level="INFO" doc:name="Logger"/>
    </flow>
</mule>

```

2. HTTP-Salesforce-Magento-Customer

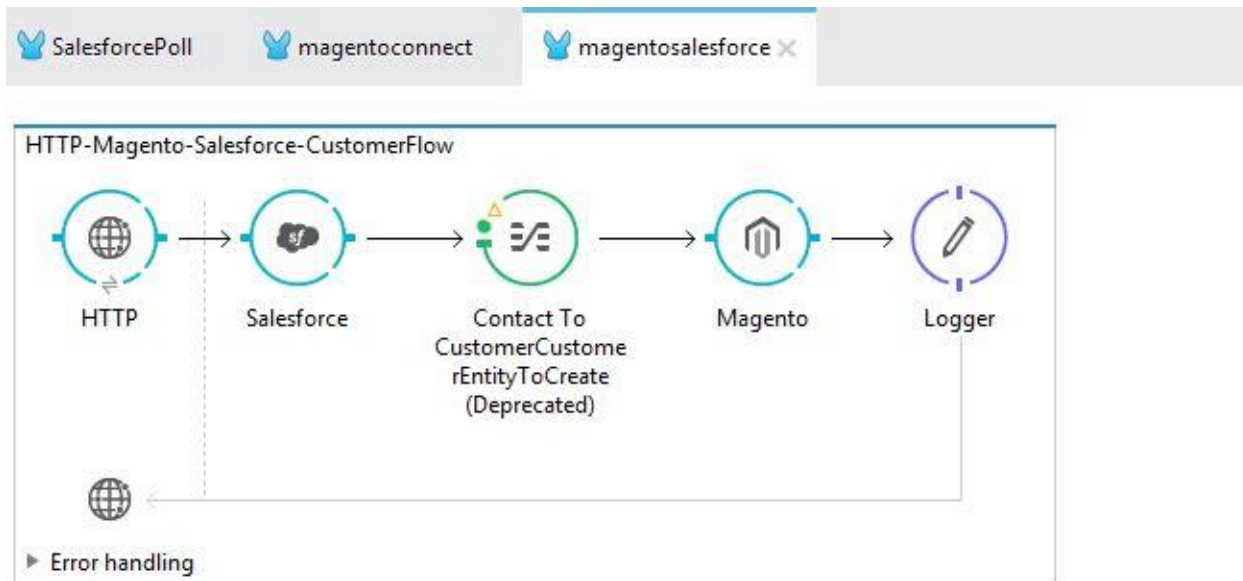
The purpose of this flow is to create customer from salesforce while creating account in salesforce with the details provided.

I. Main Flow

A flow with an HTTP Listener Connector is created, set its **Path** to requests and the **Allowed Methods** field to GET.

Create a Global Element for the Connector, set the **Host** to localhost, leave the **Port** as the default 8081 and set the **Base Path** to synCustomer.

After the HTTP Connector, add a **Salesforce Connector, DataMapper, Magento, and a Logger**.

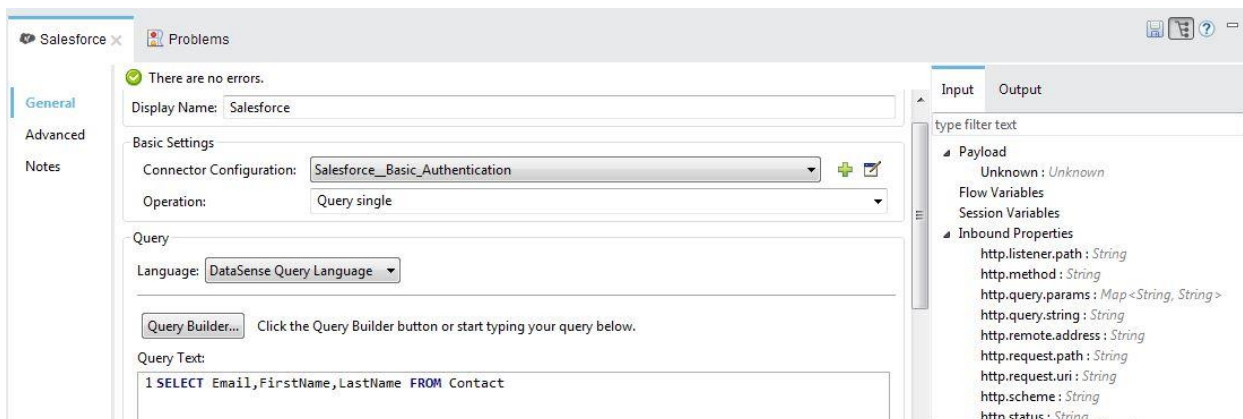


Message Flow Global Elements Configuration XML

II. Salesforce connector

Connects with Salesforce, and performs an operation to extract data.

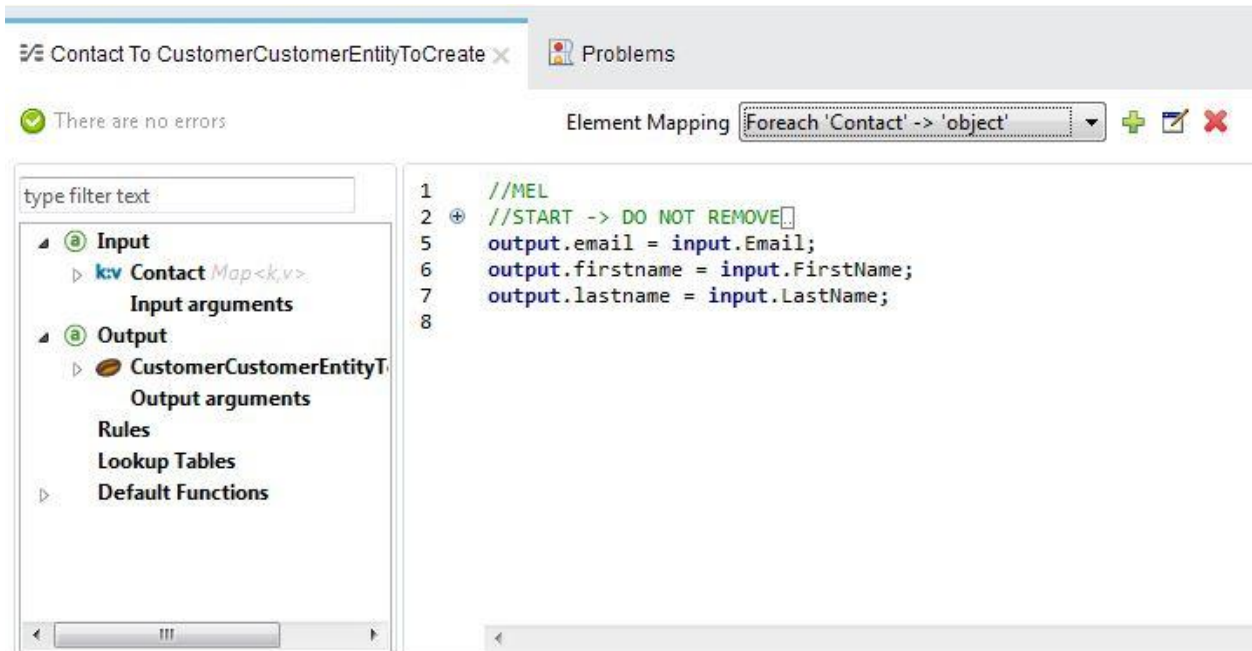
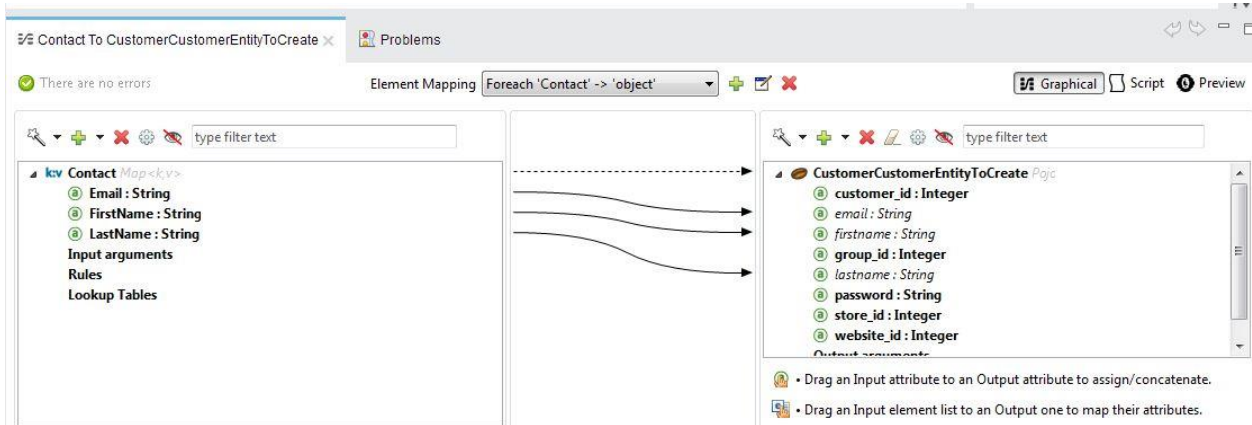
Set the **Operation** to `Query` (see image below).



After you have defined your query, click **OK**. The Query Editor saves, then displays your query in the Query Text field in the Properties Editor.

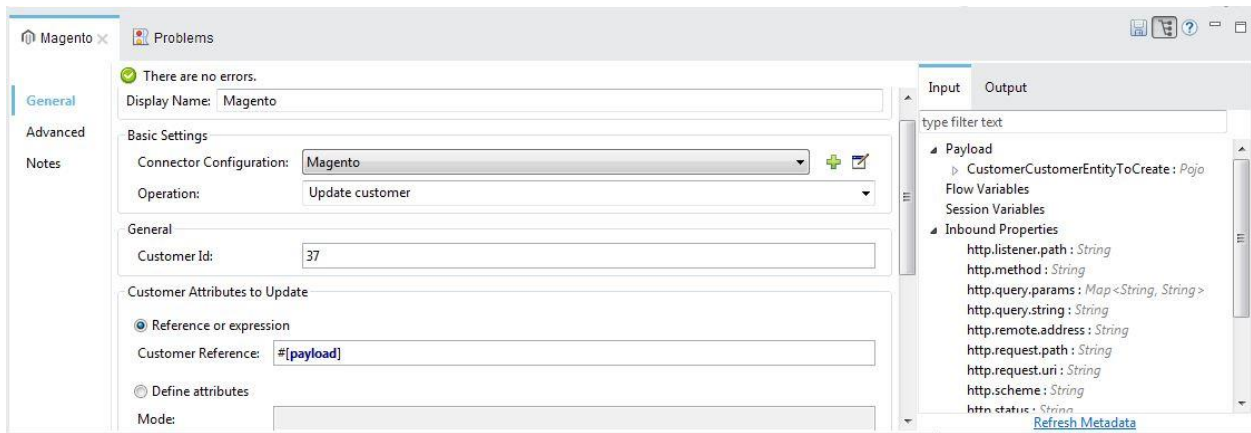
III. DATAMAPPER

As you can see, Anypoint DataMapper has automatically created a top-level mapping called **Foreach 'Contact' → 'object'**, and mapped the field Name since it is identical in the input and output panes. You can now map additional elements, such as Street 1 to street and Zipcode 1 to zipCode.



IV. Magento

Magento settings are simple, after a successful connection with the database, I set the operation to 'update customer', customer id to be updated to '37' and the customer reference to 'Payload' meaning, it will be updating records in magento with information from salesforce.



Console

```

*****
INFO 2017-05-15 13:40:51,154 [main] org.mule.module.launcher.MuleDeploymentService:
+++++++
+ Started app 'magentosalesforce' +
+++++++
INFO 2017-05-15 13:40:51,354 [main] org.mule.module.launcher.DeploymentDirectoryWatcher:
+++++++
+ Mule is up and kicking (every 5000ms) +
+++++++
INFO 2017-05-15 13:40:51,385 [main] org.mule.module.launcher.StartupSummaryDeploymentListener:
*****
* - - + DOMAIN + - - * - - + STATUS + - - *
*****
* default * DEPLOYED *
*****

*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*****
* magentosalesforce * default * DEPLOYED *
*****

INFO 2017-05-15 13:41:09,809 [[magentosalesforce].HTTP_Listener_Configuration.worker.01] org.mule.api.processor.LoggerMessageProcessor: true

```

Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```

<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:data-mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking"
xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:dw="http://www.mulesoft.org/schema/mule/ee/dw"
xmlns:metadata="http://www.mulesoft.org/schema/mule/metadata"
xmlns:json="http://www.mulesoft.org/schema/mule/json"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd

```

```

http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/json
http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/ee/dw
http://www.mulesoft.org/schema/mule/ee/dw/current/dw.xsd
http://www.mulesoft.org/schema/mule/ee/tracking
http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
  <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0"
port="8081" doc:name="HTTP Listener Configuration"/>
  <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
  <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle2"
securityToken="WJultmwDP0mBzW4mwwLZ8S13" doc:name="Salesforce: Basic
Authentication"/>
  <data-mapper:config name="Contact_To_CustomerCustomerEntityToCreate"
transformationGraphPath="contact_to_customerentitytocreate.grf"
doc:name="Contact_To_CustomerCustomerEntityToCreate"/>
  <flow name="HTTP-Magento-Salesforce-CustomerFlow">
    <http:listener config-ref="HTTP_Listener_Configuration" path="/Customer"
allowedMethods="GET" doc:name="HTTP"/>
    <sfdc:query-single config-ref="Salesforce__Basic_Authentication"
doc:name="Salesforce" query="dsql:SELECT Email,FirstName,LastName FROM Contact">
      </sfdc:query-single>
    <data-mapper:transform config-ref="Contact_To_CustomerCustomerEntityToCreate"
doc:name="Contact To CustomerCustomerEntityToCreate"/>
    <magento:update-customer config-ref="Magento" customerId="47"
doc:name="Magento">
      <magento:customer ref="#[payload]"/>
    </magento:update-customer>
    <logger message="#[payload]" level="INFO" doc:name="Logger"/>
  </flow>
</mule>

```

3. HTTP-Magento-Salesforce-Order

i. Main Flow

The next scenario we will go through is querying data from a Magento database based on a HTTP GET request. The requirement of this scenario is to get a

particular order detail from the Magento database with order ID containing a specified string. To implement this scenario, I have created the following flow containing the steps listed below:



The application started by adding HTTP endpoint as the starting of the flow, then follow by magento, DataMapper, Salesforce and Logger connector to the flow to enable us read the data in formatted format.

HTTP Endpoint is configured as seen below:

Path: /order and allow GET method is used

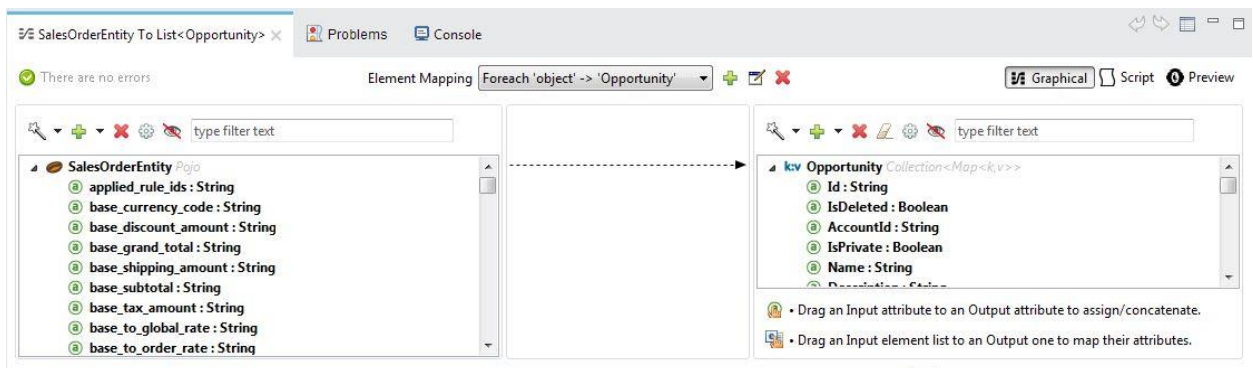
ii. Magento

After a connector configuration was successful, moved on to set the operation of the basic settings to 'get order' from the magento salesforce, where an order ID is specified as shown below:



iii. DATAMAPPER

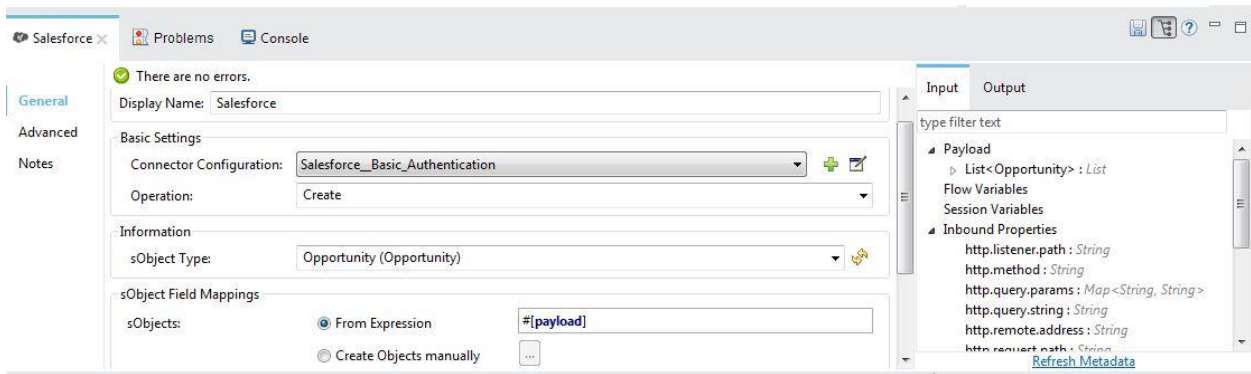
Configure the [DataMapper] so that each variable has its pairing. This output argument sets the value of a variable on each iteration of the DataMapper, overwriting it with a new value each time. Because you ordered your collection so that the last element is the newest, the value that "sticks" at the end of the DataMapper's iteration is what Mule needs for the watermark.



iv. SALESFORCE

After creating salesforce connection. Now select your desired operation like Query, Create, Update and Delete etc.

I selected the Create in operation to create new details of opportunities (orders) from the salesforce.



v. Logger

Add a Logger at the end of your flow. The logger also allows you to verify that the watermark works.

Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:json="http://www.mulesoft.org/schema/mule/json"
xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns:data-
mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/json
http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
    <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0"
port="8081" doc:name="HTTP Listener Configuration"/>
    <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle259"
```



```

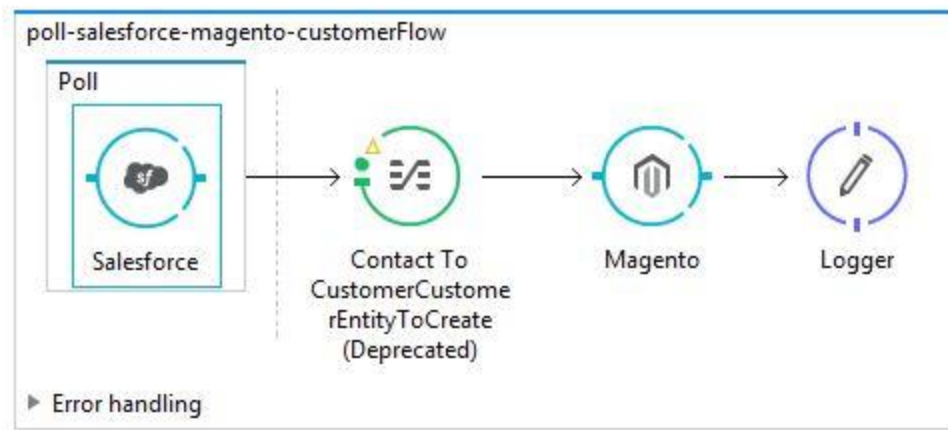
securityToken="dpFpnaArcjG6Kpwei9GFJCNH" doc:name="Salesforce: Basic
Authentication"/>
  <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
  <data-mapper:config name="SalesOrderEntity_To_List_Opportunity_"
transformationGraphPath="salesorderentity_to_list_opportunity_.grf"
doc:name="SalesOrderEntity_To_List_Opportunity_" />
  <flow name="salesforce-magento-orderFlow">
    <http:listener config-ref="HTTP_Listener_Configuration" path="/opportunity"
allowedMethods="GET" doc:name="HTTP"/>
    <sfdc:query config-ref="Salesforce__Basic_Authentication" query="dsql:SELECT
Name,OrderNumber__c,StageName FROM Opportunity" doc:name="Salesforce"/>
    <logger message="#[payload]" level="INFO" doc:name="Logger"/>
  </flow>
  <flow name="salesforce-magento-orderFlow1">
    <http:listener config-ref="HTTP_Listener_Configuration" path="/opportunity1"
allowedMethods="GET" doc:name="HTTP"/>
    <magento:get-order config-ref="Magento" orderId="100000003"
doc:name="Magento"/>
    <data-mapper:transform config-ref="SalesOrderEntity_To_List_Opportunity_"
doc:name="SalesOrderEntity To List<Opportunity>"/>
    <sfdc:create config-ref="Salesforce__Basic_Authentication" type="Opportunity"
doc:name="Salesforce">
      <sfdc:objects ref="#[payload]"/>
    </sfdc:create>
    <logger message="#[payload]" level="INFO" doc:name="Logger"/>
  </flow>
</mule>

```

4. POLL-Salesforce-Magento-Customer

i. Main Flow

The main objective of this flow is to pull salesforce records from it database and parse it magento database, the basic settings for the flow is describe below the diagram.



Message Flow Global Elements Configuration XML

ii. Poll Scope Setting

Configure the poll scope according to the diagram below. I have set the frequency to be fired in once a day (1 time per day). In the Watermark section, I am setting the name of watermark; this name will be available to us as flow variable and can be used in every run of flow and will be updated by polling component with the value of *CreatedDate* at the end of each execution of flow. When the flow has finished processing, Mule updates the value of the variable to the value of the **flow variable** by named lastCreationDate. Its default value is the result of evaluating the following expression: #["TODAY"].

Once we reach the last row in salesforce and there are no more results then the process part of flow will not be executed until unless a new data is created in salesforce and we will get the newly created data in the next run of flow i.e. after 1 day.

Poll x Problems Console
 There are no errors.

General
 Notes

Display Name: Poll

Polling Information

Fixed frequency scheduler

Frequency: 1

Start delay: 0

Time unit: DAYS

Cron scheduler

Expression:

Watermark

Do not use watermark

Enable watermark

Variable Name: lastCreationDate

Default Expression: #[TODAY]

Update Expression:

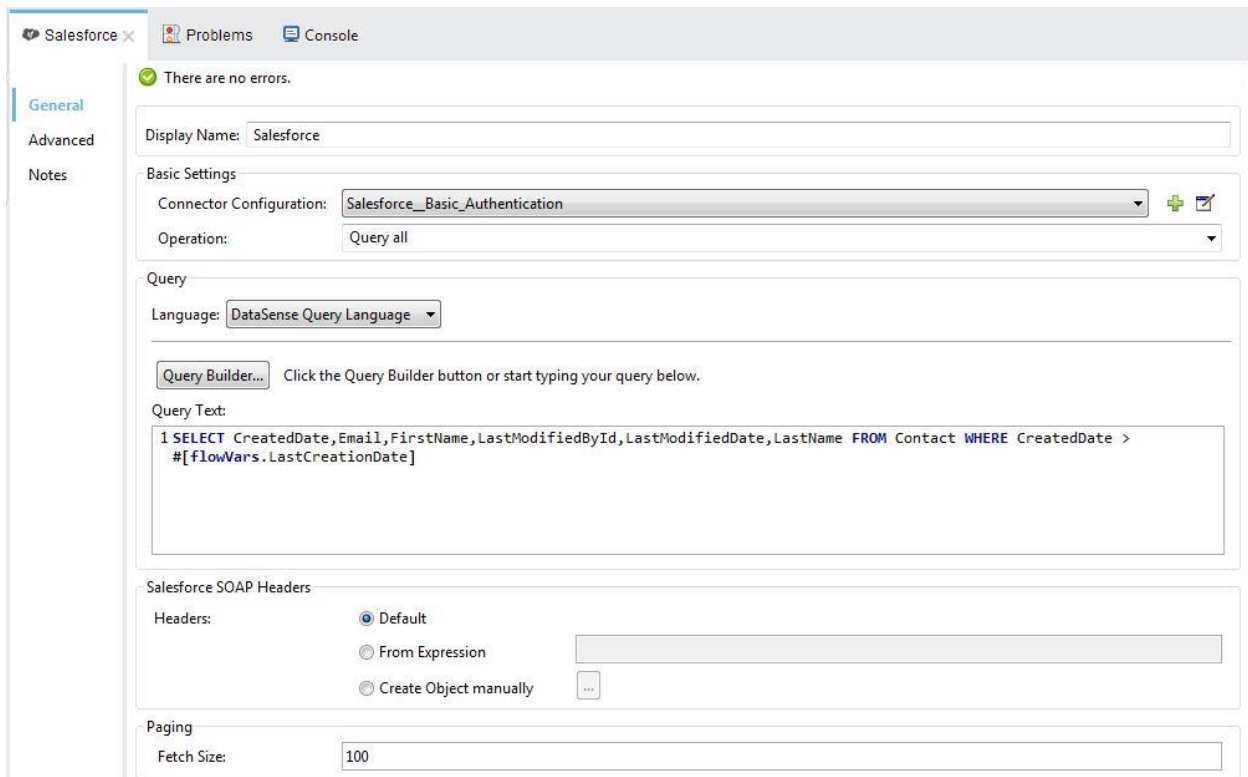
Selector: LAST

Selector Expression: #[payload.CreatedDate]

Object Store:

iii. Salesforce Connector

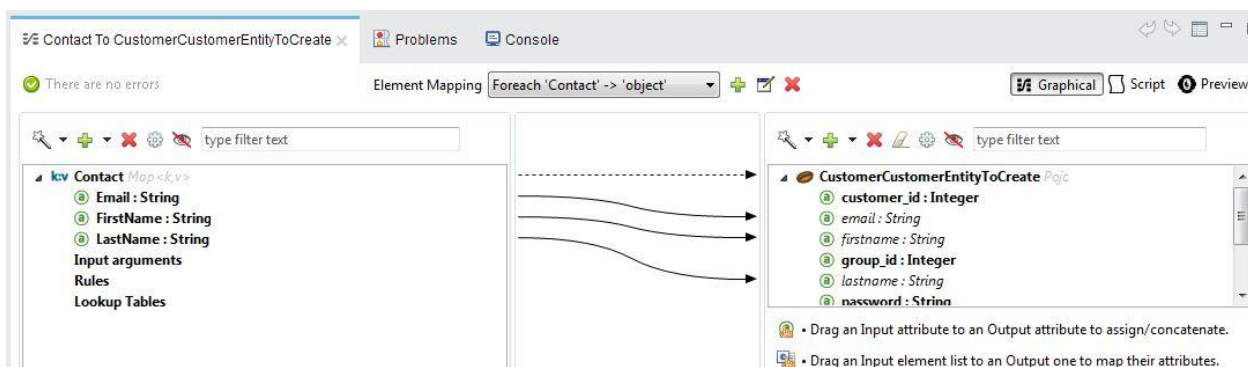
Using the MuleSoft Salesforce Cloud Connector we configure it to point to our Salesforce environment and query for changes to the Contact entity. Using DataSense, we configure which data items to pull back into our flow to form the payload message we wish to process.



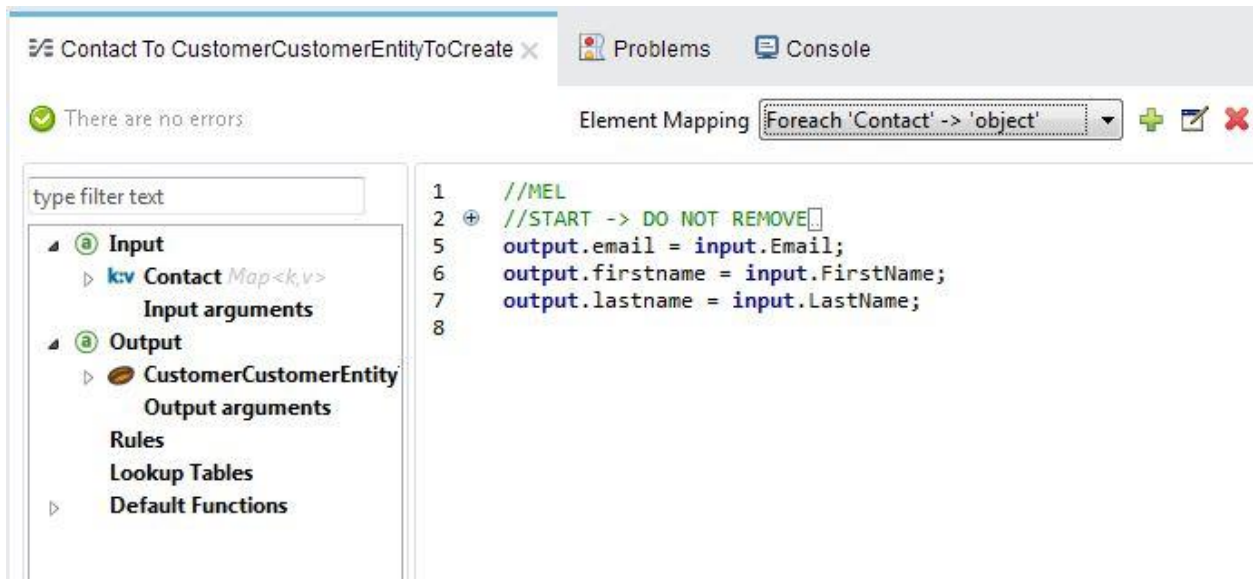
Here I am using the watermark set in polling component to query the salesforce to get the latest values from the salesforce by using **flowVars.lastCreationDate**.

iv. Datamapper

Click **Create mapping** at the bottom of the DataMapper properties editor. DataMapper displays the graphical mapping editor, shown below.

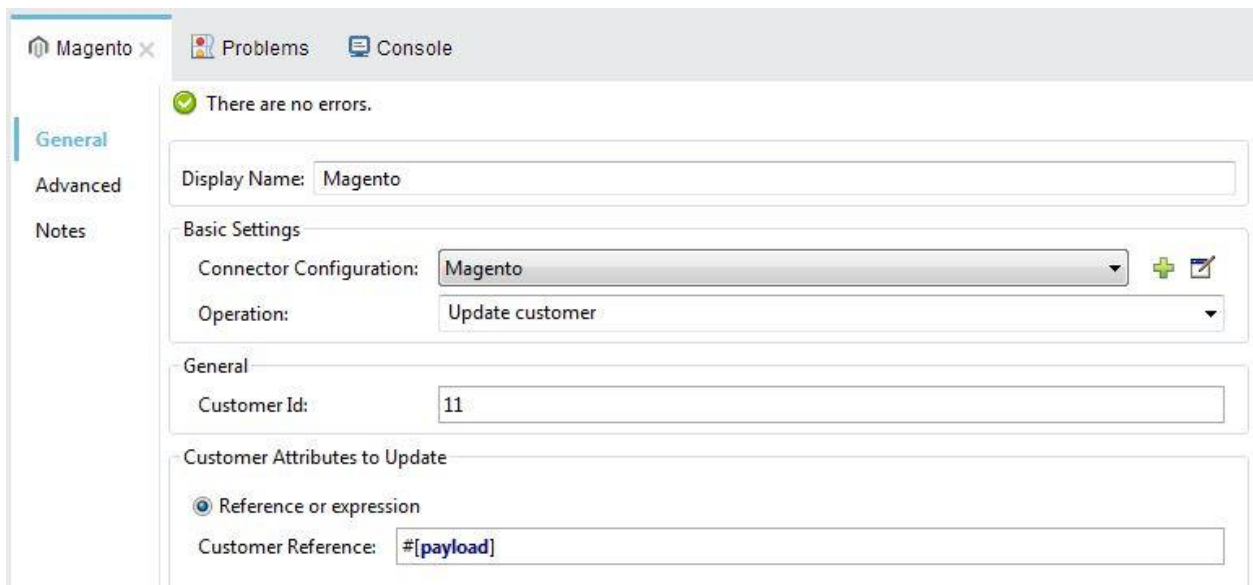


As you can see, DataMapper has automatically mapped the input and output fields, since the field names coincide in the input and output.



v. Magento Connector Settings

The magento settings here is not different from what we have seen from the previous tasks, after a successful configuration settings, we set the operation to update customer from the information that was stored in datamapper which was obtained from salesforce.



Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

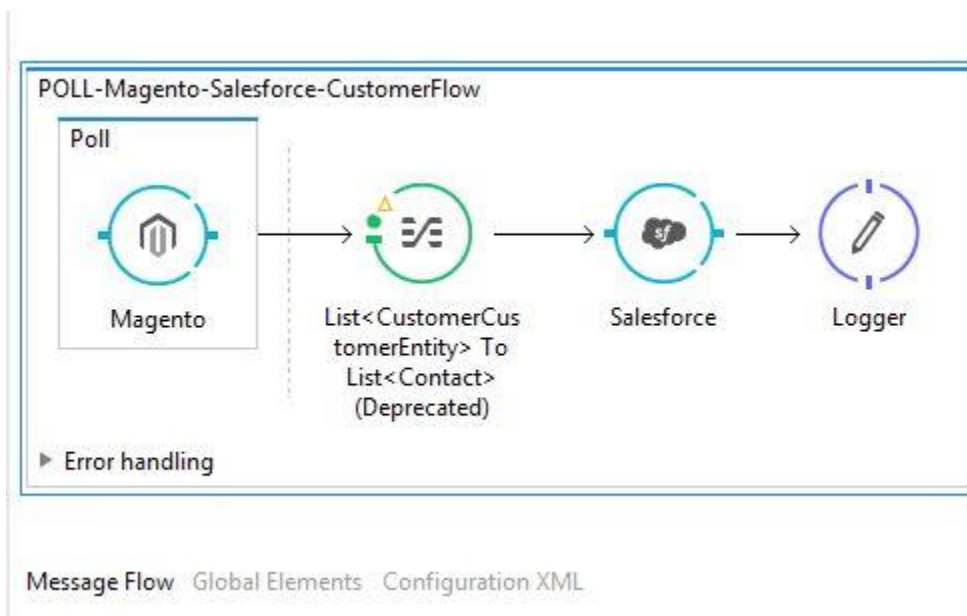
```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:magento="http://www.mulesoft.org/schema/mule/magento" xmlns:data-
mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/ee/tracking
http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
    <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle259"
securityToken="dpFpnaArcjG6Kpwei9GFJCNH" doc:name="Salesforce: Basic
Authentication"/>
    <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
    <data-mapper:config name="List_Contact__To_CustomerCustomerEntityToCreate"
transformationGraphPath="list_contact__to_customercustomerentitytcreate.grf"
doc:name="List_Contact__To_CustomerCustomerEntityToCreate"/>
    <flow name="POLL-Salesforce-Magento-CustomerFlow"
processingStrategy="synchronous">
        <poll doc:name="Poll">
            <fixed-frequency-scheduler frequency="1" timeUnit="DAYS"/>
            <watermark variable="LastCreationDate" default-expression="#['TODAY']"
selector="LAST" selector-expression="#[payload.CreatedDate]"/>
            <sfdc:query-all config-ref="Salesforce__Basic_Authentication"
query="dsql:SELECT
CreatedDate,Email,FirstName,LastModifiedById,LastModifiedDate,LastName FROM Contact
WHERE CreatedDate > #[flowVars.LastCreationDate]" doc:name="Salesforce"/>
        </poll>
        <data-mapper:transform config-
ref="List_Contact__To_CustomerCustomerEntityToCreate" doc:name="List<&lt;Contact&&gt;
To CustomerCustomerEntityToCreate"/>
            <magento:create-customer config-ref="Magento" doc:name="Magento">
                <magento:customer ref="#[payload]"/>
            </magento:create-customer>
            <logger level="INFO" doc:name="Logger" message="#[payload]"/>
        </flow>
</mule>
```

5. POLL-Magento-Salesforce-Customer

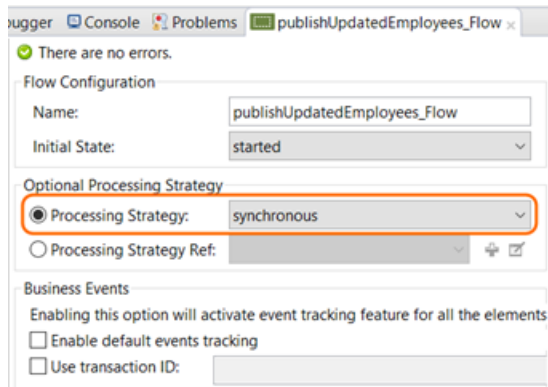
i. Main Flow

Objective of this flow is the same to what we did previously when we used 'HTTP' to query information in magento database but here we used 'POLL', which will done right here in the mule platform, it will query all records of customers in Magento and parse it to Salesforce through DataMapper and later log the details of the flow.



ii. Poll Scope Setting

Poll Scope: the poll scope requires changing the processing strategy of the flow to "Synchronous"

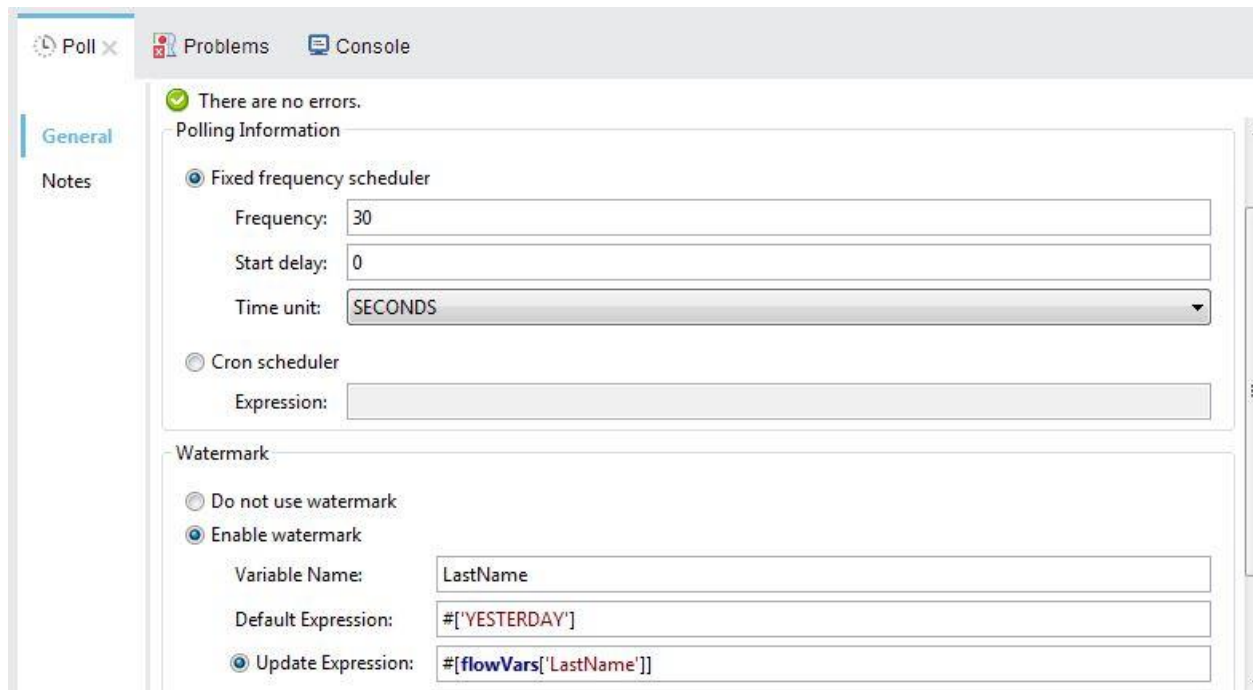


The polling scope won't work if you leave the default processing strategy, which is asynchronous which will be show, the following errors:

Message: watermarking requires synchronous polling

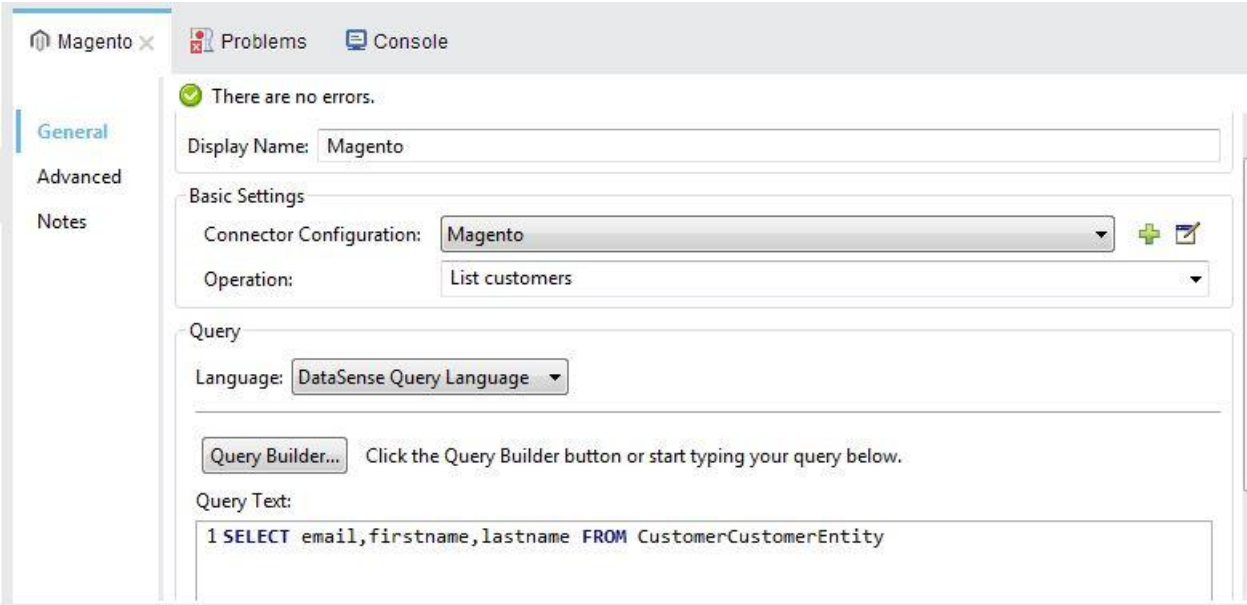
Code: MULE_ERROR-344

For this scenario I configured the polling to occur every 30 seconds. To get only updated records, I am implementing watermarking utilizing the LastName in the payload as shown below:



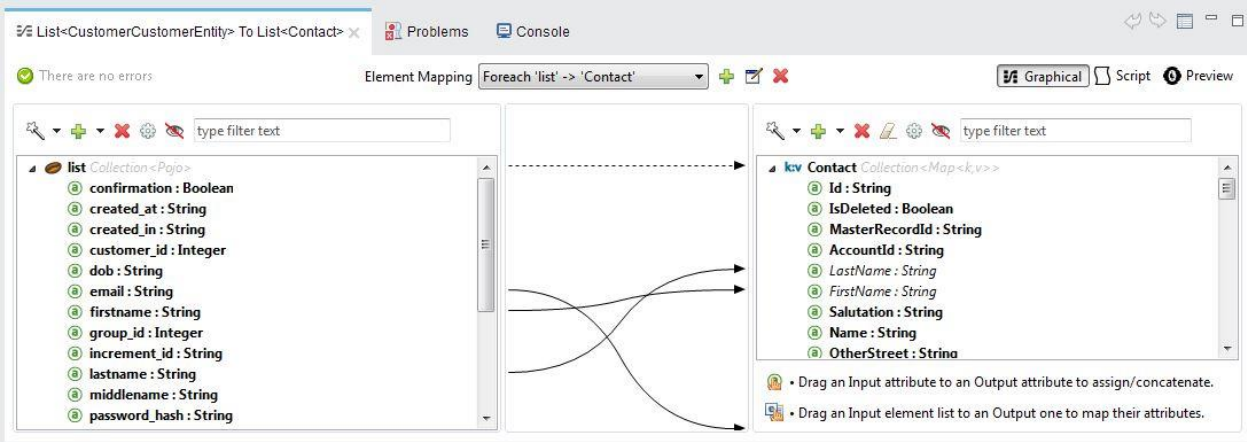
iii. Magento Connector

For the basic settings of this connector, we set operation to ‘lists customer’ (query all) all the records we have in the Magento database using a DataSense Query Language as can be seen below:

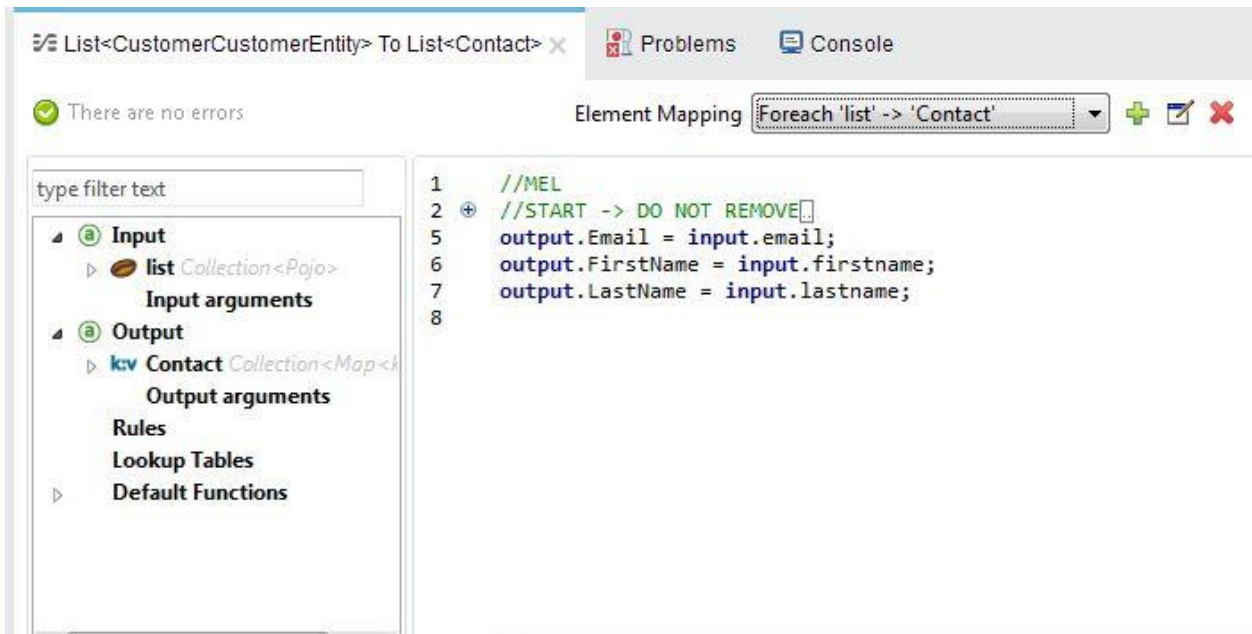


iv. DataMapper

The mapping is created. Next you must tell DataMapper what input field matches what output field. Notice that there already is an arrow joining **LastName, Email and FirstName** on both the input and output schemas, as both fields have the same name, DataMapper correctly assumed they were meant to be mapped together.

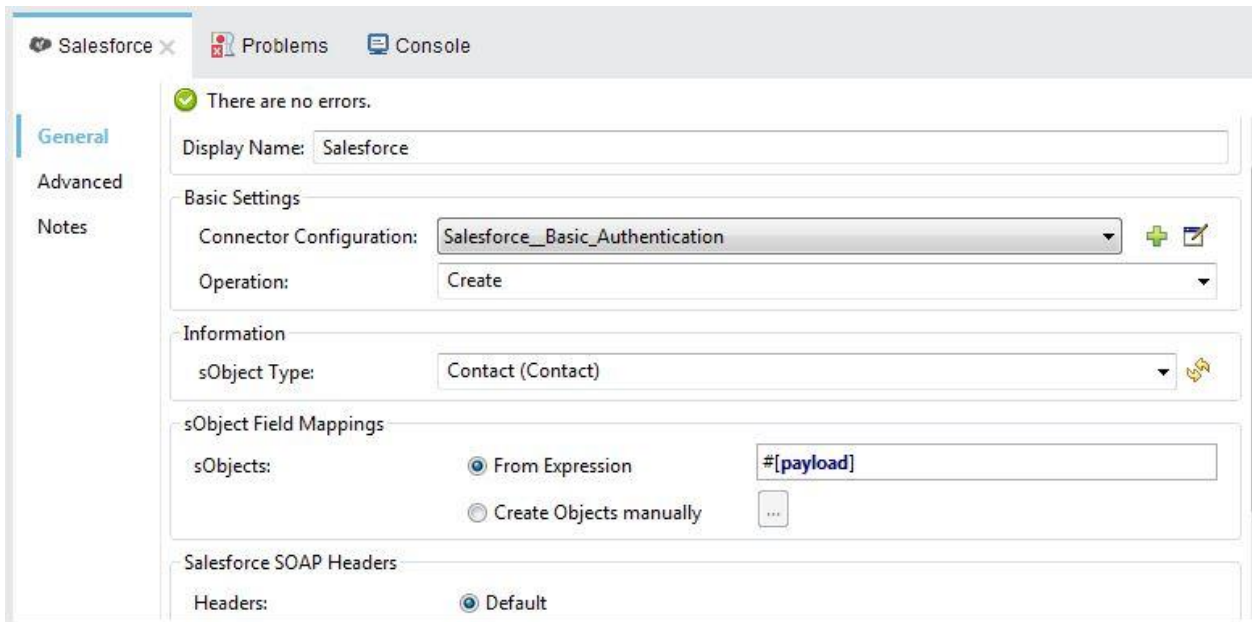


In this case, a mapping flow receives as input an XML document with lists of customers and contacts with their contact information, and generates a JSON document with a list of people and phone numbers. The input data looks like this:



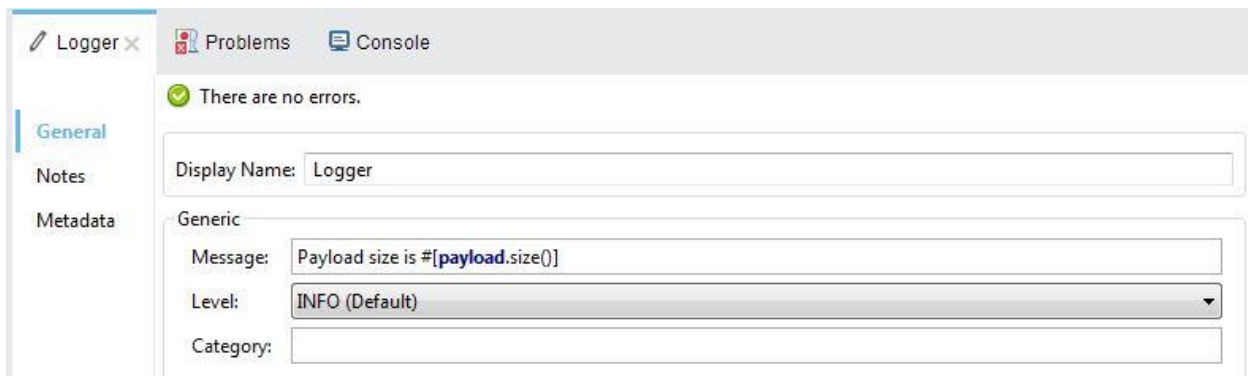
v. Salesforce Connector

Configure the connector according to the following screenshot. So it will create new contact in salesforce from the payload it receives from Magento



vi. Logger

Just as a way to test the flow and a message is placed inside to show the size of the payload:



Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:data-mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking"
xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/tracking
http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd">
  <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0"
port="8081" doc:name="HTTP Listener Configuration"/>
  <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
  <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle259"
securityToken="dpFpnaArcjG6Kpwei9GFJCNH" doc:name="Salesforce: Basic
Authentication"/>
  <data-mapper:config name="List_CustomerCustomerEntity__To_List_Contact_"
transformationGraphPath="list_customercustomerentity__to_list_contact_.grf"
doc:name="List_CustomerCustomerEntity__To_List_Contact_"/>
```

```

<flow name="POLL-Magento-Salesforce-CustomerFlow"
processingStrategy="synchronous">
  <poll doc:name="Poll">
    <fixed-frequency-scheduler frequency="30" timeUnit="SECONDS"/>
    <watermark variable="LastName" default-expression="#['YESTERDAY']"
update-expression="#[flowVars['LastName']]/>
    <magento:list-customers config-ref="Magento" filter="dsl:SELECT
email,firstname,lastname FROM CustomerCustomerEntity" doc:name="Magento"/>
  </poll>
  <data-mapper:transform config-
ref="List_CustomerCustomerEntity_To_List_Contact_"
doc:name="List<CustomerCustomerEntity> To List<Contact>"/>
  <sfdc:create config-ref="Salesforce_Basic_Authentication" type="Contact"
doc:name="Salesforce">
    <sfdc:objects ref="#[payload]"/>
  </sfdc:create>

  <logger message="Payload size is #[payload.size()]" level="INFO"
doc:name="Logger"/>
</flow>

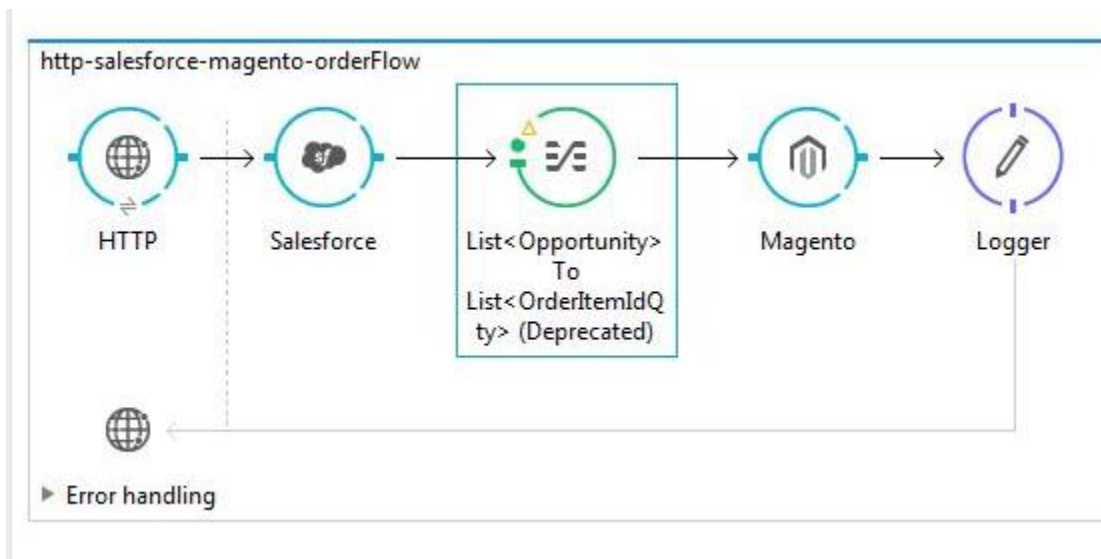
</mule>

```

6. HTTP-Salesforce-Magento-Order

i. Main Flow

The main of this flow is to query salesforce database for order (opportunity) and parse the data to magento database as shown below:



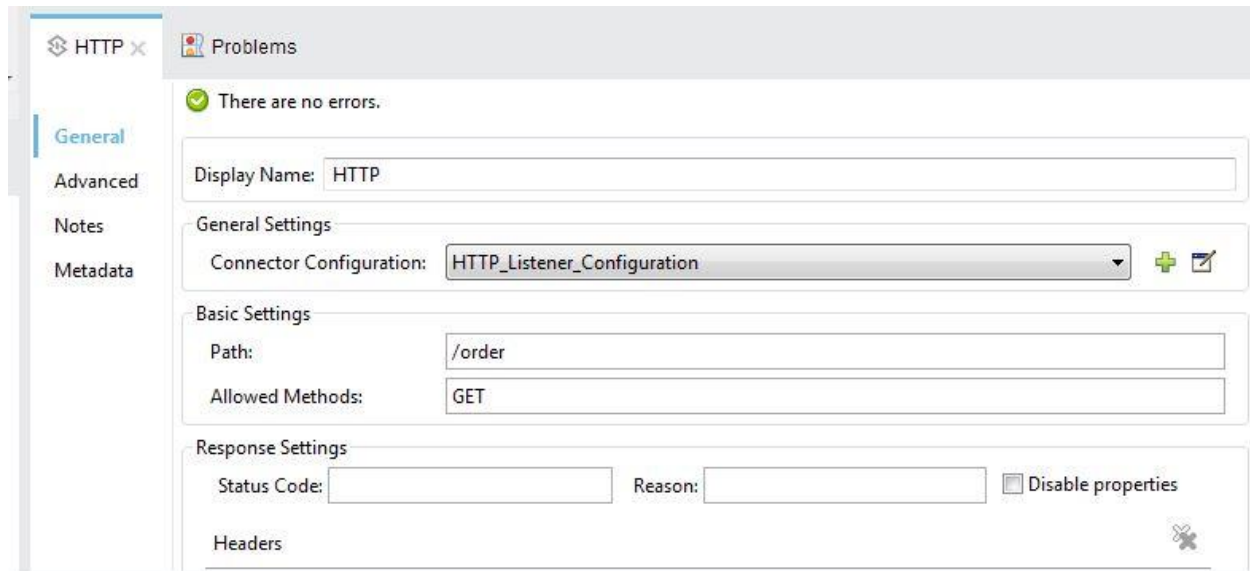
ii. HTTP Listener

The HTTP endpoint is configured as shown below:

Connector configuration: HTTP_Listener_Configuration (localhost:8081)

Path: /order (the specific path to this flow where it will listen to any instruction given to it).

Allowed method: GET.



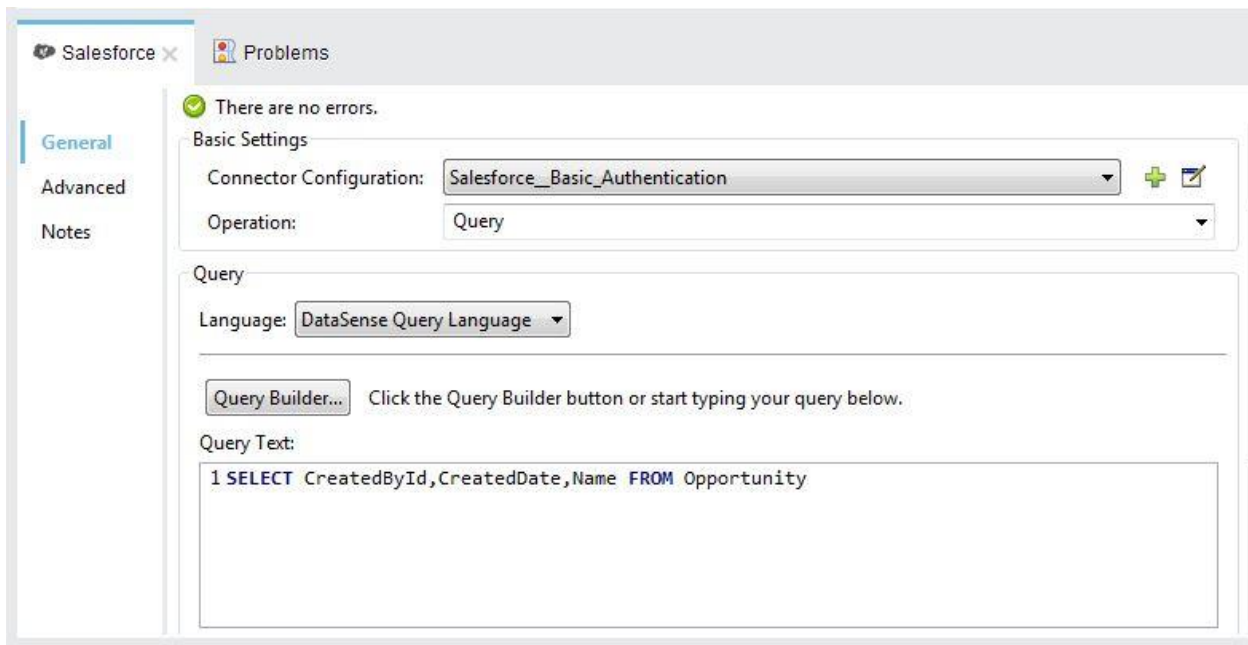
The screenshot displays the configuration for an HTTP connector. The 'General' tab is active, showing a 'Problems' section with a green checkmark and the message 'There are no errors.' Below this, the 'Display Name' is set to 'HTTP'. The 'General Settings' section shows the 'Connector Configuration' set to 'HTTP_Listener_Configuration'. The 'Basic Settings' section shows the 'Path' set to '/order' and 'Allowed Methods' set to 'GET'. The 'Response Settings' section includes fields for 'Status Code' and 'Reason', and a checkbox for 'Disable properties'. A 'Headers' section is also visible at the bottom.

iii. Salesforce Settings

Place an auto-paging-enabled connector, such as **Salesforce** inside the flow.

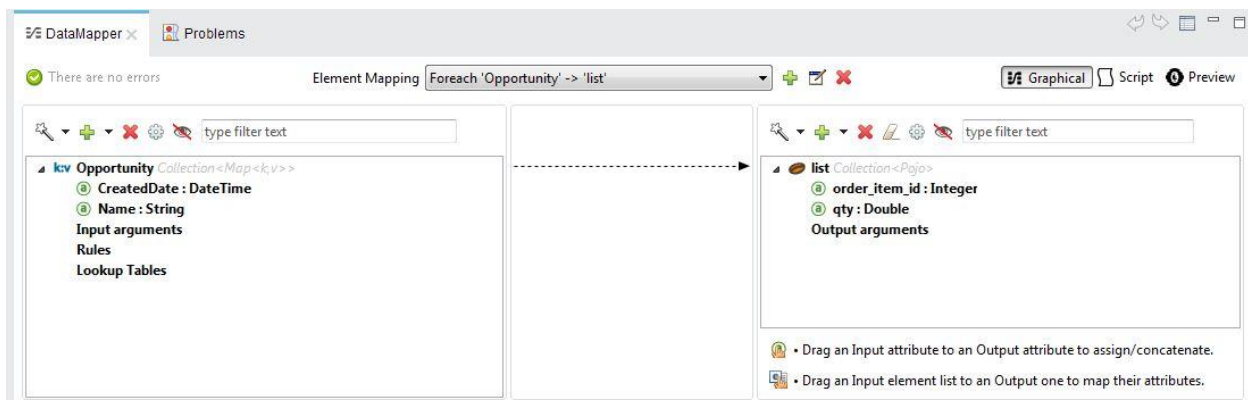
Configure the connector according to the following screenshot. Note that it queries orders. This screen sets the following values:

- Display name: Salesforce
- Config Reference: Salesforce
- Operation: Query
- Language: DataSense Query Language
- Query Text: **SELECT** CreatedById, CreatedDate, Name **FROM** Opportunity

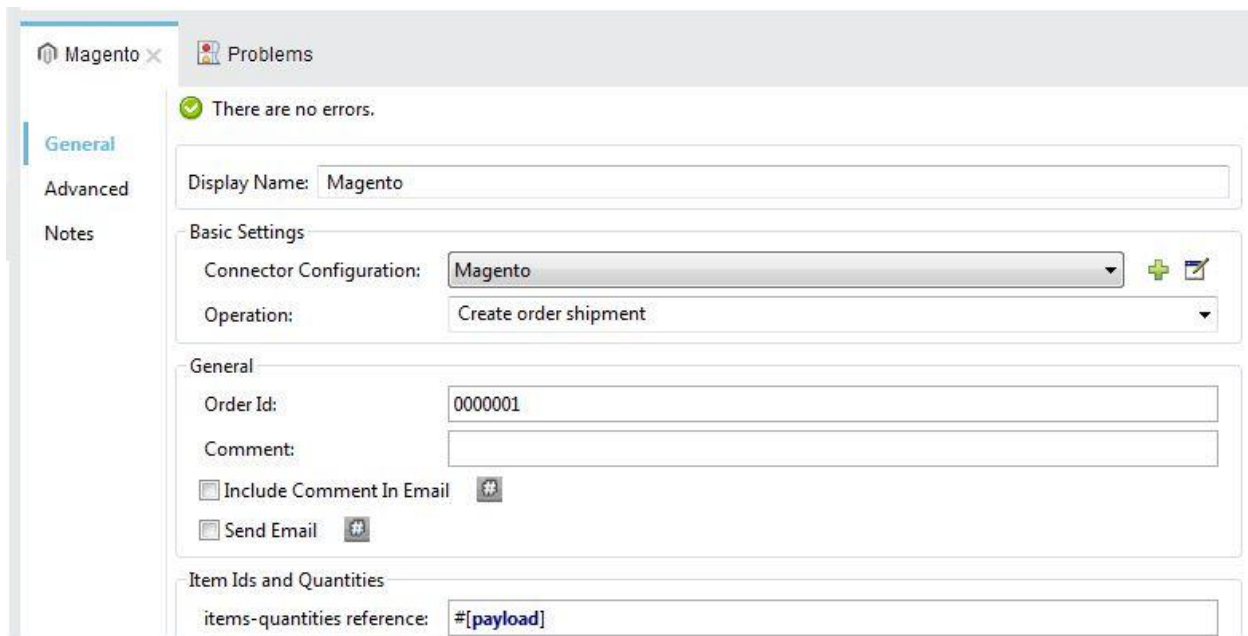


iv. DataMapper Settings

As you can see, Anypoint DataMapper has automatically created a top-level mapping called **Foreach 'Opportunity' → 'list'**, and mapped the field Name since it is identical in the input and output panes.



v. Magento Settings



Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:http="http://www.mulesoft.org/schema/mule/http" xmlns:data-
mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
    <http:listener-config name="HTTP_Listener_Configuration" host="0.0.0.0"
port="8081" doc:name="HTTP Listener Configuration"/>
    <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle259"
securityToken="dpFpnaArcjG6Kpwei9GFJCNH" doc:name="Salesforce: Basic
Authentication"/>
```

```

    <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
    <data-mapper:config name="List_Opportunity__To_List_OrderItemIdQty_"
transformationGraphPath="list_opportunity__to_list_orderitemidqty_.grf"
doc:name="List_Opportunity__To_List_OrderItemIdQty_" />
    <flow name="http-salesforce-magento-orderFlow">
        <http:listener config-ref="HTTP_Listener_Configuration" path="/order"
allowedMethods="GET" doc:name="HTTP"/>
        <sfdc:query config-ref="Salesforce__Basic_Authentication" query="dsql:SELECT
CreatedById, CreatedDate, Name FROM Opportunity" doc:name="Salesforce"/>
        <data-mapper:transform config-ref="List_Opportunity__To_List_OrderItemIdQty_"
doc:name="List<Opportunity> To List<OrderItemIdQty>"/>
        <magento:create-order-shipment config-ref="Magento" orderId="0000001"
doc:name="Magento"/>
        <logger message="#[payload]" level="INFO" doc:name="Logger"/>
    </flow>
</mule>

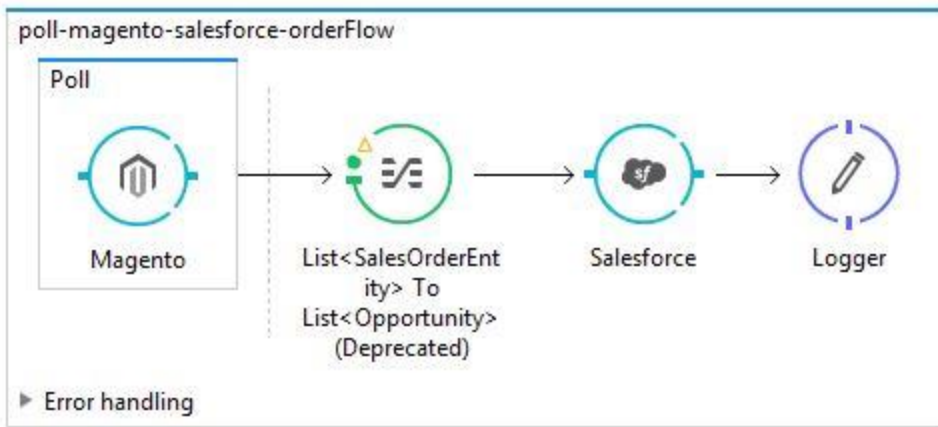
```

7. POLL-Magento-Salesforce-Order

i. Main Flow

This flow regularly polls a resource, and then performs a series of operations on the resulting payload. With every poll, the application acquires only the data that is newly created or updated since the last call to the resource. In this example, Mule stores watermarks in two variables:

- A persistent object store variable
- An exposed flow variable

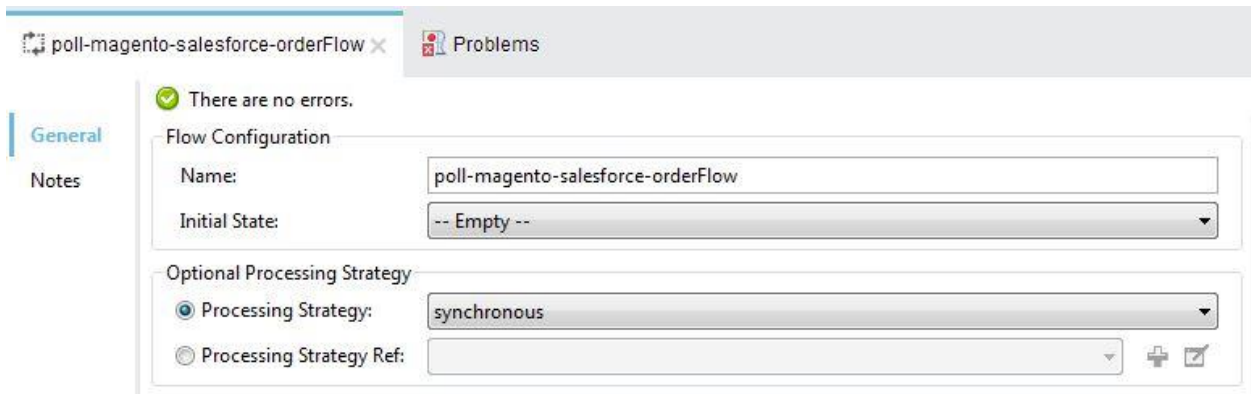


Message Flow Global Elements Configuration XML

ii. Poll Settings

Click the flow name bar to select the **flow**, and in the properties editor, set the **Processing Strategy** to **synchronous**.

*Note: All flows use an asynchronous processing strategy by default. If you do not set the processing strategy to **synchronous**, polling with watermarks does not work!*



Select the **poll** scope, and edit its properties according to the table below.

There are no errors.

Display Name: Poll

Polling Information

Fixed frequency scheduler

Frequency: 1

Start delay: 0

Time unit: HOURS

Cron scheduler

Expression:

Watermark

Do not use watermark

Enable watermark

Variable Name: lastCreationDate

Default Expression: #[TODAY]

Update Expression: #[flowVars['lastCreationDate']]

Selector:

Watermark is a tool to simplify querying for updated objects, which is a very common use case when synchronizing data.

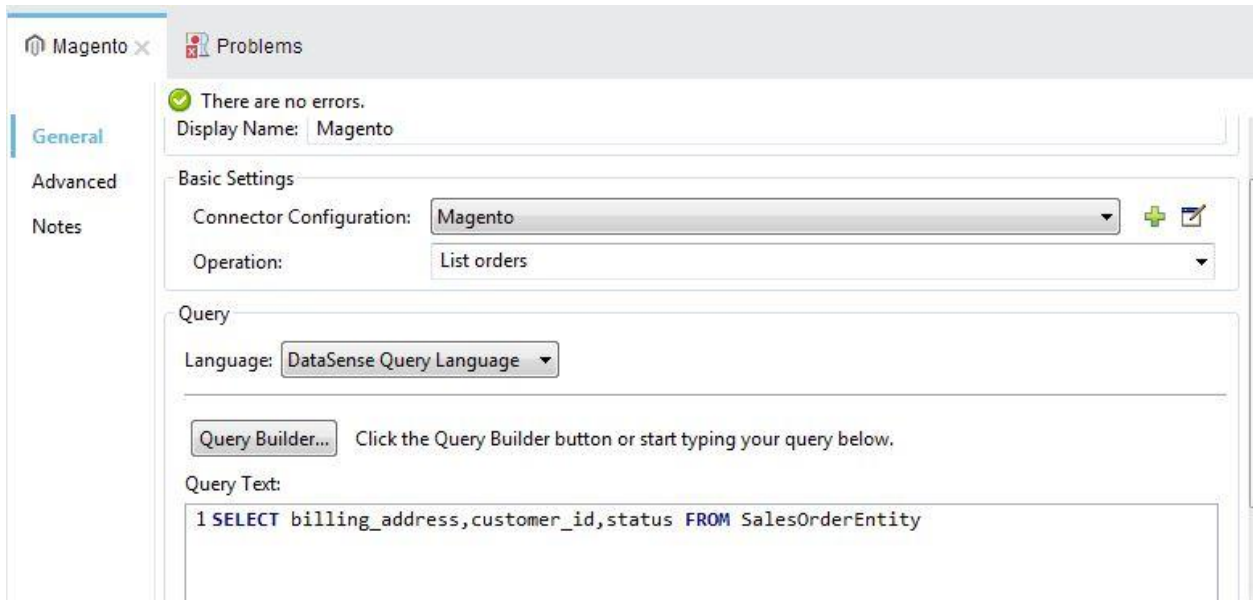
iii. Magento Settings

Click the Magento connector, configure the connector settings, and set the operation to 'list order' and use Query Builder to write your SQL so as to query the magento database.

Configure the connector according to the following screenshot. This screen sets the following values:

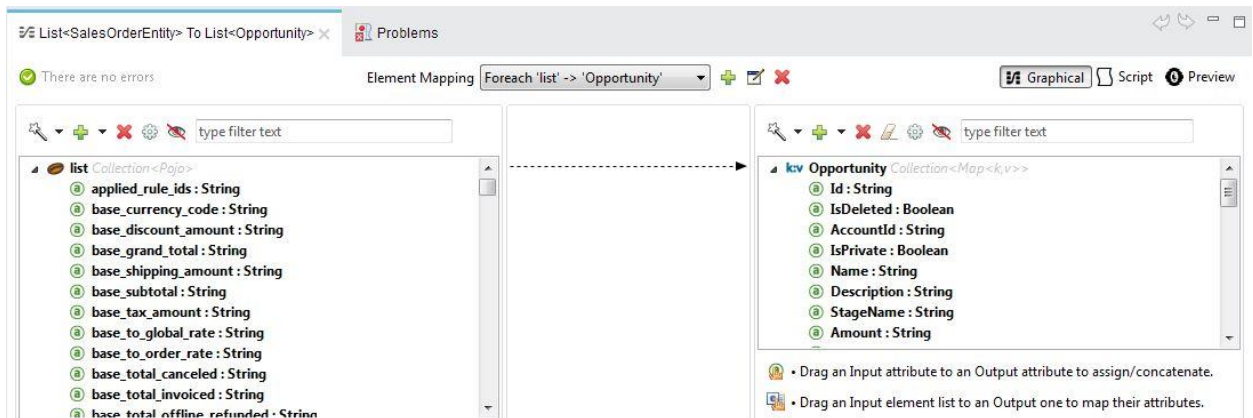
- Display name: Magento
- Config Reference: Magento
- Operation: Query
- Language: DataSense Query Language
- Query Text: (combine into one line in the connector's field)

SELECT billing_address,billing_firstname,billing_lastname,billing_name **FROM** SalesOrderEntity



iv. DataMapper Connector

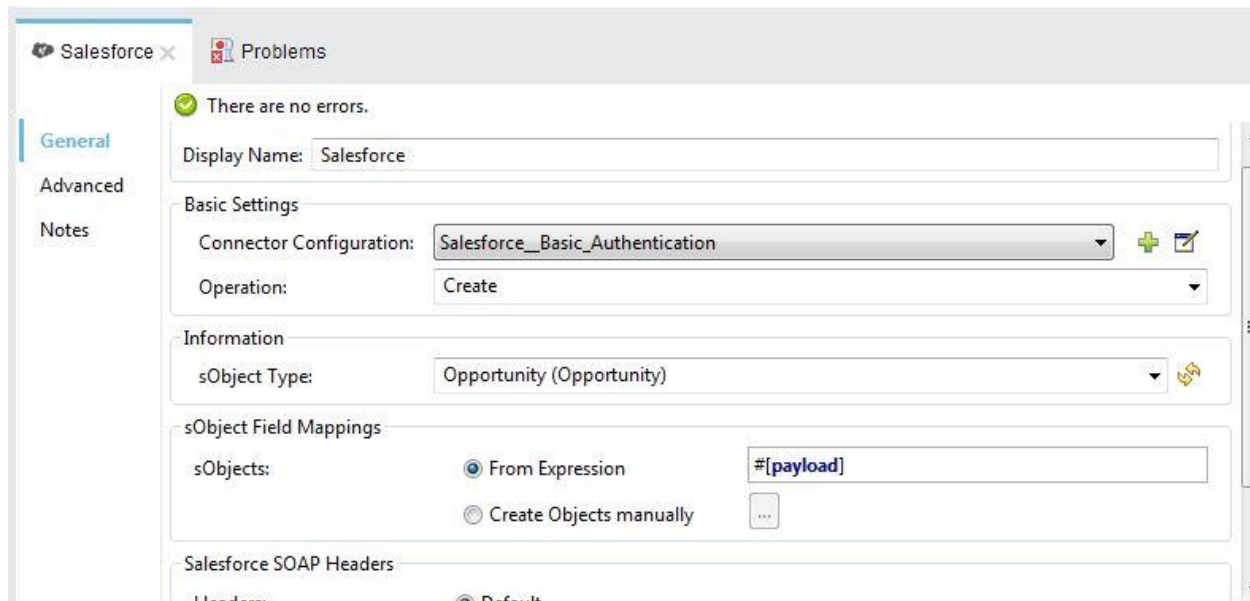
As you can see, Anypoint DataMapper has automatically created a top-level mapping called **Foreach 'list' → 'Opportunity'**, and mapped the field Name since it is identical in the input and output panes.



v. Salesforce Settings

In this salesforce connector, it is configured to create an operation from Opportunity as the object type and the field mapping will read from payload meaning, it will get it data from datamapper, the data that datamapper stored

from magento when it was queried, the results from datamapper will be added to the database of salesforce.



vi. Logger

This logger uses the MEL expression `#[payload]` to log the message payload collected by the Magento connector every 1 hour.

Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:data-mapper="http://www.mulesoft.org/schema/mule/ee/data-mapper"
xmlns:magento="http://www.mulesoft.org/schema/mule/magento"
xmlns:sfdc="http://www.mulesoft.org/schema/mule/sfdc"
xmlns:tracking="http://www.mulesoft.org/schema/mule/ee/tracking"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
    xmlns:spring="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
```

```

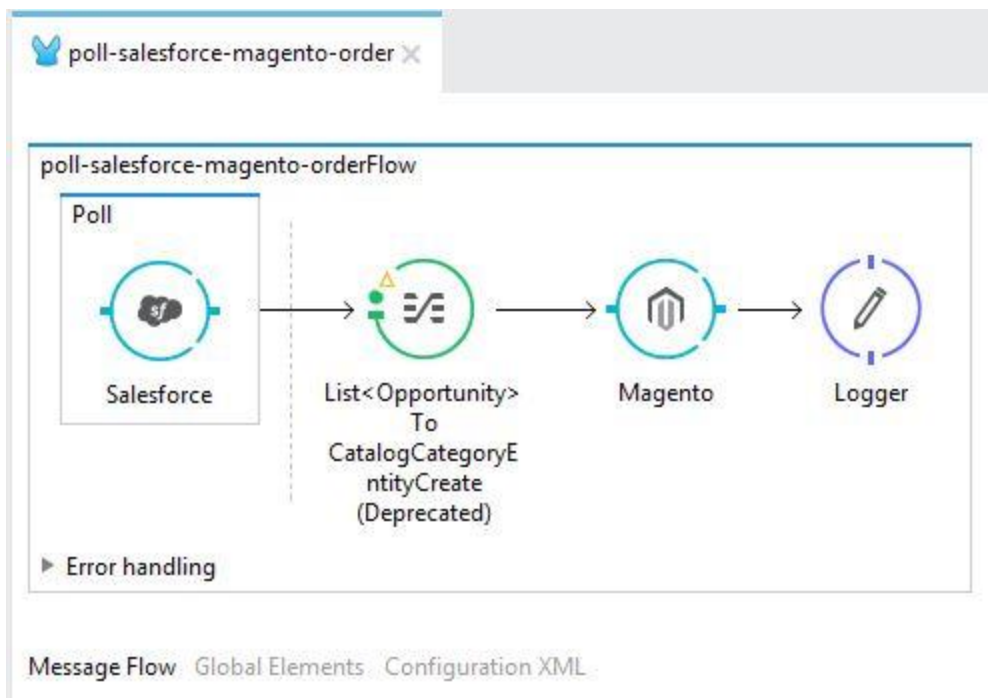
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/magento
http://www.mulesoft.org/schema/mule/magento/current/mule-magento.xsd
http://www.mulesoft.org/schema/mule/sfdc
http://www.mulesoft.org/schema/mule/sfdc/current/mule-sfdc.xsd
http://www.mulesoft.org/schema/mule/ee/tracking
http://www.mulesoft.org/schema/mule/ee/tracking/current/mule-tracking-ee.xsd
http://www.mulesoft.org/schema/mule/ee/data-mapper
http://www.mulesoft.org/schema/mule/ee/data-mapper/current/mule-data-mapper.xsd">
  <sfdc:config name="Salesforce__Basic_Authentication"
username="herbdolie25@gmail.com" password="goldeneagle259"
securityToken="dpFpnaArcjG6Kpwei9GFJCNH" doc:name="Salesforce: Basic
Authentication"/>
  <magento:config name="Magento" username="api" password="107475508th"
address="http://magento-tyumenacm.rhcloud.com/index.php/api/v2_soap"
doc:name="Magento"/>
  <data-mapper:config name="List_SalesOrderEntity__To_List_Opportunity_"
transformationGraphPath="list_salesorderentity__to_list_opportunity_.grf"
doc:name="List_SalesOrderEntity__To_List_Opportunity_"/>
  <flow name="poll-magento-salesforce-orderFlow" processingStrategy="synchronous">
    <poll doc:name="Poll">
      <fixed-frequency-scheduler frequency="1" timeUnit="HOURS"/>
      <watermark variable="LastCreationDate" default-expression="#['TODAY']"
update-expression="#[flowVars['LastCreationDate']]"/>
      <magento:list-orders config-ref="Magento" filter="dsl:SELECT
billing_address,customer_id,status FROM SalesOrderEntity" doc:name="Magento"/>
    </poll>
    <data-mapper:transform config-
ref="List_SalesOrderEntity__To_List_Opportunity_"
doc:name="List<SalesOrderEntity> To List<Opportunity>"/>
    <sfdc:create config-ref="Salesforce__Basic_Authentication" type="Opportunity"
doc:name="Salesforce">
      <sfdc:objects ref="#[payload]"/>
    </sfdc:create>
    <logger message="#[payload]" level="INFO" doc:name="Logger"/>
  </flow>
</mule>

```

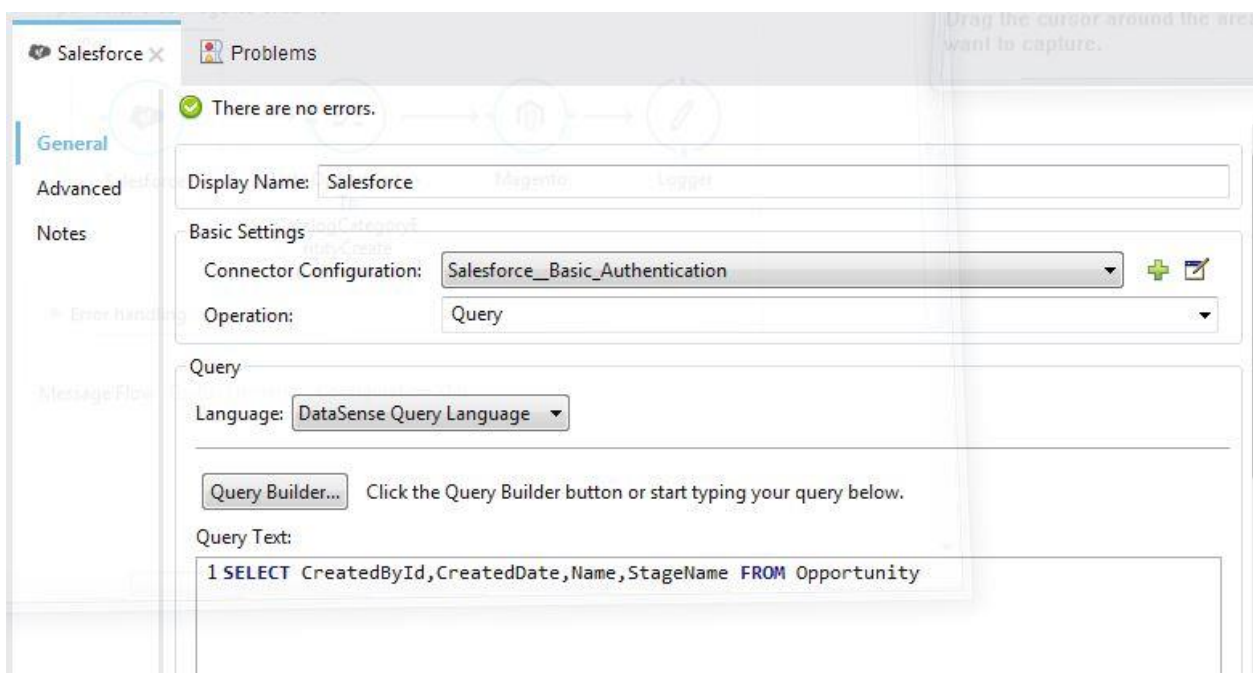
8. POLL-Salesforce-Magento-Order

i. Main Flow

The aim of this flow is to query order from magento and map it to salesforce.



ii. Salesforce Connector



Place an auto-paging-enabled connector, such as **Salesforce** inside a **poll** scope.

Configure the connector according to the following screenshot. Note that the query orders the output in ascending order of LastModifiedDate so that the last item in the list is the newest. This detail is critical. This screen sets the following values:

- Display name: Salesforce
- Config Reference: Salesforce
- Operation: Query
- Language: DataSense Query Language
- Query Text: (combine into one line in the connector's field)

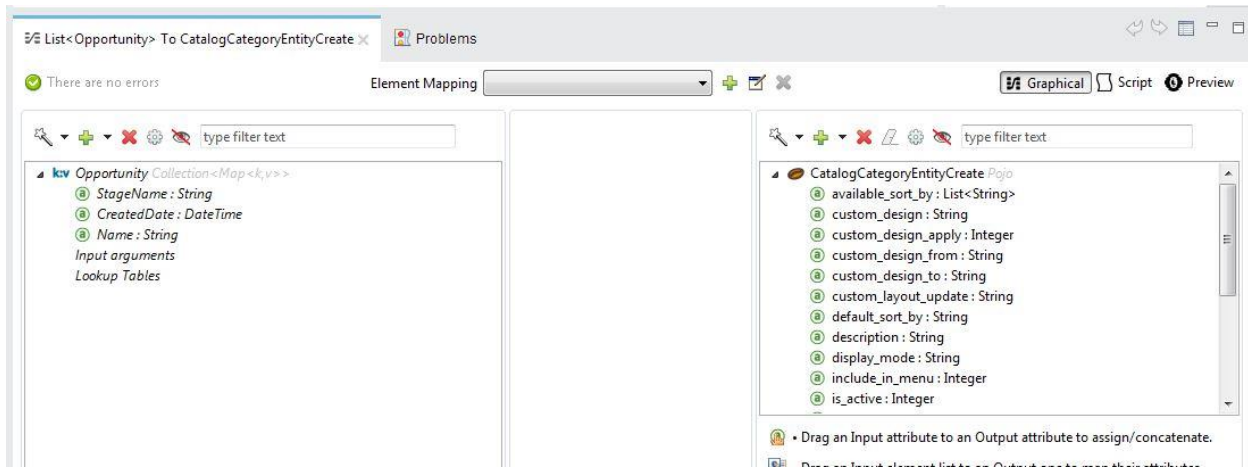
iii. Poll Settings

The screenshot shows the 'Poll Settings' configuration window in MuleSoft. The interface includes a top navigation bar with 'Poll', 'Problems', and 'Console' tabs. A status message at the top indicates 'There are no errors.' The configuration is organized into several sections:

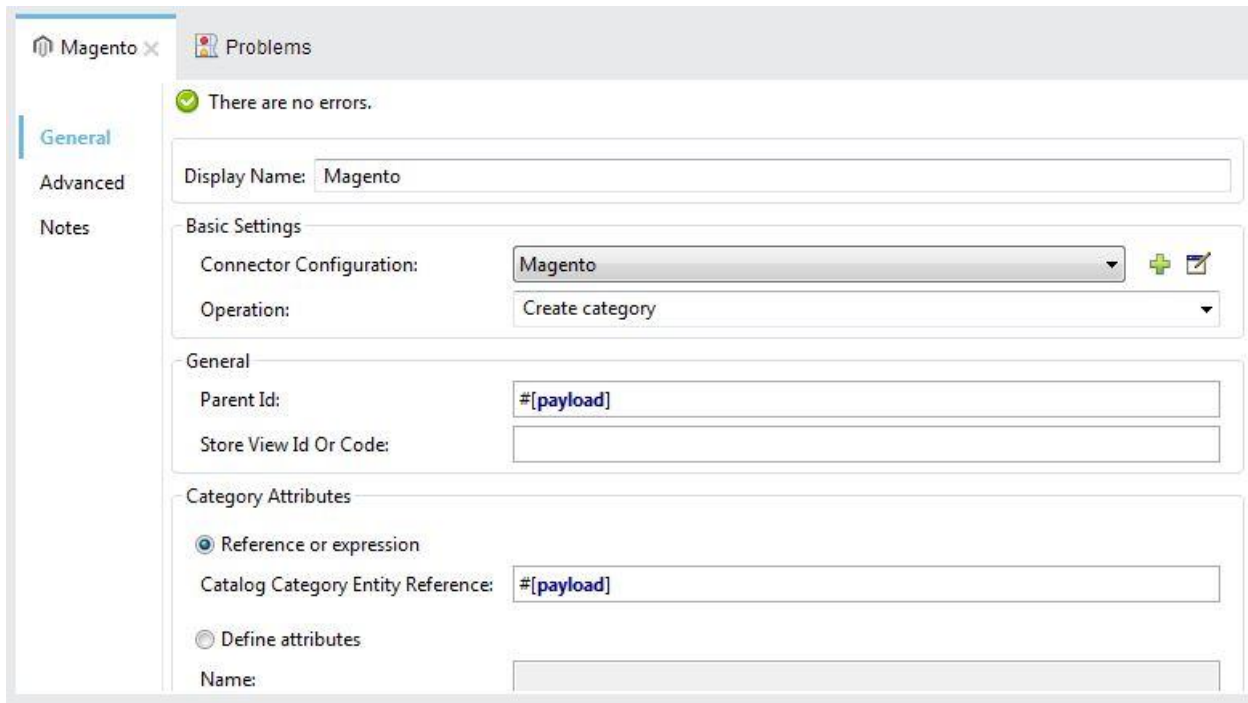
- General:** Contains a 'Display Name' field set to 'Poll'.
- Polling Information:**
 - Fixed frequency scheduler:
 - Frequency: 1
 - Start delay: 0
 - Time unit: DAYS
 - Cron scheduler:
 - Expression: (empty field)
- Watermark:**
 - Do not use watermark
 - Enable watermark:
 - Variable Name: lastCreationDate
 - Default Expression: #[TODAY]
 - Update Expression: (empty field)
 - Selector: LAST
 - Selector Expression: #[payload.CreatedDate]
 - Object Store: (empty field)

Configure the poll scope according to the table below. The watermark is a variable named lastCreationDate. When you iterate through the collection, later in your flow, Mule updates the value of the variable to the value you put in the Selector field, in this case #[payload.CreatedDate]. Its default value is the result of evaluating the following expression: [TODAY].

iv. DataMapper



v. Magento



Example Use Case Code

Paste this XML code into Anypoint Studio to experiment with the two flows described in the previous section.

List of all http urls

The first element, an HTTP listener, listens on localhost port 8081 (the default) for incoming GET requests. Hitting the listener triggers the flow. Requests to the HTTP listener must take the form:

`http://localhost:8081?<query>`

The **<query>** part of the request consists of the parameters accepted by the REST API. When the HTTP listener receives the HTTP request, the **<query>** part of the URL is recorded as a set of inbound properties. The HTTP listener passes these properties to the next element in the flow, the HTTP request connector. This outbound connector is configured to query the remote REST API at `http://baconipsum.com/api`. The HTTP request connector uses a couple of simple MEL expressions to extract the query parameters from the message it received from the listener, and to construct the full URL for the remote API, including the query parameters.

| Flow Name | Path | Method | Description |
|----------------------------------|---------------|--------|---|
| HTTP-Magento-Salesforce Customer | /sync | GET | Open the HTTP connector's properties editor and give it the path sync. Then create a Connector Configuration element for it and set its host to localhost and port to 8081. In this way, you can reach the connector via the http://localhost:8081/sync URI. |
| HTTP-Salesforce-Magento Customer | /Customer | GET | http://localhost:8081/customer |
| HTTP-Salesforce-Magento Order | /order | GET | http://localhost:8081/order |
| HTTP-Magento-Salesforce Order | /Opportunity1 | GET | http://localhost:8081/opportunity1 |

CONCLUSION

The first goal of the thesis was to explore the core of EAI (Enterprise Application Integration) and to select a set of Open Source licensed EAI products for the comparison. The EAI has been successfully explored and the ESB concept of EAI was selected as the most mature and the most powerful. Then we were looking for available Open Source licensed ESB products.

The second goal of the thesis was to find a set of criteria for comparison of the selected product (salesforce and Magento). The criteria were selecting that they were taking into account the possibility of using the ESBs in the enterprise environment.

The last and the main goal of the thesis were to perform the comparison and integration of the products according to selected criteria and to evaluate the results.

REFERENCES

1. <http://www.citeck.ru/o-kompanii/platformy-i-tehnologii/mule-esb/>
2. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.2872&rep=rep1&type=pdf>
3. https://developer.salesforce.com/docs/atlas.en.us.api_async.meta/api_async/asynch_api_reference_schema.htm
4. http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/atc_comms_services/swim/documentation/media/briefings/Service_Registry_Brown_Bag_March2011_v0.5_030911.ppt
5. <https://www.progress.com/connectors/salesforce>
6. [https://www.progress.com/blogs/export-salesforce-entity-relationship-\(er\)-diagrams-to-visio](https://www.progress.com/blogs/export-salesforce-entity-relationship-(er)-diagrams-to-visio)
7. "[magento/magento2-community-edition](#)". *GitHub*. Retrieved 2016-03-14.
8. "[Magento Community Edition 1.9.2.4 Release Notes](#)". *merch.docs.magento.com*. Retrieved 2016-03-14.
9. "[Magento Community Edition 2.0.4 Release Notes](#)". *merch.docs.magento.com*. Retrieved 2016-03-14.
10. <https://docs.mulesoft.com/mule-user-guide/v/3.7/mule-application-architecture>
11. <https://docs.mulesoft.com/mule-user-guide/v/3.7/mule-concepts>
12. <https://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb>

List of Abbreviations

The following list contains abbreviations used in this document.

| | |
|------------------|--|
| API | Application Programming Interface |
| EAI | Enterprise Application Integration |
| ESB | Enterprise Service Bus |
| JAR | Java Archive |
| Java EE 5 | Java Platform, Enterprise Edition v. 5 |
| JB1 | Java Business Integration |
| HTTP | The Hypertext Transfer Protocol |
| JDBC | Java Database Connectivity |
| JMX | Java Management Extensions |
| SOA | Service-oriented Architecture |
| SOAP | Simple Object Access Protocol (deprecated) |
| WSDL | Web Services Description Language |
| XML | eXtensible Markup Language |
| GUI | Graphic User Interface |
| API | Application Program Interface |
| UI | User Interface |
| EIP | Enterprise Integration Patterns |
| CRM | Customer Relationship Management |
| SFDC | SalesForceDotCom |
| JSON | JavaScript Object Notation |
| SQL | Structured Query Language |