

***Public Transport System Management***

**IS 301 Enterprise Integration**

***G2-3***

**Assignment**

***EUGENE CHOY WEN JIA***

***HO MIN KIT WINSTON***

***HO WEI HONG***

***SIM LI JIN***

***YIN YUKUN***

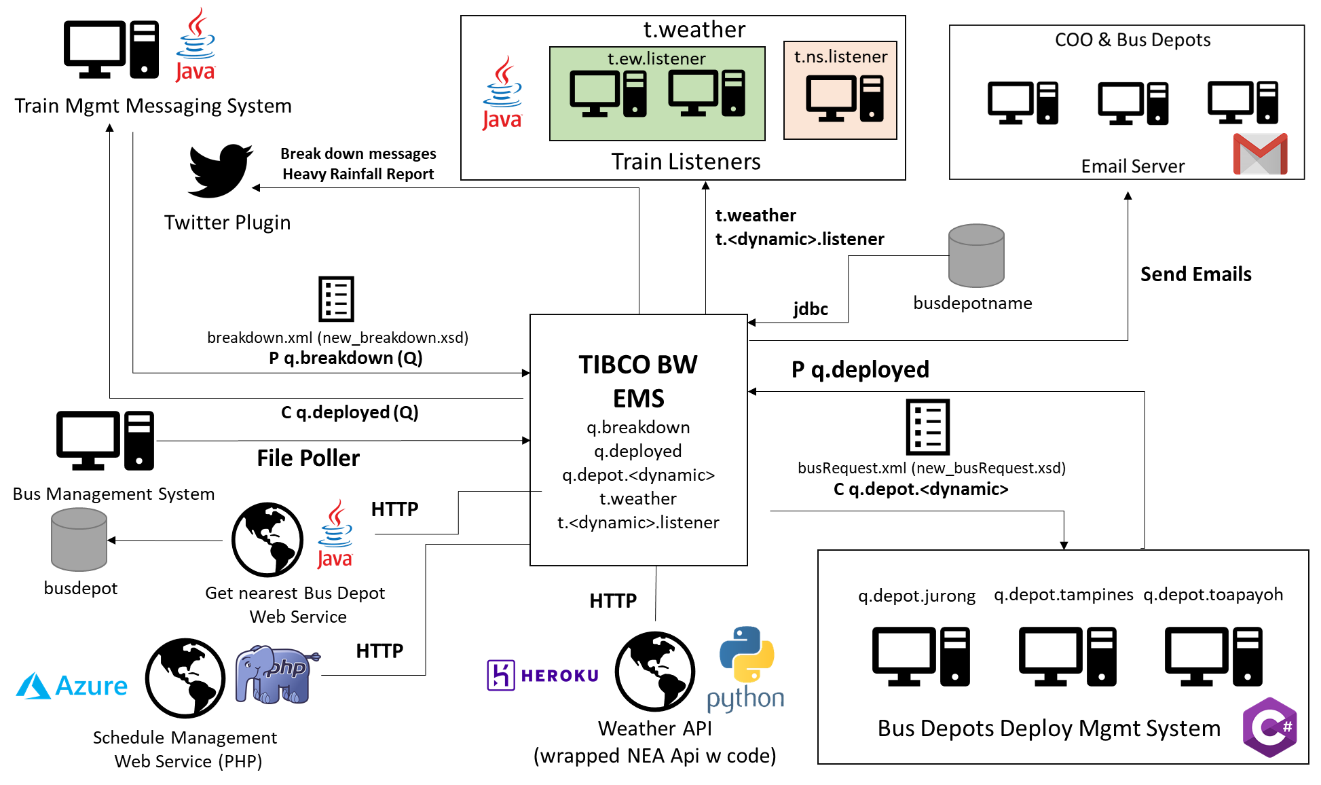
***YONG FU XIANG***

# Introduction

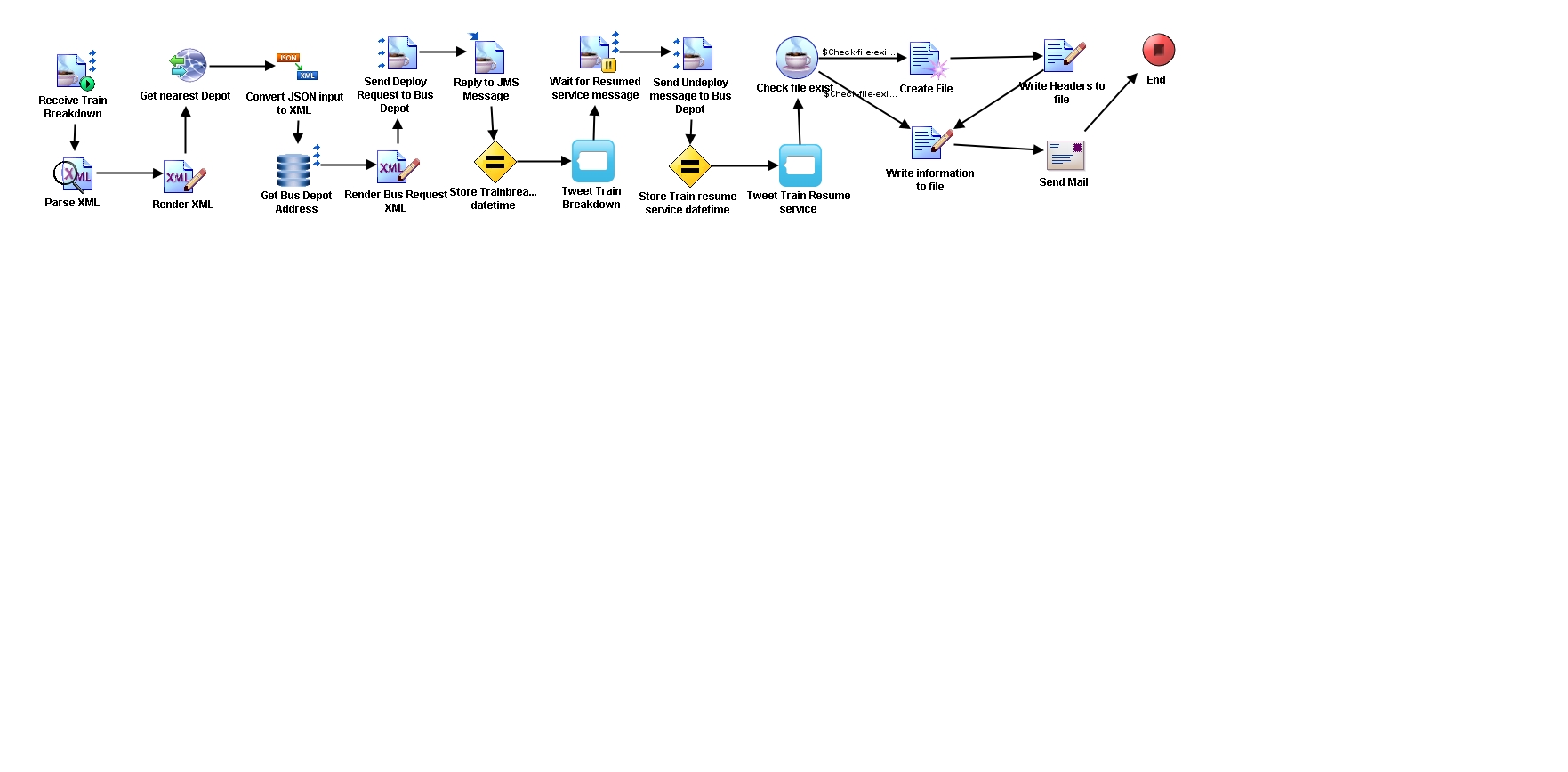
The Public Transport network in Singapore serves hundreds of thousand passengers every day. Any disruption in the service could be disastrous and mitigation actions must be undertaken to ensure minimal disruptions to the passengers. Integrations of various system in the Public Transport Network is therefore important to help enhance communication and automate processes if mitigation actions are required. On top of having to mitigate disruptions, there are also other processes such as checking the weather to determine the speed of travel and the bus schedules of bus drives which requires several steps before the message can be transferred from one end to the other. Using integration tools, we can automate a huge bulk of this process reducing the time needed for human intervention.

# Business Scenario

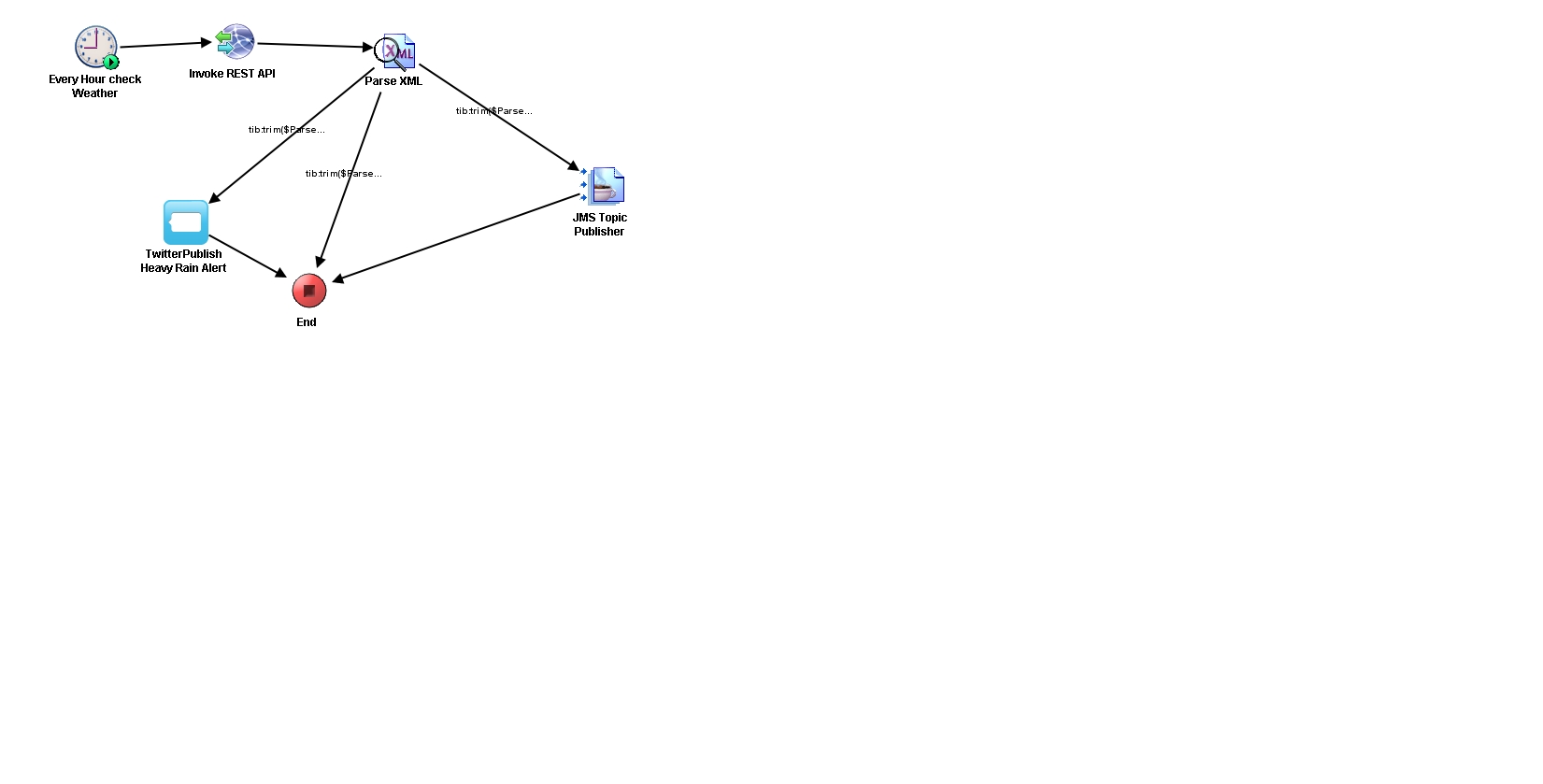
## Technical Overview Diagram

**

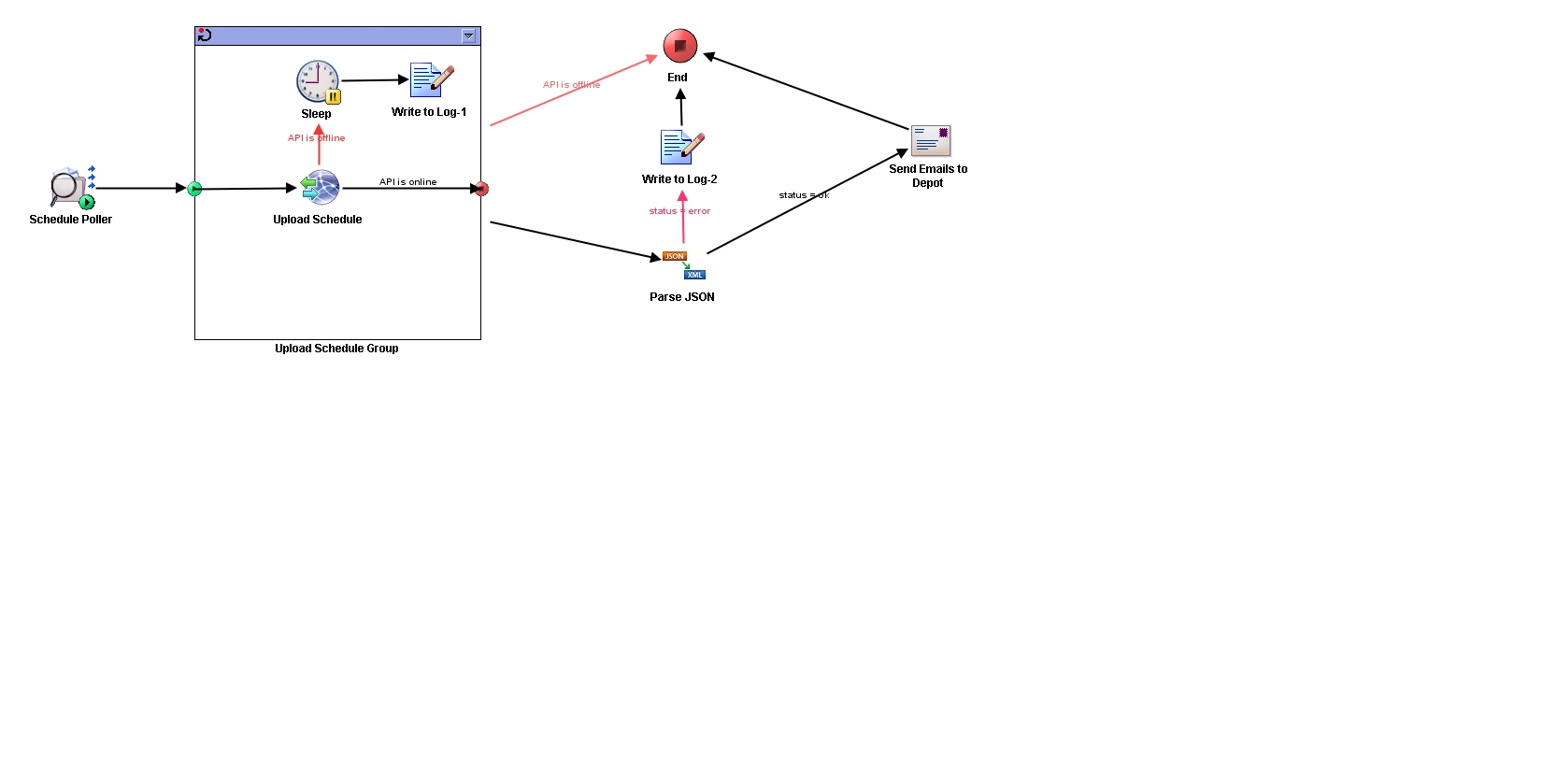
## Train Breakdown and Resumption of Service Process



## Weather Reporting Process



## Schedule Polling Process

**

# JMS Interactions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| JMS Integration | From | To | \*FF/  SRR/  ARR | Publish-Subscribe or Point-to-Point | JMS Queue/Topic (or Topic with durable subscription) | Queue/Topic Name used (case-sensitive) |
| Integration 1 | Train Management System | Tibco IM | SRR | Point-to-Point | JMS Queue | q.breakdown |
| Tibco IM | Train Management System | Point-to-Point | JMS Queue | q.deployed |
| Integration 2 | Tibco IM | Bus Depot <Dynamic> | FF | Point-to-Point | JMS Queue | q.depot.<dynamic> |
| Integration 3 | Train Management System | Tibco IM | FF | Point-to-Point | JMS Queue | q.resumed |
| Tibco IM | Bus Depot <Dynamic> | Point-to-Point | JMS Queue | q.depot.<dynamic> |
| Integration 4 | Tibco IM | Train Listener | FF | Publish-Subscribe | Topic | t.weather |

**Legend** FF - Fire & Forget SRR - Sync Request/Reply ARR - Async Request/Reply

# Web Services

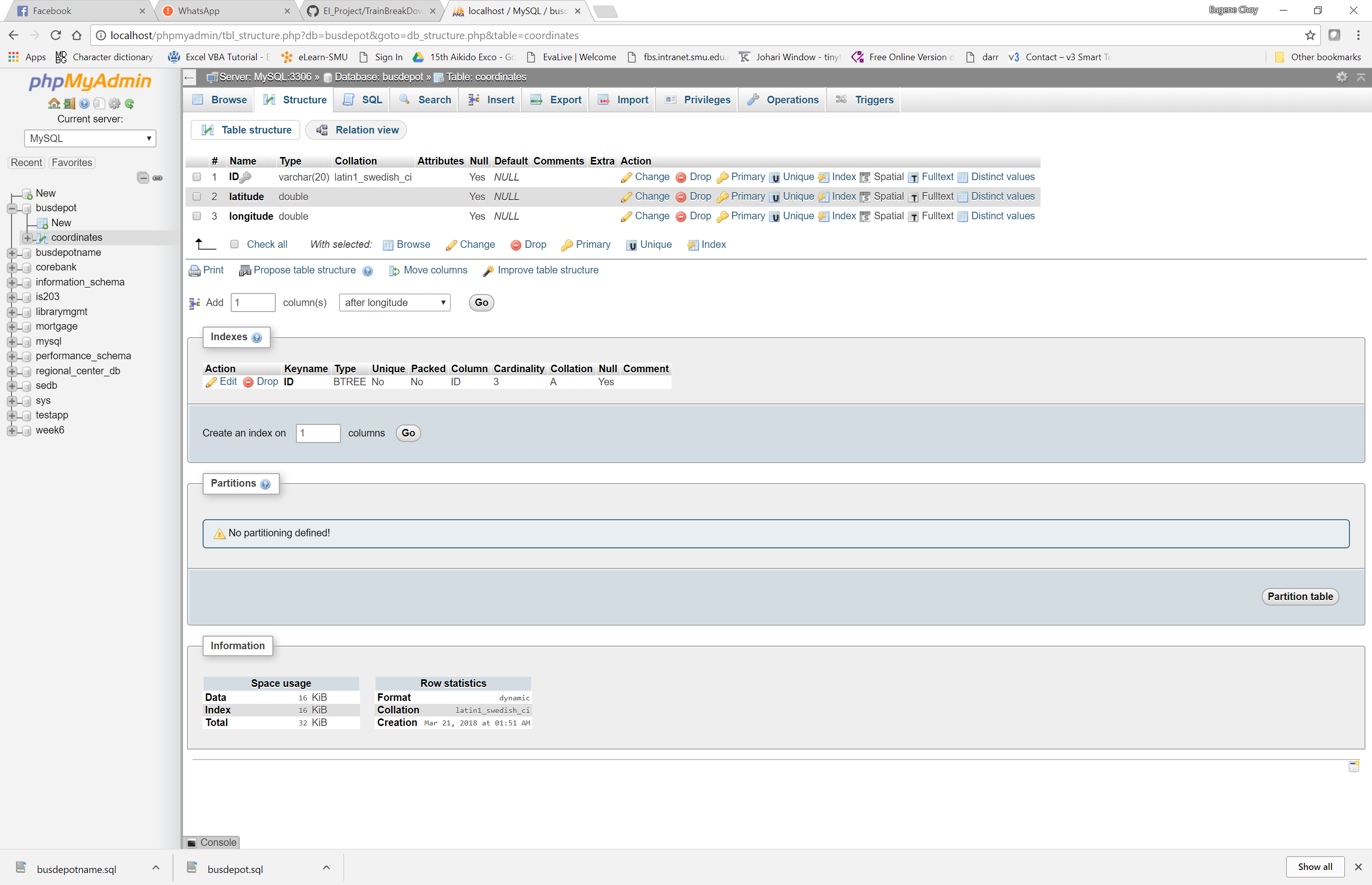
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Service | Description | Protocol | Input | Output |
| *Get Nearest Bus Depot Web Service (Java)* | *A transformed XML created from the breakdown.xml is sent into the web service as XML String to retrieve the depot ID.* ***Self-Coded in Java*** | REST POST | *XML*  *Train id*  *Fault Type*  *Time stamp*  *Coordinates* | *Depot ID* |
| *Schedule Polling* | *Upload a txt file onto the Schedule Poller service which can be downloaded by other users.* ***Self-Coded in PHP*** | REST POST | *Text File*  *Multi Part* | *JSON*  *URL of the file*  *Status Code* |
| *Weather API* | *Check the rainfall in Singapore from NEA web service.* ***NEA API + Self-Coded wrapper in Python*** *(Self-coded wrapper randomize the chance of getting a rain fall if it is not having a rainfall in Singapore now – For Demo purpose)* | REST GET | *-* | *XML*  Heavy Rain Warning |

# Design/Schema and Content of Data

## Database

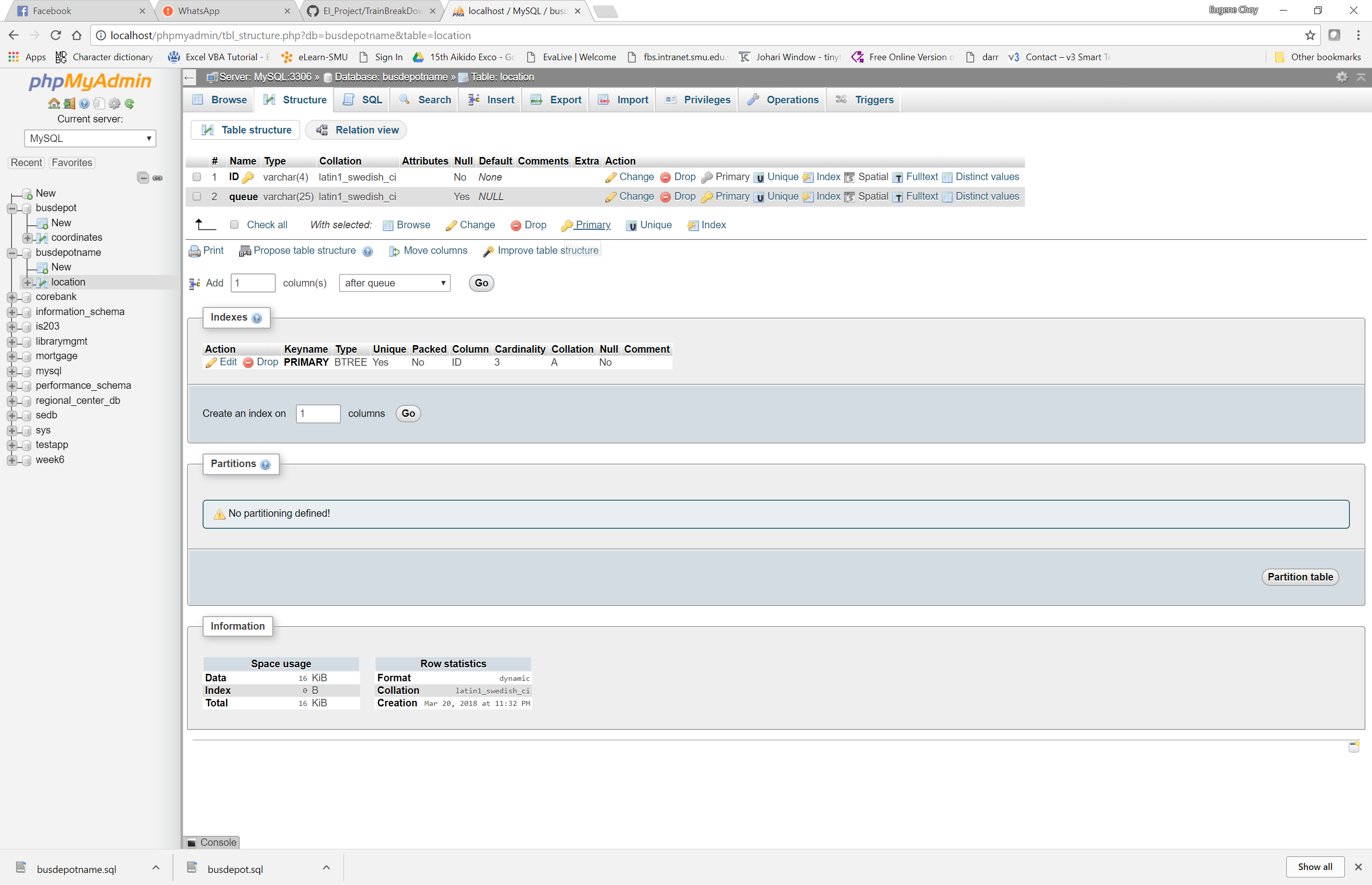
### Bus Depot

**coordinate table**



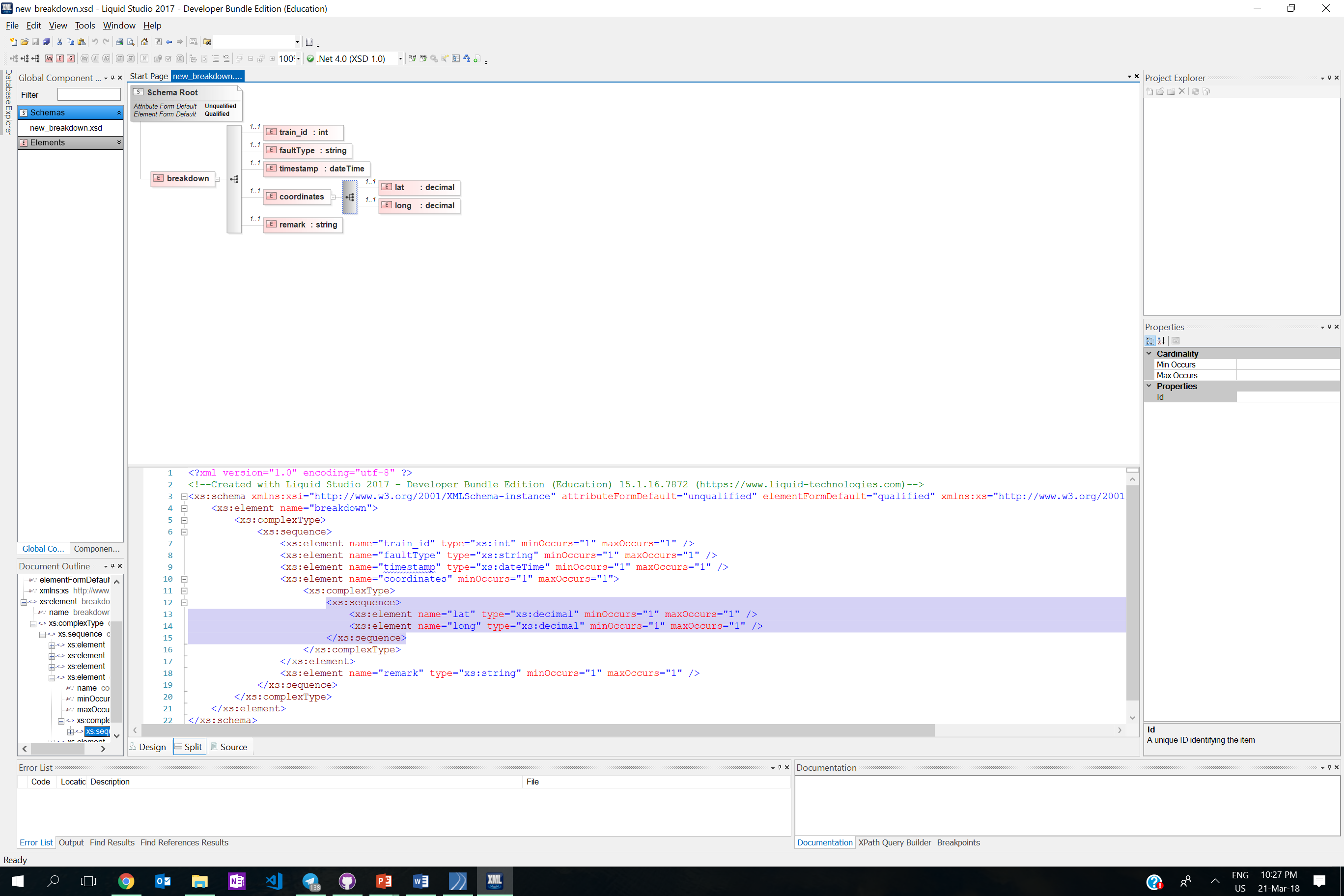
### Bus Depot Name

**location table**



## XSD

### new\_breakdown.xsd



*Sample:*

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

<breakdown xmlns:xsi = **"http://www.w3.org/2001/XMLSchema-instance"** xsi:noNamespaceSchemaLocation = **"..\\new\_breakdown.xsd"**>

<train\_id>**3**</train\_id>

<line>**ew**</line>

<faultType>**Train Fault**</faultType>

<timestamp>**2018-02-20T09:00:00**</timestamp>

<coordinates>

<lat>**1.3272383**</lat>

<long>**103.9443528**</long>

</coordinates>

<remark>**Tanah Merah Station**</remark>

</breakdown>

### new\_breakdown\_formatted.xsd



Sample:

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

<breakdown>

<train\_id>**3**</train\_id>

<faultType>**Train Fault**</faultType>

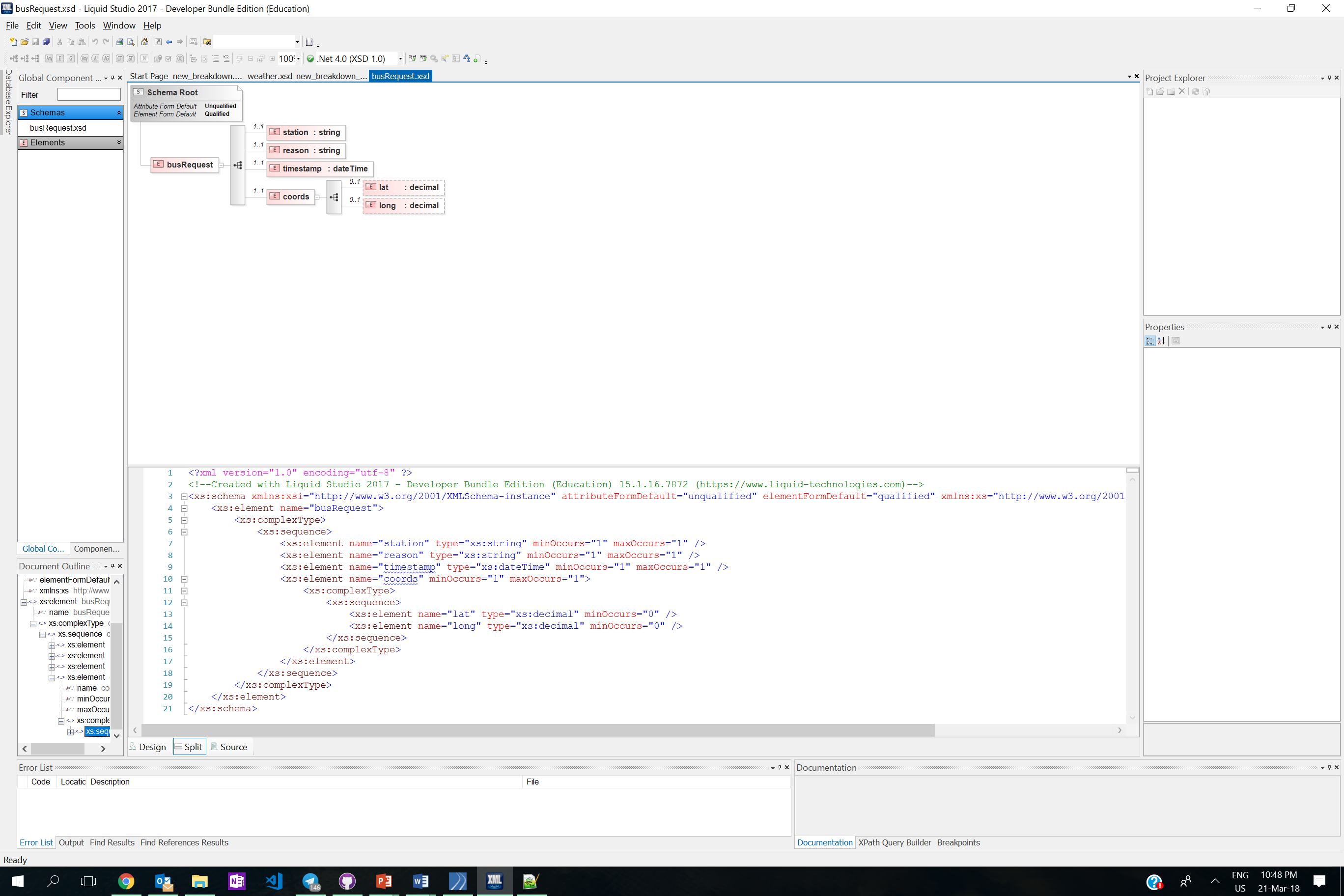
<timestamp>**2018-02-20T09:00:00**</timestamp>

<coordinates>**1.3272383,103.9443528**</coordinates>

<remark>**Tanah Merah Station**</remark>

</breakdown>

### busRequest.xsd



Sample:

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

<busRequest>

<station>**Tanah Merah Station**</station>

<reason>**Train Fault**</reason>

<timestamp>**2018-02-20T09:00:00**</timestamp>

<coords>

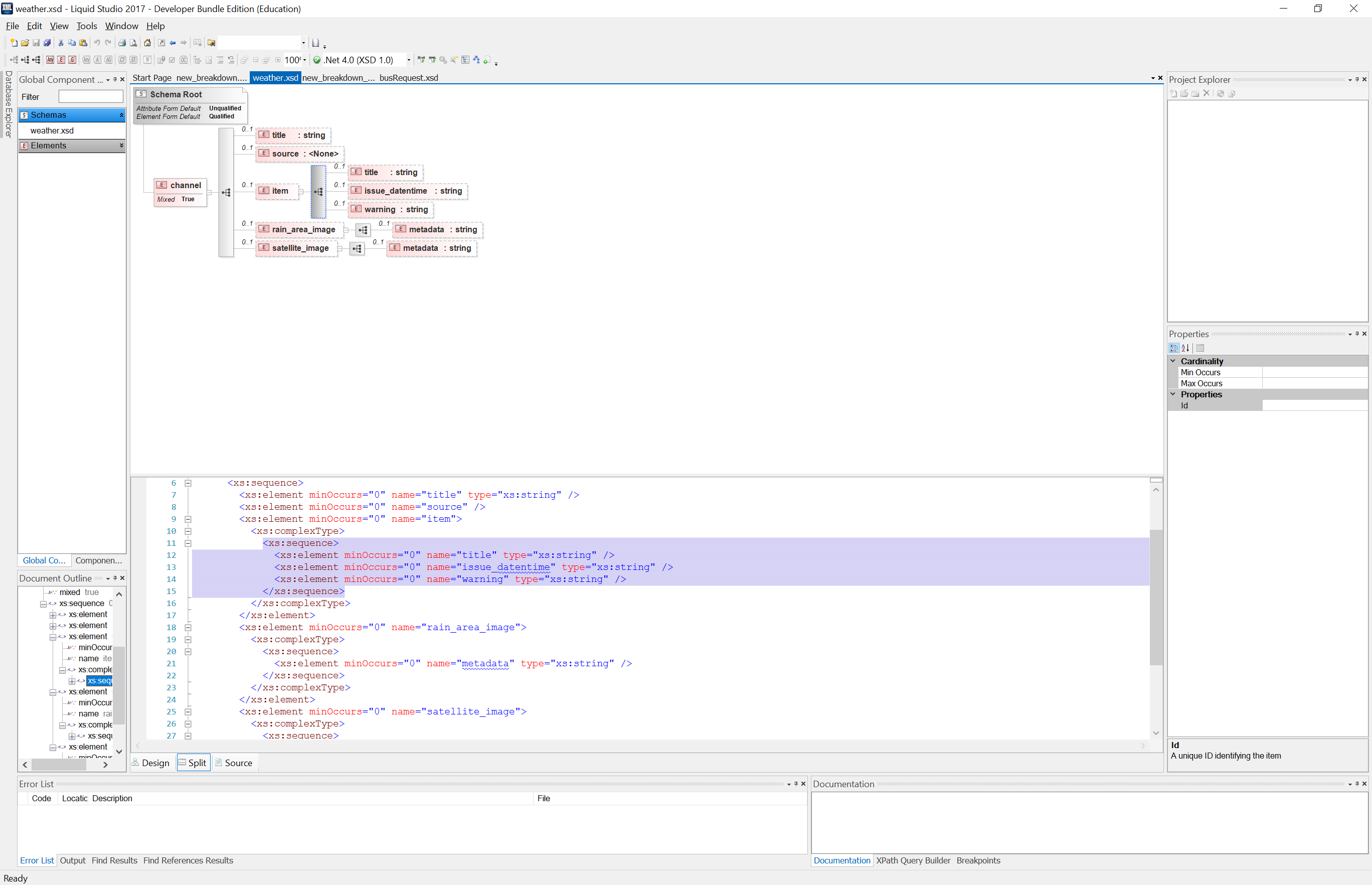
<lat>**1.3272383**</lat>

<long>**103.9443528**</long>

</coords>

</busRequest>

### weather.xsd



Sample:

<channel>

<title>**Heavy Rain Warning**</title>

<source>**Meteorological Service Singapore** </source>

<item>

<title>**HEAVY RAIN WARNING**</title>

<issue\_datentime>**-**</issue\_datentime>

<warning>**The heavy rain has eased. Thundery showers clearing in the evening.**</warning>

</item>

<rain\_area\_image><metadata>**null**</metadata></rain\_area\_image>

<satellite\_image><metadata>**null**</metadata></satellite\_image>

</channel>

## JSON

### JSON from Schedule Polling

This JSON is returned upon successful upload of a file to the server

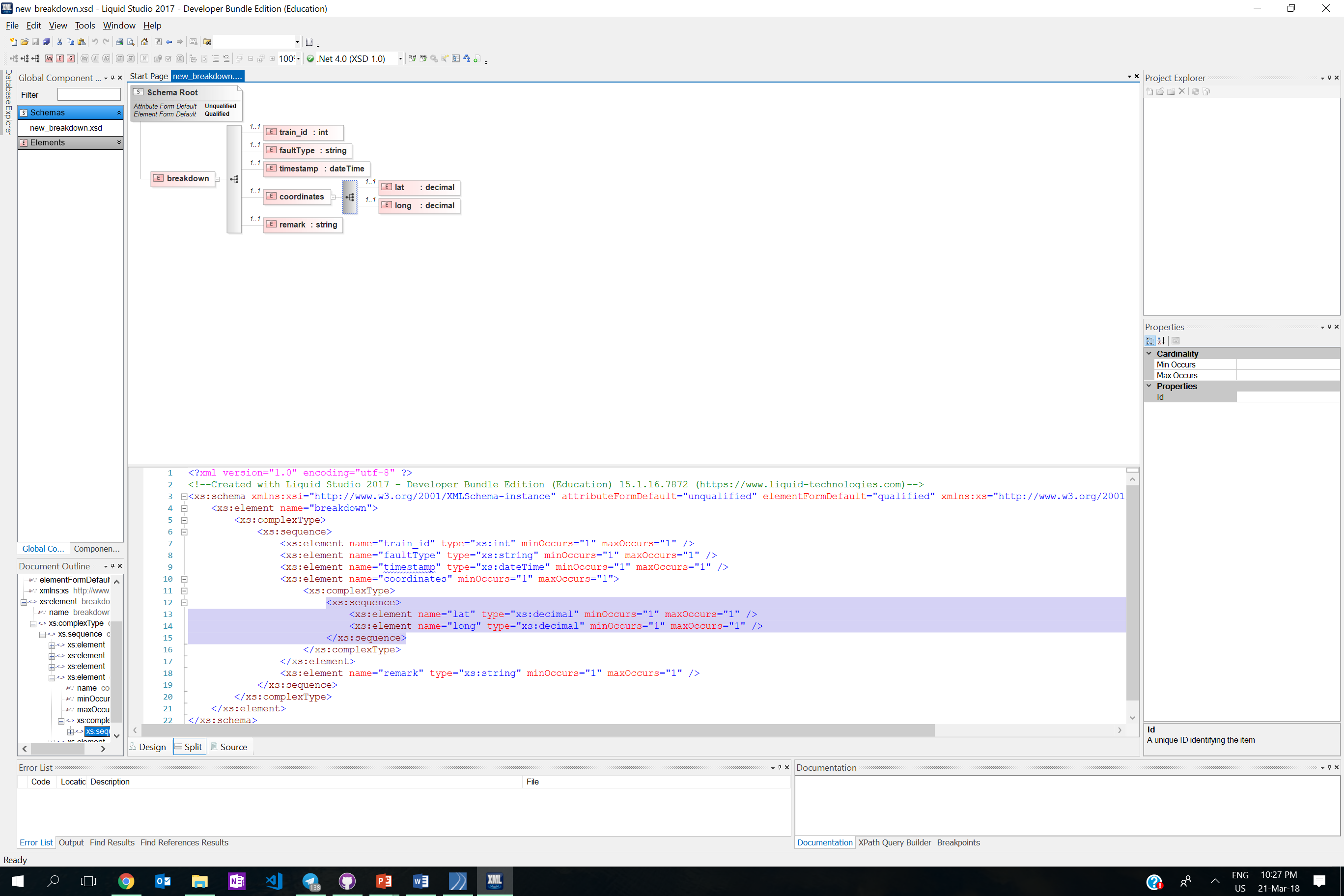
{"status":"ok",   
"statusCode":"200",   
"link":”https:\/\/eieio.blob.core.windows.net\/schedule\/January2018Schedule.txt" }

### JSON from getting nearest Bus Depot

{"message":"TMPN","status":"success"}

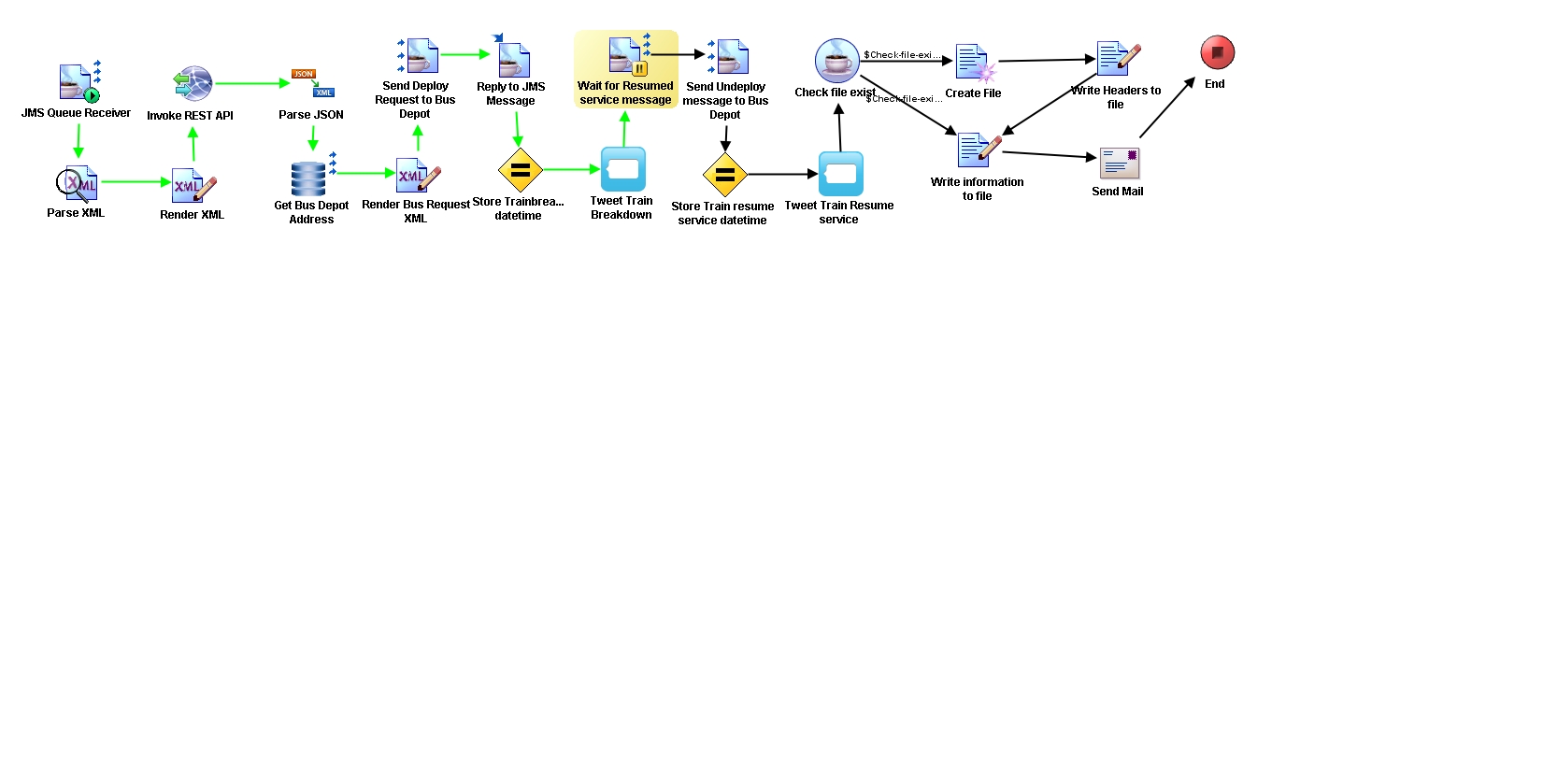
# Data Transformation

***Between Train Management System(new\_breakdown.xsd) and Get nearest Bus Depot web service (new\_breakdown\_formatted.xsd)***



The XML from new\_breakdown\_formatted expects the coordinates to be displayed in lat,long format in the coordinates tag instead of having a lat and long tag by itself. Therefore, the lat and long from new\_breakdown has to be concatenated with a comma to form the coordinates tag.

# Content-based Routing

Upon receiving the break down message, the REST API (nearest bus depot) webservice will be invoked. The Depot ID will be returned to the IM. The IM will then query the database to get the destination name for the nearest Depot.

The depots are all MsgConsumers on a queue with their name in this format q.depot.<dynamic> . With the queue name that is queried from the database, the deploy request will be sent to the relevant depot.

# Beyond the Labs

## Waiting for JMS Message

In the Train Breakdown and Resumption of Service process, upon deploying the bus as a mitigation action, the process will wait for a second message (With for JMS Message) from TMS upon successful resumption of service.

## Assigning a Variable

The Assign action is used to store a variable in TIBCO BW. The datetime of breakdown and datetime of resumption of service is stored in a variable for further processing and usage in the later part of the process.

## Twitter Plugin

We used Twitter plugin in Tibco BW for both the Train Breakdown and Resumption of Process as well as the Weather Reporting Process. The plugin is configured with the Oauth 1.0 for the usage of twitter. Some java properties have to be added in the designer.tra as well as the bwengine.tra to ensure that Twitter can be used. A tweet is sent when: service is down, service has resumed, heavy rainfall.

## Java Code

The Java Code action in Tibco BW is used in the Train breakdown and Resumption of Service process to check if the log file exist in the directory. If the file does not exist, it will return a false to prompt the IM to create a file.

## Send Mail

The Send Mail action is used to send email to the various stakeholders for the breakdown process and Schedule Polling process. Emails are sent out to the COO upon the end of a incident, to keep the COO updated the downtime of the incident. For schedule polling process, email is sent out to notify the bus depot of the URL to download their bus schedule for the month.

## Timer

We used the Timer function to run processes which require to be repeated in a short amount of time. For the weather reporting process, the timer function is used to restart the process based on the interval time specified. There is a need for constant monitoring of the weather therefore the process is put on a timer.

## File Polling

File polling in the schedule polling process listens to a certain directory for any new file added into the directory. The process will then kickstart a process to upload that new file into the server and generate a URL which allows other depot to download the file.

## Grouping Repeat-on-Error-Till-True & Sleep

If the webservice that uploads the file is down, we will sleep the process for about 10 seconds before re-attempting to upload it again. This process will repeat for a set amount of times before it ends.

## Hosting on Cloud

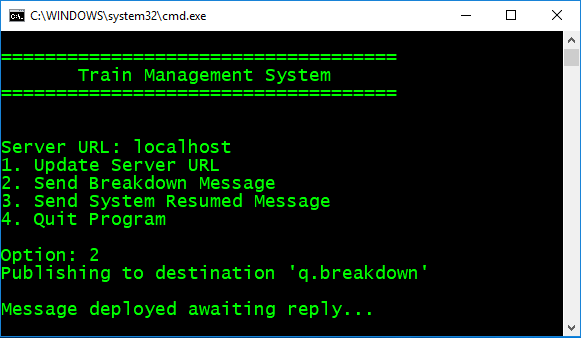
The wrapped version of NEA API is hosted on Heroku while the schedule polling website is hosted on Azure.

## Multi-Language Integration

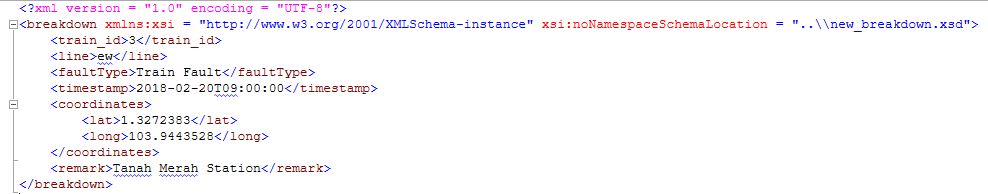
Message Producers are written in java while Message Consumers were written in Java and C#. The webservices were also write in various language such as: Java, PHP and Python to simulate integration of systems in different language.

# Scenario Walkthrough (Max 10 Pages)

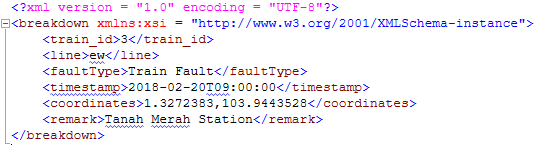
The scenario starts off when a train breaks down in one of the stations. The train would send a breakdown message to the Train Management System. (In this case, we are simulating it with the use of a command prompt to illustrate that the breakdown message is sent.)



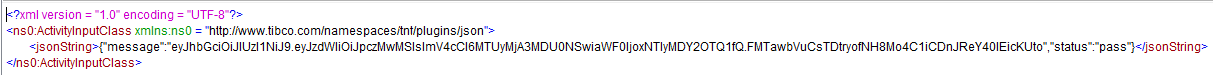
Sample of the breakdown message that is being sent to the Train Management System.

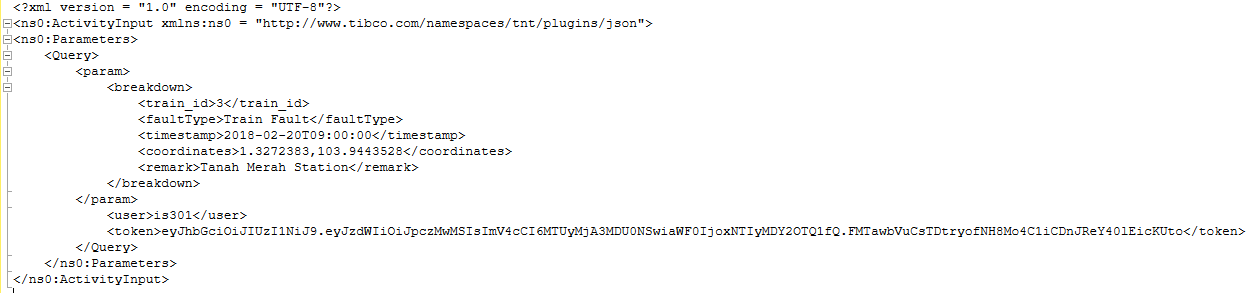


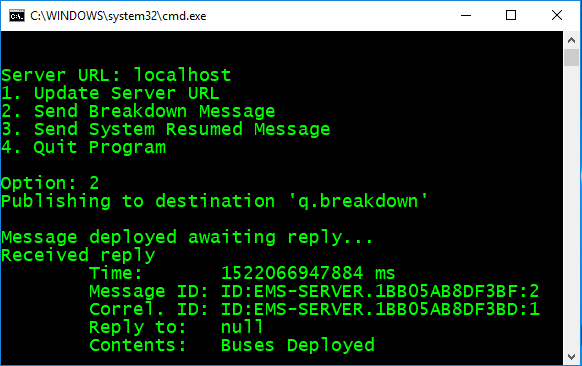
Before sending it to the Bus Depot Deploy Management System, there is a data transformation done that combines the lat and long into coordinates.



When it is communicating with the Bus Depot Deploy Management System, it will obtain a JWT token to invoke the REST API to the web service to get the nearest Depot







*[Walkthrough of your demonstration, using screen captures. Screens captured must be viewable when the document is displayed at 100% zoom scale]*